University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

Workflow Management Module for Dental Information System (DentISt) 3.0

A special problem in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Jamie D. Gerona

April 2013

ACCEPTANCE SHEET

The Special Problem entitled "Workflow Management Module for Dental Information System (DentISt) 3.0" prepared and submitted by Jamie D. Gerona in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*candidate*)		
2. Avegail D. Carpio, M.Sc.		
3. Aldrich Colin K. Co, M.Sc. (*candidate*)		
4. Perlita E. Gasmen, M.Sc. (*candidate*)		
5. Ma. Sheila A. Magboo, M.Sc.		
6. Vincent Peter C. Magboo, M.D., M.Sc.		
7. Geoffrey A. Solano, Ph.D.(*candidate*)		
8. Bernie B. Terrado, M.Sc. (*candidate*)		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Avegail D. Carpio, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

Previous dental information systems of UPCD, namely Open DentISt and DentISt2.0, allow the clinicians to store and access patient dental records electronically. Both systems however only implemented the Oral Diagnosis section of UPCD. It is inconvenient and impractical to reprogram the entire system to add the 3 other sections or if there are changes in UPCD in the future. DentISt 3.0 integrated a workflow management system to manage the flow of tasks of clinicians during patient treatment and to allow the system to adapt to changes in UPCD without the need to reprogram the entire system.

In DentISt 3.0, the system workflow can be modified and the changes made are automatically adapted by the system. Forms can also be added dynamically using the web-based form editor. The system allows clinicians to perform tasks based on their roles. All clinicians can add and edit section-specific patient records, set appointment with patients and refer patients to other UPCD sections. Faculty clinicians can also approve patient record updates.

Keywords: Dental Information System, Workflow Management System, jBPM

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

During the last forty years, use of computer technology in the dental field has developed into a research discipline known as dental informatics. It is the application of computer and information science to improve dental practice, research, education and management [2]. Dental information systems, a major advancement in dental informatics, help in information gathering and efficient management of clinical and dental records.

One of the many dental communities that realized the significance of having a dental information system is the University of the Philippines College of Dentistry (UPCD). In their first attempt to allow professors and students alike to store and access patients' data electronically, UPCD cooperated with a group of Computer Science students enrolled in Software Engineering course to create a dental information system. However the system was reported to have bugs. When the staff tried to reformat the computer in an attempt to fix the bugs, all the system data, along with the software's data, were erased. They were not able to recover it [3].

In 2011, Aurielle Lee, a BS Computer Science student, created another dental information system (Open DentIS) for UPCD. Open DentIS is an OpenMRS module and appears as a separate gutter in the OpenMRS system. OpenMRS is a free and open-source electronic health records system. Open DentIS can be installed on a server and access by UPCD clinicians via the web. Open DentIS makes use of Open- MRS' concept dictionary feature by creating a dental lexicon which is based on UPCD terminologies to standardized dental terms. Moreover, it uses the UPCD and Philippine Dental Association (PDA) standard graphical representation of the teeth which is used to store information about a tooth of a patient [3].

In 2012, Maria Cristina Balsita, another BS Computer Science student, created the second version of Open DentIS known as DentISt. The system added a number of functionalities not found in the previous version and improved the performance of the system. It gives UPCD clinicians free access and storage of electronic patient dental records. DentISt also provides a graphical representation of the teeth which runs faster compared to the one in Open DentIS. New functionalities are also added such as the search of patients by specified criteria, yearly report generation, appointment scheduling. A faculty clinician role is also added which allows professors (faculty clinician) to verify the data entry or update of a student (clinician) [4].

The UP College of Dentistry has a standard workflow of patients which involves steps from collecting the basic information of the patient to recording information on findings and treatments done on the patient. The patient is assigned to a clinician who will perform treatments on him/her. If the treatment is completed, the clinician updates the dental records of the patient, which are the dental chart and the rendered services of the patient. The faculty clinician verifies the findings of the clinician. Finally, the patient is assigned to a new clinician and proceeds to the next required treatment.

Integration of workflow management technology into the dental information system of UPCD can offer great potential for improving dental care delivery to the patients and reducing cost and complexity of the clinical processes. These workflow management technologies are applied in many organizational domains to improve the operational efficiency of business process execution [5].

Since organizations are continually evolving, there is frequently a need for structural change of procedures, such as adding or deleting a step of the procedure, or changing the order in which steps of the procedure are executed. The system supporting these organizational operations changes as well. One option is to flush the system, so that no work is in progress, and then make the structural changes. This option is inconvenient, waste a lot of resources, and in some environments, infeasible. Another option is to perform structural changes dynamically which is a feature provided by a workflow technology [6].

A workflow management system is an application level program which helps to define, execute, coordinate and monitor the flow of work within organizations or workgroups [6]. They are used to coordinate the work of multiple people in a project that has a fixed process. Workflow management systems help improve efficiency in clinical settings and allow better control of processes and tasks. It helps eliminate unnecessary steps in business and clinical processes and in standardizing working methods. More importantly, workflow management system enables system redesign without the need of major changes in business needs and software application.

B. Statement of the Problem

The creation of DentISt by Cristina Balsita is a huge improvement in UPCD system as it allows them to store and manage patient records efficiently. However, the system fails to integrate the workflow of UPCD into the system.

The use of OpenMRS platform by OpenDentIS and DentISt apparently makes programming more complicated especially when adding features not included in OpenMRS. There are limited OpenMRS resources available in the web. Also, OpenMRS is designed for doctors therefore its features mainly fit the doctors' needs. It assumes that any data input or update is final. However in UPCD, a data entered by a clinician is considered final only if it has been verified by the faculty clinician.

Dean Vicente Medina of UPCD also requested few additional features and modifications in the DentISt system. A Student Accomplishment Report and Service Rendered Form must be added in the new UPCD system. The dental chart's labeling of each tooth must be changed into three categories (Caries, Restoration, and Others) instead of listing them all at once each having a different color.

The previous versions of UPCD's dental information system have no workflow support feature. The system has to be reprogrammed if there are changes in the workflow because the system was developed according to the current workflow in UPCD. This method is wasteful and inconvenient.

Another problem in DentISt not having a workflow management is that the faculty and student clinician must be both present when inputting or updating a patient data because OpenMRS assumes that any data input is final. Like in any other business process automation, it is important for this dental system to assume the flow of tasks within UPCD and to allow for system flexibility for possible changes in workflow in the future.

C. Objectives of the Study

To create the workflow management module using jBPM for DentISt 3.0 which has the following roles and functionalities:

1. To allow the workflow administrator to
    (a) design/redesign the workflow of the system
    (b) design and manage the forms of each task in the workflow
    (c) manage the section user group that can perform a task
2. To allow the student clinician and faculty
    (a) perform the tasks that he could do based on the current workflow
    (b) see the list of his pending tasks based on the current workflow
    (c) manage section-specific records of patient
        i. add new encounter section specific record
        ii. edit section-specific record
        iii. view section-specific record

D. Significance of the Project

While an electronic record that stores and collects dental data has great value for UPCD, it gains in more value exponentially when coupled with a feature to support the workflow of those dental data. A workflow component in a dental information system simply places the patient data in the hands of the clinician who needs to act on that data and inform them of the action to be taken using status labels, reminders and annotations. This allows clinicians to be more effective by preventing loss of important data, avoiding redundancies and providing the ability to have cases that require an action from being overlooked.

If UPCD decides to modify their patient workflow in the future, the dynamic workflow management feature integrated into the dental information system can allow the system to adapt to the changes easily. The developer only needs to design the new process then the flow of the application itself will reflect the workflow. With this, the inconvenience, wasted resources or infeasibility of having to implement a new system can be prevented.

The current version of the dental system in UPCD does not allow faculty clinicians to verify the data input of the clinicians. This is due to OpenMRS's assumption that any data input is final and that it does not have a workflow support. These important issues can be solved by integrating a workflow management support into the dental information system.

E. Scope and Limitations

1. The workflow management system that will be intergrated into DentISt 3.0 is jBPM.

2. The initial workflow that will be used is the current workflow of the UP College of Dentistry.

3. The initial sections that will be included are:
    (a) Oral diagnosis
    (b) Oral medicine
    (c) Prosthodontics
    (d) Operative dentistry
4. The initial forms that Oral Diagnosis can edit are:
    (a) Patient Information Form
    (b) Physical Assessment Form
    (c) Vital Signs Form
    (d) Dental History Form
    (e) Medical History Form
    (f) Social History Form
    (g) Soft Tissue Exam Form
    (h) Radiographic Exam Form
    (i) Treatment Plan Form
5. The initial forms that can be edited by all sections are:
    (a) Dental Chart Form
    (b) Services Form
    (c) Findings Form
    (d) Consultations/Referrals Form
    (e) Appointment Form
6. The workflow administrator must be manually informed to reassign patients to other clinician.

F. Assumptions

1. Each patient is designated to only one student clinician per encounter.

2. UP College of Dentistry will assign someone to be the workflow administrator. The workflow administrator must be knowledgeable about workflow design and workflow functionalities of this system. It is also possible to let the system administrator also function as workflow administrator.

II. Review of Related Literature

Dental informatics has led to numerous innovations in dentistry by focusing on research, development and evaluation of information models and computing applications [2]. Though relatively new compared to other areas of medical informatics, it has a great potential for improving quality of health care. Dental information systems and electronic health records, some of the many advances in dental informatics, help in information gathering and efficient management of clinical and dental records.

Since the dawn of the concept of an interorganizational, comprehensive, patient-centered health record in the 1990s, the different concepts of such a record (e.g. computer-based patient record or the CCR - continuity of care record) were always driven by the idea to support health care and to maintain, respectively improve, its quality [7]. Electronic health record (EHR) describes the concept of a comprehensive, cross-institutional, and longitudinal collection of a patient's health and healthcare data. It, therefore, includes data that is not only particularly relevant to a subject's medical treatment but also to a subject's health in general. The patient is regarded as an active partner in his/her treatment by accessing, adding, and managing health-related data, thereby supporting care [8].

Electronic health records (EHRs) are a major development in the practice of dentistry, and dental schools and dental curricula have benefitted from this technology. Patient data entry, storage, retrieval, transmission, and archiving have been streamlined, and the potential for teledentistry and improvement in epidemiological research is beginning to be realized. [9]

Dental schools have recently begun an ambitious goal of converting undergraduate, graduate, and faculty clinics from paper to electronic patient records (EPRs). The functional requirements of EPRs in dental schools are different from those in medicine and nursing. The users of these systems include students, staff, and faculty. In more advanced systems, even patients have access to parts of the record or the ability to make requests such as appointments. Many students, faculty and patients in dental schools believe that EPR improves the legibility and access to a patient chart and that they would recommend such a system to dentists starting a new practice. [10]

Despite the many benefits promised by these systems, few have helped to improve clinician performance, and even fewer have influenced patient outcomes [10]; thus, their adoption in medicine, particularly in dentistry, has been slow and limited [11]. One critical reason is that the designers of these systems often fail to view the clinical workplace as a complex sociotechnical system, and therefore misunderstand that the nature of clinical work is collaborative, distributed, interpretative, interruptive and reactive [12]. This often makes the designed systems very difficult to use. Specifically, an important inhibiting factor is the systems' poor integration into dental workflow. [11].

Health IT promises many benefits for improving quality and efficiency. However, the introduction of health IT can be very disruptive to existing workflows in an organization. Health IT systems often implicitly assume a workflow structure in the way their screens and steps are organized [13]. Although several challenges still exist, introducing workflow management technology in healthcare seems to be prospective in dealing with the problem that the current healthcare information systems cannot provide sufficient support for the process management [14]. The inability to integrate electronic health records (EHRs) into clinician workflow is a well-documented barrier to implementing EHR systems. [15]

In the attempt to improve the existing computer systems, there have been many approaches toward the specification of business(including clinical) processes. However, many BPM tools supported their own languages and they are hard to compare. Because of this, a number of standards and/or widely used approaches emerged over time [16] such as XPDL, BPEL, BPMN.

XML Process Definition Language (XPDL) is the language proposed by the Workflow Management Coalition (WfMC) to interchange process definitions between different workflow products. The goal of XPDL is to provide a Lingua Franca for the workflow domain allowing for the import and export process definitions between a variety of tools ranging from workflow management systems to modelling and simulation tools. Staring point of XPDL is a minimal set of constructs present in most workflow products. Unfortunately, this minimal set does not offer direct support to many of the workflow patterns encountered in practice and present in more mature workflow products. To address this problem, XPDL offers vendor specific extensions. However, this approach definitely does not result in a Lingua Franca. Moreover, to date, even the semantics of the core constructs of XPDL remain undefined. [16]

In 2003, the Business Process Execution Language (BPEL) was proposed. This language combined Microsoft's XLANG and IBM's Web Services Flow Language (WSFL) and is therefore a language that marries two fundamentally different approaches to the specification of executable business processes. Generally speaking, BPEL is a block-structured language where business processes are specified in terms of self-contained blocks that are composed to form larger, more complex, blocks. However, BPEL is not fully block-structured as it supports the specification of dependencies that cross block boundaries through the use of so-called control links. While BPEL was a clear step forward in terms of its support for the specification of control-flow dependencies, the language provided no support for the involvement of human participants in the execution of business activities. In addition, the language has no graphical representation; specifications have an XML-based depiction. [17]

The Business Process Modeling Notation is a widely used standard for business process modeling, providing a graphical notation for specifying business processes in a Business Process Diagram (BPD that defines (1) a graphical notation for business processes, (2) a (meta)model for business processes and (3) an interchange format to exchange BPMN process definitions between different tools. The specification aims to support a wide range of different roles from business users and business analysts to technical developers involved in the designing, implementing, managing and monitoring business processes [18].

Workflow management becomes more and more important along with the development of medical technique. Traditional medical information system cannot keep up with the ever-changing demands [19]. Some recent attempts to integrate workflow management aim to solve the issues on the current medical information systems.

A proclet framework is designed [20] to address the issue on fragmented healthcare processes which is composed of separate but intertwined life-cycles running at different speeds. Using proclets, processes in which interaction is considered as a first-class citizen are effectively described instead of straightjacketing them into one monolithic workflow. Problems on healthcare processes operating at different levels of granularity and the one-to-many and many-to-many relationships that exist between entities in a workflow are also solved by using proclets.

Furthermore, TNest, a new advanced, structured and highly modular workflow modeling language, allows one to easily express data dependencies and time constraints during process design. TNest is able to check temporal controllability and model data dependencies among tasks through message passing mechanism [21]. However, as TNest is a relatively new workflow language, it is not yet completely matured and has few implementations in real settings.

Along with the many approaches towards business process specification and reenactment originating both from academia and industry, the maturity and popularity of open-source software are also increasing. Three well-known open-source workflow management systems are jBPM[1], OpenWFE[2] and Enhydra Shark[3] [22]. Other open-

[1]`http://www.jboss.org/jbpm`
[2]`http://sourceforge.net/projects/openwfe/`
[3]`http://www.enhydra.org`

source workflow management systems are Intalio BPMS[4] and YAWL[5].

OpenWFE is an active project on Sourceforge, labeled as “Production/Stable” and having more than 100,000 downloads. OpenWFE is originally written in Java and later on migrated to Ruby. OpenWFE has a powerful language for workflow specifications in terms of its support for the workflow patterns. However, OpenWFE lacks explanatory documentation for its graphical notation, the user management tool, and the command line administration tool which results to complicated user management and requires a deep understanding of operational aspects of the tool [22].

Enhydra Shark is a Java workflow engine offering from Together Teamlosungen and ObjectWeb. Enhydra Shark supports XPDL (the standard proposed by the WfMC). The workflow engine is one of the several products under constant development [22]. Unlike other open source offerings, one version of Enhydra Shark is also distributed as a closed-source product [16] and some of its desirable functionality is only present in this version. Enhydra Shark supports a relatively limited set of control-flow operators offering little support for the patterns outside the basic control-flow category [22].

Intalio BPMS as described in [23] is a venture to bring business process modeling to mainstream. Its main objective is to increase the availability of process modeling and development skills and moving the software to mainstream users.

Yet Another Workflow Language (YAWL) is based on Petri nets but extended with additional features to facilitate the modelling of complex workflows such as workflows dealing with cancelation, synchronization of active branches only, and multiple concurrently executing instances of the same task [24]. However, YAWL is not yet fully implemented and missing quite a lot of more advanced BPM features [25].

Java for Business Process Management(jBPM) is by far the most popular open

[4]`http://www.intalio.com/bpms`
[5]`http://www.yawlfoundation.org/`

source workflow management system. It is JBoss's (RedHat's) workflow management system and is distributed through SourceForge under the LGPL license [16].

JBoss jBPM enables IT flexibility by supporting multiple-process languages with the same scalable process engine platform. jBPM is based on a generic process engine, which is the foundation to support multiple process languages natively. jBPM5 focusses on BPMN 2.0 as the language for expressing business processes. BPMN 2.0 is a standardized specification that defines a visualization and XML serialization of business processes, and can be extended (if necessary) to include more advanced features. [26]

JBoss jBPM's pluggable architecture is extensible and customizable on every level: within the process engine, for each process definition, and every corresponding process instance. JBoss jBPM provides a process-oriented programming model (jPDL) that blends the best of both Java and declarative programming techniques and enables developers to structure their software around an easy to understand process graph. This approach describes business processes in a common dialect that lets business people and developers speak the same language, facilitating a more agile implementation of the processes required by business people [27].

III. Theoretical Framework

A. Dental Informatics

Dental informatics is the application of computer and information sciences to improve dental practice, research, education and management. Numerous applications that support clinical care, education and research have been developed. Dental informatics is beginning to exhibit the characteristics of a discipline: core literature, trained specialists and educational programs. Subgoals include the efficient delivery of dental care and firm support of research and education relating to the discipline. Dental informatics presents possible solutions to many long-standing problems in dentistry, but it also faces significant obstacles and challenges. [2].

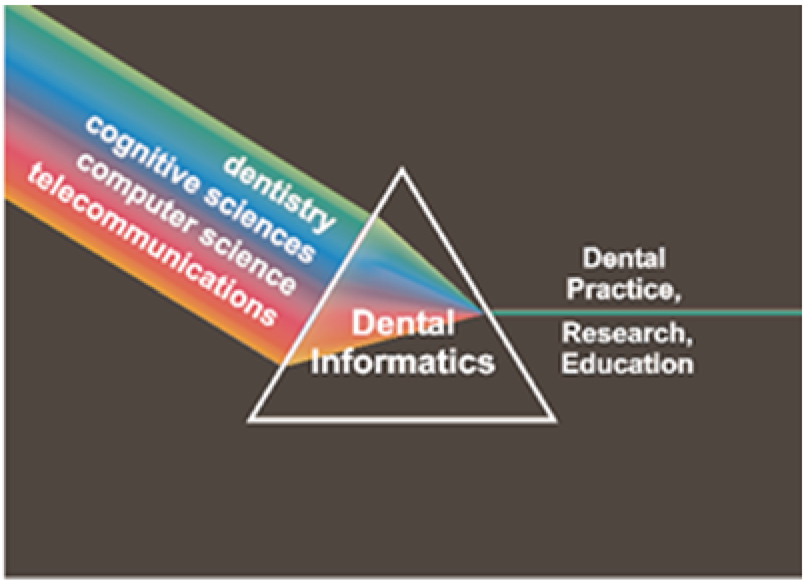


Figure 1: Dental informatics combines its methodological foundations to address problems in practice, research, and education [1]

Figure 1 shows how dental informatics combines primarily with four more component sciences of informatics to develop solutions in dental practice, research and education [1]. Dental informatics derives methods, theories, and techniques from sciences such as dentistry, computer science, cognitive science, and telecommunications.

Dentistry is defined by the World Health Organization(WHO) as the science and art of preventing, diagnosing and treating diseases, injuries and malformations of the teeth, jaws, and mouth [28].

Computer science is a discipline that involves the understanding and design of computers and computational processes. It emphasizes not on information, but how it is represented, processed, manipulated, and managed in computing systems. Computer science studies and develops data representations, algorithms, programming languages, operating systems, and computational approaches.

Cognitive science, on the other hand, is a research area that draws on several fields (such as psychology, artificial intelligence, linguistics, and philosophy) to develop theories of perception, thinking, and learning. The central hypothesis of cognitive science is that thinking can best be understood in terms of representational structures in the mind and computational procedures that operate on those structures.

Finally, telecommunications is the science that deals with communication at a distance. Key research issues in telecommunications include how computers communicate with each other, how communication traffic is routed, how bandwidth is used most efficiently, and how communication can be kept secure.

B. UP College of Dentistry

The UP College of Dentistry was first established as a Dept. Of Dentistry of the College of Medicine and Surgery on February 8, 1915. Upon the recommendation of the late Dean Antonio G. Sison of the College of Medicine, the Board of Regents of the University passed a resolution changing the status of the School of Dentistry to an independent unit as the College of Dentistry on October 21, 1948 [29]. As a newly established independent unit of the University of the Philippines, the College of Dentistry envisions itself to be the country's premier academic institution providing quality dental education, training, research and service characterized by global competence, social sensitivity and responsible leadership in the continuous pursuit of excellence for the service of God and the nation [29].

1. UPCD Structure

UP College of Dentistry accepts patients in Oral Diagnosis where basic patient information are gathered and different examinations are performed to determine what treatment does a patient need. These include soft tissue, dental and radiographic exams.

The UP College of Dentistry consists of different sections where clinicians work and perform treatments on patients. The three main sections are the following:

- Oral Medicine - Periodontics, Oral Surgery, Endodontics
- Prosthodontics - Removable Prosthodontics, Fixed Partial Prosthodontics
- Operative Dentistry - Orthodontics, Pedodontics, Restorative Dentistry

Oral medicine is a specialty of dentistry concerned with the oral health care of patients with chronic, recurrent and medically related disorders of the oral and maxillofacial region, and with their diagnosis and non-surgical management. The Prosthodontics section specializes with the diagnosis, treatment planning, rehabilitation and maintenance of the oral function, comfort, appearance and health of patients with clinical conditions associated with missing or deficient teeth and/or oral and maxillofacial tissues using biocompatible substitutes [30]. While Operative dentistry focuses primarily on the diagnosis, prevention, treatment and prognosis of diseases or trauma to teeth. Treatments conducted should restore proper tooth morphology, function, esthetics and harmonious relationship with the surrounding tissues.

Patients may also be endorsed to outside sections or clinics that can perform specific exams or treatments not covered by UPCD.

2. UPCD Patients Workflow

The standard workflow of patients of UPCD consists of steps starting from the collection of basic information to scheduling of appointments with assigned clinicians. A treatment is performed to a patient only by the clinician he/she was assigned. When a treatment cannot be completed in one day, a patient is allowed to come back for more appointments until the treatment is finished. If all the needed treatments are carried out, the patient is checked out. If not, the patient is assigned to either a new clinician or his/her previous clinician and then proceeds to another treatment. The processes going on inside sections of UPCD are no longer in scope with the proposed dental information system. Figure 2 and 3 summarizes the workflow of patients [4].

The Oral Diagnosis (OD) section of UPCD is responsible for the management of patient records. When a patient is admitted in UPCD, a patient record is created. The clinician in OD then collects patient data starting from basic information such as the name, age, adrress, occupation, birthday and contact number. Then the patient undergoes physical assessment. Medical, social and dental history are asked and any history of illness is recorded. If the patient record already exists, the clinician in OD verifies if there is a clinician assigned to the patient. If none, this patient is a returning patient with new complaints so examinations are performed again. If there is a clinician assigned to the patient, either a treatment is not finished yet or another treatment is to be performed. The patient is examined by the clinician assigned to him/her [4].

Next, the soft tissue examination is performed. If needed, the patient is requested to take dental radiographic or X-ray examinations to be studied by the clinician. Analysis of the radiographs are then written down on a patient data sheet. The patient's mouth and teeth are examined and observations are also examined and recorded in the dental status chart [4].

By then, all of the services needed by the patients, problems to be addressed and the proposed treatment are listed in the patient record. The patient is then referred to the sections in UPCD (Operative Dentistry, Oral Medicine, Prosthodontics) which will cover the treatments. A clinician belonging to that section is assigned to the patient to start the first treatment [4].

The clinician assigned to the patient then examines and double checks the information and exam results of the patient. The treatment is then carried out. If the treatment is not finished along the day, the clinician may schedule more appointments with patient until it is finished. If the treatment is finished, the clinician checks if all the treatments needed by the patient are carried out. If all treatments are carried out already, the patient is checked out. If not, the patient is either assigned to a new clinician or the current clinician who then performs the next treatment. To keep track of services rendered by clinicians to the patient, they are listed in the patient record [4].

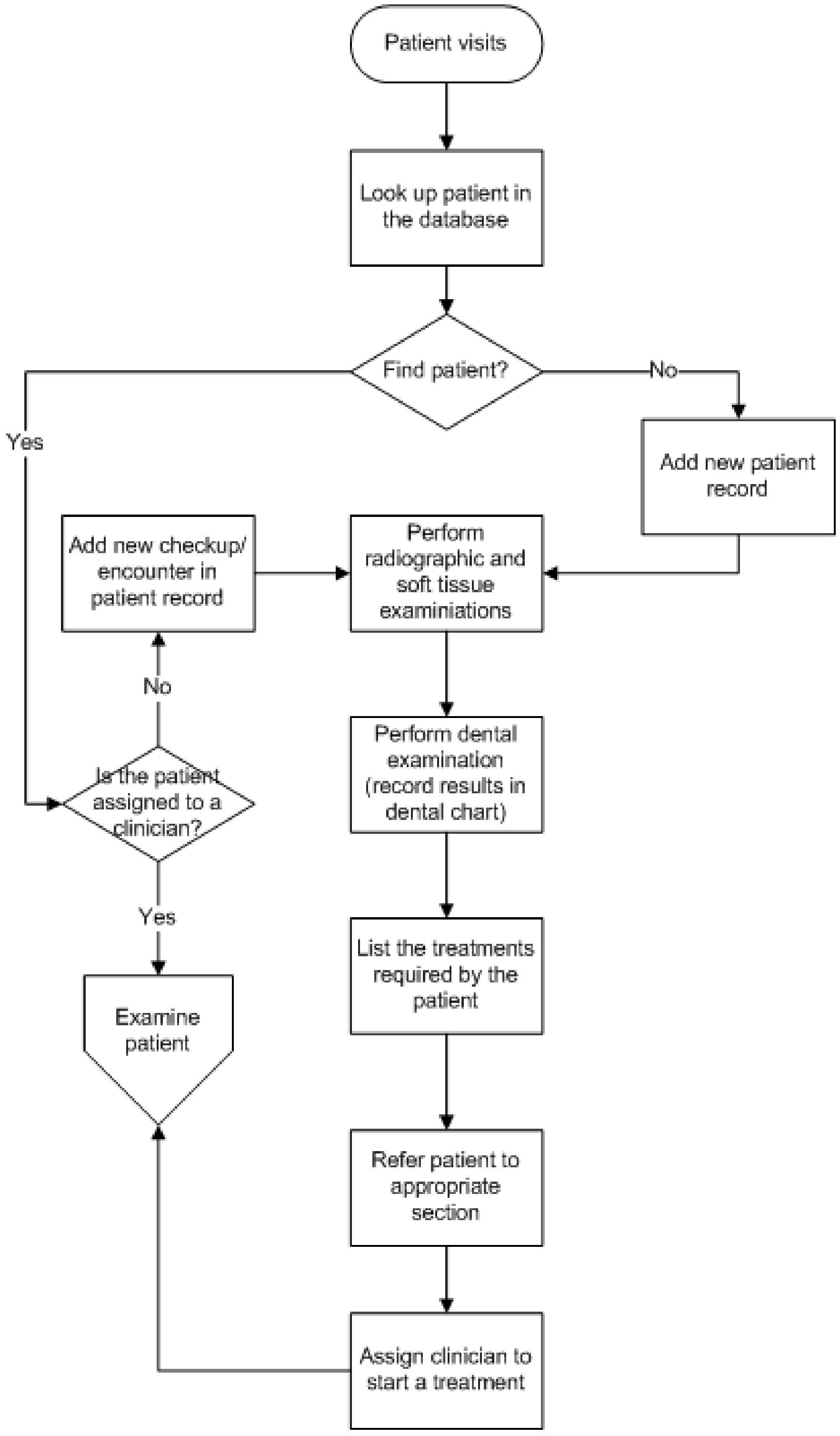

Patient visits
Look up patient in the database
Find patient?
No
Yes
Add new patient record
Add new checkup/ encounter in patient record
Perform radiographic and soft tissue examiniations
No
Perform dental examination (record results in dental chart)
Is the patient assigned to a clinician?
Yes
List the treatments required by the patient
Examine patient
Refer patient to appropriate section
Assign clinician to start a treatment

Figure 2: Workflow of Patients of UPCD

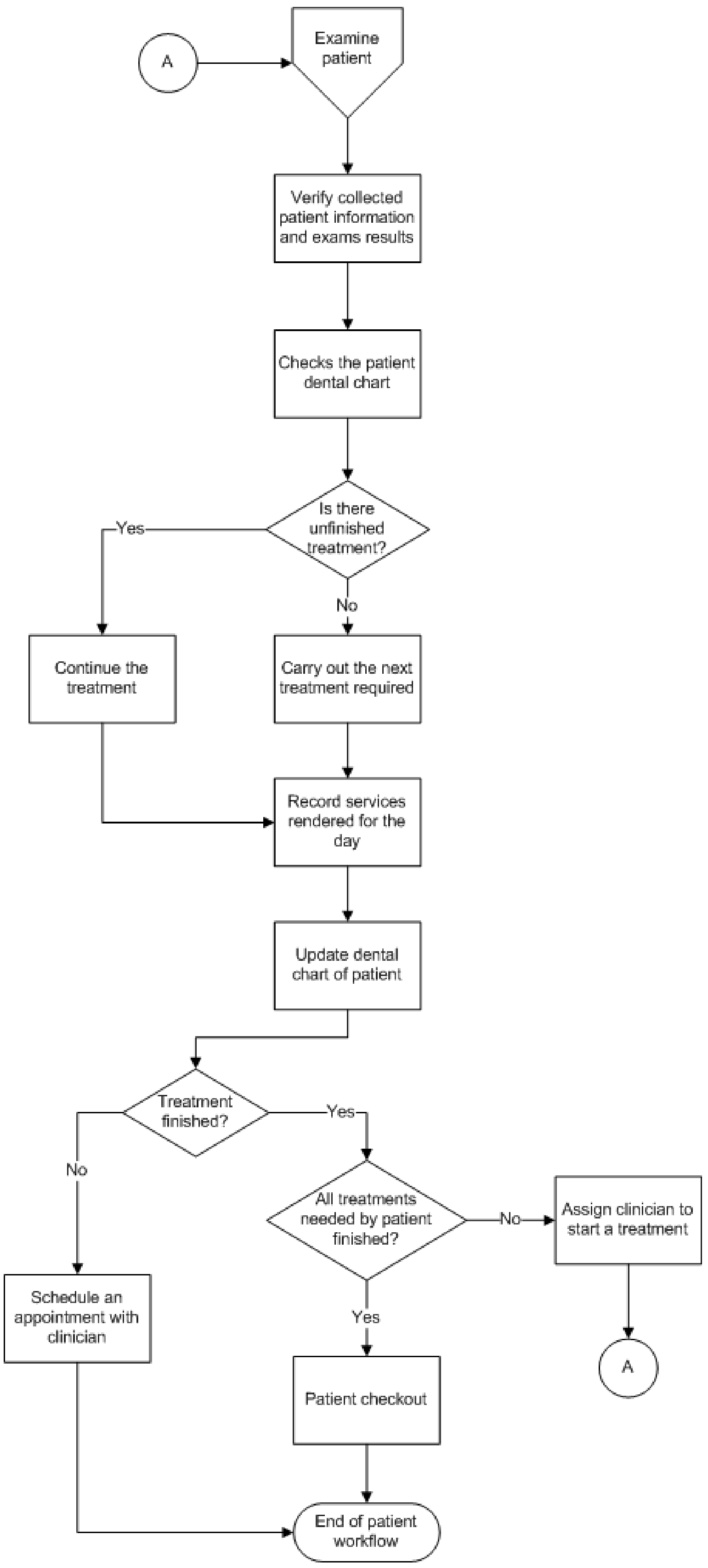

A
Examine patient
Verify collected patient information and exams results
Checks the patient dental chart
Is there unfinished treatment?
Yes
No
Continue the treatment
Carry out the next treatment required
Record services rendered for the day
Update dental chart of patient
Treatment finished?
Yes
No
All treatments needed by patient finished?
No
Assign clinician to start a treatment
A
Yes
Schedule an appointment with clinician
Patient checkout
End of patient workflow

Figure 3: Workflow of Patients of UPCD

C. Workflow Management System

1. Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is a standard developed by The Object Management Group (OMG). The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation [18].

Another goal, but no less important, is to ensure that XML languages designed for the execution of business processes, such as WSBPEL (Web Services Business Process Execution Language), can be visualized with a business-oriented notation [18].

BPMN supports the different levels of process modelling namely process maps, process descriptions and process models. Process modelling is capturing of the ordered sequence of business activities and supporting information. Process maps are simple flow chart of the activities. Process descriptions are flowcharts extended with additional information, but not enough to fully define actual performance. Finally, process models are flowcharts extended with enough information so that the process can be analysed, simulated and/or executed [31].

BPMN defines a Business Process Diagram (BPD), which based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Diagram, then, is a network of graphical objects, which are activities and the flow controls that define their order of performance [31].

2. Java for Business Process Management (jBPM)

jBPM is a flexible Business Process Management (BPM) Suite. It is light-weight, fully open-source (distributed under Apache license) and written in Java. It allows you to model, execute and monitor business processes, throughout their life cycle [32].

A business process allows you to model your business goals by describing the steps that need to be executed to achieve that goal and the order, using a flow chart. This greatly improves the visibility and agility of your business logic. jBPM focuses on executable business process, which are business processes that contain enough detail so they can actually be executed on a BPM engine. Executable business processes bridge the gap between business users and developers as they are higher-level and use domain-specific concepts that are understood by business users but can also be executed directly [32].

The core of jBPM is a light-weight, extensible workflow engine written in pure Java that allows you to execute business processes using the latest BPMN 2.0 specification. It can run in any Java environment, embedded in your application or as a service [32].

jBPM supports adaptive and dynamic processes that require flexibility to model complex, real-life situations that cannot easily be described using a rigid process. We bring control back to the end users by allowing them to control which parts of the process should be executed, to dynamically deviate from the process, etc [32].

jBPM is also not just an isolated process engine. Complex business logic can be modeled as a combination of business processes with business rules and complex event processing. jBPM can be combined with the Drools project, a business logic integration platform which provides a unified and integrated platform for rules, workflow and event processing [33], to support one unified environment that integrates these paradigms where you model your business logic as a combination of processes, rules and events [32].

3. Integrating jBPM with other systems

JBoss jBPM is designed to run as a standalone solution or be seamlessly embedded within any Java application. JBoss jBPM's pluggable architecture allows extensibility and customizability on every level; within the process engine, for each process definition and every corresponding process instance. JBoss jBPM's pluggable architecture provides workflow, orchestration and BPM in one platform, simplifying business development. Traditional standalone BPM, orchestration and workflow products force users to deploy multiple platforms to offer a similar range of capability, thereby adding complexity to the development process [34].

There are three main possibilities of embedding the jBPM framework into an application:

- Standalone Applications
- Enterprise Application
- Web Applications

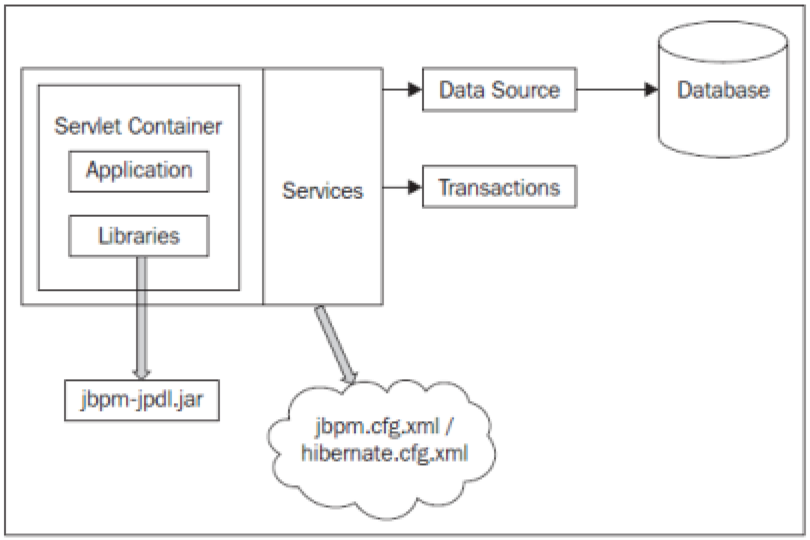


Figure 4: Web application with jBPM dependency

jBPM can be deployed in a web application as shown in Figure 4. The application can run on an application server or inside a servlet container. It can include the jBPM JARs or the container have the libraries and will also use the jBPM APIs directly. In this scenario, the user interacts with process using a webpage that is configured to access a database using a JDBC driver directly or via a DataSource configuration [35]. In jBPM4, jBPM can be installed directly as part of the web application.

IV. Design and Implementation

A. Context Diagram

The DentISt 3.0 will have four main types of roles - the System Administrator, Workflow Administrator, Student Clinicians and the Faculty. The context diagram is shown in Figure 5.

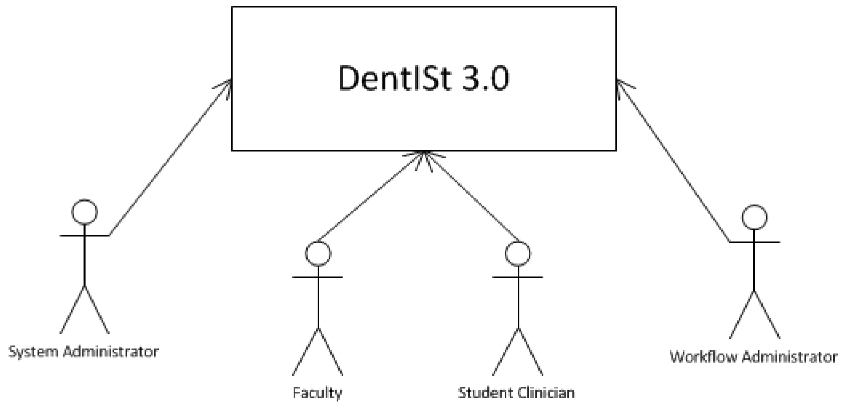


Figure 5: Context Diagram of DentISt3.0

B. Use Case Diagram

Student clinician and faculty can manage patient record and perform tasks in the section. Additionally, the faculty can also view statistics and manage sections. The workflow administrator can manage the workflow of the system. Finally, the system administrator can view statistics, manage user accounts, manage sections and manage roles. Figure 6 shows the top level use case diagram of DentISt where use cases colored in gray are the workflow management module, use cases in white are for the fined grained access control module and use cases in black for both modules.

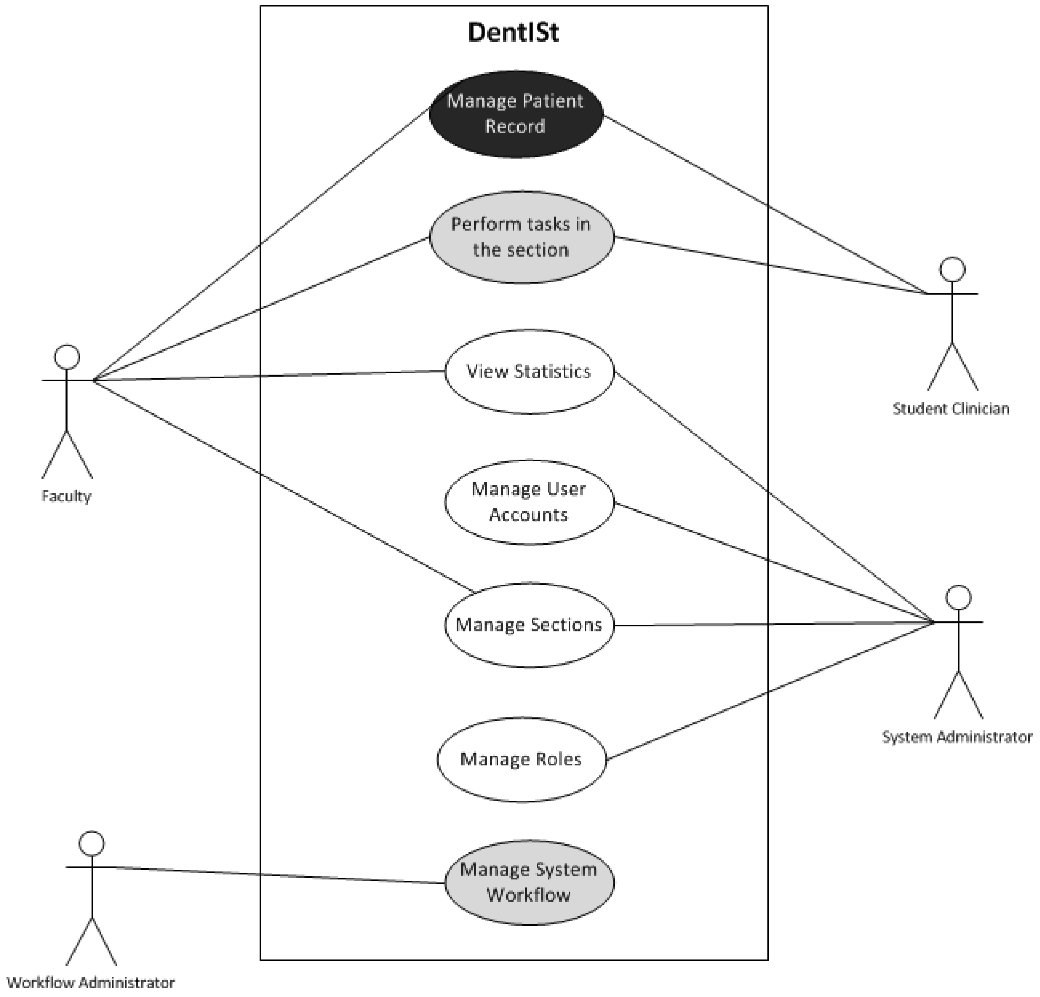


Figure 6: Top Level Use Case Diagram of DentISt3.0

1. Manage System Workflow

The Manage System Workflow Use Case involves user accounts with workflow administrator role. Workflow administrator can design the workflow specification of the system using graphical interfaces like flowchart etc. The workflow administrator can also edit and delete the current workflow of the system.Figure 7 shows the use case diagram.

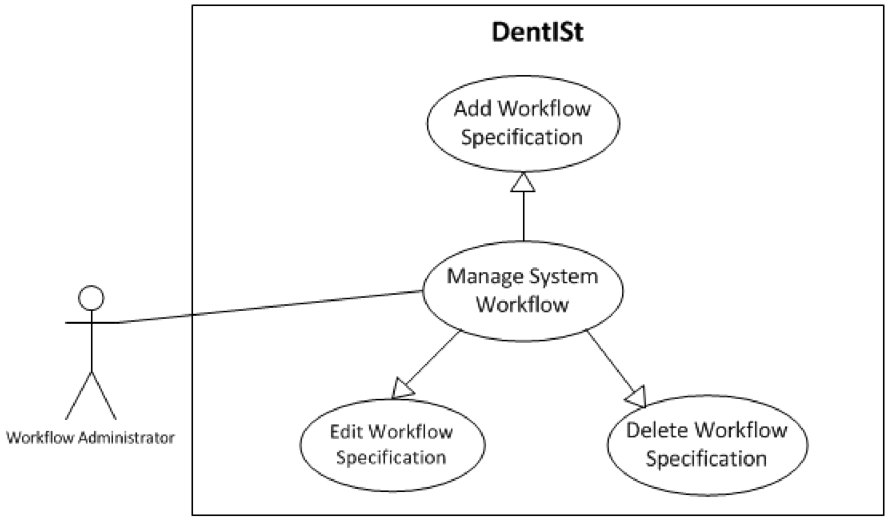

DentISt
Add Workflow Specification
Manage System Workflow
Edit Workflow Specification
Delete Workflow Specification
Workflow Administrator

Figure 7: Manage System Workflow Use Case Diagram of Workflow Administrator

Activity Diagrams of Manage System Workflow are shown in Figures 8, 9 and 10.

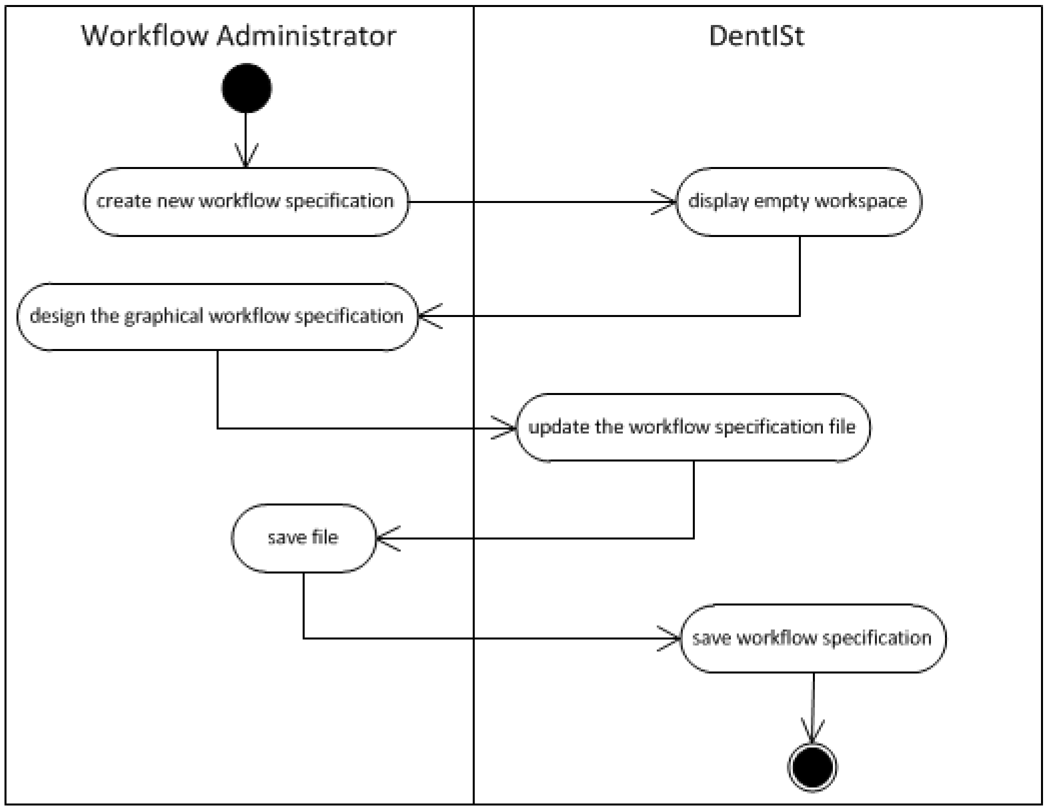

Workflow Administrator
DentISt
create new workflow specification
display empty workspace
design the graphical workflow specification
update the workflow specification file
save file
save workflow specification

Figure 8: Add Workflow Specification Activity Diagram of Workflow Administrator

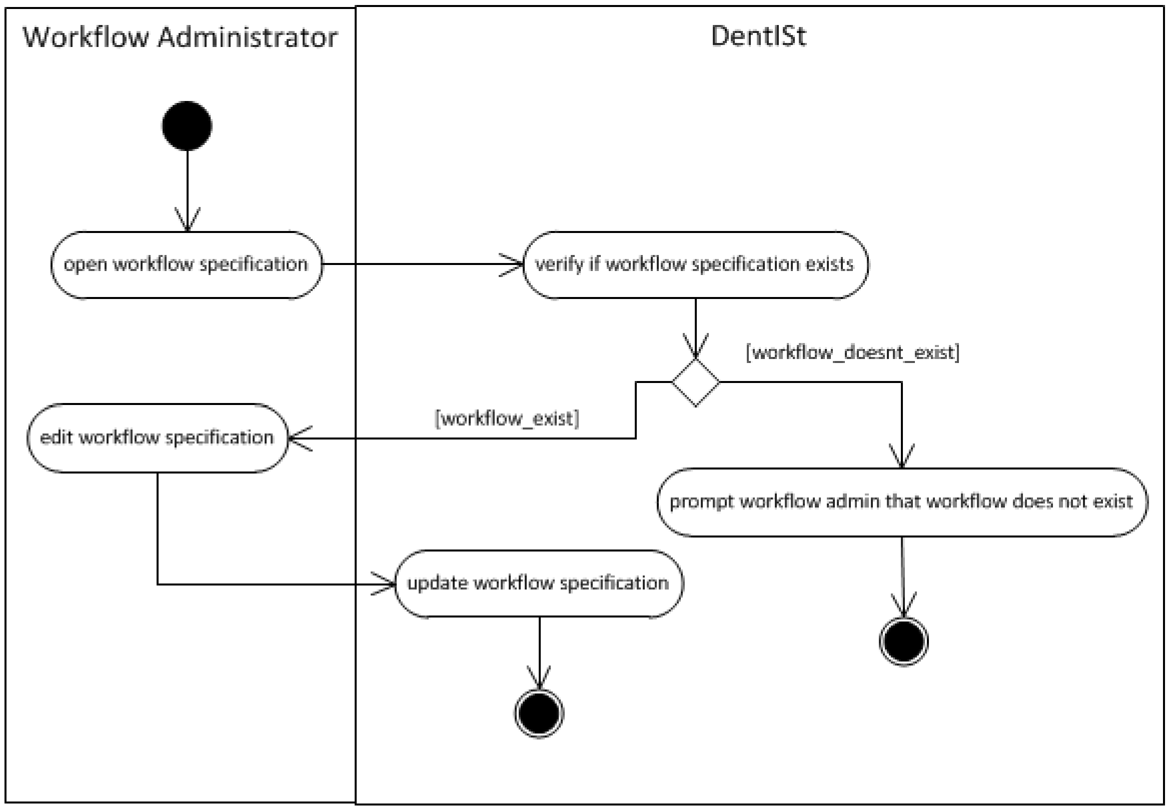

Workflow Administrator
DentISt
open workflow specification
verify if workflow specification exists
[workflow_doesnt_exist]
[workflow_exist]
edit workflow specification
prompt workflow admin that workflow does not exist
update workflow specification

Figure 9: Edit Workflow Specification Activity Diagram of Workflow Administrator

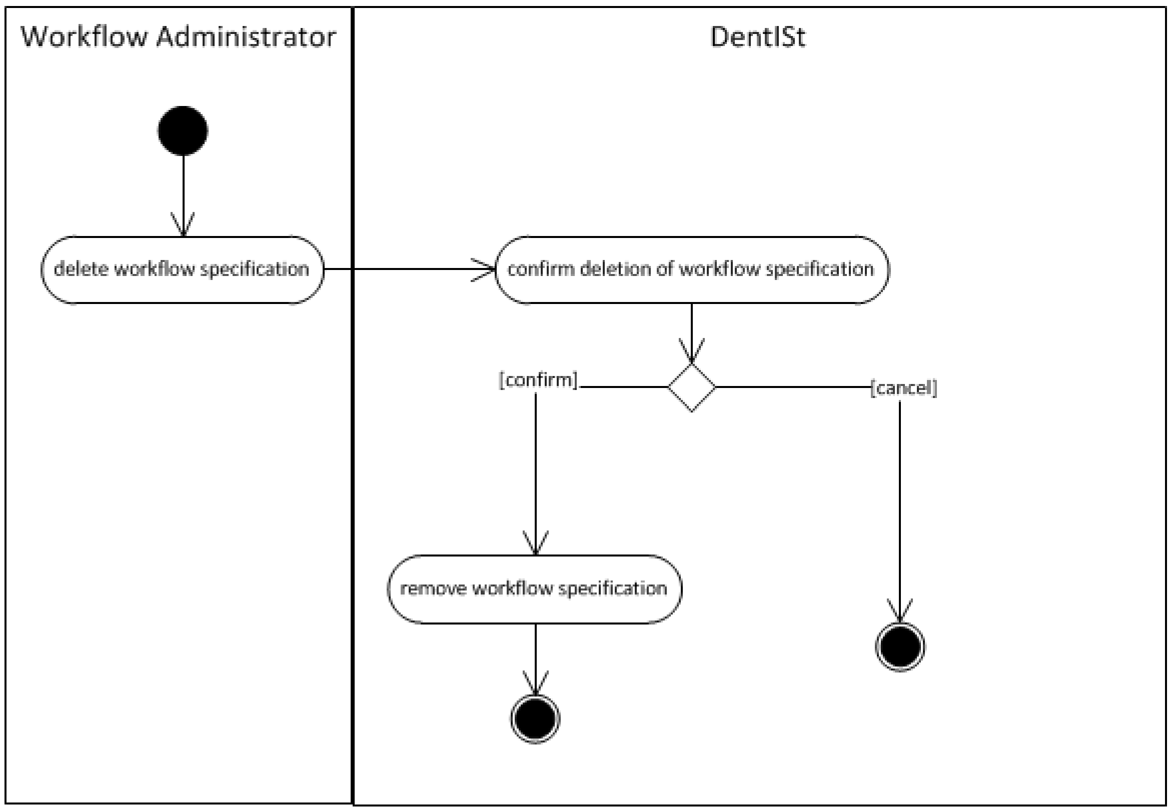

Workflow Administrator
DentISt
delete workflow specification
confirm deletion of workflow specification
[confirm]
[cancel]
remove workflow specification

Figure 10: Edit Workflow Specification Activity Diagram of Workflow Administrator

2. Manage Patient Record

The Manage Patient Record allows student clinician and faculty to manage section-specific records as shown in the use case colored in gray. The add, edit patient record, and search/view for patients meeting specific criteria are handled by the fined grained access control module. The Manage Section-Specific Record Use Case involves functionalities to be performed by student clinician and faculty. They can add a new encounter of the section-specific record, edit a section-specific record and view section-specific record. Figure 11 shows the use case diagram of manage patient record and Figure 12 shows the use case diagram of manage section-specific record.

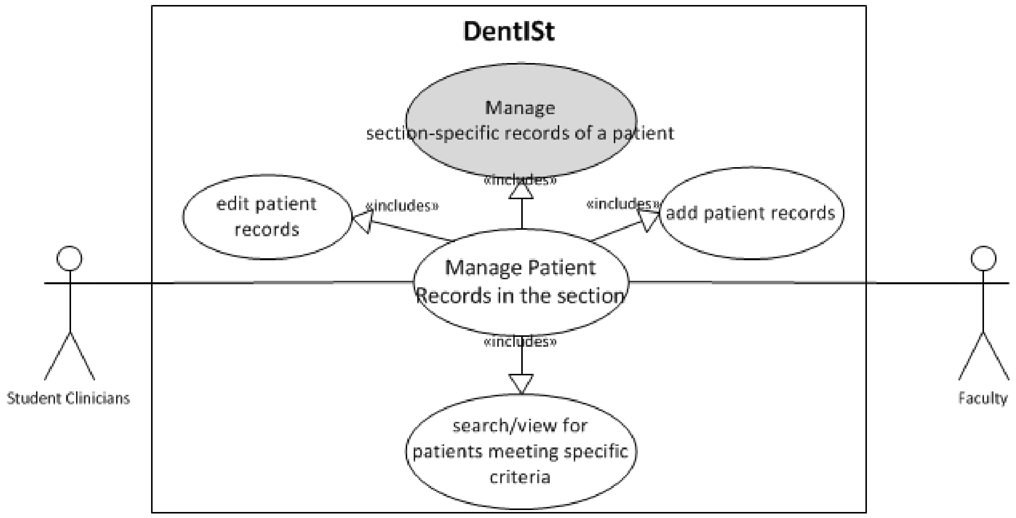

DentISt
Manage section-specific records of a patient
«includes»
edit patient records
«includes»
«includes»
add patient records
Manage Patient Records in the section
«includes»
search/view for patients meeting specific criteria
Student Clinicians
Faculty

Figure 11: Manage Patient Record Use Case Diagram of Faculty and Student Clinician

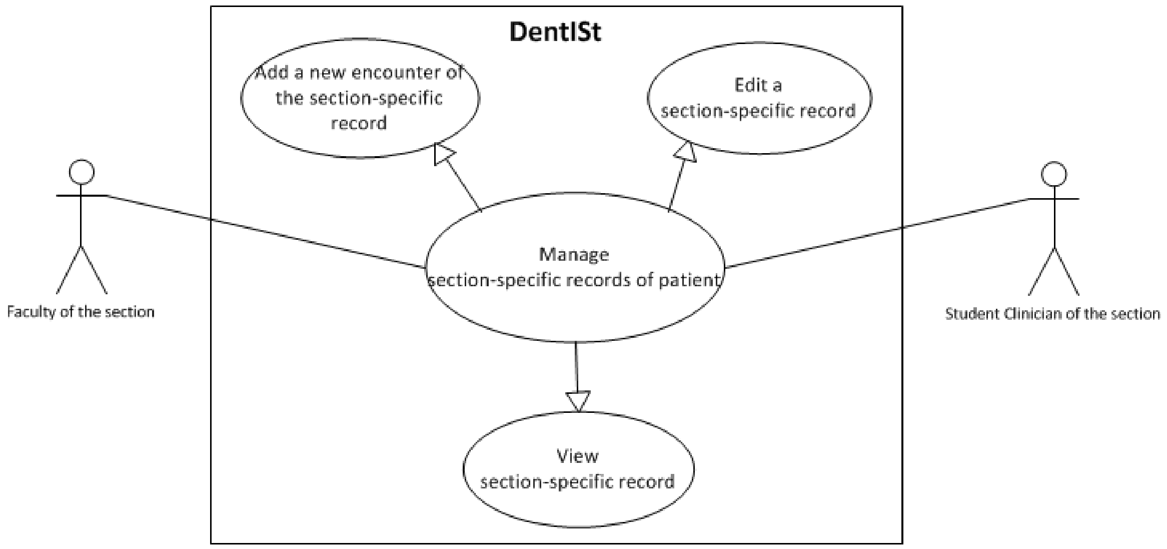

DentISt
Add a new encounter of the section-specific record
Edit a section-specific record
Manage section-specific records of patient
View section-specific record
Faculty of the section
Student Clinician of the section

Figure 12: Manage Section-Specific Record Use Case Diagram of Faculty and Student Clinician

Activity Diagrams of Manage Section-Specific Record Use Case of Student Clinician and Faculty are shown in Figures 13, 14, and 15.

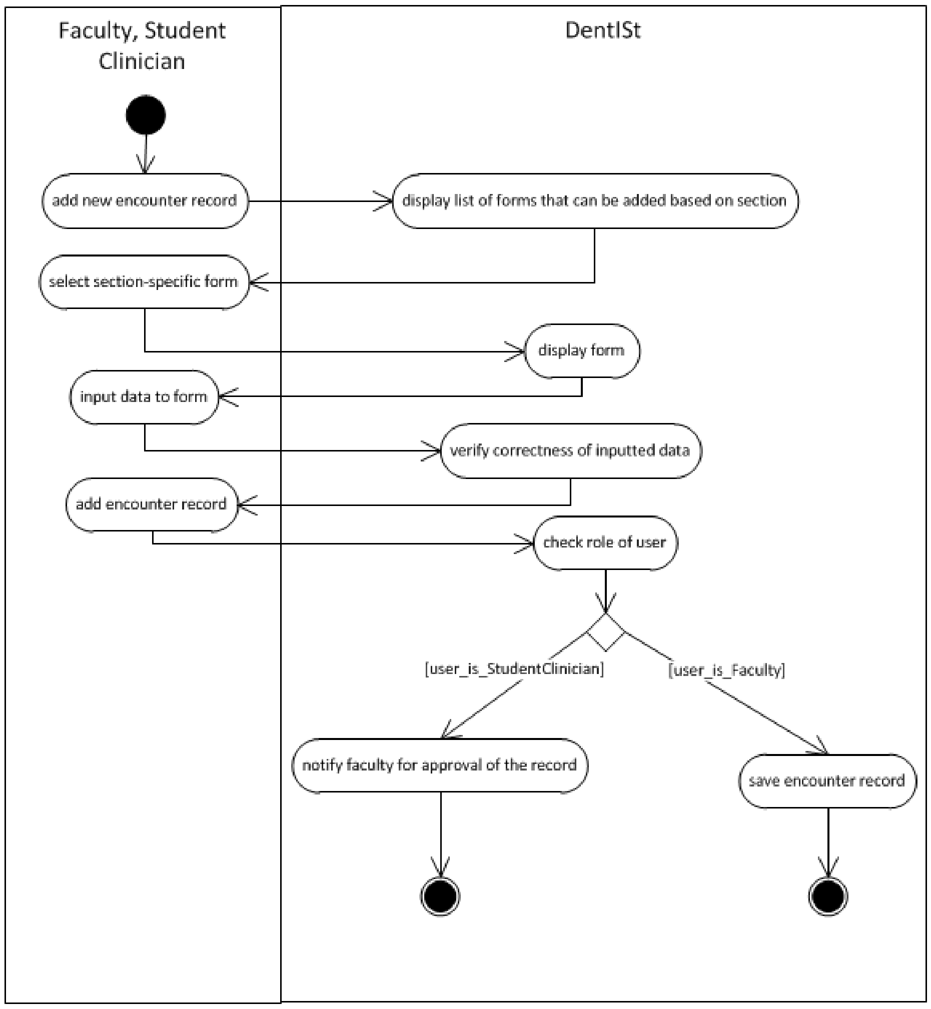

Faculty, Student Clinician
DentISt
add new encounter record
display list of forms that can be added based on section
select section-specific form
display form
input data to form
verify correctness of inputted data
add encounter record
check role of user
[user_is_StudentClinician]
[user_is_Faculty]
notify faculty for approval of the record
save encounter record

Figure 13: Add New Section-Specific Record Activity Diagram of Student Clinician and Faculty

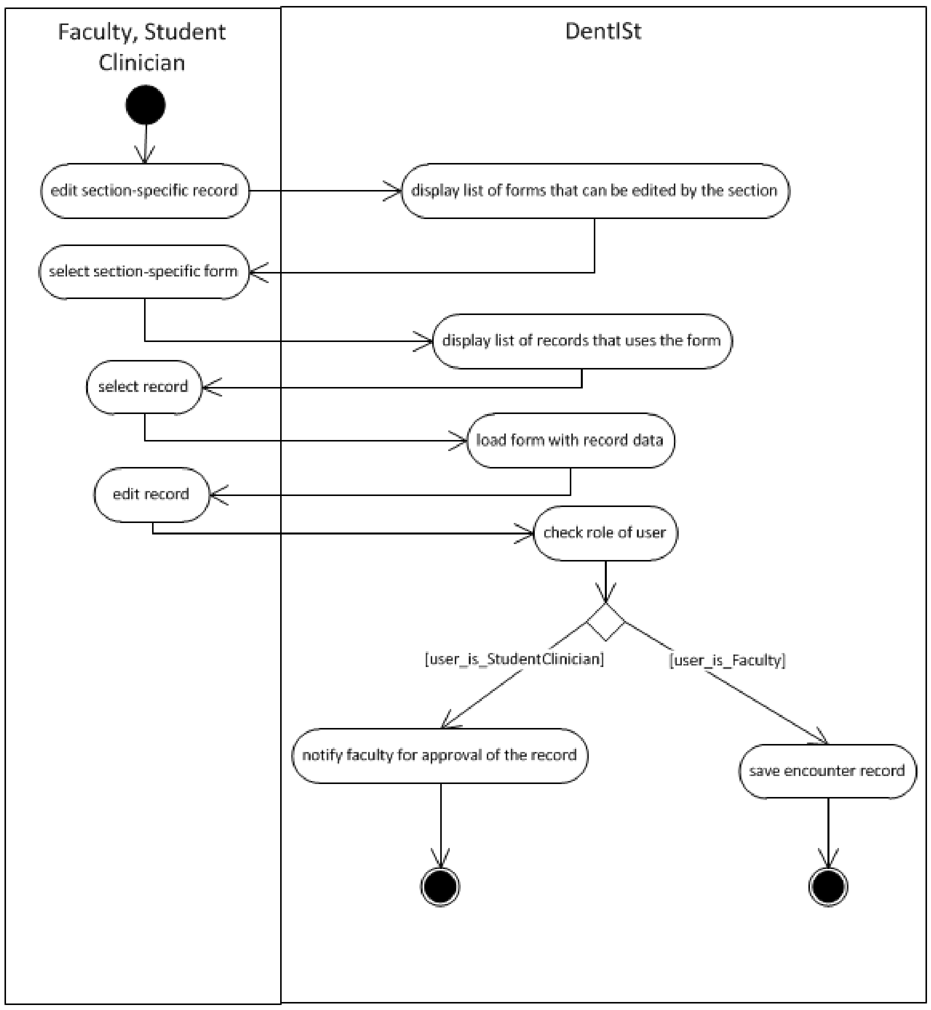

Faculty, Student Clinician
DentISt
edit section-specific record
display list of forms that can be edited by the section
select section-specific form
display list of records that uses the form
select record
load form with record data
edit record
check role of user
[user_is_StudentClinician]
[user_is_Faculty]
notify faculty for approval of the record
save encounter record

Figure 14: Edit Section-Specific Record Activity Diagram of Student Clinician and Faculty

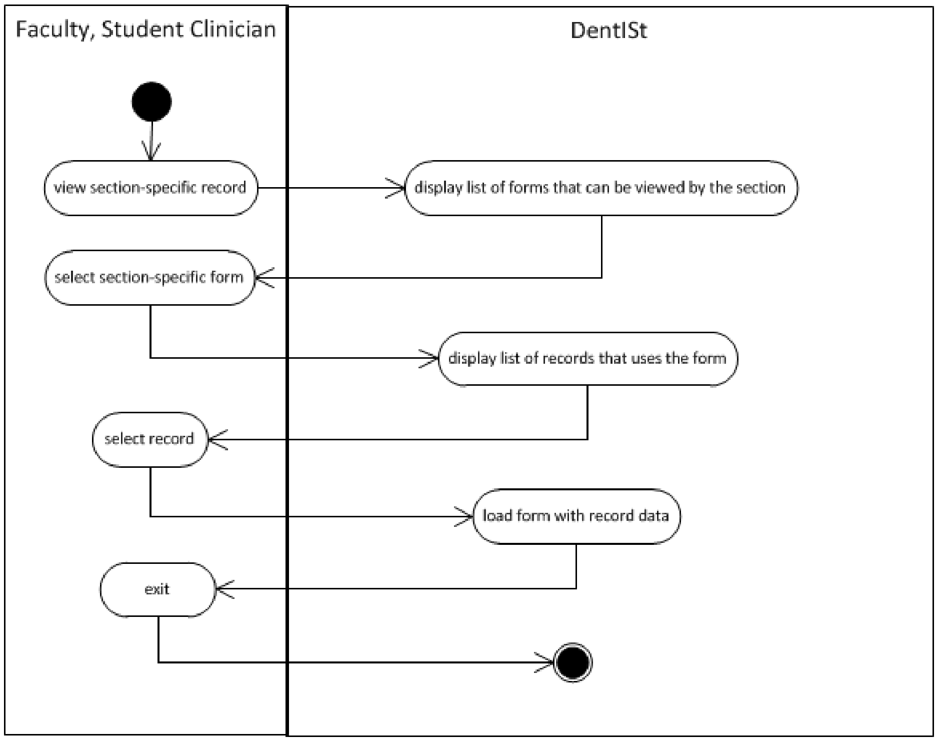

Faculty, Student Clinician
DentISt
view section-specific record
display list of forms that can be viewed by the section
select section-specific form
display list of records that uses the form
select record
load form with record data
exit

Figure 15: View Section-Specific Record Activity Diagram of Student Clinician and Faculty

3. Perform Tasks in the Section

Perform Tasks in the Section Use Case allows student clinician and faculty execute their pending tasks such as appointments, approve updates in patient record, approve student accomplishment and grant permission to access record to the student clinicians from other sections. List of tasks will be based on the role and section of the user. Figure 16 shows the use case diagram.

Activity Diagram of Perform Workflow Tasks Use Case of Student Clinician and Faculty is shown in Figure 17.

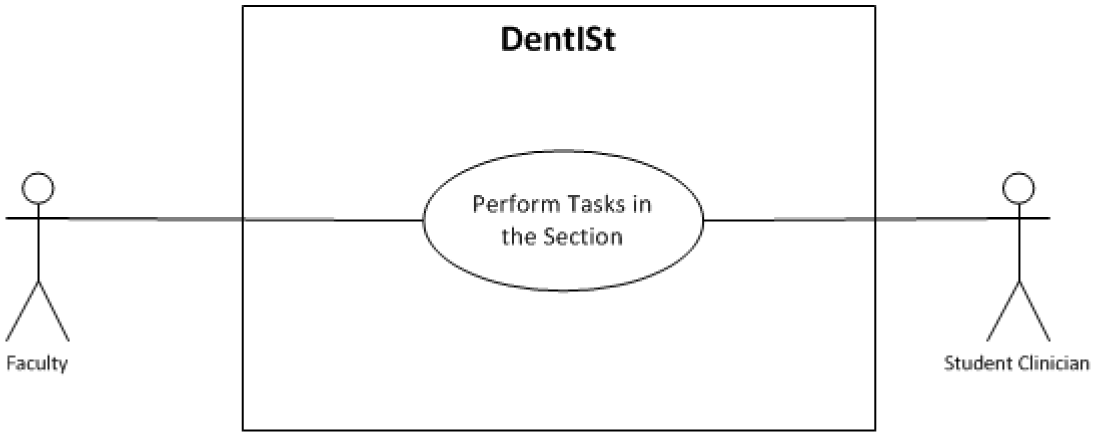


Figure 16: Perform Workflow Tasks Use Case Diagram of Student Clinician and Faculty

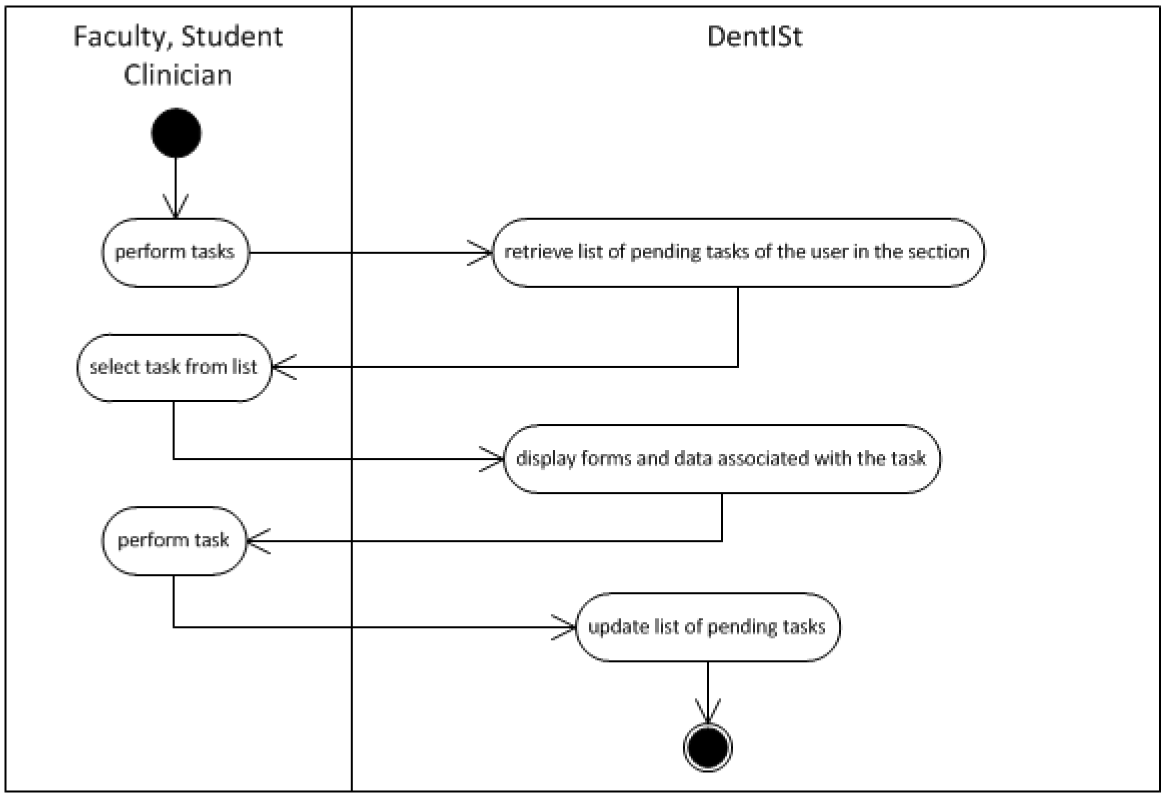

Faculty, Student Clinician
DentISt
perform tasks
retrieve list of pending tasks of the user in the section
select task from list
display forms and data associated with the task
perform task
update list of pending tasks

Figure 17: Perform Workflow Task Activity Diagram of Student Clinician and Faculty

4. View Statistics

Workflow administrator can view section-specific statistics. Figure 18 shows the View Statitics Use Case Diagram of Workflow Administrator in DentISt.

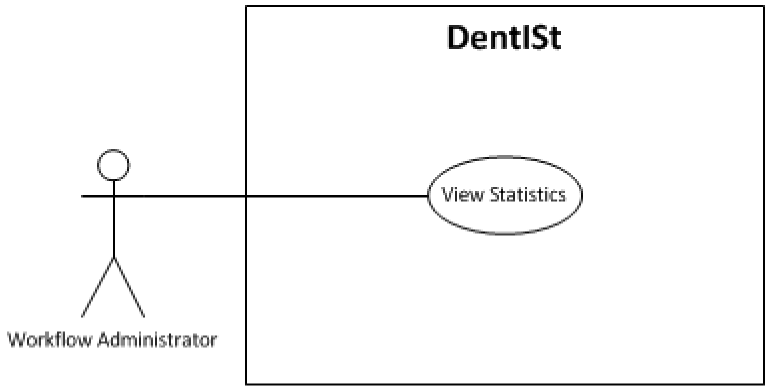


Figure 18: View Statistics Use Case Diagram of Workflow Administrator

Activity Diagram of View Statistics Use Case of Workflow Administrator is shown in Figure 19.

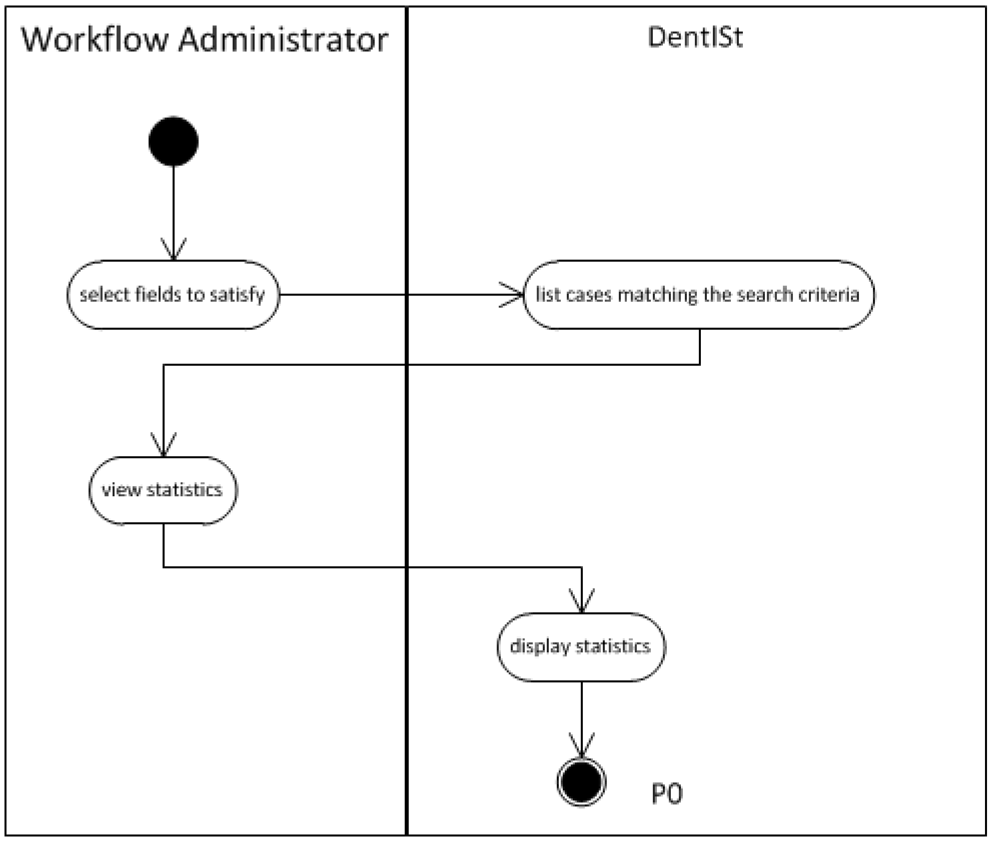


Figure 19: View Statistics Activity Diagram of Workflow Administrator

C. Entity Relationship Diagram(ERD)

The Entity Relationship Diagram of User Cluster is shown in Figure 20. The cluster consists of user, role, section,user role, role section, database role, audittrail and configuration entities.

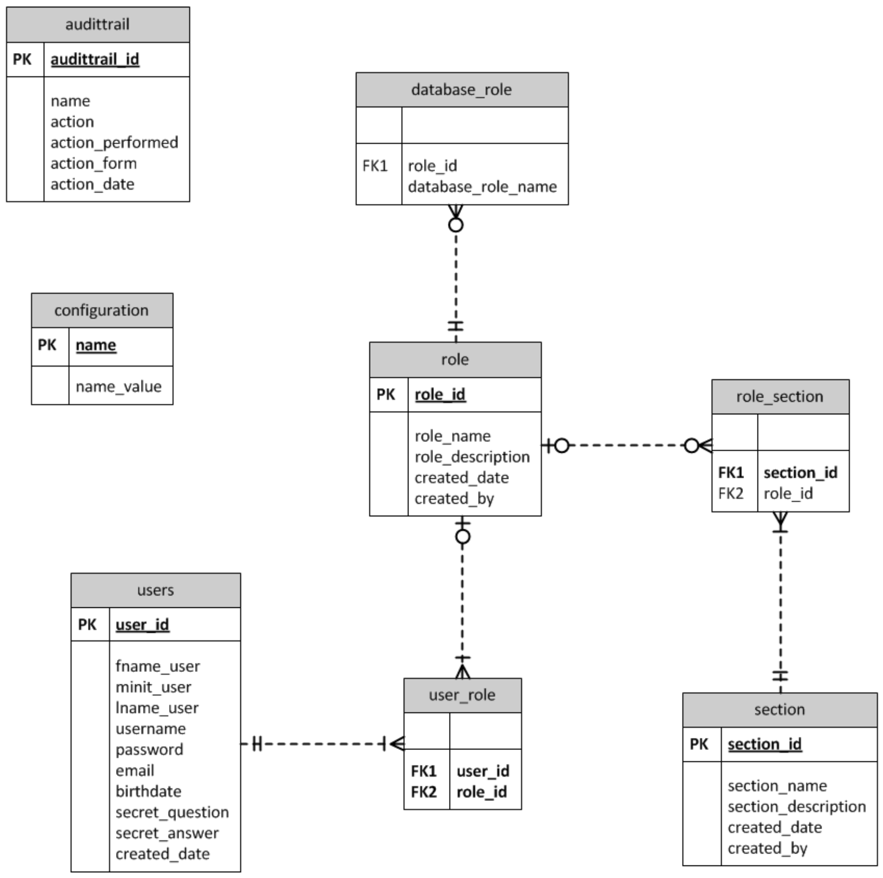


Figure 20: ERD of User Cluster

The Entity Relationship Diagram of Patient Cluster is shown in Figure 22 and Figure 21. The tables connected to patient entity are patient information, medical and social history, patient checklist, dental data, treatment plan, service rendered, consultations and findings, dental chart, caries status, recurrent status, restoration status and service needed. Tables colored in gray are built-in jBPM tables.

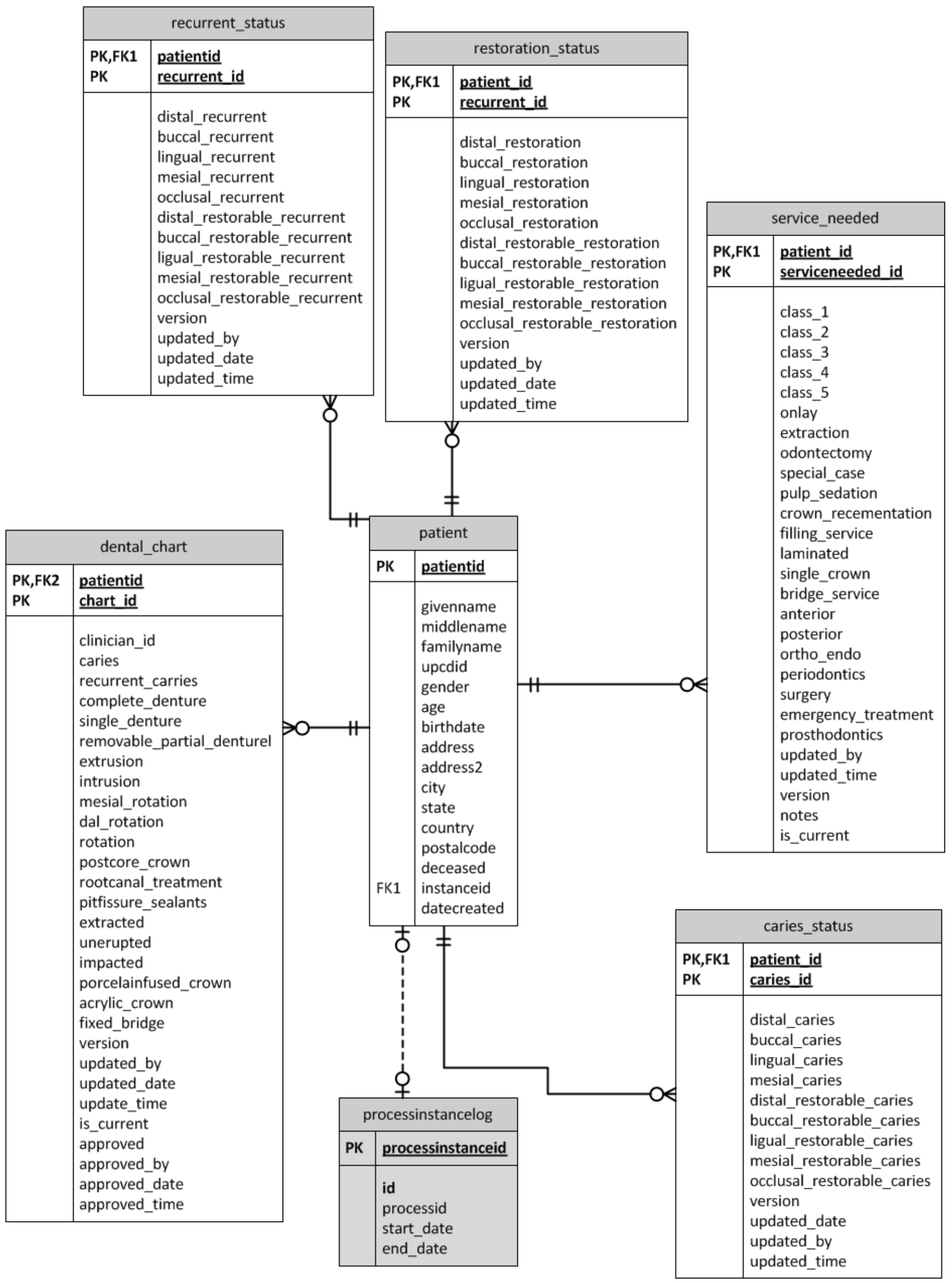


Figure 21: ERD of Patient Cluster

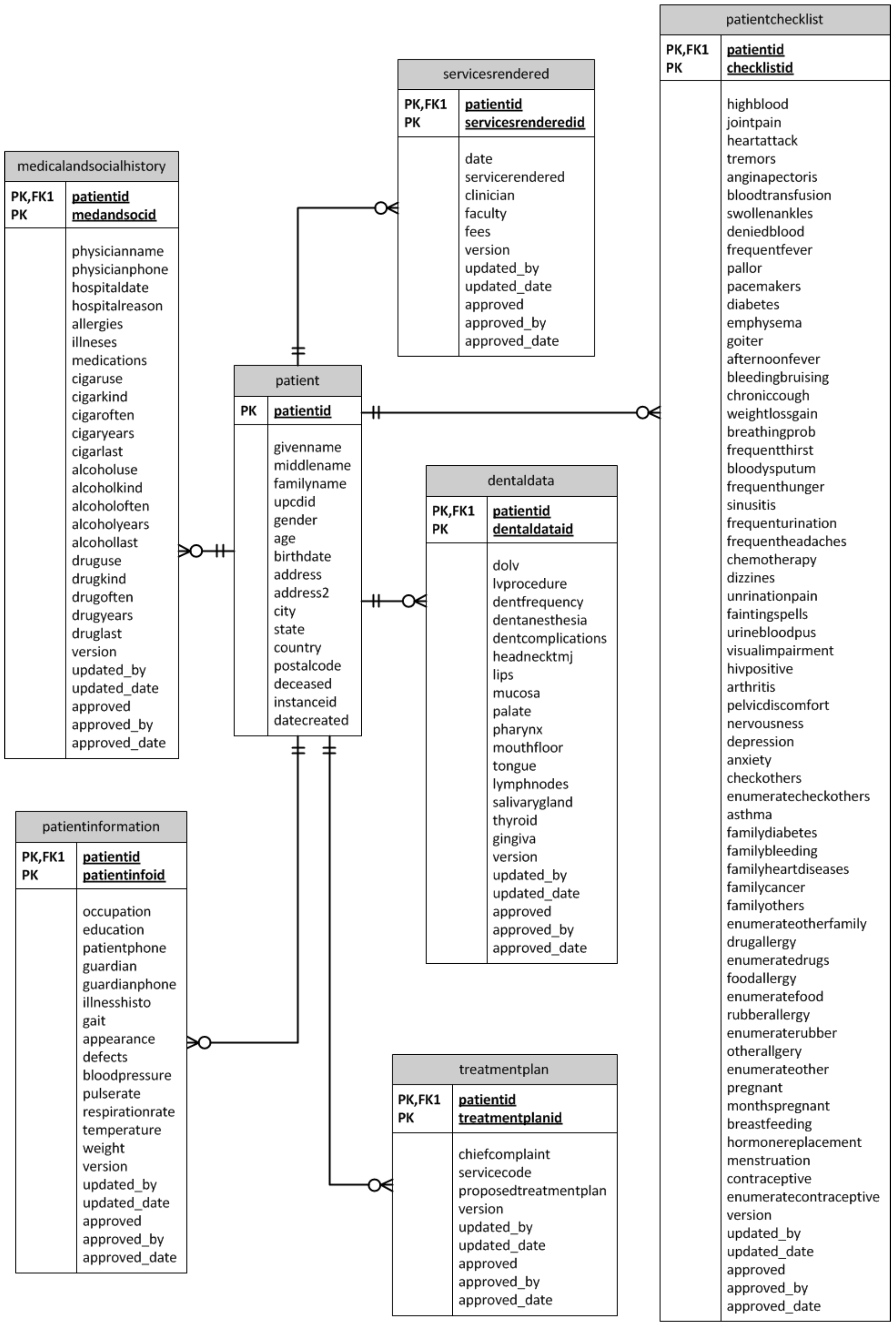


Figure 22: ERD of Patient Cluster

D. Data Dictionary

Attribute	Data Type	Description
user_id	bigserial	user identifier
fname_user	character varying(128)	first name of user
minit_user	character varying(10)	middle initial of user
lname_user	character varying(128)	last name of user
username	character varying(128)	username of user
password	character varying(128)	password of user
email	character varying(255)	email of user
secret_question	character varying(255)	question for forgotten password
secret_answer	character varying(255)	answer to question for forgotten password
created_date	character varying (128)	current date the user is created

Table 1: users Table

Attribute	Data Type	Description
section_id	bigserial	section identifier
section_name	character varying(50)	name of section
section_description	character varying(128)	description of section
created_by	character varying(128)	user who created the section
created_date	character varying (128)	current date the section is created

Table 2: section Table

Attribute	Data Type	Description
role_id	bigserial	role identifier
role_name	character varying(50)	name of role
role_description	character varying(128)	description of role
created_by	character varying(128)	user who created the role
created_date	character varying (128)	current date the role is created

Table 3: role Table

Attribute	Data Type	Description
section_id	integer(11)	section identifier the role is assigned
role_id	integer(11)	role identifier assigned to a section

Table 4: role_section Table

Attribute	Data Type	Description
user_id	integer(11)	user identifier the role is assigned
role_id	integer(11)	role identifier assigned to a user

Table 5: user_role Table

Attribute	Data Type	Description
role_id	integer	role identifier assigned to a role
database_role_name	character varying (100)	equivalent database name of a role

Table 6: database_role Table

Attribute	Data Type	Description
audittrail_id	bigserial	audittrail identifier
name	bytea	name of user who performed action in the system
action	bytea	action done in the system
action_performed	bytea	who or what the action was performed
action_form	bytea	what form the action was done
action_date	bytea	date the action was done

Table 7: auddittrail Table

Attribute	Data Type	Description
name	character varying (150)	name of the configuration setting
name_value	character varying (500)	value of the specified name

Table 8: configuration Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
givenname	bytea	patient given name
middlename	bytea	patient middle name
familyname	bytea	patient family name
upcdid	character varying(50)	patient UPCD identifier
gender	character varying(10)	patient gender
birthdate	bytea	patient birthdate
address	bytea	patient address
address2	bytea	patient address
city	bytea	patient address
state	bytea	patient address
country	bytea	patient address
postalcode	bytea	patient address
deceased	character varying(10)	is patient deceased
instanceid	bigint	patient current case id
datecreated	character varying(50)	date patient record is created

Table 9: Patient Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
patientinfoid	bigint	patient information identifier
occupation	character varying(50)	occupation of patient
education	character varying(50)	educational attainment of patient
patientphone	bytea	contact number of patient
guardian	bytea	guardian of patient
guardianphone	bytea	contact number of guardian of patient
illnesshisto	character varying(200)	present illness of patient
gait	character varying(200)	gait
appearance	character varying(200)	appearance
defects character	varying(200)	defects
bloodpressure	character varying(20)	blood pressure of patient
pulserate	character varying(20)	pulse rate of patient
respirationrate	character varying(5)	respiration rate of patient
temperature	character varying(5)	temperature of patient
weight	character varying(7)	weight of patient
version	integer	version of patient information record of patient
updated_by	character varying(100)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(50)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 10: PatientInformation Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
medandsocid	bigint	medical and social history identifier
physicianname	character varying(100)	physician of patient
physicianphone	character varying(100)	contact number of physician of patient
hospitaldate	character varying(100)	date of latest hospitalization of patient
hospitalreason	character varying(100)	reason for latest hospitalization
allergies	character varying(100)	allergies
illneses	character varying	illnesses
medications	character varying(100)	medications
childhood	character varying(100)	childhood diseases history
cigaruse	character varying(10)	Is the patient using or have used tobacco,cigarette?
cigarkind	character varying(100)	What kind does the patient smoke?
cigaroften	character varying(100)	How often does the patient smoke?
cigaryears	character varying(20)	How many years has the patient been smoking?
cigarlast	character varying(50)	If patient already stopped, how long since last used?
alcoholuse	character varying(10)	Does the patient drink alcoholic beverage?
alcoholkind	character varying(50)	What kind does the patient drink?
alcoholoften	character varying(20)	How often does the patient drink?
alcoholyears	character varying(5)	How many years has the patient been drinking?
alcohollast	character varying(50)	If patient already stopped, how long since last used?

Table 11: MedicalandSocialHistory Table

Attribute	Data Type	Description
druguse	character varying(10)	Has the patient used drugs for recreation purposes?
drugkind	character varying(50)	What kind of drug?
drugoften	character varying(20)	How often does the patient use drugs?
drugyears	character varying(5)	How many years has the patient been using?
druglast	character varying(50)	If patient already stopped, how long since last used?
version	integer	version of medical and social history record of patient
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date record is updated
approved	character varying(50)	approved status of patient
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date record is approved

Table 12: MedicalandSocialHistory Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
dentaldataid	bigint	dental data identifier
dolv	character varying(50)	date of last visit
lvprocedure	character varying(50)	last visit procedure
dentfrequency	character varying(25)	frequency of dental visit
dentanesthesia	character varying(25)	exposure and response to local anesthesia
dentcomplications	character varying(25)	complications during and or after dental procedure
headnecktmj	character varying(100)	head and neck TMJ
lips	character varying(100)	lips
mucosa	character varying(100)	mucosa
palate	character varying(100)	palate
pharynx	character varying(100)	pharynx
mouthfloor	character varying(100)	mouth floor
tongue	character varying(100)	tongue
lymphnodes	character varying(100)	lymphnodes
salivarygland	character varying(100)	salivary gland
thyroid	character varying(100)	thyroid
gingiva	character varying(100)	gingiva
version	integer	version of dental data record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 13: DentalData Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
treatmentplanid	bigint	treatment plan identifier
chiefcomplaint	character varying(35)	chief complaint of patient
servicecode	character varying(100)	service codes of treatment type
proposedtreatment	character varying(200)	proposed treatment
version	integer	version of treatment plan record
updated_by	character varying(100)	user who updated the record
updated_date	character varying(50)	date record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 14: TreatmentPlan Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
servicesrenderedid	bigint	services rendered identifier
date	character varying(50)	date when service is rendered by patient
servicerendered	character varying(100)	type of service rendered by patient
clinician	character varying(100)	clinician who performed the service
faculty	character varying(100)	faculty clinician who checked the work of the clinician
fees	character varying(20)	fees paid by patient
version	integer	version of service rendered
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date when record is approved

Table 15: ServicesRendered Table

Attribute	Data Type	Description
caries_id	bigint	caries identifier
patient_id	bigint	patient identifier
distal_caries	integer[]	distal surface with caries
buccal_caries	integer[]	buccal surface with caries
lingual_caries	integer[]	lingual surface with caries
mesial_caries	integer[]	mesial surface with caries
occlusal_caries	integer[]	occlusal surface with caries
distal_restorable_caries	character varying[]	variation of distal surface with caries
buccal_restorable_caries	character varying[]	variation of buccal surface with caries
lingual_restorable_caries	character varying[]	variation of lingual surface with caries
mesial_restorable_caries	character varying[]	variation of mesial surface with caries
occlusal_restorable_caries	character varying[]	variation of occlusal surface with caries
version	bigint	version of caries status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 16: Caries_Status Table

Attribute	Data Type	Description
recurrent_id	bigint	recurrent caries identifier
patient_id	bigint	patient identifier
distal_recurrent	integer[]	distal surface with recurrent caries
buccal_recurrent	integer[]	buccal surface with recurrent caries
lingual_recurrent	integer[]	lingual surface with recurrent caries
mesial_recurrent	integer[]	mesial surface with recurrent caries
occlusal_recurrent	integer[]	occlusal surface with recurrent caries
distal_restorable_recurrent	character varying[]	variation of distal surface with recurrent caries
buccal_restorable_recurrent	character varying[]	variation of buccal surface with recurrent caries
lingual_restorable_recurrent	character varying[]	variation of lingual surface with recurrent caries
mesial_restorable_recurrent	character varying[]	variation of mesial surface with recurrent caries
occlusal_restorable_recurrent	character varying[]	variation of occlusal surface with recurrent caries
version	bigint	version of recurrent caries status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 17: Recurrent_Status Table

Attribute	Data Type	Description
restoration_id	bigint	restoration identifier
patient_id	bigint	patient identifier
distal_restoration	integer[]	distal surface with restoration
buccal_restoration	integer[]	buccal surface with restoration
lingual_restoration	integer[]	lingual surface with restoration
mesial_restoration	integer[]	mesial surface with restoration
occlusal_restoration	integer[]	occlusal surface with restoration
distal_restorable_restoration	character varying[]	variation of distal surface with restoration
buccal_restorable_restoration	character varying[]	variation of buccal surface with restoration
lingual_restorable_restoration	character varying[]	variation of lingual surface with restoration
mesial_restorable_restoration	character varying[]	variation of mesial surface with restoration
occlusal_restorable_restoration	character varying[]	variation of occlusal surface with restoration
version	bigint	version of restoration status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 18: Restoration_Status Table

Attribute	Data Type	Description
serviceneeded_id	bigint	service needed identifier
patient_id	bigint	patient identifier
class_1	integer[]	tooth numbers that need class 1 treatment
class_2	integer[]	tooth numbers that need class 2 treatment
class_3	integer[]	tooth numbers that need class 3 treatment
class_4	integer[]	tooth numbers that need class 4 treatment
class_5	integer[]	tooth numbers that need class 1 treatment
onlay	integer[]	tooth numbers that need onlay treatment
extraction	integer[]	tooth numbers that need extraction
odontectomy	integer[]	tooth numbers that odontectomy
special_case	integer[]	tooth numbers that are special cases
pulp_sedation	integer[]	pulp sedation treatment
crown_recementation	integer[]	recementation of crowns
filling_service	integer[]	temporary fillings
laminated	integer[]	tooth numbers that need laminated fixed partial denture
single_crown	integer[]	tooth numbers that need single crown fixed partial denture
bridge_service	integer[]	tooth numbers that need bridge fixed partial denture
anterior	integer[]	anterior endodontics
posterior	integer[]	anterior endodontics
ortho_endo	integer[]	other endodontics

Table 19: Service_Needed Table

Attribute	Data Type	Description
periodontics	character varying(10)	management of periodontal disease
surgery	character varying(50)	surgery
emergency_treatment	character varying(50)	emergency treatment
prosthodontics	character varying(50)	prosthodontics
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated
version	integer	version of service needed record
notes	character varying(500)	additional notes
is_current	character varying(10)	is the record the latest version?

