University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

WebPCS:

Web-based Physician Consultation System

A special problem in partial fulfillment

of the Requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Vennedy F. Malabanan

June 2015

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreements	No

ACCEPTANCE SHEET

The Special Problem entitled "WebPCS: Web-based Physician Consultation System" prepared and submitted by Vennedy F. Malabanan in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Perlita E. Gasmen, M.Sc. (*candidate*)
Adviser

EXAMINERS

	Approved	**Disapproved**
1) Gregorio B. Baes, Ph.D. (*candidate*)	_____	_____
2) Avegail D. Carpio, M.Sc.	_____	_____
3) Richard Bryann L. Chua, M.Sc.	_____	_____
4) Ma. Sheila A. Magboo, M.Sc.	_____	_____
5) Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
6) 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.

_____
Ma. Sheila A. Magboo, 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'g
Dean
College of Arts and Sciences

ABSTRACT

WebPCS: Web-based Physician Consultation System is a system which makes use of the Web and mobile technology to improve healthcare coordination and communication between physicians. Java Web is used to integrate into one application the key appointment and consultation tools such as calendar, file transfer, messaging capabilities, and electronic mail (e-mail) and short message service (SMS) notification.

Keywords: e-consultation, appointment, web application

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1.1 Background of the Study

Medical consultation is a procedure, whereby, on request by a physician, another physician reviews the medical history, examines the patient, and makes recommendations as to care and treatment [1].

Consultation is usually sought when physicians with primary clinical responsibility recognize conditions that are beyond their level of expertise or available resources [2]. Despite evidence that the medical literature can answer questions related to patient care, physicians often turn to their colleagues rather than to print resources for answers because colleagues are familiar, reliable, immediately available, and inexpensive. Some factors that influence the information-seeking behavior of physicians are time pressures, convenience of access, and the perceived relevance of the information to the clinical question [3].

Traditionally, physicians obtained input from colleagues by either sending patients for in-person consultations or through *curbside* consultations. The latter are conversations that occurred between the two physicians about patient care when they met in the hospital hallway [4]. The use of communications technology to transmit medical information from one location to another have also evolved to remove geography as a barrier to care by allowing patients to receive care when and where they need it. These technologies bring the promise of ending access-to-care issues for medically underserved areas and under-represented specialties [5].

In the Philippines, the most devastating infectious diseases are predominantly seen in rural and underserved communities [6]. While rural physicians provide the majority of care in the country, they are at times challenged in their efforts to fully meet the needs of their patients when diagnostic uncertainty arises. Because of the advances in diagnosis and treatment of conditions over the past few decades, the need for specialists to be involved in managing patients has increased [4]. Extending healthcare access to rural communities and disadvantaged populations through the use of existing consultation models accentuates the delivery of coordinated high quality care. However, due to their limitations, the feasibility of establishing a web-based consultation system has been explored.

1.2 Statement of the Problem

District hospitals deliver a wide spectrum of care to the sparsely populated regions of the country. In the normal course of a week, the rural physician may attend to several

patients in a community [7]. Since rural physicians manage a broad range of health issues, they require specialist support at some point.

Usually, when consultation is needed, the primary care physician (PCP) formally refers the patient to the specialist. However, such appointments may be delayed because the patient is frequently required to travel some distance and there is a long wait to get in to see a specialist [8]. Other PCPs use curbside consultation with specialists they know and meet by chance. However, since many physicians no longer spend much time in the hospital where these interactions occur, opportunities for such consultations have been reduced [4].

Virtual consultations are used to access specialist advice in a more expedient fashion since it requires no direct contact between the PCP and specialist. These include the telephone consultations, but it was later regarded as inconvenient because a lot of detailed information has to be communicated and a report of consultation is lacking [9]. The use of e-mail systems has also been explored in a variety of settings, including rural health units and military medical centers [8]. However, the transfer of all relevant data from the medical record to the e-mail message makes it impractical to use. In addition, privacy and security are areas of concern that have limited the potential expansion of such systems [10]. Telemedicine is another alternative to traditional consultation, especially in remote areas. Telemedicine involves the use of modern information technology, especially two-way interactive audio or video communications to deliver health services to remote patients and to facilitate information exchange between physicians at some distances from each other [11]. However, the requirement for specialized and often expensive equipment makes it inaccessible to most rural physicians [12].

1.3 Objectives of the Study

The main goal of the study is to address the limitations of the existing consultation models, and improve the communication between physicians and the coordination of care in the country through developing a web system.

a) The system allows an **unregistered user** to:

- Request for an account in the system
- Receive an e-mail notification regarding registration updates

  After his registration, he receives a notification via e-mail that his account request will be reviewed by an administrator. Upon activation, his profile is automatically created.

b) The system allows a **physician** to:

- Book an appointment with another physician

  Given the schedule of his selected physician, the requesting physician can choose his desired consultation time and date, then book an appointment. He can provide the clinical case information using the form that covers the clinical question, laboratory studies, and diagnostic findings. Files can also be provided to support the consultation.

- Receive a reminder via SMS regarding an upcoming appointment

  The requesting physician can set the time when he opts to be reminded of his appointment. The setting also takes effect on the consultant.

- Manage own calendar

  Booked appointment is reflected on both calendars of the consultant and the requesting physician. They can view and cancel an appointment. They can also edit details such as the time and date of appointment.

- Receive notifications via SMS and e-mail regarding appointment updates

  A physician receives notifications via SMS and e-mail regarding a booked, canceled, or re-scheduled appointment.

- Access his clinical cases

  Physician can view the clinical cases he consulted to another physician and the clinical cases consulted to him. He can update their information. He can also add more files and download the supporting files provided.

- Discuss and talk about patient care via chat

- Participate in a forum

  Physician can post a new discussion and delete his existing threads. He can also view, post a reply, subscribe, and unsubscribe to another thread.

- Receive a notification via e-mail regarding forum updates

  A physician receives notifications via e-mail when a physician replied and subscribed to his discussion and when there are updates to his forum subscriptions.

- Generate a consultation report

  The report can be presented in tabular form which can be exported to PDF. It includes the identification of the requesting physician and consultant, consultation date, specialty needed, and reason for consultation.

- View online references available in the system

c) The system allows a **system administrator** to:

- Manage user accounts and account requests

  System administrator can view, edit, and delete existing user accounts. He has the capability to deactivate an existing account. He can also create new user accounts, accept, and decline account requests.

- Add and delete online references in the system

- Generate a consultation tally report

  The report can be presented in tabular and graphical form which can be exported to PDF. It includes the identification of the physician and the tally of the consultations they performed using the system.

1.4 Significance of the Study

Access to care has many facets, but the most commonly heard concerns relate to the time that a patient is required to wait before receiving medically necessary treatment. With the coordinating solution, physicians have the tools they need to increase availability, shorten response times, and improve productivity.

The two-way communication enabled by the system allows rapid clarification of clinical questions and responses between both groups of physicians. If the PCP was able to receive suitable advice from the specialist, face-to-face consultations will be avoided. This implies more windows to necessary clinic visits, and reduces overall wait times and costs of consultation caused by unnecessary travel to see a specialist.

Receipt of the comprehensive advance work on a case before the consultation allows specialists to better understand the medical history of the patient and any previous investigative work done. This also cuts the time spent on evaluating and making decisions since they are building their work on what another physician had already done.

Electronic consultations (e-consultations) can be tracked and ensures delivery of information. Consultations that have not been addressed promptly trigger automated alerts and each stage of the consultation process has notification tag.

1.5 Scope and Limitations

a) A physician can act as either a requesting physician or consultant at the time of consultation.
b) The system allows a physician to view the schedule of colleagues and book an appointment for consultation.
c) Booked consultation is automatically added to the calendars of the requesting physician and consultant.
d) The system provides a structured format of the clinical case which includes abridged and de-identified medical information of the patient.
e) The system implements two modes of sending appointment notifications such as via SMS and e-mail.
f) The system is not associated with a specific medical professional organization or medical institution.
g) The system does not support any laboratory procedures. The information and findings communicated to the consultant are extracted from records outside the system.
h) The system shoulders the cost of sending SMS messages from the server to the recipients.

1.6 Assumptions

a) Credentials of the physicians for account creation is manually checked.
b) Network coverage is available for receiving messages via SMS.
c) The server has sufficient credit to send messages from server to recipients.
d) The physician remuneration in using the system for consultation services is set by the Department of Health.
e) The requesting physician knows the urgency of the clinical case he is about to consult.
f) The requesting physician clearly communicates the medical information and the reason for consultation.
g) The consultant is able to provide expert recommendations despite insufficient medical data to the requesting physician.

h) The consultant responds in a timely manner.

i) The consultant is not liable on the patient although sought for an advice by the requesting physician.

II. REVIEW OF RELATED LITERATURE

E-consultation has been used for some years to facilitate communication between patients and physicians [13]. In the direction from the patient to the physician, information about the patient is collected as accurately as possible to construct a precise interpretation of his status. In the other direction, the patient receives comprehensible information to give his informed consent without any misunderstanding [14].

According to a research, patients want to interact online in a variety of ways. About 51% say they would use this communication to ask care-related questions, 81% want to make appointments, 68% would like to make prescription requests, 62% want to obtain laboratory results, 59% would complete medical forms, and 53% want to review and pay bills [15].

Wei developed a consultation system via the Internet to support patient-physician communications. Through the system, the patients are able to receive medical information or advice from their healthcare providers. It also helps the patients to find their related question in the healthcare knowledge base [16].

Shdefat proposed an e-consultation system which provides survey-based online diagnoses for patients, such as that produced by the physicians. The system also provides information about various diseases, presenting them in the format of online materials. Moreover, it provides an online forum where specialists, trainees, and patients can interact with each other using question-and-answer information sharing technique [13].

The EliteHealth e-consultation service provides patients with the ability to consult online with board-certified physicians in a variety of subspecialties. Registered patients are able to send secure e-mail messages to the physician and receive a response in 24 hours. They can also request a video consultation with the physician. Once the video request is sent, they will be contacted by one of the staff with the accepted time by the physician and the electronic consultation appointment will be set [17].

Another system that supports communication between patients and physicians is Consult a Doctor. The system offers a number of ways to access the physicians via phone or e-mail. The e-mail consultation has health manager which provides educational health and risk management support tools such as a chronic care program, prescription reminder, and symptom checker [18].

Currently, e-consultation is emerging as a valuable tool facilitating communication among healthcare providers. It was deemed a key feature of the successful implementation

of collaborative working models [19] and it brought about positive experiences to patients, physicians, and overall healthcare systems including improved continuity of care, access to specialists, and information transfer [20]. It has systematic structures, tools, and processes for information creation, transfer, receipt, and recognition by the requesting physician and the consultant to assist medical practices [21]. With more published studies, e-consultation has the potential to substantially improve care coordination while concurrently reducing healthcare costs associated with unnecessary referrals [4].

The conceptual definition of *referral* implies an actual transfer of responsibility for some aspect of the patient care to another physician. In contrast, a *consultation* involves seeking an opinion of a colleague about a particular aspect of the patient care. The referral workflow efficiency might be improved if e-consultations are effectively used. For instance, certain referral questions are addressed more efficiently through information exchanges or consultation between the referring physician and the consultant, which does not require a patient visit. Electronic medical records can facilitate these consultations through flexible and efficient e-consultation processes that minimize delays. A successful example of this practice is the established telemedicine modality known as *store-and-forward* in which the referring physician exchanges relevant patient data with the consultant and requests his or her opinion electronically [22].

San Francisco General Hospital created and implemented an electronic referral and consultation system dubbed eReferral. The system uses a web-based program embedded in the electronic medical record to facilitate a structured review process for new specialty clinics referrals. A free text field is also provided for pertinent clinical information and the reason for consultation. A designated clinician reviews the referrals. If the reviewer deems that the appointment is necessary and there is sufficient information for the specialist to make a clinical decision, the appointment is scheduled. However, when the clinical question is not clear or the problem can be handled in the primary care setting, an appointment is not scheduled. The reviewer can ask for clarification, guide further evaluation, or provide education as to how the referring physician can manage the issue. The reviewer and the referring physician can communicate via eReferral in an iterative fashion until both agree that the patient does not need the appointment or the appointment is scheduled [23].

A web-based consultation system was developed to enable consultation between a PCP and a nephrologist about a patient with chronic kidney disease. When the PCP enters the system directly from the file of the patient in the electronic medical record, essential patient data on medication and laboratory results are extracted and displayed in an orderly manner. The nephrologist, who is notified via e-mail or SMS that a consultation has arrived,

logs onto the website and based upon the patient information, advises the PCP how to treat the patient. The nephrologist can request additional information and defer management advice until this information is available. The PCP could adjust patient care or refer the patient according to the advice of the nephrologist [9].

The Champlain BASE: Building Access to Specialist through E-consultation tool was developed for e-scheduling applications among healthcare workers in Canada. The system allows the requesting physician to submit clinical question to the specialist. Supplementary patient data like test results can be included with the request to assist the specialist in making an informed recommendation. Depending on the request, the specialist has the options to provide answers to questions and avoid the need for a visit, to request additional information, or to recommend a formal referral. Both the specialist and the requesting physician receive e-mail notification at each stage of the process so they know the status of the e-consultation [8].

One limitation of Champlain BASE is the impersonal nature of communication. During its implementation, users remarked that the mode of selecting a specialist based on familiarity or an existing working relationship is not available in the system. Hence, lack of comfort stemmed partly from requesting physicians since they do not know the specialist with whom they were consulting [8]. In WebPCS, this limitation was addressed by allowing the requesting physicians themselves to choose the specialist with whom they want to consult.

Another interesting system is Doc2Doc that was developed by Dr. David Kendrick. It is a web-based system that was designed to simulate the lounge culture of physicians, where they gather and discuss patient cases together. It was successfully implemented in the Oklahoma Department of Corrections for use in fee-for-service settings. The said department agreed to pay specialists $50 for every completed e-consultation. Since implementation, Doc2Doc has reduced specialist visits and the transportation involved by approximately 50% and reduced costs. Although the study is not yet published, the Principal Investigator reports almost 100,000 e-consultations have taken place to date, and the system has now spread for use in Louisiana and Kentucky [24].