Table 20: Service_Needed Table

Attribute	Data Type	Description
dental_chart_id	bigint	dental chart identifier
patient_id	bigint	patient identifier
clinician_id	bigint	clinician of patient
caries	integer[]	tooth numbers with caries
recurrent_caries	integer[]	tooth numbers with recurrent caries
restoration	integer[]	tooth numbers with restoration
removable_partial_denture	integer[]	removable partial denture
extrusion	integer[]	tooth numbers with extrusion
intrusion	integer[]	tooth numbers with intrusion
mesial_rotation	integer[]	tooth numbers with mesial rotation
distal_rotation	integer[]	tooth numbers with distal rotation
rotation	integer[]	tooth numbers with rotation
postcore_crown	integer[]	tooth numbers with post core crown
rootcanal_treatment	integer[]	tooth numbers with root canal treatment
pitfissure_sealants	integer[]	tooth numbers with pit and fissure sealant
extracted	integer[]	extracted teeth
missing	integer[]	missing teeth
unerupted	integer[]	unerupted teeth
impacted	integer[]	impacted teeth
porcelain_crown	integer[]	tooth numbers with porcelain crown
acrylic_crown	integer[]	tooth numbers with acrylic crown

Table 21: Dental_Chart Table

V. Architecture

A. System Architecture

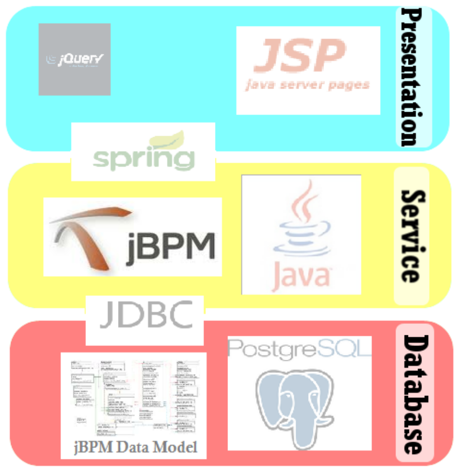


Figure 23: System Architecture of DentISt

Figure 23 shows the system architecture of DentISt.The system has three layers-presentation, service and database. The presentation layer has the web interface that is coded in JSP and uses JQuery as its Javascript framework. The presentation and service layers are connected using the Spring framework. The service layer uses jBPM 5.2 to manage the workflow and other services. It was developed with the Java programming language and uses JDBC to connect to PostgreSQL at the database layer.

B. Technical Architecture

DentISt 3.0 is compatible for the following operating systems:

- Ubuntu Linux 12.04.2 or Redhat Linux

It will also use the following software:

- Apache Tomcat 6.25

- PostgreSQL 8 or 9

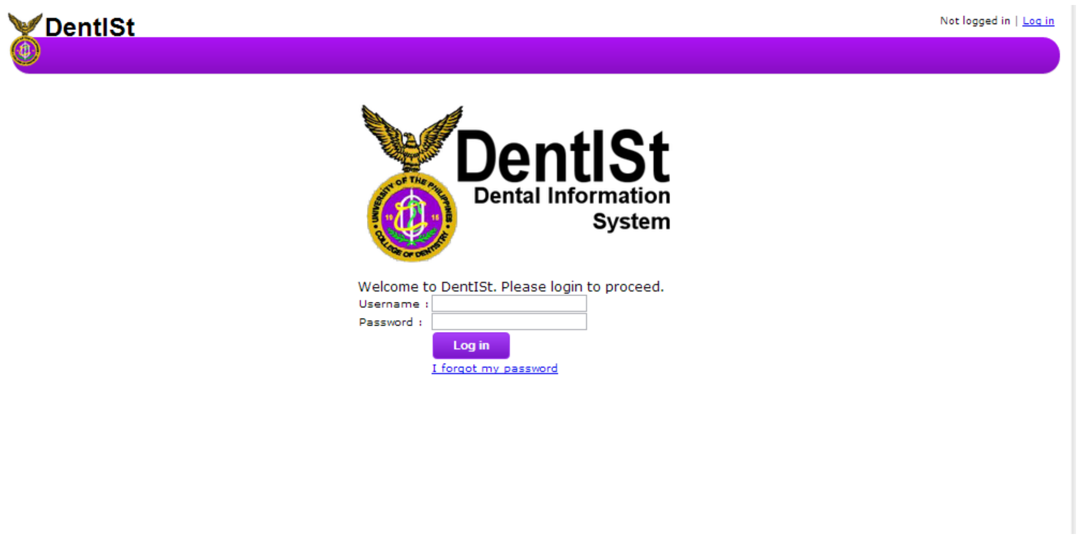
The client side must have any of the following compatible web browsers:

- Mozilla Firefox 16.0.1
- Google Chrome 22.0.1229.94
- Safari 5.1.7
- Opera 12.02

VI. Results

A. System Functionalities Screenshots

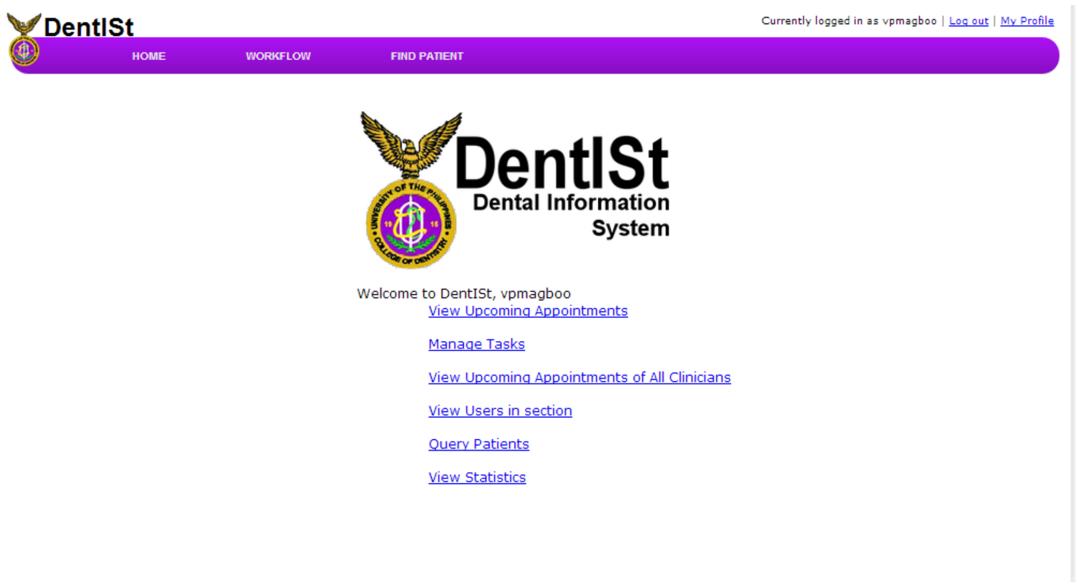
The login page of DentISt 3.0(shown in Figure 24) is displayed once the site is opened



Welcome to DentISt. Please login to proceed.

Figure 24: Login Page of DentISt 3.0

If the user has sucessfully logged in, he will be able to see navigation menus and links of DentISt 3.0 as shown in Figure 25. Workflow tab is for editing processes, tasks and forms of the system while Find Patient tab is for patient and tasks management. Different links such as view appointments, manage tasks, view users in the section, query patient and view statictics are shown in home page as well.


DentISt
HOME
WORKFLOW
FIND PATIENT
Currently logged in as vpmagboo | Log out | My Profile
DentISt
Dental Information System

Welcome to DentISt, vpmagboo

View Upcoming Appointments

Manage Tasks

View Upcoming Appointments of All Clinicians

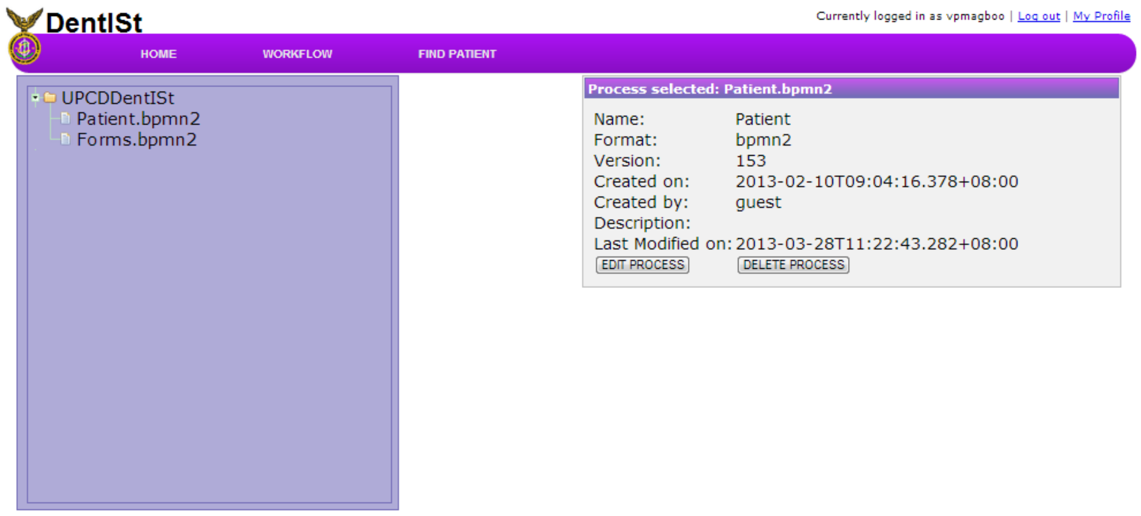
View Users in section

Query Patients

View Statistics

Figure 25: Home Page of DentISt 3.0

In the workflow dashboard of DentISt 3.0 shown in Figure 26, the workflow administrator can view the list of processes (Figure 28), add new workflow processes (Figure 27) and delete workflow processes (Figure 29). He can also add and edit the forms of the tasks (Figure 30). The documentation on how to use jBPM is found on jBPM website [6].



Process selected: Patient.bpmn2

Name:	Patient
Format:	bpmn2
Version:	153
Created on:	2013-02-10T09:04:16.378+08:00
Created by:	guest
Description:	
Last Modified on:	2013-03-28T11:22:43.282+08:00

Figure 26: Workflow Dashboard

[6]`http://www.jboss.org/jbpm/documentation`

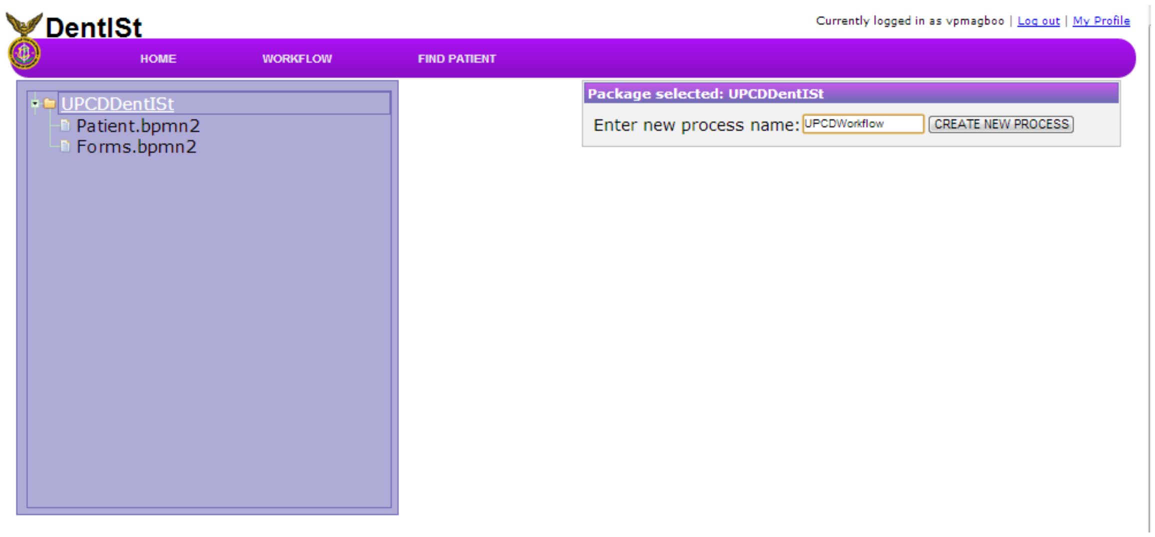

DentISt
Currently logged in as vpmagboo | Log out | My Profile
HOME
WORKFLOW
FIND PATIENT
UPCDDentISt
Patient.bpmn2
Forms.bpmn2
Package selected: UPCDDentISt
Enter new process name: UPCDWorkflow
CREATE NEW PROCESS

Figure 27: Add Workflow Process

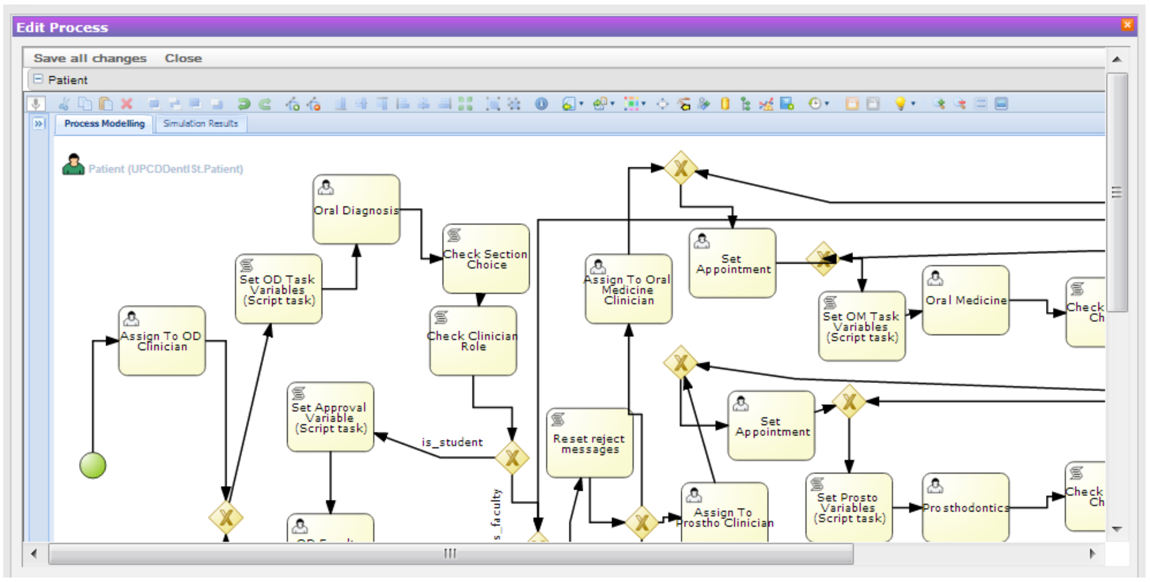
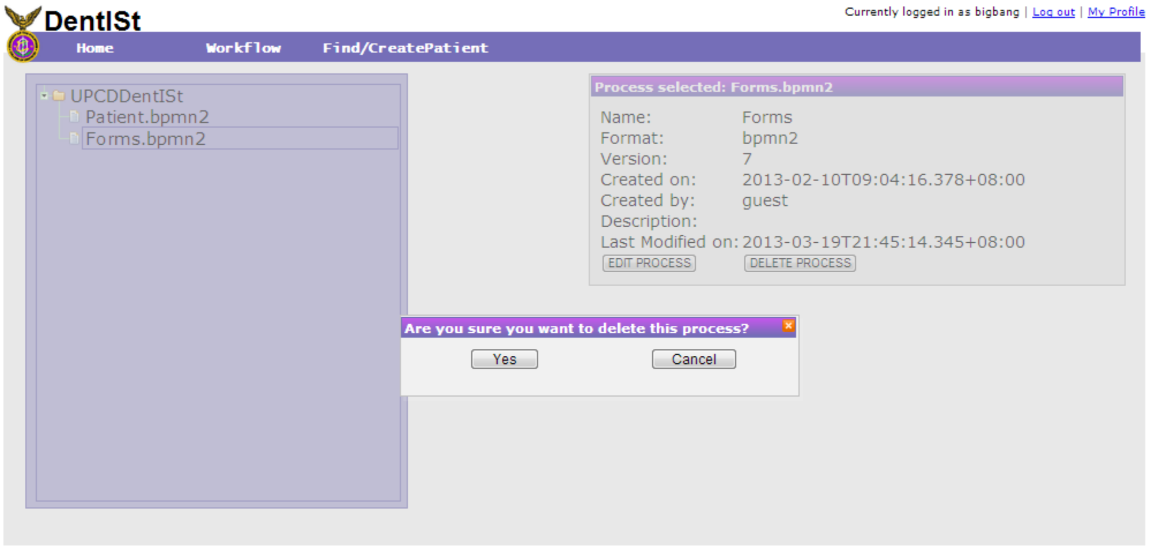
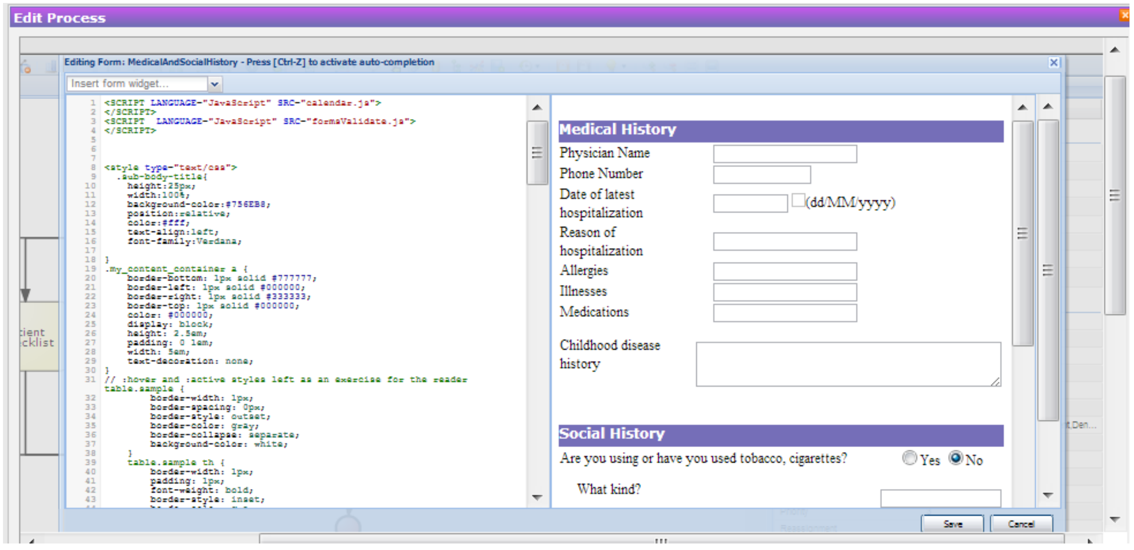


Figure 28: Edit Workflow Process


DentISt
Currently logged in as bigbang | Log out | My Profile
Home
Workflow
Find/CreatePatient
UPCDDentISt
Patient.bpmn2
Forms.bpmn2
Process selected: Forms.bpmn2
EDIT PROCESS
DELETE PROCESS
Are you sure you want to delete this process?
Yes
Cancel

Name:	Forms
Format:	bpmn2
Version:	7
Created on:	2013-02-10T09:04:16.378+08:00
Created by:	guest
Description:	
Last Modified on:	2013-03-19T21:45:14.345+08:00

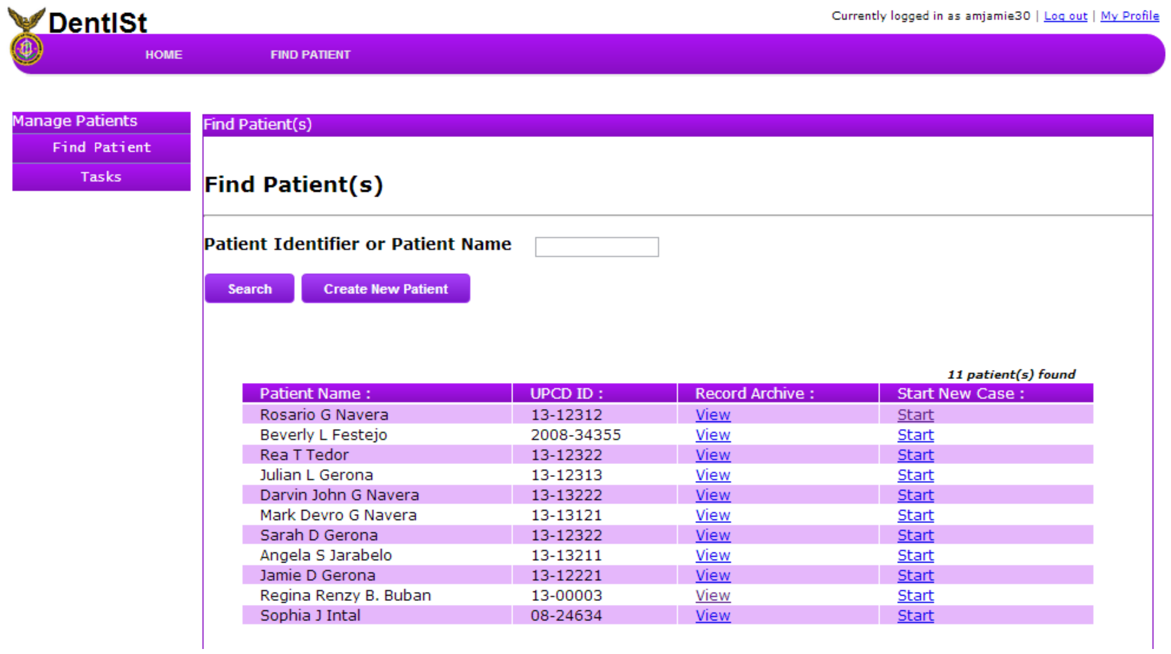
Figure 29: Delete Workflow Process


Edit Process
Editing Form: MedicalAndSocialHistory - Press [Ctrl-Z] to activate auto-completion
Insert form widget...
Medical History
Physician Name
Phone Number
Date of latest hospitalization
(dd/MM/yyyy)
Reason of hospitalization
Allergies
Illnesses
Medications
Childhood disease history
Social History
Are you using or have you used tobacco, cigarettes?
Yes [✓ No]
What kind?
Save
Cancel

```
<SCRIPT LANGUAGE="JavaScript" SRC="calendar.js">
</SCRIPT>
<SCRIPT  LANGUAGE="JavaScript" SRC="formsValidate.js">
</SCRIPT>



<style type="text/css">
  .sub-body-title{
    height:25px;
    width:100%;
    background-color:#756EB8;
    position:relative;
    color:#fff;
    text-align:left;
    font-family:Verdana;

}
.my_content_container a {
    border-bottom: 1px solid #777777;
    border-left: 1px solid #000000;
    border-right: 1px solid #333333;
    border-top: 1px solid #000000;
    color: #000000;
    display: block;
    height: 2.5em;
    padding: 0 1em;
    width: 5em;
    text-decoration: none;
}
// :hover and :active styles left as an exercise for the reader
table.sample {
        border-width: 1px;
        border-spacing: 0px;
        border-style: outset;
        border-color: gray;
        border-collapse: separate;
        background-color: white;
    }
    table.sample th {
        border-width: 1px;
        padding: 1px;
        font-weight: bold;
        border-style: inset;
```

Figure 30: Edit Task Forms in Workflow Designer

To find or create a patient, users must click the Find Patient menu. Only clinicians in Oral Diagnosis can create a patient and start a new case for the patient while all other clinicians and faculty clinicians can search a patient by name. Figure 31 shows how to search for a patient while Figure 32 show how to add a patient.



Find Patient(s)

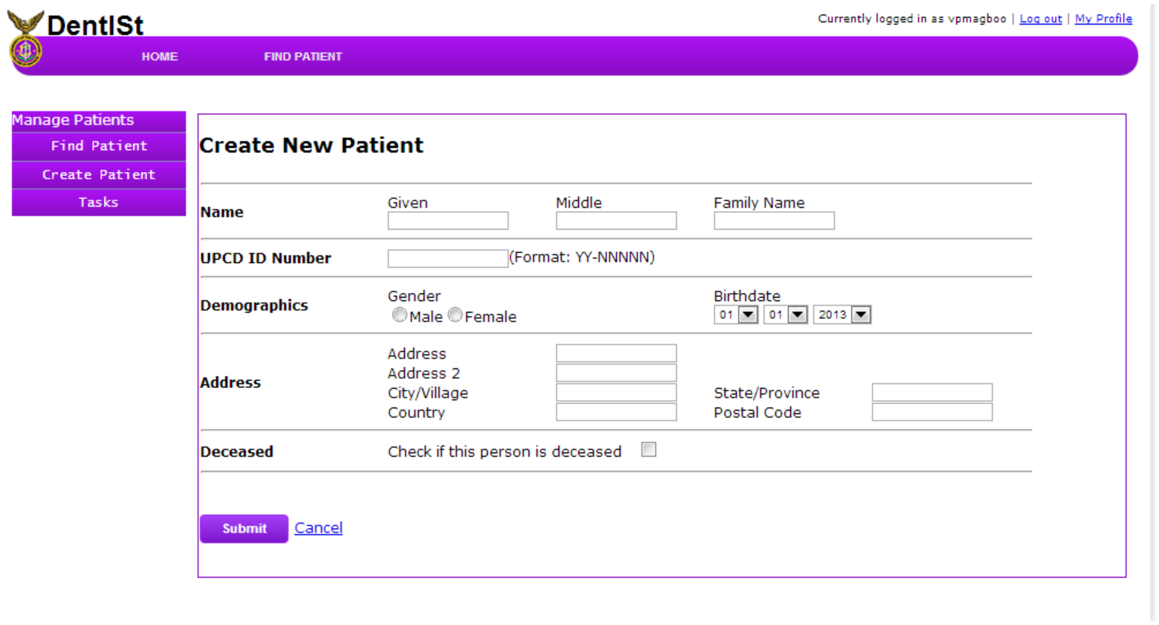
Patient Identifier or Patient Name

11 patient(s) found

Patient Name :	UPCD ID :	Record Archive :	Start New Case :
Rosario G Navera	13-12312	View	Start
Beverly L Festejo	2008-34355	View	Start
Rea T Tedor	13-12322	View	Start
Julian L Gerona	13-12313	View	Start
Darvin John G Navera	13-13222	View	Start
Mark Devro G Navera	13-13121	View	Start
Sarah D Gerona	13-12322	View	Start
Angela S Jarabelo	13-13211	View	Start
Jamie D Gerona	13-12221	View	Start
Regina Renzy B. Buban	13-00003	View	Start
Sophia J Intal	08-24634	View	Start

Figure 31: Find Patients

In creating a new patient, the identifier to use is UPCD Identication Number with the format NN-NNNNN, the first two is the year the patient is registered, while the last five numbers are any numbers. The UPCD ID number is automatically generated by the system.

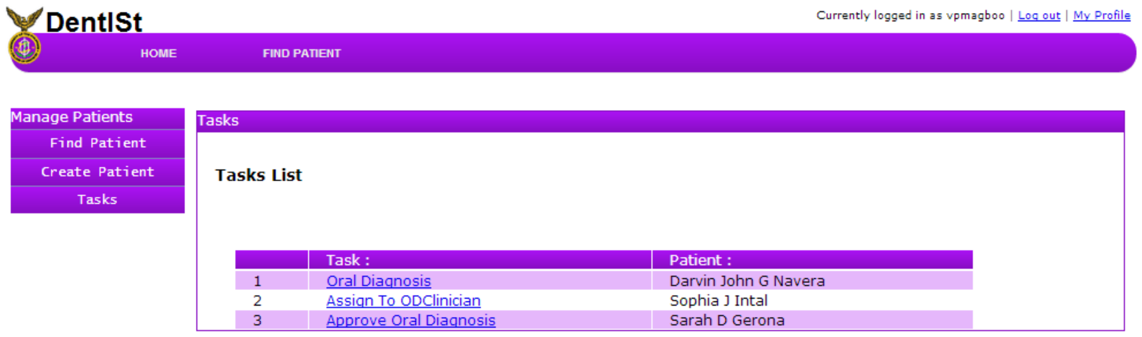


Create New Patient

Name	Given	Middle	Family Name	
UPCD ID Number	(Format: YY-NNNNN)			
Demographics	Gender: Male / Female		Birthdate: 01 / 01 / 2013	
Address	Address			
	Address 2			
	City/Village		State/Province	
	Country		Postal Code	
Deceased	Check if this person is deceased			

Figure 32: Create New Patient

After the patient is created, a new case is started for the patient and clinician is directed to the task list shown in Figure 33. The clinician can then start working on the patient treatment workflow. The patient is first assigned to the oral diagnosis clinician.

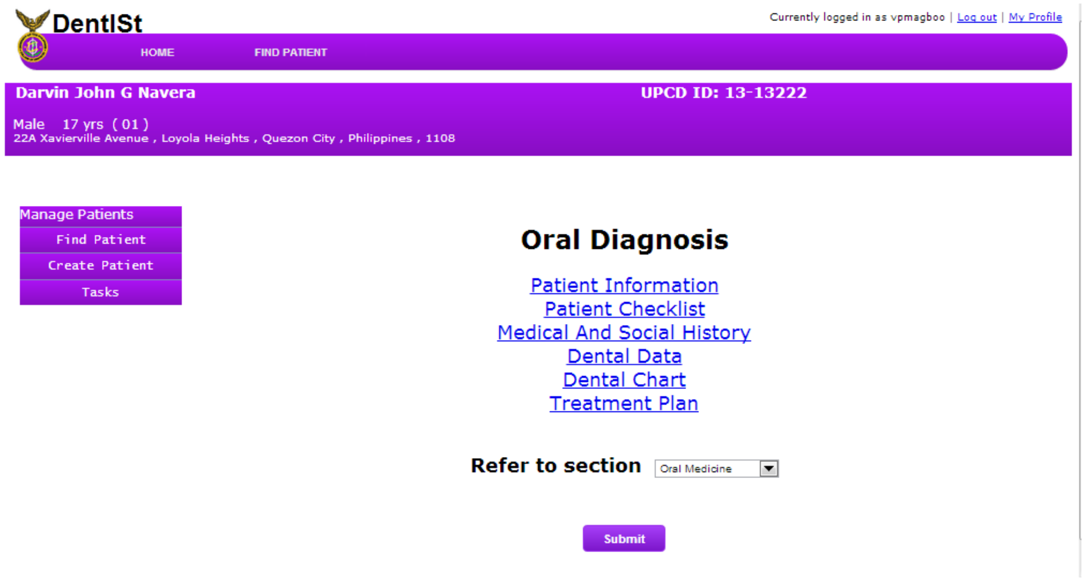


Tasks List

	Task :	Patient :
1	Oral Diagnosis	Darvin John G Navera
2	Assign To ODClinician	Sophia J Intal
3	Approve Oral Diagnosis	Sarah D Gerona

Figure 33: View List of Tasks

The assigned oral diagnosis clinician can then edit the oral diagnosis forms and refer the patient to a UPCD section. The form dashboard as shown in Figure 34 consists of six forms namely Patient Information, Patient Checklist, Medical and Social History, Dental Data, Dental Chart and Treatment Plan.



Darvin John G Navera UPCD ID: 13-13222
Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Oral Diagnosis

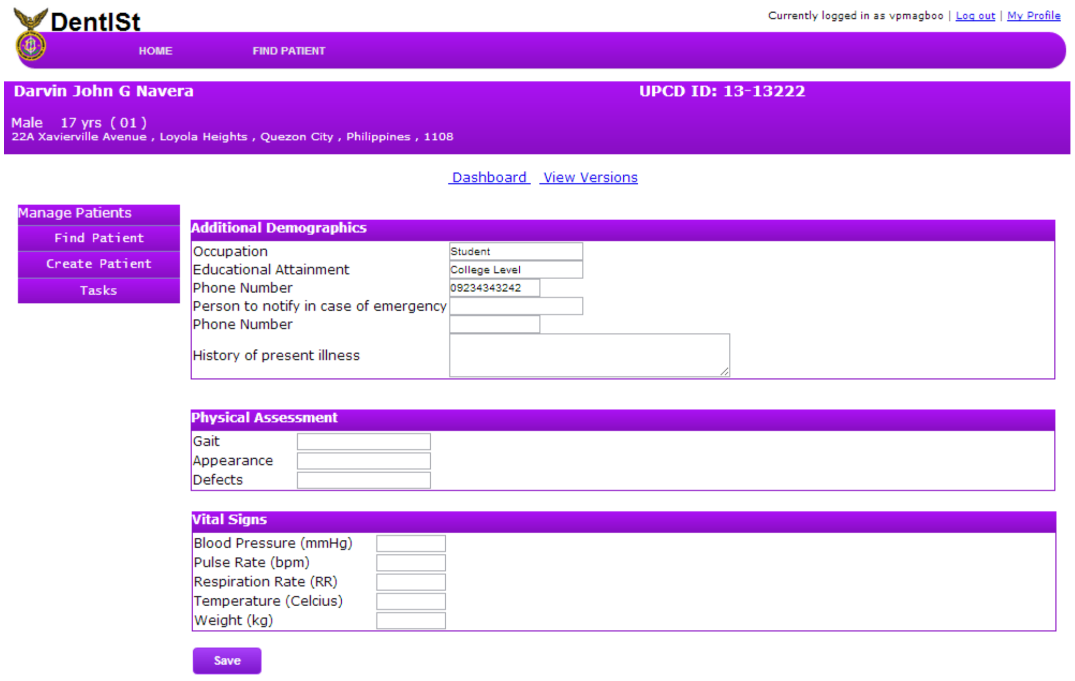
Patient Information
Patient Checklist
Medical And Social History
Dental Data
Dental Chart
Treatment Plan

Refer to section: Oral Medicine

Submit

Figure 34: Form Dashboard of Section Tasks

In the Patient Information link, clinicians can edit basic information, physical assessment and vital signs information of the patient. The form is shown in Figure 35. In the Patient Checklist link (shown in Figure 36), the clinicians can edit the patient checklist form.



Darvin John G Navera

UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Additional Demographics

Occupation	Student
Educational Attainment	College Level
Phone Number	09234343242
Person to notify in case of emergency	
Phone Number	
History of present illness	

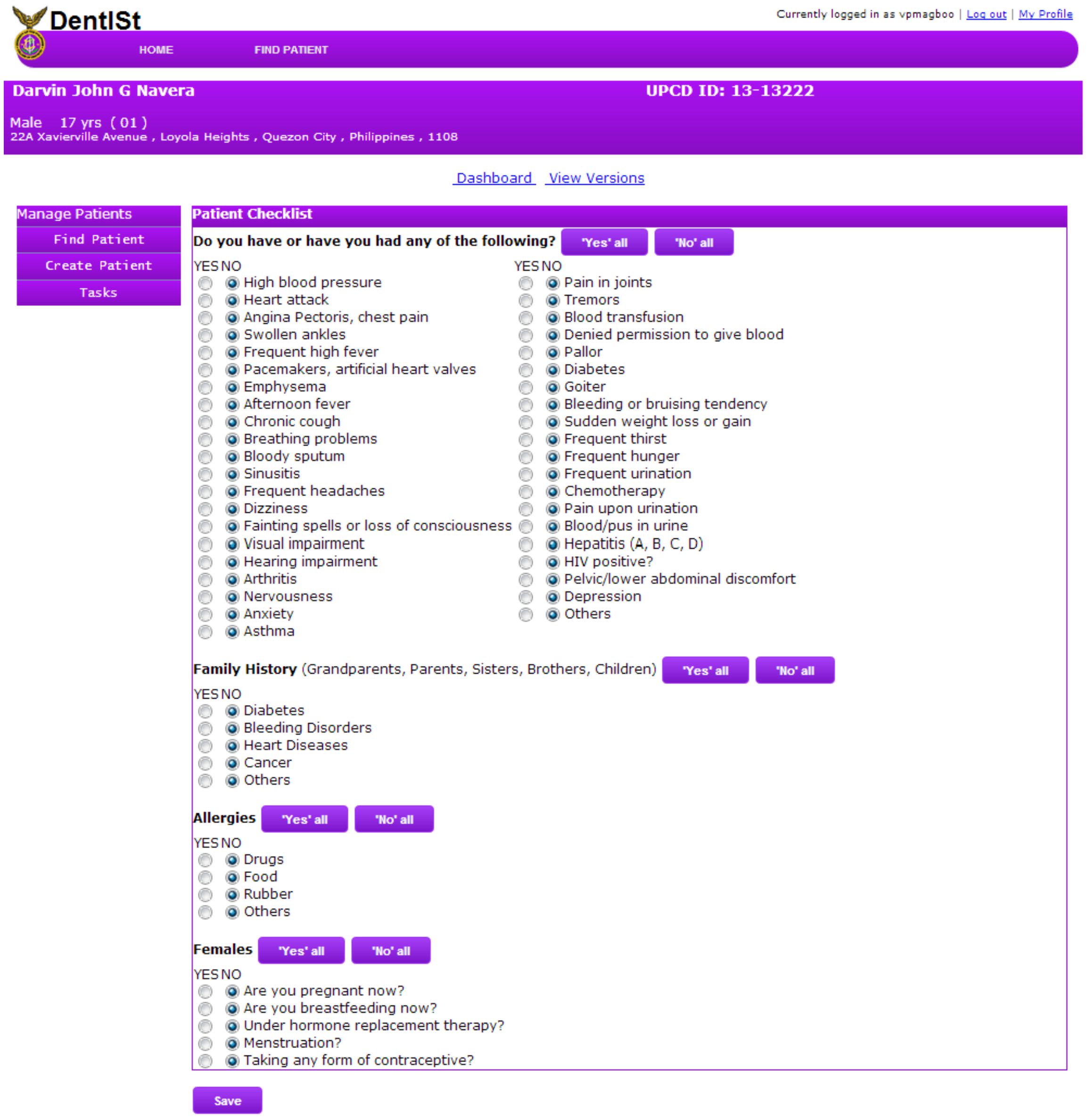
Physical Assessment

Gait	
Appearance	
Defects	

Vital Signs

Blood Pressure (mmHg)	
Pulse Rate (bpm)	
Respiration Rate (RR)	
Temperature (Celcius)	
Weight (kg)	

Figure 35: Edit Patient Information



Currently logged in as vpmagboo | Log out | My Profile

Darvin John G Navera **UPCD ID: 13-13222**

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Dashboard View Versions

Patient Checklist

Do you have or have you had any of the following? 'Yes' all 'No' all

YES	NO		YES	NO	
○	●	High blood pressure	○	●	Pain in joints
○	●	Heart attack	○	●	Tremors
○	●	Angina Pectoris, chest pain	○	●	Blood transfusion
○	●	Swollen ankles	○	●	Denied permission to give blood
○	●	Frequent high fever	○	●	Pallor
○	●	Pacemakers, artificial heart valves	○	●	Diabetes
○	●	Emphysema	○	●	Goiter
○	●	Afternoon fever	○	●	Bleeding or bruising tendency
○	●	Chronic cough	○	●	Sudden weight loss or gain
○	●	Breathing problems	○	●	Frequent thirst
○	●	Bloody sputum	○	●	Frequent hunger
○	●	Sinusitis	○	●	Frequent urination
○	●	Frequent headaches	○	●	Chemotherapy
○	●	Dizziness	○	●	Pain upon urination
○	●	Fainting spells or loss of consciousness	○	●	Blood/pus in urine
○	●	Visual impairment	○	●	Hepatitis (A, B, C, D)
○	●	Hearing impairment	○	●	HIV positive?
○	●	Arthritis	○	●	Pelvic/lower abdominal discomfort
○	●	Nervousness	○	●	Depression
○	●	Anxiety	○	●	Others
○	●	Asthma			

Family History (Grandparents, Parents, Sisters, Brothers, Children) 'Yes' all 'No' all

YES	NO	
○	●	Diabetes
○	●	Bleeding Disorders
○	●	Heart Diseases
○	●	Cancer
○	●	Others

Allergies 'Yes' all 'No' all

YES	NO	
○	●	Drugs
○	●	Food
○	●	Rubber
○	●	Others

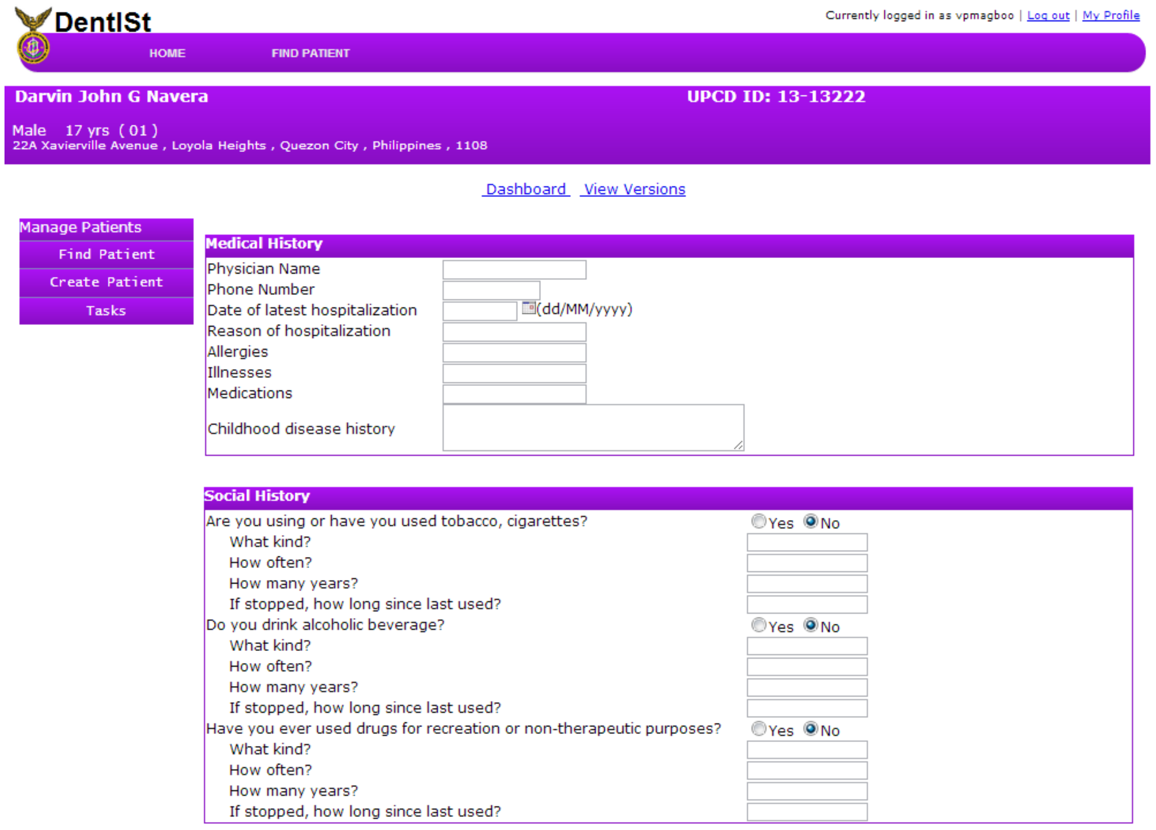
Females 'Yes' all 'No' all

YES	NO	
○	●	Are you pregnant now?
○	●	Are you breastfeeding now?
○	●	Under hormone replacement therapy?
○	●	Menstruation?
○	●	Taking any form of contraceptive?

Save

Figure 36: Edit Patient Checklist

In the Medical and Social History link, medical history and social history forms are found. To update these data, clinicians need to fill out the forms shown in Figure 37. In the Dental Data and Treatment Plan link, the dental data form and treatment plan form can be edited respectively.



Darvin John G Navera UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

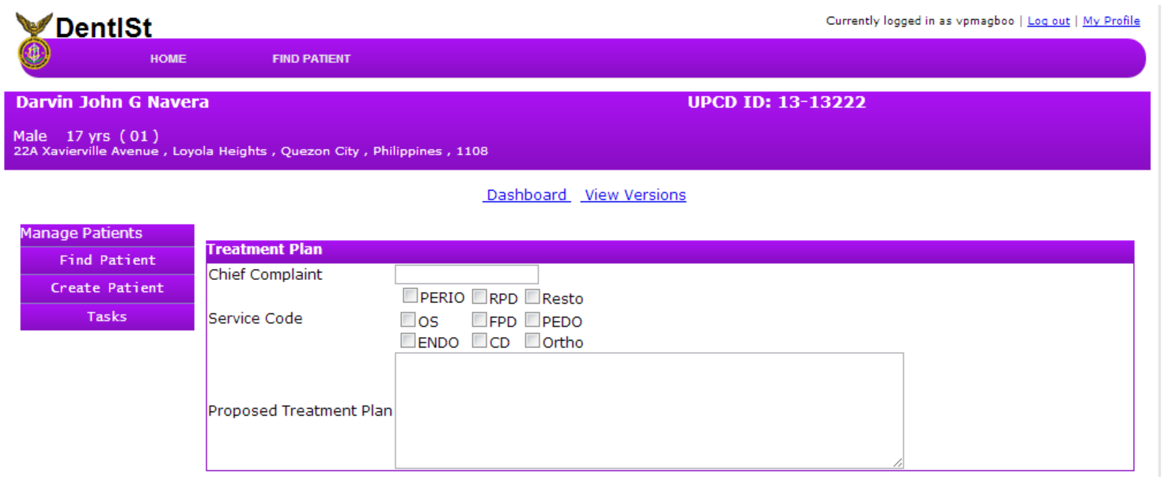
Medical History

Physician Name	
Phone Number	
Date of latest hospitalization	(dd/MM/yyyy)
Reason of hospitalization	
Allergies	
Illnesses	
Medications	
Childhood disease history	

Social History

Are you using or have you used tobacco, cigarettes?	Yes [✓ No]
What kind?	
How often?	
How many years?	
If stopped, how long since last used?	
Do you drink alcoholic beverage?	Yes [✓ No]
What kind?	
How often?	
How many years?	
If stopped, how long since last used?	
Have you ever used drugs for recreation or non-therapeutic purposes?	Yes [✓ No]
What kind?	
How often?	
How many years?	
If stopped, how long since last used?	

Figure 37: Edit Medical and Social History


Currently logged in as vpmagboo | Log out | My Profile

DentISt

HOME FIND PATIENT

Darvin John G Navera UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

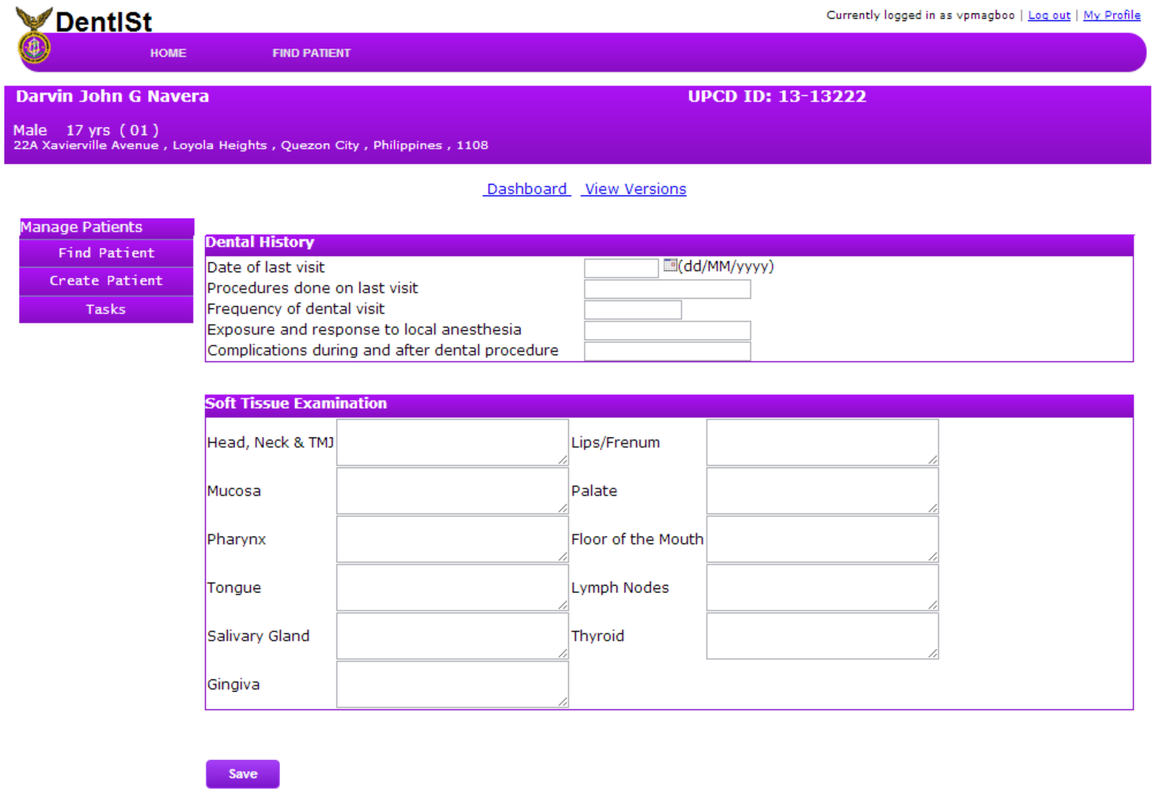
Dashboard View Versions

Manage Patients
Find Patient
Create Patient
Tasks

Treatment Plan

Chief Complaint	
Service Code	☐PERIO ☐RPD ☐Resto ☐OS ☐FPD ☐PEDO ☐ENDO ☐CD ☐Ortho
Proposed Treatment Plan	

Figure 38: Edit Treatment Plan


Currently logged in as vpmagboo | Log out | My Profile

DentISt

HOME FIND PATIENT

Darvin John G Navera UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Dashboard View Versions

Manage Patients
Find Patient
Create Patient
Tasks

Dental History

Date of last visit	(dd/MM/yyyy)
Procedures done on last visit	
Frequency of dental visit	
Exposure and response to local anesthesia	
Complications during and after dental procedure	

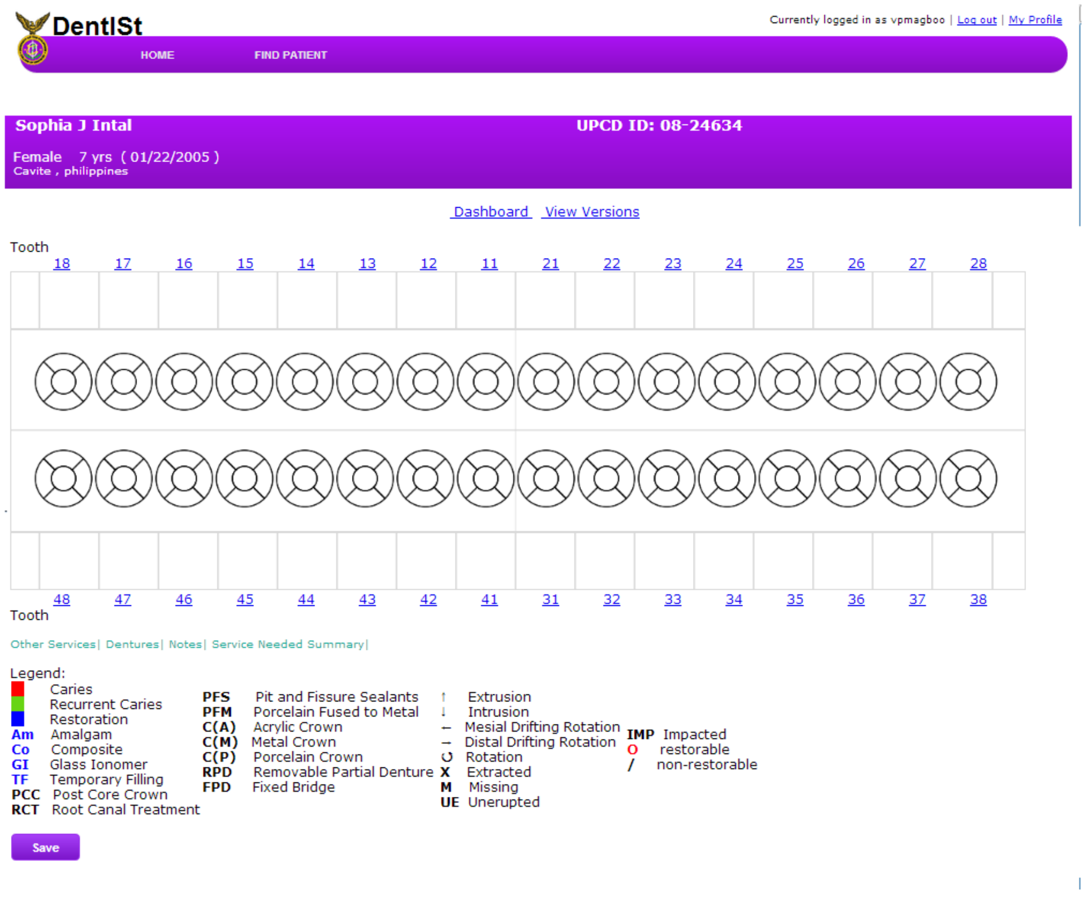
Soft Tissue Examination

Head, Neck & TMJ		Lips/Frenum	
Mucosa		Palate	
Pharynx		Floor of the Mouth	
Tongue		Lymph Nodes	
Salivary Gland		Thyroid	
Gingiva			

Save

Figure 39: Edit Dental Data

The dental chart is found in the Dental Chart link as shown in Figure 40


DentISt

Currently logged in as vpmagboo | Log out | My Profile

HOME FIND PATIENT

Sophia J Intal **UPCD ID: 08-24634**

Female 7 yrs (01/22/2005)
Cavite , philippines

Dashboard View Versions

Tooth
18 17 16 15 14 13 12 11 21 22 23 24 25 26 27 28

48 47 46 45 44 43 42 41 31 32 33 34 35 36 37 38
Tooth

Other Services| Dentures| Notes| Service Needed Summary|

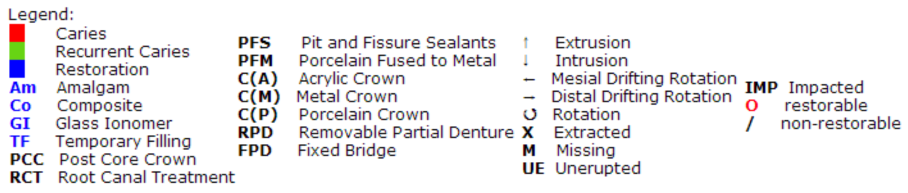
Legend:

🟥	Caries	PFS	Pit and Fissure Sealants	↑	Extrusion	IMP	Impacted
🟩	Recurrent Caries	PFM	Porcelain Fused to Metal	↓	Intrusion	O	restorable
🟦	Restoration	C(A)	Acrylic Crown	–	Mesial Drifting Rotation	/	non-restorable
Am	Amalgam	C(M)	Metal Crown	–	Distal Drifting Rotation		
Co	Composite	C(P)	Porcelain Crown	↺	Rotation		
GI	Glass Ionomer	RPD	Removable Partial Denture	X	Extracted		
TF	Temporary Filling	FPD	Fixed Bridge	M	Missing		
PCC	Post Core Crown			UE	Unerupted		
RCT	Root Canal Treatment						

Save

Figure 40: Dental Chart

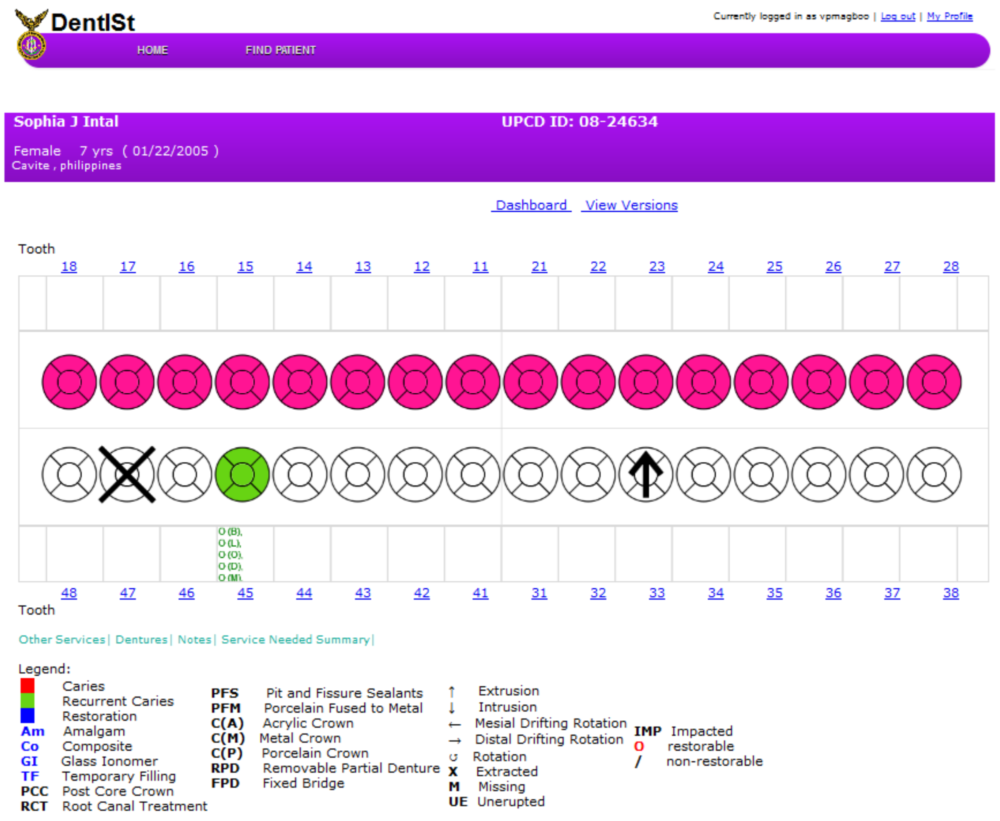
The legend per graphical representation as shown in Figure 41 is always present below the dental chart for easier viewing. The dental chart has three colors. Red represents caries, green represents recurrent caries and blue for restoration. Black is for whole tooth status.


Legend:

🟥	Caries	PFS	Pit and Fissure Sealants	↑	Extrusion	IMP	Impacted
🟩	Recurrent Caries	PFM	Porcelain Fused to Metal	↓	Intrusion	O	restorable
🟦	Restoration	C(A)	Acrylic Crown	–	Mesial Drifting Rotation	/	non-restorable
Am	Amalgam	C(M)	Metal Crown	–	Distal Drifting Rotation		
Co	Composite	C(P)	Porcelain Crown	↺	Rotation		
GI	Glass Ionomer	RPD	Removable Partial Denture	X	Extracted		
TF	Temporary Filling	FPD	Fixed Bridge	M	Missing		
PCC	Post Core Crown			UE	Unerupted		
RCT	Root Canal Treatment						

Figure 41: Legend - Dental Chart

The assigned clinician can edit the dental chart by clicking on the tooth numbers and adding the necessary conditions and services needed. The summarized services needed can also be viewed in the dental chart. Figures 42, 43, 44, 45, 46, 47, 48



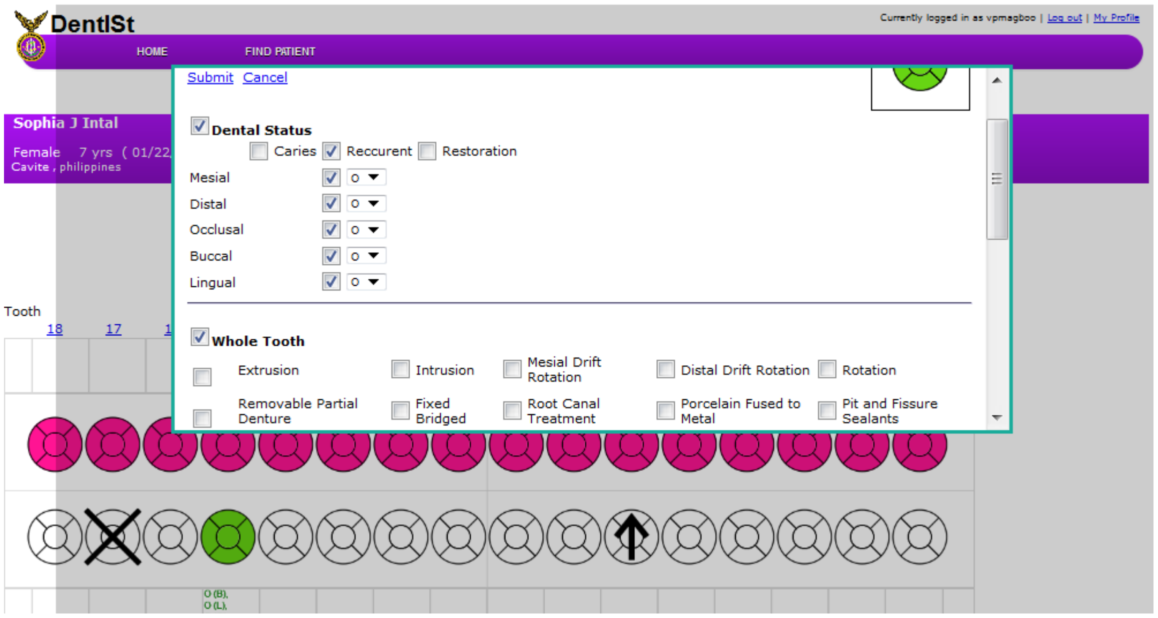
Sophia J Intal **UPCD ID: 08-24634**

Female 7 yrs (01/22/2005)
Cavite , philippines

Legend:

🟥	Caries	**PFS**	Pit and Fissure Sealants	↑	Extrusion	**IMP**	Impacted
🟩	Recurrent Caries	**PFM**	Porcelain Fused to Metal	↓	Intrusion	**O**	restorable
🟦	Restoration	**C(A)**	Acrylic Crown	←	Mesial Drifting Rotation	**/**	non-restorable
Am	Amalgam	**C(M)**	Metal Crown	→	Distal Drifting Rotation		
Co	Composite	**C(P)**	Porcelain Crown	↺	Rotation		
GI	Glass Ionomer	**RPD**	Removable Partial Denture	**X**	Extracted		
TF	Temporary Filling	**FPD**	Fixed Bridge	**M**	Missing		
PCC	Post Core Crown			**UE**	Unerupted		
RCT	Root Canal Treatment						

Figure 42: Update Dental Chart



Submit Cancel

☑ **Dental Status**

☐ Caries ☑ Reccurent ☐ Restoration

Mesial	☑	0
Distal	☑	0
Occlusal	☑	0
Buccal	☑	0
Lingual	☑	0

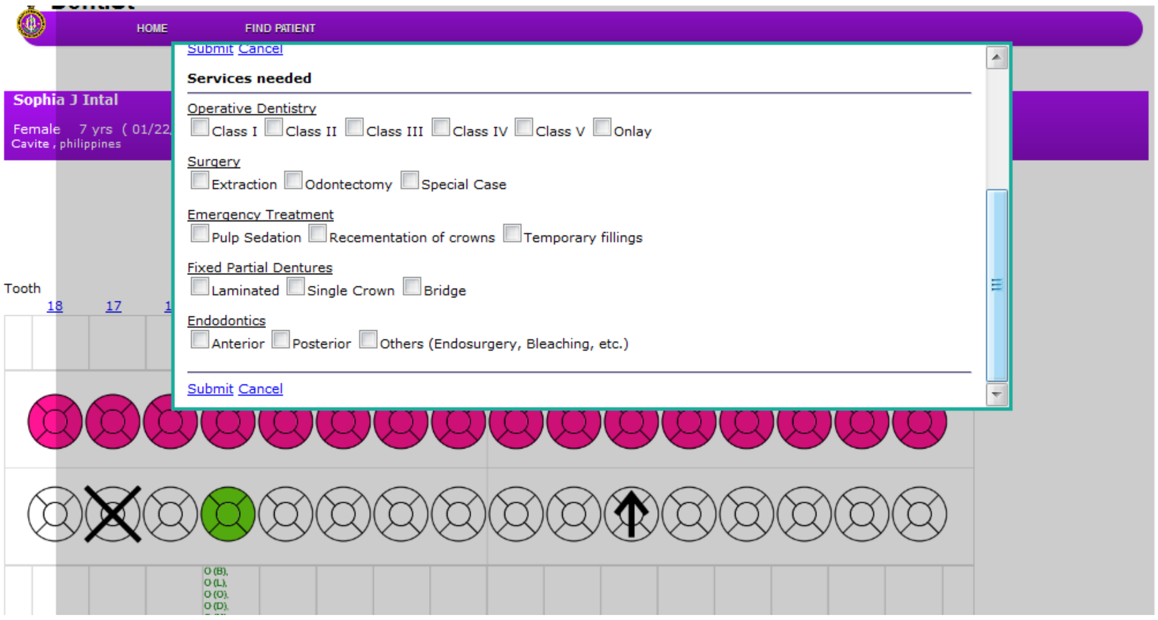
☑ **Whole Tooth**

☐ Extrusion ☐ Intrusion ☐ Mesial Drift Rotation ☐ Distal Drift Rotation ☐ Rotation

☐ Removable Partial Denture ☐ Fixed Bridged ☐ Root Canal Treatment ☐ Porcelain Fused to Metal ☐ Pit and Fissure Sealants

O (B),
O (L),

Figure 43: Edit Dental Chart



Submit Cancel

Services needed

Operative Dentistry

☐ Class I ☐ Class II ☐ Class III ☐ Class IV ☐ Class V ☐ Onlay

Surgery

☐ Extraction ☐ Odontectomy ☐ Special Case

Emergency Treatment

☐ Pulp Sedation ☐ Recementation of crowns ☐ Temporary fillings

Fixed Partial Dentures

☐ Laminated ☐ Single Crown ☐ Bridge

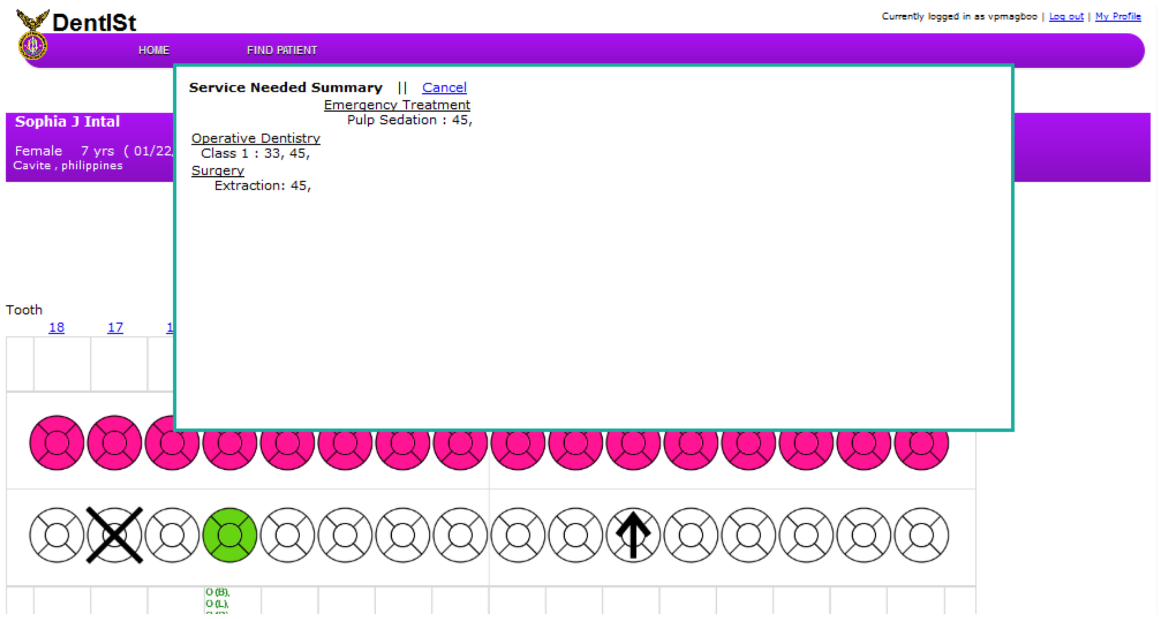
Endodontics

☐ Anterior ☐ Posterior ☐ Others (Endosurgery, Bleaching, etc.)

Submit Cancel

O (B),
O (L),
O (O),
O (D),

Figure 44: Update Dental Chart - Services Needed


DentISt

Currently logged in as vpmagboo | Log out | My Profile

HOME FIND PATIENT

Sophia J Intal

Female 7 yrs (01/22,
Cavite , philippines

Service Needed Summary || Cancel

Emergency Treatment
Pulp Sedation : 45,

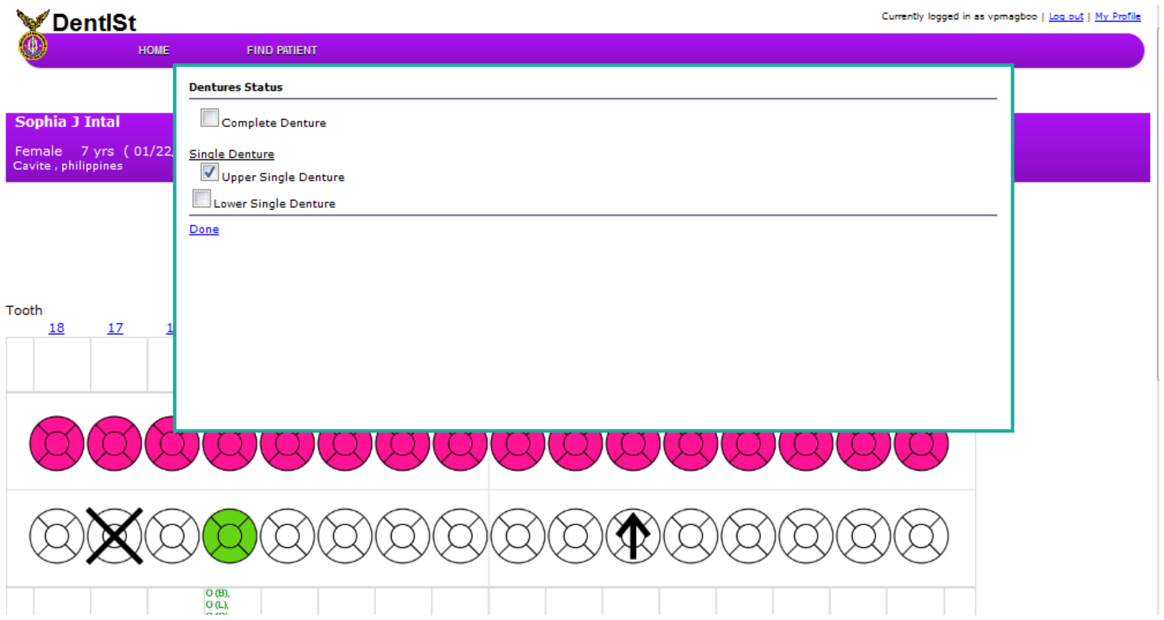
Operative Dentistry
Class 1 : 33, 45,

Surgery
Extraction: 45,

Tooth
18 17

O (B),
O (L),

Figure 45: Update Dental Chart - Service Needed Summary


DentISt

Currently logged in as vpmagboo | Log out | My Profile

HOME FIND PATIENT

Sophia J Intal

Female 7 yrs (01/22,
Cavite , philippines

Dentures Status

Complete Denture

Single Denture

[✓ Upper Single Denture]

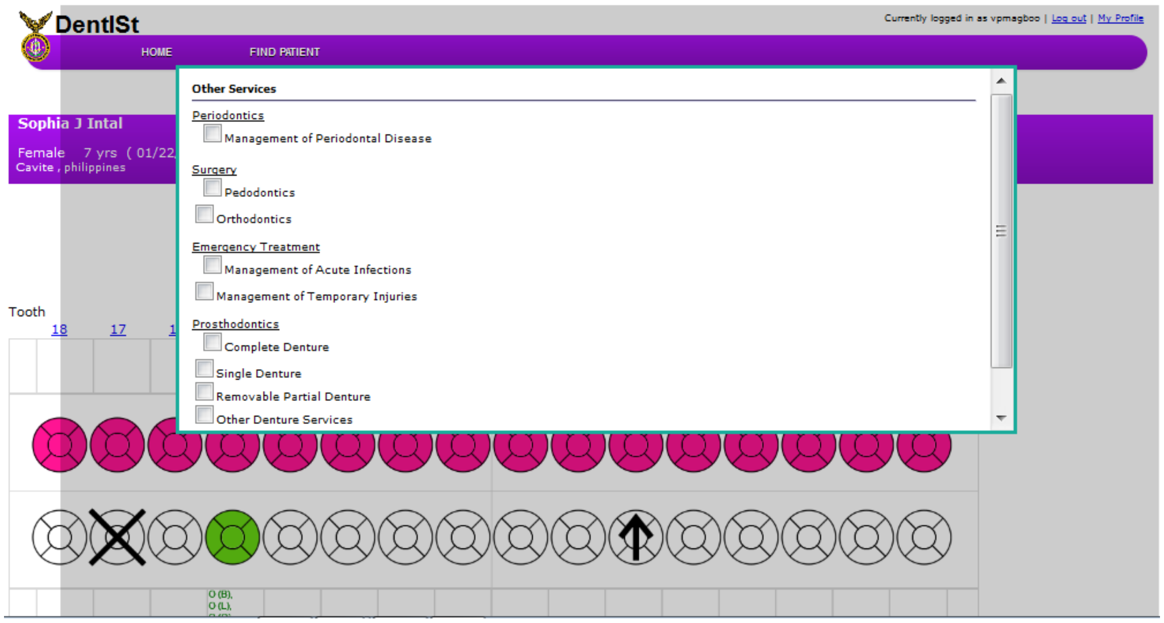
Lower Single Denture

Done

Tooth
18 17

O (B),
O (L),

Figure 46: Update Dental Chart - Dentures



DentISt

Currently logged in as vpmagboo | Log out | My Profile

HOME FIND PATIENT

Sophia J Intal

Female 7 yrs (01/22,

Cavite , philippines

Tooth

18 17

Other Services

Periodontics

- [ ] Management of Periodontal Disease

Surgery

- [ ] Pedodontics
- [ ] Orthodontics

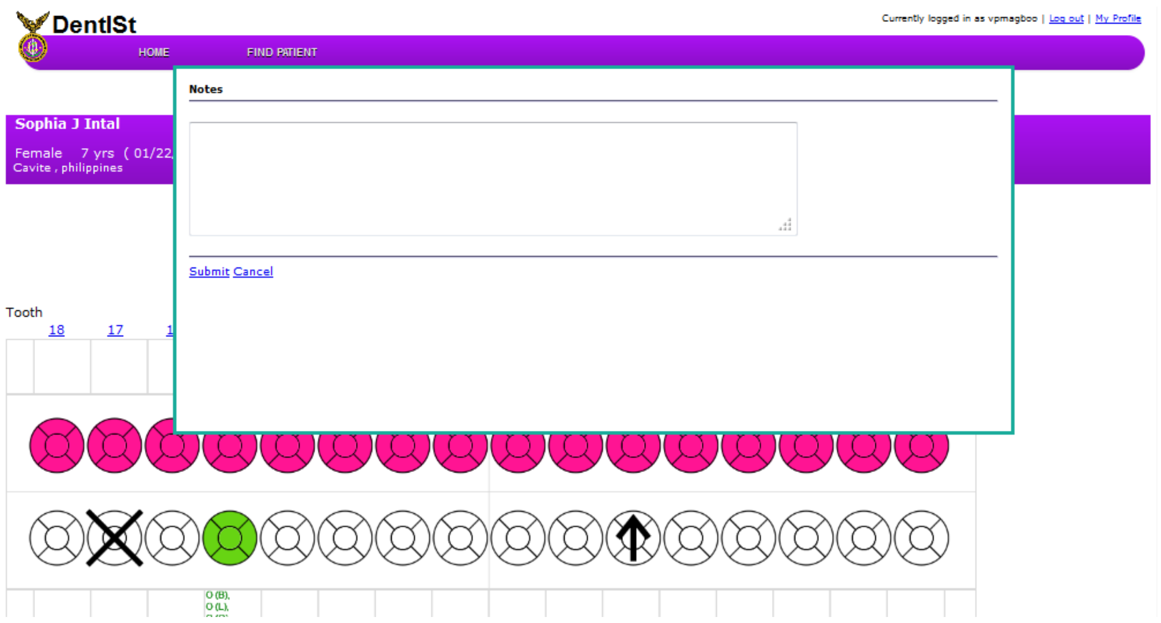
Emergency Treatment

- [ ] Management of Acute Infections
- [ ] Management of Temporary Injuries

Prosthodontics

- [ ] Complete Denture
- [ ] Single Denture
- [ ] Removable Partial Denture
- [ ] Other Denture Services

Figure 47: Update Dental Chart - Other Services



DentISt

Currently logged in as vpmagboo | Log out | My Profile

HOME FIND PATIENT

Sophia J Intal

Female 7 yrs (01/22,

Cavite , philippines

Tooth

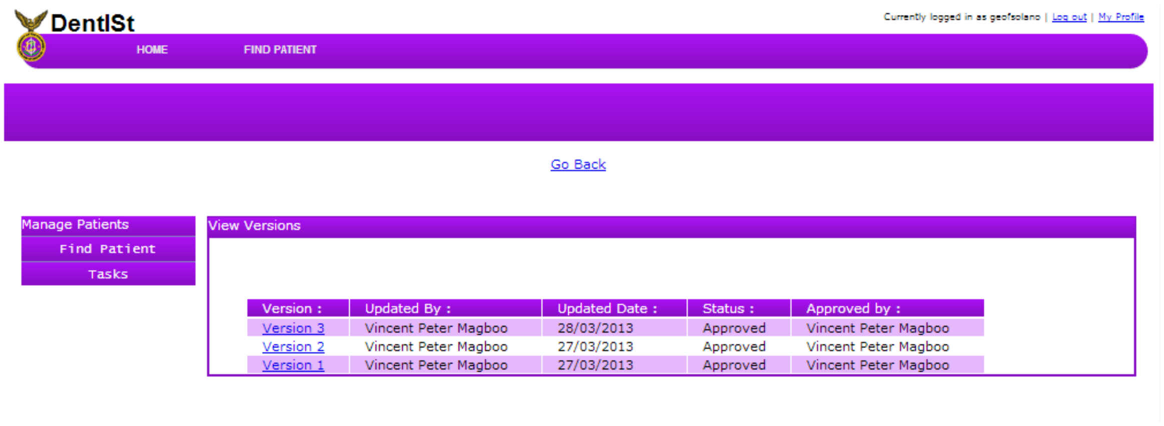
18 17

Notes

Submit Cancel

Figure 48: Update Dental Chart - Notes

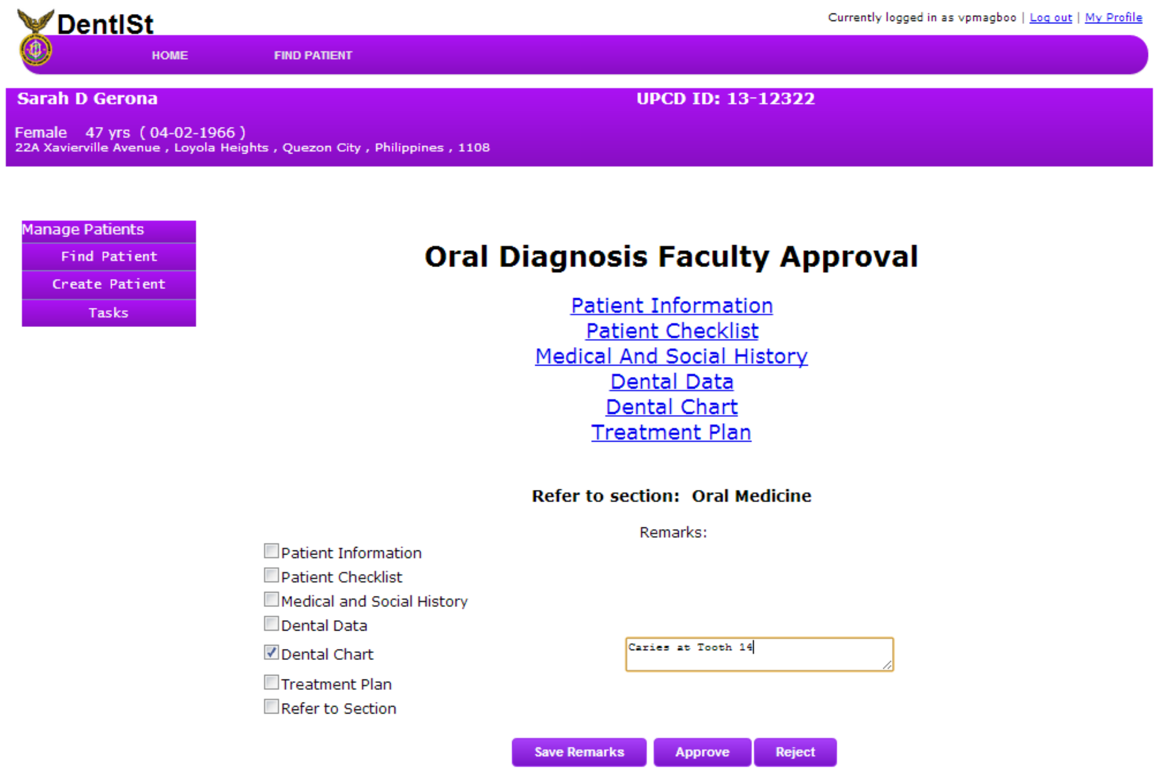
The clinician can also view previous versions of the form record along with who updated/approved the record and the respective dates as shown in Figure 49.



Version :	Updated By :	Updated Date :	Status :	Approved by :
Version 3	Vincent Peter Magboo	28/03/2013	Approved	Vincent Peter Magboo
Version 2	Vincent Peter Magboo	27/03/2013	Approved	Vincent Peter Magboo
Version 1	Vincent Peter Magboo	27/03/2013	Approved	Vincent Peter Magboo

Figure 49: View Versions of Patient Form

If the clinician is a faculty, the patient forms are submitted to the section indicated in Refer to Section field. However if the clinician is a student, the forms are forwarded to the faculty clinicians of the section for checking and approval. The faculty can approve or reject the updates of the clinicians. The faculty can add remarks and reject the updates so that the forms are submitted back to the assigned student clinician for modifications. The faculty also has the choice to save remarks temporarily and decide later whether to approve or reject. Approval of updates is shown in Figure 50. Figure 51 shows when a record update is rejected by a clinician.



Sarah D Gerona UPCD ID: 13-12322

Female 47 yrs (04-02-1966)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Oral Diagnosis Faculty Approval

Patient Information
Patient Checklist
Medical And Social History
Dental Data
Dental Chart
Treatment Plan

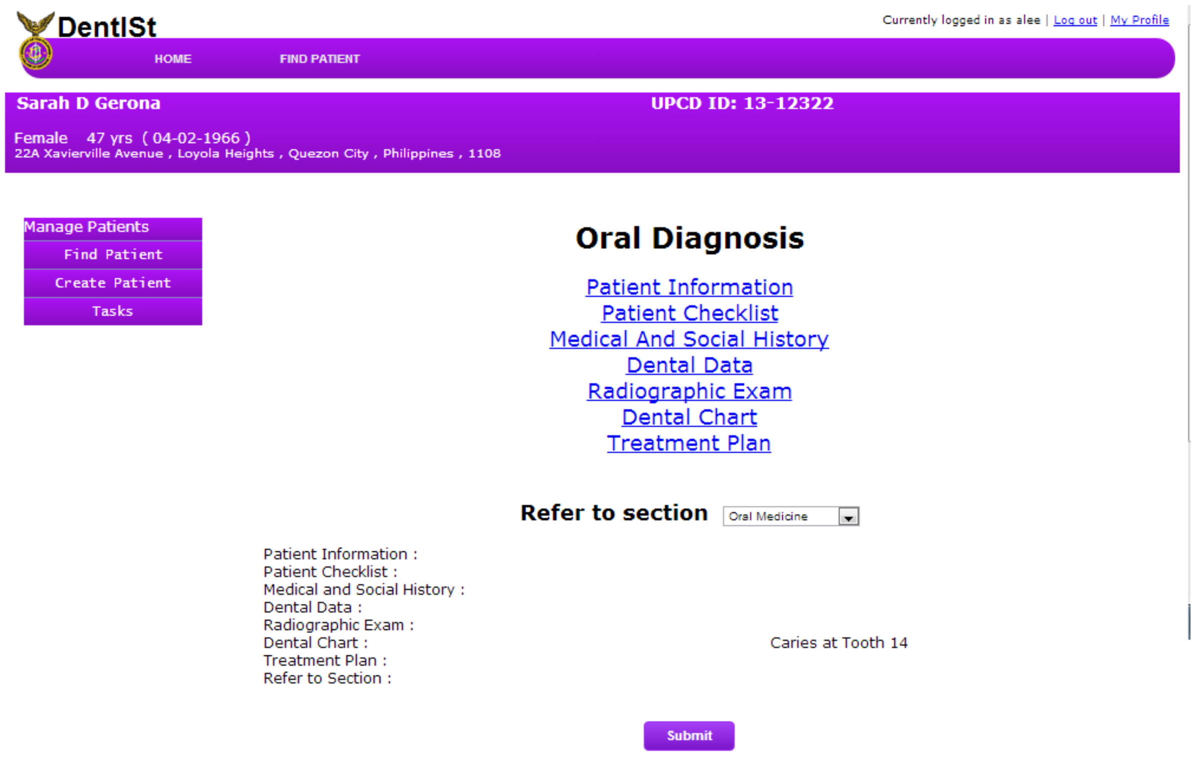
Refer to section: Oral Medicine

Remarks:

- [ ] Patient Information
- [ ] Patient Checklist
- [ ] Medical and Social History
- [ ] Dental Data
- [✓] Dental Chart
- [ ] Treatment Plan
- [ ] Refer to Section

Caries at Tooth 14

Figure 50: Faculty Approval of Student Clinician Updates



Currently logged in as alee | Log out | My Profile

Sarah D Gerona UPCD ID: 13-12322

Female 47 yrs (04-02-1966)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Oral Diagnosis

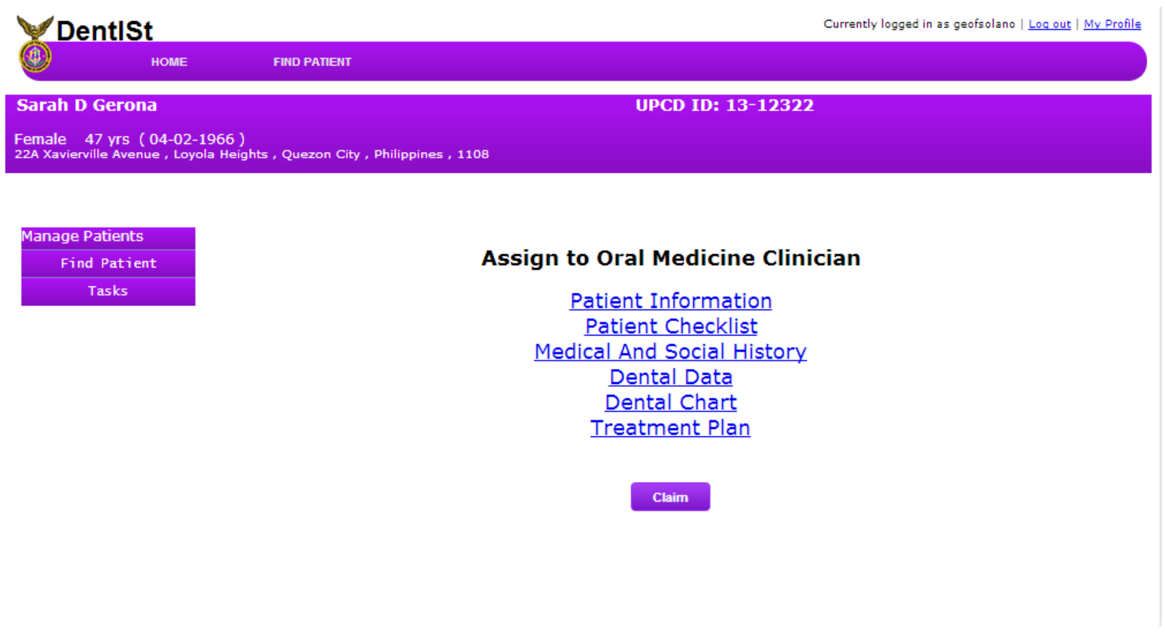
Patient Information
Patient Checklist
Medical And Social History
Dental Data
Radiographic Exam
Dental Chart
Treatment Plan

Refer to section: Oral Medicine

Patient Information :
Patient Checklist :
Medical and Social History :
Dental Data :
Radiographic Exam :
Dental Chart : Caries at Tooth 14
Treatment Plan :
Refer to Section :

Figure 51: Faculty Rejects Student Clinician Updates

The patient record is forwarded to the appropriate section once the record has been finalized by a faculty clinician. The clinicians of the section can then claim the patient case as shown in Figure 52. The clinician who claimed the case will be the assigned clinician and will be the only one who can edit the record while the patient is in the section.



DentISt

Currently logged in as geofsolano | Log out | My Profile

HOME FIND PATIENT

Sarah D Gerona UPCD ID: 13-12322

Female 47 yrs (04-02-1966)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Manage Patients
Find Patient
Tasks

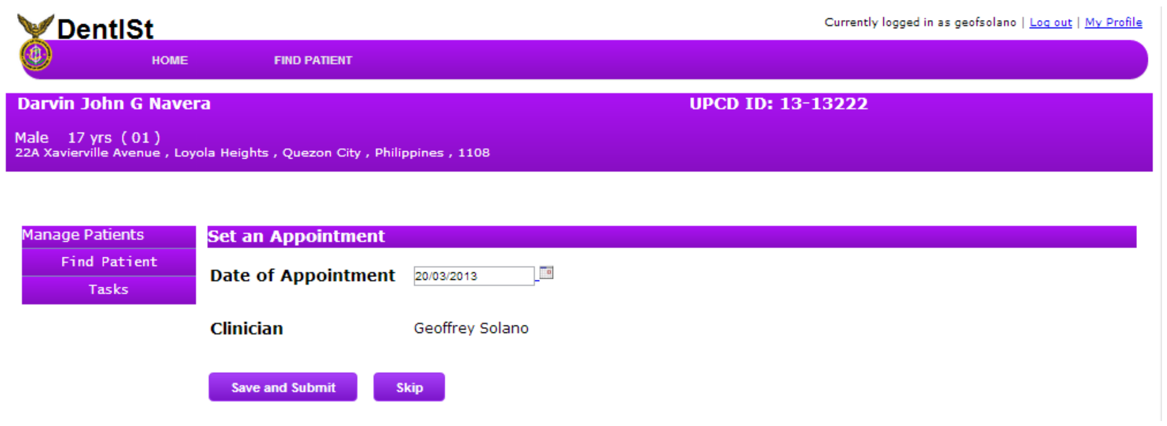
Assign to Oral Medicine Clinician

Patient Information
Patient Checklist
Medical And Social History
Dental Data
Dental Chart
Treatment Plan

Claim

Figure 52: Claim Patient Case of Clinicians in the Section

The assigned clinician can set appointment with the patient or choose to skip it as shown in Figure 53.



DentISt

Currently logged in as geofsolano | Log out | My Profile

HOME FIND PATIENT

Darvin John G Navera UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Manage Patients
Find Patient
Tasks

Set an Appointment

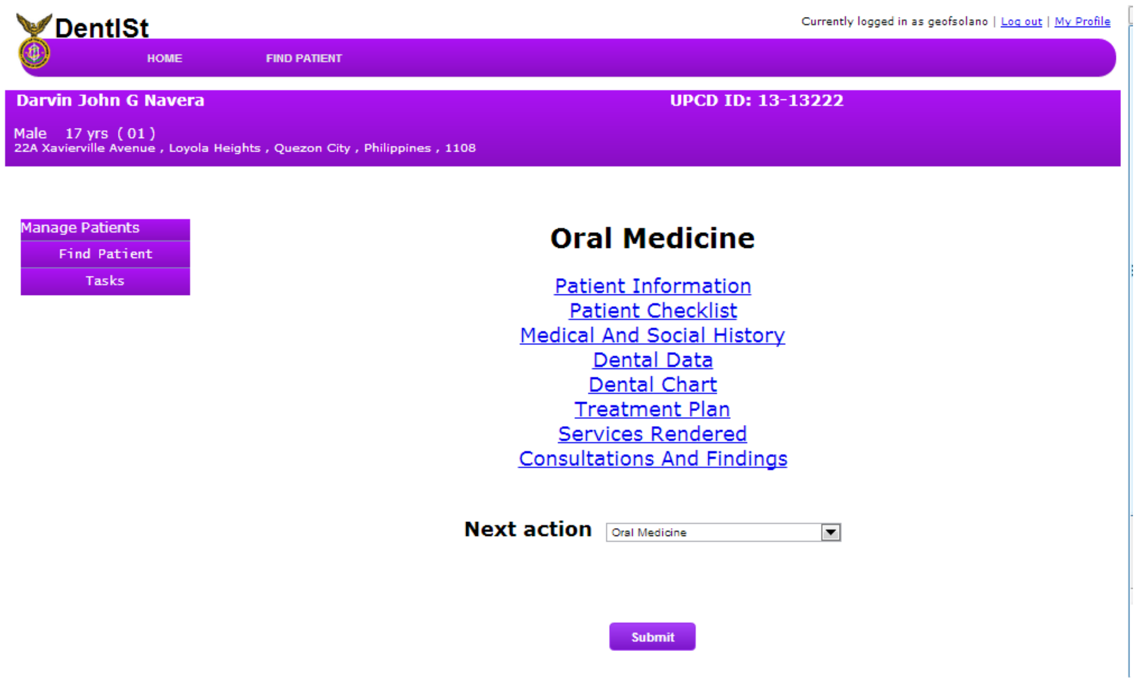
Date of Appointment 20/03/2013

Clinician Geoffrey Solano

Save and Submit Skip

Figure 53: Set Appointment

After setting an appointment, the patient case is added to the list of appointments of the clinician with the corresponding date of appointment. The task can be accessed from the appointment list or the task list. If the clinician chose to skip the appointment, no appointment is saved and the task can only be seen from the task list. The clinician can directly edit the patient forms shown in Figure 54. The forms that can be edited, in addition to the Oral Diagnosis forms are the section specific forms such as Services Rendered and Consultations and Findings Form. The clinician can refer the patient to other sections, set another appointment with the patient or end the patient case. On submission, the forms will be forwarded to faculty for approval if the clinician is a student.


DentISt

Currently logged in as geofsolano | Log out | My Profile

HOME FIND PATIENT

Darvin John G Navera UPCD ID: 13-13222

Male 17 yrs (01)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

Manage Patients
Find Patient
Tasks

Oral Medicine

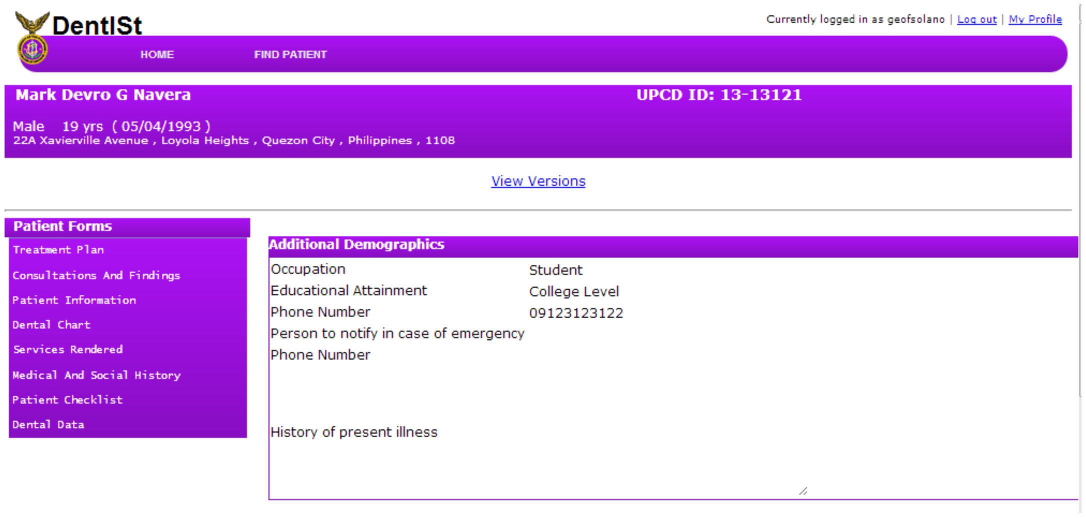
Patient Information
Patient Checklist
Medical And Social History
Dental Data
Dental Chart
Treatment Plan
Services Rendered
Consultations And Findings

Next action: Oral Medicine

Submit

Figure 54: Section Form Dashboard

All clinicians can also view all versions of the patient record. The forms are not editable as shown in Figure 55. They can also view their list of upcoming appointments which is linked to corresponding task as shown in Figure 56. Faculty clinicians, on the other hand, can view list of appointments of all clinicians shown in Figure 57.