Mawell eConsultation is a web-based remote medical consultation solution that enables clinicians to rapidly consult with each other on medical image diagnosis. The solution is integrated with Picture Archiving and Communication System and Radiology Information System. Consultation requests can be sent to an organization, individual, or a consulting forum, where specialists can accept requests and respond to them. Mawell informs the specialist of a new consultation request via SMS or e-mail. The requestor is also

informed when the specialist has completed the consultation. It also allows the requestor to send feedback to the specialist regarding the received consultation. At any time, the specialist may switch himself to an off-duty status, during which he will receive no consultation requests. This procedure will prevent requests from going unnoticed and unprocessed in the consultation portal [25].

A study described the design of a prototype remote consultation system intended to provide the social, institutional, and infrastructural context for sustained growth of a globally-distributed Ghanaian medical community. The system resembles online social networking services and supports two types of conversational threads: case consultations and discussions. Physicians generally log on to one of the web servers to refer new cases to colleagues or groups, to review responses to their own cases, or to review cases that have been referred to them. They can also start a free-form discussion with any colleague or group. Groups correspond to particular specialties, institutions or hospitals, countries, and medical professional organizations [26].

In Germany, stroke patients are medically treated in so-called stroke units. Within the context of teleneuroconsultation, deliberation among physicians regarding diagnostic and therapeutic approach for the acute stroke patient can take place. In teleneuromedical settings, the stroke expert is connected by video and sound transmission, observing the examination of the patient, which is performed by the physician at the regional hospital. The radiological image data collected is electronically transmitted to a server platform that can be accessed by the expert. This information and other clinical impressions determine the remote diagnosis and related therapeutic instructions or recommendations to be given to the physician with data consultation sheet [27]. The architecture of teleconsultations in a teleneuromedical network is shown in Figure 1 and Figure 2.

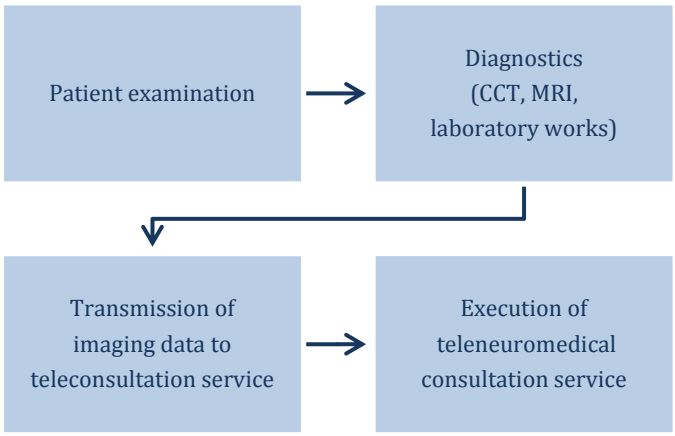


Figure 1 – Teleconsultations in a Teleneuromedical Network

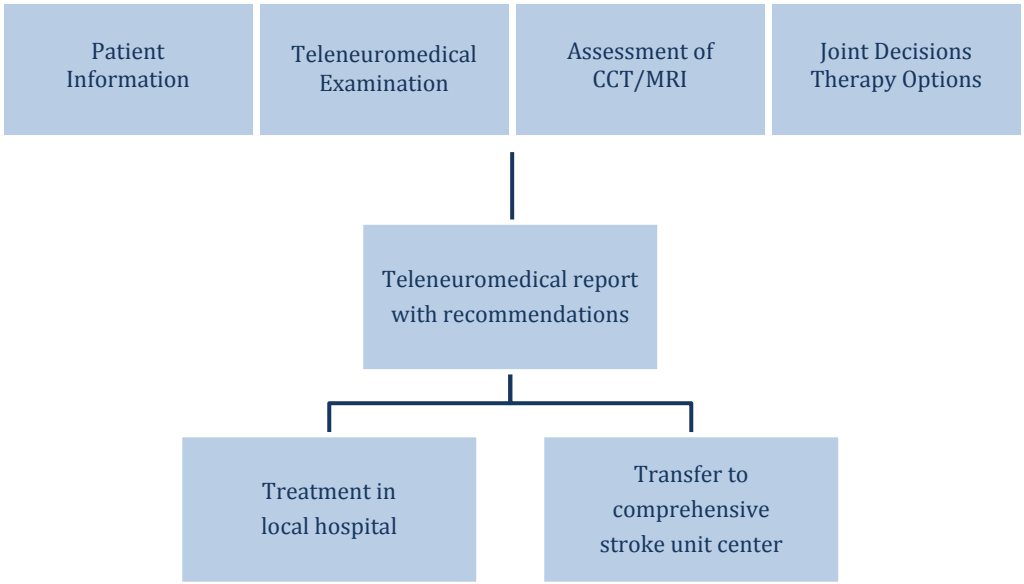


Figure 2 – Execution of Teleneuromedical Consultation Service

Los Angeles Care Health Plan developed a web-based software platform dubbed eConsult that allows PCPs and specialists to securely share health information and discuss patient care. The PCP provides relevant information about the condition of the patient like photographs and can ask questions about treatment options. The specialist answers the questions of the PCP and if necessary, seeks additional information. After one or more such exchanges, the specialist recommends whether a visit is needed [28]. The system workflow map is shown in Figure 3.

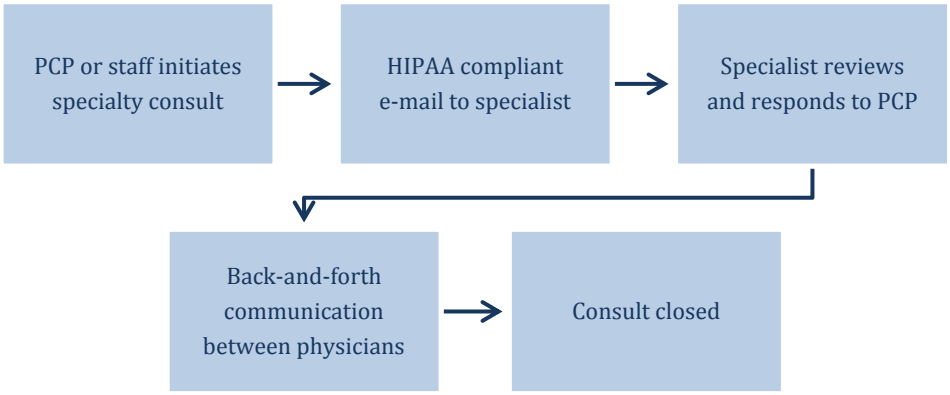


Figure 3 – eConsult System Workflow Map

III. THEORETICAL FRAMEWORK

3.1 Consultation

Current Procedural Terminology 2006 defined *consultation* as a type of service provided by a physician whose advice regarding evaluation and/or management of a specific problem is requested by another physician or other appropriate source [29].

Restrictions have not been set regarding individuals who may be considered an appropriate source, but some examples include a physician assistant, nurse practitioner, or insurance company. A consultation requested by a patient would not be reported as a consultative service.

A service is considered a consultation if the following criteria were satisfied:

a) A request for consultation must be documented by either the consultant or requesting practitioner in the medical record of the patient.
b) An opinion is rendered by the consultant. This opinion, along with any other service provided, is documented in the medical record of the patient.
c) A written report of the findings and recommendations of the consultant is communicated back to the requesting practitioner.

Consultation Process

The consultation generally involves a professional dialogue between physicians. In professional dialogue, physicians share their opinions and knowledge with the aim of improving their ability to provide the best care to their patients. Such dialogue may be part of the overall efforts of the physician to maintain current scientific and professional knowledge or may arise in response to the needs of a particular patient.

In a professional dialogue, the second physician is typically asked a question and he does not talk with or examine the patient. The first physician should not attribute an opinion to him. Sometimes, professional dialogue leads to a formal request for consultation. If, for instance, a physician is asked to provide an opinion regarding a patient and believes an examination of the patient or the medical record is necessary to answer the question appropriately, he should ask to see the patient for a formal consultation [2].

There are several levels of consultation. Their descriptions are as follows:

a) **Single-visit Consultation**

It involves examination of the patient record and performance of diagnostic tests or therapeutic procedures. The findings and recommendations of the consultant are recorded in the medical record of the patient or provided to the PCP in a written report, and a fee may be charged. The subsequent care of the patient continues to be provided by the referring physician. Examples of such consultations are confirming the findings of a pelvic examination and interpreting an electronic fetal monitoring tracing or imaging studies.

b) **Continuing Collaborative Care**

It describes a relationship in which the consultant provides ongoing care in conjunction with the referring physician. Thus, the consultant assumes at least partial responsibility of patient care. An example is a high-risk obstetric patient with a complication of pregnancy who is periodically assessed by the consultant, whereas the referring physician is responsible for the day-to-day management of the patient.

c) **Transfer of Primary Clinical Responsibility**

This may be appropriate for the management of problems outside the scope of the referring physician's education and experience or in cases in which the patient must be transferred to another facility. Examples are the transfer of patient care in preterm labor from a birth center to a consultant in a perinatal center or referral of a patient with ovarian cancer to a gynecologic oncologist. In many of these situations, patients will eventually return to the care of the referring physician when the problem for which the consultation was sought is resolved.

General Principles

The American Medical Association noted nine ethical principles of consultation. Three of these pertain to the referring physician: (1) the consultations are indicated on request in doubtful or difficult cases, or when they enhance the quality of medical care, (2) the consultations are primarily for the benefit of the patient, and (3) a case summary should be sent to the consultant unless a verbal description of the case has already been given.

The other six principles of consultation address the responsibilities and role of the consultant: (4) one physician should be in charge of the patient care, (5) the attending physician has overall responsibility for patient treatment, (6) the consultant should not assume primary care without the consent of the referring physician, (7) the consultation should be done punctually, (8) discussions during the consultation should be with the referring physician and patient only by prior consent of the referring physician, and (9) conflicts of opinion should be resolved by a second consultation [30].

Electronic Consultation

Electronic consultation addresses the challenges of the traditional consultation and improves the communication and coordination of care. It can occur through e-mail or other computer applications like shared electronic medical record systems that have messaging capabilities and web-based platforms. While the advice given by colleagues is informal, the communications are documented giving it an advantage over other consultation modalities such as the telephone or curbside conversations [4].

The potential of e-consultation varies according to specialty. It is likely to be most effective for physicians who provide cognitive advice rather than perform procedures. Specialties that rely heavily on laboratory tests such as endocrinology and nephrology have the most potential to become heavy users of e-consultation. However, it may offer strong potential for uses in specialties in short supply like geriatrics and in dermatology because photographs of skin lesions can be attached to consultation requests [4].

3.2 Collaborative Care

The Canadian Medical Association (CMA) supports collaboration among providers in the interest of better patient care. In the context of clinical practice, the CMA defines collaborative care as follows:

> *Collaborative care entails physicians and other providers using complementary skills, knowledge and competencies, and working together to provide care to a common group of patients based on trust, respect, and understanding of each other's skills and knowledge. This involves a mutually agreed upon division of roles and responsibilities that may vary according to the nature of the practice personalities and skill sets of the individuals. The relationship must be beneficial to the patient, the physician, and other providers.*

However, the CMA identified several barriers to the successful implementation of collaborative care including:

a) Ineffective support in information technology
b) Unresolved remuneration and pay issues
c) Lack of adequate program funding
d) Lack of time for proper implementation
e) Accountability for patient care is ambiguous
f) Unresolved medical liability issues
g) Resistance to change for professions, institutions, and governments

The College of Family Physicians of Canada acknowledges collaboration and notes that family physicians see themselves as important members of collaborative care teams and as clinical leaders of these teams. Similarly, the Royal College of Physicians and Surgeons of Canada identifies collaborator as one of the seven key roles of physicians. The two associated competencies of collaboration include consulting effectively with other physicians and contributing effectively to other interdisciplinary team activities. Examples of successful collaborative practice arrangements abound in areas are chronic disease management, community-based primary care medicine, geriatric care, palliative care, and rehabilitative medicine [31].

Collaborative Care Programs

The Department of Health (DOH) is the principal health agency in the Philippines. Its mandate is to develop plans, technical standards, and guidelines on health. Aside from the regulation of health goods and services, the DOH also provides special tertiary health care services and technical assistance to health providers and stakeholders [32].

The DOH has implemented a number of deployment programs to improve access and health service delivery. In these programs, selected health professionals are physically transferred in areas of need to complement the existing Human Resources for Health in health facilities and for efficient health service delivery. Currently, the department deploys nurses, physicians, and other health professionals under the following programs [33]:

a) Doctors to the Barrios Program (DTTB)

The program was created in 1993 to respond to the need for adequate health care services in 271 poor and far-flung municipalities which had no medical doctors. The program aims to attract competent, committed, and dedicated medical doctors to render health services in the underserved areas. The DOH developed a financial package to entice medical doctors to participate in the program. As a result, the DOH has deployed medical doctors to around two-thirds of the 271 municipalities.

b) Medical Pool Placement and Utilization Program (MPPUP)

Two types of medical doctors are being deployed under this program:

- Medical Officer 3 that includes new doctors or general practitioners who would replace the doctors from district to higher levels of hospitals who are on study leave
- Medical Specialists 2 who are deployed in tertiary health institutions or public medical centers to augment the required number of specialists

Although the MPPUP has been operating for 13 years, no systematic external assessment has been made about it. The perceived benefits and challenges faced by the participating medical doctors and the receiving hospitals should be recognized.

c) Registered Nurses for Health Enhancement and Local Service (RN HEALS)

Deployed nurses are assigned for 6 months in the community and another 6 months for hospital service. Since 2011, three batches of nurses have already been deployed. Although this is a fairly recent program, issues regarding the deployment processes and financial arrangements have been raised.

d) Rural Health Midwives Program

Midwives are assigned in Barangay Health Stations and Rural Health Units to provide quality health services with focus on maternal, newborn, and child health nutrition. These facilities can then provide Basic Emergency Obstetric and Newborn Care (BEmONC) or Comprehensive Emergency Obstetric and Newborn Care (CEmONC).

e) Rural Health Team Placement Program (RHTPP)

The program deploys health professionals to complement existing workforce of hospitals and Rural Health Units. This may include medical technologists, nutritionist-dieticians, dentists, pharmacists, and physical and occupational therapists.

IV. DESIGN AND IMPLEMENTATION

4.1 Context Diagram

The system will have three access levels such as the unregistered user, physician, and system administrator. The physician can be subdivided into requesting physician and consultant. The context diagram is shown in Figure 4.

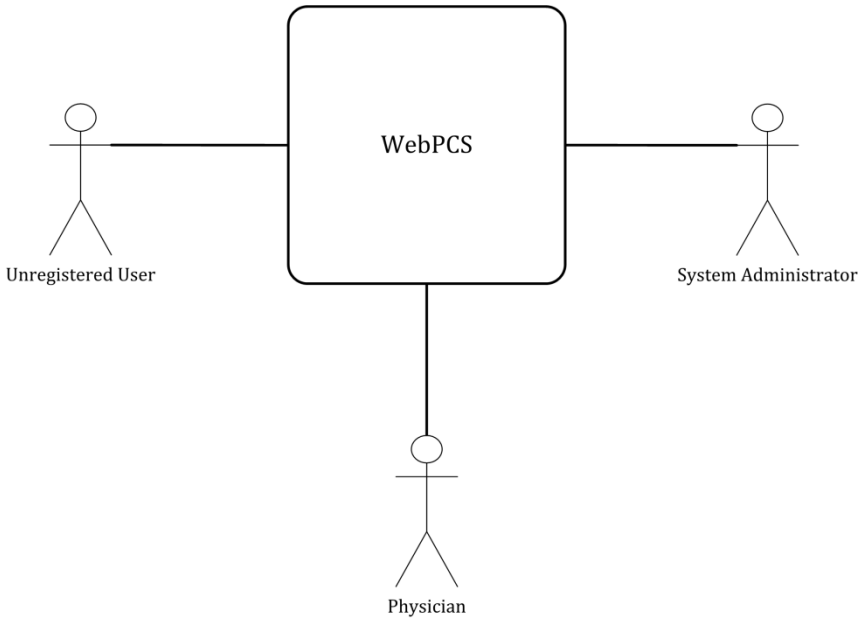


Figure 4 – Context Diagram, WebPCS

4.2 Use-Case Diagram

An unregistered user can request for an account in the system. Upon its approval, he will become a registered user and has an extended capability in the system.

A physician can initiate a consultation with a colleague and book an appointment. The requesting physician can provide clinical case information and upload pertinent files that will support the consultation. A consultation report can also be generated. Generally, the physicians can manage their own calendars, participate in forums, and view the online references available in the system.

The system administrator manages the entire system. He can update user accounts. He can generate reports of the total consultation tally. He can also update online references in the system.

The top level use-case diagram for the system is shown in Figure 5.

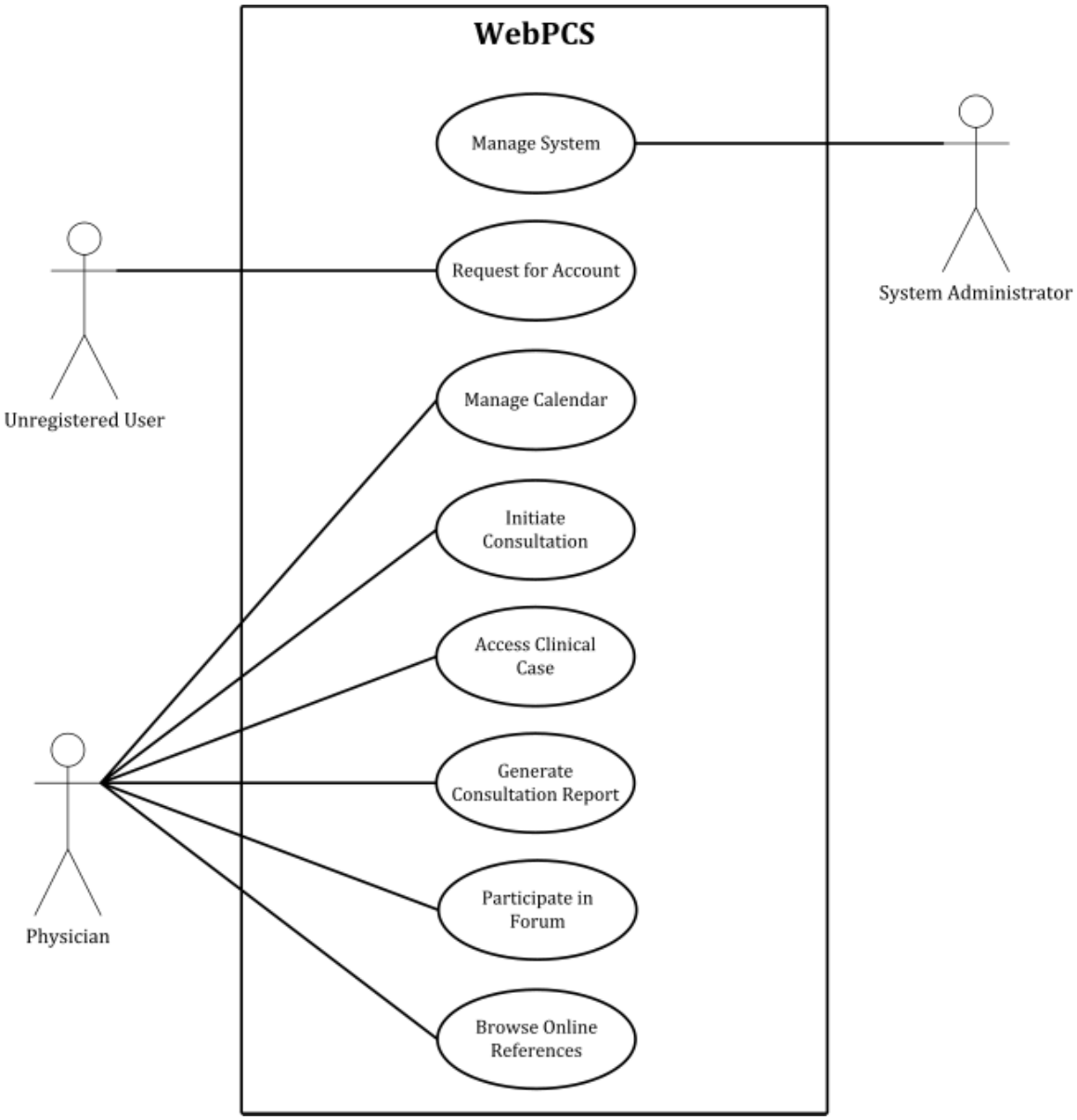


Figure 5 – Top Level Use-Case Diagram, WebPCS

4.3 Activity Diagram

a) Request for Account

An unregistered user can request for a user account. The activity diagram is shown in Figure 6.

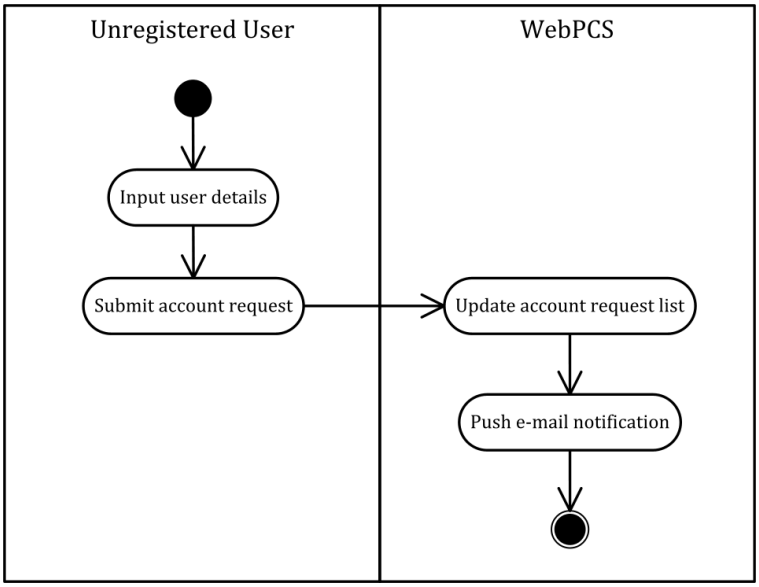


Figure 6 – Request for Account Activity Diagram, WebPCS

b) View Online References

A physician can view the list of online references available in the system. The activity diagram is shown in Figure 7.

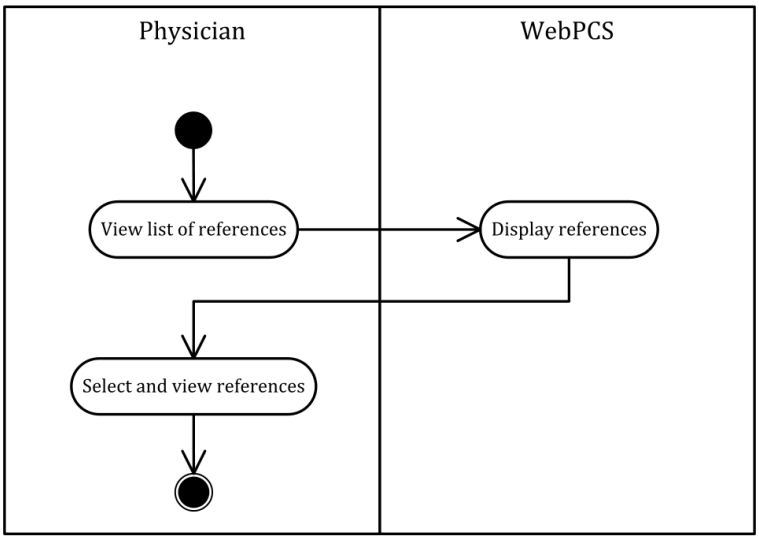


Figure 7 – View Online References Activity Diagram, WebPCS

c) Manage Calendar

A physician can view, edit, and cancel appointments in his calendar. The use-case diagram is shown in Figure 8.

The activity diagrams for view, edit, and cancel appointment are shown in Figures 9, 10, and 11, respectively.