Mark Devro G Navera UPCD ID: 13-13121

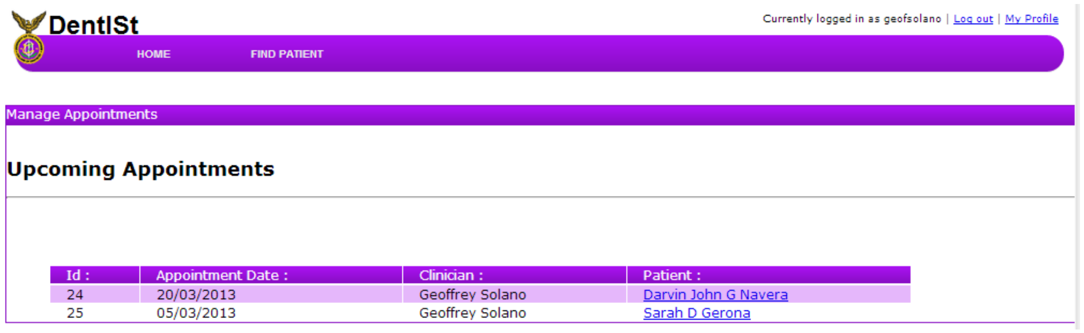
Male 19 yrs (05/04/1993)
22A Xavierville Avenue , Loyola Heights , Quezon City , Philippines , 1108

View Versions

Additional Demographics

Occupation	Student
Educational Attainment	College Level
Phone Number	09123123122
Person to notify in case of emergency	
Phone Number	
History of present illness	

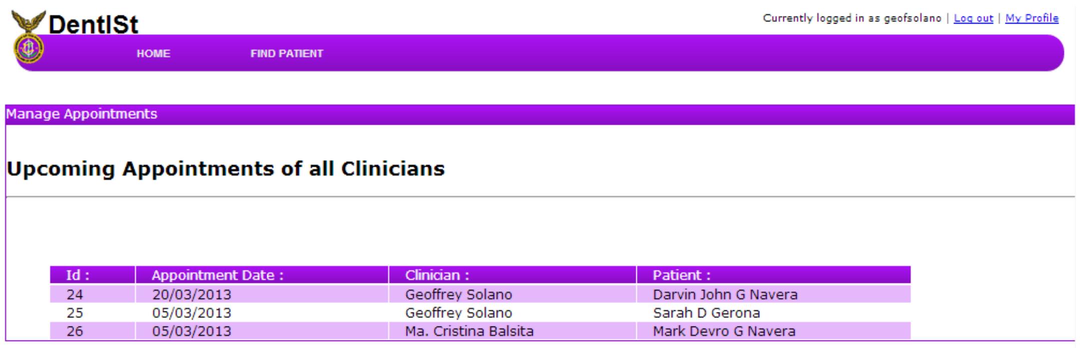
Figure 55: View Patient Record



Upcoming Appointments

Id :	Appointment Date :	Clinician :	Patient :
24	20/03/2013	Geoffrey Solano	Darvin John G Navera
25	05/03/2013	Geoffrey Solano	Sarah D Gerona

Figure 56: View Own Upcoming Appointments



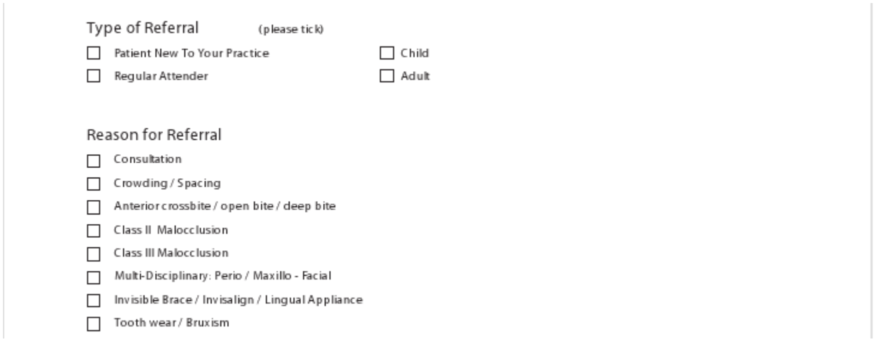
Upcoming Appointments of all Clinicians

Id :	Appointment Date :	Clinician :	Patient :
24	20/03/2013	Geoffrey Solano	Darvin John G Navera
25	05/03/2013	Geoffrey Solano	Sarah D Gerona
26	05/03/2013	Ma. Cristina Balsita	Mark Devro G Navera

Figure 57: View All Clinicians Upcoming Appointments - Faculty Clinician

B. Create New Section in the Workflow

The workflow administrator can create new section in the system by modifying the workflows of DentISt 3.0. He must be knowledgeable in jBPM. When adding a section, forms needed in the section must be added first in the Forms Workflow to use them in the Patient Workflow. The workflow administrator needs to coordinate with the database administrator to create database tables and stored functions of the forms to be added. For illustrations, Orthodontics section and Orthodontics Referral Form will be used as examples. Figure 58 shows the sample Orthodontics Referral Form.


Type of Referral (please tick)

☐ Patient New To Your Practice ☐ Child

☐ Regular Attender ☐ Adult

Reason for Referral

☐ Consultation

☐ Crowding / Spacing

☐ Anterior crossbite / open bite / deep bite

☐ Class II Malocclusion

☐ Class III Malocclusion

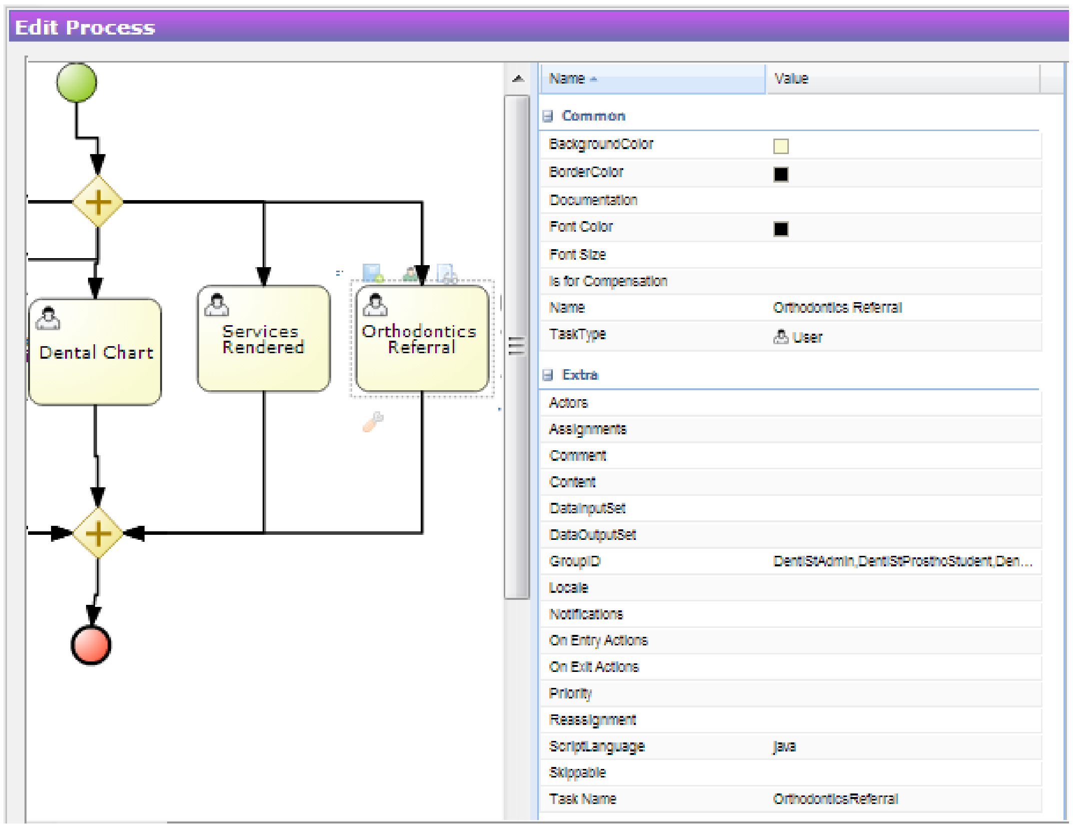
☐ Multi-Disciplinary: Perio / Maxillo - Facial

☐ Invisible Brace / Invisalign / Lingual Appliance

☐ Tooth wear / Bruxism

Figure 58: Sample Orthodontics Referral Form

The following are the steps to create the forms of the section. Go to the Workflow Dashboard and open the Forms Workflow. Add new user task and edit its properties such as the name, actors(username of the users in DentISt 3.0) or groups (the database role names in DentISt 3.0) who can access the forms and the task name. The task name will be the displayed name of the form in DentISt 3.0. It should not contain special characters such as spaces, slash, etc. If the form name contains two or more words, capitalize the first letter of each word to display them with spaces in DentISt3.0. Figure 59 shows the Orthodontics Referral form added in the Forms Workflow.



Name	Value
Common	
BackgroundColor	🟨
BorderColor	■
Documentation	
Font Color	■
Font Size	
Is for Compensation	
Name	Orthodontics Referral
TaskType	User
Extra	
Actors	
Assignments	
Comment	
Content	
DataInputSet	
DataOutputSet	
GroupID	DentIStAdmin,DentIStProsthoStudent,Den...
Locale	
Notifications	
On Entry Actions	
On Exit Actions	
Priority	
Reassignment	
ScriptLanguage	java
Skippable	
Task Name	OrthodonticsReferral

Figure 59: Add Orthodontics Referral Form in the Workflow

Note the properties of the Orthodontics Referral task shown in the rightmost panel of the window in Figure 59. Name is Orthodontontics Referral, GroupID are DentIStAdmin, DentIStProsthoStudent, DentIStOperativeDentStudent, DentIStOralMedFac, etc. separated by a comma. These are the database role names in DentISt 3.0. The Task Name is OrthodonticsReferral. The form name that will be displayed in system is Orthodontics Referral which is based on the Task Name given.

Next step is to create the UI of the form. To do this, click on the middle green icon on top of the task as shown in Figure 60. The form editor will be displayed as shown in Figure 61. The editor supports HTML, Javascript and CSS.

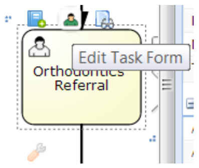
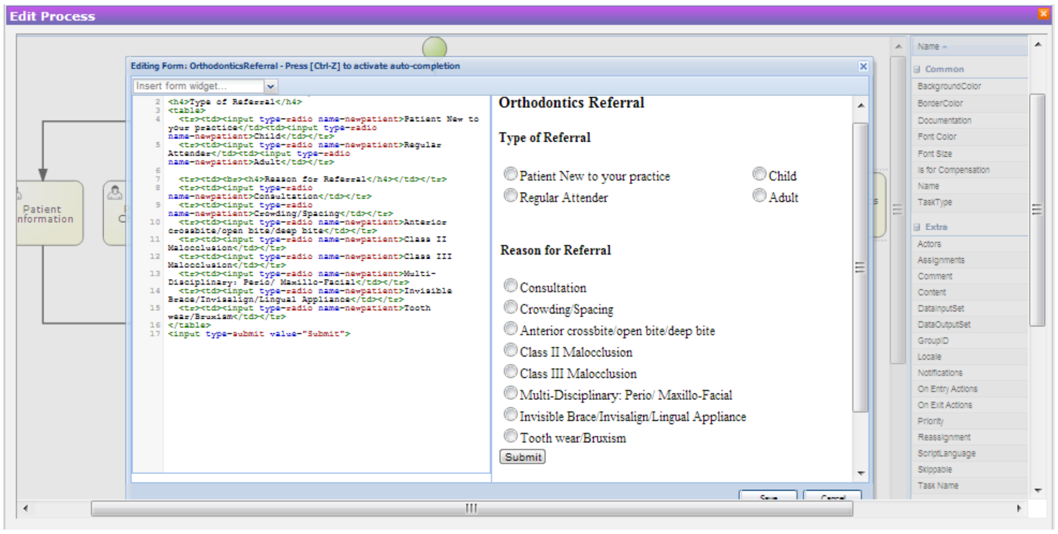


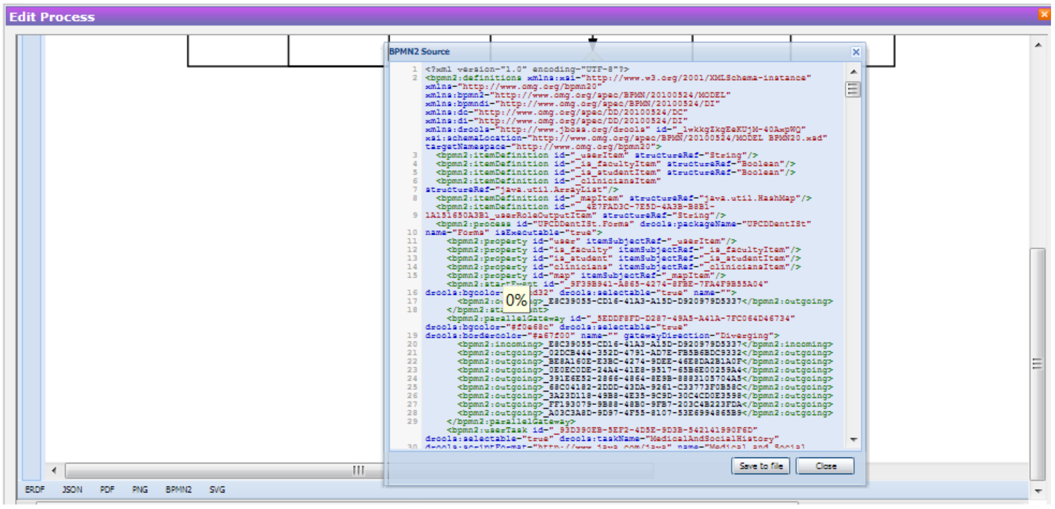
Figure 60: Edit Task Form Icon



```
<h4>Type of Referral</h4>
<table>
  <tr><td><input type=radio name=newpatient>Patient New to your practice</td><td><input type=radio name=newpatient>Child</td></tr>
  <tr><td><input type=radio name=newpatient>Regular Attender</td><td><input type=radio name=newpatient>Adult</td></tr>

  <tr><td><br><h4>Reason for Referral</h4></td></tr>
  <tr><td><input type=radio name=newpatient>Consultation</td></tr>
  <tr><td><input type=radio name=newpatient>Crowding/Spacing</td></tr>
  <tr><td><input type=radio name=newpatient>Anterior crossbite/open bite/deep bite</td></tr>
  <tr><td><input type=radio name=newpatient>Class II Malocclusion</td></tr>
  <tr><td><input type=radio name=newpatient>Class III Malocclusion</td></tr>
  <tr><td><input type=radio name=newpatient>Multi-Disciplinary: Perio/ Maxillo-Facial</td></tr>
  <tr><td><input type=radio name=newpatient>Invisible Brace/Invisalign/Lingual Appliance</td></tr>
  <tr><td><input type=radio name=newpatient>Tooth wear/Bruxism</td></tr>
</table>
<input type=submit value="Submit">
```

Figure 61: Edit Task Form

Lastly, click the BPMN2 button at the bottom of the workflow editor and save the BPMN2 Export File as shown in Figure 62. Save and close the Forms Workflow. The BPMN2 file must be sent to the database administrator together with the list of the fields in the form in order of how they are displayed in the form. The BPMN2 export file will be the basis of the table grants in the database. The format of the stored function is insert_*taskname* for inserting data and get_*taskname* for retrieving patient data. *taskname* is the value given in the Task Name property of the newly added task(i.e. OrthodonticsReferral) in lowercase. The parameters of the insert_*taskname* stored function are patientid, the form fields, version, updated_by, updated_date, updated_time, approved, approved_by, approved_date, approved_time in this order. patientid and version are of type integer while the others are of type text. The parameter of get_*taskname* is the patientid of type integer. In the Orthodontics Referral example, the stored functions are insert_orthodonticsreferral and get_orthodonticsreferral.



```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.omg.org/bpmn20"
xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:drools="http://www.jboss.org/drools" id="_lwkkgZkgEeKUjM-40AxpWQ"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd"
targetNamespace="http://www.omg.org/bpmn20">
  <bpmn2:itemDefinition id="_userItem" structureRef="String"/>
  <bpmn2:itemDefinition id="_is_facultyItem" structureRef="Boolean"/>
  <bpmn2:itemDefinition id="_is_studentItem" structureRef="Boolean"/>
  <bpmn2:itemDefinition id="_cliniciansItem"
structureRef="java.util.ArrayList"/>
  <bpmn2:itemDefinition id="_mapItem" structureRef="java.util.HashMap"/>
  <bpmn2:itemDefinition id="__4E7FAD3C-7E5D-4A3B-B8B1-
1A151650A3B1_userRoleOutputItem" structureRef="String"/>
  <bpmn2:process id="UPCDDentISt.Forms" drools:packageName="UPCDDentISt"
name="Forms" isExecutable="true">
    <bpmn2:property id="user" itemSubjectRef="_userItem"/>
    <bpmn2:property id="is_faculty" itemSubjectRef="_is_facultyItem"/>
    <bpmn2:property id="is_student" itemSubjectRef="_is_studentItem"/>
    <bpmn2:property id="clinicians" itemSubjectRef="_cliniciansItem"/>
    <bpmn2:property id="map" itemSubjectRef="_mapItem"/>
    <bpmn2:startEvent id="_9F39B941-A865-4274-8FBE-7FA4F9B55A04"
drools:bgcolor=[illegible]d32" drools:selectable="true" name="">
      <bpmn2:o[illegible]g>_E8C39055-CD16-41A3-A15D-D920979D5337</bpmn2:outgoing>
    </bpmn2:st[illegible]nt>
    <bpmn2:parallelGateway id="_5EDDF8FD-D287-49A5-A41A-7FC064D46734"
drools:bgcolor="#f0e68c" drools:selectable="true"
drools:bordercolor="#a67f00" name="" gatewayDirection="Diverging">
      <bpmn2:incoming>_E8C39055-CD16-41A3-A15D-D920979D5337</bpmn2:incoming>
      <bpmn2:outgoing>_02DCB444-352D-4791-AD7E-FB5B6BDC9332</bpmn2:outgoing>
      <bpmn2:outgoing>_BE8A160E-E3BC-4274-9DEE-46E8DA2B1A0F</bpmn2:outgoing>
      <bpmn2:outgoing>_0E0EC0DE-24A4-41E8-9517-65B6E00259A4</bpmn2:outgoing>
      <bpmn2:outgoing>_391E6E52-2866-4864-8E9B-8883105704A5</bpmn2:outgoing>
      <bpmn2:outgoing>_68C04182-2DDD-43DA-9261-C33773F0B58C</bpmn2:outgoing>
      <bpmn2:outgoing>_3A23D118-49B8-4E35-9C9D-30C4CD0E3598</bpmn2:outgoing>
      <bpmn2:outgoing>_FF193079-9B88-48B0-9FB7-203C4B223FDA</bpmn2:outgoing>
      <bpmn2:outgoing>_A03C3A8D-9D97-4F55-8107-53E6994865B9</bpmn2:outgoing>
    </bpmn2:parallelGateway>
    <bpmn2:userTask id="_93D390EB-5EF2-4D5E-9D3B-542141990F6D"
drools:selectable="true" drools:taskName="MedicalAndSocialHistory"
drools:scriptFormat="http://www.java.com/java" name="Medical and Social
```

Figure 62: BPMN Export File

The Orthodontics section can now be added to the Patient Workflow. Open the Patient Workflow and add a process variable of boolean type. This variable will used to direct the workflow to the section. Edit the Oral Diagnosis User Task to add the section in the refer to section options in the task form. Edit the Check Section Choice Script Task to add a condition to check whether the section is chosen. Edit the OD Faculty Approval User Task and Faculty Approval Result Script Task to add remarks field and variables.

Add the needed tasks in the section. These tasks can be Assign to Section Clinician, Set Appointment with Section Clinician, Edit Section Forms and Section Faculty Approval and some other script tasks to set variables like section choice and remarks variables. In the Edit Section Forms User Task, add the link to the Orthodontics Referral Form that was created using the <a href="OrthodonticsReferral">tag. The value of the href attribute must be the Task Name of the form.

And finally, build the UPCDDentISt package from Drools Guvnor, the jBPM repository where the processes(i.e. Patient and Forms Workflow) and the task forms are stored. Restart the server so that changes can take effect in the system.

VII. Discussions

Dental Information System 3.0 or DentISt 3.0 is the third version of the dental information system of the University of the Philippines College of Dentistry which helps clinicians and faculty store and access patient dental records electronically. DentISt 3.0 has a workflow support feature which helps to coordinate and monitor the flow of work during patient treatment. The system has 10 initial roles- system administrator, workflow administrator, faculty in oral diagnosis, student in oral diagnosis, faculty in oral medicine, student in oral medicine, faculty in prosthodontics, student in prosthodontics, faculty in operative dentistry and student in operative dentistry. The system stores patient dental record consisting of basic information, physical assessment, vital signs, dental history, medical history, social history, soft tissue exam, treatment plan, dental chart, services rendered, consultations and findings form. Clinicians can also view and perform tasks based on their roles. All clinicians can set appointment with their patients or refer patients to other UPCD sections. Additionally, student clinicians can update patient record and submit them for faculty approval. Faculty clinician can perform all student clinician functionalities and also approve the student clinicians' updates. Workflow administrator, on the other hand, can add, edit, and delete the workflow of the system.

The workflow support not found in the two previous versions of dental information system of UPCD is a significant improvement in the system. The system can now easily adapt to the workflow changes in UPCD. The workflow of system can be modified without the difficulty of recoding the whole system. The changes made will immediately take effect after a system restart. The workflow administrator can add, delete or change the order of tasks and manage the users who can perform each task. He can also control which forms and patient records are accessible to each section clinicians. The workflow admin can also add new UPCD patient forms or modify them easily via a web-based editor. Moreover with the help of jBPM workflow management system, clinicians can add and edit patient record depending on which section the patient currently is. Clinicians can also view list of their pending personal and per section tasks. A clinician of a section can claim a pending patient case. All updates and data entries made by student clinicians are automatically forwarded to faculty clinicians in the section for finalization and approval. The clinicians can also view all versions of the patient record per form, see its current status and who updated and approved the record. Oral diagnosis clinicians can create new patient and start a new case for returning patients. All clinicians can search patients, view and print records. Clinicians can also view and set appointment with their patients or refer them to other UPCD sections.

The use of jBPM workflow management allowed the dental information system to simulate the actual procedure of patient treatment in UPCD. It also prevents data redundancy and increases the credibility of patient record now that all updates are automatically forwarded to faculty clinicians. It also prevents the inefficiency and waste of resources of reprogramming the system whenever it needs modification. Unlike other workflow management systems, jBPM is flexible and can be used exactly according to the system's needs and developer's preferences. Although it has its own server and database, which is JBOSS server and h2 database respectively, it can also use other popular web servers like Tomcat or databases like MySQL, PostgreSQL or Oracle database. In DentISt 3.0, Tomcat 6.0 and PostgreSQL are used. jBPM is relatively new and less developed compared to the other existing workflow management system. jBPM stores form data as blob object in the database. However, to allow patient querying and statistics viewing in DentISt 3.0, jBPM is modified so that each form has a corresponding table in the database. If a form is added or modified, the workflow admin has to coordinate with the database admin to change the corresponding tables as well.

VIII. Conclusions

Dental Information System 3.0 is a much improved version of DentISt in terms of UPCD workflow integration into the system. This system not only stores patient records, the flow of work starting from the creation the patient record to completion of the treatment is also efficiently managed with the use of jBPM workflow management system. The faculty approval of records is assured in the workflow unlike in OpenMRS which assumes that any data entry is final. The labels of tooth conditions in the dental chart are also modified to comply with the standard legend used in paper based patient record.

The system can be easily edited especially when there are changes in the UPCD patient workflow or in the forms used to record patient data without the need to reprogram the system.

DentISt 3.0 now assumes the flow of tasks in the UP College of Dentistry and allows system flexibility for possible changes in workflow in the future.

The use of jBPM workflow management system allows clinicians to manage their tasks and patient appointments efficiently by providing them with a list pending tasks. This prevents a task from being overlooked. Misinformation is also avoided because all data entries are rechecked by the faculty. The workflow also ensures that only the assigned clinician has the privilege to edit the patient record. Furthermore only clinicians belonging to the section, where the patient currently is, can claim and attend to the case. The forms editable by a clinician can also be controlled depending on what sections he belongs.

IX. Recommendations

DentISt 3.0 improved on the integration of workflow of UPCD into dental information system. However, many section-specific forms are not yet implemented in the system. In addition to Oral Diagnosis forms and forms common to all sections, each UPCD section has its own set of section-specific forms which can be included in the system in the future. The Student Accomplishment Report previously requested by Dean Vicente Medina can also be included when they have already agreed on the final format of the form. UPCD can completely shift to electronic records if all the dental forms used by the college are already implemented in the system.

A concept dictionary integrated into system will also be highly beneficial in preventing data input errors and organizing dental terms and records.

Use of the new Form-Builder tool of jBPM will also help in a better and easier form creation. It is web-based form designer for creating task and process forms. It has a drag and drop feature and supports various layouts and scripting.

The Eclipse BPMN2 process designer plugin can substitute the web-based process designer to help reduce the overall system file size. It can help ease up system deployment.

An export file with the list of fields and stored functions of a form to be added can also be implemented. The export file can be used by the database administrator to generate SQL scripts for creating the database table.

The default server and database used by jBPM, which are JBoss and h2 database respectively, can be used by the system for easier configuration and deployment.

X. Bibliography

[1] T. Schleyer, "Dental informatics: An emerging biomedical informatics discipline," *Advances in Dental Research*, vol. 17, pp. 4–8, December 2003.

[2] T. Schleyer and H. Spallek, "Dental informatics: A cornerstone of dental practice," *J Am Dent Assoc.*, vol. 5, pp. 605–613, 2001.

[3] A. J. Lee, "Developing a dental information system with openmrs (open dentis)," 2011.

[4] M. C. Balsita, "Dentist: Dental information system 2.0," 2012.

[5] H. A. Reijers, N. Russell, S. van der Geer, and G. A. M. Krekels, "Workflow for healthcare: A methodology for realizing flexible medical treatment processes," *Lecture Notes in Business Information Processing*, vol. 43, pp. 593–604, 2010.

[6] C. Ellis and K. Keddara, "Dynamic change within workflow systems," *University of Colorado Report CU-CS-667-93*, pp. 10–21, 1993.

[7] C. Waegemann, "Status report 2002: Electronic health records," *Medical Records Institute*, 2002.

[8] A. Hoerbst and E. Ammenwerth, "Electronic health records: A systematic review on quality requirements," *Methods Inf Med*, vol. 49, pp. 320–336, 2010.

[9] R. A. Cederberg and J. A. Valenza, "Ethics and the electronic health record in dental school clinics," *Journal of Dental Education*, vol. 76, no. 5, pp. 584–589, 2012.

[10] M. F. Walji, D. Taylor, J. R. L. II, and J. A. Valenza, "Factors influencing implementation and outcomes of a dental electronic patient record system," *Journal of Dental Education*, vol. 73, no. 5, pp. 589–600, 2009.

[11] E. Mendoca, "Clinical decision support systems: perspectives in dentistry," *J Dent Educ*, vol. 16, no. 1, pp. 117–121, 2004.

[12] R. Wears and M. Berg, "Computer technology and clinical work. still waiting for godot," *JAMA*, vol. 293, pp. 1261–1263, 2005.

[13] C. Cain, S. Haque, and R. Hughes, eds., *Patient Safety and Quality: An Evidence-Based Handbook for Nurses.* Agency for Healthcare Research and Quality (US), 2008.

[14] J. Zhang, X. Lu, H. Nie, Z. Huang, and W. M. P. van der Aalst, "Radiology information system: a workflow-based approach," *International Journal of Computer Assisted Radiology and Surgery*, vol. 4, no. 5, pp. 509–516, 2009.

[15] L. Washington, "Analyzing workflow for a health it implementation," *Journal of AHIMA*, vol. 79, no. 1, pp. 64–65, 2008.

[16] A. H. M. ter Hofstede, W. M. P. van der Aalst, M. Adams, and N. Russell, eds., *Modern Business Process Automation: YAWL and its Support Environment.* Springer, 2010.

[17] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business process execution language for web services," *OASIS*, 2003.

[18] OMG, "Business process model and notation (bpmn), version 2.0," *Object Management Group (OMG)*, 2011.

[19] J. yan Zhang, X. dong Lu, H. long Duan, and H. chao Nie, "A medical information system architecture based on workflow technology," *IT in Medicine and Education*, vol. 1, pp. 1117 – 1121, 2009.

[20] R. Mans, N. Russell, W. van der Aalst, P. Bakker, A. Moleman, and M. Jaspers, "Proclets in healthcare," *J Biomed Inform*, vol. 43, no. 4, pp. 632–649, 2010.

[21] C. Combi, M. Gambini, S. Migliorini, and R. Posenato, "Modelling temporal, data-centric medical processes," *2nd ACM International Health Informatics Symposium*, pp. 141–150, 2012.

[22] P. Wohed, N. Russell, A. H. ter Hofstede, B. Andersson, and W. M. van der Aalst, "Patterns-based evaluation of open source bpm systems: The cases of jbpm, openwfe, and enhydra shark," *Information and Software Technology*, vol. 51, no. 8, p. 11871216, 2009.

[23] J. Hill and N. Dracos, "Research on workflow patterns based on jbpm and jpdl," *Gartner*, 2006.

[24] W. van der Aalst and A. ter Hofstede, "Yawl: yet another workflow language," *Information and Software Technology*, vol. 30, no. 4, p. 245275, 2005.

[25] T. Liebeskind, "Frameworks yawl / jbpm," *Seminar in Summer Semester 2009*, 2009.

[26] "jbpm." `http://www.jboss.org/jbpm`. Accessed on October, 2012..

[27] L. Peng and B. Zhou, "Research on workflow patterns based on jbpm and jpdl," *Computational Intelligence and Industrial Application*, vol. 2, pp. 838–843, 2008.

[28] "Dentist. what is a dentist?," August 2012.

[29] "University of the philippines college of dentistry." `http://cd.upm.edu.ph`. Accessed on October, 2012.

[30] "What is prosthodontics?." `http://www.ada.org.au/societies/aanzp/prosthodontics.aspx`. Accessed on October, 2012.

[31] S. White, "Introduction to bpmn," *IBM Corporation*, 2004.

[32] "jbpm overview." `http://docs.jboss.org/jbpm/v5.3/userguide/ch.overview.html`. Accessed on October, 2012..

[33] "Drools - the business logic integration platform." `http://www.jboss.org/drools/`. Accessed on October, 2012.

[34] "jbpm overview." `http://www.indicthreads.com/1446/simplified-bpm-integration-with-jboss-jbpm/`. Accessed on October, 2012..

[35] M. Salatino, ed., *jBPM Developer Guide.* Packt Publishing, 2009.

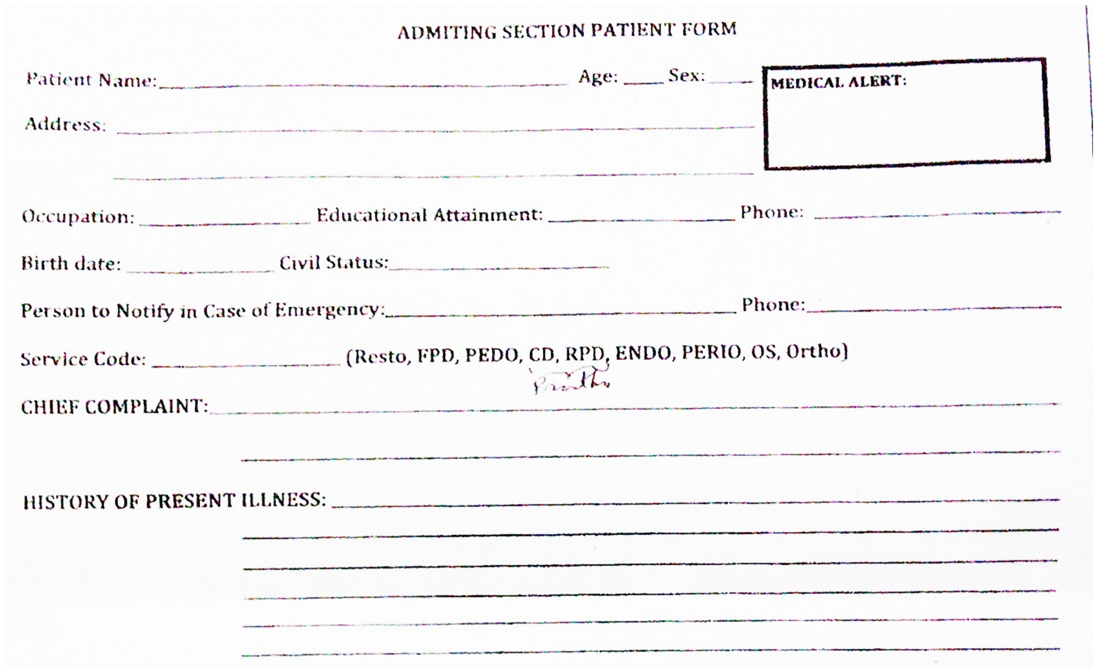
XI. Appendix

A. Forms and Stored Functions Mapping

Form Name	Stored Functions
Patient Information	insert_patientinformation get_patientinformation
Patient Checklist	insert_patientchecklist get_patientchecklist
Medical and Social History	insert_medicalandsocialhistory get_medicalandsocialhistory
Dental Data	insert_dentaldata get_dentaldata
Radiographic Exam	insert_radiographicexam get_radiographicexam
Dental Chart	insert_dentalchart get_dentalchart
Treatment Plan	insert_treatmentplan get_treatmentplan
Services Rendered	insert_servicesrendered get_servicesrendered
Consultations and Findings	insert_consultationsandfindings get_consultationsandfindings

Table 23: Forms and Stored Functions Mapping

B. UPCD Patient Form


ADMITING SECTION PATIENT FORM

Patient Name: __________ Age: ____ Sex: ____

MEDICAL ALERT:

Address: __________

Occupation: __________ Educational Attainment: __________ Phone: __________

Birth date: __________ Civil Status: __________

Person to Notify in Case of Emergency: __________ Phone: __________

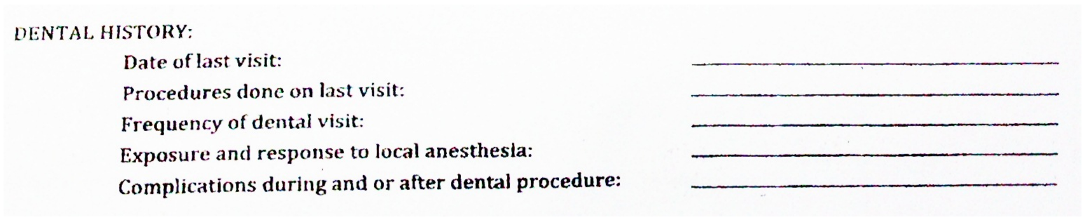
Service Code: __________ (Resto, FPD, PEDO, CD, RPD, ENDO, PERIO, OS, Ortho)

Prosth

CHIEF COMPLAINT: __________

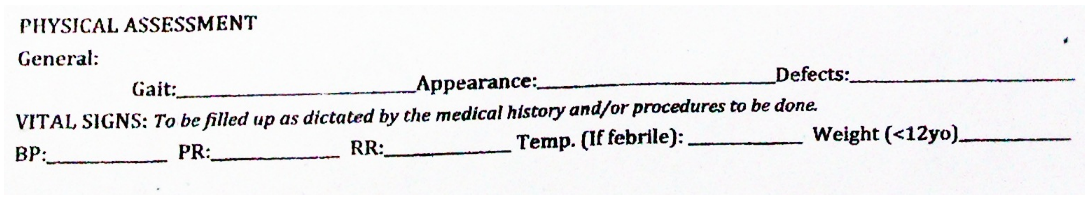
HISTORY OF PRESENT ILLNESS: __________

Figure 63: UPCD Admitting Section Patient Form with Patient Demographics, Chief Complaint, History of Present Illness


DENTAL HISTORY:

Date of last visit:	__________
Procedures done on last visit:	__________
Frequency of dental visit:	__________
Exposure and response to local anesthesia:	__________
Complications during and or after dental procedure:	__________

Figure 64: UPCD Admitting Section Patient Form with Dental History


PHYSICAL ASSESSMENT

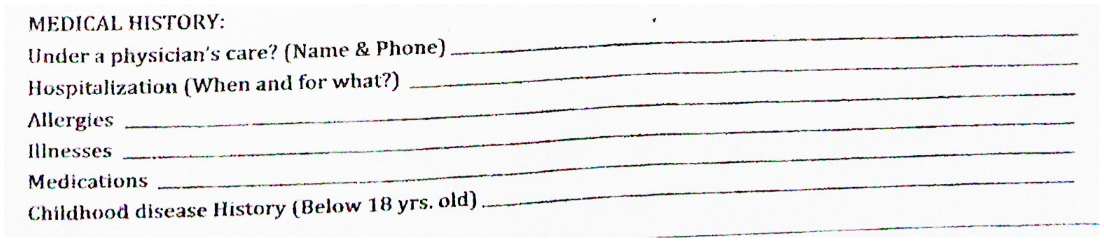
General:

Gait:__________ Appearance:__________ Defects:__________

VITAL SIGNS: *To be filled up as dictated by the medical history and/or procedures to be done.*

BP:______ PR:______ RR:______ Temp. (If febrile): ______ Weight (<12yo)______

Figure 65: UPCD Admitting Section Patient Form with Physical Assessment and Vital Signs


MEDICAL HISTORY:

Under a physician's care? (Name & Phone) __________

Hospitalization (When and for what?) __________

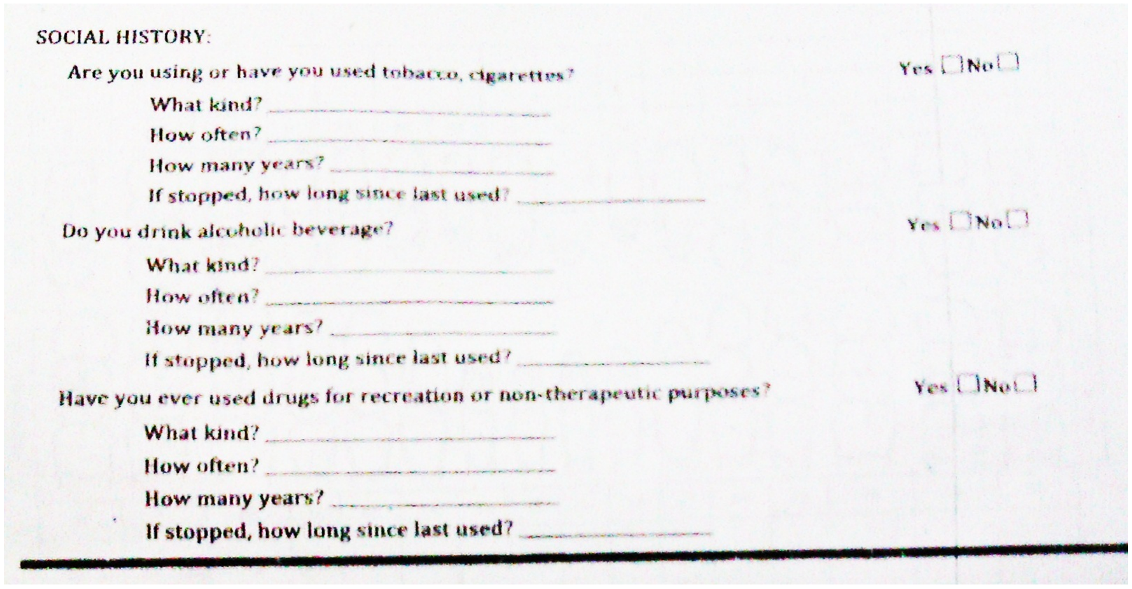
Allergies __________

Illnesses __________

Medications __________

Childhood disease History (Below 18 yrs. old) __________

Figure 66: UPCD Admitting Section Patient Form with Medical History


SOCIAL HISTORY:

Are you using or have you used tobacco, cigarettes? Yes ☐ No ☐

What kind? __________

How often? __________

How many years? __________

If stopped, how long since last used? __________

Do you drink alcoholic beverage? Yes ☐ No ☐

What kind? __________

How often? __________

How many years? __________

If stopped, how long since last used? __________

Have you ever used drugs for recreation or non-therapeutic purposes? Yes ☐ No ☐

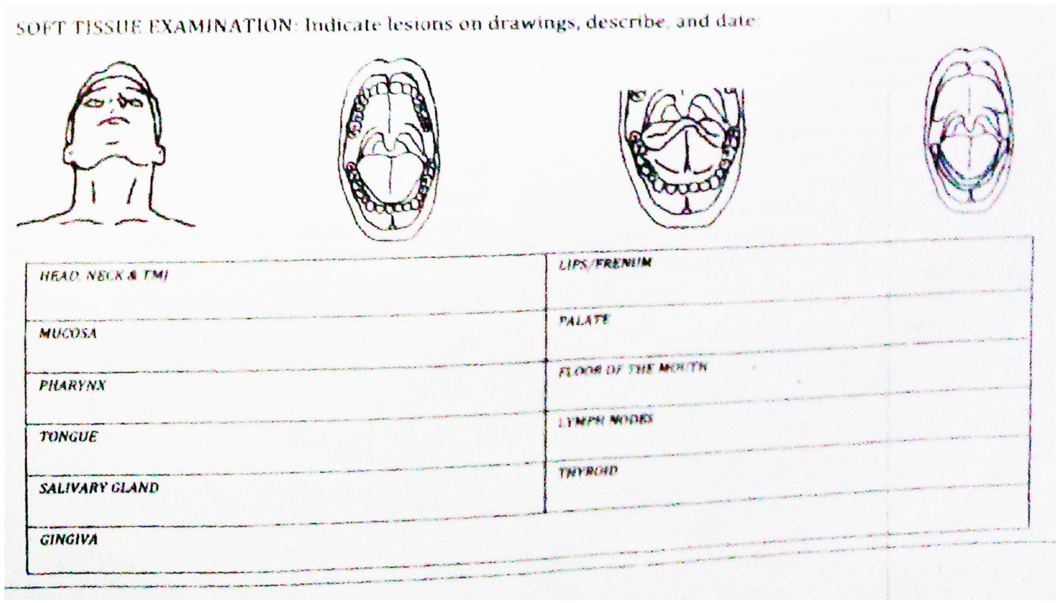
What kind? __________

How often? __________

How many years? __________

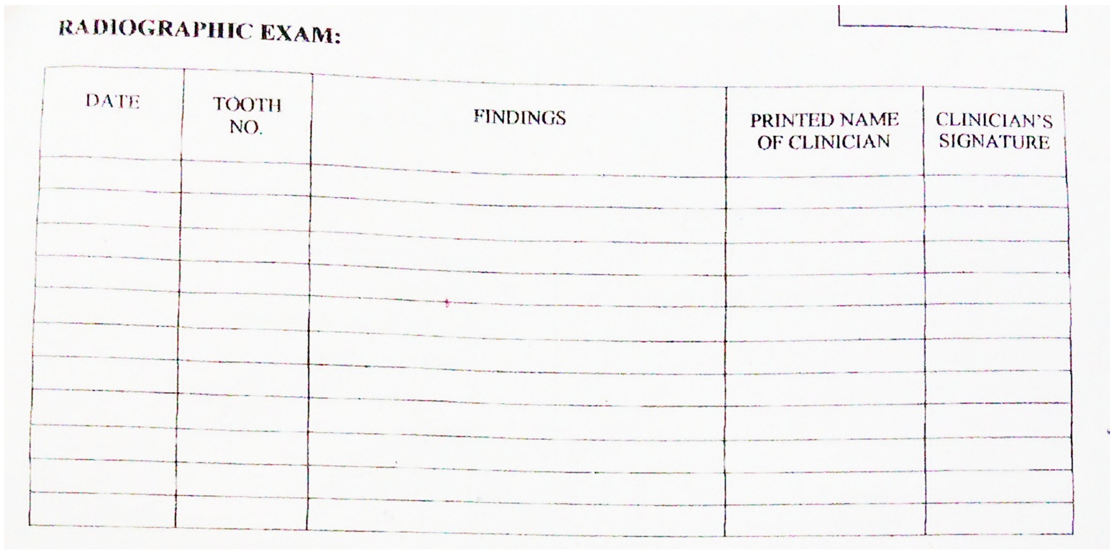
If stopped, how long since last used? __________

Figure 67: UPCD Admitting Section Patient Form with Social History


SOFT TISSUE EXAMINATION: Indicate lesions on drawings, describe, and date:

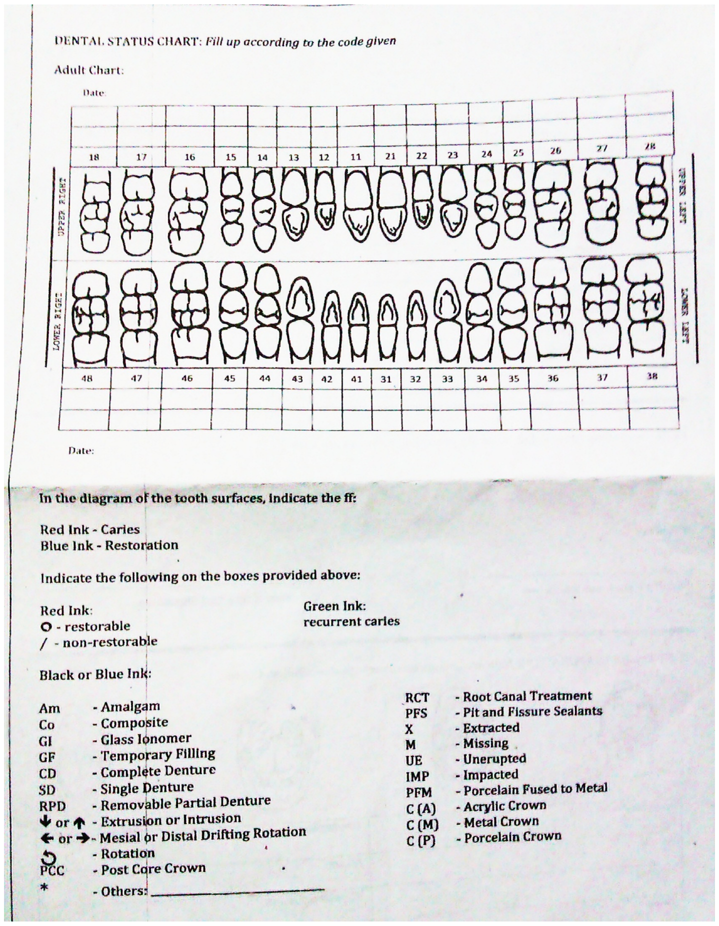
HEAD, NECK & TMJ	LIPS/FRENUM
MUCOSA	PALATE
PHARYNX	FLOOR OF THE MOUTH
TONGUE	LYMPH NODES
SALIVARY GLAND	THYROID
GINGIVA	

Figure 68: UPCD Soft Tissue Examination


RADIOGRAPHIC EXAM:

DATE	TOOTH NO.	FINDINGS	PRINTED NAME OF CLINICIAN	CLINICIAN'S SIGNATURE

Figure 69: UPCD Radiographic Examination



DENTAL STATUS CHART: *Fill up according to the code given*

Adult Chart:

Date:

18	17	16	15	14	13	12	11	21	22	23	24	25	26	27	28
48	47	46	45	44	43	42	41	31	32	33	34	35	36	37	38

UPPER RIGHT
UPPER LEFT
LOWER RIGHT
LOWER LEFT

Date:

In the diagram of the tooth surfaces, indicate the ff:

Red Ink - Caries
Blue Ink - Restoration

Indicate the following on the boxes provided above:

Red Ink:
O - restorable
/ - non-restorable

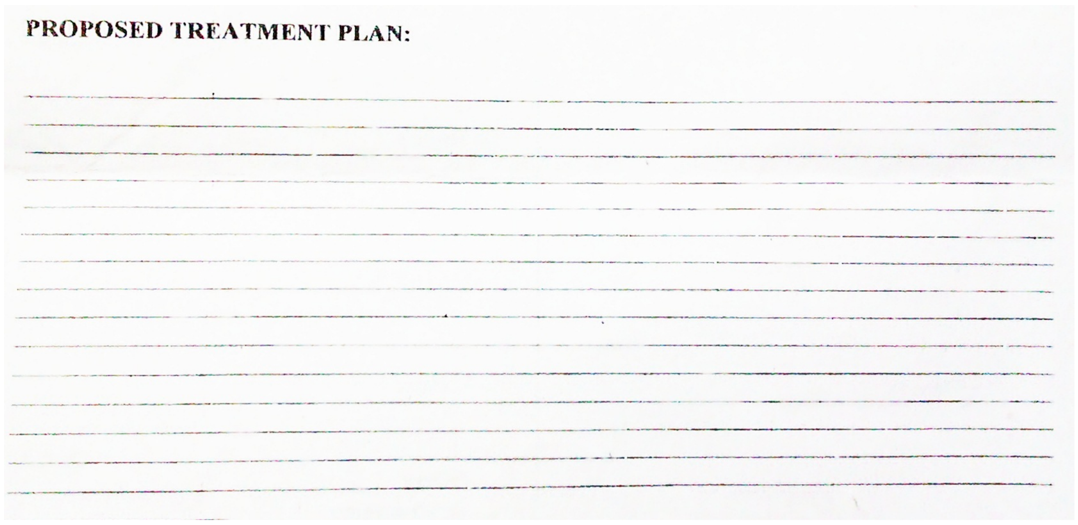
Green Ink:
recurrent caries

Black or Blue Ink:

Am	- Amalgam
Co	- Composite
GI	- Glass Ionomer
GF	- Temporary Filling
CD	- Complete Denture
SD	- Single Denture
RPD	- Removable Partial Denture
↓ or ↑	- Extrusion or Intrusion
← or →	- Mesial or Distal Drifting Rotation
↻	- Rotation
PCC	- Post Core Crown
*	- Others: ____________

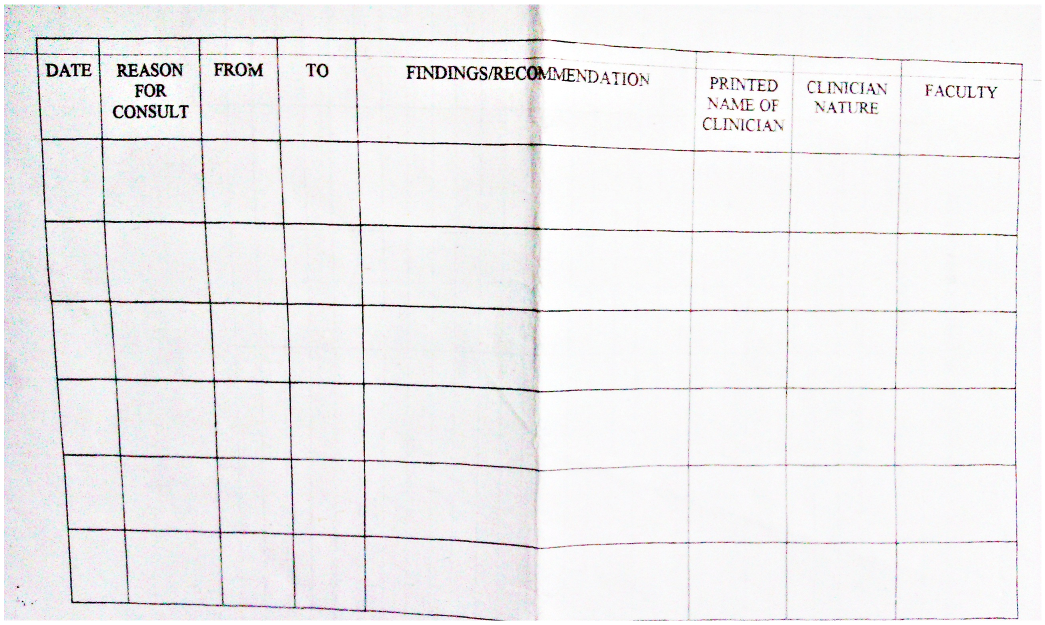
RCT	- Root Canal Treatment
PFS	- Pit and Fissure Sealants
X	- Extracted
M	- Missing
UE	- Unerupted
IMP	- Impacted
PFM	- Porcelain Fused to Metal
C (A)	- Acrylic Crown
C (M)	- Metal Crown
C (P)	- Porcelain Crown

Figure 70: UPCD Dental Status Chart



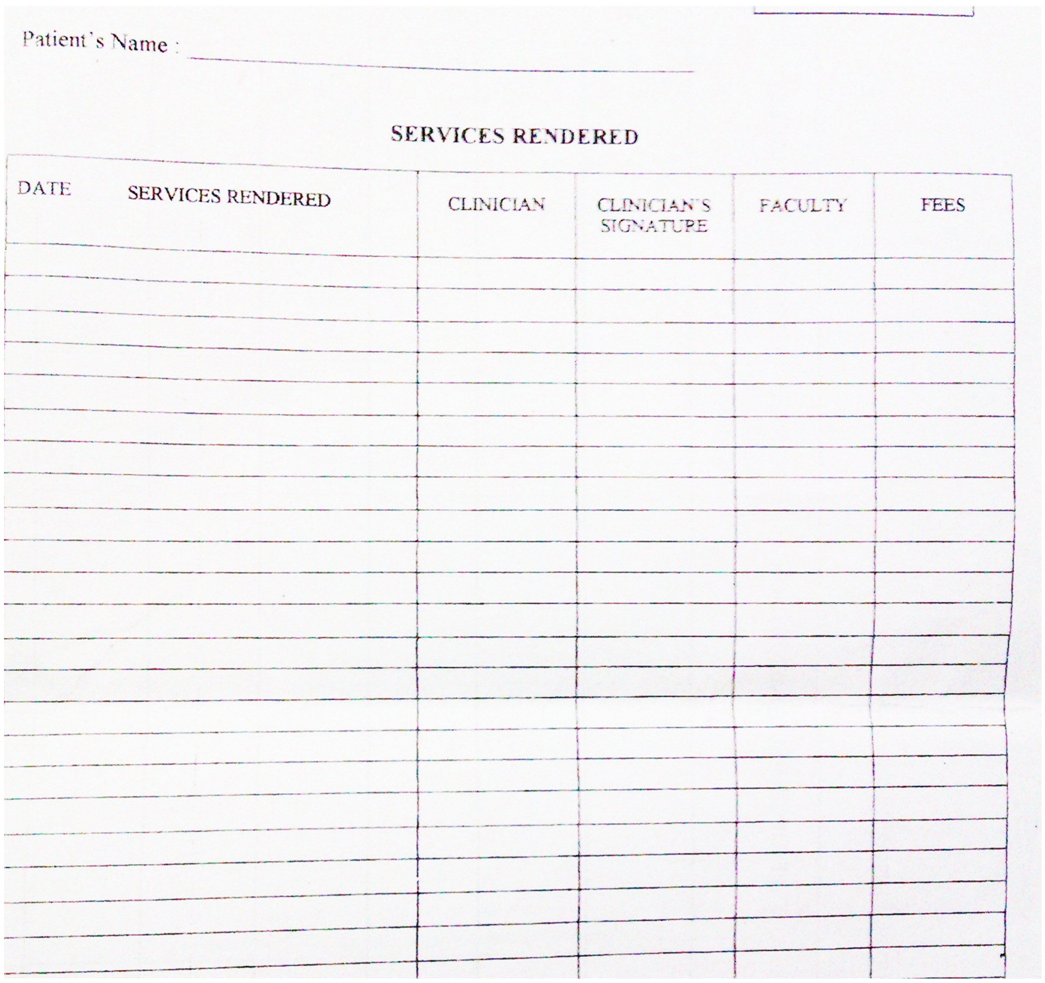
PROPOSED TREATMENT PLAN:

Figure 71: UPCD Proposed Treatment Plan



DATE	REASON FOR CONSULT	FROM	TO	FINDINGS/RECOMMENDATION	PRINTED NAME OF CLINICIAN	CLINICIAN NATURE	FACULTY

Figure 72: UPCD Consulatations/Referral

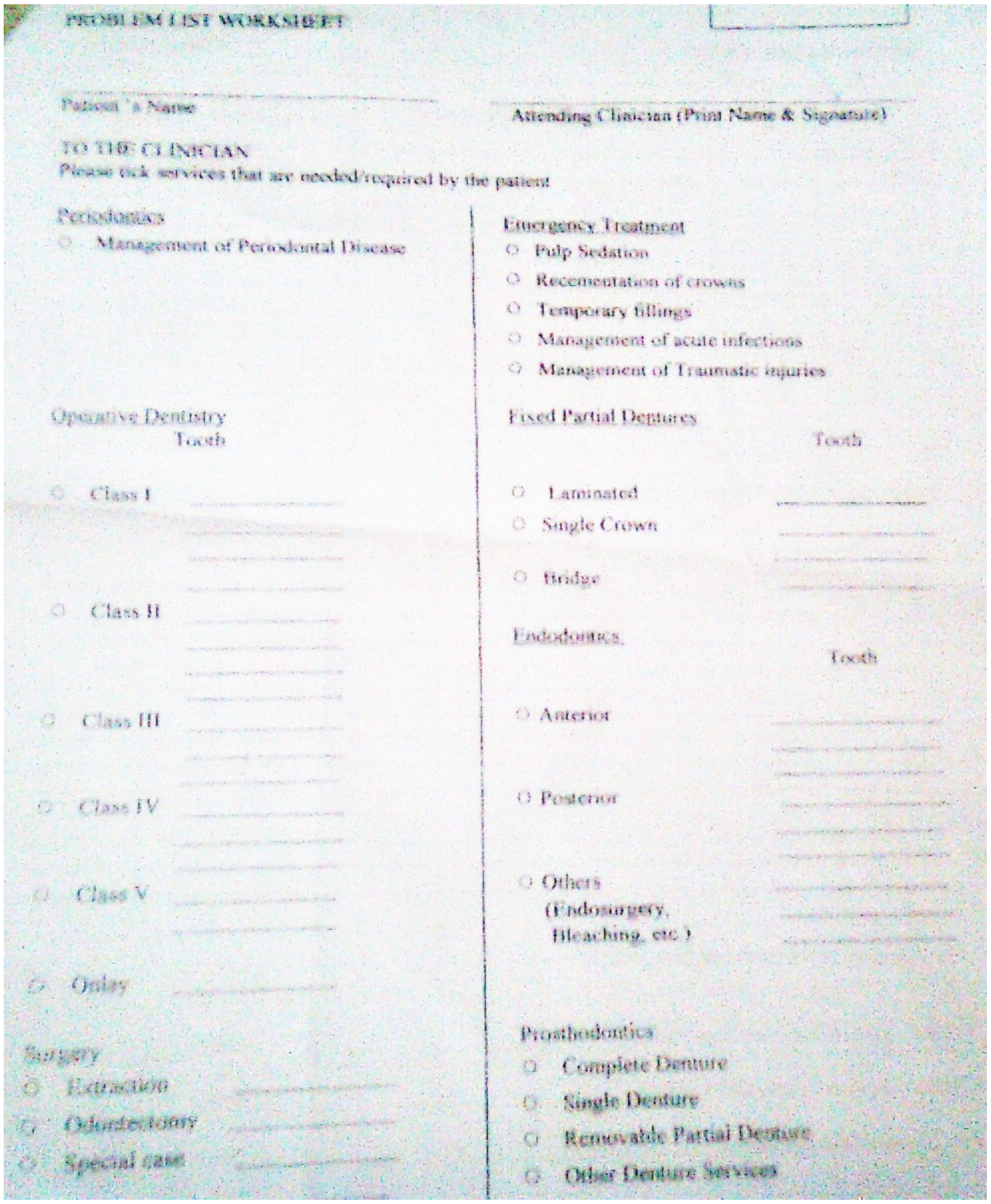


Patient's Name : ______________________________

SERVICES RENDERED

DATE SERVICES RENDERED	CLINICIAN	CLINICIAN'S SIGNATURE	FACULTY	FEES

Figure 73: UPCD Services Rendered



PROBLEM LIST WORKSHEET

Patient 's Name

Attending Clinician (Print Name & Signature)

TO THE CLINICIAN
Please tick services that are needed/required by the patient

Periodontics
○ Management of Periodontal Disease

Operative Dentistry
Tooth
○ Class I
○ Class II
○ Class III
○ Class IV
○ Class V
○ Onlay

Surgery
○ Extraction
○ Odontectomy
○ Special case

Emergency Treatment
○ Pulp Sedation
○ Recementation of crowns
○ Temporary fillings
○ Management of acute infections
○ Management of Traumatic injuries

Fixed Partial Dentures
Tooth
○ Laminated
○ Single Crown
○ Bridge

Endodontics
Tooth
○ Anterior
○ Posterior
○ Others
(Endosurgery,
Bleaching, etc.)

Prosthodontics
○ Complete Denture
○ Single Denture
○ Removable Partial Denture
○ Other Denture Services