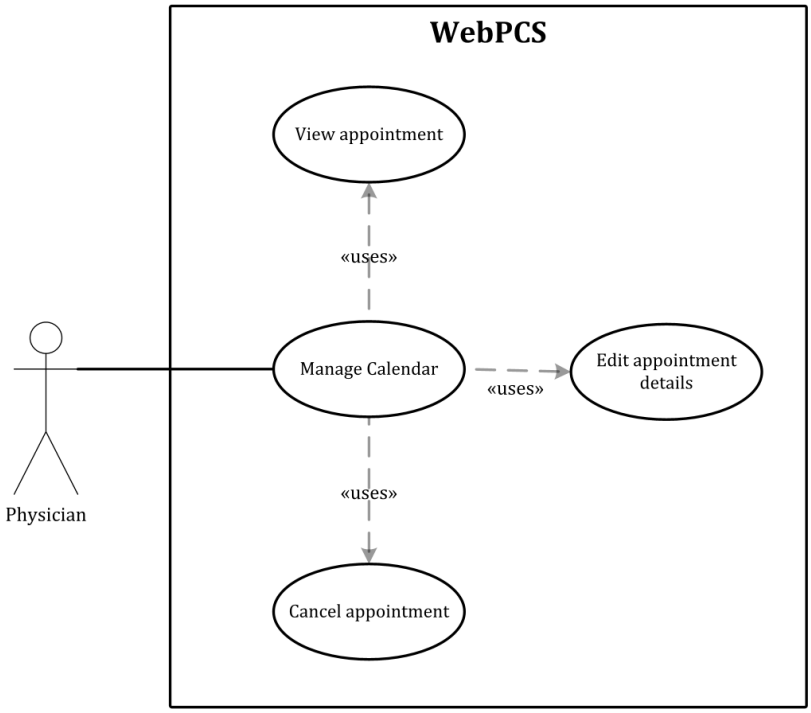


Figure 8 – Manage Calendar Use-Case Diagram, WebPCS

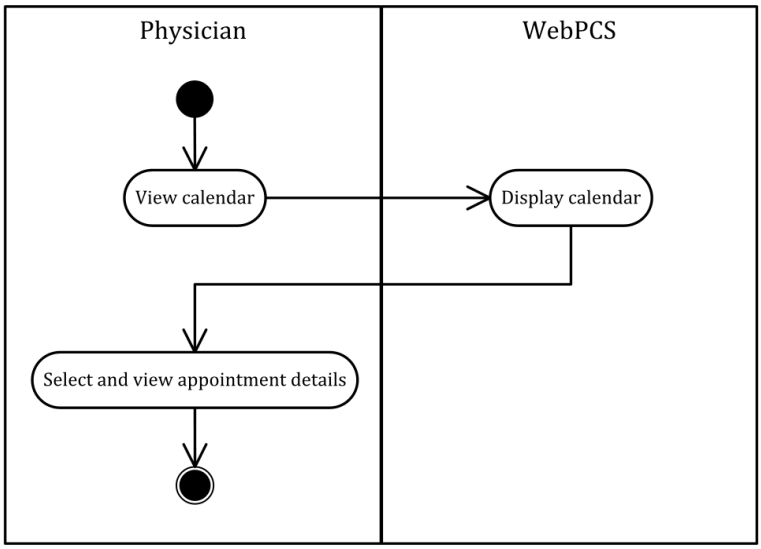


Figure 9 – View Appointment Activity Diagram, WebPCS

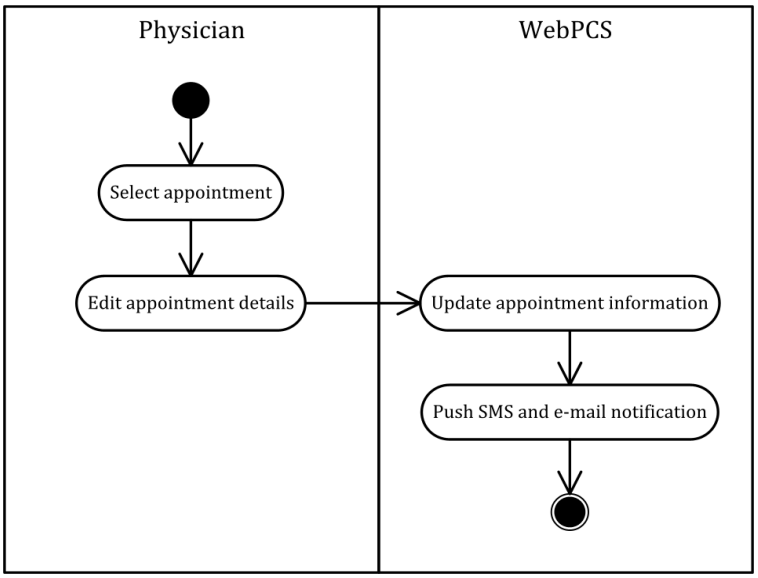


Figure 10 – Edit Appointment Details Activity Diagram, WebPCS

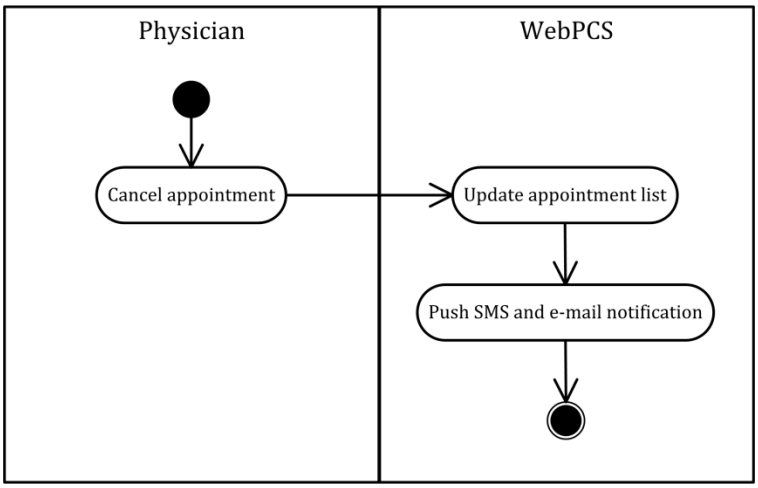


Figure 11 – Cancel Appointment Activity Diagram, WebPCS

d) Book an Appointment

A physician can book an appointment with his colleagues. He can submit the clinical case information along with supporting files. The activity diagram is shown in Figure 12.

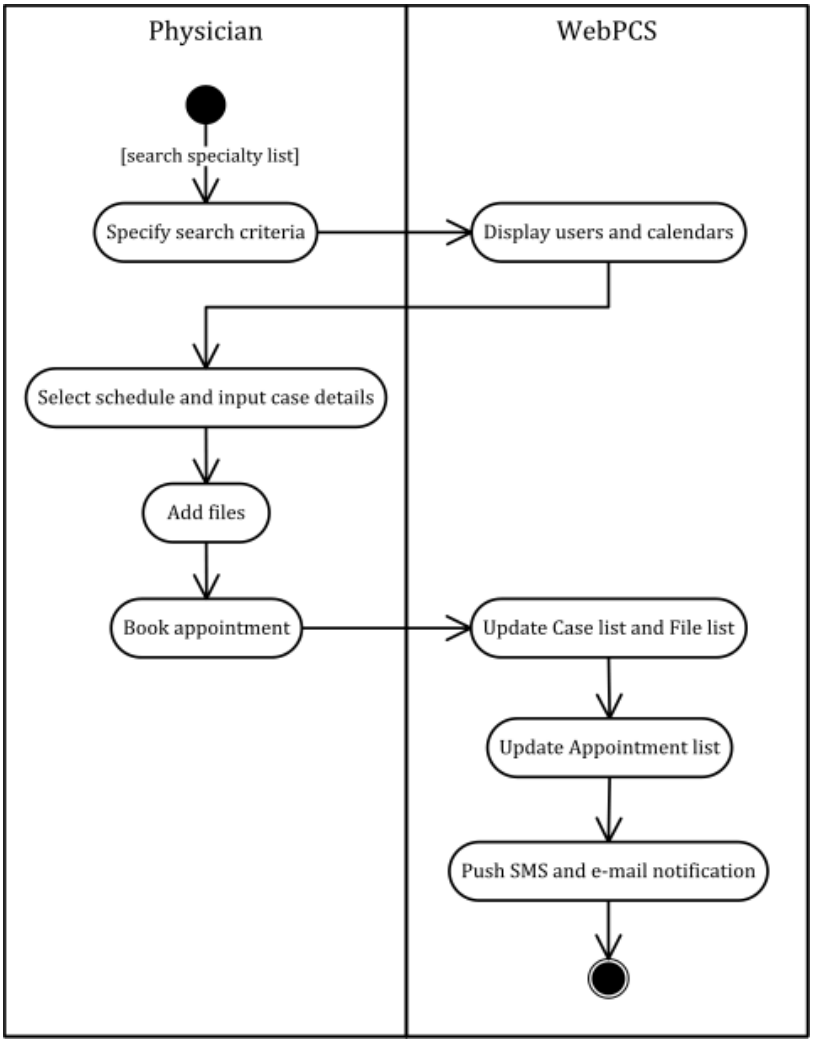


Figure 12 – Book an Appointment Activity Diagram, WebPCS

e) Access Clinical Case

A physician can view the clinical cases he consulted to another physician and the clinical cases consulted to him. He can update their information. He can also add more files and download the files attached to it. The use-case diagram is shown in Figure 13.

The activity diagrams for view clinical case, update clinical case information, and download file are shown in Figures 14, 15, 16, and 17 respectively.

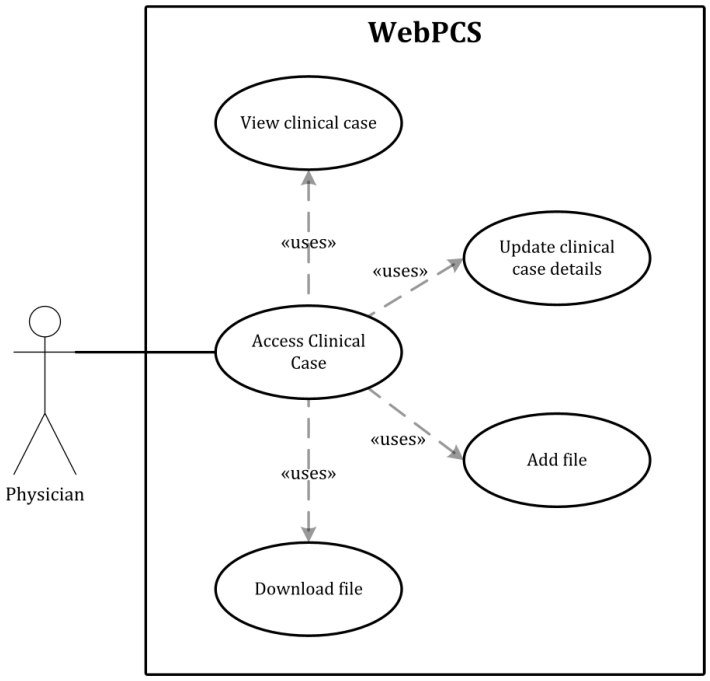


Figure 13 – Access Clinical Case Use-Case Diagram, WebPCS

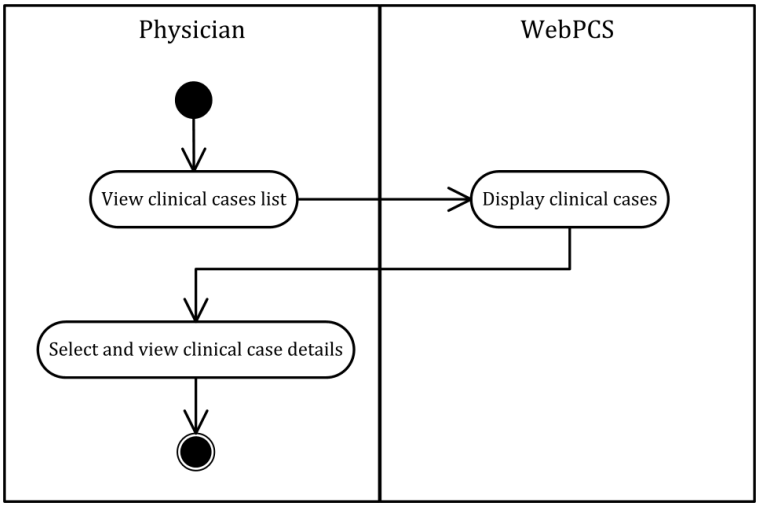


Figure 14 – View Clinical Case Activity Diagram, WebPCS

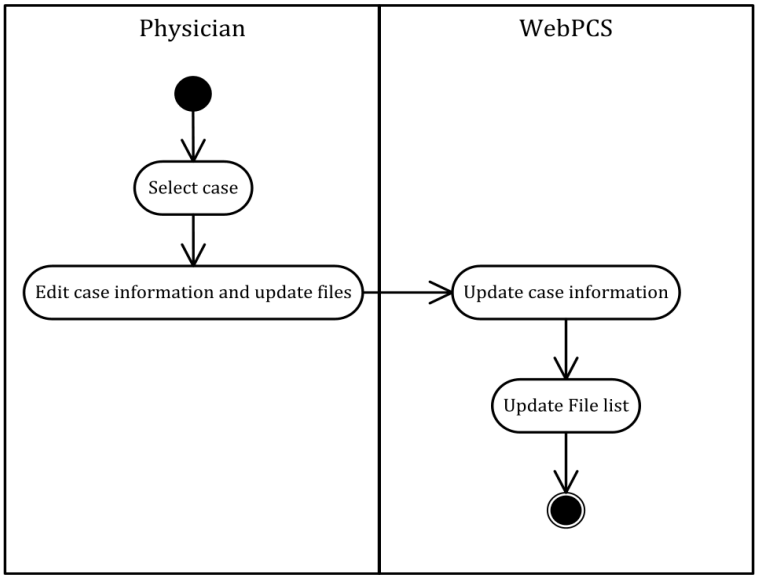


Figure 15 – Edit Case Details Activity Diagram, WebPCS

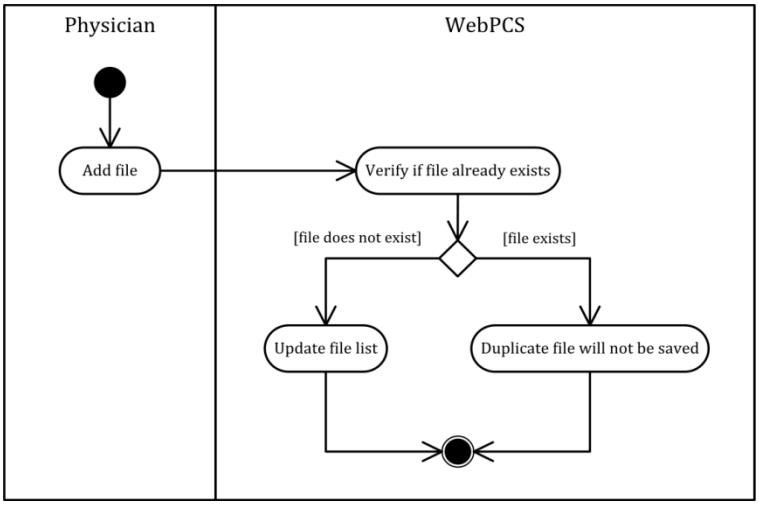


Figure 16 – Add File Activity Diagram, WebPCS

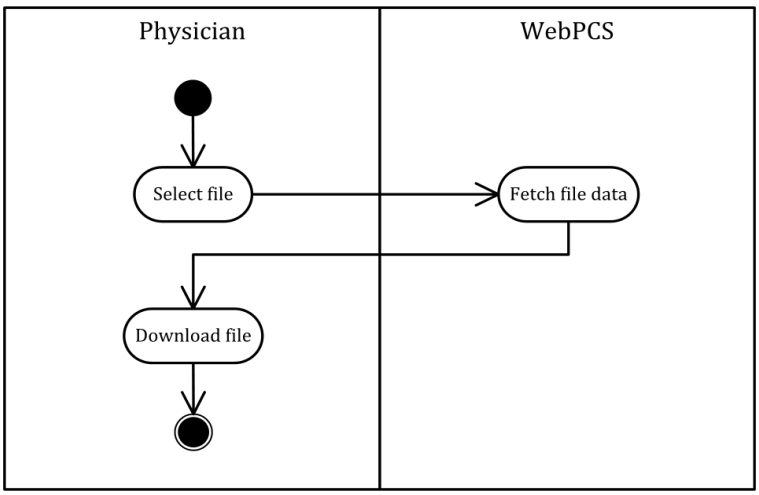


Figure 17 – Download File Activity Diagram, WebPCS

f) Participate in Consultation

A physician can post replies to consultation thread. The activity diagram is shown in Figure 18.

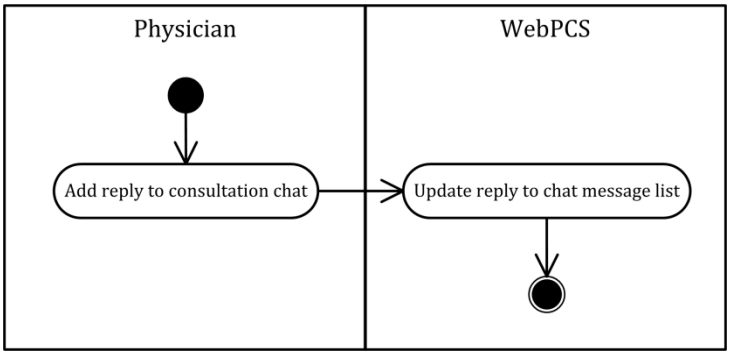


Figure 18 – Participate in Consultation Activity Diagram, WebPCS

g) Participate in Forum

A physician can post a new discussion and delete his own discussion. He can also view, post a reply, subscribe, and unsubscribe to another discussion. The use-case diagram is shown in Figure 19.