Figure 74: UPCD Problem Worksheet

C. Source Code

```
package org.dentist.version.three.
     processserver;

import java.io.IOException;
import java.net.ServerSocket;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;

import javax.persistence.
     EntityManagerFactory;
import javax.persistence.Persistence;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory
     ;
import org.drools.SystemEventListenerFactory
     ;
import org.jbpm.task.Group;
import org.jbpm.task.User;
import org.jbpm.task.service.TaskService;
import org.jbpm.task.service.
     TaskServiceSession;
import org.jbpm.task.service.mina.
     MinaTaskServer;



public class HumanTaskStartupServlet extends
      HttpServlet {

  private static Log log = LogFactory.getLog
       (HumanTaskStartupServlet.class);
  public static EntityManagerFactory emfTask
       ;
  public static TaskService taskService;
  public static TaskServiceSession
       taskSession;
  public void init() throws ServletException
        {
  // if(!isAvailable(9123)){
  //  taskSession.dispose();
  // }
        super.init();

        /*
    * Start local h2 datbase
    * This is not required if the
        application connects to a remote
        database



   try {

    DeleteDbFiles.execute("", "JPADroolsFlow
        ", true);
     Server h2Server = Server.
         createTcpServer(new String[0]);
    h2Server.start();
   } catch (SQLException e) {
    log.error(e.getMessage(), e.getCause());
    throw new RuntimeException("can't start
        h2 server db",e);
   }
   */



   try {



    System.out.println("OK 1...");
   emfTask = Persistence.
        createEntityManagerFactory( "org.jbpm
        .task" );
    System.out.println("OK 2 ...");
   taskService = new TaskService(emfTask,
          SystemEventListenerFactory.
          getSystemEventListener());
      System.out.println("OK 3 ...");


    /*
          * Add the required users
          */
          taskSession = taskService.
               createSession();


         taskSession.addUser(new User("
             sophiabianca"));
         taskSession.addUser(new User("
             Administrator"));
         taskSession.addUser(new User("krisv
             "));
         taskSession.addUser(new User("john
             "));
         taskSession.addUser(new User("mary
             "));






         /* Start Mina server for HT*/
         MinaTaskServer server = new
              MinaTaskServer(taskService);

         Thread thread = new Thread(server);
         thread.start();

          System.out.println("OK 4 ...");
        log.debug("Mina Server started ...")
            ;
    } catch (Throwable t) {
    log.error(t.getMessage(), t.getCause());
     throw new RuntimeException("can't start
          Mina server",t);

    }

   }

     public void destroy() {
        super.destroy();
      }

      protected void doGet(
           HttpServletRequest request,
           HttpServletResponse response)
        throws ServletException, IOException
      {
        doPost(request, response);
      }

      protected void doPost(
           HttpServletRequest reqquest,
           HttpServletResponse response)
        throws ServletException, IOException
      {
        response.sendError(1001, "POST
            Method Not Allowed Here");
      }
public boolean isAvailable(int port){
 boolean portTaken = false;
    ServerSocket socket = null;
    try {
        socket = new ServerSocket(port);
    } catch (IOException e) {
        portTaken = true;
    } finally {
        if (socket != null)
            try {
                socket.close();
            } catch (IOException e) { /* e.
                 printStackTrace(); */ }
}
    return portTaken;
}
}
```

```
package org.dentist.version.three.
     processserver;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.net.URL;
import java.net.URLClassLoader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.persistence.
     EntityManagerFactory;
import javax.persistence.Persistence;
import javax.transaction.
     HeuristicMixedException;
import javax.transaction.
     HeuristicRollbackException;
import javax.transaction.RollbackException;
import javax.transaction.SystemException;
import javax.transaction.UserTransaction;

import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.web.
     controller.SearchController;
import org.dentist.version.three.web.
     controller.TaskController;
import org.dentist.version.three.web.
     controller.ViewController;
import org.drools.KnowledgeBase;
import org.drools.KnowledgeBaseFactory;
import org.drools.SystemEventListenerFactory
     ;
import org.drools.base.MapGlobalResolver;
import org.drools.builder.KnowledgeBuilder;
import org.drools.builder.
     KnowledgeBuilderFactory;
import org.drools.builder.ResourceType;
import org.drools.compiler.
     BPMN2ProcessFactory;
import org.drools.compiler.
     ProcessBuilderFactory;
import org.drools.io.ResourceFactory;
import org.drools.logger.
     KnowledgeRuntimeLogger;
import org.drools.logger.
     KnowledgeRuntimeLoggerFactory;
import org.drools.marshalling.impl.
     ProcessMarshallerFactory;
import org.drools.persistence.jpa.
     JPAKnowledgeService;
import org.drools.runtime.Environment;
import org.drools.runtime.EnvironmentName;
import org.drools.runtime.
     KnowledgeSessionConfiguration;
import org.drools.runtime.
     StatefulKnowledgeSession;
import org.drools.runtime.process.
     ProcessRuntimeFactory;
import org.jbpm.bpmn2.
     BPMN2ProcessProviderImpl;
import org.jbpm.marshalling.impl.
     ProcessMarshallerFactoryServiceImpl;
import org.jbpm.process.builder.
     ProcessBuilderFactoryServiceImpl;
import org.jbpm.process.instance.
     ProcessRuntimeFactoryServiceImpl;
import org.jbpm.task.AccessType;
import org.jbpm.task.Task;
import org.jbpm.task.query.TaskSummary;
import org.jbpm.task.service.ContentData;
import org.jbpm.task.service.TaskClient;
import org.jbpm.task.service.
     TaskClientHandler;
import org.jbpm.task.service.mina.
     MinaTaskClientConnector;
import org.jbpm.task.service.mina.
     MinaTaskClientHandler;
import org.jbpm.task.service.
     responsehandlers.
     BlockingGetTaskResponseHandler;
import org.jbpm.task.service.
     responsehandlers.
     BlockingTaskOperationResponseHandler;
import org.jbpm.task.service.
     responsehandlers.
     BlockingTaskSummaryResponseHandler;

import bitronix.tm.
     TransactionManagerServices;

public class JbpmAPIUtil {

 private static String ipAddress =
      "127.0.0.1";
 private static int port = 9123;
 public static TaskClient client;
 private static Map<String, List<String>>
      groupListMap = new HashMap<String, List
      <String>>();
 private static StatefulKnowledgeSession
      ksession;


 public  void setConnection(String ipAddress
      , int port) {
  this.ipAddress = ipAddress;
  this.port = port;
 }

 /*
  * Connect to Mina client
  */
 public static void connect() {
  if (client == null) {
   client = new TaskClient(new
        MinaTaskClientConnector("org.drools.
        process.workitem.wsht.
        WSHumanTaskHandler",
         new MinaTaskClientHandler(
              SystemEventListenerFactory.
              getSystemEventListener())));
   boolean connected = client.connect(
        ipAddress, port);
   if (!connected) {
    throw new IllegalArgumentException(
     "Could not connect task client");
   }
  }
  try {
   ClassLoader loader = Thread.currentThread
        ().getContextClassLoader();
   URL url = null;
   String propertyName = "roles.properties";

   if (loader instanceof URLClassLoader) {
    URLClassLoader ucl = (URLClassLoader)
         loader;
    url = ucl.findResource(propertyName);
   }
   if (url == null) {
    url = loader.getResource(propertyName);
   }
   if (url == null) {
    System.out.println("No properties file:
         " + propertyName + " found");
   } else {
    Properties bundle = new Properties();
    InputStream is = url.openStream();
    if (is != null) {
     bundle.load(is);
     is.close();
    } else {
     throw new IOException("Properties file
          " + propertyName + " not available
          ");
    }
    Enumeration<?> propertyNames = bundle.
         propertyNames();
    while (propertyNames.hasMoreElements())
         {
     String key = (String) propertyNames.
          nextElement();
     String value = bundle.getProperty(key);
     groupListMap.put(key, Arrays.asList(
          value.split(",")));
     System.out.print("Loaded user " + key +
           ":");
     for (String role: groupListMap.get(key)
          ) {
      System.out.print(" " + role);
      }
      System.out.println();
     }
    }
   } catch (Throwable t) {
    t.printStackTrace();
   }

  }

  /*
   * Get all the tasks assigned to a user
   */
  public static List <TaskSummary>
       getAssignedTasks(String idRef) {
   connect();
   List<TaskSummary> tasks = null;
   try {
    BlockingTaskSummaryResponseHandler
         responseHandler = new
         BlockingTaskSummaryResponseHandler();
    client.getTasksAssignedAsPotentialOwner(
         idRef, "en-UK", responseHandler);
           tasks = responseHandler.getResults
                ();
   } catch (Throwable t) {
    t.printStackTrace();
   }
   return tasks;
  }
  //get group tasks
  /*
     public static List <TaskSummary>
          getAssignedGroupTasks(String idRef,
          ArrayList<String> groups) throws
          Exception{
      UserTransaction ut = (UserTransaction)
           new InitialContext().lookup( "java:
           comp/UserTransaction" );
      System.out.println("OK Submit 1 ...");
      ut.begin();
      JbpmAPIUtil.connect();
   List<TaskSummary> tasks = null;
   //List<String> group= new ArrayList<String
        >();
   //  group.add("DentIStOralDiagStudent");
   try {

    BlockingTaskSummaryResponseHandler
         responseHandler = new
         BlockingTaskSummaryResponseHandler();
    JbpmAPIUtil.client.
         getTasksAssignedAsPotentialOwner(
         idRef, groups,"en-UK",
         responseHandler);
           tasks = responseHandler.getResults
                ();

   } catch (Throwable t) {
    t.printStackTrace();
   }
   ut.commit();
   return tasks;
  }
  */
  public static List <Task>
       getAssignedGroupTasks(String idRef,
       ArrayList<String> groups) throws
       Exception{
      UserTransaction ut = (UserTransaction)
           new InitialContext().lookup( "java:
           comp/UserTransaction" );
      System.out.println("OK Submit 1 ...");
      ut.begin();
      JbpmAPIUtil.connect();
   List<Task> tasks = new ArrayList<Task>();
   ArrayList<String>ids= new ArrayList<String
        >();
   //List<String> group= new ArrayList<String
        >();
   //  group.add("DentIStOralDiagStudent");
   try {

    BlockingGetTaskResponseHandler
         responseHandler = new
         BlockingGetTaskResponseHandler();
    //JbpmAPIUtil.client.
         getTasksAssignedAsPotentialOwner(
         idRef, groups,"en-UK",
         responseHandler);
           //tasks = responseHandler.
                getResults();
    ids=JbpmAPIUtil.gettasks(idRef);
    for (String id : ids) {
     responseHandler = new
          BlockingGetTaskResponseHandler();
     JbpmAPIUtil.client.getTask(Long.
          parseLong(id), responseHandler);
     Task task = responseHandler.getTask();
     tasks.add(task);
     System.out.println("TASK YAN: "+task.
          getId());
       }
    for (String group: groups) {
     ids=JbpmAPIUtil.gettasks(group);
     for (String id : ids) {
      responseHandler = new
           BlockingGetTaskResponseHandler();
      JbpmAPIUtil.client.getTask(Long.
           parseLong(id), responseHandler);
      Task task = responseHandler.getTask();
      tasks.add(task);
      System.out.println("TASK YAN: "+task.
           getId());
        }
       }

   } catch (Throwable t) {
    t.printStackTrace();
   }
   ut.commit();
   for (Task task : tasks) {
    System.out.println("TASK ETO: "+task.
         getId());
      }
   return tasks;
  }
     public static void claimTask(long taskid
          , String idRef, ArrayList<String>
          groups) throws Exception {
      UserTransaction ut = (UserTransaction)
           new InitialContext().lookup( "java:
           comp/UserTransaction" );
      System.out.println("OK Submit 1 ...");
      ut.begin();
      JbpmAPIUtil.connect();
   List<TaskSummary> tasks = null;
   //List<String> group= new ArrayList<String
        >();
   //  group.add("DentIStOralDiagStudent");
   try {
    BlockingTaskOperationResponseHandler
         responseHandler = new
         BlockingTaskOperationResponseHandler
         ();
    JbpmAPIUtil.client.claim(taskid,idRef,
         groups, responseHandler);
    responseHandler.waitTillDone(50000);
   } catch (Throwable t) {
    t.printStackTrace();
   }
   ut.commit();

  }
         /*
   * Complete a task with a taskid and data
        for a user
   */
  public static void completeTask(long taskId
       , Map data, String userId) throws
       InterruptedException {
   connect();

   BlockingTaskOperationResponseHandler
        responseHandler = new
        BlockingTaskOperationResponseHandler()
        ;
   client.start(taskId, userId,
        responseHandler);
   responseHandler.waitTillDone(5000);
   //Thread.sleep(10000);
   responseHandler = new
        BlockingTaskOperationResponseHandler()
        ;
   ContentData contentData = new ContentData
        ();
   /*
   if (data != null) {
    ByteArrayOutputStream bos = new
         ByteArrayOutputStream();
    ObjectOutputStream out;
    try {
     out = new ObjectOutputStream(bos);
     out.writeObject(data);
```

```
      out.close();
      contentData = new ContentData();
      contentData.setContent(bos.toByteArray()
          );
      contentData.setAccessType(AccessType.
          Inline);
    } catch (IOException e) {
      e.printStackTrace();
    }
   }
   */
   client.complete(taskId, userId,
       contentData, responseHandler);
   responseHandler.waitTillDone(5000);
  }


/*
  * This is similar to 'completeTask' method
      , but to complete a task that is in '
      Progress' state.
  *  In this case client start method is not
       called
  */

 public static void completeProgressTask(
     long taskId, Map data, String userId)
     throws InterruptedException {
  connect();

  BlockingTaskOperationResponseHandler
      responseHandler = new
      BlockingTaskOperationResponseHandler()
      ;
  responseHandler.waitTillDone(5000);
  //Thread.sleep(10000);
  responseHandler = new
      BlockingTaskOperationResponseHandler()
      ;
  ContentData contentData = null;
  if (data != null) {
   ByteArrayOutputStream bos = new
       ByteArrayOutputStream();
   ObjectOutputStream out;
   try {
    out = new ObjectOutputStream(bos);
    out.writeObject(data);
    out.close();
    contentData = new ContentData();
    contentData.setContent(bos.toByteArray()
        );
    contentData.setAccessType(AccessType.
        Inline);
   } catch (IOException e) {
    e.printStackTrace();
   }
  }
  client.complete(taskId, userId,
      contentData, responseHandler);
  responseHandler.waitTillDone(5000);
 }

 /*
  * Assign the task to a user
  */
 public void assignTask(long taskId, String
     idRef, String userId) {
  connect();
  BlockingTaskOperationResponseHandler
      responseHandler = new
      BlockingTaskOperationResponseHandler()
      ;
  if (idRef == null) {
   client.release(taskId, userId,
       responseHandler);
  } else if (idRef.equals(userId)) {
   List<String> roles = groupListMap.get(
       userId);
   if (roles == null) {
    client.claim(taskId, idRef,
        responseHandler);
   } else {
    client.claim(taskId, idRef, roles,
        responseHandler);
   }
  } else {
   client.delegate(taskId, userId, idRef,
       responseHandler);
  }
  responseHandler.waitTillDone(5000);
 }
```

```
/*
 * Load the bpmn file into knowledgebase
 */
public static KnowledgeBase
    readKnowledgeBase(String process)
    throws Exception {
 ProcessBuilderFactory.
     setProcessBuilderFactoryService(new
     ProcessBuilderFactoryServiceImpl());
 ProcessMarshallerFactory.
     setProcessMarshallerFactoryService(new
      ProcessMarshallerFactoryServiceImpl()
     );
 ProcessRuntimeFactory.
     setProcessRuntimeFactoryService(new
     ProcessRuntimeFactoryServiceImpl());
 BPMN2ProcessFactory.
     setBPMN2ProcessProvider(new
     BPMN2ProcessProviderImpl());
 KnowledgeBuilder kbuilder =
     KnowledgeBuilderFactory.
     newKnowledgeBuilder();
 kbuilder.add(ResourceFactory.
     newClassPathResource(process),
     ResourceType.BPMN2);
 return kbuilder.newKnowledgeBase();
}




public static StatefulKnowledgeSession
    getSession() throws Exception {
 if (ksession == null) {
  ksession = createSession();
 }
 return ksession;
}


/*
 * Create EntityManagerFactory and register
      it in the environment
 * Create the knowledge session that uses
     JPA to persists runtime state
 */

public static StatefulKnowledgeSession
    createSession() throws Exception {

 /*
  * Create the knowledgebase using the
      required bpmn and drl files
  */

 KnowledgeBase kbase = readKnowledgeBase("
     SampleHumanTaskFormVariables.bpmn2");
 System.out.println("OK jpa 1...");
 EntityManagerFactory emf = Persistence.
     createEntityManagerFactory( "org.jbpm.
     persistence.jpa" );
 System.out.println("OK jpa 2...");
 Environment env = KnowledgeBaseFactory.
     newEnvironment();
 env.set( EnvironmentName.
     ENTITY_MANAGER_FACTORY, emf );
 env.set( EnvironmentName.
     TRANSACTION_MANAGER,
     TransactionManagerServices.
     getTransactionManager() );
 env.set( EnvironmentName.GLOBALS, new
     MapGlobalResolver() );

 Properties properties = new Properties();
 properties.put("drools.
     processInstanceManagerFactory", "org.
     jbpm.persistence.processinstance.
     JPAProcessInstanceManagerFactory");
 properties.put("drools.
     processSignalManagerFactory", "org.
     jbpm.persistence.processinstance.
     JPASignalManagerFactory");
 KnowledgeSessionConfiguration config =
     KnowledgeBaseFactory.
     newKnowledgeSessionConfiguration(
     properties);

 System.out.println("OK jpa 3...");
```

```
  //StatefulKnowledgeSession ksession =
  return JPAKnowledgeService.
       newStatefulKnowledgeSession(kbase,
       config, env);
  //return JPAKnowledgeService.
       newStatefulKnowledgeSession(1, kbase,
       config, env);

 }
 public static ArrayList<String> gettasks(
      String entity) throws Exception{
  //list of patients
  ArrayList<String> taskids= new ArrayList<
       String>();
  HashMap<String,String> map=TaskController.
       formlist;
     if(map.isEmpty()){
      map=SearchController.formlist;
      if(map.isEmpty()){
       map=ViewController.formlist;
      }
     }
     String names="(";
     for (String key : map.keySet()) {
   names=names+"'"+key+"',";
     }
     names=names.substring(0,names.length()
          -1);
     names=names+")";
   //int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);

    try{
      String update="";

      update="SELECT * FROM task AS A inner
           join i18ntext AS B ON A.id=B.
           task_names_id inner join
           peopleassignments_potowners AS C ON
            B.task_names_id=C.task_id WHERE A.
           processinstanceid!=1 AND C.
           entity_id='"+entity+"' AND A.status
           !='Completed' AND B.text NOT IN"+
           names;

      System.out.println("update statement="+
           update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();
    DentFormsSP dentforms=new DentFormsSP();
    while(rs.next()){

         String taskid= rs.getString("
              task_names_id");
       taskids.add(taskid);
     System.out.println("TASK YUN: "+taskid
          );
     }

    System.out.println("Forms started");


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }
    return taskids;
 }
}
```

```
package org.dentist.version.three.taskserver
     .mgr;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.text.MessageFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.ResourceBundle;
import java.util.Set;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.
     UrlEncodedFormEntity;
import org.apache.http.client.methods.
     HttpGet;
import org.apache.http.client.methods.
     HttpPost;
import org.apache.http.conn.
     ClientConnectionManager;
import org.apache.http.entity.mime.
     HttpMultipartMode;
import org.apache.http.entity.mime.
     MultipartEntity;
import org.apache.http.entity.mime.content.
     StringBody;
import org.apache.http.impl.client.
     DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.
     ThreadSafeClientConnManager;
import org.apache.http.message.
     BasicNameValuePair;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;

/**
 *
 * @author esanchez
 *
 */
public class JBPMRestManagementClient {
 protected String KEY_USERNAME = "j_username
      ";
 protected String KEY_PASSWORD = "j_password
      ";
 private DefaultHttpClient httpClient; //
      keep this out of the method in order to
       reuse the object for calling other
      services without losing session
 protected String address;
 protected ResourceBundle bundle;
 protected String host;
    protected String urlForm(String prop)
         throws Exception{
     host=InetAddress.getLocalHost().
          getCanonicalHostName();
     return "http://"+host+":"+bundle.
          getString("process.port")+bundle.
          getString("process.urlContext")+
          bundle.getString(prop);
    }

    protected String urlForm(String prop,
         String ... params) throws Exception{
     host=InetAddress.getLocalHost().
          getCanonicalHostName();
     String msg = bundle.getString(prop);
     msg = MessageFormat.format(msg, params)
          ;
     return "http://"+host+":"+bundle.
          getString("process.port")+bundle.
          getString("process.urlContext")+msg
          ;
    }

    public JBPMRestManagementClient(){
     if(this.bundle == null)
      this.bundle = ResourceBundle.getBundle
           ("processResources");
     this.init(this.bundle.getString("
          process.user"), this.bundle.
          getString("process.password"));
    }

    public JBPMRestManagementClient(String
         username, String password){
  this.init(username, password);
    }

    protected void relogin(String username,
        String password)throws Exception{
      String url = urlForm("user.management.
          invalidate");
      this.requestPostService(url, null);
      url = urlForm("user.management.secure.
          sid");
      requestGetService(url, null);
      this.init(username, password);

     }

     private void init(String username,
         String password){
      if(this.bundle == null)
       this.bundle = ResourceBundle.getBundle
           ("processResources");
      if(httpClient == null)
       httpClient = getThreadSafeClient();
   this.address = bundle.getString("process.
       host");
   try{
    String url = urlForm("user.management.
        secure.sid");
    String resp = this.requestPostService(url
        , null);
       this.authenticate(username, password);
       String lol = this.requestPostService(
           url, null);
      System.out.println(lol);
   }catch (Exception e) {
    e.printStackTrace();
   }
     }

// public JBPMRestManagementClient(String
     host,String port){
//      if(this.bundle == null)
//       this.bundle = ResourceBundle.
     getBundle("processResources");
//      if(httpClient == null)
//       httpClient = new DefaultHttpClient()
     ;
//   this.address = host+":"+port;
//   }

  private String authenticate(String username
      , String password) throws Exception {

   String responseString = "";
   List<NameValuePair> formparams = new
       ArrayList<NameValuePair>();
   formparams.add(new BasicNameValuePair(
       KEY_USERNAME, username));
   formparams.add(new BasicNameValuePair(
       KEY_PASSWORD, password));
   String urlAuth = urlForm("process.
       j_security_check");
   HttpPost httpPost = new HttpPost(urlAuth);
   InputStreamReader inputStreamReader = null
       ;
   BufferedReader bufferedReader = null;
   try {
    UrlEncodedFormEntity entity = new
        UrlEncodedFormEntity(formparams, "UTF
        -8");//UrlEncodedFormEntity(
        formparams, "multipart/form-data");
    httpPost.setEntity(entity);
    HttpResponse response = httpClient.
        execute(httpPost);
    InputStream inputStream = response.
        getEntity().getContent();
    EntityUtils.consume(response.getEntity())
        ;
    inputStreamReader = new InputStreamReader
        (inputStream);
    bufferedReader = new BufferedReader(
        inputStreamReader);
    StringBuilder stringBuilder = new
        StringBuilder();
    String line = bufferedReader.readLine();
    while (line != null) {
     stringBuilder.append(line);
     line = bufferedReader.readLine();
    }
    responseString = stringBuilder.toString()
        ;

   } catch (Exception e) {
   throw new RuntimeException(e);

   } finally {
```

```
 if (inputStreamReader != null) {
  try {
   inputStreamReader.close();

 } catch (Exception e) {

 throw new RuntimeException(e);

 }

 }

 if (bufferedReader != null) {
  try {
   bufferedReader.close();
  } catch (Exception e) {
   throw new RuntimeException(e);
  }
 }
 }
 return responseString;
 }

private String requestPostMultipartService(
    String url, Map<String, Object>
    parameters) throws Exception{
 String responseString = "";
 HttpPost httpPost = new HttpPost(url);
 if (parameters == null)
  parameters = new HashMap<String, Object
      >();

 MultipartEntity entity = new
     MultipartEntity(HttpMultipartMode.
     BROWSER_COMPATIBLE);

 Set<String> keys = parameters.keySet();
 for (Iterator<String> keysIterator = keys.
     iterator(); keysIterator.hasNext();) {
  String keyString = keysIterator.next();
  String value = parameters.get(keyString).
      toString();
  entity.addPart(keyString, new StringBody(
      value));
 }
 httpPost.setEntity(entity);
 HttpResponse response = httpClient.execute
     (httpPost);
 responseString = this.getRequestString(
     response);
 EntityUtils.consume(response.getEntity());
 return responseString;
}

private byte[] getRequestByteArray(
    HttpResponse response)throws Exception{
 InputStream is = response.getEntity().
     getContent();
    int len;
    int size = 1024;
    byte[] buf;

    if (is instanceof ByteArrayInputStream)
        {
      size = is.available();
      buf = new byte[size];
      len = is.read(buf, 0, size);
    } else {
      ByteArrayOutputStream bos = new
          ByteArrayOutputStream();
      buf = new byte[size];
      while ((len = is.read(buf, 0, size))
          != -1)
        bos.write(buf, 0, len);
      buf = bos.toByteArray();
    }
 return buf;
}


private String getRequestString(
    HttpResponse response)throws Exception{
 InputStreamReader inputStreamReader = null
     ;
 BufferedReader bufferedReader = null;
 String req = "";
 InputStream inputStream = response.
     getEntity().getContent();
 inputStreamReader = new InputStreamReader(
     inputStream);
 bufferedReader = new BufferedReader(
     inputStreamReader);
 StringBuilder stringBuilder = new
      StringBuilder();
 String line = bufferedReader.readLine();
 while (line != null) {
  stringBuilder.append(line);
  line = bufferedReader.readLine();
 }
 req = stringBuilder.toString();
 return req;
}

private String implode(String[] ary, String
      delim) {
     String out = "";
     for(int i=0; i<ary.length; i++) {
         if(i!=0) { out += delim; }
         out += ary[i];
     }
     return out;
}



 private  HttpResponse getResponseGET(
      String url, Map<String, Object>
      parameters) throws Exception{
  List<NameValuePair> formparams = new
       ArrayList<NameValuePair>();
  if (parameters == null)
   parameters = new HashMap<String, Object
        >();

  Set<String> keys = parameters.keySet();
  int i = 0;
  String[] a = null;
  if(parameters.size() > 0){
   url = url + "?";
   a = new String[parameters.size()];
  }
  for (Iterator<String> keysIterator = keys
       .iterator(); keysIterator.hasNext();)
        {
   String keyString = keysIterator.next();
   String value = parameters.get(keyString)
        .toString();
   a[i++] = keyString + "=" + value;
  }
  if(parameters.size() > 0){
   String implode = this.implode(a, "&");
   url = url + implode;
  }

  HttpGet httpGet = new HttpGet(url);
  HttpResponse response = httpClient.
       execute(httpGet);
 // EntityUtils.consume(response.getEntity
      ());
  return response;
 }

private InputStream
     getBytesFromRestGetService(String url)
     throws Exception{
  HttpResponse resp = this.getResponseGET(
       url, null);
  InputStream inresp = resp.getEntity().
       getContent();
  EntityUtils.consume(resp.getEntity());
  return inresp;
}

private String requestGetService(String url
     , Map<String, Object> parameters)
     throws Exception{
 HttpResponse response = this.
      getResponseGET(url, parameters);
 String responseString = "";
 responseString = this.getRequestString(
      response);
 EntityUtils.consume(response.getEntity());
 return responseString;
}

private String requestPostService(String
     url, Map<String, Object> parameters)
     throws Exception{

 String responseString = "";
 List<NameValuePair> formparams = new
      ArrayList<NameValuePair>();
 if (parameters == null)
  parameters = new HashMap<String, Object
        >();

  Set<String> keys = parameters.keySet();
  for (Iterator<String> keysIterator = keys.
       iterator(); keysIterator.hasNext();) {
   String keyString = keysIterator.next();
   String value = parameters.get(keyString).
        toString();
   formparams.add(new BasicNameValuePair(
        keyString, value));
  }

  HttpPost httpPost = new HttpPost(url);
  UrlEncodedFormEntity entity = new
       UrlEncodedFormEntity(formparams, "UTF
       -8");
  httpPost.setEntity(entity);
  HttpResponse response = httpClient.execute
       (httpPost);

  responseString = this.getRequestString(
       response);
  EntityUtils.consume(response.getEntity());
  return responseString;

  }


 protected String getDataFromRestService(
      String url, EnumJBPMRestType enumRest)
      throws Exception{
  return this.getDataFromRestService(url,
       enumRest, null);
 }

 protected InputStream
      getBytesFromRestService(String url,
      EnumJBPMRestType enumRest)throws
      Exception{

  InputStream bt = null;
//  String responseString = this.
     requestGetService(url, null);
//      if(responseString.contains("
     j_security_check")){
//       this.authenticate(userName, password
     );
//       bt = this.getBytesFromRestGetService
     (url);
//      }else
       bt = this.getBytesFromRestGetService(
            url);
  return bt;
 }


 protected String getDataFromRestService(
      String url, EnumJBPMRestType enumRest,
      Map<String, Object> parameters)throws
      Exception{
  String json = "";

  if(EnumJBPMRestType.MULTIPART.codigo.
       equals(enumRest.getCodigo())){
   String responseString = this.
        requestPostMultipartService(url,
        parameters);
//    if(responseString.contains("
     j_security_check")){
//        this.authenticate(userName,
     password);
//         json = this.
     requestPostMultipartService(url,
     parameters);
//       }else
        json = responseString;
  }
  if(EnumJBPMRestType.POST.codigo.equals(
       enumRest.getCodigo())){
   String responseString = this.
        requestPostService(url, parameters);
//    if(responseString.contains("
     j_security_check")){
//        this.authenticate(userName,
     password);
//         json = this.requestPostService(url,
      parameters);
//       }else
        json = responseString;
  }
  if(EnumJBPMRestType.GET.codigo.equals(
       enumRest.getCodigo())){
```

```
   String responseString = this.
        requestGetService(url, parameters);
//    if(responseString.contains("
     j_security_check")){
//        this.authenticate(userName,
     password);
//        json = this.requestGetService(url,
     parameters);
//       }else
         json = responseString;
  }

  return json;
 }

 public static DefaultHttpClient
      getThreadSafeClient() {

      DefaultHttpClient client = new
          DefaultHttpClient();
     ClientConnectionManager mgr = client.
          getConnectionManager();
     HttpParams params = client.getParams();

     client = new DefaultHttpClient(
         new ThreadSafeClientConnManager(
             params,
             mgr.getSchemeRegistry()),
                 params);

     return client;
 }
    public static void main(String[] args)
        throws Exception {


     }
}
```

```
package org.dentist.version.three.taskserver
     .mgr;


public enum EnumJBPMRestType {

 POST ("POST"),
 GET  ("GET"),
 MULTIPART ("MULTIPART"),
 GET_BYTE ("GET_BYTE")
 ;

 public String codigo;

 private EnumJBPMRestType(String codigo) {
  this.codigo = codigo;
 }

 public String getCodigo() {
  return codigo;
 }

}
```

```
package org.dentist.version.three.
     processserver.service;


import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.lang.reflect.Type;
import java.net.InetAddress;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.servlet.http.HttpSession;

import org.apache.commons.lang.StringUtils;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.
     UrlEncodedFormEntity;
import org.apache.http.client.methods.
     HttpGet;
import org.apache.http.entity.mime.
     HttpMultipartMode;
import org.apache.http.entity.mime.
     MultipartEntity;
import org.apache.http.entity.mime.content.
     ContentBody;
import org.apache.http.entity.mime.content.
     StringBody;
import org.apache.http.impl.client.
     DefaultHttpClient;
import org.apache.http.message.
     BasicNameValuePair;

import com.google.gson.Gson;
import com.google.gson.JsonParseException;
import com.google.gson.JsonParser;
import com.google.gson.reflect.TypeToken;



import org.dentist.version.three.taskserver.
     mgr.EnumJBPMRestType;
import org.dentist.version.three.taskserver.
     mgr.JBPMRestManagementClient;
import org.dentist.version.three.
     processserver.model.ActiveNodeInfoRS;
import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.InstancesRS;
import org.dentist.version.three.
     processserver.model.
     ProcessDefinitionInstancesRS;
import org.dentist.version.three.
     processserver.model.ProcessDefinitionsRS
     ;
import org.dentist.version.three.
     processserver.model.TaskRS;
import org.dentist.version.three.
     processserver.model.TaskUserRS;
import org.dentist.version.three.
     processserver.model.UserTaskVO;
import org.jbpm.task.query.TaskSummary;
import org.jbpm.task.service.
     responsehandlers.
     BlockingTaskSummaryResponseHandler;
import org.dentist.version.three.
     processserver.JbpmAPIUtil;

/**
 *
 * @author esanchez
 *
 */
public class Process extends
     JBPMRestManagementClient{

 static Process self;

    public static Process instance(){
        if(self == null){
         self = new Process();
        }
        return self;
    }

    public Process() {
     super();
 }

    public Process(String username, String
         password) {
     super(username, password);
 }
    //ADDED: AUTHENTICATION
 public static String KEY_USERNAME = "
      j_username";
public static String KEY_PASSWORD = "
     j_password";
private DefaultHttpClient httpClient = new
     DefaultHttpClient(); // keep this out of
      the method in order to reuse the object
      for calling other services without
     losing session


public void authenticate(String address,
     String username, String password) throws
      Exception{
        String responseString = "";
        List<NameValuePair> formparams = new
             ArrayList<NameValuePair>();
        formparams.add(new
            BasicNameValuePair(KEY_USERNAME,
             username));
        formparams.add(new
            BasicNameValuePair(KEY_PASSWORD,
             password));


        HttpGet httpGet = new HttpGet("http
            ://"+InetAddress.getLocalHost().
            getCanonicalHostName()+":8090/
            gwt-console-server/rs/process/
            j_security_check?j_username=
            krisv&j_password=krisv");
        InputStreamReader inputStreamReader
            = null;
        BufferedReader bufferedReader = null
            ;
        try {
           // UrlEncodedFormEntity entity =
                new UrlEncodedFormEntity(
                formparams, "UTF-8");//
                UrlEncodedFormEntity(
                formparams, "multipart/form-
                data");
    //httpGet.setEntity(entity);
            HttpResponse response =
                 httpClient.execute(httpGet);
            InputStream inputStream =
                 response.getEntity().
                 getContent();
            inputStreamReader = new
                 InputStreamReader(
                 inputStream);
            bufferedReader = new
                 BufferedReader(
                 inputStreamReader);
            StringBuilder stringBuilder =
                 new StringBuilder();
            String line = bufferedReader.
                 readLine();
            while (line != null) {
                stringBuilder.append(line);
                line = bufferedReader.
                     readLine();
            }
            responseString = stringBuilder.
                 toString();
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            if (inputStreamReader != null) {
                try {
                    inputStreamReader.close
                         ();
                } catch (Exception e) {
                    throw new
                         RuntimeException(e);
                }
            }
            if (bufferedReader != null) {
                try {
                    bufferedReader.close();
                } catch (Exception e) {
                    throw new
                         RuntimeException(e);
                }
            }
        }
        System.out.println(responseString);
    }
  public String requestPostService(String url
      , Map<String, Object> parameters,
      boolean multipart) {
        String responseString = "";

        MultipartEntity multiPartEntity =
            new MultipartEntity(
            HttpMultipartMode.
            BROWSER_COMPATIBLE);
        List<NameValuePair> formparams = new
             ArrayList<NameValuePair>();
        if (parameters == null) {
            parameters = new HashMap<String,
                 Object>();
        }
        Set<String> keys = parameters.keySet
            ();
        for (Iterator<String> keysIterator =
             keys.iterator(); keysIterator.
             hasNext();) {
            String keyString = keysIterator.
                 next();
            String value = parameters.get(
                 keyString).toString();
            formparams.add(new
                 BasicNameValuePair(keyString
                 , value));
            if (multipart) {
                try {
                    StringBody stringBody =
                         new StringBody(value
                         , "text/plain",
                         Charset.forName("UTF
                         -8"));
                    multiPartEntity.addPart(
                         keyString, (
                         ContentBody)
                         stringBody);
                } catch (Exception e) {
                    throw new
                         RuntimeException(e);
                }
            }
        }
        HttpGet httpGet = new HttpGet(url);

        InputStreamReader inputStreamReader
            = null;
        BufferedReader bufferedReader = null
            ;
        try {
            if (multipart) {
               // httpGet.setEntity(
                    multiPartEntity);
            } else {
                UrlEncodedFormEntity entity
                     = new
                     UrlEncodedFormEntity(
                     formparams, "UTF-8");//
                     UrlEncodedFormEntity(
                     formparams, "multipart/
                     form-data");
              //  httpGet.setEntity(entity);
            }
            HttpResponse response =
                 httpClient.execute(httpGet);
            InputStream inputStream =
                 response.getEntity().
                 getContent();
            inputStreamReader = new
                 InputStreamReader(
                 inputStream);
            bufferedReader = new
                 BufferedReader(
                 inputStreamReader);
            StringBuilder stringBuilder =
                 new StringBuilder();
            String line = bufferedReader.
                 readLine();
            while (line != null) {
                stringBuilder.append(line);
                line = bufferedReader.
                     readLine();
            }
            responseString = stringBuilder.
                 toString();
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            if (inputStreamReader != null) {
                try {
                    inputStreamReader.close
                         ();
                } catch (Exception e) {
                    throw new
                         RuntimeException(e);
                }
            }
            if (bufferedReader != null) {
                try {
                    bufferedReader.close();
                } catch (Exception e) {
                    throw new
                         RuntimeException(e);
                }
            }
        }
        return responseString;
    }
  //TODO: Arreglar los exceptions
  public ProcessDefinitionsRS
       getDefinitions(HttpSession session)
       throws Exception{
   ProcessDefinitionsRS proc = null;
   String url = super.urlForm("process.
       management.definitions");
url= url + "?j_username="+session.
     getAttribute("sessionUser").toString()
     +"&j_password="+session.getAttribute("
     sessionUser").toString()+"";
System.out.println(url);
   String json = "";
   json = "'" + super.
        getDataFromRestService("http://"+
        InetAddress.getLocalHost().
        getCanonicalHostName()+":8090/gwt-
        console-server/rs/process/
        j_security_check?j_username=admin&
        j_password=admin", EnumJBPMRestType
        .GET) + "'";
json = "'" + super.getDataFromRestService(
     url, EnumJBPMRestType.GET) + "'";
json = "'" + super.getDataFromRestService
     ("http://"+InetAddress.getLocalHost().
     getCanonicalHostName()+":8090/gwt-
     console-server/rs/process/
     j_security_check?j_username="+session.
     getAttribute("sessionUser").toString()
     +"&j_password="+session.getAttribute("
     sessionUser").toString()+"",
     EnumJBPMRestType.GET) + "'";
json = "'" + super.getDataFromRestService(
     url, EnumJBPMRestType.GET) + "'";
//authenticate("localhost:8080",""+session
     .getAttribute("sessionUser").toString
     ()+"",""+session.getAttribute("
     sessionUser").toString()+"");
//json = "'" + super.
     getDataFromRestService(url,
     EnumJBPMRestType.GET) + "'";
//json = "'" +  requestPostService(url,
     null, false) + "'";
  //json = json.replaceAll("\"",
       "\\\\\"");
json = json.replaceAll("\'", "");
   System.out.println(json);
      Gson gson = new Gson();
      proc = gson.fromJson(json,
           ProcessDefinitionsRS.class);
   return proc;
  }

  public ProcessDefinitionInstancesRS
       getProcessInstances(DefinitionsRS
       definitionsRS,HttpSession session)
       throws Exception{
   ProcessDefinitionInstancesRS pi = null;
   String url = super.urlForm("process.
        management.instances",definitionsRS
        .getId());
   String json="";
   try{
   json = super.getDataFromRestService(url
        , EnumJBPMRestType.GET);
   new JsonParser().parse(json);
   }
   catch(JsonParseException e){

    url= url + "?j_username="+session.
         getAttribute("sessionUser").
         toString()+"&j_password="+session.
         getAttribute("sessionUser").
         toString()+"";
    System.out.println(url);
    json = "'" + super.
         getDataFromRestService("http://"+
         InetAddress.getLocalHost().
         getCanonicalHostName()+":8090/gwt-
         console-server/rs/process/
         j_security_check?j_username=admin&
         j_password=admin",
         EnumJBPMRestType.GET) + "'";
    json = "'" + super.
         getDataFromRestService(url,
         EnumJBPMRestType.GET) + "'";
    json = "'" + super.
         getDataFromRestService("http://"+
         InetAddress.getLocalHost().
         getCanonicalHostName()+":8090/gwt-
         console-server/rs/process/
         j_security_check?j_username="+
         session.getAttribute("sessionUser
         ").toString()+"&j_password="+
         session.getAttribute("sessionUser
         ").toString()+"", EnumJBPMRestType
         .GET) + "'";
    json = "'" + super.
         getDataFromRestService(url,
         EnumJBPMRestType.GET) + "'";
    json = json.replaceAll("\'", "");
       System.out.println(json);

   }
   Gson gson = new Gson();
   pi = gson.fromJson(json,
        ProcessDefinitionInstancesRS.class)
        ;
   return pi;
  }

  /**
   * requiere definition id e instanceid
   * @param instancesRS
   * @return
   * @throws Exception
   */
  public InstancesRS getProcessInstance(
       InstancesRS instancesRS,HttpSession
       session)throws Exception{
   DefinitionsRS definitionsRS = new
        DefinitionsRS();
   definitionsRS.setId(instancesRS.
        getDefinitionId());
   ProcessDefinitionInstancesRS pi = this.
        getProcessInstances(definitionsRS,
        session);
   for (InstancesRS instance : pi.
        getInstances()) {
 if(instance.getId().equals(instancesRS.
      getId())){
  instancesRS = instance;
 }
}
   return instancesRS;
  }

  public void deleteInstance(InstancesRS
       instance)throws Exception{
   String url = super.urlForm("process.
        management.delete",instance.getId()
        );
   String json = super.
        getDataFromRestService(url,
        EnumJBPMRestType.POST);
   System.out.println(json);
  }

  public InstancesRS startInstace(
       DefinitionsRS definitionsRS,
       HttpSession session)throws Exception
       {
   InstancesRS inst;
   String url = super.urlForm("process.
        management.new_instance",
        definitionsRS.getId());
String json = "";
try{
json = super.getDataFromRestService(url,
     EnumJBPMRestType.POST);
new JsonParser().parse(json);
   }
   catch(JsonParseException e){
    url= url + "?j_username="+session.
         getAttribute("sessionUser").
         toString()+"&j_password="+session.
         getAttribute("sessionUser").
         toString()+"";
    System.out.println(url);
    json = super.getDataFromRestService("
         http://"+InetAddress.getLocalHost
         ().getCanonicalHostName()+":8090/
         gwt-console-server/rs/process/
         j_security_check?j_username=admin&
         j_password=admin",
         EnumJBPMRestType.POST);
    json = super.getDataFromRestService(
         url, EnumJBPMRestType.POST);
    json = super.getDataFromRestService("
         http://"+InetAddress.getLocalHost
         ().getCanonicalHostName()+":8090/
         gwt-console-server/rs/process/
         j_security_check?j_username="+
         session.getAttribute("sessionUser
         ").toString()+"&j_password="+
          session.getAttribute("sessionUser
          ").toString()+"", EnumJBPMRestType
          .POST);
     json = super.getDataFromRestService(
          url, EnumJBPMRestType.POST);

        System.out.println(json);
    }



    Gson gson = new Gson();
    inst = gson.fromJson(json, InstancesRS.
         class);
    return inst;
   }

   public Collection<ActiveNodeInfoRS>
        getActiveNodeInfo(InstancesRS
        instance,HttpSession session)throws
        Exception{
    Collection<ActiveNodeInfoRS> list =
         null;
    String url = super.urlForm("process.
         management.activeNodeInfo",instance
         .getId());


String json = "";
try{
json = super.getDataFromRestService(url,
     EnumJBPMRestType.GET);
new JsonParser().parse(json);
    }
    catch(JsonParseException e){
     url= url + "?j_username="+session.
          getAttribute("sessionUser").
          toString()+"&j_password="+session.
          getAttribute("sessionUser").
          toString()+"";
     System.out.println(url);
     json = "'" + super.
          getDataFromRestService("http://"+
          InetAddress.getLocalHost().
          getCanonicalHostName()+":8090/gwt-
          console-server/rs/process/
          j_security_check?j_username=admin&
          j_password=admin",
          EnumJBPMRestType.GET) + "'";
     json = "'" + super.
          getDataFromRestService(url,
          EnumJBPMRestType.GET) + "'";
     json = "'" + super.
          getDataFromRestService("http://"+
          InetAddress.getLocalHost().
          getCanonicalHostName()+":8090/gwt-
          console-server/rs/process/
          j_security_check?j_username="+
          session.getAttribute("sessionUser
          ").toString()+"&j_password="+
          session.getAttribute("sessionUser
          ").toString()+"", EnumJBPMRestType
          .GET) + "'";
     json = "'" + super.
          getDataFromRestService(url,
          EnumJBPMRestType.GET) + "'";
     json = json.replaceAll("\'", "");
        System.out.println(json);
    }



    Gson gson = new Gson();
    Type collectionType = new TypeToken<
         Collection<ActiveNodeInfoRS>>(){}.
         getType();
    list = gson.fromJson(json,
         collectionType);
    return list;
   }

   public DefinitionsRS getDefinition(
        DefinitionsRS definition,HttpSession
        session)throws Exception{
    ProcessDefinitionsRS proc = this.
         getDefinitions(session);
    Collection<DefinitionsRS> list = proc.
         getDefinitions();
    for (DefinitionsRS definitionsRS : list
         ) {
  if(definition.getId().equals(
      definitionsRS.getId())){
   definition = definitionsRS;
  }
 }

    return definition;
   }

   public String getUrlImage(DefinitionsRS
        definitionsRS)throws Exception{
    String url = super.urlForm("process.
         management.image",definitionsRS.
         getId());
    return url;
   }

   public InputStream getImage(
        DefinitionsRS definitionsRS)throws
        Exception{
    String url = this.getUrlImage(
         definitionsRS);
    InputStream img = super.
         getBytesFromRestService(url,
         EnumJBPMRestType.GET);
    return img;
   }

   public TaskUserRS getListUserTask(
        UserTaskVO userTaskVO,HttpSession
        session)throws Exception{
    TaskUserRS tu = null;
      String url = super.urlForm("task.list
           .tasks",userTaskVO.getUsername())
           ;
      super.relogin(userTaskVO.getUsername
           (), userTaskVO.getPassword());


    String json = "";
try{
json = super.getDataFromRestService(url,
     EnumJBPMRestType.GET);
new JsonParser().parse(json);
    }
    catch(JsonParseException e){
     url= url + "?j_username="+session.
          getAttribute("sessionUser").
          toString()+"&j_password="+session.
          getAttribute("sessionUser").
          toString()+"";
     System.out.println(url);
     json = "'" + super.
          getDataFromRestService("http://"+
          InetAddress.getLocalHost().
          getCanonicalHostName()+":8090/gwt-
          console-server/rs/process/
          j_security_check?j_username=admin&
          j_password=admin",
          EnumJBPMRestType.GET) + "'";
     json = "'" + super.
          getDataFromRestService(url,
          EnumJBPMRestType.GET) + "'";
     json = "'" + super.
          getDataFromRestService("http://"+
          InetAddress.getLocalHost().
          getCanonicalHostName()+":8090/gwt-
          console-server/rs/process/
          j_security_check?j_username="+
          session.getAttribute("sessionUser
          ").toString()+"&j_password="+
          session.getAttribute("sessionUser
          ").toString()+"", EnumJBPMRestType
          .GET) + "'";
     json = "'" + super.
          getDataFromRestService(url,
          EnumJBPMRestType.GET) + "'";
     json = json.replaceAll("\'", "");
        System.out.println(json);
    }


    Gson gson = new Gson();
    tu = gson.fromJson(json, TaskUserRS.
         class);
    return tu;
   }

   /**
    * Requiere username y password y el
         idTaskRS
    * @param userTaskVO
    * @return
    * @throws Exception
  */
  public TaskRS getUserTask(UserTaskVO
      userTaskVO,HttpSession session)
      throws Exception{
   TaskUserRS tu = null;
   TaskRS task = userTaskVO.getTaskRS();
   tu = this.getListUserTask(userTaskVO,
       session);
   for (TaskRS iterable_element : tu.
       getTasks()) {
 if(iterable_element.getId().equals(task.
     getId())){
  task = iterable_element;
  }
}
   return task;
  }

  public String getProcessRenderHTML(
      DefinitionsRS pd,HttpSession session
      )throws Exception{
   String url = super.urlForm("form.
       process.render",pd.getId());

   String html = super.
       getDataFromRestService(url,
       EnumJBPMRestType.GET);
   /*
   String formTag = StringUtils.
       substringBetween(html, "<form",
       ">");
   formTag = "<form"+formTag+">";
   String formTagFinal = formTag + "<input
        type='hidden' name='_e_pID' value
       ='"+pd.getId()+"' />" +"<input type
       ='hidden' name='fields' value=''>"
       +
 "<input type='hidden' name='values' value
      =''>"+
 "<input type='hidden' name='formname'
      value=''>";
   html = html.replaceAll(formTag,
       formTagFinal );
   //html = html.replaceAll(formExp,
       formExp + "<input type='hidden'
       name='_e_pID' value='"+pd.getId()
       +"' />");
   */
   html = "<input type='hidden' name='
       _e_pID' value='"+pd.getId()+"' />"
       + html;
   return html;
  }

  public String getTaskRenderHTML(
      UserTaskVO userTaskVO,HttpSession
      session)throws Exception{
   String taskId = userTaskVO.getTaskRS().
       getId();
   String url = super.urlForm("form.task.
       render",taskId) + "?j_username="+
       session.getAttribute("sessionUser")
       .toString()+"&j_password="+session.
       getAttribute("sessionUser").
       toString()+"";
   super.relogin(userTaskVO.getUsername(),
        userTaskVO.getPassword());

   String html =super.
       getDataFromRestService("http://"+
       InetAddress.getLocalHost().
       getCanonicalHostName()+":8090/gwt-
       console-server/rs/process/
       j_security_check?j_username=admin&
       j_password=admin", EnumJBPMRestType
       .GET);
   html =super.getDataFromRestService(url,
        EnumJBPMRestType.GET);
   html =super.getDataFromRestService("
       http://"+InetAddress.getLocalHost()
       .getCanonicalHostName()+":8090/gwt-
       console-server/rs/process/
       j_security_check?j_username="+
       session.getAttribute("sessionUser")
       .toString()+"&j_password="+session.
       getAttribute("sessionUser").
       toString()+"", EnumJBPMRestType.GET
       );



   html =super.getDataFromRestService(url,
       EnumJBPMRestType.GET);
  /*
  String formTag = StringUtils.
      substringBetween(html, "<form",
      ">");
  formTag = "<form"+formTag+">";
  String formTagFinal = formTag + "<input
       type='hidden' name='_e_tID' value
      ='"+taskId+"' />";
  html = html.replaceAll(formTag,
      formTagFinal );
  */
  html = "<input type='hidden' name='
      _e_tID' value='"+taskId+"' />" +
      html;

  return html;
 }

 public InputStream getProcessRender(
     DefinitionsRS pd)throws Exception{
  InputStream render = null;
  String url = super.urlForm("form.
      process.render",pd.getId());
  render = super.getBytesFromRestService(
      url, EnumJBPMRestType.GET);
  return render;
 }

 public InputStream getTaskRender(
     UserTaskVO userTaskVO)throws
     Exception{
  InputStream render = null;
  String url = super.urlForm("form.task.
      render",userTaskVO.getTaskRS().
      getId());
  super.relogin(userTaskVO.getUsername(),
       userTaskVO.getPassword());
  render = super.getBytesFromRestService(
      url, EnumJBPMRestType.GET);
  return render;
 }

 public String processComplete(
     DefinitionsRS definitionsRS)throws
     Exception{
  String resp;
  String url = super.urlForm("form.
      process.complete",definitionsRS.
      getId());
  resp = super.getDataFromRestService(url
      , EnumJBPMRestType.MULTIPART);
  System.out.println(resp);
  return resp;
 }

 public String processComplete(
     DefinitionsRS definitionsRS, Map<
     String, Object> params)throws
     Exception{
  String resp;
  String url = super.urlForm("form.
      process.complete",definitionsRS.
      getId());
  resp = super.getDataFromRestService(url
      , EnumJBPMRestType.MULTIPART,
      params);
  System.out.println(resp);
  return resp;
 }

 public String taskComplete(String taskID
     , Map<String, Object> params,
     HttpSession session)throws Exception
     {
  String resp;
  String url = super.urlForm("form.task.
      complete",taskID)  + "?j_username
      ="+session.getAttribute("
      sessionUser").toString()+"&
      j_password="+session.getAttribute("
      sessionUser").toString();
  resp =super.getDataFromRestService("
      http://"+InetAddress.getLocalHost()
      .getCanonicalHostName()+":8090/gwt-
      console-server/rs/process/
      j_security_check?j_username=admin&
      j_password=admin", EnumJBPMRestType
      .MULTIPART, params);
  resp =super.getDataFromRestService(url,
       EnumJBPMRestType.MULTIPART, params
      );
```

```
    resp =super.getDataFromRestService("
         http://"+InetAddress.getLocalHost()
         .getCanonicalHostName()+":8090/gwt-
         console-server/rs/process/
         j_security_check?j_username="+
         session.getAttribute("sessionUser")
         .toString()+"&j_password="+session.
         getAttribute("sessionUser").
         toString()+"", EnumJBPMRestType.
         MULTIPART, params);

    resp = super.getDataFromRestService(url
         , EnumJBPMRestType.MULTIPART,
         params);
    System.out.println(resp);
    return resp;
   }

   public String taskComplete(UserTaskVO
        userTaskVO, Map<String, Object>
        params)throws Exception{
    String resp;
    String url = super.urlForm("form.task.
         complete",userTaskVO.getTaskRS().
         getId());
    super.relogin(userTaskVO.getUsername(),
          userTaskVO.getPassword());
    resp = super.getDataFromRestService(url
         , EnumJBPMRestType.MULTIPART,
         params);
    //this.getSecureSid();
    System.out.println(resp);
    return resp;
   }

   public void userInvalidate()throws
        Exception{
    String url = super.urlForm("user.
         management.invalidate");
    super.getDataFromRestService(url,
         EnumJBPMRestType.POST);
   }

   public String getSid()throws Exception{
    String url = super.urlForm("user.
         management.sid");
    return super.getDataFromRestService(url
         , EnumJBPMRestType.GET);
   }

   public String getSecureSid()throws
        Exception{
    String url = super.urlForm("user.
         management.secure.sid");
    return super.getDataFromRestService(url
         , EnumJBPMRestType.GET);
   }

   public String getDataSet(TaskRS taskRS)
        throws Exception{
    String data ="";
    String url = super.urlForm("process.
         management.dataset", taskRS.getId()
         );
    data = super.getDataFromRestService(url
         , EnumJBPMRestType.GET);
    return data;
   }
   //get tasks
   public static List <TaskSummary>
        getAssignedTasks(String idRef) {
    JbpmAPIUtil.connect();
  List<TaskSummary> tasks = null;
  try {
   BlockingTaskSummaryResponseHandler
        responseHandler = new
        BlockingTaskSummaryResponseHandler();
   JbpmAPIUtil.client.
        getTasksAssignedAsPotentialOwner(
        idRef, "en-UK", responseHandler);
          tasks = responseHandler.getResults
              ();
  } catch (Throwable t) {
   t.printStackTrace();
  }
  return tasks;
 }
    //get group tasks
    public static List <TaskSummary>
         getAssignedGroupTasks(String idRef,
         ArrayList<String> groups) {
     JbpmAPIUtil.connect();
  List<TaskSummary> tasks = null;
  try {
   BlockingTaskSummaryResponseHandler
        responseHandler = new
        BlockingTaskSummaryResponseHandler();
   JbpmAPIUtil.client.
        getTasksAssignedAsPotentialOwner(
        idRef, groups,"en-UK",
        responseHandler);
          tasks = responseHandler.getResults
              ();
  } catch (Throwable t) {
   t.printStackTrace();
  }
  return tasks;
 }
    public static void main(String[] args)
         throws Exception {

     TaskRS taskRS = new TaskRS();
     taskRS.setId("2");
     String proc = Process.instance().
          getDataSet(taskRS);
     System.out.println(taskRS);
//      ProcessDefinitionsRS proc = Process.
     instance().getDefinitions();
//      DefinitionsRS definition = new
     DefinitionsRS();
//      definition.setId("Hello");
//      Process.instance().processComplete(
     definition);
//      DefinitionsRS defini = Process.
     instance().getDefinition(definition);
//      System.out.println(defini.
     getDiagramUrl());
//      ProcessDefinitionInstancesRS pi =
     Process.instance().getProcessInstances(
     definition);
////      InstancesRS instance = new
     InstancesRS();
////      instance.setId("2");
////      Process.instance().
     getActiveNodeInfo(instance);
//      TaskUserRS taskUsers = Process.
     instance().getUserTask(new UserTaskVO
     (""+session.getAttribute("sessionUser").
     toString()+"", ""+session.getAttribute("
     sessionUser").toString()+""));
//      Process.instance().getImage(
     definition);
     }

}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;

public class ProcessDefinitionInstancesRS
     implements Serializable{

 private static final long serialVersionUID
      = -9158303496971170892L;

 private Collection<InstancesRS> instances =
       new ArrayList<InstancesRS>();

 public Collection<InstancesRS> getInstances
      () {
  return instances;
 }

 public void setInstances(Collection<
      InstancesRS> instances) {
  this.instances = instances;
 }

}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class ProcessDefinitionsRS implements
      Serializable{

 private static final long serialVersionUID
      = -555493363623066337L;

 private Collection<DefinitionsRS>
      definitions = new ArrayList<
      DefinitionsRS>();

 public Collection<DefinitionsRS>
      getDefinitions() {
  return definitions;
 }

 public void setDefinitions(List<
      DefinitionsRS> definitions) {
  this.definitions = definitions;
 }

}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;

public class TaskRS implements Serializable{

 /**
  *
  */
 private static final long serialVersionUID
      = 117065476711370847L;

 private String id;
 private String processInstanceId;
 private String processId;
 private String name;
 private String assignee;
 private String isBlocking;
 private String isSignalling;
 //private outcomes;
 private String currentState;
 //private participantUsers
 //private participantGroups
 private String url;
 private String priority;
 public String getId() {
  return id;
 }
 public void setId(String id) {
  this.id = id;
 }
 public String getProcessInstanceId() {
  return processInstanceId;
 }
 public void setProcessInstanceId(String
      processInstanceId) {
  this.processInstanceId = processInstanceId
       ;
 }
 public String getProcessId() {
  return processId;
 }
 public void setProcessId(String processId)
      {
  this.processId = processId;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getAssignee() {
  return assignee;
 }
 public void setAssignee(String assignee) {
  this.assignee = assignee;
 }
 public String getIsBlocking() {
  return isBlocking;
 }
 public void setIsBlocking(String isBlocking
      ) {
  this.isBlocking = isBlocking;
 }
 public String getIsSignalling() {
  return isSignalling;
 }
 public void setIsSignalling(String
      isSignalling) {
  this.isSignalling = isSignalling;
 }
 public String getCurrentState() {
  return currentState;
 }
 public void setCurrentState(String
      currentState) {
  this.currentState = currentState;
 }
 public String getUrl() {
  return url;
 }
 public void setUrl(String url) {
  this.url = url;
 }
 public String getPriority() {
  return priority;
 }
 public void setPriority(String priority) {
  this.priority = priority;
 }


}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;

public class InstancesRS implements
     Serializable{

 /**
  *
  */
 private static final long serialVersionUID
      = -8369132192357526352L;

 private String id;
 private String definitionId;
 private String startDate;
 private String suspended;
 private RootTokenRS rootToken = new
      RootTokenRS();

 public String getId() {
  return id;
 }
 public void setId(String id) {
  this.id = id;
 }
 public String getDefinitionId() {
  return definitionId;
 }
 public void setDefinitionId(String
      definitionId) {
  this.definitionId = definitionId;
 }
 public String getStartDate() {
  return startDate;
 }
 public void setStartDate(String startDate)
      {
  this.startDate = startDate;
 }
 public String getSuspended() {
  return suspended;
 }
 public void setSuspended(String suspended)
      {
  this.suspended = suspended;
 }
 public RootTokenRS getRootToken() {
  return rootToken;
 }
 public void setRootToken(RootTokenRS
      rootToken) {
  this.rootToken = rootToken;
```

```
 }


}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;

public class DefinitionsRS implements
     Serializable{

 private static final long serialVersionUID
      = -767123909817759735 5L;

 private String id;
 private String name;
 private String version;
 private String packageName;
 private String deploymentId;
 private String suspended;
 private String diagramUrl;

 public String getId() {
  return id;
 }
 public void setId(String id) {
  this.id = id;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getVersion() {
  return version;
```

```
 }
 public void setVersion(String version) {
  this.version = version;
 }
 public String getPackageName() {
  return packageName;
 }
 public void setPackageName(String
      packageName) {
  this.packageName = packageName;
 }
 public String getDeploymentId() {
  return deploymentId;
 }
 public void setDeploymentId(String
      deploymentId) {
  this.deploymentId = deploymentId;
 }
 public String getSuspended() {
  return suspended;
 }
 public void setSuspended(String suspended)
      {
  this.suspended = suspended;
 }
 public String getDiagramUrl() {
  return diagramUrl;
 }
 public void setDiagramUrl(String diagramUrl
      ) {
  this.diagramUrl = diagramUrl;
 }
}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;

public class TaskRS implements Serializable{

 /**
  *
  */
 private static final long serialVersionUID
      = 117065476711370847L;

 private String id;
 private String processInstanceId;
 private String processId;
 private String name;
 private String assignee;
 private String isBlocking;
 private String isSignalling;
 //private outcomes;
 private String currentState;
 //private participantUsers
 //private participantGroups
 private String url;
 private String priority;
 public String getId() {
  return id;
 }
 public void setId(String id) {
  this.id = id;
 }
 public String getProcessInstanceId() {
  return processInstanceId;
 }
 public void setProcessInstanceId(String
      processInstanceId) {
  this.processInstanceId = processInstanceId
       ;
 }
 public String getProcessId() {
  return processId;
 }
 public void setProcessId(String processId)
      {
  this.processId = processId;
 }
 public String getName() {
```

```
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getAssignee() {
  return assignee;
 }
 public void setAssignee(String assignee) {
  this.assignee = assignee;
 }
 public String getIsBlocking() {
  return isBlocking;
 }
 public void setIsBlocking(String isBlocking
      ) {
  this.isBlocking = isBlocking;
 }
 public String getIsSignalling() {
  return isSignalling;
 }
 public void setIsSignalling(String
      isSignalling) {
  this.isSignalling = isSignalling;
 }
 public String getCurrentState() {
  return currentState;
 }
 public void setCurrentState(String
      currentState) {
  this.currentState = currentState;
 }
 public String getUrl() {
  return url;
 }
 public void setUrl(String url) {
  this.url = url;
 }
 public String getPriority() {
  return priority;
 }
 public void setPriority(String priority) {
  this.priority = priority;
 }


}
```

```
package org.dentist.version.three.
     processserver.model;

import java.io.Serializable;

public class UserTaskVO implements
     Serializable{

 /**
  *
  */
 private static final long serialVersionUID
      = -8392014487181046993L;

 private String username;
 private String password;
 private TaskRS taskRS = new TaskRS();



 public TaskRS getTaskRS() {
  return taskRS;
 }
 public void setTaskRS(TaskRS taskRS) {
  this.taskRS = taskRS;
 }
 public UserTaskVO(String username, String
     password) {
  super();
  this.username = username;
  this.password = password;
 }
 public String getUsername() {
  return username;
 }
 public void setUsername(String username) {
  this.username = username;
 }
 public String getPassword() {
  return password;
 }
 public void setPassword(String password) {
  this.password = password;
 }


}
```

```
package org.dentist.version.three.
     processserver.model;



public class Patient{


 public String patientid;
 private String name;
 private String upcdid;
 private String instanceid;
 private String age;
 private String address;
 private String gender;
 private String birthday;

 public String getId() {
  return patientid;
 }
 public void setId(String id) {
  this.patientid = id;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getage() {
  return age;
 }
 public void setage(String age) {
  this.age = age;
 }
 public String getaddress() {
  return address;
 }
 public void setaddress(String address) {
  this.address = address;
 }
 public String getgender() {
  return gender;
 }
 public void setgender(String gender) {
  this.gender = gender;
 }
 public String getbirthday() {
  return birthday;
 }
 public void setbirthday(String birthday) {
  this.birthday = birthday;
 }
 public String getupcdId() {
  return upcdid;
 }
 public void setupcdId(String upcdid) {
  this.upcdid = upcdid;
 }
 public String getinstanceid() {
  return instanceid;
 }
 public void setinstanceid(String instanceid
     ) {
  this.instanceid = instanceid;
 }
}
```

```
package org.dentist.version.three.
     processserver.model;

public class Forms {
 private String id;
 private String name;
 private String taskid;


 public String getId() {
  return id;
 }
 public void setId(String id) {
  this.id = id;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String gettaskid() {
  return taskid;
 }
 public void settaskid(String taskid) {
  this.taskid = taskid;
 }

}
```

```
package org.dentist.version.three.
     processserver.model;

public class Appointment {
 private String appointmentid;
 private String appointmentdate;
 private String appointmentclinician;
 private String patientid;
 private String patientname;
 private String instanceid;
 private String idtask;
 private String nametask;

 public String getappointmentid() {
  return appointmentid;
 }
 public void setappointmentid(String
     appointmentid) {
  this.appointmentid = appointmentid;
 }
 public String getappointmentdate() {
  return appointmentdate;
 }
 public void setappointmentdate(String
     appointmentdate) {
  this.appointmentdate = appointmentdate;
 }
 public String getappointmentclinician() {
  return appointmentclinician;
 }
 public void setappointmentclinician(String
     appointmentclinician) {
  this.appointmentclinician =
      appointmentclinician;
 }

 public String getpatientid() {
  return patientid;
 }
 public void setpatientid(String patientid)
     {
  this.patientid = patientid;
 }

 public String getpatientname() {
  return patientname;
 }
 public void setpatientname(String
     patientname) {
  this.patientname = patientname;
 }

 public String getinstanceid() {
  return instanceid;
 }
 public void setinstanceid(String instanceid
     ) {
  this.instanceid = instanceid;
 }
 public String getidtask() {
  return idtask;
 }
 public void setidtask(String idtask) {
  this.idtask = idtask;
 }
 public String getnametask() {
  return nametask;
 }
 public void setnametask(String nametask) {
  this.nametask = nametask;
 }
}
```

```
package org.dentist.version.three.
    processserver.model;

public class Version {

 private String version;
 private String updated_by;
 private String updated_date;
 private String updated_time;
 private String approved;
 private String approved_by;
 private String approved_date;
 private String approved_time;

 public void setApproved_time(String
     approved_time){
  this.approved_time=approved_time;
 }
 public void setApproved_date(String
     approved_date){
  this.approved_date=approved_date;
 }

 public void setApproved_by(String
     approved_by){
  this.approved_by=approved_by;
 }

 public void setApproved(String approved){
  this.approved=approved;
 }
 public void setUpdated_time(String
     updated_time){
  this.updated_time=updated_time;
 }

 public void setVersion(String version){
  this.version=version;
 }

 public void setUpdated_by(String updated_by
     ){
  this.updated_by=updated_by;
 }

 public void setUpdated_date(String
     updated_date){
  this.updated_date=updated_date;
 }

 public String getApproved_time(){
  return approved_time;
 }
 public String getApproved_date(){
  return approved_date;
 }

 public String getApproved_by(){
  return approved_by;
 }

 public String getApproved(){
  return approved;
 }
 public String getUpdated_time(){
  return updated_time;
 }

 public String getVersion(){
  return version;
 }

 public String getUpdated_by(){
  return updated_by;
 }

 public String getUpdated_date(){
  return updated_date;
 }
}
```

```
package org.dentist.version.three.
    processserver.model;

public class Update {

 private String patientid;
 private String instanceid;
 private String idtask;
 private String nametask;
 private String version;


 public String getpatientid(){
  return patientid;
 }
 public String getinstanceid(){
  return instanceid;
 }

 public String getidtask(){
  return idtask;
 }

 public String getnametask(){
  return nametask;
 }
 public String getversion(){
  return version;
 }

 public void setpatientid(String patientid){
  this.patientid=patientid;
 }
 public void setinstanceid(String instanceid
     ){
  this.instanceid=instanceid;
 }

 public void setidtask(String idtask){
  this.idtask=idtask;
 }

 public void setnametask(String nametask){
  this.nametask=nametask;
 }
 public void setversion(String version){
  this.version=version;
 }

 }
```

```
package org.dentist.version.three.web.
     controller;

import org.jbpm.api.ExecutionService;
import org.jbpm.api.ProcessEngine;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import javax.naming.InitialContext;
import javax.persistence.
     EntityManagerFactory;
import javax.persistence.Persistence;
import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.transaction.UserTransaction;

import org.jboss.resteasy.util.
     HttpServletRequestDelegate;

import org.jbpm.task.query.TaskSummary;
import org.jbpm.task.service.
     responsehandlers.
     BlockingTaskOperationResponseHandler;

import org.springframework.stereotype.
     Controller;
import org.springframework.validation.
     BindingResult;
import org.springframework.web.bind.
     annotation.ModelAttribute;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;
import org.springframework.web.bind.
     annotation.RequestParam;
import org.springframework.web.multipart.
     support.
     DefaultMultipartHttpServletRequest;
import org.springframework.web.servlet.
     ModelAndView;

import org.apache.commons.lang.StringUtils;
import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.db.
     DentalChartSP;
import org.dentist.version.three.form.
     CariesStatus;
import org.dentist.version.three.form.
     DentalChart;
import org.dentist.version.three.form.
     RecurrentStatus;
import org.dentist.version.three.form.
     RestorationStatus;
import org.dentist.version.three.form.
     ServiceNeeded;
import org.dentist.version.three.
     processserver.HumanTaskStartupServlet;
import org.dentist.version.three.
     processserver.JbpmAPIUtil;
import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.InstancesRS;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.TaskRS;
import org.dentist.version.three.
     processserver.model.UserTaskVO;
import org.dentist.version.three.
     processserver.service.Process;


import bitronix.tm.
     TransactionManagerServices;

import com.dentist.version.three.db.
     SectionSP;
/**
FOR AUDIT TRAIL

*/
import com.dentist.version.three.db.AdminSP;
import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.User;



/**
END AUDIT TRAIL

**/
@Controller
public class CompleteController {

    @RequestMapping(value="complete",method
        = RequestMethod.GET)
    public
    @ModelAttribute("message")
    String getInitialMessage() throws
        Exception{
     return "esto es por post del ortix XD";
    }

    @RequestMapping(value="complete",method
        = RequestMethod.POST)
//     public
//     @ModelAttribute("message")
    public ModelAndView  getGreeting(
        HttpServletRequest request,
        HttpServletResponse response,
        HttpSession session) throws
        Exception{
     DentFormsSP dentFormsSP= new
          DentFormsSP();
     ModelAndView mav =new ModelAndView("
          complete");
     String resp;
     DefinitionsRS defnitionRS = new
          DefinitionsRS();
     String _e_pID = null, _e_tID = null;
     String patientid=request.getParameter("
          patientid");
     TaskRS taskRS = new TaskRS();
     UserTaskVO userTaskVO = (UserTaskVO)
          session.getAttribute("userTask");
     Map<String, Object> params = new
          HashMap<String, Object>();
     String intid=request.getParameter("
          instanceid");
     AdminSP adminSP= new AdminSP();
     HashMap<String,String> map=
          TaskController.formlist; //FOR
          APPROVE STATUS
     if(map.isEmpty()){
      map=SearchController.formlist;
      if(map.isEmpty()){
       map=ViewController.formlist;
      }
     }

     if(request instanceof
          DefaultMultipartHttpServletRequest)
          {
      DefaultMultipartHttpServletRequest def
           = (
           DefaultMultipartHttpServletRequest
           ) request;

         for(Enumeration e = def.
             getParameterNames(); e.
             hasMoreElements(); ){
                String attName = (String)e.
                    nextElement();
                System.out.println(attName);
                if("_e_pID".equals(attName))
                    {
                 _e_pID = def.getParameter(
                      attName);


                }
                else
                    if("_e_tID".equals(
                        attName))
                     _e_tID = def.
                          getParameter(
                    attName);
                //else
                //params.put(attName,
                     def.getParameter(
                     attName));
    }

}
if(_e_pID!=null){
 dentFormsSP.setDatabase_username(
      session.getAttribute("
      sessionUserRole").toString());

 ArrayList<String> clinicians=
      dentFormsSP.listAllClinicians();
 if(session.getAttribute("
      sessionUserRole").toString().
      toLowerCase().contains("fac")){
         params.put("user", "Faculty");
         params.put("is_faculty", true);
         params.put("is_student", false)
              ;
         params.put("clinicians",
              clinicians);
         System.out.println("i am a
              faculty");

         }
         else {
          params.put("user", "Student");
          params.put("is_faculty", false
               );
          params.put("is_student", true)
               ;
          System.out.println("i am a
               student");

         }
 params.put("rejectMsg", "");
 defnitionRS.setId(_e_pID);
 resp = Process.instance().
      processComplete(defnitionRS,
      params);


}
String cont=request.getParameter("
     continue");
String approvalRes=request.getParameter
     ("approvalRes");
System.out.println("continue: "+ cont
     +"YES");
if(cont!=null && !cont.equalsIgnoreCase
     ("Save")){
 //if(getstatus(_e_tID).equals("") ||
      getstatus(_e_tID).equals("
      Completed")){
 // ModelAndView mav2 =new ModelAndView
      ("greet");
 // mav2.addObject("msgtasktaken","The
      record has been checked by another
       section faculty");
 // return mav2;
 //}
 if(request instanceof
      DefaultMultipartHttpServletRequest
      ){
     DefaultMultipartHttpServletRequest
           def = (
          DefaultMultipartHttpServletRequest
          ) request;

         for(Enumeration e = def.
              getParameterNames(); e.
              hasMoreElements(); ){
                String attName = (String
                     )e.nextElement();

                params.put(attName, def.
                     getParameter(attName
                     ));
                System.out.println(
                     attName +" "+def.
                     getParameter(attName
                     ));

         }

    }
 if(cont.equalsIgnoreCase("Submit")){
  ArrayList<String> clinicians=
       dentFormsSP.listAllClinicians();
  if(session.getAttribute("
       sessionUserRole").toString().
       toLowerCase().contains("fac") ||
       session.getAttribute("
       sessionUserRole").toString().
       toLowerCase().contains("workflow
       ")){
             params.put("userRole", "
                  Faculty");
             params.put("is_faculty",
                  true);
             params.put("is_student",
                  false);
             if(approvalRes.contains("
                  Approve")){
             params.put("approval", "
                  Approve");
             }
             else if(approvalRes.
                  contains("Reject")){
             params.put("approval", "
                  Reject");
             }
             params.put("cliniciansOut",
                   clinicians);
             System.out.println("i am a
                  faculty");

             }
             else {
              params.put("userRole", "
                   Student");
              params.put("is_faculty",
                   false);
              params.put("is_student",
                   true);
              params.put("cliniciansOut
                   ", clinicians);
              System.out.println("i am a
                   student");

             }
  String clinician = request.
       getParameter("clinician");
  if(clinician!=null){
  params.put("clinician", clinician);
  }

  String section = request.getParameter
       ("section");
  if(section!=null){
/**
 SECTION CHOICE START
 -section na string nakasave what
     section na pinili
 -possible values ay "Oral Medicine", "
     Prosthodontics", "Operative
     Dentistry"


 **/
  params.put("sectionChoice", section);
  if(!session.getAttribute("
       sessionUserRole").toString().
       toLowerCase().contains("fac") &&
       !session.getAttribute("
       sessionUserRole").toString().
       toLowerCase().contains("workflow
       ")){
   mav.addObject("message","approval");
  }
  dentFormsSP.setDatabase_username(
       session.getAttribute("
       sessionUserRole").toString());
  ArrayList<Integer> listpatientids=
       new ArrayList<Integer>();
  boolean patient_exist=false;
 listpatientids= dentFormsSP.
      getpatientIDList();
  if(!listpatientids.isEmpty() ||
       listpatientids!=null){
  for(Integer patientidss :
       listpatientids){
   if(patientidss==Integer.parseInt(
        patientid))
    patient_exist=true;
  }
 } try{
   if(patient_exist){
    dentFormsSP.setDatabase_username("
         abjarabelo");
    dentFormsSP.update_specificsection(
         Integer.parseInt(patientid),
         section);

  }else{

   dentFormsSP.insert_specificsection(
        Integer.parseInt(patientid),
        section);

  }
}
catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }
  /**SECTION CHOICE END**/
 }
long _e_tIDLong= Long.parseLong(_e_tID
     );
deleteAppointment(patientid);

}
else if(cont.equalsIgnoreCase("Approve
     ")){
 ArrayList<String> clinicians=
      dentFormsSP.listAllClinicians();
 if(session.getAttribute("
      sessionUserRole").toString().
      toLowerCase().contains("fac")){

           params.put("approval", "
               Approve");
           String rejectMessage =
               request.getParameter("
               rejectMessage");
           if(rejectMessage!=null){
         params.put("rejMesg",
             rejectMessage);
         }
           System.out.println("i am a
               faculty");

           /**START OF FOR APPROVE
               STATUS*/
           for (String key : map.
               keySet()) {
            try{
            int version=dentFormsSP.
                 getCurrentVersion(
                 Integer.parseInt(
                 patientid), key.
                 toLowerCase());
            if(version!=0){
               dentFormsSP.
                    updateApprovedStat(
                    patientid, key,
                    session.
                    getAttribute("
                    sessionName").
                    toString());

               dentFormsSP.
                    updateApproved(
                    patientid, key, "
                    Approved",version);
            }
           }
            catch (Exception e) {
                // TODO Auto-generated
                     catch block
               e.printStackTrace();
              }
           }
           /**END OF FOR APPROVE
               STATUS*/
           }
           else {

            System.out.println("i am a
                  student");

           }

long _e_tIDLong= Long.parseLong(_e_tID
     );

}
else if(cont.equalsIgnoreCase("Reject
     ")){
 ArrayList<String> clinicians=
      dentFormsSP.listAllClinicians();
 if(session.getAttribute("
      sessionUserRole").toString().
      toLowerCase().contains("fac")){

           params.put("approval", "
               Reject");
           String rejectMessage =
               request.getParameter("
               rejectMessage");

           if(rejectMessage!=null){
         params.put("rejMesg",
             rejectMessage);
         }
           System.out.println("i am a
               faculty");
           /**START OF FOR APPROVE
               STATUS*/
           for (String key : map.
               keySet()) {
            try{
            int version=dentFormsSP.
                 getCurrentVersion(
                 Integer.parseInt(
                 patientid), key.
                 toLowerCase());
            if(version!=0){
               //dentFormsSP.
                    updateApprovedStat(
                    patientid, key,
                    session.
                    getAttribute("
                    sessionName").
                    toString());
               dentFormsSP.
                    updateApproved(
                    patientid, key, "
                    Rejected", version)
                    ;
            }
            }
            catch (Exception e) {
                // TODO Auto-generated
                     catch block
               e.printStackTrace();
              }
            }
           /**END OF FOR APPROVE
               STATUS*/
           }

           else {

            System.out.println("i am a
                  student");

           }

long _e_tIDLong= Long.parseLong(_e_tID
     );

}
else if(cont.equalsIgnoreCase("Save
     Remarks")){
 ArrayList<String> clinicians=
      dentFormsSP.listAllClinicians();
 if(session.getAttribute("
      sessionUserRole").toString().
      toLowerCase().contains("fac")){

           params.put("approval", "
               SaveRemarks");
           System.out.println("i am a
               faculty");
           }

           else {

            System.out.println("i am a
                  student");

           }

long _e_tIDLong= Long.parseLong(_e_tID
     );

}
else if(cont.equalsIgnoreCase("Claim")
     ){
 //ArrayList<String> clinicians=
      dentFormsSP.listAllClinicians();
 String clinician=session.getAttribute
```

```
            ("sessionUser").toString().trim()
            ;
        String clinicianname=session.
            getAttribute("sessionName").
            toString().trim();

        System.out.println("-"+clinician+"-")
            ;
        params.put("clinician", clinician);
        params.put("clinicianname",
            clinicianname);

      long _e_tIDLong= Long.parseLong(_e_tID
          );

      }
      else if(cont.equalsIgnoreCase("
          Continue")){
       params.put("contChoice", "Continue");
                  System.out.println("i am a
                      faculty");





      }
      else if(cont.equalsIgnoreCase("End")){
       params.put("contChoice", "End");
                  System.out.println("i am a
                      faculty");





      }
      else if(cont.equalsIgnoreCase("Skip"))
          {

                  System.out.println("Skip
                      set appointment");





      }
      else if(cont.equalsIgnoreCase("Save
          and Submit")){
       try{
              String formname="
                  SetAppointment";
              String fields=request.
                  getParameter("fields");
                  String values=request.
                      getParameter("values");

       dentFormsSP.setDatabase_username(
           session.getAttribute("
           sessionUserRole").toString());
       /**
       END OF MULTI-ROLE
       **/
 //System.out.println("Current Role: "+
      currentSessionRole);
 dentFormsSP.executeForms(formname,fields
      ,values,Long.parseLong(patientid),
      session);


/**
START OF AUDIT TRAIL
**/
 adminSP= new AdminSP();
     adminSP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());
 UserSP userSP= new UserSP();
     userSP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());

//date time
 DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
 Date date = new Date();
 System.out.println(dateFormat.format(
      date));
 String dateString= dateFormat.format(
      date).toString();
//end for date time
//audittrail
 int sessionUserID= Integer.parseInt(
      session.getAttribute("sessionUserId
      ").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
    Patient patient= dentFormsSP.
         getPatient(Integer.parseInt(
         patientid));
 /*
  getname for patient?
 */

 String action_performed=patient.getName
      ();//patientname
 String action_encounter="";
 int version=dentFormsSP.
      getCurrentVersion(Integer.parseInt(
      patientid),formname);

 if(version==1)
  action_encounter="INSERT";
 else
  action_encounter="UPDATE";
 String sessionName= sessionUser.
      getFname_user()+" "+ sessionUser.
      getMinit_user() + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(sessionName,
      action_encounter, action_performed,
      formname,dateString);
//end of audittrail
/**
 END OF AUDITTRAIL

**/
    }

    catch (Exception e) {
     // TODO Auto-generated catch block
     System.out.println("ERROR");
     e.printStackTrace();
    }

   }
   ArrayList<String> roles = new
        ArrayList<String>();
   for (Object object : Arrays.asList((
        String[])session.getAttribute("
        currentDatabaseList"))) {
       roles.add(object != null ? object.
            toString() : null);
       System.out.println("WEH: "+ object
            .toString());
      }
   System.out.println("PASS 1: ");

   if(!getstatus(_e_tID).equalsIgnoreCase
        ("Reserved")){
       JbpmAPIUtil.claimTask(Long.
            parseLong(_e_tID), session.
            getAttribute("sessionUser").
            toString(), roles);
   }
       System.out.println("PASS 2: ");
   Process.instance().taskComplete(_e_tID
        , params, session);
   System.out.println("PASS 3: ");
   Patient patient= dentFormsSP.
        getPatient(Integer.parseInt(
        patientid));

   System.out.println("PASS 4: ");
  }
  else if(_e_pID!=null || (cont!=null &&
       cont.equalsIgnoreCase("Save"))){
       //SAVE TO DB
       String formname=request.
            getParameter("formname");
       String fields=request.getParameter
            ("fields");
       String values=request.getParameter
            ("values");

       String app=request.getParameter("
            approve");
       System.out.println("continue: "+
            app +"YES");
       if(app!=null){
        dentFormsSP.updatePatientStatus(
            patientid , " ");
         dentFormsSP.updateApprovedStat(
             patientid , formname,session.
             getAttribute("sessionName").
             toString());
        }
        if(app==null){
        System.out.println("viewflag"+
            request.getParameter("viewflag
            "));
        if(!request.getParameter("viewflag
            ").equals("true")){
         if(_e_pID!=null){
               formname=substringAfter(
                   formname,".");
               }
        try {
         /**
START MULTI-ROLE


 boolean privilegeCheck=false;
String errorMessage="";
String currentSessionRole="";
ArrayList<String> potentialOwnerList=
     dentFormsSP.get_taskowners(Integer.
     parseInt(_e_tID));
String[] currentSessionRoleList= (String
     []) session.getAttribute("
     currentDatabaseList");
 for(int i=0; i<currentSessionRoleList.
       length;i++){
   if(currentSessionRoleList[i].
        toLowerCase().indexOf("oraldiag")
        !=-1){ //check if oraldiag
        substring ng string :p
    privilegeCheck= adminSP.
          checkPrivilege(
          currentSessionRoleList[i], "
          patient", "insert");
    if(privilegeCheck){
     currentSessionRole=
          currentSessionRoleList[i];
      break;
     }// end if privilegeCheck condition
    }// end if substring check condition

 } //end of i loop
if(currentSessionRole!=null || !
     currentSessionRole.isEmpty())
 dentFormsSP.setDatabase_username(
       currentSessionRole);
else

System.out.println("Current Role Entered
     : "+currentSessionRole);

END OF MULTI-ROLE
**/
dentFormsSP.setDatabase_username(session
     .getAttribute("sessionUserRole").
     toString());

if(formname.equals("CreateNewPatient")||
     formname.equals("Patient")){
         dentFormsSP.executeFormsCreate("
              patient",fields,values,
              dentFormsSP.getInstanceID());
         patientid=dentFormsSP.
              getPatientID().getId();
         /**
         START OF AUDIT TRAIL
         **/
         adminSP= new AdminSP();
         adminSP.setDatabase_username(
              session.getAttribute("
              sessionUserRole").toString())
              ;
         UserSP userSP= new UserSP();
         userSP.setDatabase_username(
              session.getAttribute("
              sessionUserRole").toString())
              ;

         //date time
         DateFormat dateFormat = new
              SimpleDateFormat("dd/MM/yyyy
              ");
         Date date = new Date();
         System.out.println(dateFormat.
              format(date));
         String dateString= dateFormat.
              format(date).toString();
 //end for date time

 //audittrail
 int sessionUserID= Integer.
      parseInt(session.getAttribute
      ("sessionUserId").toString())
      ;
 User sessionUser= userSP.getUser(
      sessionUserID);
 Patient patient= dentFormsSP.
      getPatient(Integer.parseInt(
      patientid));
 /*
 getname for patient?
 */

 String action_performed=patient.
      getName();//patientname
 String action_encounter="INSERT";

 String sessionName= sessionUser.
      getFname_user()+" "+
      sessionUser.getMinit_user() +
       " "+ sessionUser.
      getLname_user()+" ("+
      sessionUser.getUsername()+")
      ";
 adminSP.insertAuditTrail(
      sessionName,action_encounter,
       action_performed, "patient",
      dateString);
 System.out.println("inserted
      audit trail");
 //end of audittrail
 /**
 END OF AUDITTRAIL
 **/
}
else if(formname.equals("
     ReturningPatient")){
 dentFormsSP.setDatabase_username(
      session.getAttribute("
      sessionUserRole").toString())
      ;
 int patientidint=Integer.parseInt
      (patientid);
 int instanceid=dentFormsSP.
      getInstanceID();
 System.out.println("Patient ID:"+
      patientidint+"Instance ID:"+
      instanceid+"*");
 dentFormsSP.updateInstanceID(
      patientid, instanceid);
   }
else if(formname.equals("
     DentalChart")){
 String[] caries=request.
      getParameterValues("caries");
 String[] recurrentcaries=request.
      getParameterValues("
      recurrentcaries");
 String[] restoration=request.
      getParameterValues("
      restoration");
 String completedenture=request.
      getParameter("completedenture
      ");
 String singledenture=request.
      getParameter("singledenture")
      ;
 String[] removablepartial=request
      .getParameterValues("
      removablepartial");
  String[] extrusion=request.
       getParameterValues("
       extrusion");
  String[] intrusion=request.
       getParameterValues("
       intrusion");
  String[] mesialdrift=request.
       getParameterValues("
       mesialdrift");
  String[] distaldrift=request.
       getParameterValues("
       distaldrift");
  String[] rotation=request.
       getParameterValues("rotation
       ");
  String[] postcorecrown=request.
       getParameterValues("postcore
       ");
```

```
String[] rootcanal=request.
     getParameterValues("
     rootcanal");
String[] pitandfissure=request.
     getParameterValues("
     pitandfissure");
String[] extracted=request.
     getParameterValues("
     extracted");
String[] missing=request.
     getParameterValues("missing
     ");
String[] unerupted=request.
     getParameterValues("
     unerupted");
String[] impacted=request.
     getParameterValues("impacted
     ");
String[] porcelainfused=request.
     getParameterValues("
     porcelainfused");
String[] acryliccrown=request.
     getParameterValues("acrylic
     ");
String[] metalcrown=request.
     getParameterValues("metal");
String[] porcelaincrown=request.
     getParameterValues("
     porcelain");
String[] fixedbridge=request.
     getParameterValues("
     fixedbridge");

String[] distalcaries=request.
     getParameterValues("
     distalcaries");
String[] buccalcaries=request.
     getParameterValues("
     buccalcaries");
String[] mesialcaries=request.
     getParameterValues("
     mesialcaries");
String[] lingualcaries=request.
     getParameterValues("
     lingualcaries");
String[] occlusalcaries=request.
     getParameterValues("
     occlusalcaries");
String[] selectCariesMesial=
     request.getParameterValues("
     selectCariesMesial");
String[] selectCariesBuccal=
     request.getParameterValues("
     selectCariesBuccal");
String[] selectCariesLingual=
     request.getParameterValues("
     selectCariesLingual");
String[] selectCariesOcclusal=
     request.getParameterValues("
     selectCariesOcclusal");
String[] selectCariesDistal=
     request.getParameterValues("
     selectCariesDistal");


String[] distalrecurrent=request
     .getParameterValues("
     distalrecurrent");
String[] buccalrecurrent=request
     .getParameterValues("
     buccalrecurrent");
String[] mesialrecurrent=request
     .getParameterValues("
     mesialrecurrent");
String[] lingualrecurrent=
     request.getParameterValues("
     lingualrecurrent");
String[] occlusalrecurrent=
     request.getParameterValues("
     occlusalrecurrent");
String[] selectRecurrentMesial=
     request.getParameterValues("
     selectRecurrentMesial");
String[] selectRecurrentBuccal=
     request.getParameterValues("
     selectRecurrentBuccal");
String[] selectRecurrentLingual=
     request.getParameterValues("
     selectRecurrentLingual");
String[] selectRecurrentOcclusal
     =request.getParameterValues
     ("selectRecurrentOcclusal");
String[] selectRecurrentDistal=
     request.getParameterValues("
     selectRecurrentDistal");

 String[] distalrestoration=
      request.getParameterValues("
      distalrestoration");
 String[] buccalrestoration=
      request.getParameterValues("
      buccalrestoration");
 String[] mesialrestoration=
      request.getParameterValues("
      mesialrestoration");
 String[] lingualrestoration=
      request.getParameterValues("
      lingualrestoration");
 String[] occlusalrestoration=
      request.getParameterValues("
      occlusalrestoration");
 String[] selectRestorationMesial
      =request.getParameterValues
      ("selectRestorationMesial");
 String[] selectRestorationBuccal
      =request.getParameterValues
      ("selectRestorationBuccal");
 String[]
      selectRestorationLingual=
      request.getParameterValues("
      selectRestorationLingual");
 String[]
      selectRestorationOcclusal=
      request.getParameterValues("
      selectRestorationOcclusal");
 String[] selectRestorationDistal
      =request.getParameterValues
      ("selectRestorationDistal");

 String[] class_1= request.
      getParameterValues("class1")
      ;
 String[] class_2= request.
      getParameterValues("class2")
      ;
 String[] class_3= request.
      getParameterValues("class3")
      ;
 String[] class_4= request.
      getParameterValues("class4")
      ;
 String[] class_5= request.
      getParameterValues("class5")
      ;
 String[] onlay= request.
      getParameterValues("onlay");
 String[] extraction= request.
      getParameterValues("
      extraction");
 String[] odontectomy= request.
      getParameterValues("
      odontectomy");
 String[] special_case= request.
      getParameterValues("
      specialcase");
 String[] pulp_sedation= request.
      getParameterValues("
      pulpsedation");
 String[] crown_recementation=
      request.getParameterValues("
      crownrecementation");
 String[] filling_service=
      request.getParameterValues("
      fillingservice");
 String[] laminated= request.
      getParameterValues("
      laminated");
 String[] single_crown= request.
      getParameterValues("
      singlecrown");
 String[] bridge_service= request
      .getParameterValues("
      bridgeservice");
 String[] anterior= request.
      getParameterValues("anterior
      ");
 String[] posterior= request.
      getParameterValues("
      posterior");
 String[] ortho_endo= request.
      getParameterValues("
      otherendodontics");
 String periodontics= request.
      getParameter("periodontics")
      ;
 String surgery= request.
     getParameter("surgery");
 String emergency_treatment=
     request.getParameter("
     emergencytreatment");
 String prosthodontics= request.
     getParameter("prosthodontics
     ");
    String notes=request.
        getParameter("notes");

 //newly added//
  String patient_id= patientid;
  int version=dentFormsSP.
      getCurrentVersion(Integer.
      parseInt(patientid),"
      dentalchart");
  version = version +1;
    String is_current="yes";
//end//
    //get current date and time
    DateFormat dateFormat = new
        SimpleDateFormat("dd/MM/
        yyyy");
    //get current date time with
        Date()
    Date date = new Date();
    System.out.println(dateFormat
        .format(date));
    String currdate=dateFormat.
        format(date);

    Calendar cal = Calendar.
        getInstance();
    cal.getTime();
    SimpleDateFormat sdf = new
        SimpleDateFormat("HH:mm:
        ss");
    System.out.println( sdf.
        format(cal.getTime()) );
    String currtime=sdf.format(
        cal.getTime()) ;

DentalChart dentalChart= new
    DentalChart();

dentalChart.setPatient_id(Integer
    .parseInt(patient_id));
dentalChart.setClinician_id(1);
dentalChart.setCaries(caries);
dentalChart.setRecurrentcaries(
    recurrentcaries);
dentalChart.setRestoration(
    restoration);
dentalChart.setComplete_denture(
    completedenture);
dentalChart.setSingle_denture(
    singledenture);
dentalChart.
    setRemovable_partial_denture(
    removablepartial);
dentalChart.setExtrusion(
    extrusion);
dentalChart.setIntrusion(
    intrusion);
dentalChart.setMesial_rotation(
    mesialdrift);
dentalChart.setDistal_rotation(
    distaldrift);
dentalChart.setRotation(rotation)
    ;
dentalChart.setPostcore_crown(
    postcorecrown);
dentalChart.
    setRootcanal_treatment(
    rootcanal);
dentalChart.
    setPitfissure_sealants(
    pitandfissure);
dentalChart.setExtracted(
    extracted);
dentalChart.setMissing(missing);
dentalChart.setUnerupted(
    unerupted);
dentalChart.setImpacted(impacted)
    ;
dentalChart.setPorcelain_crown(
    porcelaincrown);
dentalChart.setMetal_crown(
    metalcrown);
dentalChart.setAcrylic_crown(
```

```
            acryliccrown);
        dentalChart.setPorcelain_infused(
            porcelaininfused);
        dentalChart.setFixed_bridge(
            fixedbridge);



        dentalChart.setVersion(version);
        dentalChart.setUpdated_by(session
            .getAttribute("sessionName").
            toString());
        dentalChart.setIs_current(
            is_current);





        //end for date time

           dentalChart.setUpdated_time(
               currtime);
           dentalChart.setUpdated_date(
               currdate);
//             dentalChart.setIs_current(
    is_current);
//             dentalChart.setApproved(
    approved);
//             dentalChart.setApproved_by(
    approved_by);
//              dentalChart.
    setApproved_date(approved_date);
//             dentalChart.
    setApprovated_time(approvated_time);

         //FOR CARRIES
         CariesStatus cariesStatus= new
             CariesStatus();
         cariesStatus.setPatient_id(
             Integer.parseInt(patient_id)
             );
         cariesStatus.setBuccal_caries(
             buccalcaries);
         cariesStatus.setDistal_caries(
             distalcaries);
         cariesStatus.setMesial_caries(
             mesialcaries);
         cariesStatus.setOcclusal_caries(
             occlusalcaries);
         cariesStatus.setLingual_caries(
             lingualcaries);
         //selectCariesMesial
         cariesStatus.
             setBuccal_restorable_caries(
             selectCariesBuccal);
         cariesStatus.
             setDistal_restorable_caries(
             selectCariesDistal);
         cariesStatus.
             setMesial_restorable_caries(
             selectCariesMesial);
         cariesStatus.
             setOcclusal_restorable_caries
             (selectCariesOcclusal);
         cariesStatus.
             setLingual_restorable_caries
             (selectCariesLingual);

         cariesStatus.setVersion(version)
             ;
         cariesStatus.setUpdated_by(
             session.getAttribute("
             sessionName").toString());

         cariesStatus.setUpdated_time(
             currtime);
         cariesStatus.setUpdated_date(
             currdate);

         //END FOR CARRIES

           //FOR RECURRENT
         RecurrentStatus recurrentStatus=
              new RecurrentStatus();
         recurrentStatus.setPatient_id(
             Integer.parseInt(patient_id)
             );
         recurrentStatus.
             setBuccal_recurrent(
             buccalrecurrent);
         recurrentStatus.
             setDistal_recurrent(
     distalrecurrent);
recurrentStatus.
     setMesial_recurrent(
     mesialrecurrent);
recurrentStatus.
     setOcclusal_recurrent(
     occlusalrecurrent);
recurrentStatus.
     setLingual_recurrent(
     lingualrecurrent);

recurrentStatus.
     setBuccal_restorable_recurrent
     (selectRecurrentBuccal);
recurrentStatus.
     setDistal_restorable_recurrent
     (selectRecurrentDistal);
recurrentStatus.
     setMesial_restorable_recurrent
     (selectRecurrentMesial);
recurrentStatus.
     setOcclusal_restorable_recurrent
     (selectRecurrentOcclusal);
recurrentStatus.
     setLingual_restorable_recurrent
     (selectRecurrentLingual);

recurrentStatus.setVersion(
     version);
recurrentStatus.setUpdated_by(
     session.getAttribute("
     sessionName").toString());

recurrentStatus.setUpdated_time(
     currtime);
recurrentStatus.setUpdated_date(
     currdate);
//END FOR RECURRENT

   //FOR restoration
RestorationStatus
     restorationStatus= new
     RestorationStatus();
restorationStatus.setPatient_id(
     Integer.parseInt(patient_id)
     );
restorationStatus.
     setBuccal_restoration(
     buccalrestoration);
restorationStatus.
     setDistal_restoration(
     distalrestoration);
restorationStatus.
     setMesial_restoration(
     mesialrestoration);
restorationStatus.
     setOcclusal_restoration(
     occlusalrestoration);
restorationStatus.
     setLingual_restoration(
     lingualrestoration);

restorationStatus.
     setBuccal_restorable_restoration
     (selectRestorationBuccal);
restorationStatus.
     setDistal_restorable_restoration
     (selectRestorationDistal);
restorationStatus.
     setMesial_restorable_restoration
     (selectRestorationMesial);
restorationStatus.
     setOcclusal_restorable_restoration
     (selectRestorationOcclusal);
restorationStatus.
     setLingual_restorable_restoration
     (selectRestorationLingual);

restorationStatus.setVersion(
     version);
restorationStatus.setUpdated_by(
     session.getAttribute("
     sessionName").toString());

restorationStatus.
     setUpdated_time(currtime);
restorationStatus.
     setUpdated_date(currdate);
//END FOR restoration

//START OF SERVICE NEEDED
ServiceNeeded serviceNeeded= new
      ServiceNeeded();
 serviceNeeded.setPatient_id(
      Integer.parseInt(patient_id)
      );
 serviceNeeded.setClass_1(class_1
      );
 serviceNeeded.setClass_2(class_2
      );
 serviceNeeded.setClass_3(class_3
      );
 serviceNeeded.setClass_4(class_4
      );
 serviceNeeded.setClass_5(class_5
      );
 serviceNeeded.setOnlay(onlay);
 serviceNeeded.setExtraction(
      extraction);
 serviceNeeded.setOdontectomy(
      odontectomy);
 serviceNeeded.setSpecial_case(
      special_case);
 serviceNeeded.setPulp_sedation(
      pulp_sedation);
 serviceNeeded.
      setCrown_recementation(
      crown_recementation);
 serviceNeeded.setFilling_service
      (filling_service);
 serviceNeeded.setLaminated(
      laminated);
 serviceNeeded.setSingle_crown(
      single_crown);
 serviceNeeded.setBridge_service(
      bridge_service);
 serviceNeeded.setAnterior(
      anterior);
 serviceNeeded.setPosterior(
      posterior);
 serviceNeeded.setOrtho_endo(
      ortho_endo);
 serviceNeeded.setPeriodontics(
      periodontics);
 serviceNeeded.setSurgery(surgery
      );
 serviceNeeded.
      setEmergency_treatment(
      emergency_treatment);
 serviceNeeded.setProsthodontics(
      prosthodontics);
 serviceNeeded.setNotes(notes);

 serviceNeeded.setIs_current(
      is_current);
 serviceNeeded.setVersion(version
      );
 serviceNeeded.setUpdated_by(
      session.getAttribute("
      sessionUser").toString());
     serviceNeeded.
          setUpdated_time(currtime
          );
     serviceNeeded.
          setUpdated_date(currdate
          );

//END OF SERVICE NEEDED


DentalChartSP dentalChartSP= new
    DentalChartSP();

try {

 /**
 START MULTI-ROLE

 **/
 String idTask=map.get(formname.
      trim());
       idTask=StringUtils.
            substringBetween(
            idTask,"idTask=","&
            nameTask");

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentSessionRole="";
 ArrayList<String>
      potentialOwnerList=
      dentFormsSP.get_taskowners(
      Integer.parseInt(idTask));
 String[] currentSessionRoleList=
                (String[]) session.
                getAttribute("
                currentDatabaseList");
            for(int i=0; i<
                currentSessionRoleList.
                length;i++){
            for(int j=0;j<potentialOwnerList
                .size();j++){
            System.out.println("PUMASOK BA?
                "+potentialOwnerList.get(j)
                +" " +
                currentSessionRoleList[i]);
            if(currentSessionRoleList[i].
                equals(potentialOwnerList.
                get(j).trim())){
            privilegeCheck= adminSP.
                checkPrivilege(
                currentSessionRoleList[i],
                formname.toLowerCase(), "
                insert");
            if(privilegeCheck){
            currentSessionRole=
                currentSessionRoleList[i];
            break;
            }else{
            currentSessionRole="
                DentIStOralDiagFac";
            }// end if privilegeCheck
                condition
            }// end if equal condition

            }//end of j loop
            } //end of i loop
            if(currentSessionRole!=null || !
                currentSessionRole.isEmpty()
                 || !currentSessionRole.
                equals(""))
            dentFormsSP.setDatabase_username
                (currentSessionRole);
            else
            dentFormsSP.setDatabase_username
                (session.getAttribute("
                sessionUserRole").toString()
                );
            /**
            END OF MULTI-ROLE
            **/
            for(int i=1; i<version;i++){
            dentalChartSP.updateDentalChart(
                Integer.parseInt(patientid),
                 i, "no");
            }
            dentalChartSP.insertDentalChart(
                dentalChart);
            dentalChartSP.insertCariesStatus
                (cariesStatus);
            dentalChartSP.
                insertRecurrentStatus(
                recurrentStatus);
            dentalChartSP.
                insertRestorationStatus(
                restorationStatus);
            dentalChartSP.
                insertServiceNeeded(
                serviceNeeded);

  /**
START OF AUDIT TRAIL
**/
 adminSP= new AdminSP();
        adminSP.setDatabase_username(
            session.getAttribute("
            sessionUserRole").toString())
            ;
 UserSP userSP= new UserSP();
        userSP.setDatabase_username(
            session.getAttribute("
            sessionUserRole").toString())
            ;

//date time
 dateFormat = new SimpleDateFormat("dd/
      MM/yyyy");
 date = new Date();
 System.out.println(dateFormat.format(
      date));
 String dateString= dateFormat.format(
      date).toString();
//end for date time

//audittrail
 int sessionUserID= Integer.parseInt(
      session.getAttribute("sessionUserId
      ").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
    Patient patient= dentFormsSP.
         getPatient(Integer.parseInt(
         patientid));
 /*
  getname for patient?
 */

 String action_performed=patient.getName
      ();//patientname
 String action_encounter="";

 if(version==1)
  action_encounter="INSERT";
 else
  action_encounter="UPDATE";
 String sessionName= sessionUser.
      getFname_user()+" "+ sessionUser.
      getMinit_user() + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(sessionName,
      action_encounter, action_performed,
       "dentalchart",dateString);
//end of audittrail
/**
 END OF AUDITTRAIL

**/

 mav.addObject("notif","success");
 if(request.getParameter("app")!=null){
        mav.addObject("app","app");
        }
       } catch (Exception e) {
        // TODO Auto-generated catch
             block
        System.out.println("ERROR");
        e.printStackTrace();
       }

       System.out.println("PASOK DENTAL
            CHART!!! :: "+ patient_id);
                 }
      else{
       /**
       START OF PRIVILEGE CHECK
       */


       /**
       START MULTI-ROLE
       **/
       String idTask=map.get(formname.
            trim());
       idTask=StringUtils.
            substringBetween(idTask,"
            idTask=","&nameTask");

       boolean privilegeCheck=false;
       String errorMessage="";
       String currentSessionRole="";
       ArrayList<String>
            potentialOwnerList=
            dentFormsSP.
            checkaccess_section(Integer.
            parseInt(patientid));
       String[] currentSessionRoleList=
            (String[]) session.
            getAttribute("
            currentDatabaseList");
       for(int i=0; i<
            currentSessionRoleList.length
            ;i++){
       for(int j=0;j<potentialOwnerList.
            size();j++){
       System.out.println("PUMASOK BA?
            "+potentialOwnerList.get(j)+"
             " + currentSessionRoleList[i
            ]);
       if(currentSessionRoleList[i].
            equals(potentialOwnerList.get
            (j).trim()) ||
            potentialOwnerList.get(j).
            trim().equals(session.
            getAttribute("sessionUser").
            toString())){
       System.out.println("PACHECK NAMAN
            : "+potentialOwnerList.get(j
      ).trim());
privilegeCheck= adminSP.
      checkPrivilege(
      currentSessionRoleList[i],
      formname.toLowerCase(), "
      insert");
if(privilegeCheck){
currentSessionRole=
      currentSessionRoleList[i];
break;
}// end if privilegeCheck
      condition
}// end if equal condition
}//end of j loop
} //end of i loop
if(currentSessionRole!=null && !
      currentSessionRole.isEmpty()
      && !currentSessionRole.equals
      ("") ){
dentFormsSP.setDatabase_username(
      currentSessionRole);

/**
END OF MULTI-ROLE
**/
System.out.println("Current Role:
       "+currentSessionRole);
dentFormsSP.executeForms(formname
      ,fields,values,Long.parseLong
      (patientid),session);


/**
START OF AUDIT TRAIL
**/
adminSP= new AdminSP();
adminSP.setDatabase_username(
      session.getAttribute("
      sessionUserRole").toString())
      ;
UserSP userSP= new UserSP();
userSP.setDatabase_username(
      session.getAttribute("
      sessionUserRole").toString())
      ;

//date time
DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy
      ");
Date date = new Date();
System.out.println(dateFormat.
      format(date));
String dateString= dateFormat.
      format(date).toString();
//end for date time

//audittrail
int sessionUserID= Integer.
      parseInt(session.getAttribute
      ("sessionUserId").toString())
      ;
User sessionUser= userSP.getUser(
      sessionUserID);
Patient patient= dentFormsSP.
      getPatient(Integer.parseInt(
      patientid));
/*
getname for patient?
*/

String action_performed=patient.
      getName();//patientname
String action_encounter="";
int version=dentFormsSP.
      getCurrentVersion(Integer.
      parseInt(patientid),formname)
      ;
if(version==1)
action_encounter="INSERT";
else
action_encounter="UPDATE";
String sessionName= sessionUser.
      getFname_user()+" "+
      sessionUser.getMinit_user() +
       " "+ sessionUser.
      getLname_user()+" ("+
      sessionUser.getUsername()+")
      ";
adminSP.insertAuditTrail(
      sessionName,action_encounter,
       action_performed, formname,
            dateString);
        //end of audittrail
        /**
        END OF AUDITTRAIL
        **/
        if(cont!=null){
        //dentFormsSP.updateInstanceID(
            patientid, dentFormsSP.
            getInstanceID());
        }
        mav.addObject("notif","success");
        if(request.getParameter("app")!=
            null){
        mav.addObject("app","app");
        }
        } //end section priv check
        else{
        ModelAndView mav2 =new
            ModelAndView("searchPatient")
            ;
        return mav2;
        } //else part
        }
        /**
        END OF PRIVILEGE CHECK
        */
      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
      }
}
      if(_e_pID==null){
      String formurl="task?idTask="+
           request.getParameter("idTask")
           +"&nameTask="+formname+"&
           username=krisv&password=krisv&
           patientid="+patientid+"&dashid
           ="+request.getParameter("dashid
           ")+"&dashname="+request.
           getParameter("dashname");
      mav.addObject("formurl",formurl);
      }
      /**START OF FOR APPROVE STATUS*/
      if(!session.getAttribute("
           sessionUserRole").toString().
           toLowerCase().contains("fac")
           && !formname.equals("Patient"))
           {

         //dentFormsSP.
              updateApprovedStat(
              patientid, key, session.
              getAttribute("sessionName")
              .toString());
         int version=dentFormsSP.
              getCurrentVersion(Integer.
              parseInt(patientid),
              formname);
         dentFormsSP.updateApproved(
              patientid, formname, "For
              Approval",version);
         for(int i=1;i<version;i++){
       dentFormsSP.updateApproved(
            patientid, formname, " ",i);
             }
 }
  /**END OF FOR APPROVE STATUS*/

 }



    //mav.addObject("message", resp);
 if(_e_pID==null && cont!=null && cont.
      equals("Save Remarks") ){
  String formname=request.getParameter("
      formname");
  while(!deletetasks()){

  }
  String taskid=dentFormsSP.getidtask(
       patientid);
  while(taskid.equals("")){
   taskid=dentFormsSP.getidtask(
        patientid);
  }
     String formurl="task?idTask="+
          taskid+"&nameTask="+formname+"&
          patientid="+patientid+"&dashid
          ="+request.getParameter("dashid
          ")+"&dashname="+request.
```

```
            getParameter("dashname");
        mav.addObject("formurl",formurl);
        mav.addObject("notif",null);
        }

        String pos=request.getParameter("
            pos");
        if(pos==null)pos="0";


        mav.addObject("patientid",patientid
            );
        mav.addObject("instanceid",intid);
        mav.addObject("flag","true");
        mav.addObject("pos",pos);
        return mav;
    }
    public static String substringAfter(
        String str, String separator) {

        if (separator == null) {
            return "";
        }
        int pos = str.indexOf(separator);
        if (pos == -1) {
            return "";
        }
        return str.substring(pos + separator
            .length());
    }
    public String getstatus(String id)
        throws Exception{
  //list of patients


   //int patientidint=Integer.parseInt(
       patientid);
   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
    conn.setAutoCommit(false);
    String status="";
    try{
     String update="";

     update="SELECT * FROM task WHERE id="+
         id;

     System.out.println("update statement="+
         update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();
    DentFormsSP dentforms=new DentFormsSP();

    while(rs.next()){

     status = rs.getString("status");
     if(status.equalsIgnoreCase("Reserved"))
         {
      break;
     }

      }

    System.out.println("Forms started");


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }
    return status;
 }

    public boolean deleteAppointment(String
        patientid) throws Exception{
  //list of patients
```

```
   //int patientidint=Integer.parseInt(
       patientid);
   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","abjarabelo
         ","yes?bakitpo?");
    conn.setAutoCommit(false);
    boolean ended=false;
    String end_date="";
    try{
     String update="";

     update="DELETE FROM setappointment
         WHERE patientid="+patientid;

     System.out.println("update statement="+
         update);
     Statement st = conn.createStatement();
     boolean rs = st.execute(update);
     conn.commit();




    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }
    return ended;
 }
    public boolean deletetasks() throws
        Exception{
  //list of patients
   boolean deleted=false;

   //int patientidint=Integer.parseInt(
       patientid);
   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="";

     update="SELECT * FROM task WHERE status
         ='Completed'";

     System.out.println("update statement="+
         update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();
    DentFormsSP dentforms=new DentFormsSP();
    while(rs.next()){

          String taskid= rs.getString("id");
        dentforms.deleteTask(taskid);
        deleted=true;
      }

    System.out.println("Forms started");


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }
    return deleted;
 }
}
```

```
package org.dentist.version.three.web.
    controller;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.
     annotation.Autowired;
import org.springframework.stereotype.
     Controller;
import org.springframework.web.bind.
     annotation.ModelAttribute;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;
import org.springframework.web.bind.
     annotation.RequestParam;
import org.springframework.web.bind.
     annotation.ResponseBody;
import org.springframework.web.servlet.
     ModelAndView;

import org.apache.commons.lang.StringUtils;
import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.
     processserver.JbpmAPIUtil;
import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.ProcessDefinitionsRS
     ;
import org.dentist.version.three.
     processserver.model.UserTaskVO;
import org.dentist.version.three.
     processserver.service.Process;
import org.jbpm.task.Task;
import org.jbpm.task.query.TaskSummary;
import org.jbpm.userprofile.
     OrganizationalEntity;

import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.User;

@Controller
public class GreetController {

//     @Autowired
//     private UserDao userDao;

    @RequestMapping(value = "greet",method =
         RequestMethod.GET)
    public
    //@ModelAttribute("message")
    ModelAndView getInitialMessage(
         HttpServletRequest request,
         HttpSession session) throws
         Exception{
     DefinitionsRS defnitionRS = new
          DefinitionsRS();
     defnitionRS.setId("com.sample.humantask
          ");
     ArrayList<String> roles = new ArrayList
          <String>();
     ArrayList<String> workflowadmin = new
          ArrayList<String>();
     for (Object object : Arrays.asList((
          String[])session.getAttribute("
          currentDatabaseList"))) {
      if(!object.toString().toLowerCase().
           contains("workflow")){
       roles.add(object != null ? object.
            toString() : null);
          System.out.println("Role: "+
               object.toString());
      }
      else{
       workflowadmin.add(object != null ?
            object.toString() : null);

      }
    }
    java.util.List<Task> tasks =
         JbpmAPIUtil.getAssignedGroupTasks(
         session.getAttribute("sessionUser")
         .toString(), roles);
    java.util.List<Task> task= new
         ArrayList();
java.util.List<Long> taskIDs= new
     ArrayList();
ArrayList<Patient> patientlist = new
     ArrayList<Patient>();
ArrayList<String> tasknames = new
     ArrayList<String>();
//String instanceid=request.getParameter("
     instanceid");
DentFormsSP dentFormsSP= new DentFormsSP()
     ;
//TaskSummary task = null;
HashMap<String,String> map=TaskController.
     formlist;
    if(map.isEmpty()){
     map=SearchController.formlist;
     if(map.isEmpty()){
      map=ViewController.formlist;
     }
    }
for(int i=0;i<tasks.size();i++){

 if (!taskIDs.contains(tasks.get(i).getId
      ())){
  System.out.println(tasks.get(i).getId()
       +"*"+tasks.get(i).getTaskData().
       getStatus()+"*");
  taskIDs.add(tasks.get(i).getId());
  task.add(tasks.get(i));
  tasknames.add(getFormName(Long.toString(
       tasks.get(i).getId())));
  patientlist.add(getPatient(Long.toString
       (tasks.get(i).getTaskData().
       getProcessInstanceId())));
 }


}
    if(!workflowadmin.isEmpty()){
     java.util.List<Task> wtasks =
          JbpmAPIUtil.getAssignedGroupTasks(
          session.getAttribute("sessionUser
          ").toString(), workflowadmin);
     java.util.List<Task> wtask= new
          ArrayList();
     java.util.List<Long> wtaskIDs= new
          ArrayList();
     ArrayList<Patient> wpatientlist = new
          ArrayList<Patient>();
     ArrayList<String> wtasknames = new
          ArrayList<String>();
     for(int i=0;i<wtasks.size();i++){
      String assigned=null;
  for (org.jbpm.task.OrganizationalEntity
       owner : wtasks.get(i).
       getPeopleAssignments().
       getPotentialOwners()) {
   if(!owner.toString().trim().toLowerCase
        ().contains("workflow")){
     assigned=StringUtils.substringBetween(
          owner.toString().trim(),"'","'");
   }
  }
  UserSP userSP=new UserSP();
  int user_id=userSP.getUserID(assigned);
  System.out.println("USER :"+assigned
       +"|");
  User user=userSP.getUser(user_id);
     if (!wtaskIDs.contains(wtasks.get(i).
          getId()) && !session.getAttribute
          ("sessionUser").toString().
          equalsIgnoreCase(assigned)){
      System.out.println(wtasks.get(i).
           getId()+"*"+wtasks.get(i).
           getTaskData().getStatus()+"*");
      wtaskIDs.add(wtasks.get(i).getId());
      wtask.add(wtasks.get(i));
      wtasknames.add(getFormName(Long.
           toString(wtasks.get(i).getId()))
           );
      Patient p=getPatient(Long.toString(
           wtasks.get(i).getTaskData().
           getProcessInstanceId()));


      String clinician=user.getFname_user
```

```
            ()+" "+user.getLname_user();
        p.setName(p.getName()+"</td><td>"+
            clinician+"</td>");
        wpatientlist.add(p);
       }

      }
      tasks.addAll(wtasks);
      task.addAll(wtask);
      taskIDs.addAll(wtaskIDs);
      patientlist.addAll(wpatientlist);
      tasknames.addAll(wtasknames);
     }
     //removeDuplicateWithOrder((ArrayList)
         tasks);
     //ProcessDefinitionsRS pd = Process.
         instance().getDefinitions();
String cont =request.getParameter("
     continue");

String pos=request.getParameter("pos");
String formurl="";

//String patientid=request.getParameter("
     patientid");
//Patient patient=dentFormsSP.getPatient(
     Integer.parseInt(patientid));
session.setAttribute("definition", task);
ModelAndView mav = new ModelAndView("greet
     ");
     mav.addObject("definitions", task);
     //mav.addObject("patientid", patientid)
         ;
     //mav.addObject("instanceid",
         instanceid);
     mav.addObject("flag", request.
         getParameter("flag"));
     mav.addObject("pos",pos);
     mav.addObject("formurl",formurl );
     mav.addObject("patientlist",patientlist
         );
     mav.addObject("name",tasknames );
        return mav;
    }
    @RequestMapping(value = "greet",method =
         RequestMethod.POST)
    public @ResponseBody
    //@ModelAttribute("message")
    String getInitialMessage(
         HttpServletRequest request,
         HttpSession session,
         HttpServletResponse response) throws
         Exception{
     DefinitionsRS defnitionRS = new
         DefinitionsRS();
     defnitionRS.setId("com.sample.humantask
         ");
     ArrayList<String> roles = new ArrayList
         <String>();
     ArrayList<String> workflowadmin = new
         ArrayList<String>();
     for (Object object : Arrays.asList((
         String[])session.getAttribute("
         currentDatabaseList"))) {
      if(!object.toString().toLowerCase().
           contains("workflow")){
       roles.add(object != null ? object.
           toString() : null);
          System.out.println("Role: "+
              object.toString());
      }
      else{
       workflowadmin.add(object != null ?
           object.toString() : null);

      }
     }
     java.util.List<Task> tasks =
         JbpmAPIUtil.getAssignedGroupTasks(
         session.getAttribute("sessionUser")
         .toString(), roles);
     java.util.List<Task> task= new
         ArrayList();
java.util.List<Long> taskIDs= new
     ArrayList();
ArrayList<Patient> patientlist = new
     ArrayList<Patient>();
ArrayList<String> tasknames = new
     ArrayList<String>();
//String instanceid=request.getParameter("
     instanceid");
DentFormsSP dentFormsSP= new DentFormsSP()
     ;
//TaskSummary task = null;
HashMap<String,String> map=TaskController.
     formlist;
    if(map.isEmpty()){
     map=SearchController.formlist;
     if(map.isEmpty()){
      map=ViewController.formlist;
     }
    }
for(int i=0;i<tasks.size();i++){

  if (!taskIDs.contains(tasks.get(i).getId
      ())){
   System.out.println(tasks.get(i).getId()
       +"*"+tasks.get(i).getTaskData().
       getStatus()+"*");
   taskIDs.add(tasks.get(i).getId());
   task.add(tasks.get(i));
   tasknames.add(getFormName(Long.toString(
       tasks.get(i).getId())));
   patientlist.add(getPatient(Long.toString
       (tasks.get(i).getTaskData().
       getProcessInstanceId())));
  }


}
if(!workflowadmin.isEmpty()){
     java.util.List<Task> wtasks =
         JbpmAPIUtil.getAssignedGroupTasks(
         session.getAttribute("sessionUser
         ").toString(), workflowadmin);
     java.util.List<Task> wtask= new
         ArrayList();
     java.util.List<Long> wtaskIDs= new
         ArrayList();
     ArrayList<Patient> wpatientlist = new
         ArrayList<Patient>();
     ArrayList<String> wtasknames = new
         ArrayList<String>();
     for(int i=0;i<wtasks.size();i++){
      String assigned=null;
   for (org.jbpm.task.OrganizationalEntity
       owner : wtasks.get(i).
       getPeopleAssignments().
       getPotentialOwners()) {
    if(!owner.toString().trim().toLowerCase
        ().contains("workflow")){
     assigned=StringUtils.substringBetween(
         owner.toString().trim(),"'","'");
    }
   }
   UserSP userSP=new UserSP();
   int user_id=userSP.getUserID(assigned);
   System.out.println("USER :"+assigned
       +"|");
   User user=userSP.getUser(user_id);
      if (!wtaskIDs.contains(wtasks.get(i).
          getId()) && !session.getAttribute
          ("sessionUser").toString().
          equalsIgnoreCase(assigned)){
       System.out.println(wtasks.get(i).
           getId()+"*"+wtasks.get(i).
           getTaskData().getStatus()+"*");
       wtaskIDs.add(wtasks.get(i).getId());
       wtask.add(wtasks.get(i));
       wtasknames.add(getFormName(Long.
           toString(wtasks.get(i).getId()))
           );
       Patient p=getPatient(Long.toString(
           wtasks.get(i).getTaskData().
           getProcessInstanceId()));


       String clinician=user.getFname_user
           ()+" "+user.getLname_user();
       p.setName(p.getName()+"</td><td>"+
           clinician+"</td>");
       wpatientlist.add(p);
      }


     }
     tasks.addAll(wtasks);
     task.addAll(wtask);
     taskIDs.addAll(wtaskIDs);
     patientlist.addAll(wpatientlist);
     tasknames.addAll(wtasknames);
    }

    String html="<br><br><br><table class
```

```
        =\"list-table\"><thead><tr class=\"
        head-list-table\"><td> </td><td>
        Task :</td> <td>Patient :</td>";

    if(StringUtils.join(((String[])session.
        getAttribute("currentDatabaseList")
        ),"").toLowerCase().contains("
        workflow")){
     html=html+"<td>Assigned Clinician :</
         td>";
    }
    html=html+"</tr></thead>";
    int loop=0;
    for (Task itask : task) {
     String row="";
     String output = tasknames.get(loop).
         replaceAll("(\\p{Ll})(\\p{Lu})","
         $1 $2");
     if(loop%2==0){row=row+"<tr class=\"
         even\">";}
     else{row=row+"<tr class=\"odd\">";}
     int id=loop+1;
     row=row+"<td>"+id+"</td>";
     row=row+"<td><a href=\"task?idTask="+
          itask.getId()+"&nameTask="+
          tasknames.get(loop)+"&patientid="+
          patientlist.get(loop).getId()
          +"\">"+output+"</a></td>";
     row=row+"<td>"+patientlist.get(loop).
         getName()+"</td>";
     System.out.println("ROW: "+row);
     html=html+row;
     loop++;
    }
    html=html+"";
    //removeDuplicateWithOrder((ArrayList)
        tasks);
    //ProcessDefinitionsRS pd = Process.
        instance().getDefinitions();
String cont =request.getParameter("
     continue");

String pos=request.getParameter("pos");
String formurl="";

//response.setCharacterEncoding("UTF-8");
      //response.setContentType("text/html
          ");
      //response.getWriter().write(html);
//String patientid=request.getParameter("
     patientid");
//Patient patient=dentFormsSP.getPatient(
     Integer.parseInt(patientid));
session.setAttribute("definition", task);
ModelAndView mav = new ModelAndView("greet
     ");
   mav.addObject("definitions", task);
   //mav.addObject("patientid", patientid)
       ;
   //mav.addObject("instanceid",
       instanceid);
   mav.addObject("flag", request.
       getParameter("flag"));
   mav.addObject("pos",pos);
   mav.addObject("formurl",formurl );
   mav.addObject("patientlist",patientlist
       );
   mav.addObject("name",tasknames );
   mav.addObject("html",html);
      return html;
  }
  @RequestMapping(method = RequestMethod.
      GET)
  public
  @ModelAttribute("message")
  String getGreeting(@RequestParam("
      username") String username) {

          return "No such user exists! Use
                'emuster' or 'jdoe'";
  }
  public static void
      removeDuplicateWithOrder(ArrayList
      arlList)
  {
  Set set = new HashSet();
  List newList = new ArrayList();
  for (Iterator iter = arlList.iterator();
          iter.hasNext(); ) {
  Object element = iter.next();
    if (set.add(element))
      newList.add(element);
        }
        arlList.clear();
        arlList.addAll(newList);
   }
    public Patient getPatient(String
         instanceid) throws Exception{
  //list of patients


   //int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);
    Patient patient=new Patient();
    try{
     String update="";

     update="SELECT patientid,convert_from(
          decrypt(givenname,'fooz','bf'),'
          UTF8') AS givenname, convert_from(
          decrypt(middlename,'fooz','bf'),'
          UTF8')  AS middlename,"+
     "convert_from(decrypt(familyname,'fooz
          ','bf'),'UTF8') AS familyname,
          upcdid,gender,convert_from(decrypt(
          birthdate,'fooz','bf'),'UTF8'),"+
    " convert_from(decrypt(address,'fooz','
        bf'),'UTF8'),convert_from(decrypt(
        address2,'fooz','bf'),'UTF8'),"+
     "convert_from(decrypt(city,'fooz','bf')
          ,'UTF8'),convert_from(decrypt(state
          ,'fooz','bf'),'UTF8'),"+
     "convert_from(decrypt(country,'fooz','
          bf'),'UTF8'),convert_from(decrypt(
          postalcode,'fooz','bf'),'UTF8'),"+
     "deceased,instanceid,status,datecreated
           FROM patient WHERE instanceid="+
          instanceid;

     System.out.println("update statement="+
          update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();

    if(rs.next()){

           patient.setName(rs.getString("
                givenname")+rs.getString("
                middlename")+rs.getString("
                familyname"));
           patient.setupcdId(rs.getString("
                upcdid"));
           patient.setId(rs.getString("
                patientid"));
      }

    System.out.println("Patient started");


    st.close();
    //rs.close();
    conn.close();

    }
    catch(Exception e){
     e.printStackTrace();
    }
    return patient;
}
    public String getFormName(String id)
         throws Exception{
  //list of patients

   String name="";
   //int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="";
```

```
      update="SELECT * FROM i18ntext WHERE
           task_names_id="+id;

      System.out.println("update statement="+
           update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();

    if(rs.next()){
           name=rs.getString("text");

       }
```

```
      System.out.println("Forms started");


      st.close();
      //rs.close();
      conn.close();
      }
      catch(Exception e){
       e.printStackTrace();
      }
      return name;
   }
 }
```

```
package org.dentist.version.three.web.
     controller;

import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.springframework.stereotype.
     Controller;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;
import org.springframework.web.bind.
     annotation.RequestParam;
import org.springframework.web.servlet.
     ModelAndView;

import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.InstancesRS;
import org.dentist.version.three.
     processserver.model.
     ProcessDefinitionInstancesRS;
import org.dentist.version.three.
     processserver.service.Process;

@Controller
public class ProcessController {

 @RequestMapping(value="process",method =
      RequestMethod.GET)
 public ModelAndView getProcess(
      @RequestParam("processId") String
      processId,HttpSession session,
      HttpServletRequest request)
 throws Exception {
  if(TaskController.formlist.isEmpty() &&
       SearchController.formlist.isEmpty() &&
        ViewController.formlist.isEmpty()){
   resetProcessLog();
   resetInstanceIds();
     }
  //IOUtils.copy(is, outputStream);

      DefinitionsRS defnitionRS = new
           DefinitionsRS();
      defnitionRS.setId(processId);
      defnitionRS = Process.instance().
           getDefinition(defnitionRS,session);
      //Process.instance().startInstace(
           defnitionRS);
      String htmlRender = Process.instance().
           getProcessRenderHTML(defnitionRS,
           session);
      ProcessDefinitionInstancesRS instances
           = Process.instance().
           getProcessInstances(defnitionRS,
           session);

      UPCDIDGenerator upcdidgenerator= new
           UPCDIDGenerator();

      String upcdid=upcdidgenerator.generate(
           session.getAttribute("
           sessionUserRole").toString());
```

```
       ModelAndView mav = new ModelAndView("
            process");
       mav.addObject("process", defnitionRS);
       mav.addObject("htmlRender", htmlRender)
            ;
       mav.addObject("instances", instances.
            getInstances());
       mav.addObject("processId", defnitionRS.
            getId());
       mav.addObject("patientid", request.
            getParameter("patientid"));
       mav.addObject("upcdid", upcdid);
       return mav;
 }
 public void resetProcessLog() throws
      Exception{
  //list of patients


    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt","jgerona","
           bakitba?");
     conn.setAutoCommit(false);

     try{
      String update="DELETE FROM
           processinstancelog";
      System.out.println("delete statement="+
           update);
      Statement st = conn.createStatement();
      boolean rs = st.execute(update);
      conn.commit();
      //System.out.println("i18ntext RS="+rs)
           ;


     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
      e.printStackTrace();
     }

 }
 public static void resetInstanceIds()
      throws Exception{
  //list of patients


    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt","jgerona","
           bakitba?");
     conn.setAutoCommit(false);

     try{
     String update="UPDATE \"patient\" SET
          instanceid=0";
     System.out.println("update statement="+
          update);
     Statement st = conn.createStatement();
     int rs = st.executeUpdate(update);
     conn.commit();
     System.out.println("Your data has been
          updated into table. RS="+rs);
     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
```

```
    e.printStackTrace();
   }
 }

}
```

```
package org.dentist.version.three.web.
    controller;

import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest
    ;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.springframework.stereotype.
    Controller;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;
import org.springframework.web.bind.
    annotation.RequestParam;
import org.springframework.web.servlet.
    ModelAndView;

import org.dentist.version.three.db.
    DentFormsSP;
import org.dentist.version.three.
    processserver.model.DefinitionsRS;
import org.dentist.version.three.
    processserver.model.InstancesRS;
import org.dentist.version.three.
    processserver.model.
    ProcessDefinitionInstancesRS;
import org.dentist.version.three.
    processserver.service.Process;

@Controller
public class ReturningPatientController {
 @RequestMapping(value="returningpatient",
     method = RequestMethod.GET)
 public ModelAndView getProcess(
     @RequestParam("processId") String
     processId,HttpSession session,
     HttpServletRequest request)
 throws Exception {

  //IOUtils.copy(is, outputStream);

     DefinitionsRS defnitionRS = new
         DefinitionsRS();


     ModelAndView mav = new ModelAndView("
         returningpatient");

    // return mav;



     DentFormsSP dentFormsSP= new
         DentFormsSP();
     Map<String, Object> params = new
         HashMap<String, Object>();
     String resp;
     String patientid=request.getParameter("
         patientid");

     dentFormsSP.setDatabase_username(
         session.getAttribute("
         sessionUserRole").toString());

  ArrayList<String> clinicians=dentFormsSP.
      listAllClinicians();
  if(session.getAttribute("sessionUserRole")
      .toString().toLowerCase().contains("
      fac")){
         params.put("user", "Faculty");
         params.put("is_faculty", true);
         params.put("is_student", false);
         params.put("clinicians", clinicians
             );
         System.out.println("i am a faculty
             ");

         }
         else {
          params.put("user", "Student");
          params.put("is_faculty", false);
          params.put("is_student", true);
          System.out.println("i am a student
              ");

         }
  params.put("rejectMsg", "");
  defnitionRS.setId(processId);
  Process.instance().startInstace(
      defnitionRS,session);

  dentFormsSP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  int patientidint=Integer.parseInt(
      patientid);
  int instanceid=dentFormsSP.getInstanceID()
      ;
  System.out.println("Patient ID:"+
      patientidint+"Instance ID:"+instanceid
      +"*");
  dentFormsSP.updateInstanceID(patientid,
      instanceid);

  mav.addObject("patientid",patientid);
     mav.addObject("instanceid",Integer.
         toString(instanceid));
     mav.addObject("flag","true");

     return mav;
 }

}
```

```
package org.dentist.version.three.web.
    controller;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest
    ;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.dentist.version.three.db.
    DentFormsSP;
import org.dentist.version.three.
    processserver.HumanTaskStartupServlet;
import org.dentist.version.three.
    processserver.model.DefinitionsRS;
import org.dentist.version.three.
    processserver.model.Patient;
import org.dentist.version.three.
    processserver.service.Process;
import org.springframework.stereotype.
    Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;
import org.springframework.web.servlet.
    ModelAndView;

import com.dentist.version.three.db.AdminSP;
import com.dentist.version.three.db.RoleSP;
import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.Role;
import com.dentist.version.three.form.User;
```

```
@Controller
public class SearchController {
 public static HashMap<String,String>
      formlist=new HashMap<String,String>();;

 @RequestMapping(value = "/searchPatient",
      method = RequestMethod.GET)
public String viewSpecificUsers(
     HttpServletRequest request,
         HttpServletResponse response, Model
              model,HttpSession session)
              throws Exception {
  if(TaskController.formlist.isEmpty() &&
       formlist.isEmpty() && ViewController.
       formlist.isEmpty()){
   resetProcessLog();
   resetInstanceIds();
   deleteAppointment();

   DefinitionsRS defnitionRS = new
        DefinitionsRS();
          defnitionRS.setId("UPCDDentISt.
               Forms");
          Map<String, Object> params = new
               HashMap<String, Object>();
          Process.instance().startInstace(
               defnitionRS,session);
          getForms();
          formlist.remove("
               AssignToOralDiagnosisClinician
               ");


          ArrayList<User> userLists = new
               ArrayList<User>();
          ArrayList<Role> roleLists = new
               ArrayList<Role>();
          UserSP userSP= new UserSP();
          RoleSP roleSP= new RoleSP();
          boolean privilegeCheck=false;
          AdminSP adminSP= new AdminSP();
          String currentRole="";
          String[] currentRoleList= (String
               []) session.getAttribute("
               currentDatabaseList");
   for(int i=0; i<currentRoleList.length;i
        ++){
    privilegeCheck= adminSP.checkPrivilege(
         currentRoleList[i], "users", "select
         ");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
        privilegeCheck);
   userSP.setDatabase_username(currentRole);

  userLists= userSP.allUserList();
  roleLists= roleSP.allRoleList();
  try{
  for (User user : userLists) {
   /** start add new user to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addUser(new org.jbpm.task.User(user.
        getUsername()));
   /**end add new user to jbpm**/
      System.out.println("Add user: "+ user.
           getUsername());
     }
  for (Role role : roleLists) {
   /** start add groups to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addGroup(new org.jbpm.task.Group(
        roleSP.getDatabaseRole(role.
        getRole_id()).trim()));
   /**end add new groups to jbpm**/
      System.out.println("Add group: "+
           roleSP.getDatabaseRole(role.
           getRole_id()).trim());
     }
  }
  catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM ERROR");
  }
  if(TaskController.formlist.isEmpty() &&
       formlist.isEmpty() && ViewController.
       formlist.isEmpty()){
   response.sendRedirect("searchPatient");
  }
     }
  HashMap<String,String> map=formlist; //FOR
       APPROVE STATUS
      if(map.isEmpty()){
       map=ViewController.formlist;
       if(map.isEmpty()){
        map=TaskController.formlist;
       }
      }
 session=request.getSession(false);

 String patientname=request.getParameter("
      patientsearch");

 ArrayList<Patient> patientLists = new
      ArrayList<Patient>();

 if(patientname!=null){
  model.addAttribute("search","search");
 DentFormsSP dentSP= new DentFormsSP();
 String nameSearch= "%"+patientname+"%";
 System.out.println("CHECK:: "+nameSearch);
 try {
   dentSP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());
/*
   patientLists= dentSP.specifiedPatientList
        (nameSearch);

   for (Patient patient : patientLists) {
    if(hasEnded(patient.getinstanceid())){
      dentSP.updateInstanceID(patient.getId
           (), 0);
    }
     }
   patientLists= dentSP.specifiedPatientList
        (nameSearch);
   */
   /*
   START OF SEARCH CHECK

   */
      boolean privilegeCheck=false;
      String errorMessage="";
      String currentRole="";
       String[] currentRoleList= (String[])
            session.getAttribute("
            currentDatabaseList");
        AdminSP adminSP= new AdminSP();
       for(int i=0; i<currentRoleList.length;
            i++){
        privilegeCheck= adminSP.
             checkPrivilege(currentRoleList[i
             ], "patient_specificsection", "
             select");
        if(privilegeCheck){
         currentRole=currentRoleList[i];

         break;
        }
       }

      if(privilegeCheck){
     //
    dentSP.setDatabase_username(currentRole)
        ;
   patientLists=dentSP.list_specifiedsection
        (currentRole,nameSearch);


   }else{
   //normalsection
       patientLists= dentSP.
            specifiedPatientList(nameSearch);

   }


       for (Patient patient : patientLists) {
        if(hasEnded(patient.getinstanceid()))
             {
          dentSP.updateInstanceID(patient.
               getId(), 0);
         }
          }

      if(privilegeCheck){
      //
       dentSP.setDatabase_username(
            currentRole);
        patientLists=dentSP.
             list_specifiedsection(currentRole
             ,nameSearch);


       }else{
       //normalsection
        patientLists= dentSP.
             specifiedPatientList(nameSearch);

       }

      // patientLists= dentSP.
           specifiedPatientList(nameSearch);


       /**
       END OF SEARCH CHECK

       */
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
  System.out.println("BOOM ERROR");
  return "Error";
 }
 }
 model.addAttribute("patientLists",
      patientLists);
 model.addAttribute("formurl",map.get("
      PatientInformation"));



 return "searchPatient";

}
 public void resetProcessLog() throws
      Exception{
  //list of patients


   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="DELETE FROM
          processinstancelog";
     System.out.println("delete statement="+
          update);
     Statement st = conn.createStatement();
     boolean rs = st.execute(update);
     conn.commit();
     //System.out.println("i18ntext RS="+rs)
          ;
     update="DELETE FROM processinstanceinfo
          ";
     System.out.println("delete statement="+
          update);
     st = conn.createStatement();
     rs = st.execute(update);
     conn.commit();

    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
 public static void resetInstanceIds()
      throws Exception{
  //list of patients


   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);

    try{
```

```
    String update="UPDATE \"patient\" SET
         instanceid=0";
    System.out.println("update statement="+
         update);
    Statement st = conn.createStatement();
    int rs = st.executeUpdate(update);
    conn.commit();
    System.out.println("Your data has been
         updated into table. RS="+rs);
    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
 public static void getForms() throws
      Exception{
  //list of patients


   //int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="";

     update="SELECT * FROM i18ntext";

     System.out.println("update statement="+
          update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();

    while(rs.next()){

           String url="task?idTask="+rs.
                getString("task_names_id")+"&
                nameTask="+rs.getString("text
                ");
           formlist.put(rs.getString("text")
                , url);
      }

    System.out.println("Forms started");


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
  public boolean deleteAppointment() throws
       Exception{
   //list of patients


    //int patientidint=Integer.parseInt(
         patientid);
    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt","jgerona
           ","bakitba?");
     conn.setAutoCommit(false);
     boolean ended=false;
     String end_date="";
     try{
      String update="";

      update="DELETE FROM setappointment";

      System.out.println("update statement
           ="+update);
      Statement st = conn.createStatement();
      boolean rs = st.execute(update);
      conn.commit();
      st.close();
      //rs.close();
      conn.close();
      }
      catch(Exception e){
       e.printStackTrace();
      }
      return ended;
  }
  public boolean hasEnded(String instanceid)
        throws Exception{
   //list of patients


     //int patientidint=Integer.parseInt(
          patientid);
     Class.forName("org.postgresql.Driver").
          newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql://
            localhost:5432/DentISt","jgerona
            ","bakitba?");
      conn.setAutoCommit(false);
      boolean ended=false;
      String end_date="";
      try{
       String update="";
       update="SELECT * FROM task WHERE
            processinstanceid="+instanceid+"
            AND status !='Completed'";

       System.out.println("update statement
            ="+update);
      Statement st = conn.createStatement();
      ResultSet rs = st.executeQuery(update);
      conn.commit();
      DentFormsSP dentforms=new DentFormsSP()
           ;

      if(!rs.next()){


        ended=true;

        System.out.println("Process ended");

        }




      st.close();
      //rs.close();
      conn.close();
      }
      catch(Exception e){
       e.printStackTrace();
      }
      return ended;
  }
}
```

```
package org.dentist.version.three.web.
     controller;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.
     Controller;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;
import org.springframework.web.bind.
     annotation.RequestParam;
import org.springframework.web.servlet.
     ModelAndView;

import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.db.
     DentalChartSP;
import org.dentist.version.three.form.
     CariesStatus;
import org.dentist.version.three.form.
     DentalChart;
import org.dentist.version.three.form.
     RecurrentStatus;
import org.dentist.version.three.form.
     RestorationStatus;
import org.dentist.version.three.form.
     ServiceNeeded;
import org.dentist.version.three.
     processserver.HumanTaskStartupServlet;
import org.dentist.version.three.
     processserver.JbpmAPIUtil;
import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.TaskRS;
import org.dentist.version.three.
     processserver.model.TaskUserRS;
import org.dentist.version.three.
     processserver.model.UserTaskVO;
import org.dentist.version.three.
     processserver.model.Version;
import org.dentist.version.three.
     processserver.service.Process;
import org.jbpm.task.Content;
import org.jbpm.task.Task;
import org.jbpm.task.service.
     TaskClientHandler.GetTaskResponseHandler
     ;
import org.jbpm.task.service.
     responsehandlers.
     BlockingTaskSummaryResponseHandler;
import org.apache.commons.lang.StringUtils;

import com.dentist.version.three.db.AdminSP;
import com.dentist.version.three.db.RoleSP;
import com.dentist.version.three.db.
     SectionSP;
import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.Role;
import com.dentist.version.three.form.User;
@Controller
public class TaskController {
 public static HashMap<String,String>
      formlist=new HashMap<String,String>();;

 @RequestMapping(value="task",method =
      RequestMethod.GET)
 public ModelAndView getProcess(HttpSession
      session,HttpServletRequest request,
      @RequestParam("idTask") String idTask,
      @RequestParam("nameTask") String
      nameTask)
 throws Exception {
  if(formlist.isEmpty() && SearchController.
       formlist.isEmpty() && ViewController.
       formlist.isEmpty()){
   resetProcessLog();
   resetInstanceIds();
   DefinitionsRS defnitionRS = new
        DefinitionsRS();
          defnitionRS.setId("UPCDDentISt.
               Forms");
          Map<String, Object> params = new
               HashMap<String, Object>();
          Process.instance().startInstace(
               defnitionRS,session);
          getForms();
          formlist.remove("
               AssignToOralDiagnosisClinician
               ");
          ArrayList<User> userLists = new
               ArrayList<User>();
          ArrayList<Role> roleLists = new
               ArrayList<Role>();
          UserSP userSP= new UserSP();
        RoleSP roleSP= new RoleSP();

        boolean privilegeCheck=false;
        AdminSP adminSP= new AdminSP();
        String currentRole="";
        String[] currentRoleList= (String
             []) session.getAttribute("
             currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
      ++){
  privilegeCheck= adminSP.checkPrivilege(
       currentRoleList[i], "users", "select
       ");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
      privilegeCheck);
 userSP.setDatabase_username(currentRole);

 userLists= userSP.allUserList();
 roleLists= roleSP.allRoleList();
 try{
  for (User user : userLists) {
   /** start add new user to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addUser(new org.jbpm.task.User(user
        .getUsername()));
   /**end add new user to jbpm**/
      System.out.println("Add user: "+
           user.getUsername());
     }
  for (Role role : roleLists) {
   /** start add groups to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addGroup(new org.jbpm.task.Group(
        roleSP.getDatabaseRole(role.
        getRole_id()).trim()));
   /**end add new groups to jbpm**/
      System.out.println("Add group: "+
           roleSP.getDatabaseRole(role.
           getRole_id()).trim());
     }
  }
  catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM ERROR");
  }
   }
deletetasks();
ModelAndView mav = new ModelAndView("task
      ");
String msg=request.getParameter("message")
      ;
String notif=request.getParameter("notif")
      ;
DentFormsSP dentFormsSP= new DentFormsSP()
      ;
AdminSP adminSP= new AdminSP();
int counter=-1;
if(getstatus(idTask).equals("") && msg==
      null){
 ModelAndView mav2 =new ModelAndView("
      greet");
 return mav2;
}

HashMap<String,String> map=formlist; //FOR
       APPROVE STATUS
   if(map.isEmpty()){
    map=SearchController.formlist;
     if(map.isEmpty()){
      map=ViewController.formlist;
     }
   }

if(msg==null){
UserTaskVO userTaskVO = new UserTaskVO("
      krisv", "krisv");
TaskRS taskRS = new TaskRS();
taskRS.setId(idTask);
userTaskVO.setTaskRS(taskRS);
//taskRS = Process.instance().getUserTask(
      userTaskVO);
//userTaskVO.setTaskRS(taskRS);
int patientid=Integer.parseInt(request.
      getParameter("patientid"));
DentFormsSP dentSP= new DentFormsSP();
String version=request.getParameter("
      version");
Patient patient=dentSP.getPatient(
      patientid);
String html = Process.instance().
     getTaskRenderHTML(userTaskVO,session);
counter = StringUtils.countMatches(html,
     "\"Save Remarks\"");

String menu="";
      String dash="";

if(!nameTask.equals("DentalChart")){
 String select="";
 int sectionid=1;
 if(html.contains("<select name=\"
      ODclinicians\">")){
  select="<select name=\"ODclinicians\">";
  sectionid=1;
 }
 else if(html.contains("<select name=\"
      ORALMEDclinicians\">")){
  select="<select name=\"ORALMEDclinicians
       \">";
  sectionid=3;
 }
 else if(html.contains("<select name=\"
      PROSTHOclinicians\">")){
  select="<select name=\"PROSTHOclinicians
       \">";
  sectionid=4;
 }
 else if(html.contains("<select name=\"
      OPERATIVEDENTclinicians\">")){
  select="<select name=\"
       OPERATIVEDENTclinicians\">";
  sectionid=5;
 }
 if(html.contains(select) && !select.
      equals("")){
  SectionSP sectionSP= new SectionSP();
  ArrayList<Role> currentRoleLists= new
       ArrayList<Role>();
  ArrayList<ArrayList<User>>
       currentUserLists= new ArrayList<
       ArrayList<User>>();
  String formTag = StringUtils.
       substringBetween(html, "href=\"",
       "\"");


  currentRoleLists= sectionSP.
       getListRoleSection(sectionid);

  for (Role currentRoleListss :
       currentRoleLists) {
  currentUserLists.add(sectionSP.
       getListUserSection(currentRoleListss
       .getRole_id()));
  }
  String options="";
  for (ArrayList<User> currentUserList :
       currentUserLists) {
   for (User currentUser : currentUserList
        ) {
    options=options+"<option value=\""+
         currentUser.getUsername().trim()
         +"\">"+currentUser.getFname_user()
         +" "+currentUser.getLname_user()
         +"</option>";
   }
  }
  html=html.replaceAll(select, "<select
       name=\"clinician\">"+options);
 }


   String findStr = "href=\"";
   int lastIndex = 0;
   int count = StringUtils.countMatches(
        html, "href");

     String href="task?idTask="+request.
          getParameter("idTask")+"&nameTask
          ="+nameTask+"&patientid="+
          patientid+"&username=krisv&
          password=krisv";
    if(count!=0){
     dash= dash+"<a href='"+href+"'>
          Dashboard</a>";
    }
while ((lastIndex = html.indexOf(findStr,
     lastIndex)) != -1) {
```

```
 String formTag = StringUtils.
      substringBetween(html.substring(
      lastIndex, html.length()-1), "href
      =\"", "\"");
 formTag=formTag.trim();
 if(map.containsKey(formTag)){
  String formTagFinal="href=\""+formTag
       +"\"";
  System.out.println(formTag+" is replaced
        by "+formTagFinal);
  String subs=(String)map.get(formTag)+"&
       patientid="+request.getParameter("
       patientid")+"&dashid="+request.
       getParameter("idTask")+"&dashname="+
       nameTask;
  String formTagFinal2="href=\""+subs
       +"\"";

  if(counter!=0 || request.getParameter("
       app")!=null){subs=subs+"&app=app";
  formTagFinal2="href=\""+subs+"\" target
       =\"_blank\" title=\"This form will
       open in a new tab\"";

  }

  System.out.println(formTag+" "+subs);
  html=html.replaceAll(formTagFinal,
       formTagFinal2);
  String str=formTagFinal2;
  lastIndex += str.length() - 1;
 }
 else{
  String formTagFinal="href=\""+formTag
       +"\"";
  System.out.println(formTag+" is replaced
        by #");
  System.out.println(formTag+" # ");
  html=html.replaceAll(formTag, "#");
  String str="#";
  lastIndex += str.length() - 1;
 }
    }
if(!getstatus(idTask).equalsIgnoreCase("
     Reserved") && html.contains("value=\"
     Claim\"")){
    html=html.replaceAll("task?", "viewRec
         ");
    mav.addObject("viewOnly", "true");

}
}




    String formname=nameTask;
      String values="";
      String tablename=formname;


      if(formname.equals("DentalChart")){
       int patient_id=patientid;

       boolean privilegeCheck=false;
     String errorMessage="";
     String currentRole="";


     ArrayList<Integer> listpatientids= new
          ArrayList<Integer>();
     boolean patient_exist=false;
     DentalChartSP dentalChartSP= new
          DentalChartSP();
     DentalChart dentalChart= new
          DentalChart();
     CariesStatus cariesStatus= new
          CariesStatus();
     RecurrentStatus recurrentStatus= new
          RecurrentStatus();
     RestorationStatus restorationStatus=
          new RestorationStatus();
     ServiceNeeded services= new
          ServiceNeeded();
     try {


      dentalChartSP.setDatabase_username(
           session.getAttribute("
           sessionUserRole").toString());
      listpatientids= dentalChartSP.
           getpatientIDList();
```

```
       if(!listpatientids.isEmpty() ||
            listpatientids!=null){
       for(Integer patientidss :
            listpatientids){
        if(patientidss==patient_id)
         patient_exist=true;
       }
      }
      if(patient_exist){
       int ver;
          if(version!=null){
           ver=Integer.parseInt(version);
          }
          else{
           ver=dentFormsSP.getCurrentVersion
                (patientid,"dentalchart");
          }
       System.out.println("Patient exists");
       dentalChart=dentalChartSP.
            getDentalChart(patient_id, ver);
       cariesStatus=dentalChartSP.
            getCariesStatus(patient_id,ver);
       recurrentStatus=dentalChartSP.
            getRecurrentStatus(patient_id,ver
            );
       restorationStatus=dentalChartSP.
            getRestorationStatus(patient_id,
            ver);
       services=dentalChartSP.
            getServiceNeeded(patient_id,ver);

       System.out.println("Dental Chart: "+
            dentalChart.getDental_chart_id())
            ;
       System.out.println("CariesStatus
            Chart: "+cariesStatus.
            getCaries_id());
       System.out.println("RecurrentStatus
            Chart: "+recurrentStatus.
            getrecurrent_id());
       System.out.println("RestorationStatus
             Chart: "+restorationStatus.
            getrestoration_id());
       System.out.println("Service Needed:
            "+ services.getServiceneeded_id()
            );

          mav.addObject("dentalChart",
               dentalChart);
          mav.addObject("cariesStatus",
               cariesStatus);
          mav.addObject("recurrentStatus",
               recurrentStatus);
          mav.addObject("restorationStatus",
                restorationStatus);
          mav.addObject("restorationStatus",
                restorationStatus);
          mav.addObject("services", services
               );
          mav.addObject("is_current",
               dentalChart.getIs_current());
          mav.addObject("version",
               dentalChart.getVersion());

         }
         else{
          mav.addObject("version",1);
          mav.addObject("is_current", "yes")
               ;

         }

         } catch (Exception e) {
          // TODO Auto-generated catch block

          e.printStackTrace();
         }

       }
       else{

try {
 if(version!=null){
   values= dentFormsSP.getVersionFormFields(
        patientid,tablename,Integer.parseInt(
        version));

   if(html.contains("tablelist")){
    ArrayList<Version> tableLists = new
         ArrayList<Version>();
    tableLists= dentFormsSP.listAllVersion(
         patientid, formname);
    ArrayList<String> valueLists = new
         ArrayList<String>();
    String vals="";
    mav.addObject("tableLists", tableLists);
    for (Version ver : tableLists) {
     vals= dentFormsSP.getVersionFormFields(
          patientid,tablename,Integer.
          parseInt(ver.getVersion()));
     if(ver.getApproved().equalsIgnoreCase("
          Approved") || ver.getApproved().
          equalsIgnoreCase("For Approval")){
      if(!nameTask.equalsIgnoreCase("
           ConsultationsAndFindings")){
      valueLists.add(vals);
      }
      else{
       int count=0;
       String findings="";
       String tokenizevals=vals;
       StringTokenizer st = new
            StringTokenizer(tokenizevals,
            "|");
       String date="";
       while(st.hasMoreTokens() && count<4)
            {
            findings=st.nextToken();

            System.out.println(findings + "-
                 column added -" + count);
            count++;
           }
       count=0;
       st = new StringTokenizer(tokenizevals
            , "|");

       while(st.hasMoreTokens() && count<9)
            {
            date=st.nextToken();

            System.out.println(findings + "-
                 column added -" + count);
            count++;
           }
       System.out.println(findings + "-eto
            na column added -"+ count);
       if(!findings.trim().equals("")){
        valueLists.add(date+" | "+vals);
       }
      }
     }
    }
    int count = StringUtils.countMatches(
         vals, "|");
    count=count-8;
    mav.addObject("colCount", count);
    mav.addObject("valueLists", valueLists);
    //values= dentFormsSP.getFormFields(
         patientid,tablename);
   }
  }
  else if(html.contains("tablelist")){
   ArrayList<Version> tableLists = new
        ArrayList<Version>();
   tableLists= dentFormsSP.listAllVersion(
        patientid, formname);
   ArrayList<String> valueLists = new
        ArrayList<String>();
   String vals="";
   mav.addObject("tableLists", tableLists);
   for (Version ver : tableLists) {
    vals= dentFormsSP.getVersionFormFields(
         patientid,tablename,Integer.parseInt(
         ver.getVersion()));
    if(ver.getApproved().equalsIgnoreCase("
         Approved") || ver.getApproved().
         equalsIgnoreCase("For Approval")){
     if(!nameTask.equalsIgnoreCase("
          ConsultationsAndFindings")){
     valueLists.add(vals);
     }
     else{
      int count=0;
      String findings="";
      String tokenizevals=vals;
      StringTokenizer st = new
           StringTokenizer(tokenizevals, "|");
      String date="";
      while(st.hasMoreTokens() && count<4) {
           findings=st.nextToken();

           System.out.println(findings + "-
                column added -" + count);
           count++;
          }
      count=0;
      st = new StringTokenizer(tokenizevals,
           "|");

      while(st.hasMoreTokens() && count<9) {
           date=st.nextToken();

           System.out.println(findings + "-
                column added -" + count);
           count++;
          }
      System.out.println(findings + "-eto na
           column added -"+ count);
      if(!findings.trim().equals("")){
       valueLists.add(date+" | "+vals);
      }
      else{
       mav.addObject("msginputfindings", "*
            Please input findings");
      }
     }
    }
   }
   int count = StringUtils.countMatches(vals,
        "|");
   count=count-8;
   mav.addObject("colCount", count);
   mav.addObject("valueLists", valueLists);
   values= dentFormsSP.getFormFields(
        patientid,tablename);
  }
  else {
   try{
       values= dentFormsSP.getFormFields(
            patientid,tablename);
   }
   catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
  }

      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
  }
         if(!nameTask.equalsIgnoreCase("
              AssignToOralDiagnosisClinician")
              && map.containsKey(nameTask)){
           if(request.getParameter("dashid")!=
                null){
            if(request.getParameter("app")!=
                 null && request.getParameter("
                 app").equals("app")){
             mav.addObject("app", "app");
            }
            else{

             menu = menu + "<a href='task?
                  idTask="+request.getParameter
                  ("dashid")+"&nameTask="+
                  request.getParameter("dashname
                  ")+"&patientid="+patientid+"'>
                     Dashboard    </a>&nbsp;&
                  nbsp;";

            }

          menu = menu +"<a href='listVersions?
               patientid="+patientid+"&formname
               ="+nameTask+"&idTask="+request.
               getParameter("idTask")+"&dashid
               ="+request.getParameter("dashid
               ")+"&dashname="+request.
               getParameter("dashname")+"' id='
               version' >   View Versions   </a
               >";
          }
          }
      patient=dentFormsSP.getPatient(
           patientid);
      mav.addObject("fieldValues", values);
      mav.addObject("userTask", userTaskVO);
      mav.addObject("html", html);
      mav.addObject("taskRSname", nameTask);
      mav.addObject("patientid",patientid);
      mav.addObject("patient",patient);
      mav.addObject("instanceid",request.
           getParameter("instanceid"));
    mav.addObject("idTask",request.
         getParameter("idTask"));
    mav.addObject("pos",request.
         getParameter("pos"));
    mav.addObject("menu",menu);
    mav.addObject("dashid",request.
         getParameter("dashid"));
    mav.addObject("dashname",request.
         getParameter("dashname"));
    if(notif!=null && counter==0){
 String notifs="Patient record update is
      successful";
 mav.addObject("notif", notifs);
}
    /*
START DATABASE SECURITY CHECKING
*/
dentFormsSP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());
System.out.println("checkcheckcheckcheck")
     ;
boolean privilegeCheck=false;
String errorMessage="";
String currentSessionRole="";
ArrayList<String> potentialOwnerList=
     dentFormsSP.checkaccess_section(
     Integer.parseInt(request.getParameter
     ("patientid")));
String[] currentSessionRoleList= (String
     []) session.getAttribute("
     currentDatabaseList");
for(int i=0; i<currentSessionRoleList.
     length;i++){
for(int j=0;j<potentialOwnerList.size();j
     ++){
System.out.println("PUMASOK BA? "+
     potentialOwnerList.get(j)+" " +
     currentSessionRoleList[i]);
if(currentSessionRoleList[i].equals(
     potentialOwnerList.get(j).trim()) ||
     potentialOwnerList.get(j).trim().
     equals(session.getAttribute("
     sessionUser").toString())){
System.out.println("PACHECK NAMAN : "+
     potentialOwnerList.get(j).trim());
privilegeCheck= true;
break;
}// end if equal condition
}//end of j loop
} //end of i loop
if(privilegeCheck){

return mav;
}
else{
 ModelAndView mav2=new ModelAndView("greet
      ");

return mav2;
}
/*
END DATABASE SECURITY CHECKING
*/

}
else{
 String message="<form action='greet'><br
      ><br><center><h3>Patient record
      successfully updated and sent for
      faculty approval</h3><br><input type
      ='submit' value' OK '></center>";
 mav.addObject("message", message);
}


return mav; }

   public static void getForms() throws
        Exception{
//list of patients


 //int patientidint=Integer.parseInt(
      patientid);
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
  conn.setAutoCommit(false);

   try{
    String update="";

    update="SELECT * FROM i18ntext";

    System.out.println("update statement="+
         update);
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery(update);
   conn.commit();

   while(rs.next()){
          if(!rs.getString("text").equals("
               AssignToOralDiagnosisClinician
               ")){
          String url="task?idTask="+rs.
               getString("task_names_id")+"&
               nameTask="+rs.getString("text
               ");
          formlist.put(rs.getString("text")
               , url);
          }
     }

   System.out.println("Forms started");


   st.close();
   //rs.close();
   conn.close();
   }
   catch(Exception e){
    e.printStackTrace();
   }

}
   public void deletetasks() throws
        Exception{
 //list of patients


  //int patientidint=Integer.parseInt(
       patientid);
  Class.forName("org.postgresql.Driver").
       newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
   conn.setAutoCommit(false);

   try{
    String update="";

    update="SELECT * FROM task WHERE status
         ='Completed'";

    System.out.println("update statement="+
         update);
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery(update);
   conn.commit();
   DentFormsSP dentforms=new DentFormsSP();
   while(rs.next()){

        String taskid= rs.getString("id");
      dentforms.deleteTask(taskid);
     }

   System.out.println("Forms started");


   st.close();
   //rs.close();
   conn.close();
   }
   catch(Exception e){
    e.printStackTrace();
   }

}
   public String getstatus(String id)
        throws Exception{
 //list of patients


  //int patientidint=Integer.parseInt(
       patientid);
  Class.forName("org.postgresql.Driver").
       newInstance();
    Connection conn=DriverManager.
```

```
            getConnection("jdbc:postgresql://
            localhost:5432/DentISt","jgerona","
            bakitba?");
     conn.setAutoCommit(false);
     String status="";
     try{
      String update="";

      update="SELECT * FROM task WHERE id="+
            id;

      System.out.println("update statement="+
            update);
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery(update);
     conn.commit();
     DentFormsSP dentforms=new DentFormsSP();

     while(rs.next()){

      status = rs.getString("status");
      if(status.equalsIgnoreCase("Reserved"))
            {
       break;
      }

       }

     System.out.println("Forms started");


     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
      e.printStackTrace();
     }
     return status;
 }
     public void resetProcessLog() throws
          Exception{
  //list of patients


    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
            getConnection("jdbc:postgresql://
            localhost:5432/DentISt","jgerona","
            bakitba?");
     conn.setAutoCommit(false);

     try{
      String update="DELETE FROM
            processinstancelog";
      System.out.println("delete statement="+
            update);
      Statement st = conn.createStatement();
      boolean rs = st.execute(update);
      conn.commit();
      //System.out.println("i18ntext RS="+rs)
            ;


     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
      e.printStackTrace();
     }

 }
 public static void resetInstanceIds()
      throws Exception{
  //list of patients


    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
            getConnection("jdbc:postgresql://
            localhost:5432/DentISt","jgerona","
            bakitba?");
     conn.setAutoCommit(false);

     try{
     String update="UPDATE \"patient\" SET
          instanceid=0";
     System.out.println("update statement="+
          update);
     Statement st = conn.createStatement();
     int rs = st.executeUpdate(update);
     conn.commit();
     System.out.println("Your data has been
          updated into table. RS="+rs);
     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
      e.printStackTrace();
     }

 }
}
```

```
package org.dentist.version.three.web.
     controller;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;


public class UPCDIDGenerator {

 public String generate(String rolename){

  setDatabase_username(rolename);
  String latest_upcdid="";
  String upcdid="";
  try {

    DateFormat dateFormat = new
         SimpleDateFormat("yyyy/MM/dd");
       //get current date time with Date()
   Date date = new Date();

   String[] dateSplit=dateFormat.format(date
        ).toString().split("/");

   String year= dateSplit[0].substring(2);
   System.out.println("Year: "+year);

   latest_upcdid=getLastUPCDID().trim();
   System.out.println("LATEST: "+
        latest_upcdid);
   if(latest_upcdid!=null && !latest_upcdid.
         isEmpty()){

    String[] idsplit=latest_upcdid.split
         ("-");

    int currentNum= Integer.parseInt(idsplit
         [1]) +1;

    //format number
    upcdid=year+"-"+String.format("%05d",
         currentNum);
   }else{

    String currentNum="00001";

    upcdid=year+"-"+currentNum;

   }

  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

  return upcdid;
 }


 /*
  *
  * DATABASE PART
  *
  */


 private String database_username;
 private final String database_password="yes
      ?bakitpo?";

 protected void setDatabase_username(String
      database_username) {
  this.database_username = database_username
       ;
 }


 protected String getDatabase_username() {
  return database_username;
 }

 //list all roles
 public String getLastUPCDID() throws
      Exception{
  Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password)
          ;
    conn.setAutoCommit(false);
      CallableStatement calstat=conn.
          prepareCall("{call get_upcdid()}");

        ResultSet rs = calstat.executeQuery();

        String upcdid="";
        while(rs.next()){
          upcdid=rs.getString(1);
        //System.out.println(rs.getString(1));
            }


       conn.close();
       calstat.close();
       System.out.println("Successful call for
            listallroles function");
        return upcdid;
  }

}
```

```
package org.dentist.version.three.web.
     controller;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.Version;
import org.springframework.stereotype.
     Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;

@Controller
public class VersionController {
 @RequestMapping(value = "/listVersions",
      method = RequestMethod.GET)
 public String viewSpecificUsers(
      HttpServletRequest request,
          HttpServletResponse response, Model
                model,HttpSession session)
               throws Exception {

  session=request.getSession(false);

  String formname=request.getParameter("
       formname");
  String patientid=request.getParameter("
       patientid");

  ArrayList<Version> versionsLists = new
       ArrayList<Version>();

  if(formname!=null && patientid!=null){
  DentFormsSP dentSP= new DentFormsSP();

  System.out.println("CHECK:: "+formname);
  try {
    dentSP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());

    versionsLists= dentSP.listAllVersion(
         Integer.parseInt(patientid),
         formname);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM ERROR");
   return "Error";
  }
  }
  else{
   System.out.println("BABO:: ");

  }
  model.addAttribute("versionLists",
       versionsLists);
  model.addAttribute("patientid",patientid);
  model.addAttribute("instanceid",request.
       getParameter("instanceid"));
  model.addAttribute("formname",formname);
  model.addAttribute("idTask",request.
       getParameter("idTask"));
  model.addAttribute("dashid",request.
       getParameter("dashid"));
  model.addAttribute("dashname",request.
       getParameter("dashname"));

  return "listVersions";

 }

}
```

```
package org.dentist.version.three.web.
     controller;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.lang.StringUtils;
import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.db.
     DentalChartSP;
import org.dentist.version.three.form.
     CariesStatus;
import org.dentist.version.three.form.
     DentalChart;
import org.dentist.version.three.form.
     RecurrentStatus;
import org.dentist.version.three.form.
     RestorationStatus;
import org.dentist.version.three.form.
     ServiceNeeded;
import org.dentist.version.three.
     processserver.HumanTaskStartupServlet;
import org.dentist.version.three.
     processserver.model.DefinitionsRS;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.
```

```
      ProcessDefinitionInstancesRS;
import org.dentist.version.three.
      processserver.model.TaskRS;
import org.dentist.version.three.
      processserver.model.UserTaskVO;
import org.dentist.version.three.
      processserver.model.Version;
import org.dentist.version.three.
      processserver.service.Process;
import org.springframework.stereotype.
      Controller;
import org.springframework.web.bind.
      annotation.RequestMapping;
import org.springframework.web.bind.
      annotation.RequestMethod;
import org.springframework.web.bind.
      annotation.RequestParam;
import org.springframework.web.servlet.
      ModelAndView;

/**
FOR AUDIT TRAIL

*/
import com.dentist.version.three.db.AdminSP;
import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.Role;
import com.dentist.version.three.form.User;


/**
END AUDIT TRAIL

**/
import com.dentist.version.three.db.RoleSP;
@Controller
public class ViewController {
 public static HashMap<String,String>
      formlist=new HashMap<String,String>();;

     @RequestMapping(value="viewForm",method
          = RequestMethod.GET)
//   public
//   @ModelAttribute("message")
   public ModelAndView getProcess(HttpSession
         session,HttpServletRequest request,
        HttpServletResponse response )
 throws Exception {
      if(formlist.isEmpty() && TaskController
            .formlist.isEmpty() &&
            SearchController.formlist.isEmpty()
            ){
       resetProcessLog();
    resetInstanceIds();

        DefinitionsRS defnitionRS = new
             DefinitionsRS();
           defnitionRS.setId("UPCDDentISt.
                Forms");
           Map<String, Object> params = new
                HashMap<String, Object>();
           Process.instance().startInstace(
                defnitionRS,session);
           getForms();
           formlist.remove("
                AssignToOralDiagnosisClinician
                ");

           ArrayList<User> userLists = new
                ArrayList<User>();
           ArrayList<Role> roleLists = new
                ArrayList<Role>();
           UserSP userSP= new UserSP();
           RoleSP roleSP= new RoleSP();

           boolean privilegeCheck=false;
           AdminSP adminSP= new AdminSP();
           String currentRole="";
           String[] currentRoleList= (String
                []) session.getAttribute("
                currentDatabaseList");
    for(int i=0; i<currentRoleList.length;i
         ++){
     privilegeCheck= adminSP.checkPrivilege(
          currentRoleList[i], "users", "select
          ");
     if(privilegeCheck){
      currentRole=currentRoleList[i];
      break;
     }
    }
    System.out.println(currentRole+"-"+
       privilegeCheck);
  userSP.setDatabase_username(currentRole);

 userLists= userSP.allUserList();
 roleLists= roleSP.allRoleList();
 try{
  for (User user : userLists) {
   /** start add new user to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addUser(new org.jbpm.task.User(user.
        getUsername()));
   /**end add new user to jbpm**/
      System.out.println("Add user: "+ user
           .getUsername());
     }
  for (Role role : roleLists) {
   /** start add groups to jbpm**/
   HumanTaskStartupServlet.taskSession.
        addGroup(new org.jbpm.task.Group(
        roleSP.getDatabaseRole(role.
        getRole_id()).trim()));
   /**end add new groups to jbpm**/
      System.out.println("Add group: "+
           roleSP.getDatabaseRole(role.
           getRole_id()).trim());
     }
  }
  catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM ERROR");
  }
  if(formlist.isEmpty() && TaskController.
       formlist.isEmpty() &&
       SearchController.formlist.isEmpty()){
   response.sendRedirect("viewForm?
        patientid="+request.getParameter("
        patientid"));
  }
    }
    HashMap<String,String> map=formlist; //
         FOR APPROVE STATUS
    if(map.isEmpty()){
     map=SearchController.formlist;
     if(map.isEmpty()){
      map=TaskController.formlist;
     }
    }
    String idTask=request.getParameter("
         idTask");
    if(idTask==null){
     idTask=map.get("PatientInformation");
     idTask=StringUtils.substringBetween(
          idTask,"idTask=","&nameTask");
    }
    String nameTask=request.getParameter("
         nameTask");
    if(nameTask==null){
     nameTask="PatientInformation";
    }
      ModelAndView mav =new ModelAndView("
           viewForm");
      //mav.addObject("message", resp);
      UserTaskVO userTaskVO = new
           UserTaskVO("krisv", "krisv");
TaskRS taskRS = new TaskRS();
taskRS.setId(idTask);
userTaskVO.setTaskRS(taskRS);
//taskRS = Process.instance().getUserTask(
     userTaskVO);
//userTaskVO.setTaskRS(taskRS);
String html = Process.instance().
     getTaskRenderHTML(userTaskVO,session);

    mav.addObject("userTask", userTaskVO);
    mav.addObject("html", html);



    //GET FROM DB
    AdminSP adminSP= new AdminSP();
    String formname=nameTask;
      String values="";
      String tablename=formname;
      DentFormsSP dentFormsSP= new
           DentFormsSP();
      int patientid=Integer.parseInt(
           request.getParameter("patientid")
           );
      String version=request.getParameter("
           version");
      if(formname.equals("DentalChart")){
```

```
    int patient_id=patientid;

    boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";


  ArrayList<Integer> listpatientids= new
       ArrayList<Integer>();
  boolean patient_exist=false;
  DentalChartSP dentalChartSP= new
       DentalChartSP();
  DentalChart dentalChart= new
       DentalChart();
  CariesStatus cariesStatus= new
       CariesStatus();
  RecurrentStatus recurrentStatus= new
       RecurrentStatus();
  RestorationStatus restorationStatus=
       new RestorationStatus();
  ServiceNeeded services= new
       ServiceNeeded();
  try {


            dentalChartSP.
                 setDatabase_username(
                 session.getAttribute("
                 sessionUserRole").
                 toString());
   listpatientids= dentalChartSP.
        getpatientIDList();
   if(!listpatientids.isEmpty() ||
        listpatientids!=null){
   for(Integer patientidss :
        listpatientids){
    if(patientidss==patient_id)
     patient_exist=true;
   }
  }
  if(patient_exist){
   int ver;
      if(version!=null){
       ver=Integer.parseInt(version);
      }
      else{
       ver=dentFormsSP.getCurrentVersion
            (patientid,"dentalchart");
      }
   System.out.println("Patient exists");
   dentalChart=dentalChartSP.
        getDentalChart(patient_id, ver);
   cariesStatus=dentalChartSP.
        getCariesStatus(patient_id,ver);
   recurrentStatus=dentalChartSP.
        getRecurrentStatus(patient_id,ver
        );
   restorationStatus=dentalChartSP.
        getRestorationStatus(patient_id,
        ver);
   services=dentalChartSP.
        getServiceNeeded(patient_id,ver);

   System.out.println("Dental Chart: "+
        dentalChart.getDental_chart_id()
        ;
   System.out.println("CariesStatus
        Chart: "+cariesStatus.
        getCaries_id());
   System.out.println("RecurrentStatus
        Chart: "+recurrentStatus.
        getrecurrent_id());
   System.out.println("RestorationStatus
         Chart: "+restorationStatus.
        getrestoration_id());
   System.out.println("Service Needed:
        "+ services.getServiceneeded_id()
        );

   /**
START OF AUDIT TRAIL
**/

       adminSP.setDatabase_username(
            session.getAttribute("
            sessionUserRole").toString())
            ;
 UserSP userSP= new UserSP();
       userSP.setDatabase_username(
            session.getAttribute("
            sessionUserRole").toString())
            ;
//date time
 DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
 Date date = new Date();
 System.out.println(dateFormat.format(
      date));
 String dateString= dateFormat.format(
      date).toString();
//end for date time

//audittrail
 int sessionUserID= Integer.parseInt(
      session.getAttribute("sessionUserId
      ").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
    Patient patient= dentFormsSP.
         getPatient(patientid);
 /*
  getname for patient?
 */

 String action_performed=patient.getName
      ();//patientname
 String action_encounter="SELECT";


 String sessionName= sessionUser.
      getFname_user()+" "+ sessionUser.
      getMinit_user() + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(sessionName,
      action_encounter, action_performed,
       "dentalchart",dateString);
//end of audittrail
/**
 END OF AUDITTRAIL

**/
       mav.addObject("dentalChart",
            dentalChart);
       mav.addObject("cariesStatus",
            cariesStatus);
       mav.addObject("recurrentStatus",
            recurrentStatus);
       mav.addObject("restorationStatus",
             restorationStatus);
       mav.addObject("restorationStatus",
             restorationStatus);
       mav.addObject("services", services
            );
       mav.addObject("is_current",
            dentalChart.getIs_current());
       mav.addObject("version",
            dentalChart.getVersion());

      }
      else{
       mav.addObject("version",1);
       mav.addObject("is_current", "yes")
            ;

      }

      } catch (Exception e) {
       // TODO Auto-generated catch block

       e.printStackTrace();
      }

    }
    else{
    try {


     if(version!=null){
    values= dentFormsSP.
         getVersionFormFields(patientid,
         tablename,Integer.parseInt(
         version));

    if(html.contains("tablelist")){
     ArrayList<Version> tableLists = new
          ArrayList<Version>();
     tableLists= dentFormsSP.
          listAllVersion(patientid,
          formname);
     ArrayList<String> valueLists = new
          ArrayList<String>();
     String vals="";
    mav.addObject("tableLists",
         tableLists);
    for (Version ver : tableLists) {
     vals= dentFormsSP.
          getVersionFormFields(patientid,
          tablename,Integer.parseInt(ver.
          getVersion()));
     if(ver.getApproved().
          equalsIgnoreCase("Approved") ||
           ver.getApproved().
          equalsIgnoreCase("For Approval
          ")){
      if(!nameTask.equalsIgnoreCase("
           ConsultationsAndFindings")){
      valueLists.add(vals);
      }
      else{
       int count=0;
       String findings="";
       String tokenizevals=vals;
       StringTokenizer st = new
            StringTokenizer(tokenizevals,
             "|");
       String date="";
       while(st.hasMoreTokens() && count
            <4) {
            findings=st.nextToken();

            System.out.println(findings +
                 "-column added -" +
                 count);
            count++;
           }
       count=0;
       st = new StringTokenizer(
            tokenizevals, "|");

       while(st.hasMoreTokens() && count
            <9) {
            date=st.nextToken();

            System.out.println(findings +
                 "-column added -" +
                 count);
            count++;
           }
       System.out.println(findings + "-
            eto na column added -"+ count
            );
       if(!findings.trim().equals("")){
        valueLists.add(date+" | "+vals);
       }
      }
     }
    }
    int count = StringUtils.countMatches
         (vals, "|");
    count=count-8;
    mav.addObject("colCount", count);
    mav.addObject("valueLists",
         valueLists);
    //values= dentFormsSP.getFormFields(
         patientid,tablename);
   }
  }
  else if(html.contains("tablelist")){
   ArrayList<Version> tableLists = new
        ArrayList<Version>();
   tableLists= dentFormsSP.listAllVersion
        (patientid, formname);
   ArrayList<String> valueLists = new
        ArrayList<String>();
   String vals="";
   mav.addObject("tableLists", tableLists
        );
   for (Version ver : tableLists) {
    vals= dentFormsSP.
         getVersionFormFields(patientid,
         tablename,Integer.parseInt(ver.
         getVersion()));
    if(ver.getApproved().equalsIgnoreCase
         ("Approved") || ver.getApproved()
         .equalsIgnoreCase("For Approval")
         ){
     if(!nameTask.equalsIgnoreCase("
          ConsultationsAndFindings")){
     valueLists.add(vals);
     }
     else{
      int count=0;
      String findings="";
      String tokenizevals=vals;
      StringTokenizer st = new
           StringTokenizer(tokenizevals,
           "|");
      String date="";
      while(st.hasMoreTokens() && count
           <4) {
           findings=st.nextToken();

           System.out.println(findings +
                "-column added -" + count)
                ;
           count++;
          }
      count=0;
      st = new StringTokenizer(
           tokenizevals, "|");

      while(st.hasMoreTokens() && count
           <9) {
           date=st.nextToken();

           System.out.println(findings +
                "-column added -" + count)
                ;
           count++;
          }
      System.out.println(findings + "-eto
            na column added -"+ count);
      if(!findings.trim().equals("")){
       valueLists.add(date+" | "+vals);
      }
     }
    }
   }
   int count = StringUtils.countMatches(
        vals, "|");
   count=count-8;
   mav.addObject("colCount", count);
   mav.addObject("valueLists", valueLists
        );
   values= dentFormsSP.getFormFields(
        patientid,tablename);
  }
     else{
   values= dentFormsSP.getFormFields(
        patientid,tablename);
     }
     /**
     START OF AUDIT TRAIL
     **/

     adminSP.setDatabase_username(session
          .getAttribute("sessionUserRole")
          .toString());
     UserSP userSP= new UserSP();
     userSP.setDatabase_username(session.
          getAttribute("sessionUserRole").
          toString());

     //date time
     DateFormat dateFormat = new
          SimpleDateFormat("dd/MM/yyyy");
     Date date = new Date();
     System.out.println(dateFormat.format
          (date));
     String dateString= dateFormat.format
          (date).toString();
     //end for date time

     //audittrail
     int sessionUserID= Integer.parseInt(
          session.getAttribute("
          sessionUserId").toString());
     User sessionUser= userSP.getUser(
          sessionUserID);
     Patient patient= dentFormsSP.
          getPatient(patientid);
     /*
     getname for patient?
     */

     String action_performed=patient.
          getName();//patientname
     String action_encounter="SELECT";

     String sessionName= sessionUser.
          getFname_user()+" "+ sessionUser
          .getMinit_user() + " "+
          sessionUser.getLname_user()+"
          ("+ sessionUser.getUsername()+")
          ";
     adminSP.insertAuditTrail(sessionName
            ,action_encounter ,
            action_performed , tablename ,
            dateString);
        System.out.println("INSERTED AUDIT
            TRAIL");
        //end of audittrail
        /**
        END OF AUDITTRAIL
        **/
    } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
      }

    Patient patient=dentFormsSP.getPatient(
        patientid);
    mav.addObject("fieldValues", values);
    mav.addObject("taskRSname", formname);
    mav.addObject("patientid",request.
        getParameter("patientid"));
    mav.addObject("patient",patient);
    mav.addObject("instanceid",request.
        getParameter("instanceid"));
    mav.addObject("pos",request.
        getParameter("pos"));
    mav.addObject("flag", request.
        getParameter("flag"));
    mav.addObject("formlist",map);
    mav.addObject("idTask",idTask);
      return mav;
  }
  public static String substringAfter(String
       str, String separator) {

      if (separator == null) {
          return "";
      }
      int pos = str.indexOf(separator);
      if (pos == -1) {
          return "";
      }
      return str.substring(pos + separator.
          length());
  }
  public static void getForms() throws
      Exception{
  //list of patients


    //int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="";

     update="SELECT * FROM i18ntext";

     System.out.println("update statement="+
         update);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(update);
    conn.commit();

    while(rs.next()){
           if(!rs.getString("text").equals("
               AssignToOralDiagnosisClinician
               ")){
           String url="task?idTask="+rs.
               getString("task_names_id")+"&
               nameTask="+rs.getString("text
               ");
           formlist.put(rs.getString("text")
               , url);
           }
      }

    System.out.println("Forms started");


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
  public void resetProcessLog() throws
      Exception{
  //list of patients


   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
    conn.setAutoCommit(false);

    try{
     String update="DELETE FROM
         processinstancelog";
     System.out.println("delete statement="+
         update);
     Statement st = conn.createStatement();
     boolean rs = st.execute(update);
     conn.commit();
     //System.out.println("i18ntext RS="+rs)
         ;


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
 public static void resetInstanceIds()
     throws Exception{
  //list of patients


   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
    conn.setAutoCommit(false);

    try{
    String update="UPDATE \"patient\" SET
        instanceid=0";
    System.out.println("update statement="+
        update);
    Statement st = conn.createStatement();
    int rs = st.executeUpdate(update);
    conn.commit();
    System.out.println("Your data has been
        updated into table. RS="+rs);
    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

 }
}
```

```
package org.dentist.version.three.web.
    controller;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;

import javax.servlet.http.HttpServletRequest
    ;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.lang.StringUtils;
import org.dentist.version.three.db.
```

```
      DentFormsSP;
import org.dentist.version.three.db.
      DentalChartSP;
import org.dentist.version.three.form.
      CariesStatus;
import org.dentist.version.three.form.
      DentalChart;
import org.dentist.version.three.form.
      RecurrentStatus;
import org.dentist.version.three.form.
      RestorationStatus;
import org.dentist.version.three.form.
      ServiceNeeded;
import org.dentist.version.three.
      processserver.model.DefinitionsRS;
import org.dentist.version.three.
      processserver.model.Patient;
import org.dentist.version.three.
      processserver.model.
      ProcessDefinitionInstancesRS;
import org.dentist.version.three.
      processserver.model.TaskRS;
import org.dentist.version.three.
      processserver.model.UserTaskVO;
import org.dentist.version.three.
      processserver.model.Version;
import org.dentist.version.three.
      processserver.service.Process;
import org.springframework.stereotype.
      Controller;
import org.springframework.web.bind.
      annotation.RequestMapping;
import org.springframework.web.bind.
      annotation.RequestMethod;
import org.springframework.web.bind.
      annotation.RequestParam;
import org.springframework.web.servlet.
      ModelAndView;

import com.dentist.version.three.db.RoleSP;
@Controller
public class ViewRecordController {
     @RequestMapping(value="viewRec",method =
           RequestMethod.GET)
//   public
//   @ModelAttribute("message")
   public ModelAndView getProcess(HttpSession
         session,HttpServletRequest request,
        @RequestParam("idTask") String idTask,
          @RequestParam("nameTask") String
        nameTask)
 throws Exception {
      if(TaskController.formlist.isEmpty() &&
            SearchController.formlist.isEmpty
           () && ViewController.formlist.
           isEmpty()){
       DefinitionsRS defnitionRS = new
            DefinitionsRS();
           defnitionRS.setId("UPCDDentISt.
                Forms");
           Map<String, Object> params = new
                HashMap<String, Object>();
           String resp = Process.instance().
                processComplete(defnitionRS,
                params);
           TaskController.getForms();

      }
         ModelAndView mav =new ModelAndView("
              viewRec");
         //mav.addObject("message", resp);
         UserTaskVO userTaskVO = new
              UserTaskVO("krisv", "krisv");
   TaskRS taskRS = new TaskRS();
   taskRS.setId(idTask);
   userTaskVO.setTaskRS(taskRS);
   //taskRS = Process.instance().getUserTask(
        userTaskVO);
   //userTaskVO.setTaskRS(taskRS);
   String html = Process.instance().
        getTaskRenderHTML(userTaskVO,session);

      mav.addObject("userTask", userTaskVO);
      mav.addObject("html", html);

      //GET FROM DB
      String formname=nameTask;
        String values="";
        String tablename=formname;
        DentFormsSP dentFormsSP= new
             DentFormsSP();
        int patientid=Integer.parseInt(
        request.getParameter("patientid")
        );
  String version=request.getParameter("
        version");
  if(formname.equals("DentalChart")){
   int patient_id=patientid;

   boolean privilegeCheck=false;
String errorMessage="";
String currentRole="";


ArrayList<Integer> listpatientids= new
      ArrayList<Integer>();
boolean patient_exist=false;
DentalChartSP dentalChartSP= new
      DentalChartSP();
DentalChart dentalChart= new
      DentalChart();
CariesStatus cariesStatus= new
      CariesStatus();
RecurrentStatus recurrentStatus= new
      RecurrentStatus();
RestorationStatus restorationStatus=
      new RestorationStatus();
ServiceNeeded services= new
      ServiceNeeded();
try {


  dentalChartSP.setDatabase_username(
        session.getAttribute("
        sessionUserRole").toString());
  listpatientids= dentalChartSP.
        getpatientIDList();
  if(!listpatientids.isEmpty() ||
        listpatientids!=null){
  for(Integer patientidss :
        listpatientids){
   if(patientidss==patient_id)
    patient_exist=true;
  }
 }
if(patient_exist){
  int ver;
     if(version!=null){
      ver=Integer.parseInt(version);
     }
     else{
      ver=dentFormsSP.getCurrentVersion
           (patientid,"dentalchart");
     }
  System.out.println("Patient exists");
  dentalChart=dentalChartSP.
        getDentalChart(patient_id, ver);
  cariesStatus=dentalChartSP.
        getCariesStatus(patient_id,ver);
  recurrentStatus=dentalChartSP.
        getRecurrentStatus(patient_id,ver
        );
  restorationStatus=dentalChartSP.
        getRestorationStatus(patient_id,
        ver);
  services=dentalChartSP.
        getServiceNeeded(patient_id,ver);

  System.out.println("Dental Chart: "+
        dentalChart.getDental_chart_id())
        ;
  System.out.println("CariesStatus
        Chart: "+cariesStatus.
        getCaries_id());
  System.out.println("RecurrentStatus
        Chart: "+recurrentStatus.
        getrecurrent_id());
  System.out.println("RestorationStatus
         Chart: "+restorationStatus.
        getrestoration_id());
  System.out.println("Service Needed:
        "+ services.getServiceneeded_id()
        );

     mav.addObject("dentalChart",
          dentalChart);
     mav.addObject("cariesStatus",
          cariesStatus);
     mav.addObject("recurrentStatus",
          recurrentStatus);
     mav.addObject("restorationStatus",
          restorationStatus);
     mav.addObject("restorationStatus",
          restorationStatus);
    mav.addObject("services", services
         );
    mav.addObject("is_current",
         dentalChart.getIs_current());
    mav.addObject("version",
         dentalChart.getVersion());

   }
   else{
    mav.addObject("version",1);
    mav.addObject("is_current", "yes")
         ;

   }

   } catch (Exception e) {
    // TODO Auto-generated catch block

    e.printStackTrace();
   }

 }
 else{
 try {
  /*

          if(version!=null){
   values= dentFormsSP.
        getVersionFormFields(patientid,
        tablename,Integer.parseInt(
        version));
  }
  else if(html.contains("tablelist")){
 ArrayList<Version> tableLists = new
      ArrayList<Version>();
 tableLists= dentFormsSP.
      listAllVersion(patientid,
      formname);
 ArrayList<String> valueLists = new
      ArrayList<String>();
 String vals="";
 mav.addObject("tableLists",
      tableLists);
 for (Version ver : tableLists) {
  vals= dentFormsSP.
       getVersionFormFields(patientid,
       tablename,Integer.parseInt(ver.
       getVersion()));
  valueLists.add(vals);
 }
 int count = StringUtils.countMatches(
      vals, "|");
 count=count-8;
 mav.addObject("colCount", count);
 mav.addObject("valueLists",
      valueLists);
 values="";
}

  */
  if(version!=null){
 values= dentFormsSP.
      getVersionFormFields(patientid,
      tablename,Integer.parseInt(
      version));

 if(html.contains("tablelist")){
  ArrayList<Version> tableLists = new
       ArrayList<Version>();
  tableLists= dentFormsSP.
       listAllVersion(patientid,
       formname);
  ArrayList<String> valueLists = new
       ArrayList<String>();
  String vals="";
  mav.addObject("tableLists",
       tableLists);
  for (Version ver : tableLists) {
   vals= dentFormsSP.
        getVersionFormFields(patientid,
        tablename,Integer.parseInt(ver.
        getVersion()));
   if(ver.getApproved().
        equalsIgnoreCase("Approved") ||
         ver.getApproved().
        equalsIgnoreCase("For Approval
        ")){
    if(!nameTask.equalsIgnoreCase("
         ConsultationsAndFindings")){
    valueLists.add(vals);
    }
    else{
     int count=0;
     String findings="";
     String tokenizevals=vals;
     StringTokenizer st = new
          StringTokenizer(tokenizevals,
          "|");
     String date="";
     while(st.hasMoreTokens() && count
          <4) {
         findings=st.nextToken();

         System.out.println(findings +
              "-column added -" +
             count);
         count++;
        }
     count=0;
     st = new StringTokenizer(
          tokenizevals, "|");

     while(st.hasMoreTokens() && count
          <9) {
         date=st.nextToken();

         System.out.println(findings +
              "-column added -" +
             count);
         count++;
        }
     System.out.println(findings + "-
          eto na column added -"+ count
          );
     if(!findings.trim().equals("")){
      valueLists.add(date+" | "+vals);
     }
    }
   }
  }
  int count = StringUtils.countMatches
       (vals, "|");
  count=count-8;
  mav.addObject("colCount", count);
  mav.addObject("valueLists",
       valueLists);
  //values= dentFormsSP.getFormFields(
       patientid,tablename);
 }
}
else if(html.contains("tablelist")){
 ArrayList<Version> tableLists = new
      ArrayList<Version>();
 tableLists= dentFormsSP.listAllVersion
      (patientid, formname);
 ArrayList<String> valueLists = new
      ArrayList<String>();
 String vals="";
 mav.addObject("tableLists", tableLists
      );
 for (Version ver : tableLists) {
  vals= dentFormsSP.
       getVersionFormFields(patientid,
       tablename,Integer.parseInt(ver.
       getVersion()));
  if(ver.getApproved().equalsIgnoreCase
       ("Approved") || ver.getApproved()
       .equalsIgnoreCase("For Approval")
       ){
   if(!nameTask.equalsIgnoreCase("
        ConsultationsAndFindings")){
   valueLists.add(vals);
   }
   else{
    int count=0;
    String findings="";
    String tokenizevals=vals;
    StringTokenizer st = new
         StringTokenizer(tokenizevals,
         "|");
    String date="";
    while(st.hasMoreTokens() && count
         <4) {
        findings=st.nextToken();

        System.out.println(findings +
             "-column added -" + count)
             ;
        count++;
       }
    count=0;
    st = new StringTokenizer(
         tokenizevals, "|");
```

```
            while(st.hasMoreTokens() && count
                 <9) {
                date=st.nextToken();

                System.out.println(findings +
                     "-column added -" +  count)
                     ;
                count++;
               }
            System.out.println(findings + "-eto
                 na column added -"+ count);
            if(!findings.trim().equals("")){
             valueLists.add(date+" | "+vals);
            }
          }
         }
        }
        int count = StringUtils.countMatches(
             vals, "|");
        count=count-8;
        mav.addObject("colCount", count);
        mav.addObject("valueLists", valueLists
             );
        values= dentFormsSP.getFormFields(
             patientid,tablename);
       }
          else{
        values= dentFormsSP.getFormFields(
             patientid,tablename);
          }
       } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
       }
         }

    Patient patient=dentFormsSP.getPatient(
         patientid);
    mav.addObject("fieldValues", values);
    mav.addObject("taskRSname", formname);
    mav.addObject("patientid",request.
         getParameter("patientid"));
    mav.addObject("patient",patient);
    mav.addObject("instanceid",request.
         getParameter("instanceid"));
    mav.addObject("pos",request.
         getParameter("pos"));
    mav.addObject("flag", request.
         getParameter("flag"));
    mav.addObject("formlist",TaskController
         .formlist);
    mav.addObject("idTask",request.
         getParameter("idTask"));
      return mav;
  }
  public static String substringAfter(String
        str, String separator) {

      if (separator == null) {
          return "";
      }
      int pos = str.indexOf(separator);
      if (pos == -1) {
          return "";
      }
      return str.substring(pos + separator.
           length());
  }

}
```

```
package org.dentist.version.three.web.
     controller;

import java.util.ArrayList;
import java.util.HashMap;

import javax.servlet.http.HttpServletRequest
     ;
import javax.servlet.http.
     HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.dentist.version.three.db.
     DentFormsSP;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.Version;
import org.springframework.stereotype.
     Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;

@Controller
public class ViewVersionController {
 @RequestMapping(value = "/listViewVersions
      ", method = RequestMethod.GET)
 public String viewSpecificUsers(
      HttpServletRequest request,
          HttpServletResponse response, Model
               model,HttpSession session)
               throws Exception {

  session=request.getSession(false);

  String formname=request.getParameter("
       formname");
  String patientid=request.getParameter("
       patientid");

  ArrayList<Version> versionsLists = new
       ArrayList<Version>();

  if(formname!=null && patientid!=null){
  DentFormsSP dentSP= new DentFormsSP();

  System.out.println("CHECK:: "+formname);
  try {
    dentSP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());

    versionsLists= dentSP.listAllVersion(
         Integer.parseInt(patientid),
         formname);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM ERROR");
   return "Error";
  }
  }
  else{
   System.out.println("BABO:: ");

  }
  HashMap<String,String> map=TaskController.
       formlist; //FOR APPROVE STATUS
     if(map.isEmpty()){
      map=SearchController.formlist;
      if(map.isEmpty()){
       map=ViewController.formlist;
      }
     }
  DentFormsSP dentFormsSP= new DentFormsSP()
       ;
  Patient patient=dentFormsSP.getPatient(
       Integer.parseInt(patientid));
  model.addAttribute("versionLists",
       versionsLists);
  model.addAttribute("patientid",patientid);
  model.addAttribute("instanceid",request.
       getParameter("instanceid"));
  model.addAttribute("formname",formname);
  model.addAttribute("idTask",request.
       getParameter("idTask"));
  model.addAttribute("dashid",request.
       getParameter("dashid"));
  model.addAttribute("dashname",request.
       getParameter("dashname"));
  model.addAttribute("formlist",map);
  model.addAttribute("patientname",patient.
       getName());


  return "listViewVersions";

 }

}
```

```
package org.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.StringTokenizer;

import javax.servlet.http.HttpSession;

import org.apache.commons.lang.StringUtils;
import org.dentist.version.three.
     processserver.model.Appointment;
import org.dentist.version.three.
     processserver.model.Patient;
import org.dentist.version.three.
     processserver.model.Update;
import org.dentist.version.three.
     processserver.model.Version;

import com.dentist.version.three.form.Role;
import com.dentist.version.three.form.
     Section;
import com.dentist.version.three.form.User;
import com.dentist.version.three.form.
     UserRoleSection;
import com.dentist.version.three.mapper.
     RoleMapper;
import com.dentist.version.three.mapper.
     SectionMapper;
import com.dentist.version.three.mapper.
     UserMapper;

public class DentFormsSP {

 private String database_username;
 private final String database_password="yes
      ?bakitpo?";

 public void setDatabase_username(String
      database_username) {
  this.database_username = database_username
       ;
 }


 public String getDatabase_username() {
  return database_username;
 }

 public ArrayList<Patient>
      specifiedPatientList(String
      specifiedSearch) throws Exception{
  //list of patients

    ArrayList<Patient> listPatients= new
         ArrayList<Patient>();
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);
     CallableStatement calstat=conn.
          prepareCall("{call
          listspecifiedpatients(?)}");
       calstat.setString(1, specifiedSearch)
            ;

       ResultSet rs = calstat.executeQuery()
            ;

       while(rs.next()){
            Patient patient = new Patient();
            patient.setId(rs.getString(1));
            patient.setName(rs.getString(2)+"
                  "+rs.getString(3)+" "+rs.
                 getString(4));
            patient.setupcdId(rs.getString(5)
                 );
            patient.setinstanceid(rs.
                 getString(16));
            listPatients.add(patient);
      System.out.println(patient.getId()+"
           Patient Name: "+patient.getName()
           +"Patient UPCDId: "+patient.
           getupcdId());
        }
     conn.close();
     calstat.close();
     System.out.println("Successful call for
           listspecifiedpatients function");
       return listPatients;
}
public Patient specifiedPatient(String
     givenname,String familyname,String
     birthdate,String gender) throws
     Exception{
 //list of patients


   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);
     CallableStatement calstat=conn.
          prepareCall("{call
          getspecifiedpatient(?,?,?,?)}");
       calstat.setString(1, givenname);
       calstat.setString(2, familyname);
     calstat.setString(3, birthdate);
     calstat.setString(4, gender);

       ResultSet rs = calstat.executeQuery()
            ;
     Patient patient = null;
       if(rs.next()){
      patient = new Patient();
           patient.setId(rs.getString(1));
           patient.setName(rs.getString(2)+"
                 "+rs.getString(3)+" "+rs.
                getString(4));
           patient.setupcdId(rs.getString(5)
                );
           patient.setinstanceid(rs.
                getString(16));

      System.out.println(patient.getId()+"
           Patient Name: "+patient.getName()
           +"Patient UPCDId: "+patient.
           getupcdId());
        }
     conn.close();
     calstat.close();
     System.out.println("Successful call for
           listspecifiedpatients function");
       return patient;
}

public Patient getPatientID() throws
     Exception{
 //list of patients


   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);
     CallableStatement calstat=conn.
          prepareCall("{call get_patientid()
          }");


       ResultSet rs = calstat.executeQuery()
            ;
      Patient patient = new Patient();
       while(rs.next()){

           patient.setId(rs.getString(1));
           patient.setName(rs.getString(2)+"
                 "+rs.getString(3)+" "+rs.
                getString(4));
           patient.setupcdId(rs.getString(5)
                );

      System.out.println(patient.getId()+"
           Patient Name: "+patient.getName()
```

```
            +"Patient UPCDId: "+patient.
            getupcdId());
        }
      conn.close();
      calstat.close();
      System.out.println("Successful call for
           listspecifiedpatients function");
        return patient;
}
public int getInstanceID() throws Exception
     {
 //list of patients
 int instanceid=0;

  Class.forName("org.postgresql.Driver").
       newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt",
           database_username,database_password
           );
    conn.setAutoCommit(false);
      CallableStatement calstat=conn.
           prepareCall("{call get_instanceid()
           }");


        ResultSet rs = calstat.executeQuery()
             ;

        while(rs.next()){

       instanceid=rs.getInt(2);


       System.out.println();
        }
      conn.close();
      calstat.close();
      System.out.println("Successful call for
           listspecifiedpatients function");
        return instanceid;
}
public String getProcessId(String
     instanceid) throws Exception{
 //list of patients
 String processid="";

 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt","jgerona","
         bakitba?");
  conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call get_processid(?)
         }");
    int id=Integer.parseInt(instanceid);
    calstat.setInt(1,id);
    ResultSet rs = calstat.executeQuery();

    while(rs.next()){
     processid=rs.getString(4);



  System.out.println(processid);
    }
 conn.close();
 calstat.close();
 System.out.println("Successful call for
      getprocessid function="+processid);
    return processid;

}
public void updatePatientStatus(String
     patientid, String status) throws
     Exception{
 //list of patients

  int patientidint=Integer.parseInt(
       patientid);
  Class.forName("org.postgresql.Driver").
       newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt",
           database_username,database_password
           );
    conn.setAutoCommit(false);
     try{
     String update="UPDATE \"patient\" SET
          status='"+status+"' WHERE patientid
          ="+Integer.toString(patientidint);
     System.out.println("update statement="+
          update);
     Statement st = conn.createStatement();
     int rs = st.executeUpdate(update);
     conn.commit();
     System.out.println("Your data has been
          updated into table. RS="+rs);
     st.close();
     //rs.close();
     conn.close();
     }
     catch(Exception e){
      e.printStackTrace();
     }

}
/**START OF FOR APPROVE STATUS*/
public void updateApprovedStat(String
     patientid, String table, String name)
     throws Exception{
 //list of patients

   int patientidint=Integer.parseInt(
        patientid);
   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
     conn.setAutoCommit(false);
     int version=getCurrentVersion(
          patientidint,table.toLowerCase());
     try{
      DateFormat dateFormat = new
           SimpleDateFormat("dd/MM/yyyy");
         //get current date time with Date()
         Date date = new Date();
         System.out.println(dateFormat.format
              (date));
      String update="";
      if(table.equals("DentalChart")){
      update="UPDATE dentalchart SET approved
           ='Approved', approved_by='"+name
           +"',approved_date='"+dateFormat.
           format(date)+"' WHERE patient_id="+
           Integer.toString(patientidint)+"
           AND version="+Integer.toString(
           version)+" AND approved!='Approved
           '";
      }
      else if(table.equalsIgnoreCase("
           ServicesRendered")){
       update="UPDATE servicesrendered SET
            approved='Approved', approved_by
            ='"+name+"', faculty='"+name+"',
            approved_date='"+dateFormat.format
            (date)+"'  WHERE patientid="+
            Integer.toString(patientidint)+"
            AND version="+Integer.toString(
            version)+" AND approved!='Approved
            '";
      }
      else{
       update="UPDATE \""+table.toLowerCase()
            +"\" SET approved='Approved',
            approved_by='"+name+"',
            approved_date='"+dateFormat.format
            (date)+"'  WHERE patientid="+
            Integer.toString(patientidint)+"
            AND version="+Integer.toString(
            version)+" AND approved!='Approved
            '";

      }System.out.println("update statement
           ="+update);
     Statement st = conn.createStatement();
     int rs = st.executeUpdate(update);
     conn.commit();
     System.out.println("Your data has been
          updated into table. RS="+rs);





     st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

}

public void updateApproved(String patientid
     , String table, String status,int
     version) throws Exception{
 //list of patients

   int patientidint=Integer.parseInt(
       patientid);
  Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);

    try{
     String update="";
     if(table.equals("DentalChart")){
     update="UPDATE dentalchart SET approved
          ='"+status+"' WHERE patient_id="+
          Integer.toString(patientidint)+"
          AND version="+Integer.toString(
          version)+" AND approved='For
          Approval'";
     }
     else{
      update="UPDATE \""+table.toLowerCase()
           +"\" SET approved='"+status+"'
           WHERE patientid="+Integer.toString
           (patientidint)+" AND version="+
           Integer.toString(version)+" AND
           approved='For Approval'";

     }System.out.println("update statement
          ="+update);
    Statement st = conn.createStatement();
    int rs = st.executeUpdate(update);
    conn.commit();
    System.out.println("Your data has been
         updated into table. RS="+rs);




    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

}
/**END OF FOR APPROVE STATUS*/
public void updateInstanceID(String
     patientid, int instanceid) throws
     Exception{
 //list of patients

   int patientidint=Integer.parseInt(
       patientid);
  Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);
    /* CallableStatement calstat=conn.
         prepareCall("{call update_instanceid
         (?,?)}");


     calstat.setInt(1,patientidint);
     calstat.setInt(2,instanceid);


        ResultSet rs= calstat.executeQuery()
            ;
```

```
        conn.close();
        calstat.close();
        System.out.println("Your data has
             been updated into table.");

      */
    try{
    String update="UPDATE \"patient\" SET
         instanceid="+Integer.toString(
         instanceid)+" WHERE patientid="+
         Integer.toString(patientidint);
    System.out.println("update statement="+
         update);
    Statement st = conn.createStatement();
    int rs = st.executeUpdate(update);
    conn.commit();
    System.out.println("Your data has been
         updated into table. RS="+rs);


    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

}
public void deleteTask(String taskid)
     throws Exception{
 //list of patients


   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt","jgerona","
          bakitba?");
    conn.setAutoCommit(false);
    /* CallableStatement calstat=conn.
         prepareCall("{call update_instanceid
         (?,?)}");


     calstat.setInt(1,patientidint);
     calstat.setInt(2,instanceid);


        ResultSet rs= calstat.executeQuery()
            ;



        conn.close();
        calstat.close();
        System.out.println("Your data has
             been updated into table.");

      */
    try{
     String update="DELETE FROM i18ntext
         WHERE task_names_id="+taskid;
     System.out.println("delete i18ntxt
         statement="+update);
     Statement st = conn.createStatement();
     boolean rs = st.execute(update);
     conn.commit();
     System.out.println("i18ntext RS="+rs);

      update="DELETE FROM
          peopleassignments_bas WHERE
          task_id="+taskid;
     System.out.println("delete
         peopleassignments_bas statement="+
         update);
     st = conn.createStatement();
     rs = st.execute(update);
     conn.commit();
     System.out.println("
         peopleassignments_bas RS="+rs);

     update="DELETE FROM
         peopleassignments_potowners WHERE
         task_id="+taskid;
     System.out.println("delete
         peopleassignments_potowners
         statement="+update);
     st = conn.createStatement();
     rs = st.execute(update);
     conn.commit();
     System.out.println("
          peopleassignments_potowners RS="+rs
          );

   update="DELETE FROM task WHERE id="+
        taskid;
    System.out.println("delete task
         statement="+update);
    st = conn.createStatement();
    rs = st.execute(update);
    conn.commit();
    System.out.println("Your data has been
         updated into table. task RS="+rs);
    st.close();
    //rs.close();
    conn.close();
    }
    catch(Exception e){
     e.printStackTrace();
    }

}

public Patient getPatient(int patientid)
     throws Exception{
 //list of patients


   Class.forName("org.postgresql.Driver").
        newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );
    conn.setAutoCommit(false);
     CallableStatement calstat=conn.
          prepareCall("{call getpatient(?)}")
          ;

     calstat.setInt(1,(int)patientid);
       ResultSet rs = calstat.executeQuery()
            ;
      Patient patient = new Patient();
       while(rs.next()){

           patient.setId(rs.getString(1));
           patient.setName(rs.getString(2)+"
                "+rs.getString(3)+" "+rs.
               getString(4));
           patient.setupcdId(rs.getString(5)
               );
          patient.setinstanceid(rs.getString
               ("instanceid"));

          patient.setbirthday(rs.getString("
               birthdate"));
        patient.setage(getAge(patient.
             getbirthday()));
          patient.setgender(rs.getString("
               gender"));
          String address="";
          address=rs.getString("address");
          if(!rs.getString("address2").
               equals("  ")){address+=","+rs.
               getString("address2");}
        if(!rs.getString("city").equals("
             ")){address+=","+rs.getString("
             city");}
      if(!rs.getString("country").equals("
           ")){address+=","+rs.getString("
           country");}
    if(!rs.getString("postalcode").equals("
          ")){address+=","+rs.getString("
         postalcode");}

          patient.setaddress(address);
      System.out.println(patient.getId()+"
           Patient Name: "+patient.getName()
           +"Patient UPCDId: "+patient.
           getupcdId());
       }
     conn.close();
     calstat.close();
     System.out.println("Successful call for
          listspecifiedpatients function");
       return patient;
}
public String getAge(String bday) throws
     Exception{
 //list of patients


  Class.forName("org.postgresql.Driver").
       newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call get_age(?)}");

    calstat.setString(1,bday);
      ResultSet rs = calstat.executeQuery()
           ;
     String age = "";
      while(rs.next()){
         age=rs.getString(1);
      }
    conn.close();
    calstat.close();
    System.out.println("Successful call for
         getage function");
      return age;
 }
public void addApproveUpdates(int patientid,
     int instanceid,int idtask,String
     nametask,int version) throws Exception{

 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt",
        database_username,database_password)
        ;
  String qmark="(";


     CallableStatement calstat=conn.
          prepareCall("{call
          insert_approveupdates(?,?,?,?,?)
          }");



     calstat.setInt(1,patientid);
     calstat.setInt(2,instanceid);
     calstat.setInt(3,idtask);
     calstat.setString(4,nametask);
     calstat.setInt(5,version);
         ResultSet rs = calstat.
              executeQuery();
         calstat.close();
         conn.close();
         System.out.println("Your data has
              been inserted into table
              approveupdates");


 }

 public void executeForms(String table,
      String fields,String values, long
      patientid,HttpSession session) throws
      Exception{

 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt",
        database_username,database_password)
        ;
  String qmark="(";

  int count = StringUtils.countMatches(
       values, "|");
  for(int i=1;i<count;i++){
   qmark=qmark+"?,";
  }
  if(table.equalsIgnoreCase("SetAppointment
       ")){
   qmark=qmark+"?,";
   table="setappointment";
  }
   qmark=qmark+"?,?,?,?,?,?,?,?,";

  qmark=qmark+"?,?)";
   System.out.println("{call insert_"+table
      +qmark+"}");
    System.out.println(values);
    CallableStatement calstat=conn.
        prepareCall("{call insert_"+table+
        qmark+"}");


    StringTokenizer st = new
        StringTokenizer(values, "|");
    calstat.setInt(1,(int)patientid);
    int j=2;
    while(st.hasMoreTokens() && j<=count
        +1) {
     String val=st.nextToken();
     calstat.setString(j,val);
     System.out.println(val + "inserted");
     j++;
    }
    if(table.equalsIgnoreCase("
        SetAppointment")){
     calstat.setString(j,session.
         getAttribute("sessionName").
         toString());
     System.out.println(session.
         getAttribute("sessionName").
         toString() + "inserted");
     j=j+1;
  }
    int version= getCurrentVersion((int)
        patientid,table);
    version=version+1;
    System.out.println("j: "+j);
    calstat.setInt(j,version);
    calstat.setString(j+1,session.
        getAttribute("sessionName").
        toString());
  //date time
    DateFormat dateFormat = new
        SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
        date));
    calstat.setString(j+2,dateFormat.
        format(date));
    Calendar cal = Calendar.getInstance();
    cal.getTime();
    SimpleDateFormat sdf = new
        SimpleDateFormat("HH:mm:ss");
    System.out.println( sdf.format(cal.
        getTime()) );
    calstat.setString(j+3,sdf.format(cal.
        getTime()));
    if(session.getAttribute("
        sessionUserRole").toString().
        toLowerCase().contains("fac")){
     calstat.setString(j+4,"Approved");
     calstat.setString(j+5,session.
         getAttribute("sessionName").
         toString());

    }
    else{
      calstat.setString(j+4,"For Approval
          ");
      calstat.setString(j+5,"");

    }
    calstat.setString(j+6,"");
    calstat.setString(j+7,"");
        ResultSet rs = calstat.
            executeQuery();
        calstat.close();
        conn.close();
        System.out.println("Your data has
            been inserted into table "+
            table);


  }
//CREATE PATIENT
  public void executeFormsCreate(String
      table, String fields,String values,int
       processId) throws Exception{

   Class.forName("org.postgresql.Driver").
       newInstance();
     Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
```

```
   String qmark="(";

   int count = StringUtils.countMatches(
       values, "|");
   for(int i=1;i<count;i++){
    qmark=qmark+"?,";
   }

   qmark=qmark+"?,?)";
    System.out.println("{call insert_"+
        table+qmark+"}");
      System.out.println(values);
      CallableStatement calstat=conn.
          prepareCall("{call insert_"+table
          +qmark+"}");


      StringTokenizer st = new
          StringTokenizer(values, "|");

      int j=1;
      while(st.hasMoreTokens() && j<=count)
           {
       String val=st.nextToken();
       calstat.setString(j,val);
       System.out.println(val + "inserted")
           ;
       j++;
      }
      calstat.setInt(j,processId);
      //date time

      /*
          calstat.setString(1,section_name)
              ;
          calstat.setString(2,
              section_description);
          calstat.setString(3,created_by);
          calstat.setString(4, created_date
              );
*/
          ResultSet rs = calstat.
              executeQuery();
          conn.close();
          calstat.close();
          System.out.println("Your data has
               been inserted into table "+
              table);


  }
 public int getCurrentVersion(int patientid
     , String tablename) throws Exception{
  Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql://
       localhost:5432/DentISt",
       database_username,database_password)
       ;
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call get_"+tablename
       +"(?)}");
    calstat.setInt(1, patientid);

    ResultSet rs = calstat.executeQuery();
    int version=0;
   if(rs.next()){
     version=rs.getInt("version");

          }
    System.out.println(version);

     conn.close();
     calstat.close();
     System.out.println("Successful call
         for version "+tablename+" function
         ");
      return version;
 }
//get form fields
 public String getFormFields(int role_id,
     String tablename) throws Exception{
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt",
        database_username,database_password
        );
   conn.setAutoCommit(false);
```

```
    CallableStatement calstat=conn.
         prepareCall("{call get_"+tablename
         +"(?)}");
      calstat.setInt(1, role_id);

      ResultSet rs = calstat.executeQuery()
           ;
    ResultSetMetaData rsmd = rs.getMetaData
         ();

    int columnsNumber = rsmd.getColumnCount
         ();

          String values="";
        System.out.println(columnsNumber +"
              : "+values);

     if(rs.next()){
     values="";
     for(int i=3;i<=columnsNumber;i++){
     values= values+ rs.getString(i) +" |
          ";

     }

          }
    System.out.println(values);

     conn.close();
     calstat.close();
     System.out.println("Successful call
          for "+tablename+" function");
      return values;
}
public ArrayList<Version> listAllVersion(
     int patientid, String tablename)
     throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call get_"+tablename
         +"(?)}");
      calstat.setInt(1, patientid);

      ResultSet rs = calstat.executeQuery()
           ;


    ArrayList<Version> versionslist=new
         ArrayList();;
       // System.out.println(columnsNumber
            +" : "+values);

     while(rs.next()){
      Version version=new Version();
     version.setVersion(rs.getString("
          version"));
     version.setUpdated_by(rs.getString("
          updated_by"));
     version.setUpdated_date(rs.getString("
          updated_date"));
     version.setUpdated_time(rs.getString("
          updated_time"));
     version.setApproved(rs.getString("
          approved"));
     version.setApproved_by(rs.getString("
          approved_by"));
     version.setApproved_date(rs.getString
          ("approved_date"));
     version.setApproved_time(rs.getString
          ("approved_time"));
     versionslist.add(version);
          }
    //System.out.println(values);

     conn.close();
     calstat.close();
     System.out.println("Successful call
          for "+tablename+" version function
          ");
      return versionslist;
}
public ArrayList<Update> listAllUpdates(
     int patientid) throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call
         get_approveupdates(?)}");
      calstat.setInt(1, patientid);

      ResultSet rs = calstat.executeQuery()
           ;


    ArrayList<Update> approvelist=new
         ArrayList();;
       // System.out.println(columnsNumber
            +" : "+values);

     while(rs.next()){
      Update update=new Update();
     update.setpatientid(rs.getString("
          patientid"));
     update.setinstanceid(rs.getString("
          instanceid"));
     update.setidtask(rs.getString("idtask
          "));
     update.setnametask(rs.getString("
          nametask"));
     update.setversion(rs.getString("
          version"));
     approvelist.add(update);
          }
    //System.out.println(values);

     conn.close();
     calstat.close();
     System.out.println("Successful call
          for approveupdates table");
      return approvelist;
}
public ArrayList<Patient> listAllPending()
     throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call
         listallpendingapp()}");


      ResultSet rs = calstat.executeQuery()
           ;


    ArrayList<Patient> approvelist=new
         ArrayList();;
       // System.out.println(columnsNumber
            +" : "+values);

     while(rs.next()){
      Patient patient=new Patient();
     patient.setId(rs.getString("patientid
          "));
     patient.setinstanceid(rs.getString("
          instanceid"));
     patient.setName(rs.getString("
          givenname")+rs.getString("
          middlename")+rs.getString("
          familyname"));
     approvelist.add(patient);
          }
    //System.out.println(values);

     conn.close();
     calstat.close();
     System.out.println("Successful call
          for approveupdates table");
      return approvelist;
}

public String getVersionFormFields(int
     role_id, String tablename, int version
     ) throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call get_"+tablename
         +"(?)}");
      calstat.setInt(1, role_id);

      ResultSet rs = calstat.executeQuery()
          ;
    ResultSetMetaData rsmd = rs.getMetaData
         ();

    int columnsNumber = rsmd.getColumnCount
         ();

           String values="";
        System.out.println(columnsNumber +"
              : "+values);

     while(rs.next()){
      if(rs.getInt("version")==version){
     values="";
     for(int i=3;i<=columnsNumber;i++){
     values= values+ rs.getString(i) +" |
          ";

     }
      }
           }
    System.out.println(values);

     conn.close();
     calstat.close();
     System.out.println("Successful call
          for "+tablename+" version function
          ");
      return values;
}
public ArrayList<String> listAllClinicians
     () throws Exception{
 ArrayList<String> listallclinicians= new
      ArrayList<String>();
 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call listalluserroles
         ()}");

      ResultSet rs = calstat.executeQuery()
          ;

      while(rs.next()){
       String clinician=rs.getString(4)+"
            "+rs.getString(5)+" "+rs.
            getString(6);
            System.out.println(clinician);
          listallclinicians.add(clinician);
          }


     conn.close();
     calstat.close();
     System.out.println("Section list
          selected.");
      return listallclinicians;
}
/**START OF APPOINTMENT LIST*/
public ArrayList<Appointment>
     appointmentList(String clinician)
     throws Exception{
 ArrayList<Appointment> listclinicianapp=
      new ArrayList<Appointment>();


 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call listclinicianapp
         (?,?,?)}");
    calstat.setString(1, "");
    calstat.setString(2, "");
    calstat.setString(3, clinician);

      ResultSet rs = calstat.executeQuery()
          ;

      while(rs.next()){
     Appointment app=new Appointment();
     app.setappointmentid(rs.getString("
          appointmentid"));
     Patient patient=getPatient(Integer.
          parseInt(rs.getString("patientid")
          ));
     app.setpatientid(patient.getId());
     app.setpatientname(patient.getName());
     app.setinstanceid(patient.
          getinstanceid());
     app.setappointmentdate(rs.getString("
          appointmentdate"));
     app.setappointmentclinician(rs.
          getString("appointmentclinician"))
          ;
     String idtask=getidtask(patient.getId
          ());
     app.setidtask(idtask);
     app.setnametask(getnametask(idtask));
     listclinicianapp.add(app);
          }


     conn.close();
     calstat.close();
     System.out.println("Clinician
          appointment selected.");
      return listclinicianapp;
}

public ArrayList<Appointment>
     appointmentAllList() throws Exception{
 ArrayList<Appointment> listclinicianapp=
      new ArrayList<Appointment>();


 Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call
         listallclinicianapp(?,?)}");
    calstat.setString(1, "");
    calstat.setString(2, "");


      ResultSet rs = calstat.executeQuery()
          ;

      while(rs.next()){
     Appointment app=new Appointment();
     app.setappointmentid(rs.getString("
          appointmentid"));
     Patient patient=getPatient(Integer.
          parseInt(rs.getString("patientid")
          ));
     app.setpatientid(patient.getId());
     app.setpatientname(patient.getName());
     app.setinstanceid(patient.
          getinstanceid());
     app.setappointmentdate(rs.getString("
          appointmentdate"));
     app.setappointmentclinician(rs.
          getString("appointmentclinician"))
          ;
     String idtask=getidtask(patient.getId
          ());
     app.setidtask(idtask);
     app.setnametask(getnametask(idtask));
     if(Integer.parseInt(app.getinstanceid
          ())!=0){
     listclinicianapp.add(app);
     }
          }
    conn.close();
    calstat.close();
    System.out.println("Clinician
        appointment selected.");
     return listclinicianapp;
}


public String getidtask(String patientid)
    throws Exception{
 //list of patients
 DentFormsSP dentSP= new DentFormsSP();
 Patient patient=dentSP.getPatient(Integer
     .parseInt(patientid));

  //int patientidint=Integer.parseInt(
      patientid);
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt","jgerona
        ","bakitba?");
   conn.setAutoCommit(false);
   String taskid="";
   try{
    String update="";

    update="SELECT * FROM task WHERE
        processinstanceid="+patient.
        getinstanceid();

    System.out.println("update statement
        ="+update);
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery(update);
   conn.commit();
   DentFormsSP dentforms=new DentFormsSP()
       ;

   if(rs.next()){

   taskid = rs.getString("id");


     }

   System.out.println("Forms started");


   st.close();
   //rs.close();
   conn.close();
   }
   catch(Exception e){
    e.printStackTrace();
   }
   return taskid;
}
public String getnametask(String taskid)
    throws Exception{
 //list of patients
 //int patientidint=Integer.parseInt(
     patientid);
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt","jgerona
        ","bakitba?");
   conn.setAutoCommit(false);
   String nametask="";
   try{
    String update="";

    update="SELECT * FROM i18ntext WHERE
        task_names_id="+taskid;

    System.out.println("update statement
        ="+update);
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery(update);
   conn.commit();
  // DentFormsSP dentforms=new DentFormsSP
      ();

   if(rs.next()){

    nametask = rs.getString("text");


     }
```

```
   System.out.println("Forms started");


   st.close();
   //rs.close();
   conn.close();
   }
   catch(Exception e){
    e.printStackTrace();
   }
   return nametask;
}
/**END OF APPOINTMENT LIST*/
public ArrayList<String> get_taskowners(
    int taskid) throws Exception{
 ArrayList<String> ownerslist= new
     ArrayList<String>();
 Class.forName("org.postgresql.Driver").
     newInstance();
 Connection conn=DriverManager.
     getConnection("jdbc:postgresql://
     localhost:5432/DentISt","jgerona","
     bakitba?");
 conn.setAutoCommit(false);
 CallableStatement calstat=conn.
     prepareCall("{call
     get_taskpotentialowner(?)}");
 calstat.setInt(1,taskid);
 ResultSet rs = calstat.executeQuery();
 while(rs.next()){
 String taskowner= rs.getString(1);
 ownerslist.add(taskowner);
 //System.out.println(rs.getString(1));
 }
 conn.close();
 calstat.close();
 System.out.println("Successful call for
     get_taskpotentialowner function");
 return ownerslist;
 }
public ArrayList<String>
    checkaccess_section(int patientid)
    throws Exception{
 ArrayList<String> list= new ArrayList<
     String>();
 Class.forName("org.postgresql.Driver").
     newInstance();
 Connection conn=DriverManager.
     getConnection("jdbc:postgresql://
     localhost:5432/DentISt","jgerona","
     bakitba?");
 conn.setAutoCommit(false);
 CallableStatement calstat=conn.
     prepareCall("{call
     accesssection_check(?)}");
 calstat.setInt(1, patientid);
 ResultSet rs = calstat.executeQuery();
 while(rs.next()){
 String patient= rs.getString(1);

 list.add(patient);
 //System.out.println(rs.getString(1));
 }

 conn.close();
 calstat.close();
 System.out.println("Successful call for
     checkpriv function");
 return list;
 }
public ArrayList<Integer> getpatientIDList
    () throws Exception{
 ArrayList<Integer> listpatientids= new
     ArrayList<Integer>();
 Class.forName("org.postgresql.Driver").
     newInstance();
 System.out.println("Check"+
     database_username);
 Connection conn=DriverManager.
     getConnection("jdbc:postgresql://
     localhost:5432/DentISt","abjarabelo
     ","yes?bakitpo?");


   conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call
       listallpatientids_section()}");

     ResultSet rs = calstat.executeQuery()
         ;
```

```
      while(rs.next()){
          // System.out.println(rs.
               getString(1));
     listpatientids.add(rs.getInt(1));
         }


     conn.close();
     calstat.close();
     System.out.println("Successful call
          for listallusername function");

   return listpatientids;

  }




  public void insert_specificsection(int
       patientid, String section_name) throws
        Exception{

  Class.forName("org.postgresql.Driver").
       newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password)
         ;

      CallableStatement calstat=conn.
           prepareCall("{call
           insert_specificsection(?,?)}");
           calstat.setInt(1,patientid);
           calstat.setString(2,section_name);


           ResultSet rs = calstat.
                executeQuery();


           conn.close();
           calstat.close();
           System.out.println("Your data has
                been inserted into
                specificsection.");

  }

   public void update_specificsection(int
        patientid, String section_name)
        throws Exception{

  Class.forName("org.postgresql.Driver").
       newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password)
         ;

      CallableStatement calstat=conn.
           prepareCall("{call
           update_specificsection(?,?)}");
           calstat.setInt(1,patientid);
           calstat.setString(2,section_name);


           ResultSet rs = calstat.
                executeQuery();


           conn.close();
           calstat.close();
           System.out.println("Your data has
                been updated into
                specificsection.");

  }
   //list of users
   public ArrayList<Patient>
        list_specifiedsection(String
        currentRole,String nameSearch) throws
         Exception{
    ArrayList<Patient> listPatients= new
         ArrayList<Patient>();
    Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt",
           database_username,
           database_password);
     conn.setAutoCommit(false);
      CallableStatement calstat=conn.
           prepareCall("{call
           list_specifiedsection(?,?)}");
       calstat.setString(1,currentRole);
     calstat.setString(2,nameSearch);
         ResultSet rs = calstat.executeQuery
              ();

         while(rs.next()){
            Patient patient = new Patient();
              patient.setId(rs.getString(1));
              patient.setName(rs.getString(2)
                   +" "+rs.getString(3)+" "+rs
                   .getString(4));
              patient.setupcdId(rs.getString
                   (5));
              patient.setinstanceid(rs.
                   getString(16));
              listPatients.add(patient);
         System.out.println(patient.getId()+"
              Patient Name: "+patient.getName
              ()+"Patient UPCDId: "+patient.
              getupcdId());

            }


        conn.close();
        calstat.close();
        System.out.println("Successful call
             for listallroles function");
         return listPatients;
   }
}

package org.dentist.version.three.db;

import java.sql.Array;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import org.dentist.version.three.form.
     CariesStatus;
import org.dentist.version.three.form.
     DentalChart;
import org.dentist.version.three.form.
     RecurrentStatus;
import org.dentist.version.three.form.
     RestorationStatus;
import org.dentist.version.three.form.
     ServiceNeeded;
import org.dentist.version.three.mapper.
     CariesStatusMapper;
import org.dentist.version.three.mapper.
     DentalChartMapper;
import org.dentist.version.three.mapper.
     RecurrentStatusMapper;
import org.dentist.version.three.mapper.
     RestorationStatusMapper;
import org.dentist.version.three.mapper.
     ServicesNeededMapper;
import com.dentist.version.three.mapper.
     UserMapper;

public class DentalChartSP {

 private String database_username;
 private final String database_password="yes
      ?bakitpo?";

 public void setDatabase_username(String
      database_username) {
  this.database_username = database_username
       ;
 }


 public String getDatabase_username() {
  return database_username;
 }
```

```
//INSERT DentalChart
 public void insertDentalChart(DentalChart
      dentalChart) throws Exception{
  Integer[] temporary= {0};
  Array caries = null;
        Array recurrentcaries=null;
        Array restoration = null;
        Array removablepartial = null;
        Array extrusion = null;
        Array intrusion = null;
        Array mesial_rotation = null;
        Array distal_rotation = null;
        Array rotation = null;
        Array postcore_crown = null;
        Array rootcanal_treatment = null;
        Array pitfissure_sealants = null;
        Array extracted = null;
        Array missing = null;
        Array unerupted = null;
        Array impacted = null;
        Array acrylic_crown = null;
        Array porcelain_crown = null;
        Array metal_crown = null;
        Array porcelain_infused = null;
        Array fixed_bridge = null;

   Class.forName("org.postgresql.Driver").
        newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql://
         localhost:5432/DentISt",
         database_username,database_password
         );

    if (dentalChart.getCaries() != null) {
         caries = conn.createArrayOf("int4
              ", dentalChart.getCaries());
    }
    else{
          caries = conn.createArrayOf("int4
               ",temporary);
    }
    if (dentalChart.getRecurrentcaries() !=
          null) {
         recurrentcaries = conn.
              createArrayOf("int4",
              dentalChart.getRecurrentcaries
              ());
    }
    else{
     recurrentcaries = conn.createArrayOf("
          int4",temporary);
    }
    if (dentalChart.getRestoration() !=
         null) {

         restoration = conn.createArrayOf("
              int4", dentalChart.
              getRestoration());
    }
    else{
     restoration = conn.createArrayOf("int4
          ",temporary);
    }

    if (dentalChart.
         getRemovable_partial_denture() !=
         null) {

         removablepartial = conn.
              createArrayOf("int4",
              dentalChart.
              getRemovable_partial_denture()
              );
    }
    else{
     removablepartial = conn.createArrayOf
          ("int4",temporary);
    }
    if (dentalChart.getExtrusion() != null)
          {

         extrusion = conn.createArrayOf("
              int4", dentalChart.
              getExtrusion());
    }
    else{
     extrusion = conn.createArrayOf("int4",
          temporary);
    }
    if (dentalChart.getIntrusion() != null)
          {

         intrusion = conn.createArrayOf("
              int4", dentalChart.
              getIntrusion());
    }
    else{
     intrusion = conn.createArrayOf("int4",
          temporary);
    }
    if (dentalChart.getMesial_rotation() !=
          null) {

         mesial_rotation = conn.
              createArrayOf("int4",
              dentalChart.getMesial_rotation
              ());
    }
    else{
     mesial_rotation = conn.createArrayOf("
          int4",temporary);
    }
    if (dentalChart.getDistal_rotation() !=
          null) {

         distal_rotation = conn.
              createArrayOf("int4",
              dentalChart.getDistal_rotation
              ());
    }
    else{
     distal_rotation = conn.createArrayOf("
          int4",temporary);
    }
    if (dentalChart.getRotation() != null)
          {

         rotation = conn.createArrayOf("
              int4", dentalChart.getRotation
              ());
    }
    else{
     rotation = conn.createArrayOf("int4",
          temporary);
    }
    if (dentalChart.getPostcore_crown() !=
         null) {

         postcore_crown = conn.
              createArrayOf("int4",
              dentalChart.getPostcore_crown
              ());
    }
    else{
     postcore_crown = conn.createArrayOf("
          int4",temporary);
    }
    if (dentalChart.getRootcanal_treatment
         () != null) {

         rootcanal_treatment = conn.
              createArrayOf("int4",
              dentalChart.
              getRootcanal_treatment());
    }
    else{
     rootcanal_treatment = conn.
          createArrayOf("int4",temporary);
    }
    if (dentalChart.getPitfissure_sealants
         () != null) {

         pitfissure_sealants = conn.
              createArrayOf("int4",
              dentalChart.
              getPitfissure_sealants());
    }
    else{
     pitfissure_sealants = conn.
          createArrayOf("int4",temporary);
    }
    if (dentalChart.getExtracted() != null)
          {

         extracted = conn.createArrayOf("
              int4", dentalChart.
              getExtracted());
    }
    else{
     extracted = conn.createArrayOf("int4",
          temporary);
    }
```

```
if (dentalChart.getMissing() != null) {

     missing = conn.createArrayOf("int4
          ", dentalChart.getMissing());
}
else{
 missing = conn.createArrayOf("int4",
      temporary);
}
if (dentalChart.getUnerupted() != null)
      {

     unerupted = conn.createArrayOf("
          int4", dentalChart.
          getUnerupted());
}
else{
 unerupted = conn.createArrayOf("int4",
      temporary);
}
if (dentalChart.getImpacted() != null)
     {

     impacted = conn.createArrayOf("
          int4", dentalChart.getImpacted
          ());
}
else{
 impacted = conn.createArrayOf("int4",
      temporary);
}
if (dentalChart.getAcrylic_crown() !=
     null) {

     acrylic_crown = conn.createArrayOf
          ("int4", dentalChart.
          getAcrylic_crown());
}
else{
 acrylic_crown = conn.createArrayOf("
      int4",temporary);
}
if (dentalChart.getPorcelain_crown() !=
      null) {

     porcelain_crown = conn.
          createArrayOf("int4",
          dentalChart.getPorcelain_crown
          ());
}
else{
 porcelain_crown = conn.createArrayOf("
      int4",temporary);
}
if (dentalChart.getMetal_crown() !=
     null) {

     metal_crown = conn.createArrayOf("
          int4", dentalChart.
          getMetal_crown());
}
else{
 metal_crown = conn.createArrayOf("int4
      ",temporary);
}
if (dentalChart.getPorcelain_infused()
     != null) {

     porcelain_infused = conn.
          createArrayOf("int4",
          dentalChart.
          getPorcelain_infused());
}
else{
 porcelain_infused = conn.createArrayOf
      ("int4",temporary);
}
if (dentalChart.getFixed_bridge() !=
     null) {

     fixed_bridge = conn.createArrayOf
          ("int4", dentalChart.
          getFixed_bridge());
}
else{
 fixed_bridge = conn.createArrayOf("
      int4",temporary);
}


  CallableStatement calstat=conn.
       prepareCall("{call
          insert_dental_chart
          (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
          }");
         calstat.setInt(1,dentalChart.
              getPatient_id());
         calstat.setInt(2,dentalChart.
              getClinician_id());
         calstat.setArray(3,caries);
         calstat.setArray(4,
              recurrentcaries);
         calstat.setArray(5,restoration);
         calstat.setString(6,dentalChart.
              getComplete_denture());
         calstat.setString(7,dentalChart.
              getSingle_denture());
         calstat.setArray(8,
              removablepartial);
         calstat.setArray(9,extrusion);
         calstat.setArray(10,intrusion);
         calstat.setArray(11,
              mesial_rotation);
         calstat.setArray(12,
              distal_rotation);
         calstat.setArray(13,rotation);
         calstat.setArray(14,
              postcore_crown);
         calstat.setArray(15,
              rootcanal_treatment);
         calstat.setArray(16,
              pitfissure_sealants);
         calstat.setArray(17,extracted);
         calstat.setArray(18,missing);
         calstat.setArray(19,unerupted);
         calstat.setArray(20,impacted);
         calstat.setArray(21,
              porcelain_crown);
         calstat.setArray(22,metal_crown);
         calstat.setArray(23,acrylic_crown
              );
         calstat.setArray(24,
              porcelain_infused);
         calstat.setArray(25,fixed_bridge)
              ;
         calstat.setInt(26,dentalChart.
              getVersion());
         calstat.setString(27,dentalChart.
              getUpdated_by());
         calstat.setString(28,dentalChart.
              getUpdated_date());
         calstat.setString(29,dentalChart.
              getUpdated_time());
         calstat.setString(30,dentalChart.
              getIs_current());
           calstat.setString(31,
                dentalChart.getApproved());
          calstat.setString(32,dentalChart
               .getApproved_by());
           calstat.setString(33,
                dentalChart.
                getApproved_date());
         calstat.setString(34,dentalChart.
              getApprovated_time());



         ResultSet rs = calstat.
              executeQuery();


         conn.close();
         calstat.close();
         System.out.println("Your data has
               been inserted into table.");

   }

//INSERT CariesStatus
 public void insertCariesStatus(
      CariesStatus cariesStatus) throws
      Exception{
  Integer[] temporary= {0};
  String[] stringTemp={" "};
   Array distalcaries=null;
        Array buccalcaries = null;
        Array lingualcaries = null;
        Array mesialcaries = null;
        Array occlusalcaries = null;
        Array distalcariesrestore=null;
        Array buccalcariesrestore =null;
        Array lingualcariesrestore = null;
        Array mesialcariesrestore = null;
```

```
    Array occlusalcariesrestore = null;


Class.forName("org.postgresql.Driver").
    newInstance();
 Connection conn=DriverManager.
     getConnection("jdbc:postgresql://
     localhost:5432/DentISt",
     database_username,database_password
     );

 if (cariesStatus.getDistal_caries() !=
     null) {
  distalcaries = conn.createArrayOf("
      int4", cariesStatus.
      getDistal_caries());
 }
 else{
  distalcaries = conn.createArrayOf("
      int4",temporary);
 }

 if (cariesStatus.getBuccal_caries() !=
     null) {
  buccalcaries = conn.createArrayOf("
      int4", cariesStatus.
      getBuccal_caries());
 }
 else{
  buccalcaries = conn.createArrayOf("
      int4",temporary);
 }

 if (cariesStatus.getLingual_caries() !=
      null) {
  lingualcaries = conn.createArrayOf("
      int4", cariesStatus.
      getLingual_caries());
 }
 else{
  lingualcaries = conn.createArrayOf("
      int4",temporary);
 }

 if (cariesStatus.getMesial_caries() !=
     null) {
  mesialcaries = conn.createArrayOf("
      int4", cariesStatus.
      getMesial_caries());
 }
 else{
  mesialcaries = conn.createArrayOf("
      int4",temporary);
 }

 if (cariesStatus.getOcclusal_caries()
     != null) {
  occlusalcaries = conn.createArrayOf("
      int4", cariesStatus.
      getOcclusal_caries());
 }
 else{
  occlusalcaries = conn.createArrayOf("
      int4",temporary);
 }



 //for caries
 if (cariesStatus.
     getDistal_restorable_caries() !=
     null) {
 distalcariesrestore=conn.createArrayOf
     ("varchar", cariesStatus.
     getDistal_restorable_caries());
      }
 else{
  distalcariesrestore = conn.
      createArrayOf("varchar",stringTemp
      );
 }

 if (cariesStatus.
     getBuccal_restorable_caries() !=
     null) {
       buccalcariesrestore = conn.
           createArrayOf("varchar",
           cariesStatus.
           getBuccal_restorable_caries()
           );
 }
 else{
       buccalcariesrestore = conn.
           createArrayOf("varchar",
           stringTemp);
 }

 if (cariesStatus.
     getLingual_restorable_caries() !=
     null) {
       lingualcariesrestore = conn.
           createArrayOf("varchar",
           cariesStatus.
           getLingual_restorable_caries
           ());
 }
 else{
       lingualcariesrestore = conn.
           createArrayOf("varchar",
           stringTemp);
 }

 if (cariesStatus.
     getMesial_restorable_caries() !=
     null) {
       mesialcariesrestore = conn.
           createArrayOf("varchar",
           cariesStatus.
           getMesial_restorable_caries()
           );
 }
 else{
       mesialcariesrestore = conn.
           createArrayOf("varchar",
           stringTemp);
 }

 if (cariesStatus.
     getOcclusal_restorable_caries() !=
     null) {
       occlusalcariesrestore = conn.
           createArrayOf("varchar",
           cariesStatus.
           getOcclusal_restorable_caries
           ());
 }
 else{
       occlusalcariesrestore = conn.
           createArrayOf("varchar",
           stringTemp);
 }



    CallableStatement calstat=conn.
        prepareCall("{call
        insert_caries_status
        (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
        }");
       calstat.setInt(1,cariesStatus.
           getPatient_id());
       calstat.setArray(2,distalcaries);
       calstat.setArray(3,buccalcaries )
           ;
       calstat.setArray(4,lingualcaries
           );
       calstat.setArray(5,mesialcaries )
           ;
       calstat.setArray(6,occlusalcaries
            );
       calstat.setArray(7,
           distalcariesrestore);
       calstat.setArray(8,
           buccalcariesrestore);
       calstat.setArray(9,
           lingualcariesrestore);
       calstat.setArray(10,
           mesialcariesrestore);
       calstat.setArray(11,
           occlusalcariesrestore);
       calstat.setInt(12,cariesStatus.
           getVersion());
       calstat.setString(13,cariesStatus
           .getUpdated_by());
       calstat.setString(14,cariesStatus
           .getUpdated_date());
       calstat.setString(15,cariesStatus
           .getUpdated_time());




       ResultSet rs = calstat.
```

```
                executeQuery();

            conn.close();
            calstat.close();
            System.out.println("Your data has
                  been inserted into table.");

    }
  //INSERT recurrentStatus
   public void insertRecurrentStatus(
        RecurrentStatus recurrentStatus)
        throws Exception{
    Integer[] temporary= {0};
    String[] stringTemp={" "};
     Array distalrecurrent=null;
          Array buccalrecurrent = null;
          Array lingualrecurrent = null;
          Array mesialrecurrent = null;
          Array occlusalrecurrent = null;
          Array distalrecurrentrestore=null;
          Array buccalrecurrentrestore =null;
          Array lingualrecurrentrestore =
               null;
          Array mesialrecurrentrestore = null
               ;
          Array occlusalrecurrentrestore =
               null;


     Class.forName("org.postgresql.Driver").
          newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql://
           localhost:5432/DentISt",
           database_username,database_password
           );

      if (recurrentStatus.getDistal_recurrent
           () != null) {
       distalrecurrent = conn.createArrayOf("
            int4", recurrentStatus.
            getDistal_recurrent());
      }
      else{
       distalrecurrent = conn.createArrayOf("
            int4",temporary);
      }

      if (recurrentStatus.getBuccal_recurrent
           () != null) {
       buccalrecurrent = conn.createArrayOf("
            int4", recurrentStatus.
            getBuccal_recurrent());
      }
      else{
       buccalrecurrent = conn.createArrayOf("
            int4",temporary);
      }

      if (recurrentStatus.
           getLingual_recurrent() != null) {
       lingualrecurrent = conn.createArrayOf
            ("int4", recurrentStatus.
            getLingual_recurrent());
      }
      else{
       lingualrecurrent = conn.createArrayOf
            ("int4",temporary);
      }

      if (recurrentStatus.getMesial_recurrent
           () != null) {
       mesialrecurrent = conn.createArrayOf("
            int4", recurrentStatus.
            getMesial_recurrent());
      }
      else{
       mesialrecurrent = conn.createArrayOf("
            int4",temporary);
      }

      if (recurrentStatus.
           getOcclusal_recurrent() != null) {
       occlusalrecurrent = conn.createArrayOf
            ("int4", recurrentStatus.
            getOcclusal_recurrent());
      }
      else{
       occlusalrecurrent = conn.createArrayOf
            ("int4",temporary);
      }
```

```
//for recurrent
if (recurrentStatus.
     getDistal_restorable_recurrent() !=
      null) {
distalrecurrentrestore=conn.
     createArrayOf("varchar",
     recurrentStatus.
     getDistal_restorable_recurrent());
         }
else{
 distalrecurrentrestore = conn.
      createArrayOf("varchar",stringTemp
      );
}

if (recurrentStatus.
     getBuccal_restorable_recurrent() !=
      null) {
       buccalrecurrentrestore = conn.
            createArrayOf("varchar",
            recurrentStatus.
            getBuccal_restorable_recurrent
            ());
}
else{
       buccalrecurrentrestore = conn.
            createArrayOf("varchar",
            stringTemp);
}

if (recurrentStatus.
     getLingual_restorable_recurrent()
     != null) {
       lingualrecurrentrestore = conn.
            createArrayOf("varchar",
            recurrentStatus.
            getLingual_restorable_recurrent
            ());
}
else{
       lingualrecurrentrestore = conn.
            createArrayOf("varchar",
            stringTemp);