The activity diagrams for the aforesaid cases are shown in Figure 20, 21, 22, 23, and 24, respectively.

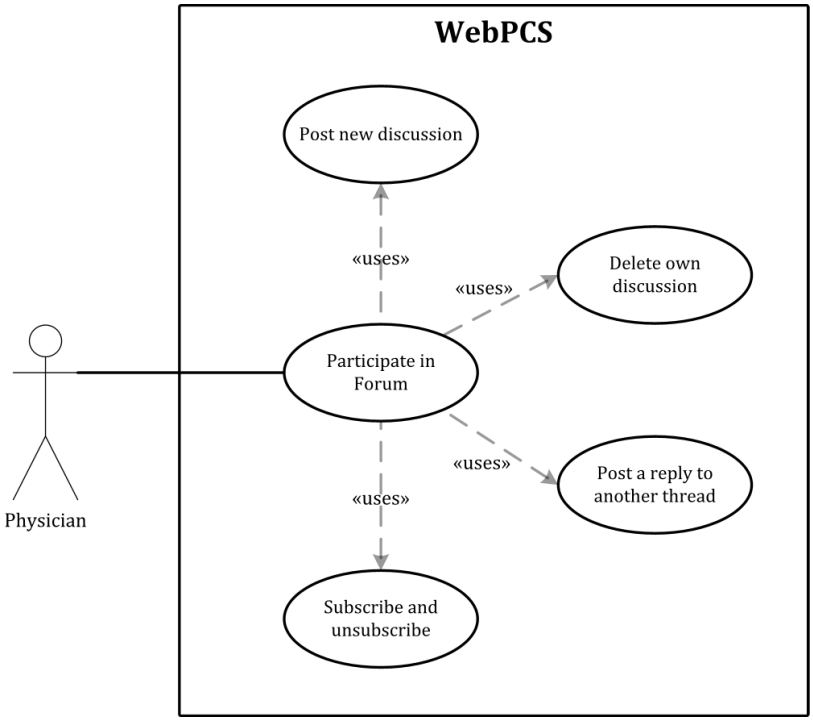


Figure 19 – Participate in Forum Use-Case Diagram, WebPCS

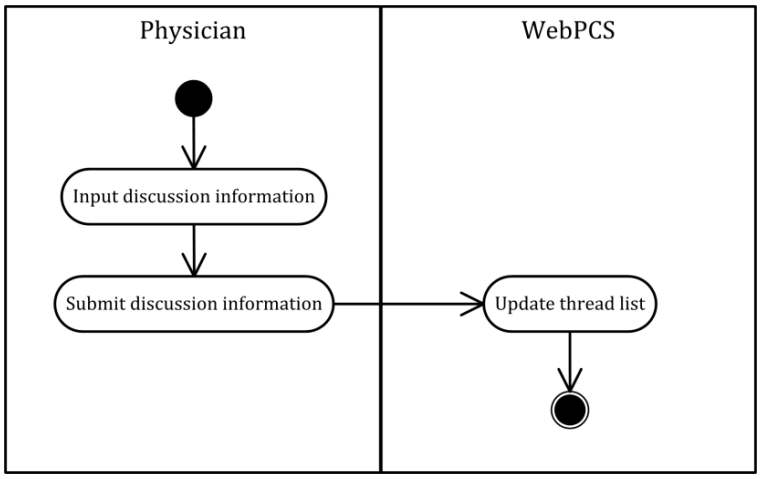


Figure 20 – Post New Discussion Activity Diagram, WebPCS

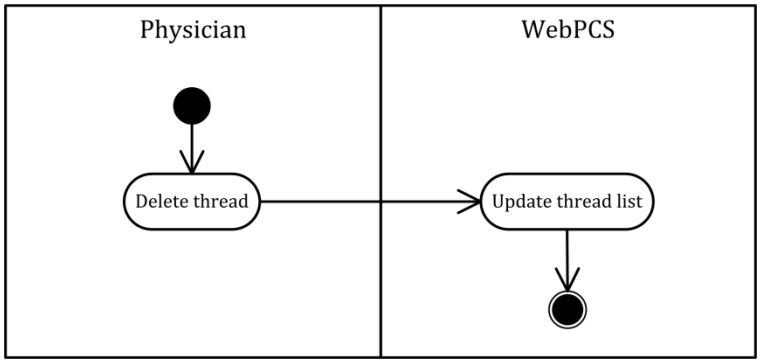


Figure 21 – Delete Discussion Activity Diagram, WebPCS

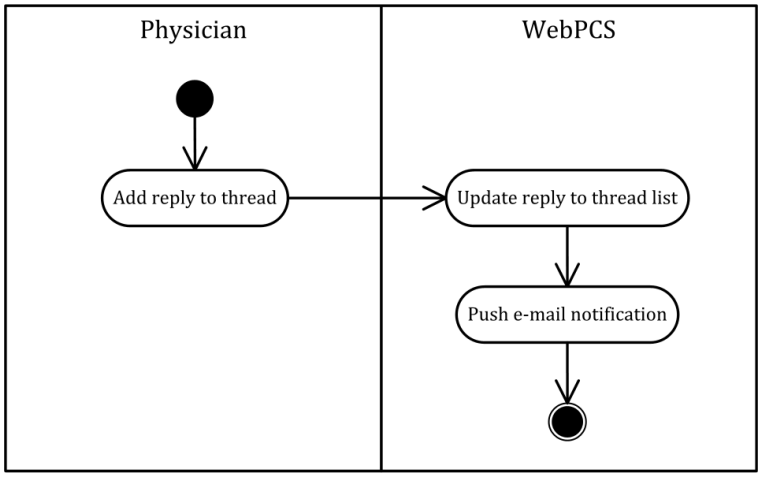


Figure 22 – Post Reply to Thread Activity Diagram, WebPCS

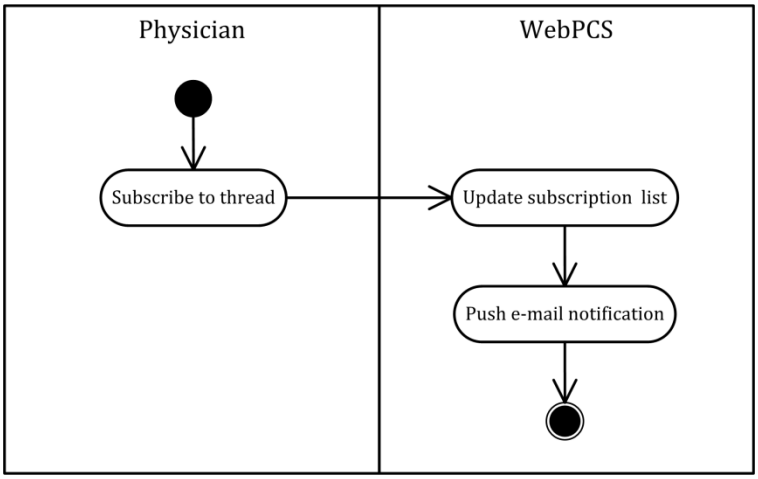


Figure 23 – Subscribe to Thread Activity Diagram, WebPCS

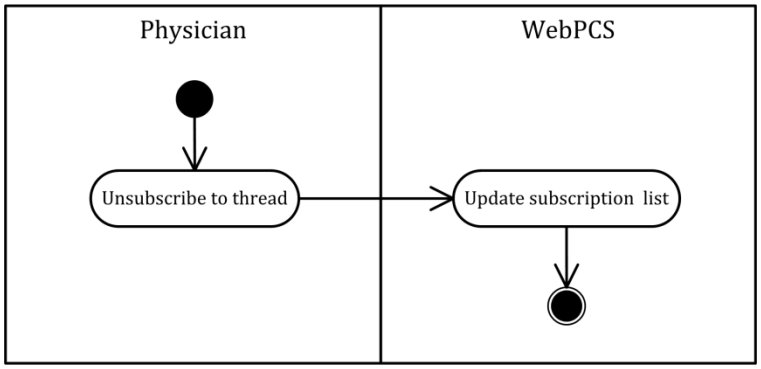


Figure 24 – Unsubscribe to Thread Activity Diagram, WebPCS

h) Generate Consultation Report

A physician can generate a report of all consultations he performed using the system. The activity diagram is shown in Figure 25.

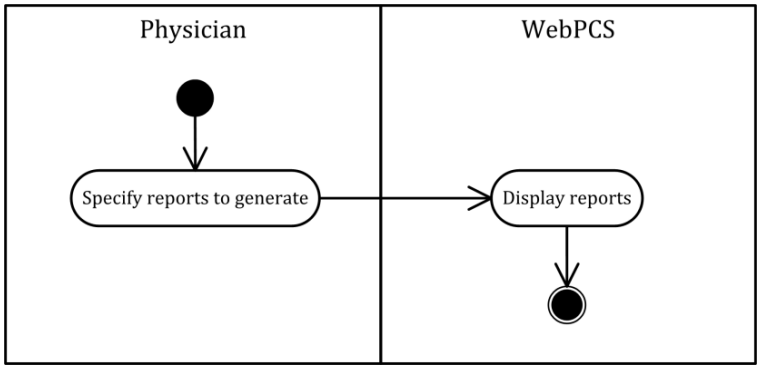


Figure 25 – Generate Consultation Report Activity Diagram, WebPCS

i) Manage System

The system administrator manages user accounts and online references. He can also generate reports. The use-case diagram is shown in Figure 26.

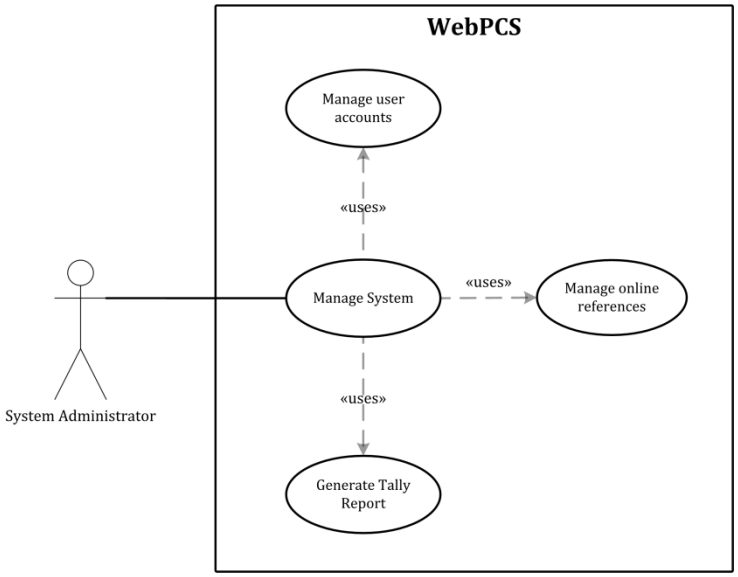


Figure 26 – Manage System Use-Case Diagram, WebPCS

Manage User Accounts

The system administrator can approve or decline account requests. He can also add, edit, view, and delete existing user accounts. The activity diagrams for the aforesaid cases are shown in Figures 27, 28, 29, 30, and 31, respectively.