}

if (recurrentStatus.
     getMesial_restorable_recurrent() !=
      null) {
       mesialrecurrentrestore = conn.
            createArrayOf("varchar",
            recurrentStatus.
            getMesial_restorable_recurrent
            ());
}
else{
       mesialrecurrentrestore = conn.
            createArrayOf("varchar",
            stringTemp);
}

if (recurrentStatus.
     getOcclusal_restorable_recurrent()
     != null) {
       occlusalrecurrentrestore = conn.
            createArrayOf("varchar",
            recurrentStatus.
            getOcclusal_restorable_recurrent
            ());
}
else{
       occlusalrecurrentrestore = conn.
            createArrayOf("varchar",
            stringTemp);
}



   CallableStatement calstat=conn.
        prepareCall("{call
        insert_recurrent_status
        (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
        }");
       calstat.setInt(1,recurrentStatus.
            getPatient_id());
       calstat.setArray(2,
            distalrecurrent);
       calstat.setArray(3,
            buccalrecurrent );
       calstat.setArray(4,
                lingualrecurrent );
            calstat.setArray(5,
                mesialrecurrent );
            calstat.setArray(6,
                occlusalrecurrent );
            calstat.setArray(7,
                distalrecurrentrestore);
            calstat.setArray(8,
                buccalrecurrentrestore);
            calstat.setArray(9,
                lingualrecurrentrestore);
            calstat.setArray(10,
                mesialrecurrentrestore);
            calstat.setArray(11,
                occlusalrecurrentrestore);
            calstat.setInt(12,recurrentStatus
                .getVersion());
            calstat.setString(13,
                recurrentStatus.getUpdated_by
                ());
            calstat.setString(14,
                recurrentStatus.
                getUpdated_date());
            calstat.setString(15,
                recurrentStatus.
                getUpdated_time());





            ResultSet rs = calstat.
                executeQuery();


            conn.close();
            calstat.close();
            System.out.println("Your data has
                 been inserted into table.");

    }
  //INSERT restorationStatus
   public void insertRestorationStatus(
       RestorationStatus restorationStatus)
       throws Exception{
    Integer[] temporary= {0};
    String[] stringTemp={" "};
     Array distalrestoration=null;
          Array buccalrestoration = null;
          Array lingualrestoration = null;
          Array mesialrestoration = null;
          Array occlusalrestoration = null;
          Array distalrestorationrestore=null
              ;
          Array buccalrestorationrestore =
              null;
          Array lingualrestorationrestore =
              null;
          Array mesialrestorationrestore =
              null;
          Array occlusalrestorationrestore =
              null;


     Class.forName("org.postgresql.Driver").
         newInstance();
      Connection conn=DriverManager.
          getConnection("jdbc:postgresql://
          localhost:5432/DentISt",
          database_username,database_password
          );

      if (restorationStatus.
          getDistal_restoration() != null) {
       distalrestoration = conn.createArrayOf
           ("int4", restorationStatus.
           getDistal_restoration());
      }
      else{
       distalrestoration = conn.createArrayOf
           ("int4",temporary);
      }

      if (restorationStatus.
          getBuccal_restoration() != null) {
       buccalrestoration = conn.createArrayOf
           ("int4", restorationStatus.
           getBuccal_restoration());
      }
      else{
       buccalrestoration = conn.createArrayOf
           ("int4",temporary);
      }

      if (restorationStatus.
          getLingual_restoration() != null) {
       lingualrestoration = conn.
           createArrayOf("int4",
           restorationStatus.
           getLingual_restoration());
      }
      else{
       lingualrestoration = conn.
           createArrayOf("int4",temporary);
      }

      if (restorationStatus.
          getMesial_restoration() != null) {
       mesialrestoration = conn.createArrayOf
           ("int4", restorationStatus.
           getMesial_restoration());
      }
      else{
       mesialrestoration = conn.createArrayOf
           ("int4",temporary);
      }

      if (restorationStatus.
          getOcclusal_restoration() != null)
          {
       occlusalrestoration = conn.
           createArrayOf("int4",
           restorationStatus.
           getOcclusal_restoration());
      }
      else{
       occlusalrestoration = conn.
           createArrayOf("int4",temporary);
      }



      //for restoration
      if (restorationStatus.
          getDistal_restorable_restoration()
          != null) {
      distalrestorationrestore=conn.
          createArrayOf("varchar",
          restorationStatus.
          getDistal_restorable_restoration())
          ;      }
      else{
       distalrestorationrestore = conn.
           createArrayOf("varchar",stringTemp
           );
      }

      if (restorationStatus.
          getBuccal_restorable_restoration()
          != null) {
            buccalrestorationrestore = conn.
                createArrayOf("varchar",
                restorationStatus.
                getBuccal_restorable_restoration
                ());
      }
      else{
            buccalrestorationrestore = conn.
                createArrayOf("varchar",
                stringTemp);
      }

      if (restorationStatus.
          getLingual_restorable_restoration()
           != null) {
            lingualrestorationrestore = conn.
                createArrayOf("varchar",
                restorationStatus.
                getLingual_restorable_restoration
                ());
      }
      else{
            lingualrestorationrestore = conn.
                createArrayOf("varchar",
                stringTemp);
      }

      if (restorationStatus.
          getMesial_restorable_restoration()
          != null) {
            mesialrestorationrestore = conn.
                createArrayOf("varchar",
                restorationStatus.
                getMesial_restorable_restoration
            ());
    }
    else{
          mesialrestorationrestore = conn.
               createArrayOf("varchar",
               stringTemp);
    }

    if (restorationStatus.
         getOcclusal_restorable_restoration
         () != null) {
          occlusalrestorationrestore = conn
               .createArrayOf("varchar",
               restorationStatus.
               getOcclusal_restorable_restoration
               ());
    }
    else{
          occlusalrestorationrestore = conn
               .createArrayOf("varchar",
               stringTemp);
    }



      CallableStatement calstat=conn.
           prepareCall("{call
           insert_restoration_status
           (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
           }");
          calstat.setInt(1,
               restorationStatus.
               getPatient_id());
          calstat.setArray(2,
               distalrestoration);
          calstat.setArray(3,
               buccalrestoration );
          calstat.setArray(4,
               lingualrestoration );
          calstat.setArray(5,
               mesialrestoration );
          calstat.setArray(6,
               occlusalrestoration );
          calstat.setArray(7,
               distalrestorationrestore);
          calstat.setArray(8,
               buccalrestorationrestore);
          calstat.setArray(9,
               lingualrestorationrestore);
          calstat.setArray(10,
               mesialrestorationrestore);
          calstat.setArray(11,
               occlusalrestorationrestore);
          calstat.setInt(12,
               restorationStatus.getVersion
               ());
          calstat.setString(13,
               restorationStatus.
               getUpdated_by());
          calstat.setString(14,
               restorationStatus.
               getUpdated_date());
          calstat.setString(15,
               restorationStatus.
               getUpdated_time());




          ResultSet rs = calstat.
               executeQuery();


          conn.close();
          calstat.close();
          System.out.println("Your data has
                been inserted into table.");

  }

//INSERT ServiceNeeded
 public void insertServiceNeeded(
      ServiceNeeded serviceNeeded) throws
      Exception{
  Integer[] temporary= {0};

   Array class_1=null;
   Array class_2=null;
         Array class_3=null;
        Array class_4=null;
        Array class_5=null;
        Array onlay=null;
        Array extraction=null;
        Array odontectomy=null;
        Array special_case=null;
        Array pulp_sedation=null;
        Array crown_recementation=null;
        Array filling_service=null;
        Array laminated=null;
        Array single_crown=null;
        Array bridge_service=null;
        Array anterior=null;
        Array posterior=null;
        Array ortho_endo=null;


Class.forName("org.postgresql.Driver").
     newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password
      );

 /*
  * convert to array_sql
  */
 if (serviceNeeded.getClass_1() != null)
      {
  class_1 = conn.createArrayOf("int4", (
       serviceNeeded.getClass_1()));
 }
 else{
  class_1 = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getClass_2() != null)
      {
  class_2 = conn.createArrayOf("int4", (
       serviceNeeded.getClass_2()));
 }
 else{
  class_2 = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getClass_3() != null)
      {
  class_3 = conn.createArrayOf("int4", (
       serviceNeeded.getClass_3()));
 }
 else{
  class_3 = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getClass_4() != null)
      {
  class_4 = conn.createArrayOf("int4", (
       serviceNeeded.getClass_4()));
 }
 else{
  class_4 = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getClass_5() != null)
      {
  class_5 = conn.createArrayOf("int4", (
       serviceNeeded.getClass_5()));
 }
 else{
  class_5 = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getOnlay() != null) {
  onlay = conn.createArrayOf("int4", (
       serviceNeeded.getOnlay()));
 }
 else{
  onlay = conn.createArrayOf("int4",
       temporary);
 }
 if (serviceNeeded.getExtraction() !=
      null) {
  extraction = conn.createArrayOf("int4
       ", (serviceNeeded.getExtraction())
       );
 }
 else{
  extraction = conn.createArrayOf("int4
       ",temporary);
 }
 if (serviceNeeded.getOdontectomy() !=
      null) {
  odontectomy = conn.createArrayOf("int4
      ", (serviceNeeded.getOdontectomy()
      ));
}
else{
 odontectomy = conn.createArrayOf("int4
      ",temporary);
}
if (serviceNeeded.getSpecial_case() !=
     null) {
 special_case = conn.createArrayOf("
      int4", (serviceNeeded.
      getSpecial_case()));
}
else{
 special_case = conn.createArrayOf("
      int4",temporary);
}
if (serviceNeeded.getPulp_sedation() !=
      null) {
 pulp_sedation = conn.createArrayOf("
      int4", (serviceNeeded.
      getPulp_sedation()));
}
else{
 pulp_sedation = conn.createArrayOf("
      int4",temporary);
}
if (serviceNeeded.
     getCrown_recementation() != null) {
 crown_recementation = conn.
      createArrayOf("int4", (
      serviceNeeded.
      getCrown_recementation()));
}
else{
 crown_recementation = conn.
      createArrayOf("int4",temporary);
}
if (serviceNeeded.getFilling_service()
     != null) {
 filling_service = conn.createArrayOf("
      int4", (serviceNeeded.
      getFilling_service()));
}
else{
 filling_service = conn.createArrayOf("
      int4",temporary);
}
if (serviceNeeded.getLaminated() !=
     null) {
 laminated = conn.createArrayOf("int4",
       (serviceNeeded.getLaminated()));
}
else{
 laminated = conn.createArrayOf("int4",
      temporary);
}
if (serviceNeeded.getSingle_crown() !=
     null) {
 single_crown = conn.createArrayOf("
      int4", (serviceNeeded.
      getSingle_crown()));
}
else{
 single_crown = conn.createArrayOf("
      int4",temporary);
}
if (serviceNeeded.getBridge_service()
     != null) {
 bridge_service = conn.createArrayOf("
      int4", (serviceNeeded.
      getBridge_service()));
}
else{
 bridge_service = conn.createArrayOf("
      int4",temporary);
}
if (serviceNeeded.getAnterior() != null
     ) {
 anterior = conn.createArrayOf("int4",
      (serviceNeeded.getAnterior()));
}
else{
 anterior = conn.createArrayOf("int4",
      temporary);
}
if (serviceNeeded.getPosterior() !=
     null) {
 posterior = conn.createArrayOf("int4",
       (serviceNeeded.getPosterior()));
}
else{
```

```
 posterior = conn.createArrayOf("int4",
      temporary);
}
if (serviceNeeded.getOrtho_endo() !=
     null) {
 ortho_endo = conn.createArrayOf("int4
      ", (serviceNeeded.getOrtho_endo())
      );
}
else{
 ortho_endo = conn.createArrayOf("int4
      ",temporary);
}

   CallableStatement calstat=conn.
        prepareCall("{call
        insert_serviceneeded
        (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
        }");
        calstat.setInt(1,serviceNeeded.
             getPatient_id());
        calstat.setArray(2,class_1);
        calstat.setArray(3,class_2);
        calstat.setArray(4,class_3);
        calstat.setArray(5,class_4);
        calstat.setArray(6,class_5);
        calstat.setArray(7,onlay);
        calstat.setArray(8,extraction);
        calstat.setArray(9,odontectomy);
        calstat.setArray(10,special_case)
             ;
        calstat.setArray(11,pulp_sedation
             );
        calstat.setArray(12,
             crown_recementation);
        calstat.setArray(13,
             filling_service);
        calstat.setArray(14,laminated);
        calstat.setArray(15,single_crown)
             ;
        calstat.setArray(16,
             bridge_service);
        calstat.setArray(17,anterior);
        calstat.setArray(18,posterior);
        calstat.setArray(19,ortho_endo);
        calstat.setString(20,
             serviceNeeded.getPeriodontics
             ());
        calstat.setString(21,
             serviceNeeded.getSurgery());
        calstat.setString(22,
             serviceNeeded.
             getEmergency_treatment());
        calstat.setString(23,
             serviceNeeded.
             getProsthodontics());
        calstat.setString(24,
             serviceNeeded.getUpdated_by()
             );
        calstat.setString(25,
             serviceNeeded.getUpdated_date
             ());
        calstat.setString(26,
             serviceNeeded.getUpdated_time
             ());
        calstat.setInt(27,serviceNeeded.
             getVersion());
        calstat.setString(28,
             serviceNeeded.getNotes());
        calstat.setString(29,
             serviceNeeded.getIs_current()
             );



        ResultSet rs = calstat.
             executeQuery();


        conn.close();
        calstat.close();
        System.out.println("Your data has
              been inserted into table.");

}


//NEWLY ADDED
public DentalChart getDentalChart(int
     patient_id, int version) throws
     Exception{
```

```
 Class.forName("org.postgresql.Driver").
      newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call getdentalchart
         (?,?)}");
       calstat.setInt(1, patient_id);
       calstat.setInt(2, version);

        ResultSet rs= calstat.executeQuery
             ();
      int result=0;

      DentalChart dentalChart= new
           DentalChart();

      while(rs.next()){
              DentalChartMapper chartmap=
                   new DentalChartMapper();
      dentalChart= chartmap.mapRow(rs, 35)
           ;

             }


         conn.close();
         calstat.close();

         return dentalChart;


}

public CariesStatus getCariesStatus(int
     patient_id, int version) throws
     Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call getcariesstatus
         (?,?)}");
       calstat.setInt(1, patient_id);
       calstat.setInt(2, version);

        ResultSet rs= calstat.executeQuery
             ();
      int result=0;

      CariesStatus cariesStatus= new
           CariesStatus();

      while(rs.next()){
              CariesStatusMapper chartmap=
                   new CariesStatusMapper();
              cariesStatus= chartmap.mapRow
                   (rs, 16);

             }


         conn.close();
         calstat.close();

         return cariesStatus;


}
public RecurrentStatus getRecurrentStatus(
     int patient_id, int version) throws
     Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call
         getrecurrentstatus(?,?)}");
       calstat.setInt(1, patient_id);
       calstat.setInt(2, version);

        ResultSet rs= calstat.executeQuery
             ();
      int result=0;

      RecurrentStatus recurrentStatus= new
           RecurrentStatus();

      while(rs.next()){
              RecurrentStatusMapper
                   chartmap= new
                   RecurrentStatusMapper();
              recurrentStatus= chartmap.
                   mapRow(rs, 16);

             }


         conn.close();
         calstat.close();

         return recurrentStatus;


}
public RestorationStatus
     getRestorationStatus(int patient_id,
     int version) throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call
         getrestorationstatus(?,?)}");
       calstat.setInt(1, patient_id);
       calstat.setInt(2, version);

        ResultSet rs= calstat.executeQuery
             ();
      int result=0;

      RestorationStatus restorationStatus=
           new RestorationStatus();

      while(rs.next()){
              RestorationStatusMapper
                   chartmap= new
                   RestorationStatusMapper()
                   ;
              restorationStatus= chartmap.
                   mapRow(rs, 16);

             }


         conn.close();
         calstat.close();

         return restorationStatus;


}

public ServiceNeeded getServiceNeeded(int
     patient_id, int version) throws
     Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call
         getserviceneeded(?,?)}");
       calstat.setInt(1, patient_id);
       calstat.setInt(2, version);

        ResultSet rs= calstat.executeQuery
             ();
      //int result=0;
```

```
        ServiceNeeded service= new
            ServiceNeeded();

        while(rs.next()){
          ServicesNeededMapper chartmap= new
               ServicesNeededMapper();
          service= chartmap.mapRow(rs, 30);

              }


          conn.close();
          calstat.close();

          return service;


  }

  public ArrayList<Integer> getpatientIDList
       () throws Exception{
   ArrayList<Integer> listpatientids= new
        ArrayList<Integer>();
   Class.forName("org.postgresql.Driver").
        newInstance();
   System.out.println("Check"+
        database_username);
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql://
        localhost:5432/DentISt",
        database_username,database_password);


     conn.setAutoCommit(false);
     CallableStatement calstat=conn.
         prepareCall("{call
         listallpatientids()}");

       ResultSet rs = calstat.executeQuery()
           ;

       while(rs.next()){
           // System.out.println(rs.
                getString(1));
      listpatientids.add(rs.getInt(1));
           }
```

```
       conn.close();
       calstat.close();
       System.out.println("Successful call
            for listallusername function");

    return listpatientids;

  }

  //update User
   public void updateDentalChart(int
        patient_id, int version, String
        is_current) throws Exception{

     Class.forName("org.postgresql.Driver").
          newInstance();

       Connection conn=DriverManager.
            getConnection("jdbc:postgresql://
            localhost:5432/DentISt",
            database_username,
            database_password);

         CallableStatement calstat=conn.
              prepareCall("{call
              update_dentalchart(?,?,?)}");
            //calstat.registerOutParameter(10,
                 java.sql.Types.INTEGER);
             calstat.setInt(1,patient_id);
               calstat.setInt(2, version);
               calstat.setString(3,
                    is_current);

             ResultSet rs= calstat.
                  executeQuery();

             conn.close();
             calstat.close();
             System.out.println("update
                  version");

    }
}
```

```
package org.dentist.version.three.form;

import java.util.ArrayList;

public class CariesStatus {

 private int caries_id;
 private int patient_id;
 private Integer[] distal_caries;
 private Integer[] buccal_caries;
 private Integer[] lingual_caries;
 private Integer[] mesial_caries;
 private Integer[] occlusal_caries;
 private String[] distal_restorable_caries;
 private String[] buccal_restorable_caries;
 private String[] lingual_restorable_caries;
 private String[] mesial_restorable_caries;
 private String[] occlusal_restorable_caries
      ;
 private int version;
 private String updated_by;
 private String updated_date;
 private String updated_time;


 private String[] distal_caries_string;
 private String[] buccal_caries_string;
 private String[] lingual_caries_string;
 private String[] mesial_caries_string;
 private String[] occlusal_caries_string;

 private Integer[] splitStringToInteger(
      String[] temp){
  Integer[] result= new Integer[temp.length
       ];
  String tempResult= "";
  for(int i=0; i<temp.length;i++){
   tempResult=temp[i];
   if(temp[i].indexOf("{")!=-1 || temp[i].
        indexOf("}")!=-1){
    if(temp[i].indexOf("{")!=-1)
     tempResult=tempResult.replace("{", "").
          trim();
     if(temp[i].indexOf("}")!=-1)
```

```
     tempResult=tempResult.replace("}", "").
          trim();

    System.out.println("Check: "+ tempResult
         );
   }

   result[i]=Integer.parseInt(tempResult);
  }
  return result;
 }

 private String[] splitStringToSpecific(
      String[] temp){
  ArrayList<String> tempResult= new
       ArrayList<String>();
  String tempAr= "";
  String[] result=null;
  if(temp!=null){
  for(int i=0; i<temp.length;i++){
   if(temp[i].length()>2)
   tempResult.add(temp[i]);
  }
  result= new String[tempResult.size()];
  for(int j=0;j<tempResult.size();j++){
   tempAr=tempResult.get(j);

   if(tempResult.get(j).indexOf("{")!=-1 ||
        tempResult.get(j).indexOf("}")!=-1){
    if(tempResult.get(j).indexOf("{")!=-1)
     tempAr=tempAr.replace("{", "").trim();
     if(tempResult.get(j).indexOf("}")!=-1)
      tempAr=tempAr.replace("}", "").trim();

    System.out.println("Check: "+ tempAr +"
         noms");
   }
   result[j]=tempAr;
  }
  }
  else{
   result=new String[1];
   result[0]="-1";
  }
 return result;
}

public void setValuesNeeded(String name,
     String value){
 String[] partial=null;
 if(name.equals("distal_caries")){
  partial=value.split(",");
  this.setDistal_caries(partial);
 }
 else if(name.equals("buccal_caries")){
  partial=value.split(",");
  this.setBuccal_caries(partial);
 }
 else if(name.equals("lingual_caries")){
  partial=value.split(",");
  this.setLingual_caries(partial);
 }
 else if(name.equals("mesial_caries")){
  partial=value.split(",");
  this.setMesial_caries(partial);
 }
 else if(name.equals("occlusal_caries")){
  partial=value.split(",");
  this.setOcclusal_caries(partial);
 }
 else if(name.equals("
      distal_restorable_caries")){
  partial=value.split(",");
  this.setDistal_restorable_caries(partial)
       ;
 }
 else if(name.equals("
      buccal_restorable_caries")){
  partial=value.split(",");
  this.setBuccal_restorable_caries(partial)
       ;
 }
 else if(name.equals("
      lingual_restorable_caries")){
  partial=value.split(",");
  this.setLingual_restorable_caries(partial
       );
 }
 else if(name.equals("
      mesial_restorable_caries")){
  partial=value.split(",");
  this.setMesial_restorable_caries(partial)
       ;
 }
 else if(name.equals("
      occlusal_restorable_caries")){
  partial=value.split(",");
  this.setOcclusal_restorable_caries(
       partial);
 }
}

public void setCaries_id(int caries_id) {
 this.caries_id = caries_id;
}
public int getCaries_id() {
 return caries_id;
}
public void setPatient_id(int patient_id) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setDistal_caries(String[]
     distal_caries) {
 Integer[] distal_cariesResult=null;
 if (distal_caries != null) {
 distal_cariesResult=splitStringToInteger(
      distal_caries);
 }
 this.setDistal_caries_string(
      splitStringToSpecific(distal_caries));
 this.distal_caries = distal_cariesResult;
}
public Integer[] getDistal_caries() {
 return distal_caries;
}

public void setBuccal_caries(String[]
     buccal_caries) {
 Integer[] buccal_cariesResult=null;
 if (buccal_caries != null) {
 buccal_cariesResult=splitStringToInteger(
      buccal_caries);
 }
 this.setBuccal_caries_string(
      splitStringToSpecific(buccal_caries));
 this.buccal_caries = buccal_cariesResult;
}

public Integer[] getBuccal_caries() {
 return buccal_caries;
}

public void setLingual_caries(String[]
     lingual_caries) {
 Integer[] lingual_cariesResult=null;
 if (lingual_caries != null ) {
 lingual_cariesResult=splitStringToInteger(
      lingual_caries);
 }
 this.setLingual_caries_string(
      splitStringToSpecific(lingual_caries))
      ;
 this.lingual_caries = lingual_cariesResult
      ;
}

public Integer[] getLingual_caries() {
 return lingual_caries;
}

public void setMesial_caries(String[]
     mesial_caries) {
 Integer[] mesial_cariesResult=null;
 if (mesial_caries != null ) {
 mesial_cariesResult=splitStringToInteger(
      mesial_caries);
 }
 this.setMesial_caries_string(
      splitStringToSpecific(mesial_caries));
 this.mesial_caries = mesial_cariesResult;
}

public Integer[] getMesial_caries() {
 return mesial_caries;
}

public void setOcclusal_caries(String[]
     occlusal_caries) {
 Integer[] occlusal_cariesResult=null;
 if (occlusal_caries != null ) {
 occlusal_cariesResult=splitStringToInteger
      (occlusal_caries);
 }
 this.setOcclusal_caries_string(
      splitStringToSpecific(occlusal_caries)
      );
 this.occlusal_caries =
      occlusal_cariesResult;
}

public Integer[] getOcclusal_caries() {
 return occlusal_caries;
}

public void setDistal_restorable_caries(
     String[] distal_restorable_caries) {
 String[] distal_restorable_cariessResult=
      null;
 if (distal_restorable_caries != null ) {
  distal_restorable_cariessResult=
       splitStringToSpecific(
       distal_restorable_caries);
 }
 this.distal_restorable_caries =
      distal_restorable_cariessResult;
}

public String[] getDistal_restorable_caries
     () {
 return distal_restorable_caries;
}

public void setBuccal_restorable_caries(
     String[] buccal_restorable_caries) {
 String[] buccal_restorable_cariesResult=
      null;
 if (buccal_restorable_caries != null ) {
  buccal_restorable_cariesResult=
       splitStringToSpecific(
       buccal_restorable_caries);
 }
 this.buccal_restorable_caries =
      buccal_restorable_cariesResult;
}
```

```
 public String[] getBuccal_restorable_caries
     () {
  return buccal_restorable_caries;
 }

 public void setLingual_restorable_caries(
     String[] lingual_restorable_caries) {
  String[] lingual_restorable_cariesResult=
      null;
  if (lingual_restorable_caries != null ) {
   lingual_restorable_cariesResult=
       splitStringToSpecific(
       lingual_restorable_caries);
  }
  this.lingual_restorable_caries =
      lingual_restorable_cariesResult;
 }

 public String[]
     getLingual_restorable_caries() {
  return lingual_restorable_caries;
 }

 public void setMesial_restorable_caries(
     String[] mesial_restorable_caries) {
  String[] mesial_restorable_cariesResult=
      null;
  if (mesial_restorable_caries != null) {
   mesial_restorable_cariesResult=
       splitStringToSpecific(
       mesial_restorable_caries);
  }
  this.mesial_restorable_caries =
      mesial_restorable_cariesResult;
 }

 public String[] getMesial_restorable_caries
     () {
  return mesial_restorable_caries;
 }

 public void setOcclusal_restorable_caries(
     String[] occlusal_restorable_caries) {
  String[] occlusal_restorable_cariesResult=
      null;
  if (occlusal_restorable_caries != null) {
   occlusal_restorable_cariesResult=
       splitStringToSpecific(
       occlusal_restorable_caries);
  }
  this.occlusal_restorable_caries =
      occlusal_restorable_cariesResult;
 }

 public String[]
     getOcclusal_restorable_caries() {
  return occlusal_restorable_caries;
 }

 public void setVersion(int version) {
  this.version = version;
 }

 public int getVersion() {
  return version;
 }

 public void setUpdated_by(String updated_by
     ) {
  this.updated_by = updated_by;
 }

 public String getUpdated_by() {
  return updated_by;
 }

 public void setUpdated_date(String
     updated_date) {
  this.updated_date = updated_date;
 }

 public String getUpdated_date() {
  return updated_date;
 }

 public void setUpdated_time(String
     updated_time) {
  this.updated_time = updated_time;
 }

 public String getUpdated_time() {
  return updated_time;
 }

 public void setDistal_caries_string(String
     [] distal_caries_string) {
  this.distal_caries_string =
      distal_caries_string;
 }

 public String[] getDistal_caries_string() {
  return distal_caries_string;
 }

 public void setBuccal_caries_string(String
     [] buccal_caries_string) {
  this.buccal_caries_string =
      buccal_caries_string;
 }

 public String[] getBuccal_caries_string() {
  return buccal_caries_string;
 }

 public void setLingual_caries_string(String
     [] lingual_caries_string) {
  this.lingual_caries_string =
      lingual_caries_string;
 }

 public String[] getLingual_caries_string()
     {
  return lingual_caries_string;
 }

 public void setOcclusal_caries_string(
     String[] occlusal_caries_string) {
  this.occlusal_caries_string =
      occlusal_caries_string;
 }

 public String[] getOcclusal_caries_string()
      {
  return occlusal_caries_string;
 }

 public void setMesial_caries_string(String
     [] mesial_caries_string) {
  this.mesial_caries_string =
      mesial_caries_string;
 }

 public String[] getMesial_caries_string() {
  return mesial_caries_string;
 }



}
```

```
package org.dentist.version.three.form;

import java.util.ArrayList;

public class DentalChart {

 private int dental_chart_id;
 private int patient_id;
 private int clinician_id;
 private Integer[] caries;
 private Integer[] recurrentcaries;
 private Integer[] restoration;
 private String complete_denture;
 private String single_denture;
 private Integer[] removable_partial_denture
     ;
 private Integer[] extrusion;
 private Integer[] intrusion;
 private Integer[] mesial_rotation;
 private Integer[] distal_rotation;
 private Integer[] rotation;
 private Integer[] postcore_crown;
 private Integer[] rootcanal_treatment;
 private Integer[] pitfissure_sealants;
 private Integer[] extracted;
 private Integer[] missing;
 private Integer[] unerupted;
 private Integer[] impacted;
private Integer[] porcelain_crown;
private Integer[] acrylic_crown;
private Integer[] metal_crown;
private Integer[] porcelain_infused;
private Integer[] fixed_bridge;
private int version;
private String updated_by;
private String updated_date;
private String updated_time;
private String is_current;
private String approved;
private String approved_by;
private String approved_date;
private String approvated_time;


//String private
private String[] caries_string;
private String[] recurrentcaries_string;
private String[] restoration_string;
private String[]
     removable_partial_denture_string;
private String[] extrusion_string;
private String[] intrusion_string;
private String[] mesial_rotation_string;
private String[] distal_rotation_string;
private String[] rotation_string;
private String[] postcore_crown_string;
private String[] rootcanal_treatment_string
     ;
private String[] pitfissure_sealants_string
     ;
private String[] extracted_string;
private String[] missing_string;
private String[] unerupted_string;
private String[] impacted_string;
private String[] porcelain_crown_string;
private String[] acrylic_crown_string;
private String[] metal_crown_string;
private String[] porcelain_infused_string;
private String[] fixed_bridge_string;

private Integer[] splitStringToInteger(
     String[] temp){
 Integer[] result= new Integer[temp.length
      ];
 String tempResult= "";
 for(int i=0; i<temp.length;i++){
  tempResult=temp[i];
  if(temp[i].indexOf("{")!=-1 || temp[i].
       indexOf("}")!=-1){
   if(temp[i].indexOf("{")!=-1)
    tempResult=tempResult.replace("{", "").
         trim();
    if(temp[i].indexOf("}")!=-1)
    tempResult=tempResult.replace("}", "").
         trim();

   System.out.println("Check: "+ tempResult
        );
  }

  result[i]=Integer.parseInt(tempResult);
 }
 return result;
}

private String[] splitStringToSpecific(
     String[] temp){
 ArrayList<String> tempResult= new
      ArrayList<String>();
 String tempAr= "";
 String[] result=null;
 if(temp!=null){
 for(int i=0; i<temp.length;i++){
  if(temp[i].length()>2)
  tempResult.add(temp[i]);
 }
 result= new String[tempResult.size()];
 for(int j=0;j<tempResult.size();j++){
  tempAr=tempResult.get(j);

  if(tempResult.get(j).indexOf("{")!=-1 ||
       tempResult.get(j).indexOf("}")!=-1){
   if(tempResult.get(j).indexOf("{")!=-1)
    tempAr=tempAr.replace("{", "").trim();
    if(tempResult.get(j).indexOf("}")!=-1)
     tempAr=tempAr.replace("}", "").trim();

  }
  System.out.println("Check: "+ tempAr +"
       noms");
  result[j]=tempAr;
 }
 }
 else{
  result=new String[1];
  result[0]="-1";
 }

 return result;
}

public void setValuesNeeded(String name,
    String value){
 String[] partial=null;
 if(name.equals("caries")){
  partial=value.split(",");
  this.setCaries(partial);
 }
 else if(name.equals("recurrent_caries")){
  partial=value.split(",");
  this.setRecurrentcaries(partial);
 }
 else if(name.equals("restoration")){
  partial=value.split(",");
  this.setRestoration(partial);
 }
 else if(name.equals("
     removable_partial_denture")){
  partial=value.split(",");
  this.setRemovable_partial_denture(partial
       );
 }
 else if(name.equals("extrusion")){
  partial=value.split(",");
  this.setExtrusion(partial);
 }
 else if(name.equals("intrusion")){
  partial=value.split(",");
  this.setIntrusion(partial);
 }
 else if(name.equals("mesial_rotation")){
  partial=value.split(",");
  this.setMesial_rotation(partial);
 }
 else if(name.equals("distal_rotation")){
  partial=value.split(",");
  this.setDistal_rotation(partial);
 }
 else if(name.equals("rotation")){
  partial=value.split(",");
  this.setRotation(partial);
 }
 else if(name.equals("postcore_crown")){
  partial=value.split(",");
  this.setPostcore_crown(partial);
 }
 else if(name.equals("rootcanal_treatment")
     ){
  partial=value.split(",");
  this.setRootcanal_treatment(partial);
 }
 else if(name.equals("pitfissure_sealants")
     ){
  partial=value.split(",");
  this.setPitfissure_sealants(partial);
 }
 else if(name.equals("extracted")){
  partial=value.split(",");
  this.setExtracted(partial);
 }
 else if(name.equals("missing")){
  partial=value.split(",");
  this.setMissing(partial);
 }
 else if(name.equals("unerupted")){
  partial=value.split(",");
  this.setUnerupted(partial);
 }
 else if(name.equals("impacted")){
  partial=value.split(",");
  this.setImpacted(partial);
 }
 else if(name.equals("porcelain_crown")){
  partial=value.split(",");
  this.setPorcelain_crown(partial);
 }
 else if(name.equals("acrylic_crown")){
  partial=value.split(",");
  this.setAcrylic_crown(partial);
 }
 else if(name.equals("metal_crown")){
  partial=value.split(",");
  this.setMetal_crown(partial);
 }
 else if(name.equals("porcelain_infused")){
  partial=value.split(",");
  this.setPorcelain_infused(partial);
 }
 else if(name.equals("fixed_bridge")){
  partial=value.split(",");
  this.setFixed_bridge(partial);
 }
}


public void setDental_chart_id(int
     dental_chart_id) {
 this.dental_chart_id = dental_chart_id;
}
public int getDental_chart_id() {
 return dental_chart_id;
}
public void setPatient_id(int patient_id) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setClinician_id(int
     clinician_id) {
 this.clinician_id = clinician_id;
}
public int getClinician_id() {
 return clinician_id;
}
public void setCaries(String[] caries) {
 Integer[] cariesResult=null;
if (caries != null) {
 cariesResult=splitStringToInteger(caries);
}
 this.setCaries_string(
      splitStringToSpecific(caries));
 this.caries = cariesResult;
}


public Integer[] getCaries() {
 return caries;
}

public void setRecurrentcaries(String[]
     recurrentcaries) {
 Integer[] recurrentResult=null;
 if (recurrentcaries != null) {

 recurrentResult=splitStringToInteger(
      recurrentcaries);

 }
 this.setRecurrentcaries_string(
      splitStringToSpecific(recurrentcaries)
      );
 this.recurrentcaries = recurrentResult;
}
public Integer[] getRecurrentcaries() {
 return recurrentcaries;
}


public void setRestoration(String[]
     restoration) {
 Integer[] restorationtResult=null;
 if (restoration != null) {

 restorationtResult=splitStringToInteger(
      restoration);

 }
 this.setRestoration_string(
      splitStringToSpecific(restoration));
 this.restoration = restorationtResult;
}


public Integer[] getRestoration() {
 return restoration;
}


public void setRemovable_partial_denture(
     String[] removable_partial_denture) {
 Integer[] removable_partial_dentureResult=
      null;
 if (removable_partial_denture != null) {
 removable_partial_dentureResult=
      splitStringToInteger(
      removable_partial_denture);
 }
 this.setRemovable_partial_denture_string(
      splitStringToSpecific(
      removable_partial_denture));
 this.removable_partial_denture =
      removable_partial_dentureResult;
}


public Integer[]
     getRemovable_partial_denture() {
 return removable_partial_denture;
}


public void setExtrusion(String[] extrusion
     ) {
 Integer[] extrusionResult=null;
 if (extrusion != null) {
 extrusionResult=splitStringToInteger(
      extrusion);
 }
 this.setExtrusion_string(
      splitStringToSpecific(extrusion));
 this.extrusion = extrusionResult;
}


public Integer[] getExtrusion() {
 return extrusion;
}


public void setIntrusion(String[] intrusion
     ) {
 Integer[] intrusionResult=null;
 if (intrusion != null) {
 intrusionResult=splitStringToInteger(
      intrusion);
 }
 this.setIntrusion_string(
      splitStringToSpecific(intrusion));
 this.intrusion = intrusionResult;
}


public Integer[] getIntrusion() {
 return intrusion;
}


public void setMesial_rotation(String[]
     mesial_rotation) {
 Integer[] mesial_rotationResult=null;
 if (mesial_rotation != null) {
 mesial_rotationResult=splitStringToInteger
      (mesial_rotation);
 }
 this.setMesial_rotation_string(
      splitStringToSpecific(mesial_rotation)
      );
 this.mesial_rotation =
      mesial_rotationResult;
}


public Integer[] getMesial_rotation() {
 return mesial_rotation;
}


public void setDistal_rotation(String[]
     distal_rotation) {
 Integer[] distal_rotationResult=null;
 if (distal_rotation != null) {
 distal_rotationResult=splitStringToInteger
      (distal_rotation);
 }
 this.setDistal_rotation_string(
      splitStringToSpecific(distal_rotation)
      );
 this.distal_rotation =
      distal_rotationResult;
}


public Integer[] getDistal_rotation() {
 return distal_rotation;
}
```

```
public void setRotation(String[] rotation)
    {
 Integer[] rotationResult=null;
 if (rotation != null) {
 rotationResult=splitStringToInteger(
     rotation);
 }
 this.setRotation_string(
     splitStringToSpecific(rotation));
 this.rotation = rotationResult;
}


public Integer[] getRotation() {
 return rotation;
}


public void setPostcore_crown(String[]
    postcore_crown) {
 Integer[] postcore_crownResult=null;

 if (postcore_crown != null ) {
 postcore_crownResult=splitStringToInteger(
     postcore_crown);
 }
 this.setPostcore_crown_string(
     splitStringToSpecific(postcore_crown))
     ;
 this.postcore_crown = postcore_crownResult
     ;
}


public Integer[] getPostcore_crown() {
 return postcore_crown;
}


public void setRootcanal_treatment(String[]
     rootcanal_treatment) {
 Integer[] rootcanal_treatmentResult=null;

 if (rootcanal_treatment != null) {
 rootcanal_treatmentResult=
     splitStringToInteger(
     rootcanal_treatment);
 }
 this.setRootcanal_treatment_string(
     splitStringToSpecific(
     rootcanal_treatment));
 this.rootcanal_treatment =
     rootcanal_treatmentResult;
}


public Integer[] getRootcanal_treatment() {
 return rootcanal_treatment;
}


public void setPitfissure_sealants(String[]
     pitfissure_sealants) {
 Integer[] pitfissure_sealantsResult=null;

 if (pitfissure_sealants != null ) {
 pitfissure_sealantsResult=
     splitStringToInteger(
     pitfissure_sealants);
 }
 this.setPitfissure_sealants_string(
     splitStringToSpecific(
     pitfissure_sealants));
 this.pitfissure_sealants =
     pitfissure_sealantsResult;
}


public Integer[] getPitfissure_sealants() {
 return pitfissure_sealants;
}


public void setExtracted(String[] extracted
    ) {
 Integer[] extractedResult=null;

 if (extracted != null) {
 extractedResult=splitStringToInteger(
     extracted);
 }
 this.setExtracted_string(
     splitStringToSpecific(extracted));
 this.extracted = extractedResult;
}


public Integer[] getExtracted() {
 return extracted;
}


public void setMissing(String[] missing) {
 Integer[] missingResult=null;

 if (missing != null ) {
 missingResult=splitStringToInteger(missing
     );
 }
 this.setMissing_string(
     splitStringToSpecific(missing));
 this.missing = missingResult;
}


public Integer[] getMissing() {
 return missing;
}


public void setUnerupted(String[] unerupted
    ) {
 Integer[] uneruptedResult=null;


 if (unerupted != null ) {
 uneruptedResult=splitStringToInteger(
     unerupted);
 }
 this.setUnerupted_string(
     splitStringToSpecific(unerupted));
 this.unerupted = uneruptedResult;
}


public Integer[] getUnerupted() {
 return unerupted;
}


public void setImpacted(String[] impacted)
    {
 Integer[] impactedResult=null;

 if (impacted != null ) {
 impactedResult=splitStringToInteger(
     impacted);
 }

 this.setImpacted_string(
     splitStringToSpecific(impacted));
 this.impacted = impactedResult;
}


public Integer[] getImpacted() {
 return impacted;
}


public void setPorcelain_crown(String[]
    porcelain_crown) {
 Integer[] porcelain_crownResult=null;

 if (porcelain_crown != null) {
 porcelain_crownResult=splitStringToInteger
     (porcelain_crown);
 }
 this.setPorcelain_crown_string(
     splitStringToSpecific(porcelain_crown)
     );
 this.porcelain_crown =
     porcelain_crownResult;
}


public Integer[] getPorcelain_crown() {
 return porcelain_crown;
}
```

```
public void setVersion(int version) {
 this.version = version;
}


public int getVersion() {
 return version;
}


public void setUpdated_by(String updated_by
     ) {
 this.updated_by = updated_by;
}


public String getUpdated_by() {
 return updated_by;
}


public void setUpdated_time(String
     updated_time) {
 this.updated_time = updated_time;
}


public String getUpdated_time() {
 return updated_time;
}


public void setIs_current(String is_current
     ) {
 this.is_current = is_current;
}


public String getIs_current() {
 return is_current;
}


public void setApproved(String approved) {
 this.approved = approved;
}


public String getApproved() {
 return approved;
}


public void setApproved_by(String
     approved_by) {
 this.approved_by = approved_by;
}


public String getApproved_by() {
 return approved_by;
}


public void setApproved_date(String
     approved_date) {
 this.approved_date = approved_date;
}


public String getApproved_date() {
 return approved_date;
}


public void setApprovated_time(String
     approvated_time) {
 this.approvated_time = approvated_time;
}


public String getApprovated_time() {
 return approvated_time;
}


public void setAcrylic_crown(String[]
     acrylic_crown) {
 Integer[] acrylic_crownResult=null;

 if (acrylic_crown != null) {
 acrylic_crownResult=splitStringToInteger(
     acrylic_crown);
 }
 this.setAcrylic_crown_string(
     splitStringToSpecific(acrylic_crown));
 this.acrylic_crown = acrylic_crownResult;
}


public Integer[] getAcrylic_crown() {
 return acrylic_crown;
}


public void setMetal_crown(String[]
     metal_crown) {
 Integer[] metal_crownResult=null;

 if (metal_crown != null ) {
 metal_crownResult=splitStringToInteger(
     metal_crown);
 }
 this.setMetal_crown_string(
     splitStringToSpecific(metal_crown));
 this.metal_crown = metal_crownResult;
}


public Integer[] getMetal_crown() {
 return metal_crown;
}


public void setPorcelain_infused(String[]
     porcelain_infused) {
 Integer[] porcelain_infusedResult=null;

 if (porcelain_infused != null ) {
 porcelain_infusedResult=
     splitStringToInteger(porcelain_infused
     );
 }

 this.setPorcelain_infused_string(
     splitStringToSpecific(
     porcelain_infused));
 this.porcelain_infused =
     porcelain_infusedResult;
}


public Integer[] getPorcelain_infused() {
 return porcelain_infused;
}


public void setFixed_bridge(String[]
     fixed_bridge) {
 Integer[] fixed_bridgeResult=null;

 if (fixed_bridge != null) {
 fixed_bridgeResult=splitStringToInteger(
     fixed_bridge);
 }

 this.setFixed_bridge_string(
     splitStringToSpecific(fixed_bridge));
 this.fixed_bridge = fixed_bridgeResult;
}


public Integer[] getFixed_bridge() {
 return fixed_bridge;
}


public void setUpdated_date(String
     updated_date) {
 this.updated_date = updated_date;
}


public String getUpdated_date() {
 return updated_date;
}

public void setRecurrentcaries_string(
     String[] recurrentcaries_string) {
 this.recurrentcaries_string =
     recurrentcaries_string;
}

public String[] getRecurrentcaries_string()
```

```
    {
 return recurrentcaries_string;
}

public void setRestoration_string(String[]
     restoration_string) {
 this.restoration_string =
      restoration_string;
}

public String[] getRestoration_string() {
 return restoration_string;
}

public void
     setRemovable_partial_denture_string(
  String[] removable_partial_denture_string
      ) {
 this.removable_partial_denture_string =
      removable_partial_denture_string;
}

public String[]
     getRemovable_partial_denture_string() {
 return removable_partial_denture_string;
}

public void setExtrusion_string(String[]
     extrusion_string) {
 this.extrusion_string = extrusion_string;
}

public String[] getExtrusion_string() {
 return extrusion_string;
}

public void setIntrusion_string(String[]
     intrusion_string) {
 this.intrusion_string = intrusion_string;
}

public String[] getIntrusion_string() {
 return intrusion_string;
}

public void setMesial_rotation_string(
     String[] mesial_rotation_string) {
 this.mesial_rotation_string =
      mesial_rotation_string;
}

public String[] getMesial_rotation_string()
      {
 return mesial_rotation_string;
}

public void setDistal_rotation_string(
     String[] distal_rotation_string) {
 this.distal_rotation_string =
      distal_rotation_string;
}

public String[] getDistal_rotation_string()
      {
 return distal_rotation_string;
}

public void setRotation_string(String[]
     rotation_string) {
 this.rotation_string = rotation_string;
}

public String[] getRotation_string() {
 return rotation_string;
}

public void setPostcore_crown_string(String
     [] postcore_crown_string) {
 this.postcore_crown_string =
      postcore_crown_string;
}

public String[] getPostcore_crown_string()
     {
 return postcore_crown_string;
}

public void setRootcanal_treatment_string(
  String[] rootcanal_treatment_string) {
 this.rootcanal_treatment_string =
      rootcanal_treatment_string;
}

public String[]
     getRootcanal_treatment_string() {
 return rootcanal_treatment_string;
}

public void setPitfissure_sealants_string(
  String[] pitfissure_sealants_string) {
 this.pitfissure_sealants_string =
      pitfissure_sealants_string;
}

public String[]
     getPitfissure_sealants_string() {
 return pitfissure_sealants_string;
}

public void setExtracted_string(String[]
     extracted_string) {
 this.extracted_string = extracted_string;
}

public String[] getExtracted_string() {
 return extracted_string;
}

public void setMissing_string(String[]
     missing_string) {
 this.missing_string = missing_string;
}
public String[] getMissing_string() {
 return missing_string;
}

public void setUnerupted_string(String[]
     unerupted_string) {
 this.unerupted_string = unerupted_string;
}

public String[] getUnerupted_string() {
 return unerupted_string;
}

public void setImpacted_string(String[]
     impacted_string) {
 this.impacted_string = impacted_string;
}

public String[] getImpacted_string() {
 return impacted_string;
}

public void setPorcelain_crown_string(
     String[] porcelain_crown_string) {
 this.porcelain_crown_string =
      porcelain_crown_string;
}

public String[] getPorcelain_crown_string()
      {
 return porcelain_crown_string;
}

public void setAcrylic_crown_string(String
     [] acrylic_crown_string) {
 this.acrylic_crown_string =
      acrylic_crown_string;
}

public String[] getAcrylic_crown_string() {
 return acrylic_crown_string;
}

public void setMetal_crown_string(String[]
     metal_crown_string) {
 this.metal_crown_string =
      metal_crown_string;
}

public String[] getMetal_crown_string() {
 return metal_crown_string;
}

public void setPorcelain_infused_string(
     String[] porcelain_infused_string) {
 this.porcelain_infused_string =
      porcelain_infused_string;
}

public String[] getPorcelain_infused_string
     () {
 return porcelain_infused_string;
 }

 public void setFixed_bridge_string(String[]
      fixed_bridge_string) {
  this.fixed_bridge_string =
      fixed_bridge_string;
 }

 public String[] getFixed_bridge_string() {
  return fixed_bridge_string;
 }

 public void setCaries_string(String[]
      caries_string) {
  this.caries_string = caries_string;
 }

 public String[] getCaries_string() {
  return caries_string;
 }
```

```
 public void setComplete_denture(String
      complete_denture) {
  this.complete_denture = complete_denture;
 }

 public String getComplete_denture() {
  return complete_denture;
 }

 public void setSingle_denture(String
      single_denture) {
  this.single_denture = single_denture;
 }

 public String getSingle_denture() {
  return single_denture;
 }


}
```

```
package org.dentist.version.three.form;

import java.util.ArrayList;

public class RecurrentStatus {

 private int recurrent_id;
 private int patient_id;
 private Integer[] distal_recurrent;
 private Integer[] buccal_recurrent;
 private Integer[] lingual_recurrent;
 private Integer[] mesial_recurrent;
 private Integer[] occlusal_recurrent;
 private String[]
      distal_restorable_recurrent;
 private String[]
      buccal_restorable_recurrent;
 private String[]
      lingual_restorable_recurrent;
 private String[]
      mesial_restorable_recurrent;
 private String[]
      occlusal_restorable_recurrent;
 private int version;
 private String updated_by;
 private String updated_date;
 private String updated_time;

 private String[] distal_recurrent_string;
 private String[] buccal_recurrent_string;
 private String[] lingual_recurrent_string;
 private String[] mesial_recurrent_string;
 private String[] occlusal_recurrent_string;


 private Integer[] splitStringToInteger(
      String[] temp){
  Integer[] result= new Integer[temp.length
       ];
  String tempResult= "";
  for(int i=0; i<temp.length;i++){
   tempResult=temp[i];
   if(temp[i].indexOf("{")!=-1 || temp[i].
        indexOf("}")!=-1){
     if(temp[i].indexOf("{")!=-1)
      tempResult=tempResult.replace("{", "").
           trim();
      if(temp[i].indexOf("}")!=-1)
      tempResult=tempResult.replace("}", "").
           trim();

   }
   System.out.println("Check: "+ tempResult)
        ;
   result[i]=Integer.parseInt(tempResult);
  }
  return result;
 }

 private String[] splitStringToSpecific(
      String[] temp){
  ArrayList<String> tempResult= new
       ArrayList<String>();
  String tempAr= "";
  String[] result=null;
  if(temp!=null){
  for(int i=0; i<temp.length;i++){
   if(temp[i].length()>2)
   tempResult.add(temp[i]);
```

```
  }
  result= new String[tempResult.size()];
  for(int j=0;j<tempResult.size();j++){
   tempAr=tempResult.get(j);

    if(tempResult.get(j).indexOf("{")!=-1 ||
         tempResult.get(j).indexOf("}")!=-1){
     if(tempResult.get(j).indexOf("{")!=-1)
      tempAr=tempAr.replace("{", "").trim();
      if(tempResult.get(j).indexOf("}")!=-1)
       tempAr=tempAr.replace("}", "").trim();

     System.out.println("Check: "+ tempAr +"
          noms");
    }
    result[j]=tempAr;
   }
  }
  else{
   result=new String[1];
   result[0]="-1";
  }

  return result;
 }

 public void setValuesNeeded(String name,
      String value){
  String[] partial=null;
  if(name.equals("distal_recurrent")){
   partial=value.split(",");
   this.setDistal_recurrent(partial);
  }
  else if(name.equals("buccal_recurrent")){
   partial=value.split(",");
   this.setBuccal_recurrent(partial);
  }
  else if(name.equals("lingual_recurrent")){
   partial=value.split(",");
   this.setLingual_recurrent(partial);
  }
  else if(name.equals("mesial_recurrent")){
   partial=value.split(",");
   this.setMesial_recurrent(partial);
  }
  else if(name.equals("occlusal_recurrent"))
       {
   partial=value.split(",");
   this.setOcclusal_recurrent(partial);
  }
  else if(name.equals("
       distal_restorable_recurrent")){
   partial=value.split(",");
   this.setDistal_restorable_recurrent(
        partial);
  }
  else if(name.equals("
       buccal_restorable_recurrent")){
   partial=value.split(",");
   this.setBuccal_restorable_recurrent(
        partial);
  }
  else if(name.equals("
       lingual_restorable_recurrent")){
   partial=value.split(",");
   this.setLingual_restorable_recurrent(
        partial);
  }
  else if(name.equals("
      mesial_restorable_recurrent")){
  partial=value.split(",");
  this.setMesial_restorable_recurrent(
       partial);
 }
 else if(name.equals("
      occlusal_restorable_recurrent")){
  partial=value.split(",");
  this.setOcclusal_restorable_recurrent(
       partial);
 }
}

public void setrecurrent_id(int
     recurrent_id) {
 this.recurrent_id = recurrent_id;
}
public int getrecurrent_id() {
 return recurrent_id;
}
public void setPatient_id(int patient_id) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setDistal_recurrent(String[]
     distal_recurrent) {
 Integer[] distal_recurrentResult=null;
 if (distal_recurrent != null) {
 distal_recurrentResult=
      splitStringToInteger(distal_recurrent)
      ;
 }
 this.setDistal_recurrent_string(
      splitStringToSpecific(distal_recurrent
      ));
 this.distal_recurrent =
      distal_recurrentResult;
}
public Integer[] getDistal_recurrent() {
 return distal_recurrent;
}

public void setBuccal_recurrent(String[]
     buccal_recurrent) {
 Integer[] buccal_recurrentResult=null;
 if (buccal_recurrent != null) {
 buccal_recurrentResult=
      splitStringToInteger(buccal_recurrent)
      ;
 }
 this.setBuccal_recurrent_string(
      splitStringToSpecific(buccal_recurrent
      ));
 this.buccal_recurrent =
      buccal_recurrentResult;
}

public Integer[] getBuccal_recurrent() {
 return buccal_recurrent;
}

public void setLingual_recurrent(String[]
     lingual_recurrent) {
 Integer[] lingual_recurrentResult=null;
 if (lingual_recurrent != null ) {
 lingual_recurrentResult=
      splitStringToInteger(lingual_recurrent
      );
 }
 this.setLingual_recurrent_string(
      splitStringToSpecific(
      lingual_recurrent));
 this.lingual_recurrent =
      lingual_recurrentResult;
}

public Integer[] getLingual_recurrent() {
 return lingual_recurrent;
}

public void setMesial_recurrent(String[]
     mesial_recurrent) {
 Integer[] mesial_recurrentResult=null;
 if (mesial_recurrent != null ) {
 mesial_recurrentResult=
      splitStringToInteger(mesial_recurrent)
      ;
 }
 this.setMesial_recurrent_string(
      splitStringToSpecific(mesial_recurrent
      ));
 this.mesial_recurrent =
      mesial_recurrentResult;
}

public Integer[] getMesial_recurrent() {
 return mesial_recurrent;
}

public void setOcclusal_recurrent(String[]
     occlusal_recurrent) {
 Integer[] occlusal_recurrentResult=null;
 if (occlusal_recurrent != null ) {
 occlusal_recurrentResult=
      splitStringToInteger(
      occlusal_recurrent);
 }
 this.setOcclusal_recurrent_string(
      splitStringToSpecific(
      occlusal_recurrent));
 this.occlusal_recurrent =
      occlusal_recurrentResult;
}

public Integer[] getOcclusal_recurrent() {
 return occlusal_recurrent;
}

public void setDistal_restorable_recurrent(
     String[] distal_restorable_recurrent) {
 String[]
      distal_restorable_recurrentsResult=
      null;
 if (distal_restorable_recurrent != null )
      {
  distal_restorable_recurrentsResult=
       splitStringToSpecific(
       distal_restorable_recurrent);
 }
 this.distal_restorable_recurrent =
      distal_restorable_recurrentsResult;
}

public String[]
     getDistal_restorable_recurrent() {
 return distal_restorable_recurrent;
}

public void setBuccal_restorable_recurrent(
     String[] buccal_restorable_recurrent) {
 String[] buccal_restorable_recurrentResult
      =null;
 if (buccal_restorable_recurrent != null )
      {
  buccal_restorable_recurrentResult=
       splitStringToSpecific(
       buccal_restorable_recurrent);
 }
 this.buccal_restorable_recurrent =
      buccal_restorable_recurrentResult;
}

public String[]
     getBuccal_restorable_recurrent() {
 return buccal_restorable_recurrent;
}

public void setLingual_restorable_recurrent
     (String[] lingual_restorable_recurrent)
      {
 String[]
      lingual_restorable_recurrentResult=
      null;
 if (lingual_restorable_recurrent != null )
       {
  lingual_restorable_recurrentResult=
       splitStringToSpecific(
       lingual_restorable_recurrent);
 }
 this.lingual_restorable_recurrent =
      lingual_restorable_recurrentResult;
}

public String[]
     getLingual_restorable_recurrent() {
 return lingual_restorable_recurrent;
}

public void setMesial_restorable_recurrent(
     String[] mesial_restorable_recurrent) {
 String[] mesial_restorable_recurrentResult
      =null;
  if (mesial_restorable_recurrent != null) {
   mesial_restorable_recurrentResult=
        splitStringToSpecific(
        mesial_restorable_recurrent);
  }
  this.mesial_restorable_recurrent =
       mesial_restorable_recurrentResult;
 }

 public String[]
      getMesial_restorable_recurrent() {
  return mesial_restorable_recurrent;
 }

 public void
      setOcclusal_restorable_recurrent(String
      [] occlusal_restorable_recurrent) {
  String[]
       occlusal_restorable_recurrentResult=
       null;
  if (occlusal_restorable_recurrent != null)
        {
   occlusal_restorable_recurrentResult=
        splitStringToSpecific(
        occlusal_restorable_recurrent);
  }
  this.occlusal_restorable_recurrent =
       occlusal_restorable_recurrentResult;
 }

 public String[]
      getOcclusal_restorable_recurrent() {
  return occlusal_restorable_recurrent;
 }

 public void setVersion(int version) {
  this.version = version;
 }

 public int getVersion() {
  return version;
 }

 public void setUpdated_by(String updated_by
      ) {
  this.updated_by = updated_by;
 }

 public String getUpdated_by() {
  return updated_by;
 }

 public void setUpdated_date(String
      updated_date) {
  this.updated_date = updated_date;
 }

 public String getUpdated_date() {
  return updated_date;
 }

 public void setUpdated_time(String
      updated_time) {
  this.updated_time = updated_time;
 }

 public String getUpdated_time() {
  return updated_time;
 }

 public void setDistal_recurrent_string(
      String[] distal_recurrent_string) {
  this.distal_recurrent_string =
       distal_recurrent_string;
 }

 public String[] getDistal_recurrent_string
      () {
  return distal_recurrent_string;
 }

 public void setBuccal_recurrent_string(
      String[] buccal_recurrent_string) {
  this.buccal_recurrent_string =
       buccal_recurrent_string;
 }

 public String[] getBuccal_recurrent_string
      () {
  return buccal_recurrent_string;
 }

 public void setLingual_recurrent_string(
      String[] lingual_recurrent_string) {
  this.lingual_recurrent_string =
       lingual_recurrent_string;
 }

 public String[] getLingual_recurrent_string
      () {
  return lingual_recurrent_string;
 }

 public void setOcclusal_recurrent_string(
      String[] occlusal_recurrent_string) {
  this.occlusal_recurrent_string =
       occlusal_recurrent_string;
 }

 public String[]
      getOcclusal_recurrent_string() {
  return occlusal_recurrent_string;
 }

 public void setMesial_recurrent_string(
      String[] mesial_recurrent_string) {
  this.mesial_recurrent_string =
       mesial_recurrent_string;
 }

 public String[] getMesial_recurrent_string
      () {
  return mesial_recurrent_string;
 }



}
```

```
package org.dentist.version.three.form;

import java.util.ArrayList;

public class RestorationStatus {

 private int restoration_id;
 private int patient_id;
 private Integer[] distal_restoration;
 private Integer[] buccal_restoration;
 private Integer[] lingual_restoration;
 private Integer[] mesial_restoration;
 private Integer[] occlusal_restoration;
 private String[]
      distal_restorable_restoration;
 private String[]
      buccal_restorable_restoration;
 private String[]
      lingual_restorable_restoration;
 private String[]
      mesial_restorable_restoration;
 private String[]
      occlusal_restorable_restoration;
 private int version;
 private String updated_by;
 private String updated_date;
 private String updated_time;

 private String[] distal_restoration_string;
 private String[] buccal_restoration_string;
 private String[] lingual_restoration_string
      ;
 private String[] mesial_restoration_string;
 private String[]
      occlusal_restoration_string;

 private Integer[] splitStringToInteger(
      String[] temp){
  Integer[] result= new Integer[temp.length
       ];
  String tempResult= "";
  for(int i=0; i<temp.length;i++){
   tempResult=temp[i];
   if(temp[i].indexOf("{")!=-1 || temp[i].
        indexOf("}")!=-1){
    if(temp[i].indexOf("{")!=-1)
     tempResult=tempResult.replace("{", "").
          trim();
     if(temp[i].indexOf("}")!=-1)
     tempResult=tempResult.replace("}", "").
```

```
        trim();

   System.out.println("Check: "+ tempResult
        );
  }

  result[i]=Integer.parseInt(tempResult);
 }
 return result;
}

private String[] splitStringToSpecific(
     String[] temp){
 ArrayList<String> tempResult= new
      ArrayList<String>();
 String tempAr= "";
 String[] result=null;
 if(temp!=null){
 for(int i=0; i<temp.length;i++){
  if(temp[i].length()>2)
  tempResult.add(temp[i]);
 }
 result= new String[tempResult.size()];
 for(int j=0;j<tempResult.size();j++){
  tempAr=tempResult.get(j);

  if(tempResult.get(j).indexOf("{")!=-1 ||
       tempResult.get(j).indexOf("}")!=-1){
   if(tempResult.get(j).indexOf("{")!=-1)
    tempAr=tempAr.replace("{", "").trim();
    if(tempResult.get(j).indexOf("}")!=-1)
     tempAr=tempAr.replace("}", "").trim();

   System.out.println("Check: "+ tempAr +"
        noms");
  }
  result[j]=tempAr;
 }
 }
 else{
  result=new String[1];
  result[0]="-1";
  }

 return result;
}



public void setValuesNeeded(String name,
     String value){
 String[] partial=null;
 if(name.equals("distal_restoration")){
  partial=value.split(",");
  this.setDistal_restoration(partial);
 }
 else if(name.equals("buccal_restoration"))
      {
  partial=value.split(",");
  this.setBuccal_restoration(partial);
 }
 else if(name.equals("lingual_restoration")
      ){
  partial=value.split(",");
  this.setLingual_restoration(partial);
 }
 else if(name.equals("mesial_restoration"))
      {
  partial=value.split(",");
  this.setMesial_restoration(partial);
 }
 else if(name.equals("occlusal_restoration
      ")){
  partial=value.split(",");
  this.setOcclusal_restoration(partial);
 }
 else if(name.equals("
      distal_restorable_restoration")){
  partial=value.split(",");
  this.setDistal_restorable_restoration(
       partial);
 }
 else if(name.equals("
      buccal_restorable_restoration")){
  partial=value.split(",");
  this.setBuccal_restorable_restoration(
       partial);
 }
 else if(name.equals("
      lingual_restorable_restoration")){
  partial=value.split(",");
  this.setLingual_restorable_restoration(
       partial);
 }
 else if(name.equals("
      mesial_restorable_restoration")){
  partial=value.split(",");
  this.setMesial_restorable_restoration(
       partial);
 }
 else if(name.equals("
      occlusal_restorable_restoration")){
  partial=value.split(",");
  this.setOcclusal_restorable_restoration(
       partial);
 }
}


public void setrestoration_id(int
     restoration_id) {
 this.restoration_id = restoration_id;
}
public int getrestoration_id() {
 return restoration_id;
}
public void setPatient_id(int patient_id) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setDistal_restoration(String[]
     distal_restoration) {
 Integer[] distal_restorationResult=null;
 if (distal_restoration != null) {
 distal_restorationResult=
      splitStringToInteger(
      distal_restoration);
 }
 this.setDistal_restoration_string(
      splitStringToSpecific(
      distal_restoration));
 this.distal_restoration =
      distal_restorationResult;
}
public Integer[] getDistal_restoration() {
 return distal_restoration;
}

public void setBuccal_restoration(String[]
     buccal_restoration) {
 Integer[] buccal_restorationResult=null;
 if (buccal_restoration != null) {
 buccal_restorationResult=
      splitStringToInteger(
      buccal_restoration);
 }
 this.setBuccal_restoration_string(
      splitStringToSpecific(
      buccal_restoration));
 this.buccal_restoration =
      buccal_restorationResult;
}

public Integer[] getBuccal_restoration() {
 return buccal_restoration;
}

public void setLingual_restoration(String[]
      lingual_restoration) {
 Integer[] lingual_restorationResult=null;
 if (lingual_restoration != null ) {
 lingual_restorationResult=
      splitStringToInteger(
      lingual_restoration);
 }
 this.setLingual_restoration_string(
      splitStringToSpecific(
      lingual_restoration));
 this.lingual_restoration =
      lingual_restorationResult;
}

public Integer[] getLingual_restoration() {
 return lingual_restoration;
}

public void setMesial_restoration(String[]
     mesial_restoration) {
 Integer[] mesial_restorationResult=null;
 if (mesial_restoration != null ) {
 mesial_restorationResult=
      splitStringToInteger(
      mesial_restoration);
 }
 this.setMesial_restoration_string(
      splitStringToSpecific(
      mesial_restoration));
 this.mesial_restoration =
      mesial_restorationResult;
}

public Integer[] getMesial_restoration() {
 return mesial_restoration;
}

public void setOcclusal_restoration(String
     [] occlusal_restoration) {
 Integer[] occlusal_restorationResult=null;
 if (occlusal_restoration != null ) {
 occlusal_restorationResult=
      splitStringToInteger(
      occlusal_restoration);
 }
 this.setOcclusal_restoration_string(
      splitStringToSpecific(
      occlusal_restoration));
 this.occlusal_restoration =
      occlusal_restorationResult;
}

public Integer[] getOcclusal_restoration()
     {
 return occlusal_restoration;
}

public void
     setDistal_restorable_restoration(String
     [] distal_restorable_restoration) {
 String[]
      distal_restorable_restorationsResult=
      null;
 if (distal_restorable_restoration != null
      ) {
  distal_restorable_restorationsResult=
       splitStringToSpecific(
       distal_restorable_restoration);
 }
 this.distal_restorable_restoration =
      distal_restorable_restorationsResult;
}

public String[]
     getDistal_restorable_restoration() {
 return distal_restorable_restoration;
}

public void
     setBuccal_restorable_restoration(String
     [] buccal_restorable_restoration) {
 String[]
      buccal_restorable_restorationResult=
      null;
 if (buccal_restorable_restoration != null
      ) {
  buccal_restorable_restorationResult=
       splitStringToSpecific(
       buccal_restorable_restoration);
 }
 this.buccal_restorable_restoration =
      buccal_restorable_restorationResult;
}

public String[]
     getBuccal_restorable_restoration() {
 return buccal_restorable_restoration;
}

public void
     setLingual_restorable_restoration(
     String[] lingual_restorable_restoration
     ) {
 String[]
      lingual_restorable_restorationResult=
      null;
 if (lingual_restorable_restoration != null
       ) {
  lingual_restorable_restorationResult=
       splitStringToSpecific(
       lingual_restorable_restoration);
 }
 this.lingual_restorable_restoration =
      lingual_restorable_restorationResult;
}

public String[]
     getLingual_restorable_restoration() {
 return lingual_restorable_restoration;
}

public void
     setMesial_restorable_restoration(String
     [] mesial_restorable_restoration) {
 String[]
      mesial_restorable_restorationResult=
      null;
 if (mesial_restorable_restoration != null)
       {
  mesial_restorable_restorationResult=
       splitStringToSpecific(
       mesial_restorable_restoration);
 }
 this.mesial_restorable_restoration =
      mesial_restorable_restorationResult;
}

public String[]
     getMesial_restorable_restoration() {
 return mesial_restorable_restoration;
}

public void
     setOcclusal_restorable_restoration(
     String[]
     occlusal_restorable_restoration) {
 String[]
      occlusal_restorable_restorationResult=
      null;
 if (occlusal_restorable_restoration !=
      null) {
  occlusal_restorable_restorationResult=
       splitStringToSpecific(
       occlusal_restorable_restoration);
 }
 this.occlusal_restorable_restoration =
      occlusal_restorable_restorationResult;
}

public String[]
     getOcclusal_restorable_restoration() {
 return occlusal_restorable_restoration;
}

public void setVersion(int version) {
 this.version = version;
}

public int getVersion() {
 return version;
}

public void setUpdated_by(String updated_by
     ) {
 this.updated_by = updated_by;
}

public String getUpdated_by() {
 return updated_by;
}

public void setUpdated_date(String
     updated_date) {
 this.updated_date = updated_date;
}

public String getUpdated_date() {
 return updated_date;
}

public void setUpdated_time(String
     updated_time) {
 this.updated_time = updated_time;
}

public String getUpdated_time() {
 return updated_time;
}

public void setDistal_restoration_string(
     String[] distal_restoration_string) {
 this.distal_restoration_string =
      distal_restoration_string;
}

public String[]
     getDistal_restoration_string() {
 return distal_restoration_string;
 }

 public void setBuccal_restoration_string(
     String[] buccal_restoration_string) {
  this.buccal_restoration_string =
      buccal_restoration_string;
 }

 public String[]
     getBuccal_restoration_string() {
  return buccal_restoration_string;
 }

 public void setLingual_restoration_string(
     String[] lingual_restoration_string) {
  this.lingual_restoration_string =
      lingual_restoration_string;
 }

 public String[]
     getLingual_restoration_string() {
  return lingual_restoration_string;
 }

 public void setOcclusal_restoration_string(
     String[] occlusal_restoration_string) {
  this.occlusal_restoration_string =
      occlusal_restoration_string;
 }

 public String[]
     getOcclusal_restoration_string() {
  return occlusal_restoration_string;
 }

 public void setMesial_restoration_string(
     String[] mesial_restoration_string) {
  this.mesial_restoration_string =
      mesial_restoration_string;
 }

 public String[]
     getMesial_restoration_string() {
  return mesial_restoration_string;
 }



}
```

```
package org.dentist.version.three.form;

import java.util.ArrayList;

public class ServiceNeeded {

 private int serviceneeded_id;
 private int patient_id;
 private Integer[] class_1;
 private Integer[] class_2;
 private Integer[] class_3;
 private Integer[] class_4;
 private Integer[] class_5;
 private Integer[] onlay;
 private Integer[] extraction;
 private Integer[] odontectomy;
 private Integer[] special_case;
 private Integer[] pulp_sedation;
 private Integer[] crown_recementation;
 private Integer[] filling_service;
 private Integer[] laminated;
 private Integer[] single_crown;
 private Integer[] bridge_service;
 private Integer[] anterior;
 private Integer[] posterior;
 private Integer[] ortho_endo;
 private String periodontics;
 private String surgery;
 private String emergency_treatment;
 private String prosthodontics;
 private String updated_by;
 private String updated_date;
 private String updated_time;
 private int version;
 private String notes;
 private String is_current;
 /*
  * String[] version
  */
 private String[] class_1_string;
 private String[] class_2_string;
 private String[] class_3_string;
 private String[] class_4_string;
 private String[] class_5_string;
 private String[] onlay_string;
 private String[] extraction_string;
 private String[] odontectomy_string;
 private String[] special_case_string;
 private String[] pulp_sedation_string;
 private String[] crown_recementation_string
     ;
 private String[] filling_service_string;
 private String[] laminated_string;
 private String[] single_crown_string;
 private String[] bridge_service_string;
 private String[] anterior_string;
 private String[] posterior_string;
 private String[] ortho_endo_string;

 /*
  * private functions
  */

 private Integer[] splitStringToInteger(
     String[] temp){
  Integer[] result= new Integer[temp.length
      ];
  String tempResult= "";
  for(int i=0; i<temp.length;i++){
   tempResult=temp[i];
   if(temp[i].indexOf("{")!=-1 || temp[i].
       indexOf("}")!=-1){
    if(temp[i].indexOf("{")!=-1)
     tempResult=tempResult.replace("{", "").
         trim();
     if(temp[i].indexOf("}")!=-1)
     tempResult=tempResult.replace("}", "").
         trim();

    System.out.println("Check: "+ tempResult
        );
   }

   result[i]=Integer.parseInt(tempResult);
  }
  return result;
 }

 private String[] splitStringToSpecific(
     String[] temp){
  ArrayList<String> tempResult= new
      ArrayList<String>();
  String tempAr= "";
  String[] result=null;
  if(temp!=null){
  for(int i=0; i<temp.length;i++){
   if(temp[i].length()>2)
   tempResult.add(temp[i]);
  }
  result= new String[tempResult.size()];
  for(int j=0;j<tempResult.size();j++){
   tempAr=tempResult.get(j);

   if(tempResult.get(j).indexOf("{")!=-1 ||
       tempResult.get(j).indexOf("}")!=-1){
    if(tempResult.get(j).indexOf("{")!=-1)
     tempAr=tempAr.replace("{", "").trim();
     if(tempResult.get(j).indexOf("}")!=-1)
      tempAr=tempAr.replace("}", "").trim();

    System.out.println("Check: "+ tempAr +"
        noms");
   }
   result[j]=tempAr;
  }
  }
  else{
   result=new String[1];
   result[0]="-1";
  }

  return result;
 }

 public void setValuesNeeded(String name,
     String value){
  String[] partial=null;
  if(name.equals("class_1")){
   partial=value.split(",");
   this.setClass_1(partial);
 }
 else if(name.equals("class_2")){
  partial=value.split(",");
  this.setClass_2(partial);
 }
 else if(name.equals("class_3")){
  partial=value.split(",");
  this.setClass_3(partial);
 }
 else if(name.equals("class_4")){
  partial=value.split(",");
  this.setClass_4(partial);
 }
 else if(name.equals("class_5")){
  partial=value.split(",");
  this.setClass_5(partial);
 }
 else if(name.equals("onlay")){
  partial=value.split(",");
  this.setOnlay(partial);
 }
 else if(name.equals("extraction")){
  partial=value.split(",");
  this.setExtraction(partial);
 }
 else if(name.equals("odontectomy")){
  partial=value.split(",");
  this.setOdontectomy(partial);
 }
 else if(name.equals("special_case")){
  partial=value.split(",");
  this.setSpecial_case(partial);
 }
 else if(name.equals("pulp_sedation")){
  partial=value.split(",");
  this.setPulp_sedation(partial);
 }
 else if(name.equals("crown_recementation")
     ){
  partial=value.split(",");
  this.setCrown_recementation(partial);
 }
 else if(name.equals("filling_service")){
  partial=value.split(",");
  this.setFilling_service(partial);
 }
 else if(name.equals("laminated")){
  partial=value.split(",");
  this.setLaminated(partial);
 }
 else if(name.equals("single_crown")){
  partial=value.split(",");
  this.setSingle_crown(partial);
 }
 else if(name.equals("bridge_service")){
  partial=value.split(",");
  this.setBridge_service(partial);
 }
 else if(name.equals("anterior")){
  partial=value.split(",");
  this.setAnterior(partial);
 }
 else if(name.equals("posterior")){
  partial=value.split(",");
  this.setPosterior(partial);
 }
 else if(name.equals("ortho_endo")){
  partial=value.split(",");
  this.setOrtho_endo(partial);
 }
}


/*
 * Getters AND Setters
 */


public void setServiceneeded_id(int
    serviceneeded_id) {
 this.serviceneeded_id = serviceneeded_id;
}
public int getServiceneeded_id() {
 return serviceneeded_id;
}
public void setPatient_id(int patient_id) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setClass_1(String[] class_1) {

 Integer[] class_1Result=null;
 if (class_1 != null) {
 class_1Result=splitStringToInteger(class_1
     );
 }
 this.setClass_1_string(
     splitStringToSpecific(class_1));
 this.class_1 = class_1Result;
}
public Integer[] getClass_1() {
 return class_1;
}
public void setClass_2(String[] class_2) {

 Integer[] class_2Result=null;
 if (class_2 != null) {
 class_2Result=splitStringToInteger(class_2
     );
 }
 this.setClass_2_string(
     splitStringToSpecific(class_2));

 this.class_2 = class_2Result;
}
public Integer[] getClass_2() {
 return class_2;
}
public void setClass_3(String[] class_3) {
 Integer[] class_3Result=null;
 if (class_3 != null) {
 class_3Result=splitStringToInteger(class_3
     );
 }
 this.setClass_3_string(
     splitStringToSpecific(class_3));

 this.class_3 = class_3Result;
}
public Integer[] getClass_3() {
 return class_3;
}
public void setClass_4(String[] class_4) {

 Integer[] class_4Result=null;
 if (class_4 != null ) {
  class_4Result=splitStringToInteger(
      class_4);
 }
 this.setClass_4_string(
     splitStringToSpecific(class_4));


 this.class_4 = class_4Result;
}
public Integer[] getClass_4() {
 return class_4;
}
public void setClass_5(String[] class_5) {

 Integer[] class_5Result=null;
 if (class_5 != null) {
  class_5Result=splitStringToInteger(
      class_5);
 }
 this.setClass_5_string(
     splitStringToSpecific(class_5));


 this.class_5 = class_5Result;
}
public Integer[] getClass_5() {
 return class_5;
}
public void setOnlay(String[] onlay) {

 Integer[] onlayResult=null;
 if (onlay != null) {
  onlayResult=splitStringToInteger(onlay);
 }
 this.setOnlay_string(splitStringToSpecific
     (onlay));


 this.onlay = onlayResult;
}
public Integer[] getOnlay() {
 return onlay;
}
public void setExtraction(String[]
    extraction) {
```

```
 Integer[] extractionResult=null;
 if (extraction != null ) {
  extractionResult=splitStringToInteger(
       extraction);
 }
 this.setExtraction_string(
      splitStringToSpecific(extraction));


 this.extraction = extractionResult;
}
public Integer[] getExtraction() {
 return extraction;
}
public void setOdontectomy(String[]
     odontectomy) {

 Integer[] odontectomyResult=null;
 if (odontectomy != null) {
  odontectomyResult=splitStringToInteger(
       odontectomy);
 }
 this.setOdontectomy_string(
      splitStringToSpecific(odontectomy));

 this.odontectomy = odontectomyResult;
}
public Integer[] getOdontectomy() {
 return odontectomy;
}
public void setSpecial_case(String[]
     special_case) {


 Integer[] special_caseResult=null;
 if (special_case != null) {
  special_caseResult=splitStringToInteger(
       special_case);
 }
 this.setSpecial_case_string(
      splitStringToSpecific(special_case));



 this.special_case = special_caseResult;
}
public Integer[] getSpecial_case() {
 return special_case;
}
public void setPulp_sedation(String[]
     pulp_sedation) {

 Integer[] pulp_sedationResult=null;
 if (pulp_sedation != null) {
  pulp_sedationResult=splitStringToInteger(
       pulp_sedation);
 }
 this.setPulp_sedation_string(
      splitStringToSpecific(pulp_sedation));

 this.pulp_sedation = pulp_sedationResult;
}
public Integer[] getPulp_sedation() {
 return pulp_sedation;
}
public void setCrown_recementation(String[]
      crown_recementation) {

 Integer[] crown_recementationResult=null;
 if (crown_recementation != null) {
  crown_recementationResult=
       splitStringToInteger(
       crown_recementation);
 }
 this.setCrown_recementation_string(
      splitStringToSpecific(
      crown_recementation));

 this.crown_recementation =
      crown_recementationResult;
}
public Integer[] getCrown_recementation() {
 return crown_recementation;
}
public void setFilling_service(String[]
     filling_service) {

 Integer[] filling_serviceResult=null;
 if (filling_service != null) {
  filling_serviceResult=
       splitStringToInteger(filling_service)
       ;
 }
 this.setFilling_service_string(
      splitStringToSpecific(filling_service)
      );

 this.filling_service =
      filling_serviceResult;
}
public Integer[] getFilling_service() {
 return filling_service;
}
public void setLaminated(String[] laminated
     ) {

 Integer[] laminatedResult=null;
 if (laminated != null) {
  laminatedResult=splitStringToInteger(
       laminated);
 }
 this.setLaminated_string(
      splitStringToSpecific(laminated));

 this.laminated = laminatedResult;
}
public Integer[] getLaminated() {
 return laminated;
}
public void setBridge_service(String[]
     bridge_service) {

 Integer[] bridge_serviceResult=null;
 if (bridge_service != null) {
  bridge_serviceResult=splitStringToInteger
       (bridge_service);
 }
 this.setBridge_service_string(
      splitStringToSpecific(bridge_service))
      ;

 this.bridge_service = bridge_serviceResult
      ;
}
public Integer[] getBridge_service() {
 return bridge_service;
}
public void setSingle_crown(String[]
     single_crown) {

 Integer[] single_crownResult=null;
 if (single_crown != null ) {
  single_crownResult=splitStringToInteger(
       single_crown);
 }
 this.setSingle_crown_string(
      splitStringToSpecific(single_crown));

 this.single_crown = single_crownResult;
}
public Integer[] getSingle_crown() {
 return single_crown;
}
public void setAnterior(String[] anterior)
     {

 Integer[] anteriorResult=null;
 if (anterior != null) {
  anteriorResult=splitStringToInteger(
       anterior);
 }
 this.setAnterior_string(
      splitStringToSpecific(anterior));


 this.anterior = anteriorResult;
}
public Integer[] getAnterior() {
 return anterior;
}
public void setPosterior(String[] posterior
     ) {

 Integer[] posteriorResult=null;
 if (posterior != null) {
  posteriorResult=splitStringToInteger(
       posterior);
 }
 this.setPosterior_string(
      splitStringToSpecific(posterior));
 this.posterior = posteriorResult;
}
public Integer[] getPosterior() {
 return posterior;
}
public void setOrtho_endo(String[]
     ortho_endo) {

 Integer[] ortho_endoResult=null;
 if (ortho_endo != null ) {
  ortho_endoResult=splitStringToInteger(
       ortho_endo);
 }
 this.setOrtho_endo_string(
      splitStringToSpecific(ortho_endo));

 this.ortho_endo = ortho_endoResult;
}
public Integer[] getOrtho_endo() {
 return ortho_endo;
}
public void setPeriodontics(String
     periodontics) {
 this.periodontics = periodontics;
}
public String getPeriodontics() {
 return periodontics;
}
public void setEmergency_treatment(String
     emergency_treatment) {
 this.emergency_treatment =
      emergency_treatment;
}
public String getEmergency_treatment() {
 return emergency_treatment;
}
public void setProsthodontics(String
     prosthodontics) {
 this.prosthodontics = prosthodontics;
}
public String getProsthodontics() {
 return prosthodontics;
}
public void setUpdated_by(String updated_by
     ) {
 this.updated_by = updated_by;
}
public String getUpdated_by() {
 return updated_by;
}
public void setUpdated_date(String
     updated_date) {
 this.updated_date = updated_date;
}
public String getUpdated_date() {
 return updated_date;
}
public void setUpdated_time(String
     updated_time) {
 this.updated_time = updated_time;
}
public String getUpdated_time() {
 return updated_time;
}
public void setVersion(int version) {
 this.version = version;
}
public int getVersion() {
 return version;
}

public void setClass_1_string(String[]
     class_1_string) {
 this.class_1_string = class_1_string;
}

public String[] getClass_1_string() {
 return class_1_string;
}

public void setClass_2_string(String[]
     class_2_string) {
 this.class_2_string = class_2_string;
}

public String[] getClass_2_string() {
 return class_2_string;
}

public void setClass_3_string(String[]
     class_3_string) {
 this.class_3_string = class_3_string;
}

public String[] getClass_3_string() {
 return class_3_string;
}

public void setClass_4_string(String[]
     class_4_string) {
 this.class_4_string = class_4_string;
}

public String[] getClass_4_string() {
 return class_4_string;
}

public void setClass_5_string(String[]
     class_5_string) {
 this.class_5_string = class_5_string;
}

public String[] getClass_5_string() {
 return class_5_string;
}

public void setOnlay_string(String[]
     onlay_string) {
 this.onlay_string = onlay_string;
}

public String[] getOnlay_string() {
 return onlay_string;
}

public void setExtraction_string(String[]
     extraction_string) {
 this.extraction_string = extraction_string
      ;
}

public String[] getExtraction_string() {
 return extraction_string;
}

public void setOdontectomy_string(String[]
     odontectomy_string) {
 this.odontectomy_string =
      odontectomy_string;
}

public String[] getOdontectomy_string() {
 return odontectomy_string;
}

public void setSpecial_case_string(String[]
      special_case_string) {
 this.special_case_string =
      special_case_string;
}

public String[] getSpecial_case_string() {
 return special_case_string;
}

public void setPulp_sedation_string(String
     [] pulp_sedation_string) {
 this.pulp_sedation_string =
      pulp_sedation_string;
}

public String[] getPulp_sedation_string() {
 return pulp_sedation_string;
}

public void setCrown_recementation_string(
  String[] crown_recementation_string) {
 this.crown_recementation_string =
      crown_recementation_string;
}

public String[]
     getCrown_recementation_string() {
 return crown_recementation_string;
}

public void setFilling_service_string(
     String[] filling_service_string) {
 this.filling_service_string =
      filling_service_string;
}

public String[] getFilling_service_string()
      {
```

```
 return filling_service_string;
}

public void setLaminated_string(String[]
     laminated_string) {
 this.laminated_string = laminated_string;
}

public String[] getLaminated_string() {
 return laminated_string;
}

public void setSingle_crown_string(String[]
      single_crown_string) {
 this.single_crown_string =
      single_crown_string;
}

public String[] getSingle_crown_string() {
 return single_crown_string;
}

public void setBridge_service_string(String
     [] bridge_service_string) {
 this.bridge_service_string =
      bridge_service_string;
}

public String[] getBridge_service_string()
     {
 return bridge_service_string;
}

public void setAnterior_string(String[]
     anterior_string) {
 this.anterior_string = anterior_string;
}

public String[] getAnterior_string() {
 return anterior_string;
}

public void setPosterior_string(String[]
     posterior_string) {
 this.posterior_string = posterior_string;
}

public String[] getPosterior_string() {
 return posterior_string;
}

public void setOrtho_endo_string(String[]
     ortho_endo_string) {
 this.ortho_endo_string = ortho_endo_string
     ;
}

public String[] getOrtho_endo_string() {
 return ortho_endo_string;
}

public void setSurgery(String surgery) {
 this.surgery = surgery;
}

public String getSurgery() {
 return surgery;
}

public void setNotes(String notes) {
 this.notes = notes;
}

public String getNotes() {
 return notes;
}

public void setIs_current(String is_current
     ) {
 this.is_current = is_current;
}

public String getIs_current() {
 return is_current;
}




}
```

```
package org.dentist.version.three.mapper;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
     RowMapper;

import org.dentist.version.three.form.
     CariesStatus;

public class CariesStatusMapper implements
     RowMapper<CariesStatus>{

   public CariesStatus mapRow(ResultSet rs,
        int rowNum) throws SQLException {

    CariesStatus cariesStatus= new
         CariesStatus();

    cariesStatus.setCaries_id(rs.getInt("
         caries_id"));
    cariesStatus.setPatient_id(rs.getInt("
         patient_id"));
    cariesStatus.setValuesNeeded("
         distal_caries",rs.getArray("
         distal_caries").toString());
    cariesStatus.setValuesNeeded("
         buccal_caries",rs.getArray("
         buccal_caries").toString());
    cariesStatus.setValuesNeeded("
         lingual_caries",rs.getArray("
         lingual_caries").toString());
    cariesStatus.setValuesNeeded("
         mesial_caries",rs.getArray("
         mesial_caries").toString());
    cariesStatus.setValuesNeeded("
         occlusal_caries",rs.getArray("
         occlusal_caries").toString());
    cariesStatus.setValuesNeeded("
         distal_restorable_caries",rs.
         getArray("distal_restorable_caries")
         .toString());
    cariesStatus.setValuesNeeded("
         buccal_restorable_caries",rs.
         getArray("buccal_restorable_caries")
         .toString());
    cariesStatus.setValuesNeeded("
         lingual_restorable_caries",rs.
         getArray("lingual_restorable_caries
         ").toString());
    cariesStatus.setValuesNeeded("
         mesial_restorable_caries",rs.
         getArray("mesial_restorable_caries")
         .toString());
    cariesStatus.setValuesNeeded("
         occlusal_restorable_caries",rs.
         getArray("occlusal_restorable_caries
         ").toString());
    cariesStatus.setVersion(rs.getInt("
         version"));
    cariesStatus.setUpdated_by(rs.getString
         ("updated_by"));
    cariesStatus.setUpdated_date(rs.
         getString("updated_date"));
    cariesStatus.setUpdated_time(rs.
         getString("updated_time"));

    return cariesStatus;
   }
}
```

```
package org.dentist.version.three.mapper;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
     RowMapper;
import org.dentist.version.three.form.
     DentalChart;

public class DentalChartMapper implements
     RowMapper<DentalChart> {
  public DentalChart mapRow(ResultSet rs,
       int rowNum) {
```

```
  DentalChart dentalChart= new DentalChart()
      ;

  try {
   dentalChart.setDental_chart_id(rs.getInt
       ("dental_chart_id"));

     dentalChart.setPatient_id(rs.getInt("
         patient_id"));
     dentalChart.setClinician_id(rs.getInt("
         clinician_id"));
     dentalChart.setValuesNeeded("caries",rs
         .getArray("caries").toString());
     dentalChart.setValuesNeeded("
         recurrent_caries",rs.getArray("
         recurrent_caries").toString());
     dentalChart.setValuesNeeded("
         restoration",rs.getArray("
         restoration").toString());
     dentalChart.setComplete_denture(rs.
         getString("complete_denture"));
     dentalChart.setSingle_denture(rs.
         getString("single_denture"));
     dentalChart.setValuesNeeded("
         removable_partial_denture",rs.
         getArray("removable_partial_denture
         ").toString());
     dentalChart.setValuesNeeded("extrusion
         ",rs.getArray("extrusion").toString
         ());
     dentalChart.setValuesNeeded("intrusion
         ",rs.getArray("intrusion").toString
         ());
     dentalChart.setValuesNeeded("
         mesial_rotation",rs.getArray("
         mesial_rotation").toString());
     dentalChart.setValuesNeeded("
         distal_rotation",rs.getArray("
         distal_rotation").toString());
     dentalChart.setValuesNeeded("rotation",
         rs.getArray("rotation").toString())
         ;
     dentalChart.setValuesNeeded("
         postcore_crown",rs.getArray("
         postcore_crown").toString());
     dentalChart.setValuesNeeded("
         rootcanal_treatment",rs.getArray("
         rootcanal_treatment").toString());
     dentalChart.setValuesNeeded("
         pitfissure_sealants",rs.getArray("
         pitfissure_sealants").toString());
     dentalChart.setValuesNeeded("extracted
         ",rs.getArray("extracted").toString
         ());
     dentalChart.setValuesNeeded("missing",
         rs.getArray("missing").toString());
     dentalChart.setValuesNeeded("unerupted
         ",rs.getArray("unerupted").toString
         ());
     dentalChart.setValuesNeeded("impacted",
         rs.getArray("impacted").toString())
         ;
     dentalChart.setValuesNeeded("
         porcelain_crown",rs.getArray("
         porcelain_crown").toString());
     dentalChart.setValuesNeeded("
         acrylic_crown",rs.getArray("
         acrylic_crown").toString());
     dentalChart.setValuesNeeded("
         metal_crown",rs.getArray("
         metal_crown").toString());
     dentalChart.setValuesNeeded("
         porcelain_infused",rs.getArray("
         porcelain_infused").toString());
     dentalChart.setValuesNeeded("
         fixed_bridge",rs.getArray("
         fixed_bridge").toString());
     dentalChart.setVersion(rs.getInt("
         version"));
     dentalChart.setUpdated_by(rs.getString
         ("updated_by"));
     dentalChart.setUpdated_date(rs.
         getString("updated_date"));
     dentalChart.setUpdated_time(rs.
         getString("updated_time"));
     dentalChart.setIs_current(rs.getString
         ("is_current"));
     dentalChart.setApproved(rs.getString("
         approved"));
     dentalChart.setApproved_by(rs.getString
         ("approved_by"));
     dentalChart.setApproved_date(rs.
         getString("approved_date"));
     dentalChart.setApprovated_time(rs.
         getString("approved_time"));
  } catch (SQLException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("NO PATIENT RECORD
       FOUND");
   return dentalChart;
  }


      return dentalChart;
    }
}
```

```
package org.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import org.dentist.version.three.form.
    RecurrentStatus;

public class RecurrentStatusMapper
    implements RowMapper<RecurrentStatus>{

   public RecurrentStatus mapRow(ResultSet
       rs, int rowNum) throws SQLException {

    RecurrentStatus recurrentStatus= new
        RecurrentStatus();

    recurrentStatus.setrecurrent_id(rs.
        getInt("recurrent_id"));
    recurrentStatus.setPatient_id(rs.getInt
        ("patient_id"));
    recurrentStatus.setValuesNeeded("
        distal_recurrent",rs.getArray("
        distal_recurrent").toString());
    recurrentStatus.setValuesNeeded("
        buccal_recurrent",rs.getArray("
        buccal_recurrent").toString());
    recurrentStatus.setValuesNeeded("
        lingual_recurrent",rs.getArray("
        lingual_recurrent").toString());
    recurrentStatus.setValuesNeeded("
        mesial_recurrent",rs.getArray("
        mesial_recurrent").toString());
    recurrentStatus.setValuesNeeded("
        occlusal_recurrent",rs.getArray("
        occlusal_recurrent").toString());
    recurrentStatus.setValuesNeeded("
        distal_restorable_recurrent",rs.
        getArray("
        distal_restorable_recurrent").
        toString());
    recurrentStatus.setValuesNeeded("
        buccal_restorable_recurrent",rs.
        getArray("
        buccal_restorable_recurrent").
        toString());
    recurrentStatus.setValuesNeeded("
        lingual_restorable_recurrent",rs.
        getArray("
        lingual_restorable_recurrent").
        toString());
    recurrentStatus.setValuesNeeded("
        mesial_restorable_recurrent",rs.
        getArray("
        mesial_restorable_recurrent").
        toString());
    recurrentStatus.setValuesNeeded("
        occlusal_restorable_recurrent",rs.
        getArray("
        occlusal_restorable_recurrent").
        toString());
    recurrentStatus.setVersion(rs.getInt("
        version"));
    recurrentStatus.setUpdated_by(rs.
        getString("updated_by"));
    recurrentStatus.setUpdated_date(rs.
        getString("updated_date"));
    recurrentStatus.setUpdated_time(rs.
        getString("updated_time"));
    return recurrentStatus;
  }
}
```

```
package org.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
     RowMapper;

import org.dentist.version.three.form.
     RestorationStatus;

public class RestorationStatusMapper
     implements RowMapper<RestorationStatus>{

   public RestorationStatus mapRow(ResultSet
        rs, int rowNum) throws SQLException
        {

    RestorationStatus restorationStatus= new
         RestorationStatus();

    restorationStatus.setrestoration_id(rs.
         getInt("restoration_id"));
    restorationStatus.setPatient_id(rs.
         getInt("patient_id"));
    restorationStatus.setValuesNeeded("
         distal_restoration",rs.getArray("
         distal_restoration").toString());
    restorationStatus.setValuesNeeded("
         buccal_restoration",rs.getArray("
         buccal_restoration").toString());
    restorationStatus.setValuesNeeded("
         lingual_restoration",rs.getArray("
         lingual_restoration").toString());
    restorationStatus.setValuesNeeded("
         mesial_restoration",rs.getArray("
         mesial_restoration").toString());
    restorationStatus.setValuesNeeded("
         occlusal_restoration",rs.getArray("
         occlusal_restoration").toString());
    restorationStatus.setValuesNeeded("
         distal_restorable_restoration",rs.
         getArray("
         distal_restorable_restoration").
         toString());
    restorationStatus.setValuesNeeded("
         buccal_restorable_restoration",rs.
         getArray("
         buccal_restorable_restoration").
         toString());
    restorationStatus.setValuesNeeded("
         lingual_restorable_restoration",rs.
         getArray("
         lingual_restorable_restoration").
         toString());
    restorationStatus.setValuesNeeded("
         mesial_restorable_restoration",rs.
         getArray("
         mesial_restorable_restoration").
         toString());
    restorationStatus.setValuesNeeded("
         occlusal_restorable_restoration",rs.
         getArray("
         occlusal_restorable_restoration").
         toString());
    restorationStatus.setVersion(rs.getInt("
         version"));
    restorationStatus.setUpdated_by(rs.
         getString("updated_by"));
    restorationStatus.setUpdated_date(rs.
         getString("updated_date"));
    restorationStatus.setUpdated_time(rs.
         getString("updated_time"));



    return restorationStatus;
  }
}
```

```
package org.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
     RowMapper;

import org.dentist.version.three.form.
     ServiceNeeded;

public class ServicesNeededMapper implements
     RowMapper<ServiceNeeded>{

 public ServiceNeeded mapRow(ResultSet rs,
     int rowNum) throws SQLException {

  ServiceNeeded serviceNeeded= new
       ServiceNeeded();

   serviceNeeded.setServiceneeded_id(rs.
        getInt("serviceneeded_id"));
   serviceNeeded.setPatient_id(rs.getInt("
        patient_id"));
   serviceNeeded.setValuesNeeded("class_1",
        rs.getArray("class_1").toString());
   serviceNeeded.setValuesNeeded("class_2",
        rs.getArray("class_2").toString());
   serviceNeeded.setValuesNeeded("class_3",
        rs.getArray("class_3").toString());
   serviceNeeded.setValuesNeeded("class_4",
        rs.getArray("class_4").toString());
   serviceNeeded.setValuesNeeded("class_5",
        rs.getArray("class_5").toString());
   serviceNeeded.setValuesNeeded("onlay",rs.
        getArray("onlay").toString());
   serviceNeeded.setValuesNeeded("extraction
        ",rs.getArray("extraction").toString
        ());
   serviceNeeded.setValuesNeeded("
        odontectomy",rs.getArray("odontectomy
        ").toString());
   serviceNeeded.setValuesNeeded("
        special_case",rs.getArray("
        special_case").toString());
   serviceNeeded.setValuesNeeded("
        pulp_sedation",rs.getArray("
        pulp_sedation").toString());
   serviceNeeded.setValuesNeeded("
        crown_recementation",rs.getArray("
        crown_recementation").toString());
   serviceNeeded.setValuesNeeded("
        filling_service",rs.getArray("
        filling_service").toString());
   serviceNeeded.setValuesNeeded("laminated
        ",rs.getArray("laminated").toString()
        );
   serviceNeeded.setValuesNeeded("
        single_crown",rs.getArray("
        single_crown").toString());
   serviceNeeded.setValuesNeeded("
        bridge_service",rs.getArray("
        bridge_service").toString());
   serviceNeeded.setValuesNeeded("anterior",
        rs.getArray("anterior").toString());
   serviceNeeded.setValuesNeeded("posterior
        ",rs.getArray("posterior").toString()
        );
   serviceNeeded.setValuesNeeded("ortho_endo
        ",rs.getArray("ortho_endo").toString
        ());
   serviceNeeded.setPeriodontics(rs.
        getString("periodontics"));
   serviceNeeded.setSurgery(rs.getString("
        surgery"));
   serviceNeeded.setEmergency_treatment(rs.
        getString("emergency_treatment"));
   serviceNeeded.setProsthodontics(rs.
        getString("prosthodontics"));
   serviceNeeded.setUpdated_by(rs.getString
        ("updated_by"));
   serviceNeeded.setUpdated_date(rs.
        getString("updated_date"));
   serviceNeeded.setUpdated_time(rs.
        getString("updated_time"));
   serviceNeeded.setVersion(rs.getInt("
        version"));
   serviceNeeded.setNotes(rs.getString("
      notes"));
   serviceNeeded.setIs_current(rs.getString
      ("is_current"));
  return serviceNeeded;
 }
}
```

```
package org.domain.standalonedesignerdemo.
    session;

public class Asset {
 private String name;
 private String uuid;
 private String format;
 private String created;
 private String createdby;
 private String lastmodified;
 private String pkgname;
 private String description;
 private String version;

 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getUuid() {
  return uuid;
 }
 public void setUuid(String uuid) {
  this.uuid = uuid;
 }
 public String getFormat() {
  return format;
 }
 public void setFormat(String format) {
  this.format = format;
 }

 public String getCreated() {
  return created;
 }
 public void setCreated(String created) {
  this.created = created;
 }
 public String getCreatedby() {
  return createdby;
 }
 public void setCreatedby(String createdby)
     {
  this.createdby = createdby;
 }
 public String getLastmodified() {
  return lastmodified;
 }
 public void setLastmodified(String
     lastmodified) {
  this.lastmodified = lastmodified;
 }
 public String getPkgname() {
  return pkgname;
 }
 public void setPkgname(String pkgname) {
  this.pkgname = pkgname;
 }
 public String getDescription() {
  return description;
 }
 public void setDescription(String
     description) {
  this.description = description;
 }

 public String getVersion() {
  return version;
 }
 public void setVersion(String version) {
  this.version = version;
 }
 public boolean equals(final Object o) {
  if (o instanceof Asset) {
   if (this.getName() != null) {
    return ((Asset) o).getName() == null;
   }
   return this.getName().equals(((Asset) o).
       getName());
  }
  return false;
 }

 public int hashCode() {
  return this.getName() == null ? 0 : 3 *
      this.getName().hashCode();
 }


}
```

```
package org.domain.standalonedesignerdemo.
    session;

import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Logger;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.log.Log;
import org.jboss.seam.security.Credentials;
import org.jboss.seam.security.Identity;

@Name("authenticator")
public class Authenticator
{
    @Logger private Log log;

    @In Identity identity;
    @In Credentials credentials;

    public boolean authenticate()
    {
        log.info("authenticating {0}",
            credentials.getUsername());
        //write your authentication logic
            here,
        //return true if the authentication
            was
        //successful, false otherwise
        if ("admin".equals(credentials.
            getUsername()))
        {
            identity.addRole("admin");
            return true;
        }
        return false;
    }

}
```

```
package org.domain.standalonedesignerdemo.
    session;

import java.util.List;

public class Package {
 private String name;
 private String uuid;
 private List<Asset> packageAssets;

 public String getName() {
  return name;
 }

 public void setName(String name) {
  this.name = name;
 }
 public String getUuid() {
  return uuid;
 }

 public void setUuid(String uuid) {
  this.uuid = uuid;
 }

 public List<Asset> getPackageAssets() {
  return packageAssets;
 }

 public void setPackageAssets(List<Asset>
     packageAssets) {
  this.packageAssets = packageAssets;
 }

 public boolean equals(final Object o) {
  if (o instanceof Package) {
   if (this.getName() != null) {
    return ((Package) o).getName() == null;
   }
   return this.getName().equals(((Package) o
       ).getName());
  }
  return false;


package org.domain.standalonedesignerdemo.
     session;


import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.InetAddress;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;

import javax.faces.event.ActionEvent;
import javax.ws.rs.core.Response;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamReader;


//import org.jboss.beans.metadata.api.
     annotations.Create;
import org.apache.cxf.jaxrs.client.WebClient
     ;
import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Begin;
import org.jboss.seam.annotations.Factory;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Logger;
import org.jboss.seam.annotations.Out;
import org.jboss.seam.annotations.Scope;
import org.jboss.seam.log.Log;
import org.jboss.seam.international.
     StatusMessages;
import org.richfaces.component.html.HtmlTree
     ;
import org.richfaces.event.NodeSelectedEvent
     ;
import org.richfaces.model.TreeNode;
import org.richfaces.model.TreeNodeImpl;

@Name("StandaloneManager")
@Scope(ScopeType.SESSION)
public class StandaloneManager implements
     Serializable {
 @Logger
 private Log log;
 @In
 StatusMessages statusMessages;


 @In(create=true, required=false)
 @Out(required=false)
 private String nodeInfo;
 @In(create=true, required=false)
 @Out(required=false)
 private String iframeurl;
 @In(create=true, required=false)
 @Out(required=false)
 private String newprocessname;

 private List<Package> packages = new
      ArrayList<Package>();
 @In(create=true, required=false)
 @Out(required=false)
 private Asset selectedAsset;
 @In(create=true, required=false)
 @Out(required=false)
 private Package selectedPackage;
 @In(create=true, required=false)
 @Out(required=false)
 private String newProcessName;
 TreeNodeImpl root = new TreeNodeImpl();
 @Factory("allPackages")
 public TreeNodeImpl getAllPackages() {
  return doHttpConnectionToGetAllPackages();
 }
```

```
 }

 public int hashCode() {
  return this.getName() == null ? 0 : 3 *
       this.getName().hashCode();
 }

}


 public void standaloneManager() {
 }

 //@Create
 @Begin(join = true)
 public void init() {

 }

 protected org.richfaces.component.UITree
      sampleTreeBinding;

 public org.richfaces.component.UITree
      getSampleTreeBinding() {

 return sampleTreeBinding;

 }

 public void setSampleTreeBinding(

 org.richfaces.component.UITree
      sampleTreeBinding) {

 this.sampleTreeBinding = sampleTreeBinding;

 }

 public void assembleEditor() {

 }

 public void editAsset() throws Exception {
  System.out.println("**** selected asset: "
       + selectedAsset.getName());
  iframeurl = "http://"+InetAddress.
       getLocalHost().getCanonicalHostName()
       +":8090/drools-guvnor/org.drools.
       guvnor.Guvnor/standaloneEditorServlet?
       assetsUUIDs="
   + selectedAsset.getUuid() + "&client=oryx
       ";
  System.out.println("SET IFRAME URL TO : "
       + iframeurl);
  newProcessName=null;


 }

 public void createAsset() throws Exception
      {
  System.out.println("**** selected package:
       " + selectedPackage.getName());
  System.out.println("**** new process name
       is: " + newProcessName);


  iframeurl = "http://"+InetAddress.
       getLocalHost().getCanonicalHostName()
       +":8090/drools-guvnor/org.drools.
       guvnor.Guvnor/standaloneEditorServlet?
       assetsUUIDs={asset.UUID}&packageName="
   + selectedPackage.getName() + "&
        createNewAsset=true" + "&assetName="
        + newProcessName + "&assetFormat=
        bpmn2" +"&client=oryx";

  System.out.println("SET IFRAME URL TO : "
       + iframeurl);

 }

 public void addAsset(){
  iframeurl="";
  if(newProcessName!=null){
  Package p= selectedPackage;
  TreeNodeImpl node = (TreeNodeImpl) root.
       getChild(p.getName());

  p.setPackageAssets(
       doHttpConnectionToGetAllAssetsFor(p
   .getName()));
 for (Asset a : p.getPackageAssets()) {
  if(a.getName().equals(newProcessName)) {
   a.setPkgname(p.getName());
   TreeNodeImpl snode = new TreeNodeImpl();
   snode.setData(a.getName() + "." + a.
        getFormat());
   snode.setParent(node);
   node.addChild(a.getUuid(), snode);
  }
 }
 }


}
public void deleteAsset() throws Exception{
 String thePath = "http://"+InetAddress.
      getLocalHost().getCanonicalHostName()
      +":8090/drools-guvnor/rest/packages/"+
      selectedAsset.getPkgname()+"/assets/"
      + selectedAsset.getName();
 System.out.println("delete: "+ thePath);
 thePath = URLEncoder.encode(thePath, "UTF
      -8");
 WebClient client = WebClient.create(
      thePath, "admin", "admin", null);

 Response response = client.delete();

 if(root.getChild(selectedAsset.getPkgname
      ()).getChild(selectedAsset.getUuid())
      !=null){
  root.getChild(selectedAsset.getPkgname())
       .removeChild(selectedAsset.getUuid())
       ;
 }
 selectedPackage = null;
 selectedAsset = null;

}

public void changeProcessName(ActionEvent
     actionEvent) throws Exception {
    System.out.println("---new process name
         : " + newProcessName);
    iframeurl = "http://"+InetAddress.
         getLocalHost().getCanonicalHostName
         ()+":8090/drools-guvnor/org.drools.
         guvnor.Guvnor/
         standaloneEditorServlet?assetsUUIDs
         ={asset.UUID}&packageName="
  + selectedPackage.getName() + "&
       createNewAsset=true" + "&assetName="
       + newProcessName + "&assetFormat=
       bpmn2" +"&client=oryx";

 System.out.println("SET IFRAME URL TO : "
      + iframeurl);

}

public void processSelection(
     NodeSelectedEvent event) {
 HtmlTree tree = (HtmlTree) event.
      getComponent();
 nodeInfo = (String) tree.getRowData();
 TreeNode currentNode = tree.
      getModelTreeNode(tree.getRowKey());
 if (currentNode.isLeaf()) {
  nodeInfo = "asset:" + nodeInfo;
  String uuid = nodeInfo.substring(nodeInfo
       .indexOf("(") + 1,
    nodeInfo.length() - 1);
  String name = nodeInfo.substring(nodeInfo
       .indexOf(":") + 1,
    nodeInfo.indexOf("."));
  for(Package p : packages) {
   for(Asset a : p.getPackageAssets()) {
    if(a.getName().equals(name)) {
     selectedAsset = a;
     System.out.println(nodeInfo);
     System.out.println("name: " + name +
          selectedAsset.getName());
     System.out.println("*** setting
          selected asset: " + selectedAsset.
          getUuid());
     selectedPackage = null;
     break;
    }
   }
  }
  //TODO ADD THIS BACK IN ANOTHER METHOD!!!
  //iframeurl = "http://localhost:8080//
       drools-guvnor/org.drools.guvnor.
       Guvnor/standaloneEditorServlet?
       assetsUUIDs="
  // + uuid + "&client=oryx";
  iframeurl = "";
 } else {
  String pname = nodeInfo;
  nodeInfo = "package:" + nodeInfo;
  for(Package p : packages) {
   if(p.getName().equals(pname)) {
    selectedPackage = p;
    System.out.println("*** setting
         selected package: " +
         selectedPackage.getName());
    selectedAsset = null;
    break;
   }
  }
  iframeurl = ""; // TODO for now
 }
}

private List<Asset>
     doHttpConnectionToGetAllAssetsFor(
     String pkg) {
 HttpURLConnection connection;
 InputStream is = null;
 try {
  List<Asset> assets = new ArrayList<Asset
       >();
  URL url = new URL(
    "http://"+InetAddress.getLocalHost().
         getCanonicalHostName()+":8090/
         drools-guvnor/rest/packages/" + pkg
      + "/assets");
  connection = (HttpURLConnection) url.
       openConnection();

  connection.setRequestMethod("GET");
  connection
    .setRequestProperty(
     "User-Agent",
     "Mozilla/5.0 (Macintosh; U; Intel Mac
          OS X 10.6; en-US; rv:1.9.2.16)
         Gecko/20110319 Firefox/3.6.16");
  connection
    .setRequestProperty("Accept",
     "text/html,application/xhtml+xml,
          application/xml;q=0.9,*/*;q=0.8")
          ;
  connection.setRequestProperty("Accept-
       Language", "en-us,en;q=0.5");
  connection.setRequestProperty("Accept-
       Encoding", "gzip,deflate");
  connection.setRequestProperty("charset",
       "UTF-8");
  connection.setReadTimeout(50 * 1000);
  connection.connect();

  BufferedReader sreader = new
       BufferedReader(new InputStreamReader(
    connection.getInputStream(), "UTF-8"));
  StringBuilder stringBuilder = new
       StringBuilder();

  String line = null;
  while ((line = sreader.readLine()) !=
       null) {
   stringBuilder.append(line + "\n");
  }

  is = new ByteArrayInputStream(
       stringBuilder.toString().toString()
    .getBytes("UTF-8"));

  XMLInputFactory factory = XMLInputFactory
       .newInstance();
  XMLStreamReader reader = factory.
       createXMLStreamReader(is);
  while (reader.hasNext()) {
   if (reader.next() == XMLStreamReader.
        START_ELEMENT) {
    if ("asset".equals(reader.getLocalName
         ())) {
     Asset a = new Asset();
     assets.add(a);
     assets.get(assets.size() - 1).
          setPkgname(pkg);
    }
    if ("format".equals(reader.getLocalName
          ())) {
       assets.get(assets.size() − 1).
            setFormat(
         reader.getElementText());
      }
      if ("title".equals(reader.getLocalName
           ())) {
       assets.get(assets.size() − 1).setName(
         reader.getElementText());
      }
      if ("uuid".equals(reader.getLocalName()
           )) {
       assets.get(assets.size() − 1).setUuid(
         reader.getElementText());
      }
      if ("created".equals(reader.
           getLocalName())) {
       assets.get(assets.size() − 1).
            setCreated(
         reader.getElementText());
      }
      if ("createdBy".equals(reader.
           getLocalName())) {
       assets.get(assets.size() − 1).
            setCreatedby(
         reader.getElementText());
      }
      if ("lastModified".equals(reader.
           getLocalName())) {
       assets.get(assets.size() − 1).
            setLastmodified(
         reader.getElementText());
      }
      if ("description".equals(reader.
           getLocalName())) {
       assets.get(assets.size() − 1).
            setDescription(
         reader.getElementText());
      }
      if ("version".equals(reader.
           getLocalName())) {
       assets.get(assets.size() − 1).
            setVersion(
         reader.getElementText());
      }
     }
    }

    return assets;
   } catch (Exception e) {
    e.printStackTrace();
    return null;
   } finally {
    if (is != null) {
     try {
      is.close();
     } catch (IOException e) {
     }
    }
    ;
   }

  }

  private TreeNodeImpl
       doHttpConnectionToGetAllPackages() {

   root.setParent(null);
   root.setData("Packages");

   HttpURLConnection connection;
   InputStream is = null;

   try {
    URL url = new URL(
      "http://"+InetAddress.getLocalHost().
           getCanonicalHostName()+":8090/
           drools−guvnor/rest/packages");
    connection = (HttpURLConnection) url.
         openConnection();

    connection.setRequestMethod("GET");
    connection
      .setRequestProperty(
        "User−Agent",
        "Mozilla/5.0 (Macintosh; U; Intel Mac
              OS X 10.6; en−US; rv:1.9.2.16)
             Gecko/20110319 Firefox/3.6.16");
    connection
      .setRequestProperty("Accept",
        "text/html,application/xhtml+xml,
             application/xml;q=0.9,*/*;q=0.8")
             ;
    connection.setRequestProperty("Accept−
         Language", "en−us,en;q=0.5");
    connection.setRequestProperty("Accept−
         Encoding", "gzip,deflate");
    connection.setRequestProperty("charset",
         "UTF−8");
    connection.setReadTimeout(50 * 1000);
    connection.connect();

    BufferedReader sreader = new
         BufferedReader(new InputStreamReader(
      connection.getInputStream(), "UTF−8"));
    StringBuilder stringBuilder = new
         StringBuilder();

    String line = null;
    while ((line = sreader.readLine()) !=
         null) {
     stringBuilder.append(line + "\n");
    }

    is = new ByteArrayInputStream(
         stringBuilder.toString().toString()
       .getBytes("UTF−8"));

    XMLInputFactory factory = XMLInputFactory
         .newInstance();
    XMLStreamReader reader = factory
      .createXMLStreamReader(new
           InputStreamReader(is));
    while (reader.hasNext()) {
     if (reader.next() == XMLStreamReader.
          START_ELEMENT) {
      TreeNodeImpl node = new TreeNodeImpl();
      if ("package".equals(reader.
           getLocalName())) {
       Package p = new Package();
       packages.add(p);
      }
      if ("title".equals(reader.getLocalName
           ())) {
       packages.get(packages.size() − 1).
            setName(
         reader.getElementText());
      }
      if ("uuid".equals(reader.getLocalName()
           )) {
       packages.get(packages.size() − 1).
            setUuid(
         reader.getElementText());
      }
     }
    }

    for (Package p : packages) {
     TreeNodeImpl node = new TreeNodeImpl();
     node.setData(p.getName());
     node.setParent(root);
     p.setPackageAssets(
          doHttpConnectionToGetAllAssetsFor(p
       .getName()));
     for (Asset a : p.getPackageAssets()) {
      if(!a.getName().equals("null") &&(a.
           getFormat().equals("bpmn") || a.
           getFormat().equals("bpmn2"))) {
       a.setPkgname(p.getName());
       TreeNodeImpl snode = new TreeNodeImpl
            ();
       snode.setData(a.getName() + "." + a.
            getFormat());
       snode.setParent(node);
       node.addChild(a.getUuid(), snode);
      }
     }
     root.addChild(p.getName(), node);

    }

    return root;
   } catch (Exception e) {
    e.printStackTrace();
    return root;
   } finally {
    if (is != null) {
     try {
      is.close();
     } catch (IOException e) {
     }
    }
    ;
   }
```

```
 }

 public boolean canShowPackageInfo() {
  return selectedPackage != null &&
       selectedPackage.getName() != null &&
       selectedPackage.getName().length() >
       0;
 }

 public boolean canShowAssetInfo() {
  return selectedAsset != null &&
       selectedAsset.getUuid() != null &&
       selectedAsset.getUuid().length() > 0;
 }

 public String getNodeInfo() {
  return nodeInfo;
 }

 public void setNodeInfo(String nodeInfo) {
  this.nodeInfo = nodeInfo;
 }

 public String getIframeurl() {
  return iframeurl;
 }

 public void setIframeurl(String iframeurl)
      {
  this.iframeurl = iframeurl;
 }

 public String getNewprocessname() {
  return newprocessname;
 }

 public void setNewprocessname(String
      newprocessname) {
  if(newprocessname != null &&
       newprocessname.length() > 0) {
   this.newprocessname = newprocessname;
  }
 }

 public Asset getSelectedAsset() {
  return selectedAsset;
 }

 public void setSelectedAsset(Asset
      selectedAsset) {
  this.selectedAsset = selectedAsset;
 }

 public Package getSelectedPackage() {
  return selectedPackage;
 }

 public void setSelectedPackage(Package
      selectedPackage) {
  this.selectedPackage = selectedPackage;
 }

 public String getNewProcessName() {
  return newProcessName;
 }

 public void setNewProcessName(String
      newProcessName) {
  this.newProcessName = newProcessName;
 }
}
```

```
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<html>
<body>

<c:if test="${message!=null}">
<c:redirect url="task?idTask=&nameTask=&
     username=&password=&message=${message
     }"/>
   </c:if>

<c:if test="${formurl==null}">
<c:if test="${instanceid==null}">
    <c:redirect url="greet?patientid=${
         patientid}&pos=${pos}"/>
   </c:if>
<c:if test="${instanceid!=null}">
    <c:redirect url="greet?patientid=${
         patientid}&instanceid=${instanceid}&
         pos=${pos}"/>
   </c:if>
</c:if>

<c:if test="${formurl!=null && app==null}">
<c:redirect url="${formurl}&notif=${notif
     }"/>
   </c:if>
<c:if test="${formurl!=null && app!=null}">
<c:redirect url="${formurl}&notif=${notif}&
     app=${app}"/>
   </c:if>
</body>
</html>
```

```
 <style type="text/css">
.black_overlay{
 display: none;
 position: absolute;
 top: 0%;
 left: 5%;
 width: 100%;
 height: 120%;
 background-color: black;
 z-index:1001;
 -moz-opacity: 0.2;
 opacity:.20;
 filter: alpha(opacity=20);
}
.white_content {
 display: none;
 position: absolute;
 top: 10%;
 left: 15%;
 width: 70%;
 height: 55%;
 padding: 16px;
 border: 5px solid #1aac9b;
 background-color: white;
 z-index:1002;
 overflow: auto;
}
</style>
<script type="text/javascript"> init(65, 65,
      627);     </script>


<div id="dentureslight" class="white_content
     ">
 <font size = "2">
 <b>Dentures Status</b><br/>
 <hr>
 <input type="checkbox" name="
      completedenture" value="yes" id="
      completedenture" onclick="
      drawCompletedenture(ctx1, ctx30, 65,
      65)" <c:if test="${dentalChart.
      complete_denture eq 'yes'}">checked="
      yes"</c:if>>Complete Denture
 <br/><br/>
 <u> Single Denture </u><br/>
 <input type="checkbox" name="singledenture"
       value="upper" id="upperdenture"
      onclick="drawUpperDenture(ctx2, ctx31,
      65, 65)" <c:if test="${dentalChart.
      single_denture eq 'upper'}">checked="
      yes"</c:if>>Upper Single Denture <br/>
 <input type="checkbox" name="singledenture"
       value="lower" id="lowerdenture"
      onclick="drawLowerDenture(ctx3, ctx32,
      65, 65)" <c:if test="${dentalChart.
      single_denture eq 'lower'}">checked="
      yes"</c:if>>Lower Single Denture <br/>
 <hr>
 <a href = "javascript:void(0)" onclick = "
      document.getElementById('dentureslight
      ').style.display='none';document.
      getElementById('denturesfade').style.
      display='none'">Done</a>

 </font>
</div>
```

```
<%


 int i = 18;
 int j = 1;
 int tn = 18;
 for(j = 0; j < 4; j++) {
  for(i = 18; i > 10; i--) {
   tn = i+j*10;
  String restorableVar= "restorable"+tn;
  String nonrestorableVar="nonrestorable"+tn
      ;
  String amVar="AM"+tn;
  String coVar="CO"+tn;
  String giVar="GI"+tn;
  String tfVar="TF"+tn;




%>
    <c:set var="tn" value="<%= tn %>"/>
     <c:set var="restorablevar" value="<%=
         restorableVar %>"/>
    <c:set var="nonrestorablevar" value="<%=
         nonrestorableVar %>"/>
    <c:set var="amvar" value="<%= amVar %>"/>
    <c:set var="covar" value="<%= coVar %>"/>
    <c:set var="givar" value="<%= giVar %>"/>
    <c:set var="tfvar" value="<%= tfVar %>"/>



<div id="light<%= tn %>" class="
     white_content" name="light<%= tn %>">
<div id="canvasesdiv" >
<canvas id="layer1mini<%= tn %>" style="
     border:solid 2px; z-index: 1; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer2mini<%= tn %>" style="
     border:solid 2px; z-index: 2; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer3mini<%= tn %>" style="
     border:solid 2px; z-index: 3; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer4mini<%= tn %>" style="
     border:solid 2px; z-index: 4; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer5mini<%= tn %>" style="
     border:solid 2px; z-index: 5; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer6mini<%= tn %>" style="
     border:solid 2px; z-index: 6; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer7mini<%= tn %>" style="
     border:solid 2px; z-index: 7; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer8mini<%= tn %>" style="
     border:solid 2px; z-index: 8; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer9mini<%= tn %>" style="
     border:solid 2px; z-index: 9; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer10mini<%= tn %>" style="
     border:solid 2px; z-index: 10; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer11mini<%= tn %>" style="
     border:solid 2px; z-index: 11; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer12mini<%= tn %>" style="
     border:solid 2px; z-index: 12; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer13mini<%= tn %>" style="
     border:solid 2px; z-index: 13; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer14mini<%= tn %>" style="
     border:solid 2px; z-index: 14; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer15mini<%= tn %>" style="
     border:solid 2px; z-index: 15; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer16mini<%= tn %>" style="
     border:solid 2px; z-index: 16; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer17mini<%= tn %>" style="
     border:solid 2px; z-index: 17; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas></div>
<b>Tooth <%= tn %>  </b><br/><br/><font
     color="red"><div id="toothvar<%= tn
     %>"></div></font><br/><br/><a href = "
     javascript:void(0)" onclick = "
     submitcond()">Submit</a>&nbsp;
<a href = "javascript:void(0)" onclick = "
     cancelcond(<%= tn %>)">Cancel</a><br/> <
     br/>
<br><input type="checkbox" name="
     dentalStatus" value="<%= tn %>" id="
     dentalStatus<%= tn %>" onclick="
     hideAllTooth()" <c:if test="${fn:
     contains(fn:join(dentalChart.
     caries_string,','),tn) || fn:contains(fn
     :join(dentalChart.recurrentcaries_string
     ,','),tn) || fn:contains(fn:join(
     dentalChart.restoration_string,','),tn)
     }">checked="yes"</c:if>><b>Dental Status
     </b><br/>


<div id="dentalstatussurface<%= tn %>" style
     ="display:none;"><table><tr><td></td><td
     ><input type='checkbox' name='caries'
     value='<%= tn %>' id='caries<%= tn %>'
     onclick='hideOtherCaries()' <c:if test
     ="${fn:contains(fn:join(dentalChart.
     caries_string,','),tn)}">checked="yes"</
     c:if>></td><td>Caries</td>

<td><input type="checkbox" name="
     recurrentcaries" value="<%= tn %>" id="
     recurrentcaries<%= tn %>" onclick="
     hideOtherReccurent()" <c:if test="${fn:
     contains(fn:join(dentalChart.
     recurrentcaries_string,','),tn)}">
     checked="yes"</c:if>></td><td>Reccurent
     </td>

<td><input type="checkbox" name="restoration
     " value="<%= tn %>" id="restoration<%=
     tn %>" onclick="hideOtherRestoration()"
      <c:if test="${fn:contains(fn:join(
     dentalChart.restoration_string,','),tn)
     }">checked="yes"</c:if>></td><td>
     Restoration</td>
</tr>
<tr><td>Mesial</td><td><div id="
     caries_surfaces<%= tn %>mesial" style="
     display:none;"><input type="checkbox"
     name="mesialcaries" value="<%= tn %>" id
     ="mesial<%= tn %>" onclick="showvar('
     mesial'+cariesSelectSurfaceid,
     mesialformid);drawConditionMini('caries
     ', 'mesial',<%= tn %>) " <c:if test="${
     fn:contains(fn:join(cariesStatus.
     mesial_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div></td>
<td><div id="mesialcariesSelectSurface<%= tn
      %>" style="display:none;"><select name
     ="selectCariesMesial" id="
     mesialcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="restorable<%= tn %>" <c:if
     test="${fn:contains(fn:join(cariesStatus
     .mesial_restorable_caries,','),
     restorablevar)}">selected</c:if>> O </
     option>
<option value="nonrestorable<%= tn %>" <c:if
      test="${fn:contains(fn:join(
     cariesStatus.mesial_restorable_caries
     ,','),nonrestorablevar)}">selected</c:if
     >> / </option>
</select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      mesial" style="display:none;"><input
      type="checkbox" name="mesialrecurrent"
      value="<%= tn %>" id="remesial<%= tn
      %>" onclick="showvar('mesial'+
      recurrentSelectSurfaceid,remesialformid
      );drawConditionMini('recurrent', '
      mesial',<%= tn %>) "  <c:if test="${fn:
      contains(fn:join(recurrentStatus.
      mesial_recurrent_string,','),tn)}">
      checked="yes"</c:if>/></div></td>
 <td><div id="mesialrecurrentSelectSurface
      <%= tn %>" style="display:none;"><
      select name="selectRecurrentMesial" id
      ="mesialrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option><option
      value="restorable<%= tn %>" <c:if test
      ="${fn:contains(fn:join(recurrentStatus
      .mesial_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      mesial_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      mesial" style="display:none;"><input
      type="checkbox" name="mesialrestoration
      " value="<%= tn %>" id="restomesial<%=
      tn %>" onclick="showvar('mesial'+
      restoreSelectSurfaceid,
      restomesialformid);drawConditionMini('
      restoration', 'mesial',<%= tn %>) "  <c
      :if test="${fn:contains(fn:join(
      restorationStatus.
      mesial_restoration_string,','),tn)}">
      checked="yes"</c:if>/></div></td><td><
      div id="mesialrestoreSelectSurface<%=
      tn %>" style="display:none;"><select
      name="selectRestorationMesial" id="
      mesialrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>" ></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,','),amvar
     )}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,','),covar
     )}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,','),givar
     )}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,','),tfvar
     )}">selected</c:if>>TF</option>
</select></div></td></tr>


<tr><td>Distal</td><td><div id="
     caries_surfaces<%= tn %>distal" style="
     display:none;"><input type="checkbox"
     name="distalcaries" value="<%= tn %>" id
     ="distal<%= tn %>" onclick="showvar('
     distal'+cariesSelectSurfaceid,
     distalformid);drawConditionMini('caries
     ', 'distal', <%= tn %> )" <c:if test="${
     fn:contains(fn:join(cariesStatus.
     distal_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div></td>
<td><div id="distalcariesSelectSurface<%= tn
      %>" style="display:none;">
<select name="selectCariesDistal" id="
      distalcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.distal_restorable_caries
       ,','),restorablevar)}">selected</c:if
       >> O </option>
  <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       cariesStatus.distal_restorable_caries
       ,','),nonrestorablevar)}">selected</c:
       if>> / </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      distal" style="display:none;"><input
      type="checkbox" name="distalrecurrent"
```

```
      value="<%= tn %>" id="redistal<%= tn
      %>" onclick="showvar('distal'+
      recurrentSelectSurfaceid,redistalformid
      );drawConditionMini('recurrent', '
      distal',<%= tn %>) " <c:if test="${fn:
      contains(fn:join(recurrentStatus.
      distal_recurrent_string,','),tn)}">
      checked="yes"</c:if> /></div></td>
 <td><div id="distalrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentDistal" id="
      distalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option>
 <option value="restorable<%= tn %>" <c:if
      test="${fn:contains(fn:join(
      recurrentStatus.
      distal_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      distal_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      distal" style="display:none;"><input
      type="checkbox" name="distalrestoration
      " value="<%= tn %>" id="restodistal<%=
      tn %>" onclick="showvar('distal'+
      restoreSelectSurfaceid,
      restodistalformid);drawConditionMini('
      restoration', 'distal',<%= tn %>) " <c:
      if test="${fn:contains(fn:join(
      restorationStatus.
      distal_restoration_string,','),tn)}">
      checked="yes"</c:if> /></div></td><td><
      div id="distalrestoreSelectSurface<%=
      tn %>" style="display:none;"><select
      name="selectRestorationDistal" id="
      distalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     distal_restorable_restoration,','),amvar
     )}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     distal_restorable_restoration,','),covar
     )}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     distal_restorable_restoration,','),givar
     )}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     distal_restorable_restoration,','),tfvar
     )}">selected</c:if>>TF</option>

</select></div></td></tr>



<tr><td>Occlusal</td><td><div id="
     caries_surfaces<%= tn %>occlusal" style
     ="display:none;"><input type="checkbox"
     name="occlusalcaries" value="<%= tn %>"
     id="occlusal<%= tn %>" onclick="showvar
     ('occlusal'+cariesSelectSurfaceid,
     occlusalformid);drawConditionMini('
     caries', 'occlusal',<%= tn %>) "  <c:if
     test="${fn:contains(fn:join(cariesStatus
     .occlusal_caries_string,','),tn)}">
     checked="yes"</c:if> /></div></td>
<td><div id="occlusalcariesSelectSurface<%=
     tn %>" style="display:none;">
<select name="selectCariesOcclusal"  id="
     occlusalcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.
       occlusal_restorable_caries,','),
       restorablevar)}">selected</c:if>> O </
       option>
  <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       cariesStatus.
       occlusal_restorable_caries,','),
       nonrestorablevar)}">selected</c:if>> /
        </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      occlusal" style="display:none;"><input
      type="checkbox" name="occlusalrecurrent
      " value="<%= tn %>" id="reocclusal<%=
      tn %>" onclick="showvar('occlusal'+
      recurrentSelectSurfaceid,
      reocclusalformid);drawConditionMini('
      recurrent', 'occlusal',<%= tn %>) "  <c
      :if test="${fn:contains(fn:join(
      recurrentStatus.
      occlusal_recurrent_string,','),tn)}">
      checked="yes"</c:if>/></div></td>
 <td><div id="occlusalrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentOcclusal" id="
      occlusalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option><option
      value="restorable<%= tn %>" <c:if test
      ="${fn:contains(fn:join(recurrentStatus
      .occlusal_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      occlusal_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      occlusal" style="display:none;"><input
      type="checkbox" name="
      occlusalrestoration" value="<%= tn %>"
      id="restoocclusal<%= tn %>" onclick="
      showvar('occlusal'+
      restoreSelectSurfaceid,
      restoocclusalformid);drawConditionMini
      ('restoration', 'occlusal',<%= tn %>) "
        <c:if test="${fn:contains(fn:join(
      restorationStatus.
      occlusal_restoration_string,','),tn)}">
      checked="yes"</c:if>/></div></td><td><
      div id="occlusalrestoreSelectSurface<%=
       tn %>" style="display:none;"><select
      name="selectRestorationOcclusal" id="
      occlusalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     occlusal_restorable_restoration,','),
     amvar)}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     occlusal_restorable_restoration,','),
     covar)}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     occlusal_restorable_restoration,','),
     givar)}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     occlusal_restorable_restoration,','),
     tfvar)}">selected</c:if>>TF</option>

</select></div></td></tr>



<tr><td>Buccal</td><td><div id="
     caries_surfaces<%= tn %>buccal" style="
     display:none;"><input type="checkbox"
     name="buccalcaries" value="<%= tn %>" id
     ="buccal<%= tn %>" onclick="showvar('
     buccal'+cariesSelectSurfaceid,
     buccalformid);drawConditionMini('caries
     ', 'buccal',<%= tn %>) " <c:if test="${
     fn:contains(fn:join(cariesStatus.
     buccal_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div></td><td><div id="
     buccalcariesSelectSurface<%= tn %>"
     style="display:none;"><select name="
     selectCariesBuccal" id="
     buccalcariesSelect<%= tn %>"><option
     value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.buccal_restorable_caries
       ,','),restorablevar)}">selected</c:if
       >> O </option>
  <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       cariesStatus.buccal_restorable_caries
       ,','),nonrestorablevar)}">selected</c:
       if>> / </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      buccal" style="display:none;"><input
      type="checkbox" name="buccalrecurrent"
      value="<%= tn %>" id="rebuccal<%= tn
      %>" onclick="showvar('buccal'+
      recurrentSelectSurfaceid,rebuccalformid
      );drawConditionMini('recurrent', '
      buccal',<%= tn %>) "  <c:if test="${fn:
      contains(fn:join(recurrentStatus.
      buccal_recurrent_string,','),tn)}">
      checked="yes"</c:if>/></div></td>
 <td><div id="buccalrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentBuccal" id="
      buccalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option>
 <option value="restorable<%= tn %>" <c:if
      test="${fn:contains(fn:join(
      recurrentStatus.
      buccal_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      buccal_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      buccal" style="display:none;"><input
      type="checkbox" name="buccalrestoration
      " value="<%= tn %>" id="restobuccal<%=
      tn %>" onclick="showvar('buccal'+
      restoreSelectSurfaceid,
      restobuccalformid);drawConditionMini('
      restoration', 'buccal',<%= tn %>) "  <c
      :if test="${fn:contains(fn:join(
      restorationStatus.
      buccal_restoration_string,','),tn)}">
      checked="yes"</c:if>/></div></td><td><
      div id="buccalrestoreSelectSurface<%=
      tn %>" style="display:none;"><select
      name="selectRestorationBuccal" id="
      buccalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),amvar
     )}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),covar
     )}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),givar
     )}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),tfvar
     )}">selected</c:if>>TF</option>

</select></div></td></tr>



<tr><td>Lingual</td><td><div id="
     caries_surfaces<%= tn %>lingual" style="
     display:none;"><input type="checkbox"
     name="lingualcaries" value="<%= tn %>"
     id="lingual<%= tn %>" onclick="showvar('
     lingual'+cariesSelectSurfaceid,
     lingualformid);drawConditionMini('caries
     ', 'lingual',<%= tn %>) "  <c:if test="$
     {fn:contains(fn:join(cariesStatus.
     lingual_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div> </td>
<td><div id="lingualcariesSelectSurface<%=
     tn %>" style="display:none;"><select
     name="selectCariesLingual" id="
     lingualcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.lingual_restorable_caries
```

```
        ,’,’),restorablevar)}”>selected</c:if
        >> O </option>
  <option value=”nonrestorable<%= tn %>” <c:
        if test=”${fn:contains(fn:join(
        cariesStatus.lingual_restorable_caries
        ,’,’),nonrestorablevar)}”>selected</c:
        if>> / </option>
 </select></div></td>
 <td><div id=”recurrent_surfaces<%= tn %>
       lingual” style=”display:none;”><input
       type=”checkbox” name=”lingualrecurrent”
        value=”<%= tn %>” id=”relingual<%= tn
       %>” onclick=”showvar(’lingual’+
       recurrentSelectSurfaceid,
       relingualformid);drawConditionMini(’
       recurrent’, ’lingual’,<%= tn %>) ” <c:
       if test=”${fn:contains(fn:join(
       recurrentStatus.
       lingual_recurrent_string,’,’),tn)}”>
       checked=”yes”</c:if>/></div></td>
 <td><div id=”lingualrecurrentSelectSurface
      <%= tn %>” style=”display:none;”>
 <select name=”selectRecurrentLingual” id=”
       lingualrecurrentSelect<%= tn %>”>
 <option value=”<%= tn %>”></option>
 <option value=”restorable<%= tn %>” <c:if
       test=”${fn:contains(fn:join(
       recurrentStatus.
       lingual_restorable_recurrent,’,’),
       restorablevar)}”>selected</c:if>> O </
       option>
 <option value=”nonrestorable<%= tn %>” <c:
       if test=”${fn:contains(fn:join(
       recurrentStatus.
       lingual_restorable_recurrent,’,’),
       nonrestorablevar)}”>selected</c:if>> /
       </option>
 </select></div></td>
 <td><div id=”restoration_surfaces<%= tn %>
       lingual” style=”display:none;”><input
       type=”checkbox” name=”
       lingualrestoration” value=”<%= tn %>”
       id=”restolingual<%= tn %>” onclick=”
       showvar(’lingual’+
       restoreSelectSurfaceid,
       restolingualformid);drawConditionMini(’
       restoration’, ’lingual’,<%= tn %>) ” <c
       :if test=”${fn:contains(fn:join(
       restorationStatus.
       lingual_restoration_string,’,’),tn)}”>
       checked=”yes”</c:if> /></div></td><td><
       div id=”lingualrestoreSelectSurface<%=
       tn %>” style=”display:none;”><select
       name=”selectRestorationLingual” id=”
       lingualrestoreTypeSelect<%= tn %>”>
<option value=”<%= tn %>”></option>
<option value=”AM<%= tn %>” <c:if test=”${fn
      :contains(fn:join(restorationStatus.
      lingual_restorable_restoration,’,’),
      amvar)}”>selected</c:if>>AM</option>
<option value=”CO<%= tn %>” <c:if test=”${fn
      :contains(fn:join(restorationStatus.
      lingual_restorable_restoration,’,’),
      covar)}”>selected</c:if>>CO</option>
<option value=”GI<%= tn %>” <c:if test=”${fn
      :contains(fn:join(restorationStatus.
      lingual_restorable_restoration,’,’),
      givar)}”>selected</c:if>>GI</option>
<option value=”TF<%= tn %>” <c:if test=”${fn
      :contains(fn:join(restorationStatus.
      lingual_restorable_restoration,’,’),
      tfvar)}”>selected</c:if>>TF</option>

</select></div></td></tr>
</table></div><hr>
<br><input type=”checkbox” name=”toothAll”
      value=”<%= tn %>” id=”alltooth<%= tn %>”
       onclick=”showAllTooth()” <c:if test=”${
      fn:contains(fn:join(dentalChart.
      extracted_string,’,’),tn) || fn:contains
      (fn:join(dentalChart.missing_string,’,’)
      ,tn) ||
fn:contains(fn:join(dentalChart.
      unerupted_string,’,’),tn) || fn:contains
      (fn:join(dentalChart.impacted_string
      ,’,’),tn) || fn:contains(fn:join(
      dentalChart.extrusion_string,’,’),tn) ||
fn:contains(fn:join(dentalChart.
      intrusion_string,’,’),tn) || fn:contains
      (fn:join(dentalChart.
      mesial_rotation_string,’,’),tn) || fn:
      contains(fn:join(dentalChart.
      distal_rotation_string,’,’),tn) ||
fn:contains(fn:join(dentalChart.
      rotation_string,’,’),tn) || fn:contains(
      fn:join(dentalChart.
      postcore_crown_string,’,’),tn) || fn:
      contains(fn:join(dentalChart.
      acrylic_crown_string,’,’),tn)||
fn:contains(fn:join(dentalChart.
      metal_crown_string,’,’),tn) || fn:
      contains(fn:join(dentalChart.
      porcelain_crown_string,’,’),tn) || fn:
      contains(fn:join(dentalChart.
      removable_partial_denture_string,’,’),tn
      ) ||
fn:contains(fn:join(dentalChart.
      fixed_bridge_string,’,’),tn) || fn:
      contains(fn:join(dentalChart.
      rootcanal_treatment_string,’,’),tn) ||
      fn:contains(fn:join(dentalChart.
      porcelain_infused_string,’,’),tn) ||
fn:contains(fn:join(dentalChart.
      pitfissure_sealants_string,’,’),tn)}”>
checked=”yes”</c:if> ><b>Whole Tooth</b>
<div id=”all_tooth_surfaces<%= tn %>” style
      =”display:none;”><table>


<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
      input type=”checkbox” name=”extrusion”
      value=”<%= tn %>” id=”extrusion<%= tn
      %>” onclick=”drawConditionMini2(’
      extrusion’,<%= tn %>)” <c:if test=”${fn:
      contains(fn:join(dentalChart.
      extrusion_string,’,’),tn)}”>checked=”yes
      ”</c:if>></td>
<td>Extrusion</td>
<td><input type=”checkbox” name=”intrusion”
      value=”<%= tn %>” id=”intrusion<%= tn
      %>” onclick=”drawConditionMini2(’
      intrusion’,<%= tn %>)” <c:if test=”${fn:
      contains(fn:join(dentalChart.
      intrusion_string,’,’),tn)}”>checked=”yes
      ”</c:if>></td>
<td>Intrusion</td>
<td><input type=”checkbox” name=”mesialdrift
      ” value=”<%= tn %>” id=”mesialdrift<%=
      tn %>” onclick=”drawConditionMini2(’
      mesialdrift’,<%= tn %>)” <c:if test=”${
      fn:contains(fn:join(dentalChart.
      mesial_rotation_string,’,’),tn)}”>
      checked=”yes”</c:if>></td>
<td>Mesial Drift Rotation</td>
<td><input type=”checkbox” name=”distaldrift
      ” value=”<%= tn %>” id=”distaldrift<%=
      tn %>” onclick=”drawConditionMini2(’
      distaldrift’,<%= tn %>)” <c:if test=”${
      fn:contains(fn:join(dentalChart.
      distal_rotation_string,’,’),tn)}”>
      checked=”yes”</c:if>></td>
<td>Distal Drift Rotation</td>
<td><input type=”checkbox” name=”rotation”
      value=”<%= tn %>” id=”rotation<%= tn %>”
       onclick=”drawConditionMini2(’rotation
      ’,<%= tn %>)” <c:if test=”${fn:contains(
      fn:join(dentalChart.rotation_string,’,’)
      ,tn)}”>checked=”yes”</c:if>></td>
<td>Rotation</td>
</tr>

<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
      input type=”checkbox” name=”
      removablepartial” value=”<%= tn %>” id=”
      removablepartial<%= tn %>” onclick=”
      drawConditionMini2(’removablepartial
      ’,<%= tn %>)” <c:if test=”${fn:contains(
      fn:join(dentalChart.
      removable_partial_denture_string,’,’),tn
      )}”>checked=”yes”</c:if>></td>
<td>Removable Partial Denture</td>
<td><input type=”checkbox” name=”fixedbridge
      ” value=”<%= tn %>” id=”fixedbridge<%=
      tn %>” onclick=”drawConditionMini2(’
      fixedbridge’,<%= tn %>)” <c:if test=”${
      fn:contains(fn:join(dentalChart.
      fixed_bridge_string,’,’),tn)}”>checked=”
      yes”</c:if>></td>
<td>Fixed Bridged</td>
<td><input type=”checkbox” name=”rootcanal”
      value=”<%= tn %>” id=”rootcanal<%= tn
      %>” onclick=”drawConditionMini2(’
     rootcanal',<%= tn %>)" <c:if test="${fn:
     contains(fn:join(dentalChart.
     rootcanal_treatment_string,','),tn)}">
     checked="yes"</c:if></td>
<td>Root Canal Treatment</td>
<td><input type="checkbox" name="
     porcelainfused" value="<%= tn %>" id="
     porcelainfused<%= tn %>" onclick="
     drawConditionMini2('porcelainfused',<%=
     tn %>)" <c:if test="${fn:contains(fn:
     join(dentalChart.porcelain_crown_string
     ,','),tn)}">checked="yes"</c:if></td>
<td>Porcelain Fused to Metal</td>
<td><input type="checkbox" name="pitfissure"
      value="<%= tn %>" id="pitfissure<%= tn
     %>" onclick="drawConditionMini2('
     pitfissure',<%= tn %>)" <c:if test="${fn
     :contains(fn:join(dentalChart.
     pitfissure_sealants_string,','),tn)}">
     checked="yes"</c:if></td>
<td>Pit and Fissure Sealants</td>

</tr>


<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
     input type="checkbox" name="postcore"
     value="<%= tn %>" id="postcore<%= tn %>"
      onclick="drawConditionMini2('postcore
     ',<%= tn %>)" <c:if test="${fn:contains(
     fn:join(dentalChart.
     postcore_crown_string,','),tn)}">checked
     ="yes"</c:if></td>
<td>Post Core Crown</td>
<td><input type="checkbox" name="acrylic"
     value="<%= tn %>" id="acrylic<%= tn %>"
     onclick="drawConditionMini2('acrylic
     ',<%= tn %>)" <c:if test="${fn:contains(
     fn:join(dentalChart.acrylic_crown_string
     ,','),tn)}">checked="yes"</c:if></td>
<td>Acrylic Crown</td>
<td><input type="checkbox" name="metal"
     value="<%= tn %>" id="metal<%= tn %>"
     onclick="drawConditionMini2('metal',<%=
     tn %>)" <c:if test="${fn:contains(fn:
     join(dentalChart.metal_crown_string,',')
     ,tn)}">checked="yes"</c:if></td>
<td>Metal Crown</td>
<td><input type="checkbox" name="porcelain"
     value="<%= tn %>" id="porcelain<%= tn
     %>" onclick="drawConditionMini2('
     porcelain',<%= tn %>)" <c:if test="${fn:
     contains(fn:join(dentalChart.
     porcelain_crown_string,','),tn)}">
     checked="yes"</c:if></td>
<td>Porcelain Crown</td>

</tr>



<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
     input type="checkbox" name="extracted"
     value="<%= tn %>" id="extracted<%= tn
     %>" onclick="drawConditionMini2('
     extracted',<%= tn %>)" <c:if test="${fn:
     contains(fn:join(dentalChart.
     extracted_string,','),tn)}">checked="yes
     "</c:if></td>
<td>Extracted</td>
<td><input type="checkbox" name="missing"
     value="<%= tn %>" id="missing<%= tn %>"
     onclick="drawConditionMini2('missing
     ',<%= tn %>)" <c:if test="${fn:contains(
     fn:join(dentalChart.missing_string,','),
     tn)}">checked="yes"</c:if></td>
<td>Missing</td>
<td><input type="checkbox" name="unerupted"
     value="<%= tn %>" id="unerupted<%= tn
     %>" onclick="drawConditionMini2('
     unerupted',<%= tn %>)" <c:if test="${fn:
     contains(fn:join(dentalChart.
     unerupted_string,','),tn)}">checked="yes
     "</c:if></td>
<td>Unerupted</td>
<td><input type="checkbox" name="impacted"
     value="<%= tn %>" id="impacted<%= tn %>"
      onclick="drawConditionMini2('impacted
     ',<%= tn %>)" <c:if test="${fn:contains(
     fn:join(dentalChart.impacted_string,',')
     ,tn)}">checked="yes"</c:if></td>
<td>Impacted</td>
</tr>


</table></div>


 <hr>
  <a href = "javascript:void(0)" onclick = "
       submitcond()">Submit</a>
  <a href = "javascript:void(0)" onclick = "
       document.getElementById('light<%= tn
       %>').style.display='none';document.
       getElementById('fade<%= tn %>').style.
       display='none'">Cancel</a>

  <br/>
  <br/>
  <b>Services needed</b>
  <br/>
      <hr>
  <u>Operative Dentistry</u><br/>
  <input type="checkbox" name="class1" id="
       class1<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_1_string,','),tn)}">
       checked="yes"</c:if>>Class I
  <input type="checkbox" name="class2" id="
       class2<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_2_string,','),tn)}">
       checked="yes"</c:if>>Class II
  <input type="checkbox" name="class3" id="
       class3<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_3_string,','),tn)}">
       checked="yes"</c:if>>Class III
  <input type="checkbox" name="class4" id="
       class4<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_4_string,','),tn)}">
       checked="yes"</c:if>>Class IV
  <input type="checkbox" name="class5" id="
       class5<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_5_string,','),tn)}">
       checked="yes"</c:if>>Class V
  <input type="checkbox" name="onlay" id="
       onlay<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.onlay_string,','),tn)}">
       checked="yes"</c:if>>Onlay
  <br/><br/>
  <u> Surgery </u><br/>
  <input type="checkbox" name="extraction"
       id="extractss<%= tn %>" value="<%= tn
       %>" <c:if test="${fn:contains(fn:join(
       services.extraction_string,','),tn)}">
       checked="yes"</c:if>>Extraction
  <input type="checkbox" name="odontectomy"
       id="odontectomy<%= tn %>" value="<%=
       tn %>" <c:if test="${fn:contains(fn:
       join(services.odontectomy_string,','),
       tn)}">checked="yes"</c:if>>Odontectomy
  <input type="checkbox" name="specialcase"
       id="specialcase<%= tn %>" value="<%=
       tn %>" <c:if test="${fn:contains(fn:
       join(services.special_case_string,',')
       ,tn)}">checked="yes"</c:if>>Special
       Case
  <br/><br/>
  <u> Emergency Treatment </u><br/>
  <input type="checkbox" name="pulpsedation"
        id="pulpsedation<%= tn %>" value="<%=
        tn %>" <c:if test="${fn:contains(fn:
       join(services.pulp_sedation_string
       ,','),tn)}">checked="yes"</c:if>>Pulp
       Sedation
  <input type="checkbox" name="
       crownrecementation" id="
       crownrecementation<%= tn %>" value
       ="<%= tn %>" <c:if test="${fn:contains
       (fn:join(services.
       crown_recementation_string,','),tn)}">
       checked="yes"</c:if>>Recementation of
       crowns
  <input type="checkbox" name="
       fillingservice" id="fillingservice<%=
       tn %>" value="<%= tn %>" <c:if test="$
       {fn:contains(fn:join(services.
       filling_service_string,','),tn)}">
       checked="yes"</c:if>>Temporary
       fillings
  <br/><br/>
  <u> Fixed Partial Dentures </u><br/>
  <input type=”checkbox” name=”laminated” id
       =”laminated<%= tn %>” value=”<%= tn
       %>” <c:if test=”${fn:contains(fn:join(
       services.laminated_string,',') ,tn)}”>
       checked=”yes”</c:if>>Laminated
  <input type=”checkbox” name=”singlecrown”
       id=”singlecrown<%= tn %>” value=”<%=
       tn %>” <c:if test=”${fn:contains(fn:
       join(services.single_crown_string,',')
       ,tn)}”>checked=”yes”</c:if>>Single
       Crown
  <input type=”checkbox” name=”bridgeservice
       ” id=”bridgeservice<%= tn %>” value
       =”<%= tn %>” <c:if test=”${fn:contains
       (fn:join(services.
       bridge_service_string,',') ,tn)}”>
       checked=”yes”</c:if>>Bridge
  <br/><br/>
  <u> Endodontics </u><br/>
  <input type=”checkbox” name=”anterior” id
       =”anterior<%= tn %>” value=”<%= tn %>”
        <c:if test=”${fn:contains(fn:join(
       services.anterior_string,',') ,tn)}”>
       checked=”yes”</c:if>>Anterior
  <input type=”checkbox” name=”posterior” id
       =”posterior<%= tn %>” value=”<%= tn
       %>” <c:if test=”${fn:contains(fn:join(
       services.posterior_string,',') ,tn)}”>
       checked=”yes”</c:if>>Posterior
  <input type=”checkbox” name=”
       otherendodontics” id=”otherendodontics
       <%= tn %>” value=”<%= tn %>” <c:if
       test=”${fn:contains(fn:join(services.
       ortho_endo_string,',') ,tn)}”>checked=”
       yes”</c:if>>Others (Endosurgery,
       Bleaching, etc.)
  <br/><br/>
  <hr>
  <a href = ”javascript:void(0)” onclick = ”
       submitcond()”>Submit</a>
  <a href = ”javascript:void(0)” onclick = ”
       document.getElementById('light<%= tn
       %>').style.display='none';document.
       getElementById('fade<%= tn %>').style.
       display='none'”>Cancel</a>


</div><div id=”fade<%= tn %>” class=”
     black_overlay”></div>

<%
  }
 }
%>

<div id=”serviceslight” class=”white_content
     ”>
 <font size = ”2”>
 <b>Other Services</b><br/>
 <hr>
 <u>Periodontics</u><br/>
 <input type=”checkbox” name=”periodontics”
      id=”periodontics” value=”yes” <c:if
      test=”${services.periodontics eq 'yes
      '}”>checked=”yes”</c:if>>Management of
      Periodontal Disease
 <br/><br/>
 <u> Surgery </u><br/>
 <input type=”checkbox” name=”surgery” id=”
      pedodontics” value=”pedodontics” <c:if
      test=”${services.surgery eq '
      pedodontics'}”>checked=”yes”</c:if>>
      Pedodontics <br/>
 <input type=”checkbox” name=”surgery” id=”
      orthodontics” value=”orthodontics” <c:
      if test=”${services.surgery eq '
      orthodontics'}”>checked=”yes”</c:if>>
      Orthodontics
 <br/><br/>
 <u> Emergency Treatment </u><br/>
 <input type=”checkbox” name=”
      emergencytreatment” id=”acuteinfections
      ” value=”acute infections” <c:if test=”
      ${services.emergency_treatment eq '
      acute infections'}”>checked=”yes”</c:if
      >>Management of Acute Infections <br/>
 <input type=”checkbox” name=”
      emergencytreatment” id=”
      traumaticinjuries” value=”traumatic
      injuries” <c:if test=”${services.
      emergency_treatment eq 'traumatic
      injuries'}”>checked=”yes”</c:if>>
      Management of Temporary Injuries
 <br/><br/>
 <u> Prosthodontics </u><br/>
 <input type=”checkbox” name=”prosthodontics
      ” id=”completedent” value=”complete
      denture” <c:if test=”${services.
      prosthodontics eq 'complete denture'}”>
      checked=”yes”</c:if>>Complete Denture<
      br/>
 <input type=”checkbox” name=”prosthodontics
      ” id=”singledent” value=”single denture
      ” <c:if test=”${services.prosthodontics
       eq 'single denture'}”>checked=”yes”</c
      :if>>Single Denture<br/>
 <input type=”checkbox” name=”prosthodontics
      ” id=”removedent” value=”removable
      partial” <c:if test=”${services.
      prosthodontics eq 'removable partial
      '}”>checked=”yes”</c:if>>Removable
      Partial Denture<br/>
 <input type=”checkbox” name=”prosthodontics
      ” id=”otherss” value=”others” <c:if
      test=”${services.prosthodontics eq '
      others'}”>checked=”yes”</c:if>>Other
      Denture Services
 <br/><br/>
 <hr>
 <a href = ”javascript:void(0)” onclick = ”
      document.getElementById('serviceslight
      ').style.display='none';document.
      getElementById('servicesfade').style.
      display='none'”>Done</a>

 </font>
</div>
<div id=”servicesfade” class=”black_overlay
     ”></div>



<div id=”noteslight” class=”white_content”>
 <font size = ”2”>
 <b>Notes</b><br/>
 <hr>
 <br/>
 <textarea name=”notes” rows=”6” cols
      =”100”></textarea>
 <br/><br/>
 <hr>
 <a href = ”javascript:void(0)” onclick = ”
      document.getElementById('noteslight').
      style.display='none';document.
      getElementById('notesfade').style.
      display='none'”>Submit</a>
 <a href = ”javascript:void(0)” onclick = ”
      document.getElementById('noteslight').
      style.display='none';document.
      getElementById('notesfade').style.
      display='none'”>Cancel</a>

 </font>
</div>

<div id=”neededlight” class=”white_content”>
<b>Service Needed Summary</b> &nbsp;&nbsp
     ;||&nbsp;&nbsp;
<a href = ”javascript:void(0)” onclick = ”
     clearneeded_service();”>Cancel</a>

<table>
<tr>
<td>
<u><div id=”periodonticsdiv”></div></u>
<div id=”periodiv”></div>
</td>

<td>
<u><div id=”emergencytreatmentdiv”></div></u
     >
<div id=”pulpsedationdiv”></div>
<div id=”crownrecementationdiv”></div>
<div id=”fillingservicediv”></div>
<div id=”acuteinfectionsdiv”></div>
<div id=”traumaticdiv”></div>

</td>
</tr>

<tr>
<td>
```

```
<u><div id="operativedentistrydiv"></div></u
    >
<div id="class1div"></div>
<div id="class2div"></div>
<div id="class3div"></div>
<div id="class4div"></div>
<div id="class5div"></div>
<div id="onlaydiv"></div>
</td>

<td>
<u><div id="fixedpartialdenturediv"></div></
    u>
<div id="laminateddiv"></div>
<div id="singlecrowndiv"></div>
<div id="bridgeservicediv"></div>
<br/>
<u><div id="endodonticsdiv"></div></u>
<div id="anteriordiv"></div>
<div id="posteriordiv"></div>
<div id="othersdiv"></div>
</td>
</tr>



 <style type="text/css">
.black_overlay{
 display: none;
 position: absolute;
 top: 0%;
 left: 5%;
 width: 100%;
 height: 120%;
 background-color: black;
 z-index:1001;
 -moz-opacity: 0.2;
 opacity:.20;
 filter: alpha(opacity=20);
}
.white_content {
 display: none;
 position: absolute;
 top: 10%;
 left: 15%;
 width: 70%;
 height: 55%;
 padding: 16px;
 border: 5px solid #1aac9b;
 background-color: white;
 z-index:1002;
 overflow: auto;
}
</style>
<script type="text/javascript"> init(65, 65,
     627);     </script>


<div id="dentureslight" class="white_content
    ">
 <font size = "2">
 <b>Dentures Status</b><br/>
 <hr>
 <input type="checkbox" name="
      completedenture" value="yes" id="
      completedenture" onclick="
      drawCompletedenture(ctx1, ctx30, 65,
      65)" <c:if test="${dentalChart.
      complete_denture eq 'yes'}">checked="
      yes"</c:if>>Complete Denture
 <br/><br/>
 <u> Single Denture </u><br/>
 <input type="checkbox" name="singledenture"
       value="upper" id="upperdenture"
      onclick="drawUpperDenture(ctx2, ctx31,
      65, 65)" <c:if test="${dentalChart.
      single_denture eq 'upper'}">checked="
      yes"</c:if>>Upper Single Denture <br/>
 <input type="checkbox" name="singledenture"
       value="lower" id="lowerdenture"
      onclick="drawLowerDenture(ctx3, ctx32,
      65, 65)" <c:if test="${dentalChart.
      single_denture eq 'lower'}">checked="
      yes"</c:if>>Lower Single Denture <br/>
 <hr>
 <a href = "javascript:void(0)" onclick = "
      document.getElementById('dentureslight
      ').style.display='none';document.
      getElementById('denturesfade').style.
      display='none'">Done</a>
```

```
<tr>
<td>
<u><div id="surgerydiv"></div></u>
<div id="extractiondiv"></div>
<div id="odontectomydiv"></div>
<div id="specialcasediv"></div>
<div id="pedonticsdiv"></div>
<div id="orthodonticsdiv"></div>
</td>

<td>
<u><div id="prosthodonticsdiv"></div></u>
<div id="completedentdiv"></div>
<div id="singledentdiv"></div>
<div id="removedentdiv"></div>
<div id="otherssdiv"></div>
</td>
</tr>
</table>

</div>

 <script type="text/javascript"> init(65,
     65, 627);     </script>


 </font>
</div>




 <%


 int i = 18;
 int j = 1;
 int tn = 18;
 for(j = 0; j < 4; j++) {
  for(i = 18; i > 10; i--) {
   tn = i+j*10;
   String restorableVar= "restorable"+tn;
   String nonrestorableVar="nonrestorable"+tn
       ;
   String amVar="AM"+tn;
   String coVar="CO"+tn;
   String giVar="GI"+tn;
   String tfVar="TF"+tn;




 %>
   <c:set var="tn" value="<%= tn %>"/>
    <c:set var="restorablevar" value="<%=
        restorableVar %>"/>
   <c:set var="nonrestorablevar" value="<%=
       nonrestorableVar %>"/>
   <c:set var="amvar" value="<%= amVar %>"/>
   <c:set var="covar" value="<%= coVar %>"/>
   <c:set var="givar" value="<%= giVar %>"/>
   <c:set var="tfvar" value="<%= tfVar %>"/>




<div id="light<%= tn %>" class="
     white_content" name="light<%= tn %>">
<div id="canvasesdiv" >
<canvas id="layer1mini<%= tn %>" style="
     border:solid 2px; z-index: 1; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer2mini<%= tn %>" style="
     border:solid 2px; z-index: 2; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer3mini<%= tn %>" style="
     border:solid 2px; z-index: 3; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer4mini<%= tn %>" style="
     border:solid 2px; z-index: 4; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer5mini<%= tn %>" style="
     border:solid 2px; z-index: 5; position:
     absolute; right:20px; top:20px;" height
     ="125px" width="125"></canvas>
<canvas id="layer6mini<%= tn %>" style="
     border:solid 2px; z-index: 6; position:
     absolute; right:20px; top:20px;" height
     =”125px” width=”125”></canvas>
<canvas id=”layer7mini<%= tn %>” style=”
     border:solid 2px; z-index: 7; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer8mini<%= tn %>” style=”
     border:solid 2px; z-index: 8; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer9mini<%= tn %>” style=”
     border:solid 2px; z-index: 9; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer10mini<%= tn %>” style=”
     border:solid 2px; z-index: 10; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer11mini<%= tn %>” style=”
     border:solid 2px; z-index: 11; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer12mini<%= tn %>” style=”
     border:solid 2px; z-index: 12; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer13mini<%= tn %>” style=”
     border:solid 2px; z-index: 13; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer14mini<%= tn %>” style=”
     border:solid 2px; z-index: 14; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer15mini<%= tn %>” style=”
     border:solid 2px; z-index: 15; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer16mini<%= tn %>” style=”
     border:solid 2px; z-index: 16; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas>
<canvas id=”layer17mini<%= tn %>” style=”
     border:solid 2px; z-index: 17; position:
     absolute; right:20px; top:20px;” height
     =”125px” width=”125”></canvas></div>
<b>Tooth <%= tn %>  </b><br/><br/><font
     color=”red”><div id=”toothvar<%= tn
     %>”></div></font><br/><br/><a href = ”
     javascript:void(0)” onclick = ”
     submitcond()”>Submit</a>&nbsp;
<a href = ”javascript:void(0)” onclick = ”
     cancelcond(<%= tn %>)”>Cancel</a><br/> <
     br/>
<br><input type=”checkbox” name=”
     dentalStatus” value=”<%= tn %>” id=”
     dentalStatus<%= tn %>” onclick=”
     hideAllTooth()” <c:if test=”${fn:
     contains(fn:join(dentalChart.
     caries_string,’,’),tn) || fn:contains(fn
     :join(dentalChart.recurrentcaries_string
     ,’,’),tn) || fn:contains(fn:join(
     dentalChart.restoration_string,’,’),tn)
     }”>checked=”yes”</c:if>><b>Dental Status
     </b><br/>


<div id=”dentalstatussurface<%= tn %>” style
     =”display:none;”><table><tr><td></td><td
     ><input type=’checkbox’ name=’caries’
     value=’<%= tn %>’ id=’caries<%= tn %>’
     onclick=’hideOtherCaries()’  <c:if test
     =”${fn:contains(fn:join(dentalChart.
     caries_string,’,’),tn)}”>checked=”yes”</
     c:if>></td><td>Caries</td>

<td><input type=”checkbox” name=”
     recurrentcaries” value=”<%= tn %>” id=”
     recurrentcaries<%= tn %>”  onclick=”
     hideOtherReccurent()”  <c:if test=”${fn:
     contains(fn:join(dentalChart.
     recurrentcaries_string,’,’),tn)}”>
     checked=”yes”</c:if>></td><td>Reccurent
     </td>

<td><input type=”checkbox” name=”restoration
     ” value=”<%= tn %>” id=”restoration<%=
     tn %>” onclick=”hideOtherRestoration()”
      <c:if test=”${fn:contains(fn:join(
     dentalChart.restoration_string,’,’),tn)
     }”>checked=”yes”</c:if>></td><td>
     Restoration</td>
</tr>
```

```
<tr><td>Mesial</td><td><div id=”
     caries_surfaces<%= tn %>mesial” style=”
     display:none;”><input type=”checkbox”
     name=”mesialcaries” value=”<%= tn %>” id
     =”mesial<%= tn %>” onclick=”showvar(’
     mesial’+cariesSelectSurfaceid,
     mesialformid);drawConditionMini(’caries
     ’, ’mesial’,<%= tn %>) ”  <c:if test=”${
     fn:contains(fn:join(cariesStatus.
     mesial_caries_string,’,’),tn)}”>checked
     =”yes”</c:if>/></div></td>
<td><div id=”mesialcariesSelectSurface<%= tn
      %>” style=”display:none;”><select name
     =”selectCariesMesial”  id=”
     mesialcariesSelect<%= tn %>”>
<option value=”<%= tn %>”></option>
<option value=”restorable<%= tn %>” <c:if
     test=”${fn:contains(fn:join(cariesStatus
     .mesial_restorable_caries,’,’),
     restorablevar)}”>selected</c:if>> O </
     option>
<option value=”nonrestorable<%= tn %>” <c:if
      test=”${fn:contains(fn:join(
     cariesStatus.mesial_restorable_caries
     ,’,’),nonrestorablevar)}”>selected</c:if
     >> / </option>
</select></div></td>
 <td><div id=”recurrent_surfaces<%= tn %>
      mesial” style=”display:none;”><input
      type=”checkbox” name=”mesialrecurrent”
      value=”<%= tn %>” id=”remesial<%= tn
      %>” onclick=”showvar(’mesial’+
      recurrentSelectSurfaceid,remesialformid
      );drawConditionMini(’recurrent’, ’
      mesial’,<%= tn %>) ”  <c:if test=”${fn:
      contains(fn:join(recurrentStatus.
      mesial_recurrent_string,’,’),tn)}”>
      checked=”yes”</c:if>/></div></td>
 <td><div id=”mesialrecurrentSelectSurface
      <%= tn %>” style=”display:none;”><
      select name=”selectRecurrentMesial” id
      =”mesialrecurrentSelect<%= tn %>”>
 <option value=”<%= tn %>”></option><option
      value=”restorable<%= tn %>” <c:if test
      =”${fn:contains(fn:join(recurrentStatus
      .mesial_restorable_recurrent,’,’),
      restorablevar)}”>selected</c:if>> O </
      option>
 <option value=”nonrestorable<%= tn %>” <c:
      if test=”${fn:contains(fn:join(
      recurrentStatus.
      mesial_restorable_recurrent,’,’),
      nonrestorablevar)}”>selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id=”restoration_surfaces<%= tn %>
      mesial” style=”display:none;”><input
      type=”checkbox” name=”mesialrestoration
      ” value=”<%= tn %>” id=”restomesial<%=
      tn %>” onclick=”showvar(’mesial’+
      restoreSelectSurfaceid,
      restomesialformid);drawConditionMini(’
      restoration’, ’mesial’,<%= tn %>) ”  <c
      :if test=”${fn:contains(fn:join(
      restorationStatus.
      mesial_restoration_string,’,’),tn)}”>
      checked=”yes”</c:if>/></div></td><td><
      div id=”mesialrestoreSelectSurface<%=
      tn %>” style=”display:none;”><select
      name=”selectRestorationMesial” id=”
      mesialrestoreTypeSelect<%= tn %>”>
<option value=”<%= tn %>” ></option>
<option value=”AM<%= tn %>” <c:if test=”${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,’,’),amvar
     )}”>selected</c:if>>AM</option>
<option value=”CO<%= tn %>” <c:if test=”${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,’,’),covar
     )}”>selected</c:if>>CO</option>
<option value=”GI<%= tn %>” <c:if test=”${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,’,’),givar
     )}”>selected</c:if>>GI</option>
<option value=”TF<%= tn %>” <c:if test=”${fn
     :contains(fn:join(restorationStatus.
     mesial_restorable_restoration,’,’),tfvar
     )}”>selected</c:if>>TF</option>
</select></div></td></tr>
```

```
<tr><td>Distal</td><td><div id="
      caries_surfaces<%= tn %>distal" style="
      display:none;"><input type="checkbox"
      name="distalcaries" value="<%= tn %>" id
      ="distal<%= tn %>" onclick="showvar('
      distal'+cariesSelectSurfaceid,
      distalformid);drawConditionMini('caries
      ', 'distal', <%= tn %> )" <c:if test="${
      fn:contains(fn:join(cariesStatus.
      distal_caries_string,','),tn)}">checked
      ="yes"</c:if>/></div></td>
<td><div id="distalcariesSelectSurface<%= tn
      %>" style="display:none;">
<select name="selectCariesDistal" id="
      distalcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
        test="${fn:contains(fn:join(
        cariesStatus.distal_restorable_caries
        ,','),restorablevar)}">selected</c:if
        >> O </option>
  <option value="nonrestorable<%= tn %>" <c:
        if test="${fn:contains(fn:join(
        cariesStatus.distal_restorable_caries
        ,','),nonrestorablevar)}">selected</c:
        if>> / </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
       distal" style="display:none;"><input
       type="checkbox" name="distalrecurrent"
       value="<%= tn %>" id="redistal<%= tn
       %>" onclick="showvar('distal'+
       recurrentSelectSurfaceid,redistalformid
       );drawConditionMini('recurrent', '
       distal',<%= tn %>) " <c:if test="${fn:
       contains(fn:join(recurrentStatus.
       distal_recurrent_string,','),tn)}">
       checked="yes"</c:if> /></div></td>
 <td><div id="distalrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentDistal" id="
       distalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option>
 <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       recurrentStatus.
       distal_restorable_recurrent,','),
       restorablevar)}">selected</c:if>> O </
       option>
 <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       recurrentStatus.
       distal_restorable_recurrent,','),
       nonrestorablevar)}">selected</c:if>> /
       </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
       distal" style="display:none;"><input
       type="checkbox" name="distalrestoration
       " value="<%= tn %>" id="restodistal<%=
       tn %>" onclick="showvar('distal'+
       restoreSelectSurfaceid,
       restodistalformid);drawConditionMini('
       restoration', 'distal',<%= tn %>) " <c:
       if test="${fn:contains(fn:join(
       restorationStatus.
       distal_restoration_string,','),tn)}">
       checked="yes"</c:if> /></div></td><td><
       div id="distalrestoreSelectSurface<%=
       tn %>" style="display:none;"><select
       name="selectRestorationDistal" id="
       distalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      distal_restorable_restoration,','),amvar
      )}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      distal_restorable_restoration,','),covar
      )}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      distal_restorable_restoration,','),givar
      )}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      distal_restorable_restoration,','),tfvar
      )}">selected</c:if>>TF</option>

</select></div></td></tr>
```

```
<tr><td>Occlusal</td><td><div id="
      caries_surfaces<%= tn %>occlusal" style
      ="display:none;"><input type="checkbox"
      name="occlusalcaries" value="<%= tn %>"
      id="occlusal<%= tn %>" onclick="showvar
      ('occlusal'+cariesSelectSurfaceid,
      occlusalformid);drawConditionMini('
      caries', 'occlusal',<%= tn %>) " <c:if
      test="${fn:contains(fn:join(cariesStatus
      .occlusal_caries_string,','),tn)}">
      checked="yes"</c:if> /></div></td>
<td><div id="occlusalcariesSelectSurface<%=
      tn %>" style="display:none;">
<select name="selectCariesOcclusal" id="
      occlusalcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
        test="${fn:contains(fn:join(
        cariesStatus.
        occlusal_restorable_caries,','),
        restorablevar)}">selected</c:if>> O </
        option>
  <option value="nonrestorable<%= tn %>" <c:
        if test="${fn:contains(fn:join(
        cariesStatus.
        occlusal_restorable_caries,','),
        nonrestorablevar)}">selected</c:if>> /
         </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
       occlusal" style="display:none;"><input
       type="checkbox" name="occlusalrecurrent
       " value="<%= tn %>" id="reocclusal<%=
       tn %>" onclick="showvar('occlusal'+
       recurrentSelectSurfaceid,
       reocclusalformid);drawConditionMini('
       recurrent', 'occlusal',<%= tn %>) " <c
       :if test="${fn:contains(fn:join(
       recurrentStatus.
       occlusal_recurrent_string,','),tn)}">
       checked="yes"</c:if>/></div></td>
 <td><div id="occlusalrecurrentSelectSurface
       <%= tn %>" style="display:none;">
 <select name="selectRecurrentOcclusal" id="
       occlusalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option><option
       value="restorable<%= tn %>" <c:if test
       ="${fn:contains(fn:join(recurrentStatus
       .occlusal_restorable_recurrent,','),
       restorablevar)}">selected</c:if>> O </
       option>
 <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       recurrentStatus.
       occlusal_restorable_recurrent,','),
       nonrestorablevar)}">selected</c:if>> /
       </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
       occlusal" style="display:none;"><input
       type="checkbox" name="
       occlusalrestoration" value="<%= tn %>"
       id="restoocclusal<%= tn %>" onclick="
       showvar('occlusal'+
       restoreSelectSurfaceid,
       restoocclusalformid);drawConditionMini
       ('restoration', 'occlusal',<%= tn %>) "
         <c:if test="${fn:contains(fn:join(
       restorationStatus.
       occlusal_restoration_string,','),tn)}">
       checked="yes"</c:if>/></div></td><td><
       div id="occlusalrestoreSelectSurface<%=
        tn %>" style="display:none;"><select
       name="selectRestorationOcclusal" id="
       occlusalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      occlusal_restorable_restoration,','),
      amvar)}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      occlusal_restorable_restoration,','),
      covar)}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
      occlusal_restorable_restoration,','),
      givar)}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
      :contains(fn:join(restorationStatus.
     occlusal_restorable_restoration,','),
     tfvar)}">selected</c:if>>TF</option>

</select></div></td></tr>



<tr><td>Buccal</td><td><div id="
     caries_surfaces<%= tn %>buccal" style="
     display:none;"><input type="checkbox"
     name="buccalcaries" value="<%= tn %>" id
     ="buccal<%= tn %>" onclick="showvar('
     buccal'+cariesSelectSurfaceid,
     buccalformid);drawConditionMini('caries
     ', 'buccal',<%= tn %>) " <c:if test="${
     fn:contains(fn:join(cariesStatus.
     buccal_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div></td><td><div id="
     buccalcariesSelectSurface<%= tn %>"
     style="display:none;"><select name="
     selectCariesBuccal" id="
     buccalcariesSelect<%= tn %>"><option
     value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.buccal_restorable_caries
       ,','),restorablevar)}">selected</c:if
       >> O </option>
  <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       cariesStatus.buccal_restorable_caries
       ,','),nonrestorablevar)}">selected</c:
       if>> / </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      buccal" style="display:none;"><input
      type="checkbox" name="buccalrecurrent"
      value="<%= tn %>" id="rebuccal<%= tn
      %>" onclick="showvar('buccal'+
      recurrentSelectSurfaceid,rebuccalformid
      );drawConditionMini('recurrent', '
      buccal',<%= tn %>) "  <c:if test="${fn:
      contains(fn:join(recurrentStatus.
      buccal_recurrent_string,','),tn)}">
      checked="yes"</c:if>/></div></td>
 <td><div id="buccalrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentBuccal" id="
      buccalrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option>
 <option value="restorable<%= tn %>" <c:if
      test="${fn:contains(fn:join(
      recurrentStatus.
      buccal_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      buccal_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      buccal" style="display:none;"><input
      type="checkbox" name="buccalrestoration
      " value="<%= tn %>" id="restobuccal<%=
      tn %>" onclick="showvar('buccal'+
      restoreSelectSurfaceid,
      restobuccalformid);drawConditionMini('
      restoration', 'buccal',<%= tn %>) "  <c
      :if test="${fn:contains(fn:join(
      restorationStatus.
      buccal_restoration_string,','),tn)}">
      checked="yes"</c:if>/></div></td><td><
      div id="buccalrestoreSelectSurface<%=
      tn %>" style="display:none;"><select
      name="selectRestorationBuccal" id="
      buccalrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),amvar
     )}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),covar
     )}">selected</c:if>>CO</option>
<option value="GI<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),givar
     )}">selected</c:if>>GI</option>
<option value="TF<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     buccal_restorable_restoration,','),tfvar
     )}">selected</c:if>>TF</option>

</select></div></td></tr>



<tr><td>Lingual</td><td><div id="
     caries_surfaces<%= tn %>lingual" style="
     display:none;"><input type="checkbox"
     name="lingualcaries" value="<%= tn %>"
     id="lingual<%= tn %>" onclick="showvar('
     lingual'+cariesSelectSurfaceid,
     lingualformid);drawConditionMini('caries
     ', 'lingual',<%= tn %>) "  <c:if test="$
     {fn:contains(fn:join(cariesStatus.
     lingual_caries_string,','),tn)}">checked
     ="yes"</c:if>/></div> </td>
<td><div id="lingualcariesSelectSurface<%=
     tn %>" style="display:none;"><select
     name="selectCariesLingual"  id="
     lingualcariesSelect<%= tn %>">
<option value="<%= tn %>"></option>
  <option value="restorable<%= tn %>" <c:if
       test="${fn:contains(fn:join(
       cariesStatus.lingual_restorable_caries
       ,','),restorablevar)}">selected</c:if
       >> O </option>
  <option value="nonrestorable<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       cariesStatus.lingual_restorable_caries
       ,','),nonrestorablevar)}">selected</c:
       if>> / </option>
 </select></div></td>
 <td><div id="recurrent_surfaces<%= tn %>
      lingual" style="display:none;"><input
      type="checkbox" name="lingualrecurrent"
       value="<%= tn %>" id="relingual<%= tn
      %>" onclick="showvar('lingual'+
      recurrentSelectSurfaceid,
      relingualformid);drawConditionMini('
      recurrent', 'lingual',<%= tn %>) " <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      lingual_recurrent_string,','),tn)}">
      checked="yes"</c:if>/></div></td>
 <td><div id="lingualrecurrentSelectSurface
      <%= tn %>" style="display:none;">
 <select name="selectRecurrentLingual" id="
      lingualrecurrentSelect<%= tn %>">
 <option value="<%= tn %>"></option>
 <option value="restorable<%= tn %>" <c:if
      test="${fn:contains(fn:join(
      recurrentStatus.
      lingual_restorable_recurrent,','),
      restorablevar)}">selected</c:if>> O </
      option>
 <option value="nonrestorable<%= tn %>" <c:
      if test="${fn:contains(fn:join(
      recurrentStatus.
      lingual_restorable_recurrent,','),
      nonrestorablevar)}">selected</c:if>> /
      </option>
 </select></div></td>
 <td><div id="restoration_surfaces<%= tn %>
      lingual" style="display:none;"><input
      type="checkbox" name="
      lingualrestoration" value="<%= tn %>"
      id="restolingual<%= tn %>" onclick="
      showvar('lingual'+
      restoreSelectSurfaceid,
      restolingualformid);drawConditionMini('
      restoration', 'lingual',<%= tn %>) " <c
      :if test="${fn:contains(fn:join(
      restorationStatus.
      lingual_restoration_string,','),tn)}">
      checked="yes"</c:if> /></div></td><td><
      div id="lingualrestoreSelectSurface<%=
      tn %>" style="display:none;"><select
      name="selectRestorationLingual" id="
      lingualrestoreTypeSelect<%= tn %>">
<option value="<%= tn %>"></option>
<option value="AM<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
     lingual_restorable_restoration,','),
     amvar)}">selected</c:if>>AM</option>
<option value="CO<%= tn %>" <c:if test="${fn
     :contains(fn:join(restorationStatus.
        lingual_restorable_restoration ,’,’),
        covar)}”>selected</c:if>>CO</option>
<option value=”GI<%= tn %>” <c:if test=”${fn
        :contains(fn:join(restorationStatus.
        lingual_restorable_restoration ,’,’),
        givar)}”>selected</c:if>>GI</option>
<option value=”TF<%= tn %>” <c:if test=”${fn
        :contains(fn:join(restorationStatus.
        lingual_restorable_restoration ,’,’),
        tfvar)}”>selected</c:if>>TF</option>

</select></div></td></tr>
</table></div><hr>
<br><input type=”checkbox” name=”toothAll”
        value=”<%= tn %>” id=”alltooth<%= tn %>”
         onclick=”showAllTooth()” <c:if test=”${
        fn:contains(fn:join(dentalChart.
        extracted_string ,’,’),tn) || fn:contains
        (fn:join(dentalChart.missing_string ,’,’)
        ,tn) ||
fn:contains(fn:join(dentalChart.
        unerupted_string ,’,’),tn) || fn:contains
        (fn:join(dentalChart.impacted_string
        ,’,’),tn) || fn:contains(fn:join(
        dentalChart.extrusion_string ,’,’),tn) ||
fn:contains(fn:join(dentalChart.
        intrusion_string ,’,’),tn) || fn:contains
        (fn:join(dentalChart.
        mesial_rotation_string ,’,’),tn) || fn:
        contains(fn:join(dentalChart.
        distal_rotation_string ,’,’),tn) ||
fn:contains(fn:join(dentalChart.
        rotation_string ,’,’),tn) || fn:contains(
        fn:join(dentalChart.
        postcore_crown_string ,’,’),tn) || fn:
        contains(fn:join(dentalChart.
        acrylic_crown_string ,’,’),tn)||
fn:contains(fn:join(dentalChart.
        metal_crown_string ,’,’),tn) || fn:
        contains(fn:join(dentalChart.
        porcelain_crown_string ,’,’),tn) || fn:
        contains(fn:join(dentalChart.
        removable_partial_denture_string ,’,’),tn
        ) ||
fn:contains(fn:join(dentalChart.
        fixed_bridge_string ,’,’),tn) || fn:
        contains(fn:join(dentalChart.
        rootcanal_treatment_string ,’,’),tn) ||
        fn:contains(fn:join(dentalChart.
        porcelain_infused_string ,’,’),tn) ||
fn:contains(fn:join(dentalChart.
        pitfissure_sealants_string ,’,’),tn)}”>
checked=”yes”</c:if> ><b>Whole Tooth</b>
<div id=”all_tooth_surfaces<%= tn %>” style
        =”display:none;”><table>


<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
        input type=”checkbox” name=”extrusion”
        value=”<%= tn %>” id=”extrusion<%= tn
        %>” onclick=”drawConditionMini2(’
        extrusion’,<%= tn %>)” <c:if test=”${fn:
        contains(fn:join(dentalChart.
        extrusion_string ,’,’),tn)}”>checked=”yes
        ”</c:if>></td>
<td>Extrusion</td>
<td><input type=”checkbox” name=”intrusion”
        value=”<%= tn %>” id=”intrusion<%= tn
        %>” onclick=”drawConditionMini2(’
        intrusion’,<%= tn %>)” <c:if test=”${fn:
        contains(fn:join(dentalChart.
        intrusion_string ,’,’),tn)}”>checked=”yes
        ”</c:if>></td>
<td>Intrusion</td>
<td><input type=”checkbox” name=”mesialdrift
        ” value=”<%= tn %>” id=”mesialdrift<%=
        tn %>” onclick=”drawConditionMini2(’
        mesialdrift’,<%= tn %>)” <c:if test=”${
        fn:contains(fn:join(dentalChart.
        mesial_rotation_string ,’,’),tn)}”>
        checked=”yes”</c:if>></td>
<td>Mesial Drift Rotation</td>
<td><input type=”checkbox” name=”distaldrift
        ” value=”<%= tn %>” id=”distaldrift<%=
        tn %>” onclick=”drawConditionMini2(’
        distaldrift’,<%= tn %>)” <c:if test=”${
        fn:contains(fn:join(dentalChart.
        distal_rotation_string ,’,’),tn)}”>
        checked=”yes”</c:if>></td>
<td>Distal Drift Rotation</td>
<td><input type=”checkbox” name=”rotation”
        value=”<%= tn %>” id=”rotation<%= tn %>”
         onclick=”drawConditionMini2(’rotation
        ’,<%= tn %>)” <c:if test=”${fn:contains(
        fn:join(dentalChart.rotation_string ,’,’)
        ,tn)}”>checked=”yes”</c:if>></td>
<td>Rotation</td>
</tr>

<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
        input type=”checkbox” name=”
        removablepartial” value=”<%= tn %>” id=”
        removablepartial<%= tn %>” onclick=”
        drawConditionMini2(’removablepartial
        ’,<%= tn %>)” <c:if test=”${fn:contains(
        fn:join(dentalChart.
        removable_partial_denture_string ,’,’),tn
        )}”>checked=”yes”</c:if>></td>
<td>Removable Partial Denture</td>
<td><input type=”checkbox” name=”fixedbridge
        ” value=”<%= tn %>” id=”fixedbridge<%=
        tn %>” onclick=”drawConditionMini2(’
        fixedbridge’,<%= tn %>)” <c:if test=”${
        fn:contains(fn:join(dentalChart.
        fixed_bridge_string ,’,’),tn)}”>checked=”
        yes”</c:if>></td>
<td>Fixed Bridged</td>
<td><input type=”checkbox” name=”rootcanal”
        value=”<%= tn %>” id=”rootcanal<%= tn
        %>” onclick=”drawConditionMini2(’
        rootcanal’,<%= tn %>)” <c:if test=”${fn:
        contains(fn:join(dentalChart.
        rootcanal_treatment_string ,’,’),tn)}”>
        checked=”yes”</c:if>></td>
<td>Root Canal Treatment</td>
<td><input type=”checkbox” name=”
        porcelainfused” value=”<%= tn %>” id=”
        porcelainfused<%= tn %>” onclick=”
        drawConditionMini2(’porcelainfused’,<%=
        tn %>)” <c:if test=”${fn:contains(fn:
        join(dentalChart.porcelain_crown_string
        ,’,’),tn)}”>checked=”yes”</c:if>></td>
<td>Porcelain Fused to Metal</td>
<td><input type=”checkbox” name=”pitfissure”
         value=”<%= tn %>” id=”pitfissure<%= tn
        %>” onclick=”drawConditionMini2(’
        pitfissure’,<%= tn %>)” <c:if test=”${fn
        :contains(fn:join(dentalChart.
        pitfissure_sealants_string ,’,’),tn)}”>
        checked=”yes”</c:if>></td>
<td>Pit and Fissure Sealants</td>

</tr>


<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
        input type=”checkbox” name=”postcore”
        value=”<%= tn %>” id=”postcore<%= tn %>”
         onclick=”drawConditionMini2(’postcore
        ’,<%= tn %>)” <c:if test=”${fn:contains(
        fn:join(dentalChart.
        postcore_crown_string ,’,’),tn)}”>checked
        =”yes”</c:if>></td>
<td>Post Core Crown</td>
<td><input type=”checkbox” name=”acrylic”
        value=”<%= tn %>” id=”acrylic<%= tn %>”
        onclick=”drawConditionMini2(’acrylic
        ’,<%= tn %>)” <c:if test=”${fn:contains(
        fn:join(dentalChart.acrylic_crown_string
        ,’,’),tn)}”>checked=”yes”</c:if>></td>
<td>Acrylic Crown</td>
<td><input type=”checkbox” name=”metal”
        value=”<%= tn %>” id=”metal<%= tn %>”
        onclick=”drawConditionMini2(’metal’,<%=
        tn %>)” <c:if test=”${fn:contains(fn:
        join(dentalChart.metal_crown_string ,’,’)
        ,tn)}”>checked=”yes”</c:if>></td>
<td>Metal Crown</td>
<td><input type=”checkbox” name=”porcelain”
        value=”<%= tn %>” id=”porcelain<%= tn
        %>” onclick=”drawConditionMini2(’
        porcelain’,<%= tn %>)” <c:if test=”${fn:
        contains(fn:join(dentalChart.
        porcelain_crown_string ,’,’),tn)}”>
        checked=”yes”</c:if>></td>
<td>Porcelain Crown</td>

</tr>


<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
```

```
      input type="checkbox" name="extracted"
      value="<%= tn %>" id="extracted<%= tn
      %>" onclick="drawConditionMini2('
      extracted',<%= tn %>)" <c:if test="${fn:
      contains(fn:join(dentalChart.
      extracted_string,','),tn)}">checked="yes
      "</c:if></td>
<td>Extracted</td>
<td><input type="checkbox" name="missing"
      value="<%= tn %>" id="missing<%= tn %>"
      onclick="drawConditionMini2('missing
      ',<%= tn %>)" <c:if test="${fn:contains(
      fn:join(dentalChart.missing_string,','),
      tn)}">checked="yes"</c:if></td>
<td>Missing</td>
<td><input type="checkbox" name="unerupted"
      value="<%= tn %>" id="unerupted<%= tn
      %>" onclick="drawConditionMini2('
      unerupted',<%= tn %>)" <c:if test="${fn:
      contains(fn:join(dentalChart.
      unerupted_string,','),tn)}">checked="yes
      "</c:if></td>
<td>Unerupted</td>
<td><input type="checkbox" name="impacted"
      value="<%= tn %>" id="impacted<%= tn %>"
       onclick="drawConditionMini2('impacted
      ',<%= tn %>)" <c:if test="${fn:contains(
      fn:join(dentalChart.impacted_string,',')
      ,tn)}">checked="yes"</c:if></td>
<td>Impacted</td>
</tr>


</table></div>



 <hr>
  <a href = "javascript:void(0)" onclick = "
       submitcond()">Submit</a>
  <a href = "javascript:void(0)" onclick = "
       document.getElementById('light<%= tn
       %>').style.display='none';document.
       getElementById('fade<%= tn %>').style.
       display='none'">Cancel</a>

  <br/>
  <br/>
  <b>Services needed</b>
  <br/>
      <hr>
  <u>Operative Dentistry</u><br/>
  <input type="checkbox" name="class1" id="
       class1<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_1_string,','),tn)}">
       checked="yes"</c:if>>Class I
  <input type="checkbox" name="class2" id="
       class2<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_2_string,','),tn)}">
       checked="yes"</c:if>>Class II
  <input type="checkbox" name="class3" id="
       class3<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_3_string,','),tn)}">
       checked="yes"</c:if>>Class III
  <input type="checkbox" name="class4" id="
       class4<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_4_string,','),tn)}">
       checked="yes"</c:if>>Class IV
  <input type="checkbox" name="class5" id="
       class5<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.class_5_string,','),tn)}">
       checked="yes"</c:if>>Class V
  <input type="checkbox" name="onlay" id="
       onlay<%= tn %>" value="<%= tn %>" <c:
       if test="${fn:contains(fn:join(
       services.onlay_string,','),tn)}">
       checked="yes"</c:if>>Onlay
  <br/><br/>
  <u> Surgery </u><br/>
  <input type="checkbox" name="extraction"
       id="extractss<%= tn %>" value="<%= tn
       %>" <c:if test="${fn:contains(fn:join(
       services.extraction_string,','),tn)}">
       checked="yes"</c:if>>Extraction
  <input type="checkbox" name="odontectomy"
       id="odontectomy<%= tn %>" value="<%=
       tn %>" <c:if test="${fn:contains(fn:
       join(services.odontectomy_string,','),
       tn)}">checked="yes"</c:if>>Odontectomy
  <input type="checkbox" name="specialcase"
       id="specialcase<%= tn %>" value="<%=
       tn %>" <c:if test="${fn:contains(fn:
       join(services.special_case_string,',')
       ,tn)}">checked="yes"</c:if>>Special
       Case
  <br/><br/>
  <u> Emergency Treatment </u><br/>
  <input type="checkbox" name="pulpsedation"
        id="pulpsedation<%= tn %>" value="<%=
        tn %>" <c:if test="${fn:contains(fn:
       join(services.pulp_sedation_string
       ,','),tn)}">checked="yes"</c:if>>Pulp
       Sedation
  <input type="checkbox" name="
       crownrecementation" id="
       crownrecementation<%= tn %>" value
       ="<%= tn %>" <c:if test="${fn:contains
       (fn:join(services.
       crown_recementation_string,','),tn)}">
       checked="yes"</c:if>>Recementation of
       crowns
  <input type="checkbox" name="
       fillingservice" id="fillingservice<%=
       tn %>" value="<%= tn %>" <c:if test="$
       {fn:contains(fn:join(services.
       filling_service_string,','),tn)}">
       checked="yes"</c:if>>Temporary
       fillings
  <br/><br/>
  <u> Fixed Partial Dentures </u><br/>
  <input type="checkbox" name="laminated" id
       ="laminated<%= tn %>" value="<%= tn
       %>" <c:if test="${fn:contains(fn:join(
       services.laminated_string,','),tn)}">
       checked="yes"</c:if>>Laminated
  <input type="checkbox" name="singlecrown"
       id="singlecrown<%= tn %>" value="<%=
       tn %>" <c:if test="${fn:contains(fn:
       join(services.single_crown_string,',')
       ,tn)}">checked="yes"</c:if>>Single
       Crown
  <input type="checkbox" name="bridgeservice
       " id="bridgeservice<%= tn %>" value
       ="<%= tn %>" <c:if test="${fn:contains
       (fn:join(services.
       bridge_service_string,','),tn)}">
       checked="yes"</c:if>>Bridge
  <br/><br/>
  <u> Endodontics </u><br/>
  <input type="checkbox" name="anterior" id
       ="anterior<%= tn %>" value="<%= tn %>"
        <c:if test="${fn:contains(fn:join(
       services.anterior_string,','),tn)}">
       checked="yes"</c:if>>Anterior
  <input type="checkbox" name="posterior" id
       ="posterior<%= tn %>" value="<%= tn
       %>" <c:if test="${fn:contains(fn:join(
       services.posterior_string,','),tn)}">
       checked="yes"</c:if>>Posterior
  <input type="checkbox" name="
       otherendodontics" id="otherendodontics
       <%= tn %>" value="<%= tn %>" <c:if
       test="${fn:contains(fn:join(services.
       ortho_endo_string,','),tn)}">checked="
       yes"</c:if>>Others (Endosurgery,
       Bleaching, etc.)
  <br/><br/>
  <hr>
  <a href = "javascript:void(0)" onclick = "
       submitcond()">Submit</a>
  <a href = "javascript:void(0)" onclick = "
       document.getElementById('light<%= tn
       %>').style.display='none';document.
       getElementById('fade<%= tn %>').style.
       display='none'">Cancel</a>


</div><div id="fade<%= tn %>" class="
     black_overlay"></div>

<%
   }
  }
%>

<div id="serviceslight" class="white_content
     ">
 <font size = "2">
 <b>Other Services</b><br/>
 <hr>
 <u>Periodontics</u><br/>
 <input type="checkbox" name="periodontics"
      id="periodontics" value="yes" <c:if
      test="${services.periodontics eq 'yes
      '}">checked="yes"</c:if>>Management of
      Periodontal Disease
 <br/><br/>
 <u> Surgery </u><br/>
 <input type="checkbox" name="surgery" id="
      pedodontics" value="pedodontics" <c:if
      test="${services.surgery eq '
      pedodontics'}">checked="yes"</c:if>>
      Pedodontics <br/>
 <input type="checkbox" name="surgery" id="
      orthodontics" value="orthodontics" <c:
      if test="${services.surgery eq '
      orthodontics'}">checked="yes"</c:if>>
      Orthodontics
 <br/><br/>
 <u> Emergency Treatment </u><br/>
 <input type="checkbox" name="
      emergencytreatment" id="acuteinfections
      " value="acute infections" <c:if test="
      ${services.emergency_treatment eq '
      acute infections'}">checked="yes"</c:if
      >>Management of Acute Infections <br/>
 <input type="checkbox" name="
      emergencytreatment" id="
      traumaticinjuries" value="traumatic
      injuries" <c:if test="${services.
      emergency_treatment eq 'traumatic
      injuries'}">checked="yes"</c:if>>
      Management of Temporary Injuries
 <br/><br/>
 <u> Prosthodontics </u><br/>
 <input type="checkbox" name="prosthodontics
      " id="completedent" value="complete
      denture" <c:if test="${services.
      prosthodontics eq 'complete denture'}">
      checked="yes"</c:if>>Complete Denture<
      br/>
 <input type="checkbox" name="prosthodontics
      " id="singledent" value="single denture
      " <c:if test="${services.prosthodontics
       eq 'single denture'}">checked="yes"</c
      :if>>Single Denture<br/>
 <input type="checkbox" name="prosthodontics
      " id="removedent" value="removable
      partial" <c:if test="${services.
      prosthodontics eq 'removable partial
      '}">checked="yes"</c:if>>Removable
      Partial Denture<br/>
 <input type="checkbox" name="prosthodontics
      " id="otherss" value="others" <c:if
      test="${services.prosthodontics eq '
      others'}">checked="yes"</c:if>>Other
      Denture Services
 <br/><br/>
 <hr>
 <a href = "javascript:void(0)" onclick = "
      document.getElementById('serviceslight
      ').style.display='none';document.
      getElementById('servicesfade').style.
      display='none'">Done</a>

 </font>
</div>
<div id="servicesfade" class="black_overlay
     "></div>



<div id="noteslight" class="white_content">
 <font size = "2">
 <b>Notes</b><br/>
 <hr>
 <br/>
 <textarea name="notes" rows="6" cols
      ="100"></textarea>
 <br/><br/>
 <hr>
 <a href = "javascript:void(0)" onclick = "
      document.getElementById('noteslight').
      style.display='none';document.
      getElementById('notesfade').style.
      display='none'">Submit</a>
 <a href = "javascript:void(0)" onclick = "
      document.getElementById('noteslight').
      style.display='none';document.
      getElementById('notesfade').style.
      display='none'">Cancel</a>

 </font>
</div>

<div id="neededlight" class="white_content">
<b>Service Needed Summary</b> &nbsp;&nbsp
    ;||&nbsp;&nbsp;
<a href = "javascript:void(0)" onclick = "
     clearneeded_service();">Cancel</a>

<table>
<tr>
<td>
<u><div id="periodonticsdiv"></div></u>
<div id="periodiv"></div>
</td>

<td>
<u><div id="emergencytreatmentdiv"></div></u
    >
<div id="pulpsedationdiv"></div>
<div id="crownrecementationdiv"></div>
<div id="fillingservicediv"></div>
<div id="acuteinfectionsdiv"></div>
<div id="traumaticdiv"></div>

</td>
</tr>

<tr>
<td>
<u><div id="operativedentistrydiv"></div></u
    >
<div id="class1div"></div>
<div id="class2div"></div>
<div id="class3div"></div>
<div id="class4div"></div>
<div id="class5div"></div>
<div id="onlaydiv"></div>
</td>

<td>
<u><div id="fixedpartialdenturediv"></div></
    u>
<div id="laminateddiv"></div>
<div id="singlecrowndiv"></div>
<div id="bridgeservicediv"></div>
<br/>
<u><div id="endodonticsdiv"></div></u>
<div id="anteriordiv"></div>
<div id="posteriordiv"></div>
<div id="othersdiv"></div>
</td>
</tr>

<tr>
<td>
<u><div id="surgerydiv"></div></u>
<div id="extractiondiv"></div>
<div id="odontectomydiv"></div>
<div id="specialcasediv"></div>
<div id="pedonticsdiv"></div>
<div id="orthodonticsdiv"></div>
</td>

<td>
<u><div id="prosthodonticsdiv"></div></u>
<div id="completedentdiv"></div>
<div id="singledentdiv"></div>
<div id="removedentdiv"></div>
<div id="otherssdiv"></div>
</td>
</tr>
</table>

</div>

 <script type="text/javascript"> init(65,
     65, 627);     </script>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

 pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
```

```
     Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">

        <title>DentISt</title>
<script src="//ajax.googleapis.com/ajax/libs
     /jquery/1.8.3/jquery.min.js" >
</script>

<script>
    function updateTaskList() {
            $.ajax({
                type : 'POST',
                url : 'greet',//this is url
                     mapping for controller
               error: function(){
        //alert("error");
    },
    success: function(data){
  $("#tasklist").replaceWith('<div id=
       tasklist>'+ data + '</div>');
        //alert("success");
    }            });
        }
</script>
<script>
 function insertSpace(name){
  var myString = name;
var newString = "";
var wasUpper = false;
for (var i = 0; i < myString.length; i++)
{
    if (!wasUpper && myString[i] == myString
         .toUpperCase()[i])
    {
        newString = newString + "\n";
        wasUpper = true;
    }
    else
    {
        wasUpper = false;
    }
    newString = newString + myString[i];
}
document.write(newString);

 }

</script>
<script>

$(document).ready(function(){



setInterval(function () {updateTaskList()
     ;},1000);




});
</script>

</head>
<body>
<c:if test="${msgtasktaken!=null}">
<script>var str='<c:out value="${
     msgtasktaken}"/>';alert(str);</script>
   </c:if>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

        </div>

<div id="layout-menu" style="top:15%;">
<br>
<%@include file="/WEB-INF/views/patientMenu.
     jsp"%>
</div>


<div id="layout-body">

<div class="form-body" style="top:15%;left-
     margin:15%">

<div class="sub-body-title">Tasks</div>
<br>
<h3>&nbsp;&nbsp;&nbsp;Tasks List</h3>


  <div id="tasklist">
<table class="list-table">
<thead>
<tr class="head-list-table">
<td> </td>
<td>Task :</td>
<td>Patient :</td>
<c:if test="${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     workflow')}">
<td>Assigned Clinician :</td>
 </c:if>

</tr>
</thead>

<c:if test="${not empty definitions}">

<tbody>
  <c:forEach items="${definitions}" var="
        defi" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
       <c:set var="rowColor" value="even"
            scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
            scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">



 <td>
         ${loop.index + 1}
    </td>
    <td>
         <a name="task[]" href="task?idTask=
              ${defi.id}&nameTask=${name[loop
              .index]}&patientid=${
              patientlist[loop.index].id}&
              instanceid=${instanceid}&pos=${
              loop.index}"><script>
              insertSpace("<c:out value='${
              name[loop.index]}' />");</
              script></a>
 </td>
 <td>
        ${patientlist[loop.index].name}
    </td>
 </tr>
  </c:forEach>
<br><br>
</c:if>

<c:if test="${empty definitions}">
<br>
</c:if>
</div>
</div>
</div>
<br>
<br>
<%@include file="/WEB-INF/views/footer.jsp
     "%>
</body>
</html>


<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl
     /functions" prefix="fn" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">
```

```
<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">         <
     title>DentISt</title>
<link rel="stylesheet" type="text/css" href
     ="${pageContext.request.contextPath}/
     theme.css" />

 <script src="//ajax.googleapis.com/ajax/
      libs/jquery/1.8.3/jquery.min.js" >
</script>
        <script type="text/javascript">
    function DisplayFormValues()
    {


        var elem = document.getElementById('
             form').elements;
  var str=' <c:out value="${fieldValues
       }"/>';

   var n=str.split("|");
   var j=0;
   var temp="";
        for(var i = 0; i < elem.length; i++)
        {
   if(str!="" && n[j]!=" null "){
   if(n[j]!=" null "){
   n[j]=jQuery.trim(n[j]);
           if(elem[i].type!='hidden' && elem
                [i].type!='checkbox' && elem[
                i].type!='radio' && elem[i].
                type!='submit'){

            elem[i].value =n[j];

   j++;
            }
   if(elem[i].type=='checkbox' && elem[i+1].
        name!=elem[i].name){
    j++;
   }
   if(elem[i].type=='radio' && elem[i+1].
        name!=elem[i].name){
      if(temp==" "+elem[i].value+" " || temp
           ==elem[i].value){
     elem[i].checked=true;}

        j++;
   }
   else if(elem[i].type=='radio' && elem[i
        +1].name==elem[i].name){

    temp=n[j];

    if(temp==" "+elem[i].value+" " || temp==
         elem[i].value){
     elem[i].checked=true;}
   }
   }
   }
           }

    }
</script>



</head>
<body>
<c:set var="processId" value="${taskRSname
     }"/>

<c:if test="${processId == 'UPCDDentISt.
     CreateNewPatient'}">
<div id="layout-header">
<%@include file="/WEB-INF/views/header.jsp
     "%>
</div>
</c:if>

<c:if test="${taskRSname == 'DentalChart'}">
<br><br>
<%@include file="/WEB-INF/views/scripts.js
     "%>
</c:if>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

        </div>



<br><br><br><br>

<div class="sub-body-title" style="height:75
     px">
<h3>&nbsp;&nbsp;${patientname}</h3>

</div>
<br>
<div>
<center>
     <a href="javascript:history.back()">Go
          Back</a>
</center>


</div>
<div id="layout-menu" style="top:20%;display
     :block;">
<br><br><br><br> <br><br>
<%@include file="/WEB-INF/views/patientMenu.
     jsp"%>
</div>
<div id="layout-body" style="top:20%;border:
     none;z-index:-1;">
<br><br><br><br> <br><br>
<div class="form-body" style="top:20%;">
<div class="sub-body-title" style="width
     =75%;align:right;">View Versions</div>

<div class="box" style="width=75%;align:
     right;">
<c:if test="${empty versionLists}">
<br>
</c:if>
<c:if test="${not empty versionLists}">
<table class="list-table" style="width=75%;
     align:right;">
<thead>
<tr class="head-list-table">
<td></td>
<td>Version :</td>
<td>Updated By :</td>
<td>Updated Date :</td>
<td>Status :</td>
<td>Approved by :</td>
<td>Approved Date :</td>
</tr>
</thead>
<tbody>
  <c:forEach items="${versionLists}" var="
       versionList" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">



 <td>
         ${loop.index +1}
    </td>
    <td>
         <a href="http://127.0.0.1:8090/
              dentISt/app/task?idTask=${
              idTask}&username=krisv&password
              =krisv&nameTask=${formname}&
              patientid=${patientid}&pos=0&
              version=${fn:length(
              versionLists)-loop.index}&
              dashid=${dashid}&dashname=${
              dashname}">Version${versionList
              .version}</a>
    </td>
 <td>
         ${versionList.updated_by}
    </td>
 <td>
         ${versionList.updated_date}
    </td>

 <td>
         ${versionList.approved}
    </td>
 <td>
         ${versionList.approved_by}
    </td>
 <td>
         ${versionList.approved_date}
    </td>
 </tr>


<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

 pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl
     /functions" prefix="fn" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">          <
     title>DentISt</title>
<link rel="stylesheet" type="text/css" href
     ="${pageContext.request.contextPath}/
     theme.css" />
         <script src="//ajax.googleapis.com/
              ajax/libs/jquery/1.8.3/jquery.
              min.js" >
         </script>

<script>
  function insertSpace(name){
   var myString = name;
 var newString = "";
 var wasUpper = false;
 for (var i = 0; i < myString.length; i++)
 {
     if (!wasUpper && myString[i] == myString
          .toUpperCase()[i])
     {
         newString = newString + "\n";
         wasUpper = true;
     }
     else
     {
         wasUpper = false;
     }
     newString = newString + myString[i];
 }
 document.write(newString);

  }

</script>
</head>
<body>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

         </div>
<br><br><br><br>

<div class="sub-body-title" style="height:75
     px">
<h3>&nbsp;&nbsp;${patientname}</h3>

</div>
<br>



<br>
<div style="color:#756EB8;"><hr></div>
```

```
  </c:forEach>
<br><br>
</c:if>

     <br>
</div>
</div>
<br>
</div>
</div>

<%@include file="/WEB-INF/views/footer.jsp
       "%>
</body>
</html>


<div id="menuheader">
<div class="sub-menu-title" style="width:
     23%" align="center"><b>&nbsp;&nbsp;
     Patient Forms</b></div>


<ul class="sub-menu-subjs" style="left:1%;
     width: 22%" id="menudiv">

     <li>

<c:forEach var="form" items="${formlist}"
     varStatus="loop">
 <c:set var="taskname" value="${form.key}"/>
 <c:if test="${taskname != '
      AssignToODClinician'}">

 <font size="2">

 <c:if test="${taskname != 'DentalChart'}">
 <a name="task[]" href="${fn:replace(form.
      value,'task?', 'viewForm?')}&patientid=
      ${patientid}" id="forms{loop.index}"
      onclick="changeSrc(this);"><script>
      insertSpace("<c:out value='${form.key}'
       />");</script></a>
 </c:if>

 <c:if test="${taskname == 'DentalChart'}">
 <a name="task[]" href="${fn:replace(form.
      value,'task?', 'viewForm?')}&patientid=
      ${patientid}" id="dentalchart" onclick
      ="changeSrc(this);"><script>insertSpace
      ("<c:out value='${form.key}' />");</
      script></a>
 </c:if>
 </font>
  </c:if>


</c:forEach>
</li>
</ul>
</div>

 <div id="layout-body" style="top:20%;border
      :none;left:25%;align:right;right:25%;z-
      index:-1;">
<br><br><br><br> <br><br>
<div class="form-body" style="top:20%;border
     :none;left:25%;align:right;right:25%">
<div class="sub-body-title" style="width
     =75%;align:right;">View Versions</div>

<div class="box" style="width=75%;align:
     right;">
<c:if test="${empty versionLists}">
<br>
</c:if>
<c:if test="${not empty versionLists}">
<table class="list-table" style="width=75%;
     align:right;">
<thead>
<tr class="head-list-table">
<td></td>
<td>Version :</td>
<td>Updated By :</td>
<td>Updated Date :</td>
<td>Status :</td>
<td>Approved by :</td>
<td>Approved Date :</td>
</tr>
</thead>
<tbody>
  <c:forEach items="${versionLists}" var="
```

```
        versionList" varStatus="loop">
  <c:choose>
      <c:when test='${(loop.index)%2 eq 0}'>
        <c:set var="rowColor" value="even"
             scope="page"/>
     </c:when>
     <c:otherwise>
        <c:set var="rowColor" value="odd"
             scope="page"/>
     </c:otherwise>
   </c:choose>


<tr class="${rowColor}">



 <td>
          ${loop.index +1}
     </td>
     <td>
          <a href="http://127.0.0.1:8090/
               dentISt/app/viewForm?idTask=${
               idTask}&username=krisv&password
               =krisv&nameTask=${formname}&
               patientid=${patientid}&pos=0&
               version=${fn:length(
               versionLists)-loop.index}">
               Version${versionList.version}</
               a>
     </td>
 <td>
          ${versionList.updated_by}
     </td>
 <td>
          ${versionList.updated_date}
     </td>
 <td>
          ${versionList.approved}
     </td>
 <td>
          ${versionList.approved_by}
     </td>
<td>
          ${versionList.approved_date}
     </td>
 </tr>
  </c:forEach>
<br><br>
</c:if>
</div>
</div>
<%@include file="/WEB-INF/views/footer.jsp
     "%>
</body>
</html>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
       Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div class="form-menu">
<c:url var="odURL" value="/app/checkPatient"
      />
<c:url var="searchURL" value="/app/
     searchPatient" />
<c:url var="approveURL" value="/app/greet"
     />

<div class="sub-menu-title">Manage Patients
     </div>


<div>
<ul class="sub-menu-subjs">
<c:if test="${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     faculty') || fn:containsIgnoreCase(fn:
     join(sessionScope.currentRoleList,','),'
     student')}">

 <li><a href="${searchURL}">Find Patient</a
     ></li>
 </c:if>

<c:if test="${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     diagnosis')}">
<li><a href="${odURL}">Create Patient</a></
     li>
</c:if>
<li><a href="${approveURL}">Tasks</a></li>

</ul>
</div>

</div>
</body>
</html>
```

```
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl
     /functions" prefix="fn" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
       Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">         <
     title>DentISt</title>
<link rel="stylesheet" type="text/css" href
     ="${pageContext.request.contextPath}/
     theme.css" />
        <script src="//ajax.googleapis.com/
             ajax/libs/jquery/1.8.3/jquery.
             min.js" >
</script>

</head>
<body>


<c:if test="${processId == 'UPCDDentISt.
     Patient' ||processId == 'UPCDDentISt.
     ReturningPatient' || processId == '
     OralDiagnosisSection.DentalChart'}">
<div id="layout-header">
<%@include file="/WEB-INF/views/header.jsp
     "%>
</div>
</c:if>

<c:if test="${processId == '
     OralDiagnosisSection.DentalChart'}">
<br><br><br><br><br>
<%@include file="/WEB-INF/views/scripts.js
     "%>
</c:if>


<c:if test="${processId == 'UPCDDentISt.
     Patient' || processId == 'UPCDDentISt.
     ReturningPatient'}">
<div id="layout-menu" style="top:15%;display
     :block;">
<br>
<%@include file="/WEB-INF/views/patientMenu.
     jsp"%>
</div>
<div id="layout-body">

<div class="form-body">
</c:if>


<c:if test="${processId != '
```

```
     OralDiagnosisSection.DentalChart' &&
     processId != 'UPCDDentISt.Patient' &&
     processId != 'UPCDDentISt.
     ReturningPatient'}">
<!--
<div id="layout-menu" style="top:15%;display
     :block;">
<%@include file="/WEB-INF/views/patientMenu2
     .jsp"%>
</div>

<div id="formlayout-body">
-->
</c:if>


<c:set var="processId" value="${processId
     }"/>
<!----------------------------------------------------
     SCRIPTS------------------------------>
<script>
$(document).ready(function(){
$("button, input[type=button],input[type=
     submit]").attr("class", "classname");

  $('input[type=checkbox]').filter(function(
       i,field) {
       var name=field.name+'text';
 if(document.getElementById(name)==null){
 var $ctrl = $(document.createElement("
      input")).attr({
                    id:     name
                   ,name:   name
                   ,value:  ' '
                   ,type:   'text'
         ,style: 'visibility: hidden;width:0
              px '
         ,size: 0

             })

var lbl = '<p></p>';


$($("[name="+field.name+"]")[0]).before(
     $ctrl);
}
});


$('input[type=checkbox]').click(function() {
 $('input[type=text]').filter(function(i,
      field) {


});
 var checkname=this.name+"text";
 var checkstr="";
  $('input[name='+ checkname+']').val($('
       input[name='+checkname+']').val() +
       this.value+ '~' );


});
$('input[id=continuebtn]').click(function(){
 $("input[id=continue]").val('continue');
});
 var values="";
  $("button").click(function(){

    x=$("form textarea,input[type=text],
         input[type=radio],select").
         serializeArray();
    $.each(x, function(i, field){
        if(field.name=='birthdate'){
values= values + field.value +"/";
}
else if(field.name=='bday'){
values= values + field.value +"/";
}
else if(field.name=='byear'){
values= values + field.value +" | ";
}
else if(field.name!='bday' && field.name!='
     byear'){
values= values+ field.value + " | ";
}
    });
 $("input[name=values]").val(values);

  });
});
</script>

<script>
<%@include file="/WEB-INF/views/function.js
    "%>
</script>
<!----------------------------------------------------
     SCRIPTS------------------------------>
<c:if test="${patientid!=null}">
<form name="form" action="complete?patientid
     =${patientid}" method="POST" enctype="
     multipart/form-data">
     <input type="hidden" name="fields"
          value="">
     <input type="hidden" id=values name=
          values value="">
     <input type="hidden" name="formname"
          value="${processId}">
     <input type="hidden" id="viewflag" name
          ="viewflag" value="false">


<p></p>
     ${htmlRender}
    <input type=submit id="returningbutton"
         class="classname">
    </form>
<script>document.getElementById("
     returningbutton").click();</script>

</c:if>

   <c:if test="${fn:contains(processId, '
        OralDiagnosisSection')}">
<!--
<style type="text/css">
input[type = "text"]{
 border: 1px solid;
 disabled:false;
 background: #fff;
}
textarea{
 border: 1px solid;
 disabled:false;
 background: #fff;

}
</style>
-->



   </c:if>
<!--
<c:forEach var="instance" items="${instances
     }">
    <a href="instance?instancesId=${instance
         .id}&processId=${process.id}" >
     Id: ${instance.id} (${instance.
         startDate})
    </a> <br/>
   </c:forEach>
-->
    <form name="form" action="complete?
         patientid=${patientid}" method="POST
         " enctype="multipart/form-data">
     <input type="hidden" name="fields"
          value="">
     <input type="hidden" id=values name=
          values value="">
     <input type="hidden" name="formname"
          value="${processId}">
     <input type="hidden" id="viewflag" name
          ="viewflag" value="false">


<p></p>
     ${htmlRender}
    <button>Submit</button>
    <c:if test="${processId == 'UPCDDentISt.
         Patient' || processId == '
         UPCDDentISt.ReturningPatient'}">
    <a href="searchPatient">Cancel</a>
    </c:if>
    </form>




  <!--
   <c:if test="${(processId != 'UPCDDentISt.
        Patient' && fn:contains(processId, '
        AllSections')) || processId == '
        OralDiagnosisSection.DentalChart'}">
    <form action="allSections" method="link
         ">
   </c:if>
   <c:if test="${fn:contains(processId, '
        OralDiagnosisSection') && processId
        != 'OralDiagnosisSection.DentalChart
        '}">
    <form action="http://agila.upm.edu.ph
         :8090/dentISt/app/viewForm?processId
         =OralDiagnosisSection.
         PatientInformation" method="link">
    <input type=submit value="Cancel" style
            ="font-size:14;"><br>
         </form>
        </c:if>
      -->

<br>
<br>
 </div>
</div>
</div>
<%@include file="/WEB-INF/views/footer.jsp
     "%>
</body>
</html>
```

```
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<html>
<body>
<c:if test="${instanceid!=null}">
    <c:redirect url="greet?patientid=${
          patientid}&instanceid=${instanceid}&
          pos=${pos}"/>
    </c:if>

</body>
</html>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">         <
     title>DentISt</title>

</head>
<body>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

        </div>

<div id="layout-menu" style="top:15%;">
<br>
<%@include file="/WEB-INF/views/patientMenu.
     jsp"%>
</div>


<div id="layout-body">

<div class="form-body" style="top:15%;left-
     margin:15%;">

<div class="sub-body-title">Find Patient(s)
     </div>
<br>
<form action="searchPatient" method="GET">
<h2>Find Patient(s)</h2>
<hr>
<h3>Patient Identifier or Patient Name&nbsp
     ;&nbsp;&nbsp; <input type=text name="
     patientsearch" id="patientsearch"> </h3>
<input type=submit value="Search" class="
     classname">
</form>
<c:if test="${empty patientLists}">
<br>
</c:if>
<c:if test="${not empty patientLists}">
<table class="list-table" style="right:1%;
     width:90%;bottom:1%">
<thead>
<tr align=right><td colspan=4><font size=2><
     b><i>${fn:length(patientLists)} patient(
     s) found</i></b></font>
</tr>
<tr class="head-list-table">
<td>Patient Name :</td>
<td>UPCD ID :</td>
<td>Record Archive :</td>
<c:if test="${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     diagnosis')}">
<td>Start New Case :</td>
</c:if>

</tr>
</thead>
<tbody>
  <c:forEach items="${patientLists}" var="
       patientList" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>

<tr class="${rowColor}">



    <td>
         ${patientList.name}
    </td>
 <td>
         ${patientList.upcdId}
    </td>
    <td>
         <a href="${fn:replace(formurl,'task
              ?', 'viewForm?')}&patientid=${
              patientList.id}">View</a>
    </td>

<c:if test="${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     diagnosis')}">
<td>
      <c:if test="${patientList.instanceid
           == '0' }">
      <a href="returningpatient?processId=
           UPCDDentISt.Patient&patientid=${
           patientList.id}&instanceid=1">
           Start</a>
      </c:if>


    </td>
</c:if>
 </tr>
  </c:forEach>

<tr style="bgcolor:white"><td>&nbsp;</td>
</tr>
</c:if>


</div>
<br>
```

```
</div>
<br>

</div>
<br>


 <%@taglib prefix="form" uri="http://www.
      springframework.org/tags/form" %>
 <%@taglib uri="http://java.sun.com/jsp/jstl
      /core" prefix="c" %>
 <%@ taglib uri="http://java.sun.com/jsp/
      jstl/functions" prefix="fn" %>

  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML
      4.01 Transitional//EN" "http://www.w3.
      org/TR/html4/loose.dtd">

 <html>
  <head>

 <meta http-equiv="Content-Type" content="
      text/html; charset=ISO-8859-1">
      <title>DentISt</title>
 <link rel="stylesheet" type="text/css" href
      ="${pageContext.request.contextPath}/
      theme.css" />

  <script src="//ajax.googleapis.com/ajax/
       libs/jquery/1.8.3/jquery.min.js" >
 </script>


    <script type="text/javascript">
  function DisplayFormValues()
  {

     var elem = document.getElementById('
          form').elements;

   var str=' <c:out value="${fieldValues
        }"/>';


    var n=str.split("|");
    var j=0;
    var temp="";
   for(var i = 0; i < elem.length; i++)
   {
    if(str!="" && (n[j]!=" null " && n[j]!="
           null ") && n[j]!=null && n[j]!='
         '){
    if(n[j]!="null "){
    n[j]=jQuery.trim(n[j]);
    //alert(elem[i].name);
      if(elem[i].type!='hidden' && elem[i].
           type!='checkbox' && elem[i].type
           !='radio' && elem[i].type!='submit
           ' && elem[i].type!='button'){
                 //  alert(elem[i].name);
    elem[i].value=n[j];
    if(elem[i-1].type=='radio' && elem[i].
         value!=null && elem[i].value!="" &&
         elem[i].value!="  " ){
     var name=elem[i-1].name + "span";
     if(document.getElementById(name)==null)
          {}
     else{document.getElementById(name).
          style.display = "inline";}
    }

    j++;

    }
    if(elem[i].type=='checkbox' && elem[i
         +1].name!=elem[i].name){

     var inputtextname=elem[i].name+"text";

  if(document.getElementById(inputtextname)
       ==null){
  var $ctrl =  $(document.createElement("
       input")).attr({
        id:     inputtextname
       ,name:  inputtextname
       ,value: ' '
       ,type:  'text'
       ,style: 'display:none;z-index:-99;
            visibility: hidden;width:0px '
       ,size: 0
```

```
<br>
<%@include file="/WEB-INF/views/footer.jsp
       "%>
</body>
</html>



     })

  var lbl =  '<p></p>';


  $($("[name="+elem[i].name+"]")[0]).before(
       $ctrl);
  //alert(inputtextname);
      document.getElementById(inputtextname).
           value = n[j];
         //alert(document.getElementById(
              inputtextname).value);

  }                                       i++;

      var value=n[j].split(",");


      for (var k=0; k<value.length; k++){
       checkval(jQuery.trim(value[k]));

      }
      j++;

     }
     if(elem[i].type=='radio' && elem[i+1].
          name!=elem[i].name){
       if(temp==" "+elem[i].value+" " || temp
            ==elem[i].value){
      elem[i].checked=true;}

          j++;
     }
     else if(elem[i].type=='radio' && elem[i
          +1].name==elem[i].name){

      temp=n[j];

      if(temp==" "+elem[i].value+" " || temp
           ==elem[i].value){
       elem[i].checked=true;}
     }
     }
     }
        }

   }

  function checkval(val){
  $("input:checkbox[value="+val+"]").attr("
       checked", true);
  }
  </script>




  </head>
  <body>
  <c:set var="processId" value="${taskRSname
       }"/>

  <c:if test="${processId == 'UPCDDentISt.
       CreateNewPatient'}">
  <div id="layout-header">
  <%@include file="/WEB-INF/views/header.jsp
       "%>
  </div>
  </c:if>

  <c:if test="${taskRSname == 'DentalChart
       '}">
  <br><br>
  <%@include file="/WEB-INF/views/scripts.js
       "%>
  </c:if>



  <div id="layout-header">

  <%@include file="/WEB-INF/views/header.jsp
       "%>
  </div>


<br><br><br><br>

<div class="sub-body-title" style="height
     :90px">
<h3>&nbsp;&nbsp;${patient.name}&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;
UPCD ID:${patient.upcdId}</h3>
<font size=3>&nbsp;${patient.gender}&nbsp;&
     nbsp; ${patient.age} yrs &nbsp;(${
     patient.birthday})<br>
</font><font size=2>&nbsp;${patient.address
     }</font>

</div>
<br>

<c:if test="${message == null}">
<div style="z-index
     :9999999999999999999999999999999999999999999999999999;
     postion:relative;">
<center>

   &nbsp;${dash}&nbsp;&nbsp;${menu}
</center>


</div>
        <br>
<c:if test="${taskRSname != 'DentalChart
     '}">
<div id="layout-menu" style="top:20%;
     display:block;">
<br><br><br><br> <br><br>
<%@include file="/WEB-INF/views/patientMenu
     .jsp"%>
</div>
<div id="layout-body" style="top:20%;border
     :none;z-index:-1;">
<br><br><br><br> <br><div class="form-body"
      style="top:20%;border:none;">
</c:if>

<c:if test="${taskRSname == 'DentalChart
     '}">

<div style="top:20%;border:none;width
     =100%">
<br><br><br><br> <br><br>
<div style="top:20%;border:none;width
     =100%">
</c:if>








<!-----------------------------------------------------------------
     SCRIPTS-------------------------------->
<script>

$(document).ready(function(){
$("input[type=button],input[type=submit]").
     attr("class", "classname");
var formtaskname='<c:out value="${
     taskRSname}"/>';
if(formtaskname!='DentalChart'){



var status='${fn:containsIgnoreCase(
     fieldValues,'approved')}';

if(status=='true'){
 $("input[name=clinician][type=text]").val
      ('<c:out value="${sessionScope.
      sessionName}"/>');
}
$("input[name=clinician][type=text]").attr
     ("readonly", "readonly");

$("input[name=faculty][type=text]").attr("
     readonly", "readonly");

var role='${fn:containsIgnoreCase(fn:join(
     sessionScope.currentRoleList,','),'
     faculty')}';
if(role=='true'){
$("input[name=faculty][type=text]").val('<c
     :out value="${sessionScope.sessionName
     }"/>');

}
if(status=='true'){
$("input[name=faculty][type=text]").val('
     ');

}
$("input[name=faculty][type=text]").attr("
     readonly", "readonly");

   if($("input[name=sec][type=hidden]").val
        ()!=null && $("input[name=sec][type=
        hidden]").val()!=''){
$('select[name="section"]').val($("input[
     name=sec][type=hidden]").val());
}
else{
$('select option:first-child').attr("
     selected", "selected");
  $('input[type=checkbox]').filter(function
        (i,field) {
   var name=field.name+'text';
 if(document.getElementById(name)==null){
 var $ctrl = $(document.createElement("
      input")).attr({
       id:     name
      ,name:   name
      ,value:  ' '
      ,type:   'text'
      ,style:  'display:none;z-index:-99;
           visibility: hidden;width:0px '
      ,size: 0

   })

var lbl = '<p></p>';


$($("[name="+field.name+"]")[0]).before(
     $ctrl);
}
});


$('input[type=checkbox]').click(function()
     {
 $('input[type=text]').filter(function(i,
      field) {



});
 var checkname=this.name+"text";
 var checkstr="";
  if(this.checked==true){
  $('input[name='+ checkname+']').val($('
       input[name='+checkname+']').val() +
       this.value+ ',' );
 }
 else if(this.checked==false){
  document.getElementById(checkname).value=
       document.getElementById(checkname).
       value.replace(this.value+',',"");
 }
    // alert(document.getElementById(
         checkname).value);
});
}
$('input[id=continuebtn]').click(function()
     {
 $("input[id=continue]").val('continue');
});
$("input[type=submit][value='Submit']").
     click(function(){
 $("input[id=continue]").val('Submit');
});
$("input[type=submit][value='Approve']").
      click(function(){
 $("input[id=continue]").val('Approve');
});
$("input[type=submit][value='Reject']").
      click(function(){
 $("input[id=continue]").val('Reject');
});
$("input[type=submit][value='Refer To Other
       Section']").click(function(){
 $("input[id=continue]").val('Continue');
});
$("input[type=submit][value='End Case']").
      click(function(){
 $("input[id=continue]").val('End');
});
$("input[type=submit][value='Claim']").
      click(function(){
 $("input[id=continue]").val('Claim');
});
$("input[type=submit][value='Skip']").click
      (function(){
 $("input[id=continue]").val('Skip');
});
$("input[type=submit][value='Save Remarks
      ']").click(function(){
 $("input[id=continue]").val('Save Remarks
       ');
});
var values="";
$("input[type=submit][value='Save and
      Submit']").click(function(){
 $("input[id=continue]").val('Save and
       Submit');

 x=$("form textarea,input[type=text],input[
       type=radio]").serializeArray();
 $.each(x, function(i, field){
   values= values+ field.value + " | ";

 });
 $("input[name=values]").val(values);

});

$("input[type=submit][value='Save']").click
      (function(){
 $("input[id=continue]").val('Save');

x=$("form textarea,input[type=text],input[
      type=radio]").serializeArray();
 $.each(x, function(i, field){
   values= values+ field.value + " | ";

 });
 $("input[name=values]").val(values);
});


  $("button").click(function(){




  });

});
</script>

<script>
<%@include file="/WEB-INF/views/function.js
      "%>
<%@include file="/WEB-INF/views/calendar.js
      "%>
 <%@include file="/WEB-INF/views/
       formsValidate.js"%>


</script>
<!------------------------------------------------------------
      SCRIPTS---------------------------------->
<c:if test="${notif !=null}">
<font color="green">${notif}</font>
</c:if>

    <form id="form" name="form" action="
         complete?patientid=${patientid}&
         instanceid=${instanceid}&pos=${pos
         }&idTask=${idTask}" method="POST"
         enctype="multipart/form-data"
         target="_parent">
    <form id="form" name="form" action="
         complete?patientid=${patientid}&
         instanceid=${instanceid}&pos=${pos
         }&idTask=${idTask}" method="POST"
         enctype="multipart/form-data"
         target="_parent">
     <input type="hidden" name="fields"
          value="">
     <input type="hidden" id=values name=
          values value="">
     <input type="hidden" name="formname"
          value="${taskRSname}">
     <input type="hidden" name="userRole"
          value="Faculty">
     <input type="hidden" id="continue"
          name="continue" value="">
     <input type="hidden" id="approvalRes"
          name="approvalRes" value="">
     <input type="hidden" id="dashid" name
          ="dashid" value="${dashid}">
     <input type="hidden" id="dashname"
          name="dashname" value="${dashname
          }">

<c:if test="${app !=null}">
   <input type="hidden" id="app" name="app"
        value="${app}">

</c:if>

     <input type="hidden" id="patientid"
          name="patientid" value="${
          patientid}">
     <input type="hidden" id="viewflag"
          name="viewflag" value="false">
     <input type="hidden" name="patient_id"
           value="${patient_id}" />
     <input type="hidden" name="version"
          value="${version}" />
     <input type="hidden" name="is_current"
           value="${is_current}" />

  ${msginputfindings}
     ${html}
<c:if test="${valueLists!=null}">
<c:forEach var="values" items="${valueLists
     }" varStatus="loop">
<c:choose>
    <c:when test='${(loop.index)%2 eq 0}'>
     <c:set var="rowColor" value="even"
          scope="page"/>
   </c:when>
   <c:otherwise>
     <c:set var="rowColor" value="odd"
          scope="page"/>
   </c:otherwise>
 </c:choose>

<tr class="${rowColor}">

 <c:forTokens items="${values}"
      delims="|"
      var="val"
      varStatus="status"
   >
     <c:if test="${status.index<=colCount-1
           && taskRSname!='
          ConsultationsAndFindings'}">
    <td>   <c:out value="${val}" /></td>
 </c:if>
    <c:if test="${status.index<=colCount &&
          taskRSname=='
        ConsultationsAndFindings'}">
    <td>   <c:out value="${val}" /></td>
 </c:if>
 </c:forTokens>

</tr>
</c:forEach>
</table>
</div>
</c:if>
    <!----------DENTAL CHART--->
<c:if test="${taskRSname=='DentalChart'}">
<%@include file="/WEB-INF/views/dentalchart
      .jsp"%>



</c:if>



    <!----END OF DENTAL CHART-->
```

```
<!--
   <c:if test="${taskRSname != '
        AssignToClinician'}">
   <button>Save Record</button>
   </c:if>

   <c:if test="${taskRSname == '
        AssignToClinician'}">
   <button>Assign To</button>
   </c:if>

   <input type="submit" value="Continue"
        id='continuebtn' onclick="
        javascript:document.getElementById
        ('continue').value='continue;
        javascript:document.getElementById
        ('approvalRes').value='Approve'">
   -->
   </form>
  <div>
   <!-- <a class="edit" href="#">Edit</a>
        -->
   </div>
  <br>

  <c:if test="${processId == 'UPCDDentISt.
       CreateNewPatient'}">
   <form action="/dentISt/forms/dentist/
        login/index.html" method="link">
   <input type=submit value="Cancel" style
        ="font-size:14;"><br>
   </form>
  </c:if>


 <!--
  <c:if test="${(processId != 'UPCDDentISt
       .CreateNewPatient' && fn:contains(
       processId, 'AllSections')) ||
       processId == 'OralDiagnosisSection.
       DentalChart'}">
   <form action="allSections" method="link
        ">
  </c:if>
  <c:if test="${fn:contains(processId, '
       OralDiagnosisSection') && processId
        != 'OralDiagnosisSection.DentalChart
        '}">
    <form action="http://agila.upm.edu.ph
         :8090/dentISt/app/viewForm?
         processId=OralDiagnosisSection.
         PatientInformation" method="link">
    <input type=submit value="Cancel" style
         ="font-size:14;"><br>
    </form>
   </c:if>
  -->

 <br>
 <br>
  </div>
 </div>
 <c:if test="${taskRSname != 'DentalChart
      '}">
     <script type="text/javascript">
          DisplayFormValues();      </script>
 </c:if>
 </c:if>
 <c:if test="${message != null}">
     <div id="layout-menu" style="top:20%;
          display:block;">
 <br><br><br><br> <br><br>
 <%@include file="/WEB-INF/views/patientMenu
      .jsp"%>
 </div>
 <div id="layout-body" style="top:20%;border
      :none;">
 <br><br><br><br> <br><br>
 <div class="form-body" style="top:20%;
      border:none;">



 </div>
 </div>

 ${message}
 </c:if>
<%@include file="/WEB-INF/views/footer.jsp
     "%>
 </body>
 </html>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

 pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
       Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">         <
     title>DentISt</title>

</head>
<body>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

        </div>




<div id="formlayout-body" style="left:1%;
     right:1%">
<br>
<div class="box" style="top:15%;left-margin
     :15%;">

<div class="sub-body-title">Manage
     Appointments</div>
<br>
<c:if test="${flag=='all'}">
<input type=hidden name="flag" value="all">
<h2>Upcoming Appointments of all Clinicians
     </h2>
</c:if>
<c:if test="${flag!='all'}">
<h2>Upcoming Appointments</h2>
</c:if>

<hr>

<c:if test="${empty appointmentLists}">
<br>
</c:if>
<c:if test="${not empty appointmentLists}">
<table class="list-table">
<thead>
<tr class="head-list-table">
<td>Id :</td>
<td>Appointment Date :</td>
<td>Clinician :</td>
<td>Patient :</td>
</tr>
</thead>
<tbody>
  <c:forEach items="${appointmentLists}" var
       ="appointmentList" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
       <c:set var="rowColor" value="even"
            scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
            scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">



 <td>
         ${appointmentList.appointmentid}
    </td>
    <td>
```

```
        ${appointmentList.appointmentdate}

    </td>
  <td>
          ${appointmentList.
               appointmentclinician}
    </td>
  <td>
  <c:if test="${flag!='all'}">
  <a href="task?idTask=${appointmentList.
       idtask}&nameTask=${appointmentList.
       nametask}&patientid=${appointmentList.
       patientid}&flag=true&pos=0">${
       appointmentList.patientname}</a>
    </c:if>
<c:if test="${flag=='all'}">
 ${appointmentList.patientname}    </c:if>
```

```
         </td>

  </tr>
   </c:forEach>
<br><br>
</c:if>


</div>
</div>
</div>
<br>
<br>
<%@include file="/WEB-INF/views/footer.jsp
     "%>
</body>
</html>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl
     /functions" prefix="fn" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">        <
     title>DentISt</title>
<link rel="stylesheet" type="text/css" href
     ="${pageContext.request.contextPath}/
     theme.css" />
        <script src="//ajax.googleapis.com/
             ajax/libs/jquery/1.8.3/jquery.
             min.js" >
        </script>

<!-------------------------------------------
     SCRIPTS--------------------------->
        <script type="text/javascript">
    function DisplayFormValues()
    {


        var elem = document.getElementById('
             form').elements;
  var str=' <c:out value="${fieldValues
       }"/>';

   var n=str.split("|");
   var j=0;
   var temp="";
        for(var i = 0; i < elem.length; i++)
        {
   if(str!="" && (n[j]!=" null " && n[j]!="
        null ") && n[j]!=null && n[j]!='   ')
       {
   if(n[j]!="null "){
   n[j]=jQuery.trim(n[j]);
           if(elem[i].type!='hidden' && elem
                [i].type!='checkbox' && elem[
                i].type!='radio' && elem[i].
                type!='submit' && elem[i].
                type!='button'){

           elem[i].value =n[j];
   if(elem[i-1].type=='radio' && elem[i].
        value!=null && elem[i].value!="" &&
        elem[i].value!="   "){
     var name=elem[i-1].name + "span";
     if(document.getElementById(name)==null)
          {}
     else{document.getElementById(name).
          style.display = "inline";}
    }

   j++;

           }
   if(elem[i].type=='checkbox' && elem[i+1].
        name!=elem[i].name){
```

```
     var value=n[j].split(",");
     for (var k=0; k<value.length; k++){
        checkval(jQuery.trim(value[k]));

     }
     j++;
    }
    if(elem[i].type=='radio' && elem[i+1].
         name!=elem[i].name){
      if(temp==" "+elem[i].value+" " || temp
           ==elem[i].value){
     elem[i].checked=true;}

          j++;
    }
    else if(elem[i].type=='radio' && elem[i
         +1].name==elem[i].name){

     temp=n[j];

     if(temp==" "+elem[i].value+" " || temp==
          elem[i].value){
     elem[i].checked=true;}
    }
    }
    }
    elem[i].style='border: 1px solid;
         background-color: #fff; color: #000;
         font-size: 12px';
    if(elem[i].value=='Save'){elem[i].style='
         display:none;visibility:hidden;';}
    else if(elem[i].type=='button' && elem[i
         ].value=='Print this form'){elem[i].
         class="classname"}

    else if(elem[i].id=='approvebtn'){}
    else if(elem[i].type=='hidden'){}
    else{elem[i].readOnly=true;elem[i].
         disabled=true;}
               //if(elem[i].type=='button'
                   && elem[i].id!='
                   continuebtn' && elem[i].
                   id!='selectall' && elem[
                   i].name!='choice' &&
                   elem[i].name!='choice1'
                   && elem[i].name!='
                   choice2' && elem[i].name
                   !='choice3'){elem[i].
                   style='display:none;
                   visibility:hidden;';}
    //else
        }
        $("input[name=clinician][type=text][
             value=' ']").val('<c:out value="
             ${sessionScope.sessionName}"/>')
             ;
  $("input[name=clinician][type=text]").attr
       ("readonly", "readonly");


  $("input[name=faculty][type=text]").attr("
       readonly", "readonly");

  var role='${fn:containsIgnoreCase(fn:join(
       sessionScope.currentRoleList,','),'
       faculty')}';
  if(role=='true'){
  $("input[name=faculty][type=text][value='
       ']").val('<c:out value="${sessionScope.
       sessionName}"/>');
  $("input[name=faculty][type=text]").attr("
       readonly", "readonly");
```

```
 }
     }

function checkval(val){
$("input:checkbox[value="+val+"]").attr("
     checked", true);
}
</script>


<script type="text/javascript">
         $(document).ready(function () {
          $("[id*=forms]").click(function ()
               {



       $('#menudiv').show();

                  $('#menuheader').hover(
                       function () {

    //hide its submenu
                  $('#menudiv').show();

              }); $('#menudiv').slideDown
                   (100);
    $("iframe").width("75%");

                });

              $('#dentalchart').click(function
                   dchart() {


                  //hide its submenu
                  $('#menudiv').slideUp(100);


  $('#menuheader').hover(function () {

    //hide its submenu
                  $('#menudiv').show();

              },function () {

    //hide its submenu
                  $('#menudiv').hide();

              });
  $("iframe").width("100%");
              });

  $('#forms').click(function () {
   $("iframe").width("75%");


       $('#menudiv').show();

                  $('#menuheader').hover(
                       function () {

    //hide its submenu
                  $('#menudiv').show();

              }); $('#menudiv').slideDown
                   (100);


                });


         });

    </script>



<!--------------------------------------------------------
     SCRIPTS---------------------------------->


<script language="JavaScript">
function changeSrc(element){

 var href=element.href.split('?', 2);

 var src=href[1].split('&', 6);
```

```
 var idTask="";
 var nameTask="";
 for(var i=0; i<6;i++){
  if(src[i].indexOf("idTask") >= 0){
   idTask=src[i];
  }
  else if(src[i].indexOf("nameTask") >= 0){
   nameTask=src[i].replace("nameTask","
        formname");
  }

 }
   var vers=document.getElementById('version
        ').href.split('?',2);
   var ver=vers[1].split('&',3);
 document.getElementById('version').href=
      vers[0]+"?"+ver[0]+"&"+ver[1]+"&" +
      idTask+"&"+nameTask;

}


 function insertSpace(name){
  var myString = name;
var newString = "";
var wasUpper = false;
for (var i = 0; i < myString.length; i++)
{
    if (!wasUpper && myString[i] == myString
         .toUpperCase()[i])
    {
        newString = newString + "\n";
        wasUpper = true;
    }
    else
    {
        wasUpper = false;
    }
    newString = newString + myString[i];
}
document.write(newString);

 }

 function toggle(id) {
  if (id == 'add') {

    var welements = document.
         getElementsByName('task[]'); //
         Removed [0], that gets the **1st**
         node, not the NodeList.
    for (var i = 0, j = welements.length; i
         < j; i++) {
     var process = welements[i];

     var parts = process.href.split('=', 8);
     process.href = 'task?idTask='+parts
          [1]+"="+parts[2]+"="+parts[3]+"="+
          parts[4]+"="+parts[5]+"="+parts
          [6]+"="+parts[7];


    }
    var framesrc=document.getElementById('
         frame').src.split('=', 8);
    document.getElementById('frame').src='
         http://127.0.0.1:8090/dentISt/app/
         task?idTask='+framesrc[1]+"="+
         framesrc[2]+"="+framesrc[3]+"="+
         framesrc[4]+"="+framesrc[5]+"="+
         framesrc[6]+"="+framesrc[7];
    document.getElementById('canceldiv').
         style='display: block;';
   }
  else if (id == 'view') {
    var welements = document.
         getElementsByName('task[]'); //
         Removed [0], that gets the **1st**
         node, not the NodeList.
    for (var i = 0, j = welements.length; i
         < j; i++) {
     var process = welements[i];

     var parts = process.href.split('=', 8);
     process.href = 'viewForm?idTask='+parts
          [1]+"="+parts[2]+"="+parts[3]+"="+
          parts[4]+"="+parts[5]+"="+parts
          [6]+"="+parts[7];


    }
    var framesrc=document.getElementById('
         frame').src.replace('task?','
         viewForm?');
    document.getElementById('frame').src=
         framesrc;
    document.getElementById('canceldiv').
         style='display: none;';

    }
  }
</script>
<script>
$(function() {
toggle(id);
  $("#refresh").click(function() {
     $("#leads").load("#leads")
  })
})
</script>
</head>
<body>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

        </div>
<!------------------------------------------------------------
     SCRIPTS---------------------------------------->
<script>
$(document).ready(function(){
$('input[id=approvebtn]').click(function(){
 $("input[id=approve]").val('approve');
});
});
</script>
<!--

-->
<script>
<%@include file="/WEB-INF/views/function.js
     "%>
</script>
<!------------------------------------------------------------
     SCRIPTS---------------------------------------->



<style type="text/css">
input[type = "text"][disabled]{
 color: #000;
 border: 1px solid #fff;
 disabled:true;
 background-color: white;
 font-size:16px;
 font-weight:italic;
 font-family:verdana;
 align-text:left;

}
textarea[disabled]{
 color: #000;
 border: 1px solid #fff;
 disabled:true;
 background-color: white;
 font-size:16px;
 font-weight:italic;
 font-family:verdana;
 margin-top: 100px;
}


</style>
<c:if test="${taskRSname== 'DentalChart'}">
<br><br>
<%@include file="/WEB-INF/views/scripts.js
     "%>
</c:if>
<!--<h3>Patient: ${patient.name}</h3>-->




<br><br><br><br>

<div class="sub-body-title" style="height:90
     px">
<h3>&nbsp;&nbsp;${patient.name}&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
     ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
     nbsp;
UPCD ID:${patient.upcdId}</h3>
<font size=3>&nbsp;${patient.gender}&nbsp;&
     nbsp; ${patient.age} yrs &nbsp;(${
     patient.birthday})<br>
</font><font size=2>&nbsp;${patient.address
     }</font>

</div>
<br>


<center>

<a href="listViewVersions?patientid=${
     patientid}&instanceid=${instanceid}&
     idTask=${idTask}&formname=${taskRSname}"
      id="version" onclick="changeSrc(this)">
     View Versions</a>
</center>

<br>
<div style="color:#756EB8;"><hr></div>
<c:if test="${viewOnly !=null}">
<br><br><br><br> <br><br>
<%@include file="/WEB-INF/views/patientMenu.
     jsp"%>
</c:if>
<c:if test="${viewOnly == null}">
<div id="menuheader">
<div class="sub-menu-title" style="width:
     23%" align="center"><b>&nbsp;&nbsp;
     Patient Forms</b></div>


<ul class="sub-menu-subjs" style="left:1%;
     width: 22%;z-index:99" id="menudiv">

     <li>

<c:forEach var="form" items="${formlist}"
     varStatus="loop">
 <c:set var="taskname" value="${form.key}"/>
 <c:if test="${taskname != '
      AssignToODClinician'}">

 <font size="2">

 <c:if test="${taskname != 'DentalChart'}">
 <a name="task[]" href="${fn:replace(form.
      value,'task?', 'viewForm?')}&patientid=
      ${patientid}" id="forms{loop.index}"
      onclick="changeSrc(this);"><script>
      insertSpace("<c:out value='${form.key}'
       />");</script></a>
 </c:if>

 <c:if test="${taskname == 'DentalChart'}">
 <a name="task[]" href="${fn:replace(form.
      value,'task?', 'viewForm?')}&patientid=
      ${patientid}" id="dentalchart" onclick
      ="changeSrc(this);"><script>insertSpace
      ("<c:out value='${form.key}' />");</
      script></a>
 </c:if>
 </font>
  </c:if>


</c:forEach>
</li>
</ul>
</div>
</c:if>

<!--SCRIPTS-START-->
<script language="JavaScript">
<!--
function calcHeight()
{
  //find the height of the internal page
```

```
   var the_height=
     document.getElementById('frame').
         contentWindow.
       document.body.scrollHeight;

   //change the height of the iframe
   document.getElementById('frame').height=
       the_height;


}
//-->
</script>


<!--SCRIPTS-END-->

         <c:if test="${taskRSname != '
             DentalChart'}">

<div id="layout-body" style="top:20%;border:
     none;left:25%;align:right;right:25%;z-
     index:-1;">
<br><br><br><br> <br><br>
<div class="form-body" style="top:20%;border
     :none;left:25%;align:right;right:25%;">
</c:if>

<c:if test="${taskRSname == 'DentalChart'}">

<div style="top:20%;border:none;width=100%">
<br><br>
<div style="top:20%;border:none;width=100%">
<script>$('#menudiv').slideUp(100); $('#
     menuheader').hover(function () { $('#
     menudiv').show();           },function ()
      { $('#menudiv').hide();             });</
     script>
</c:if>
<form id="form" action="complete?patientid=$
     {patientid}&instanceid=${instanceid}&pos
     =${pos}" method="POST" enctype="
     multipart/form-data" target="_parent">
     <input type="hidden" name="fields"
          value="">
     <input type="hidden" id=values name=
          values value="">
     <input type="hidden" name="formname"
          value="${taskRSname}">
     <input type="hidden" name="userRole"
          value="Faculty">
     <input type="hidden" id="approve" name
          ="approve" value="">
     <input type="hidden" id="patientid"
          name="patientid" value="${patientid
          }">
     <input type="hidden" id="viewflag" name
          ="viewflag" value="false">
     <input type="hidden" name="patient_id"
          value="${patient_id}" />
     <input type="hidden" name="version"
          value="${version}" />
     <input type="hidden" name="is_current"
          value="${is_current}" />

<p></p>
      ${html}
<c:if test="${valueLists!=null}">
 <c:forEach var="values" items="${valueLists
      }" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>

 <tr class="${rowColor}">
 <c:forTokens items="${values}"
                delims="|"
                var="val"
                varStatus="status"
      >
            <c:if test="${status.index<=
                 colCount-1 && taskRSname!='
                 ConsultationsAndFindings
                 '}">
     <td>   <c:out value="${val}" /></td>
  </c:if>
     <c:if test="${status.index<=colCount &&
          taskRSname=='
         ConsultationsAndFindings'}">
     <td>   <c:out value="${val}" /></td>
  </c:if>


    </c:forTokens>
</tr>
</c:forEach>
</table>
</div>
</c:if>

    <!---------DENTAL CHART-->
<c:if test="${taskRSname=='DentalChart'}">
<%@include file="/WEB-INF/views/dentalchart.
     jsp"%>


</c:if>
    <INPUT TYPE="button" value="Print this
         form" onClick="window.print()" style
         ="display:inline;">


      <a href="searchPatient" target="
           _parent">Cancel</a>

    </form>




</div>
</div>
<c:if test="${taskRSname != 'DentalChart'}">
    <script type="text/javascript">
         DisplayFormValues();     </script>
    <style>input[type = "submit"]{
  visibility:hidden;
}
</style>
</c:if>
<c:if test="${taskRSname == 'DentalChart'}">
 <style>.tooth {
    pointer-events:none;
}
input[type = "submit"]{
  visibility:hidden;
}

</style>
</c:if>


<%@include file="/WEB-INF/views/footer.jsp
     "%>
</body>
</html>
```

```
<%@ page language="java" contentType="text/
     html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
     core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl
     /functions" prefix="fn" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
       Transitional//EN" "http://www.w3.org/TR
     /html4/loose.dtd">

<html>
    <head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">         <
     title>DentISt</title>
<link rel="stylesheet" type="text/css" href
     ="${pageContext.request.contextPath}/
     theme.css" />
        <script src="//ajax.googleapis.com/
             ajax/libs/jquery/1.8.3/jquery.
```

```
            min.js" >
        </script>

<!-----------------------------------------------------
     SCRIPTS------------------------------->
        <script type="text/javascript">
    function DisplayFormValues()
    {

        var elem = document.getElementById('
            form').elements;
  var str=' <c:out value="${fieldValues
      }"/>';

  var n=str.split("|");
  var j=0;
  var temp="";
       for(var i = 0; i < elem.length; i++)
       {
  if(str!="" && (n[j]!=" null " && n[j]!="
       null ") && n[j]!=null && n[j]!='  ')
      {
  if(n[j]!="null "){
  n[j]=jQuery.trim(n[j]);
           if(elem[i].type!='hidden' && elem
                [i].type!='checkbox' && elem[
                i].type!='radio' && elem[i].
                type!='submit' && elem[i].
                type!='button'){

           elem[i].value =n[j];
  if(elem[i-1].type=='radio' && elem[i].
       value!=null && elem[i].value!="" &&
       elem[i].value!=" " ){
    var name=elem[i-1].name + "span";
    if(document.getElementById(name)==null)
         {}
    else{document.getElementById(name).
         style.display = "inline";}
   }

  j++;

           }
  if(elem[i].type=='checkbox' && elem[i+1].
       name!=elem[i].name){
   var value=n[j].split(",");
   for (var k=0; k<value.length; k++){
      checkval(jQuery.trim(value[k]));

   }
   j++;
  }
  if(elem[i].type=='radio' && elem[i+1].
       name!=elem[i].name){
    if(temp==" "+elem[i].value+" " || temp
         ==elem[i].value){
    elem[i].checked=true;}

       j++;
  }
  else if(elem[i].type=='radio' && elem[i
       +1].name==elem[i].name){

   temp=n[j];

   if(temp==" "+elem[i].value+" " || temp==
        elem[i].value){
    elem[i].checked=true;}
  }
  }
  }
  elem[i].style='border: 1px solid;
       background-color: #fff; color: #000;
       font-size: 12px';
  if(elem[i].value=='Save'){elem[i].style='
       display:none;visibility:hidden;';}
  else if(elem[i].type=='button' && elem[i
       ].value=='Print this form'){elem[i].
       class="classname"}

  else if(elem[i].id=='approvebtn'){}
  else if(elem[i].type=='hidden'){}
  else{elem[i].readOnly=true;elem[i].
       disabled=true;}
                 //if(elem[i].type=='button'
                      && elem[i].id!='
                      continuebtn' && elem[i].
                      id!='selectall' && elem[
                      i].name!='choice' &&
                      elem[i].name!='choice1'
                      && elem[i].name!='
                      choice2' && elem[i].name
                      !='choice3'){elem[i].
                      style='display:none;
                      visibility:hidden;';}
       //else
            }

       }

  function checkval(val){
  $("input:checkbox[value="+val+"]").attr("
      checked", true);
  }
  </script>


  <script type="text/javascript">
          $(document).ready(function () {
           $("[id*=forms]").click(function ()
                {



          $('#menudiv').show();

                      $('#menuheader').hover(
                           function () {

      //hide its submenu
                      $('#menudiv').show();

                 });  $('#menudiv').slideDown
                      (100);
      $("iframe").width("75%");

                   });

                 $('#dentalchart').click(function
                        dchart() {


                      //hide its submenu
                      $('#menudiv').slideUp(100);


    $('#menuheader').hover(function () {

      //hide its submenu
                      $('#menudiv').show();

                 },function () {

      //hide its submenu
                      $('#menudiv').hide();

                 });
    $("iframe").width("100%");
                 });

    $('#forms').click(function () {
     $("iframe").width("75%");



          $('#menudiv').show();

                      $('#menuheader').hover(
                           function () {

      //hide its submenu
                      $('#menudiv').show();

                 });  $('#menudiv').slideDown
                      (100);


                   });



            });

      </script>



  <!-----------------------------------------------------
       SCRIPTS------------------------------->
```

```
<script language="JavaScript">
function changeSrc(element){

 var href=element.href.split('?', 2);

 var src=href[1].split('&', 6);

 var idTask="";
 var nameTask="";
 for(var i=0; i<6;i++){
  if(src[i].indexOf("idTask") >= 0){
   idTask=src[i];
  }
  else if(src[i].indexOf("nameTask") >= 0){
   nameTask=src[i].replace("nameTask","
        formname");
  }

 }
   var vers=document.getElementById('version
        ').href.split('?',2);
   var ver=vers[1].split('&',3);
 document.getElementById('version').href=
      vers[0]+"?"+ver[0]+"&"+ver[1]+"&" +
      idTask+"&"+nameTask;

}


 function insertSpace(name){
  var myString = name;
var newString = "";
var wasUpper = false;
for (var i = 0; i < myString.length; i++)
{
    if (!wasUpper && myString[i] == myString
        .toUpperCase()[i])
    {
        newString = newString + "\n";
        wasUpper = true;
    }
    else
    {
        wasUpper = false;
    }
    newString = newString + myString[i];
}
document.write(newString);

 }

 function toggle(id) {
  if (id == 'add') {

    var welements = document.
         getElementsByName('task[]'); //
         Removed [0], that gets the **1st**
         node, not the NodeList.
    for (var i = 0, j = welements.length; i
         < j; i++) {
     var process = welements[i];

     var parts = process.href.split('=', 8);
     process.href = 'task?idTask='+parts
          [1]+"="+parts[2]+"="+parts[3]+"="+
          parts[4]+"="+parts[5]+"="+parts
          [6]+"="+parts[7];


    }
    var framesrc=document.getElementById('
         frame').src.split('=', 8);
    document.getElementById('frame').src='
         http://127.0.0.1:8090/dentISt/app/
         task?idTask='+framesrc[1]+"="+
         framesrc[2]+"="+framesrc[3]+"="+
         framesrc[4]+"="+framesrc[5]+"="+
         framesrc[6]+"="+framesrc[7];
    document.getElementById('canceldiv').
         style='display: block;';
   }
  else if (id == 'view') {
    var welements = document.
         getElementsByName('task[]'); //
         Removed [0], that gets the **1st**
         node, not the NodeList.
    for (var i = 0, j = welements.length; i
         < j; i++) {
     var process = welements[i];

     var parts = process.href.split('=', 8);
     process.href = 'viewForm?idTask='+parts
          [1]+"="+parts[2]+"="+parts[3]+"="+
          parts[4]+"="+parts[5]+"="+parts
          [6]+"="+parts[7];


    }

    var framesrc=document.getElementById('
         frame').src.replace('task?','
         viewForm?');
    document.getElementById('frame').src=
         framesrc;
    document.getElementById('canceldiv').
         style='display: none;';

   }
  }
</script>
<script>
$(function() {
toggle(id);
  $("#refresh").click(function() {
     $("#leads").load("#leads")
  })
})
</script>
</head>
<body>
<div id="layout-header">

<%@include file="/WEB-INF/views/header.jsp
     "%>

         </div>

<!-----------------------------------------------
     SCRIPTS--------------------------------->
<script>
$(document).ready(function(){
DisplayFormValues();
$('input[id=approvebtn]').click(function(){
 $("input[id=approve]").val('approve');
});
});
</script>
<!--

-->
<script>
<%@include file="/WEB-INF/views/function.js
     "%>
</script>
<!-----------------------------------------------
     SCRIPTS--------------------------------->




<style type="text/css">
input[type = "text"][disabled]{
 color: #000;
 border: 1px solid #fff;
 disabled:true;
 background-color: white;
 font-size:16px;
 font-weight:italic;
 font-family:verdana;
 align-text:left;

}
textarea[disabled]{
 color: #000;
 border: 1px solid #fff;
 disabled:true;
 background-color: white;
 font-size:16px;
 font-weight:italic;
 font-family:verdana;
 margin-top: 100px;
}


</style>
<c:if test="${taskRSname== 'DentalChart'}">
<br><br>
<%@include file="/WEB-INF/views/scripts.js
     "%>
</c:if>
<!--<h3>Patient: ${patientname}</h3>-->
```

```
<br><br><br><br>

<div class="sub-body-title" style="height:90
      px">
<h3>&nbsp;&nbsp;${patient.name}&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
      ;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
      nbsp;
UPCD ID:${patient.upcdId}</h3>
<font size=3>&nbsp;${patient.gender}&nbsp;&
      nbsp; ${patient.age} yrs &nbsp;(${
      patient.birthday})<br>
</font><font size=2>&nbsp;${patient.address
      }</font>

</div>
<br>


<center>
     <a href="javascript:history.back()">
          Dashboard</a>
</center>

<br>
<div style="color:#756EB8;"><hr></div>
<div id="layout-menu" style="top:20%;display
      :block;">
<br><br><br><br> <br><br>
<%@include file="/WEB-INF/views/patientMenu.
      jsp"%>
</div>

<!--SCRIPTS-START-->
<script language="JavaScript">
<!--
function calcHeight()
{
  //find the height of the internal page
  var the_height=
    document.getElementById('frame').
         contentWindow.
       document.body.scrollHeight;

  //change the height of the iframe
  document.getElementById('frame').height=
       the_height;


}
//-->
</script>


<!--SCRIPTS-END-->

        <c:if test="${taskRSname != '
             DentalChart'}">

<div id="layout-body" style="top:20%;border:
      none;z-index:-1;">
<br><br><br><br> <br><br>
<div class="form-body" style="top:20%;border
      :none;">
</c:if>

<c:if test="${taskRSname == 'DentalChart'}">

<div style="top:20%;border:none;width=100%">
<br><br>
<div style="top:20%;border:none;width=100%">
<script>$('#menudiv').slideUp(100); $('#
      menuheader').hover(function () { $('#
      menudiv').show();           },function ()
       { $('#menudiv').hide();           });</
      script>
</c:if>
<form id="form" action="complete?patientid=$
      {patientid}&instanceid=${instanceid}&pos
      =${pos}" method="POST" enctype="
     multipart/form-data" target="_parent">
     <input type="hidden" name="fields"
          value="">
     <input type="hidden" id=values name=
          values value="">
     <input type="hidden" name="formname"
          value="${taskRSname}">
     <input type="hidden" name="userRole"
          value="Faculty">
     <input type="hidden" id="approve" name
          ="approve" value="">
     <input type="hidden" id="patientid"
          name="patientid" value="${patientid
          }">
     <input type="hidden" id="viewflag" name
          ="viewflag" value="false">
     <input type="hidden" name="patient_id"
          value="${patient_id}" />
     <input type="hidden" name="version"
          value="${version}" />
     <input type="hidden" name="is_current"
          value="${is_current}" />

<p></p>
     ${html}
<c:if test="${valueLists!=null}">
 <c:forEach var="values" items="${valueLists
      }" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq 0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>

 <tr class="${rowColor}">
 <c:forTokens items="${values}"
                 delims="|"
                 var="val"
                 varStatus="status"
      >
            <c:if test="${status.index<=
                 colCount-1 && taskRSname!='
                 ConsultationsAndFindings'}">
     <td>   <c:out value="${val}" /></td>
  </c:if>
     <c:if test="${status.index<=colCount &&
          taskRSname=='
          ConsultationsAndFindings'}">
     <td>   <c:out value="${val}" /></td>
  </c:if>


    </c:forTokens>
</tr>
</c:forEach>
</table>
</div>
</c:if>

    <!---------DENTAL CHART-->
<c:if test="${taskRSname=='DentalChart'}">
<%@include file="/WEB-INF/views/dentalchart.
     jsp"%>


</c:if>


    </form>





</div>
</div>
<c:if test="${taskRSname != 'DentalChart'}">
    <script type="text/javascript">
         DisplayFormValues();      </script>
    <style>input[type = "submit"]{
  visibility:hidden;
}
</style>
</c:if>
<c:if test="${taskRSname == 'DentalChart'}">
 <style>.tooth {
    pointer-events:none;
}
input[type = ”submit”]{
  visibility:hidden;
}

</style>
</c:if>
```

```
<%@include file=”/WEB-INF/views/footer.jsp
    ”%>
</body>
</html>
```

```
<%@ page language=”java” contentType=”text/
     html; charset=ISO-8859-1”
    pageEncoding=”ISO-8859-1”%>
<%@taglib uri=”http://java.sun.com/jsp/jstl/
     core” prefix=”c” %>
<%@taglib prefix=”form” uri=”http://www.
     springframework.org/tags/form” %>

<!DOCTYPE html PUBLIC ”-//W3C//DTD HTML 4.01
      Transitional//EN” ”http://www.w3.org/TR
     /html4/loose.dtd”>
<html>
<head>
<meta http-equiv=”Content-Type” content=”
     text/html; charset=ISO-8859-1”>
<title>Insert title here</title>
</head>
<body>
```

```
<div class=”form-menu”>
<c:url var=”processesURL” value=”/forms/
     dentist/simpleForm/dentistDesigner.html”
      />
<div class=”sub-menu-title”>Manage Workflow
     </div>

<div>
<ul class=”sub-menu-subjs”>
<li><a href=”${processesURL}”>Manage
     Processes</a></li>
</ul>
</div>

</div>
</body>
</html>
```

```
<!DOCTYPE composition PUBLIC ”-//W3C//DTD
     XHTML 1.0 Transitional//EN”
     ”http://www.w3.org/TR/xhtml1/DTD/xhtml1-
          transitional.dtd”>
<ui:composition xmlns=”http://www.w3.org
     /1999/xhtml”
 xmlns:s=”http://jboss.com/products/seam/
      taglib”
 xmlns:ui=”http://java.sun.com/jsf/facelets”
 xmlns:f=”http://java.sun.com/jsf/core”
 xmlns:h=”http://java.sun.com/jsf/html”
 xmlns:rich=”http://richfaces.org/rich”
 xmlns:a4j=”http://richfaces.org/a4j”
      template=”layout/template.xhtml”>

 <ui:define name=”body”>
  <style>
.col,.col2 {
 width: 50%;
 vertical-align: top;
}
</style>
  <h:form>
   <a4j:jsFunction name=”refreshassets” id=”
        refreshassets”
      action=”#{StandaloneManager.addAsset()
          }” />
  </h:form>
  <h:form>
   <a4j:jsFunction name=”editassets” id=”
        editassets”
      action=”#{StandaloneManager.editAsset()
          }” />
  </h:form>
  <h:form>
   <a4j:jsFunction name=”refreshModal” id=”
        refreshModal”
     bypassUpdates=”true” ajaxSingle=”true”
          reRender=”myModalDiv” />

  </h:form>

  <a4j:outputPanel id=”assettree”>
   <h:panelGrid columns=”2” width=”100%”
        columnClasses=”col1,col2”>
    <a4j:outputPanel  id=”tree”>
    <a4j:form>
     <rich:tree value=”#{allPackages}” var=”
          pkg”
      nodeSelectListener=”#{
           StandaloneManager.processSelection
           }”
      ajaxSubmitSelection=”true”
      reRender=”assetinfopanel”>
      <rich:treeNode onclick=””>
       <h:outputText value=”#{pkg}” />
      </rich:treeNode>
     </rich:tree>
    </a4j:form>
    </a4j:outputPanel>

    <a4j:outputPanel id=”assetinfopanel”>
     <a4j:form>
```

```
      <rich:panel rendered=”#{
           StandaloneManager.
           canShowPackageInfo()}” reRender=”
           assettree”>
       <f:facet name=”header”>
        <h:outputText
         value=”Package selected: ${
              StandaloneManager.
              selectedPackage.name}”></h:
              outputText>
       </f:facet>
       <h:panelGrid columns=”3”>
        <h:outputText value=”Enter new
             process name:” />
        <a4j:region>
        <h:inputText value=”${
             StandaloneManager.newProcessName
             }” >
          <a4j:support event=”onkeyup”
               ajaxSingle=”true”
               actionListener=”#{
               StandaloneManager.
               changeProcessName}” />
              </h:inputText>
        </a4j:region>
        <a4j:commandButton value=”CREATE NEW
              PROCESS” action=”#{
             StandaloneManager.createAsset()
             }”
            oncomplete=”javascript:
                 refreshModal();Richfaces.
                 showModalPanel(’EditPanel’,{
                 left:’auto’, top:’auto’});”
                 reRender=”tree”
            style=”font-size:14px”/>

       </h:panelGrid>
      </rich:panel>
      <rich:panel rendered=”#{
           StandaloneManager.canShowAssetInfo
           ()}”>
       <f:facet name=”header”>
        <h:outputText
         value=”Asset selected: ${
              StandaloneManager.selectedAsset
              .name}.${StandaloneManager.
              selectedAsset.format}”></h:
              outputText>
       </f:facet>
       <h:panelGrid columns=”2”>
        <h:outputText value=”Name: ” />
        <h:outputText value=”#{
             StandaloneManager.selectedAsset.
             name}” var=”name”/>
        <h:outputText value=”Format: ” />
        <h:outputText value=”#{
             StandaloneManager.selectedAsset.
             format}” var=”format”/>
        <h:outputText value=”Version: ” />
        <h:outputText value=”#{
             StandaloneManager.selectedAsset.
             version}” />
        <h:outputText value=”Created on: ”
        />
      <h:outputText value="#{
          StandaloneManager.selectedAsset.
          created}" />
      <h:outputText value="Created by: "
          />
      <h:outputText
        value="#{StandaloneManager.
          selectedAsset.createdby}" />
      <h:outputText value="Description: "
          />
      <h:outputText
        value="#{StandaloneManager.
          selectedAsset.description}" />
      <h:outputText value="Last Modified
          on: " />
      <h:outputText
        value="#{StandaloneManager.
          selectedAsset.lastmodified}" />
      <a4j:commandButton value="EDIT
          PROCESS" ajaxSingle="true"
          onclick="javascript:editassets()
          "
          oncomplete="javascript:
              refreshModal();Richfaces.
              showModalPanel('EditPanel',{
              left:'auto', top:'auto'});"
          style="font-size:14px" />
      <a4j:commandButton value="DELETE
          PROCESS" ajaxSingle="true" id="
          deletelink"
                oncomplete="#{rich:
                    component('
                    confirmation')}.
                    show()" reRender
                    ="assetinfopanel
                    " style="font-
                    size:14px" />




    </rich:panel>
    <rich:modalPanel id="EditPanel"
      width="1300" height="650">
      <f:facet name="header">
        <h:outputText value="Edit Process"
          />
      </f:facet>
      <f:facet name="controls">
        <h:graphicImage value="img/close.gif
          " style="cursor:pointer"
          onclick="javascript:refreshassets()
            ;javascript:refreshModal();
            Richfaces.hideModalPanel('
            EditPanel')" reRender="
            assettree"/>
      </f:facet>
      <s:div id="myModalDiv">
        <h:panelGrid columns="1" width
          ="60%">
          <iframe id="myIframe" src="#{
              StandaloneManager.iframeurl}"
            width="1280" height="600" />
          </h:panelGrid>
        </s:div>
      </rich:modalPanel>

<rich:modalPanel id="confirmation" width
    ="400" height="90">
 <f:facet name="header">Are you sure you
    want to delete this process?</f:facet>
 <f:facet name="controls">
 <h:graphicImage value="img/close.gif" style
    ="cursor:pointer"
   onclick="Richfaces.hideModalPanel('
      confirmation')"/>
 </f:facet>
 <table width="100%">
        <tbody>
          <tr>
            <td align="center" width
                ="50%"><a4j:commandButton
                value="Yes"
                          ajaxSingle="true
                              " action="#{
                              StandaloneManager
                              .deleteAsset
                              ()}"
                          oncomplete="#{
                              rich:
                              component('
                              confirmation
                              ')}.hide();"
                          reRender="
                              assettree"
                              style="width
                              :80px;font-
                              size:16px
                              "/></td>
   <td align="center" width="50%"> <a4j:
       commandButton
                          value="Cancel"
                          onclick="#{rich:
                              component('
                              confirmation
                              ')}.hide();
                              return false
                              ;" style="
                              width:100px;
                              font-size:16
                              px"/>
                   </td>
          </tr>
        </tbody>
      </table>
</rich:modalPanel>
     </a4j:form>
    </a4j:outputPanel>
   </h:panelGrid>
  </a4j:outputPanel>


 </ui:define>
</ui:composition>
```

```
</div><%@ page language="java" contentType="
      text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/
      core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
      springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN" "http://www.w3.org/TR
      /html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="
      text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/libs
      /jquery/1.8.3/jquery.min.js" >
</script>

</head>
<body>
<br><br><br><br>
 <iframe src ="http://agila.upm.edu.ph:8090/
      dentISt/dentistDesigner.html" width
      ="100%" height="700" name="frame" id="
      frame" style="border:none;top:30%;">


</body>
</html><div>
```