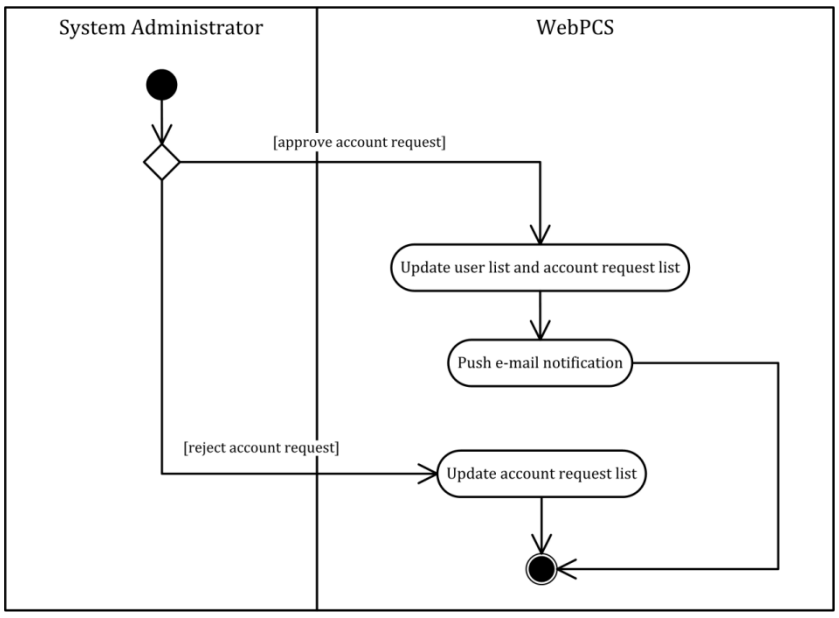


Figure 27 – Approve or Decline Account Request Activity Diagram, WebPCS

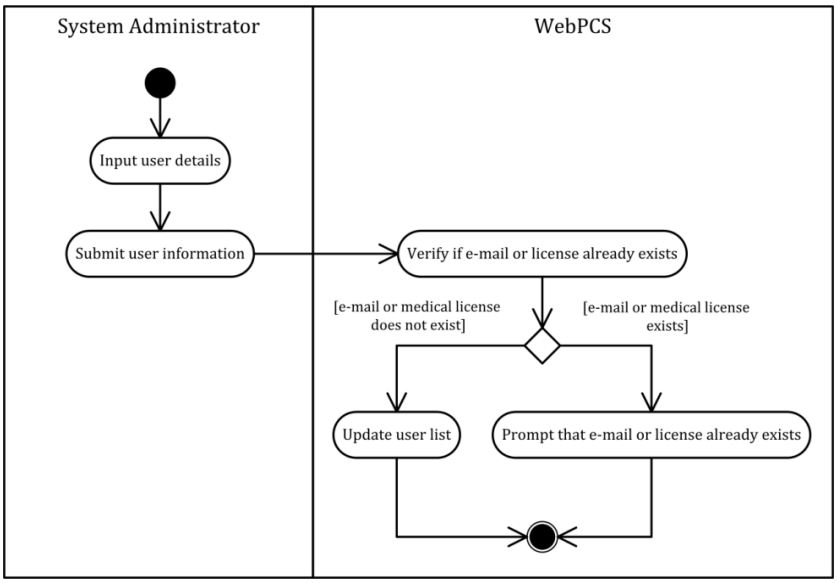


Figure 28 - Add Account Activity Diagram, WebPCS

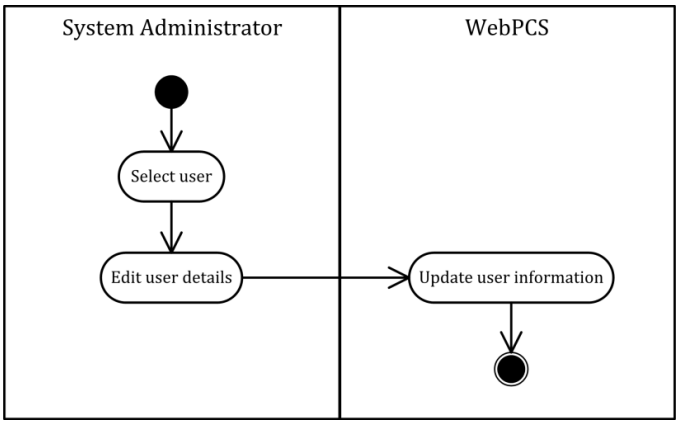


Figure 29 – Edit Account Details Activity Diagram, WebPCS

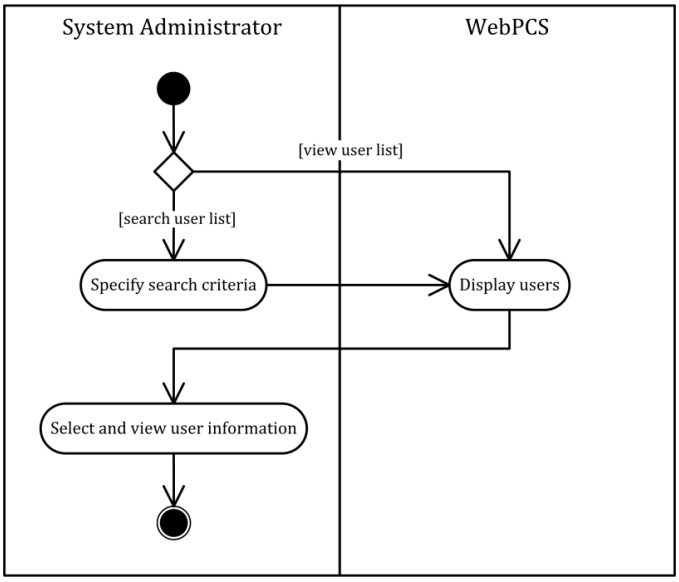


Figure 30 – View Account Activity Diagram, WebPCS

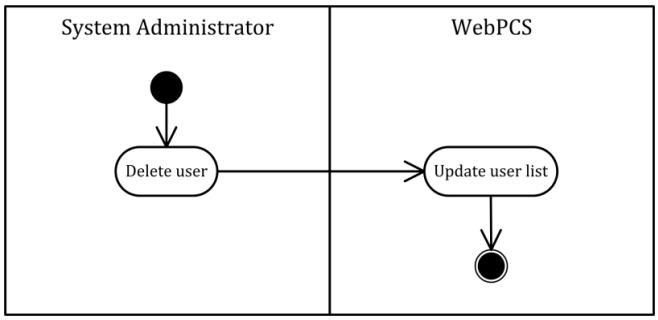


Figure 31 – Delete Account Activity Diagram, WebPCS

Manage Online References

The system administrator can add, edit, and delete references. The activity diagrams for the aforesaid cases are shown in Figures 32, 33, and 34, respectively.

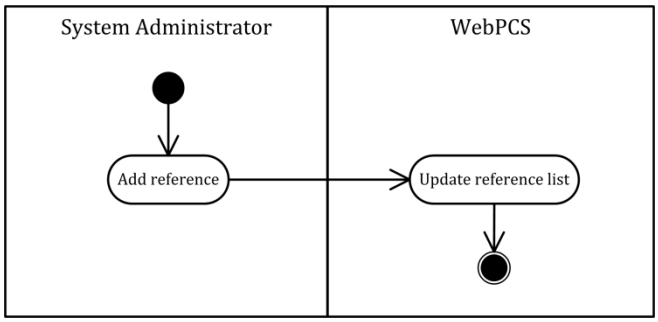


Figure 32 – Add References Activity Diagram, WebPCS

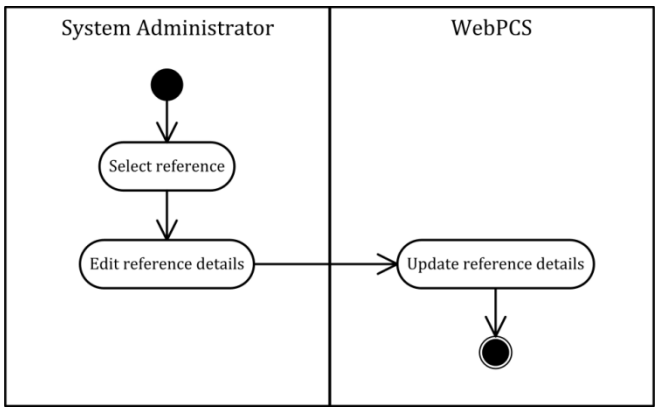


Figure 33 – Edit Reference Details Activity Diagram, WebPCS

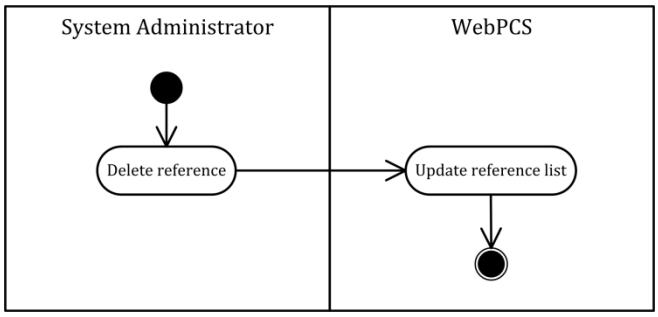


Figure 34 – Delete References Activity Diagram, WebPCS

Generate Consultation Tally Report

The system administrator can generate reports of consultation count. The activity diagram is shown in Figure 35.

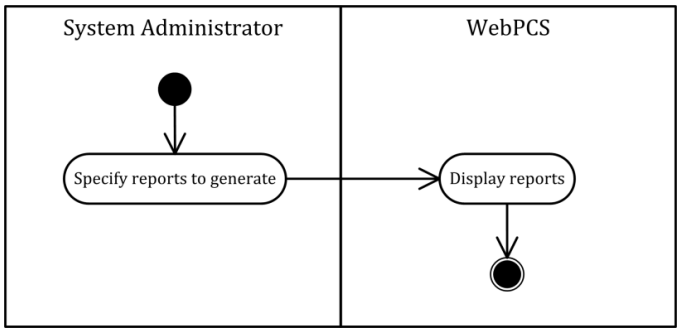


Figure 35 – Generate Consultation Tally Report Activity Diagram, WebPCS

4.4 Entity Relationship Diagram

The entity relationship diagram for the system is shown in Figure 36.

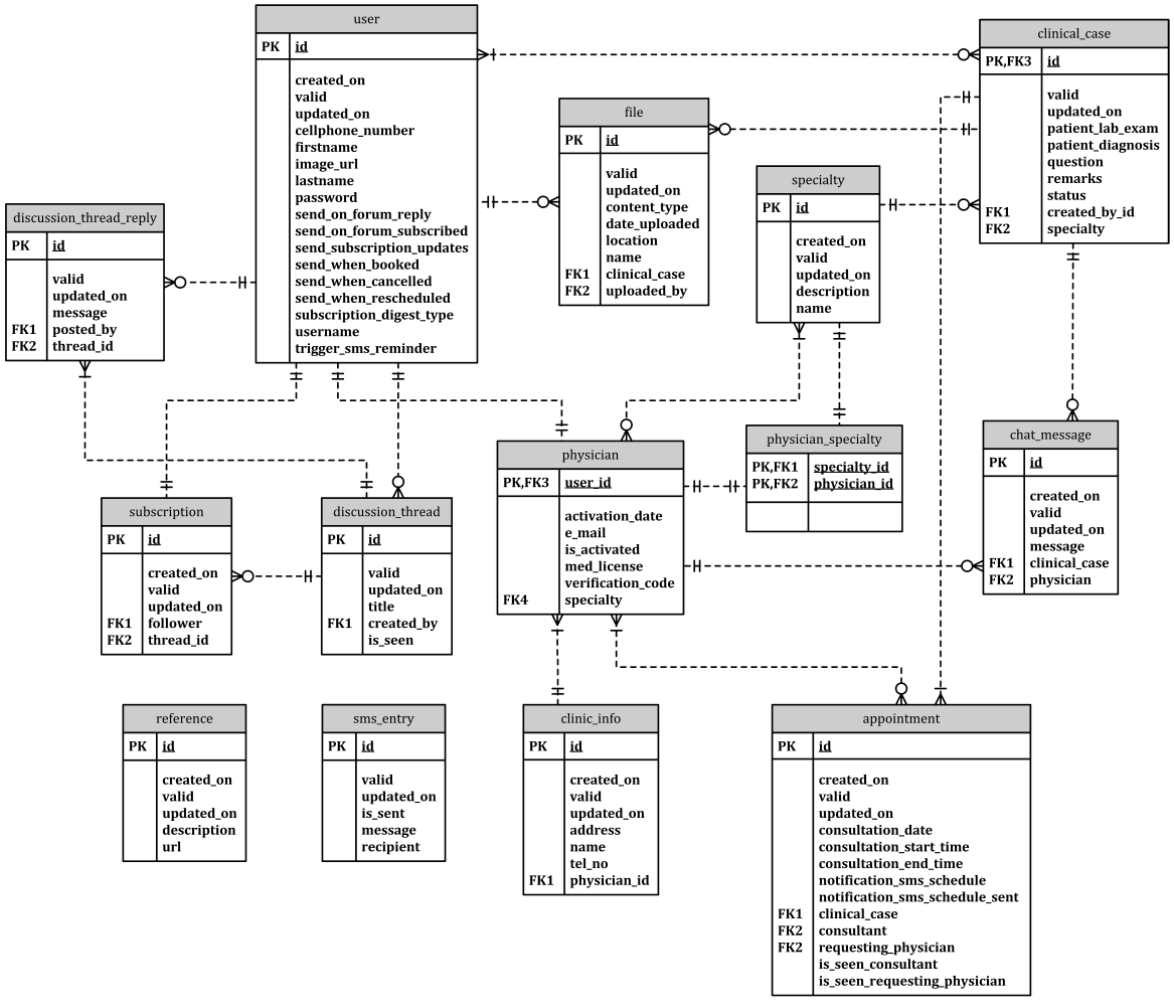


Figure 36 – Entity Relationship Diagram, WebPCS

4.1 Data Dictionary

Listed below are the database tables. Primary keys are underlined.

- **user**

This table contains the basic information and account details of users.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of user
created_on	datetime	date and time of registration
valid	bit(1)	denotes if account is active (1) or not (2)
updated_on	datetime	date and time when user last logged in
cellphone_number	varchar(255)	mobile number of user
firstname	varchar(255)	first name of user
image_url	varchar(255)	path where photo of user was saved
lastname	varchar(255)	last name of user
password	varchar(255)	password of user in the system
username	varchar(255)	e-mail address of user
send_on_forum_reply	bit(1)	denotes if account is set to receive a forum update (1) or not (0)
send_on_forum_subscribed	bit(1)	denotes if account is set to receive a forum subscriber update (1) or not (0)
send_subscription_updates	bit(1)	denotes if account is set to receive a forum subscription update (1) or not (0)
send_when_booked	bit(1)	denotes if account is set to receive a notification for a booked appointment (1) or not (0)
send_when_cancelled	bit(1)	denotes if account is set to receive a notification for a canceled appointment (1) or not (0)
send_when_rescheduled	bit(1)	denotes if account is set to receive a notification for a rescheduled appointment (1) or not (0)
trigger_sms_reminder	bit(1)	denotes if account is set to receive an SMS reminder for an appointment (1) or not (0)

Table 1 – Data Dictionary for *user* Table

- **physician**

  This table contains the professional information of users.

FIELD	TYPE	DESCRIPTION
user_id	bigint(20)	unique identifier of physician
email	varchar(255)	e-mail address of user
med_license	bigint(20)	medical license number of user
specialty	bigint(20)	medical specialty of user
activation_date	datetime	date and time when account was activated
is_activated	bit(1)	denotes if account is activated (1) or not (0)
verification_code	varchar(255)	account verification code

Table 2 – Data Dictionary for *physician* Table

- **clinic_info**

  This table contains the clinic information of users.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of clinic information
created_on	datetime	date and time when information was added
updated_on	datetime	date and time when information was updated
address	varchar(255)	address of clinic
name	varchar(255)	name of clinic
tel_no	varchar(255)	landline of clinic
physician_id	bigint(20)	user who owns the clinic information

Table 3 – Data Dictionary for *clinic_info* Table

- **specialty**

  This table contains the list of medical specialties.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of specialty
created_on	datetime	date and time when specialty was added
updated_on	datetime	date and time when specialty was updated
description	text	description of specialty
name	varchar(255)	name of specialty

Table 4 – Data Dictionary for *specialty* Table

- **clinical_case**

  This table contains the list of clinical cases.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of case
created_on	datetime	date and time when case was created
updated_on	datetime	date and time when case was updated
patient_diagnosis	text	findings
patient_lab_exam	text	laboratory exams performed
question	text	clinical question of the user
remarks	text	additional comment
specialty	bigint(20)	medical specialty classification of case
status	varchar(255)	status of the case
created_by_id	bigint(20)	user who created the case

Table 5 – Data Dictionary for *clinical_case* Table

- **appointment**

  This table contains the list of appointments.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of appointment
created_on	datetime	date and time when appointment was booked
valid	bit(1)	denotes if appointment is active (1) or not (0)
updated_on	datetime	date and time when appointment was updated
consultation_date	date	date of consultation
consultation_start_date	datetime	date and time when consultation started
consultation_end_date	datetime	date and time when consultation ended
notification_sms_schedule	datetime	date and time when reminder was sent
notification_sms_schedule_sent	bit(1)	denotes if reminder was sent (1) or not (0)
clinical_case	bigint(20)	case in which appointment was related to
requesting_physician	bigint(20)	physician who booked the appointment
consultant	bigint(20)	consultant

Table 6 – Data Dictionary for *appointment* Table

- **physician_specialty**

  This table contains the list of physicians and their corresponding specialties.

FIELD	TYPE	DESCRIPTION
physician_id	bigint(20)	unique identifier of physician
specialty_id	bigint(20)	unique identifier of specialty

Table 7 – Data Dictionary for *physician_specialty* Table

- **sms_entry**

  This table contains the actual messages that system sends to users.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of SMS entry
created_on	datetime	date and time when dispatch was activated
updated_on	datetime	date and time when message was sent
is_sent	bit(1)	denotes if message was sent (1) or not (0)
message	timestamp	actual message sent to user
recipient	varchar(255)	mobile number if recipient

Table 8 – Data Dictionary for *sms_entry* Table

- **file**

  This table contains the list of files attached to the clinical case.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of file
content_type	text	content type of file
date_uploaded	datetime	date and time when file was uploaded
location	text	path where file was saved
name	text	file name
clinical_case	bigint(20)	case in which file was attached
uploaded_by	bigint(20)	user who uploaded the file

Table 9 – Data Dictionary for *file* Table

- **chat_message**

  This table contains the actual messages exchanged by users during consultation.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of chat message
created_on	datetime	date and time when message was sent
message	text	actual message sent by user
clinical_case	bigint(20)	case in which chat was related to
physician	bigint(20)	user who sent the message

Table 10 – Data Dictionary for *chat_message* Table

- **discussion_thread**

  This table contains the list of discussions.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of discussion
created_on	datetime	date and time when discussion was posted
valid	bit(1)	denotes if discussion is active (1) or not (0)
updated_on	datetime	date and time when discussion was updated
title	varchar(255)	title of discussion
created_by	bigint(20)	user who created the discussion

Table 11 – Data Dictionary for *discussion_thread* Table

- **discussion_thread_reply**

  This table contains the list of replies to the thread.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of reply
created_on	datetime	date and time when reply was posted
message	text	reply to thread

posted_by	bigint(20)	user who posted the reply
thread_id	bigint(20)	thread in which the reply was posted

Table 12 – Data Dictionary for *discussion_thread_reply* Table

- **subscription**

  This table contains the list of subscribers of a discussion.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of subscription
created_on	datetime	date and time of subscription
follower	bigint(20)	user who subscribed to the discussion
thread_id	bigint(20)	thread in which user subscribed to

Table 13 – Data Dictionary for *subscription* Table

- **reference**

  This table contains the list of online references in the system.

FIELD	TYPE	DESCRIPTION
id	bigint(20)	unique identifier of reference
created_on	datetime	date and time when reference was created
valid	bit(1)	denotes if reference is active (1) or not (0)
updated_on	datetime	date and time when reference was updated
description	text	description of the reference
url	varchar(255)	URL of the reference

Table 14 – Data Dictionary for *reference* Table

V. ARCHITECTURE

5.1 System Architecture

a) Spring

WebPCS is built using the Spring Web MVC framework. The Spring MVC framework is an open source Java platform which affords a well-designed MVC architecture and ready components that are used to develop flexible and loosely coupled web applications. The MVC pattern separates the Input Logic, Business Logic, and UI Logic of the application while giving loose coupling between these elements.

Spring Web MVC framework is designed around a *DispatcherServlet* that handles all the HTTP requests and responses. The request processing workflow of the Spring Web MVC *DispatcherServlet* is illustrated in Figure 37.

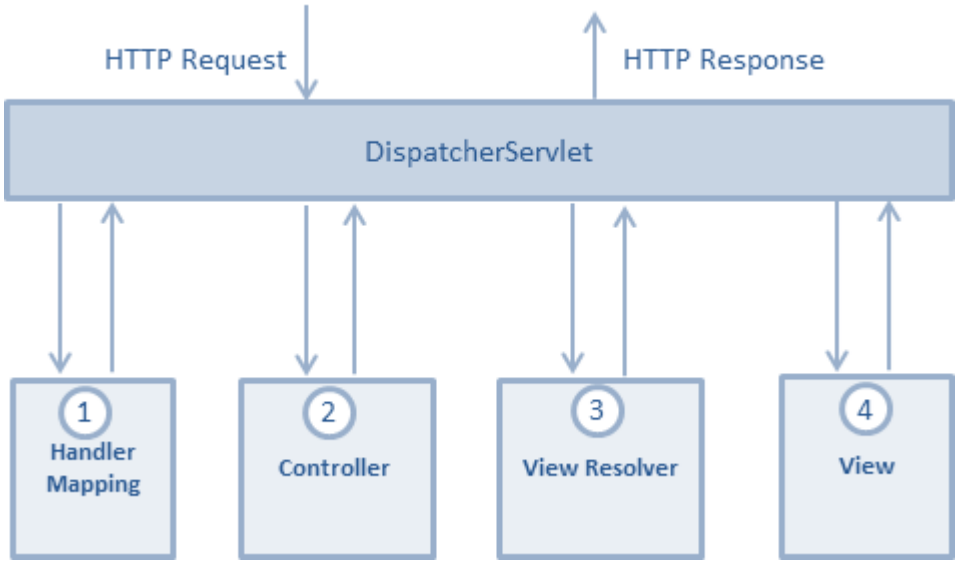


Figure 37 – Spring Web MVC *DispatcherServlet*

After receiving an HTTP request, the *DispatcherServlet* communicates with the *HandlerMapping* to call the Controller. The Controller takes the request and calls the appropriate service methods based on GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet* which will take help from *ViewResolver* to choose the defined view for the request. The *DispatcherServlet* then passes the model data to the view which is finally rendered on the browser.

b) Hibernate

When working on object-oriented systems, several mismatch problems can occur between the object model and the relational database since the latter represent data in a tabular format whereas object-oriented languages like Java represent it as an interconnected graph of objects. A programming technique called Object-Relational Mapping is the solution to handle the mismatches.

Hibernate is one persistent framework and ORM option in Java. WebPCS used it as a system model as it supports the major RDBMS and technologies like Eclipse. Hibernate maps Java classes to database tables and from Java data types to SQL data types using XML files. It uses database and configuration data to provide persistence services and persistent objects to the application

c) Bootstrap

For the front-end framework, WebPCS used the Bootstrap. It is an open source toolkit for developing web applications. It contains fundamental HTML elements styled and enhanced with extensible classes and global CSS settings, as well as optional JavaScript extensions to provide components like alerts, forms, buttons, and navigation.

d) dhtmlxScheduler

The appointment calendar used in WebPCS is the dhtmlxScheduler. It is compatible with Bootstrap and has an extensive JavaScript API which gives user a full control over the calendar. It also offers dhtmlxConnector which simplifies data communication between the calendar interface and the database.

5.2 Technical Architecture

a) Server

Hardware Requirements

- 2.4 MHz single core processor
- 512 MB RAM
- 1 GB of free disk storage

Software Requirements

- Eclipse Java EE IDE with Maven integration (Kepler Service Release 2)

- MySQL Workbench 6.3 CE
- JRE (Java Runtime Environment) 7
- Windows 7

b) Client

WebPCS requires the client to have a browser installed in his machine. It works on the following browsers:

- Mozilla Firefox (38.0.5)
- Google Chrome (43.0.2357.124)

VI. RESULTS

6.1 System Screenshots

WebPCS is an application that can be used by physicians primarily for consultation. It has two types of users – physician and system administrator. The WebPCS homepage is shown in Figure 38.

Figure 38 – WebPCS Homepage

Physicians who want to provide consultation services and gain access to the system can complete the registration form shown in Figure 39. The success of registration is shown in Figure 40.

Figure 39 – WebPCS Registration Page

Figure 40 – WebPCS Registration Form Submitted

Physicians who failed to recall their password can request for a new one via Forgot Password page shown in Figure 41. The system will then send them a temporary password which they can use to log in. The success of password reset request is shown in Figure 42.

Figure 41 – WebPCS Forgot Password Page

Figure 42 – WebPCS Forgot Password form Submitted

Figure 43 shows the Dashboard for physician access level. The physician dashboard keeps track of his Latest Appointments and shows his Consultation Trends presented in bar chart. The Dashboard also has sections for Latest Discussions and View Physicians.

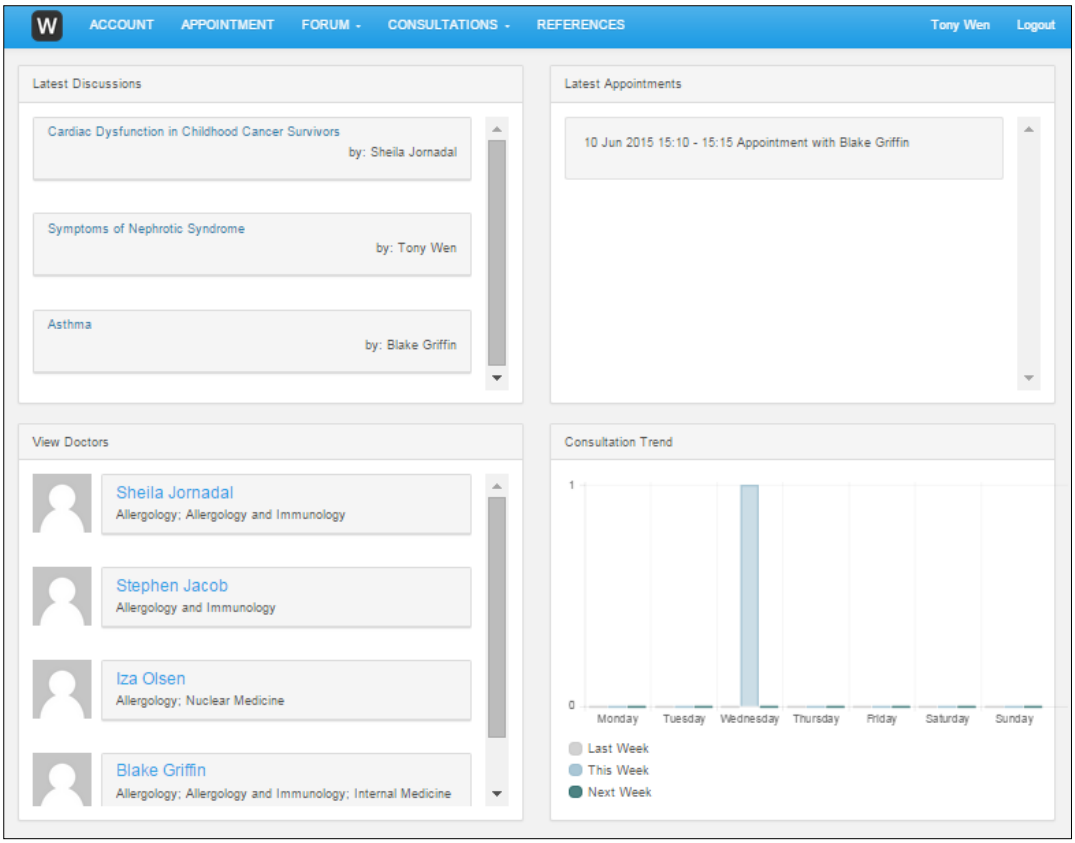


Figure 43 – WebPCS Dashboard for Physician Access Level

The list of all physicians in the system is shown in Figure 44. Clicking on each name leads to the corresponding profile of the physician as shown in Figure 45.

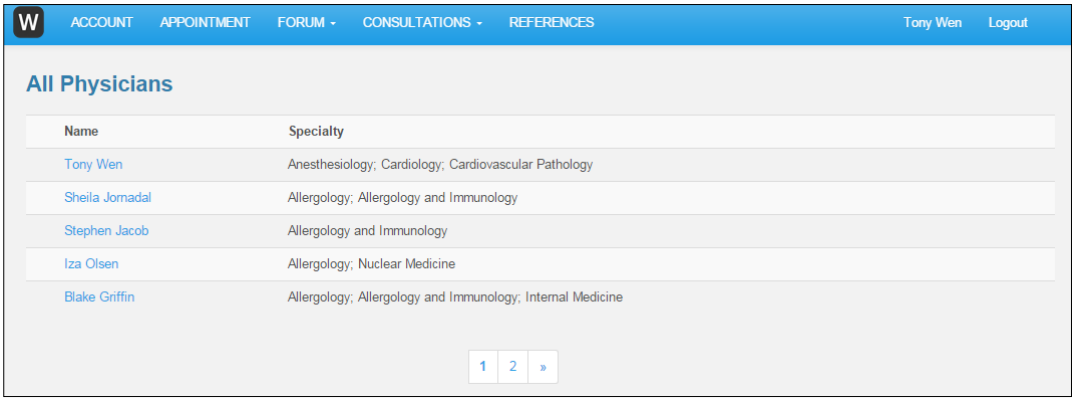


Figure 44 – WebPCS List of Physicians page

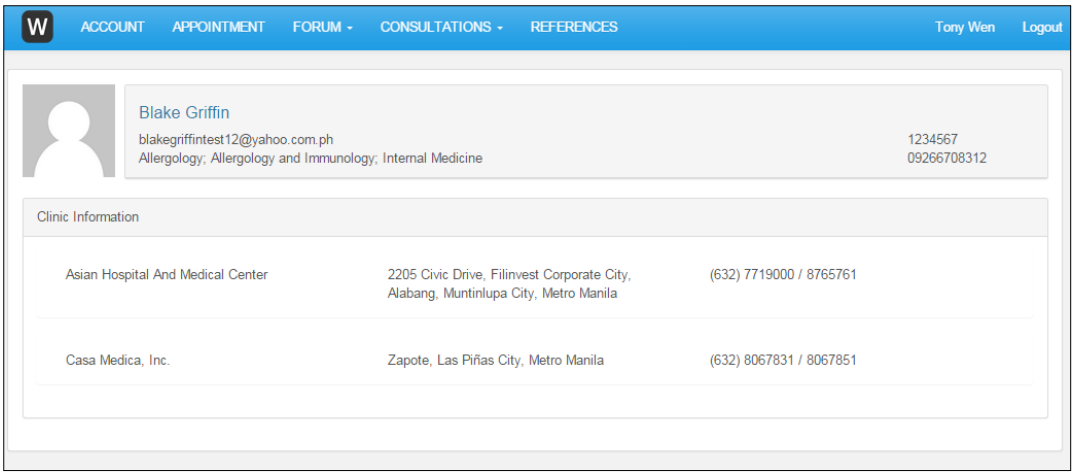


Figure 45 – WebPCS Profile page

The physician can update the information he wants to show in his profile. Figure 46 shows when physician updates his profile. This page also allows him to reset his password.