XII. Acknowledgement

Thank you God for giving me courage, strength and perseverance during my SP and indeed throughout my life.

Thank you to my family. Thank you Mama for always being there for me. You are indeed my best friend. Thank you sa pagencourage sakin lalo na kapag stress na school, sa pagtyaga sa mga rants at iyak ko kapag may bad day, thank you for always reminding me na alagaan ang sarili at ang health kahit na hindi ko sinusunod madalas hehehe. I'm sorry because I know you're also stressed when I'm stressed, nakakadagdag pa ako. Thank you Papa for being the best dad. You're strict but in a good way. Thank you for always reminding me to make time for family and play at hindi puro aral lang. Your lessons kept me sane (or not haha) during my stay in UP. Thank you kasi you still love me kahit ang dami kong pasaway sayo. No words can express how thankful I am to both you. I LOVE YOU BOTH SO MUCH! This achievement is for you at lahat ng maachieve ko pa in the future. To Sherene, Ram, Jhona, Brena, Bok, Kakie and Ken thank you for making me proud. Stay awesome and let us be barkadas forever. I love you all. Pakabait lagi (dahandahan sa mga ipapalibre at hihingin sakin hahaha). This SP is for you too.

Thank you din kay Lolo and Lola dahil ako ang favorite, pinakamabait, most responsible apo niyo(nyehehe joke). Thank you for the unconditional love and for being my second parents. The 50th wedding anniversary gift that you wished from me has finally come true. Sorry because it's one year delayed po. Please always stay healthy. I love you Lolo and Lola. Thank you sa mga titos and titas, Tita Chat, Tito Dave, Tita Lynie, Tito JunJun at lahat po na naguide sakin at nagsupport sakin. Lalo na kay Tita Chat na naging 2nd mother ko while in UP. Thank you po sa pagkupkop at sorry sa pagiging cranky ko lalo na pagstress na. Thank you sa pagalaga at pagintindi sakin. Thank you for giving me courage and confidence to not give up. I owe this achievement to you po.

Thank you to Gela. Friend, finally HUHUHUHUHU BOOM BOOM BOOM HAHAHAHAHAHA (ayan natuluyan na din sa pagkabaliw hahaha). Waaaaah hindi ko alam saan magstart sa sobrang dami ng itethank you ko sayo. Salamat sa mga pasaload, sa pautang, sa wake up calls(sorry lagi ako late haha), sa libre, sa overnights at temporary lipat bahay(super thank you kay Tita Leila at Tito Beloy sorry ulit sa kapal ng mukha ko pabalik balik na lang). Thank you sa heart to heart talks, sa madaming tawa, sa mga chismis na kahit nipromise natin hindi iseshare haha bad. Thank you kasi hindi mo ako inaaway pag may cranky at bitchy moments ako. Sorry din sa mga yon. Thank you sa spazz. Sino nagsabi wala tayong life? May runningman kaya tayo, kdramas, kpop, anime at manga hahaha. Finally ngayon buhay at tao na ulit tayo. Makakapagsuklay at makakaligo na tayo hahaha joke lang. Makakapagparty at makakapaglakwatsa na pala haha. Salamat din pala sa pakikinig sa mga angry/depressed rants ko. Hindi ko talaga alam pano ako magsusurvive sa comsci kung wala ka. Iniimagine ko pa nga lang, naiiyak na ako hahaha. Dagdag mo pa ang kasipagan at pagiging best partner mo sa mga MPs at lalo na SP naks! Thank you sa pagpush sakin hanggang finally eto na talaga yehey!!! Mamimiss pa rin kita kahit 4 years na tayong sawang sawa sa isa't isa haha. Basta thank you for everything <3

Thank you to my Smoochies family, my bestfriends. Edz, Rea, Yanie, Marczia, Shamae. HUHUHU Nakakaiyak, mamimiss ko kayo sobra. Ayoko nalang muna grumaduate dahil sainyo haha. Missing in action na naman ba ang drama ko nito lagi? Lol aigoo. Bilisan niyo dyan sa dent para makapagala na tayo at mabawasan naman ang bucket list natin kahit konti hahaha. Maraming maraming salamat sa lahat. Thank you kasi love niyo ako at love ko din kayo. Thank you sa wagas at maraming tawa. Thank you sa lasing na kwentuhan.

Thank you kasi tanggap niyo ang kabaklaan ko. Thank you dahil marami akong badingan memories dahil sainyo. Masasabi kong masaya ang college life ko at hindi ako nagsisisi sa panghaharass ng UP, sa sleepless nights at sa eyebags na napala ko kasi more important than that ay nameet ko kayo ayieee <3. I love you all mwa! Ang libre ay utang muna. Delayed pa din ang remittance ni FVR hahaha. See you all soon! <333

Thank you to the best thesis adviser, Sir Richard Bryann Chua. Thank you po sa pagtyaga po samin ni Gela, hindi po siguro namin matatapos ang SP if not for your guidance po. Thank you sa pagpush samin na tapusin at pagbutihin ang SP kahit po minsan nakakapressure pero it all paid off in the end. Yehey! Super super thank you po talaga. Sorry po sa stress at sakit ng ulo na nabigay namin sayo hehe.

Thank you sa Comsci Batch 09 at Batch 08 sa pagtanggap samin ni Gela at pagpafeel samin na ”we belong” chos hahaha. Thank you Christine and Lalay sa friendship, sa Running Man, sa marshmallows at sa pagsama sakin maghabol sa last trip ng LRT haha. Thank you to my favorite HI ’09 people Maan, Allen, Pebbe, Janella, Rachelle and Ven kasi kayo ang unang naging friends namin sa comsci kaya hindi kami nasad at naOP hehe <3333 Thank you Hainah, sino magaakala na makakahanap ako ng close friend sa mga oras na akala ko loner na naman ako hehe. Blessing ka talaga sakin <3 Thank you sa libre(babawi ako haha), sa chismis at tawanan haha. Basta thank you sa friendship, dito lang ako lagi para sayo kahit wala na ako sa UPM hehe. Mamimiss kita ¿:)¡

Thank you to my friends who made my college life fun. Sa Dent15 blockmates ko kahit saglit lang tayo nagkasama miss ko pa rin kayo at kayo ang best blockmates ever. Sa mga nakasama kong naiwan sa CAS Jorge, Lyka, Yvette, Christian, Lili and Yves salamat. Ang dami kong firsts na naexperience dahil sa inyo both good and bad hahaha. Thank you to Jerlene and Eloisa my 2NE2 barkada. Grabe

kayo lang ata nakakapagpatawa sakin ng sobrang lakas, yung abot hanggang pedro gil hahaha kaya ko kayo love eh napapasaya niyo ako at nawawala stress pag kayo kasama. How I wish naging coursemates na lang tayo. We miss you, basta bonding at magtawanan pa din tayo paminsan minsan ha. <3

At gusto ko lang ulitin, thank you sa family ko sa unending support and love. Wala ako kung wala kayo. If given a chance, kayo pa din ang pipiliin kong in the next life. Alam ko makakayanan ko lahat basta nandyan kayo. Thank you for being my inspiration. You're the best gift God has given me.