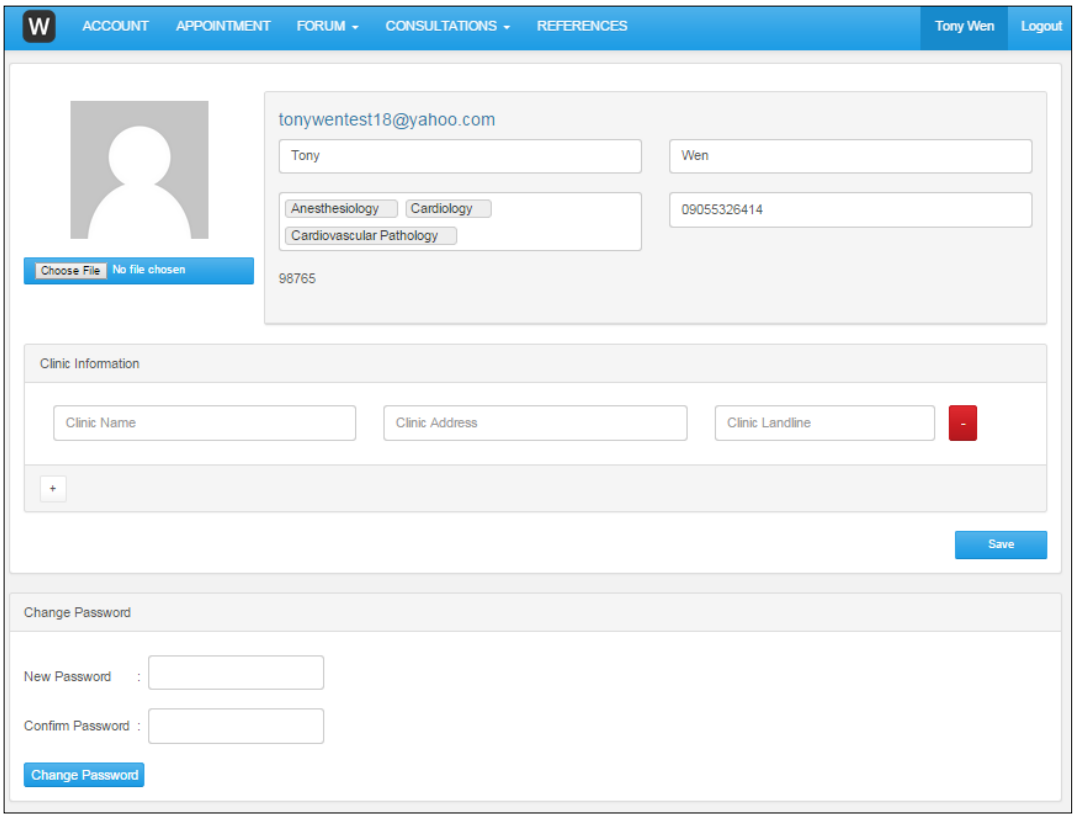


Figure 46 – WebPCS Edit Profile page

Account page is shown in Figure 47. Physician can configure in this page his receipt of SMS and e-mail notifications for appointment and forum updates.

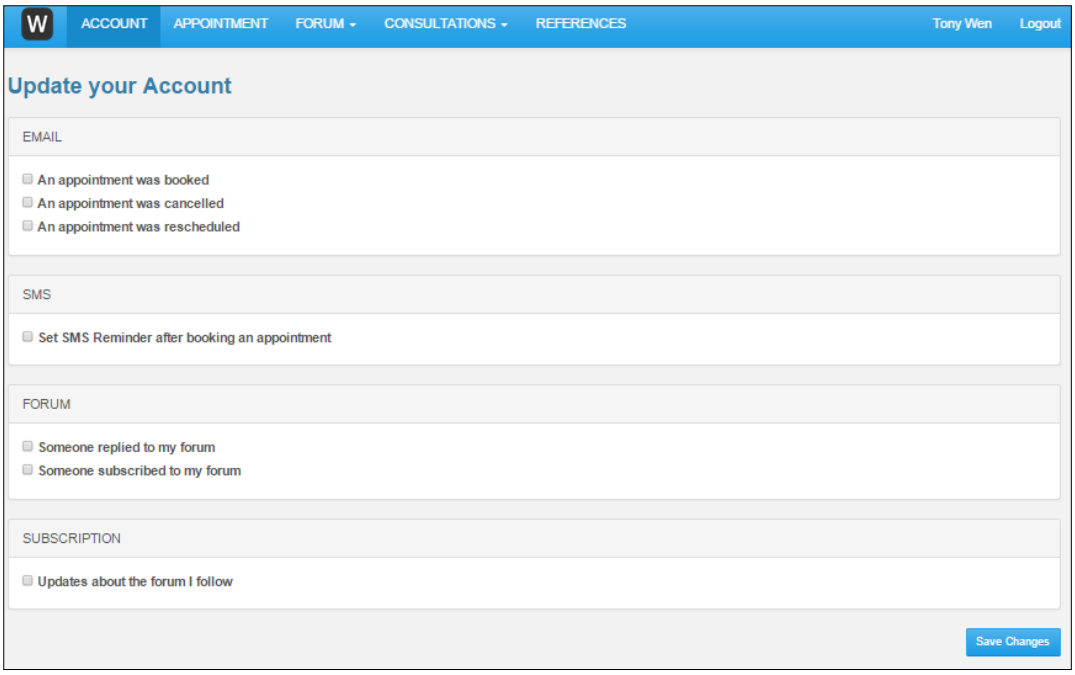


Figure 47 – WebPCS Account Setting page

The physician can monitor all his appointments on a calendar shown in Figure 48. It has the events plotted accordingly.

Figure 49 shows the modal window when a future event is clicked on the calendar. It contains the medical information and appointment details. Only Time field is modifiable. Found below are three buttons. Clicking on the Save button will save the updated schedule. The Delete button cancels the appointment. The Cancel button closes the modal window.

Figure 50 shows the modal window when a past event is clicked on the calendar. All fields are on read-only mode. The Cancel button closes the modal window.

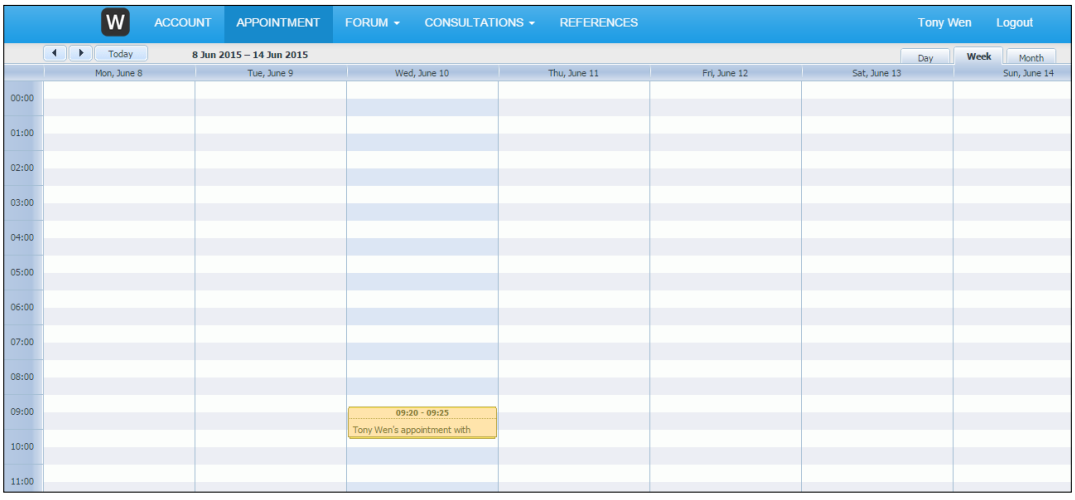


Figure 48 – WebPCS Planner page

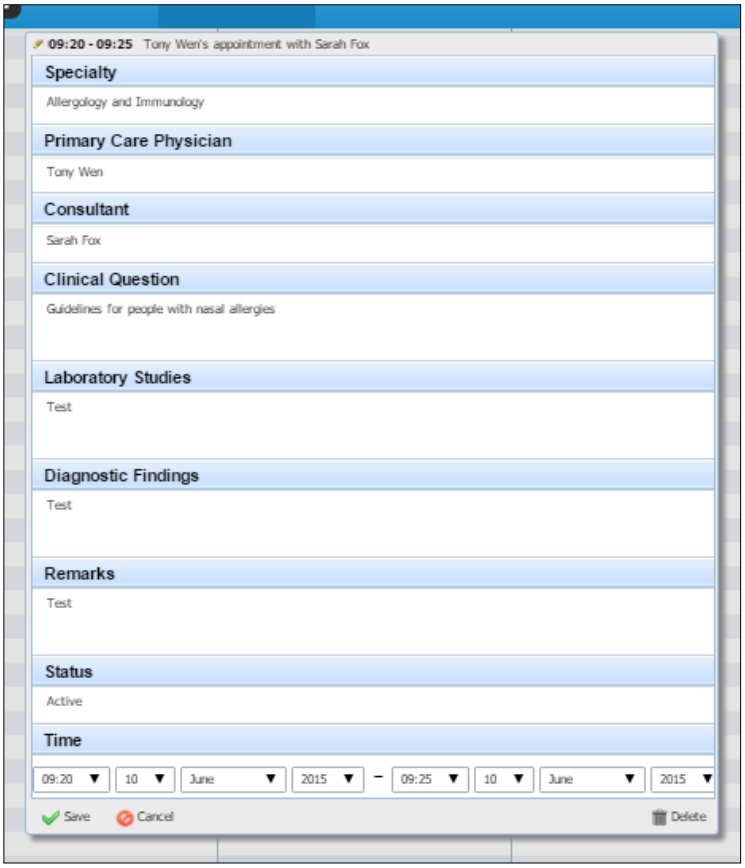


Figure 49 – WebPCS Modal Window of an Upcoming Appointment

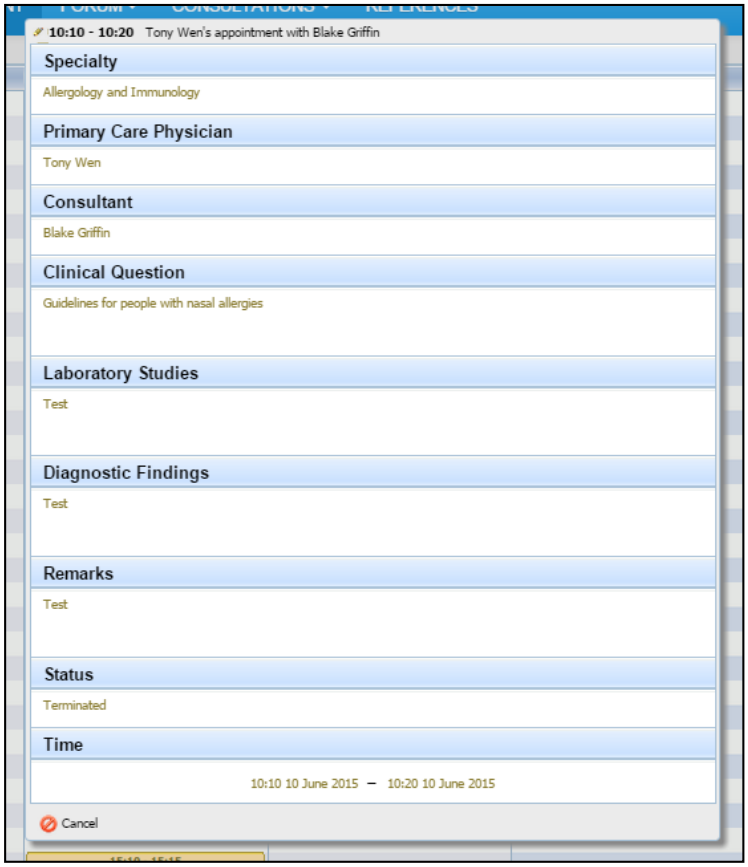


Figure 50 – WebPCS Modal Window of a Past Appointment

Figure 51 shows the Create New Discussion page which allows a physician to start a new discussion. All forums owned by him are listed in My Discussions page shown in Figure 52. Clicking on the forum title will open the thread. Forums posted by other physicians are listed in All Discussions page shown in Figure 53. View of such forum is shown in Figure 54. The Subscribe button allows a physician to subscribe to a thread. His forum subscriptions are listed in My Subscriptions page shown in Figure 55.

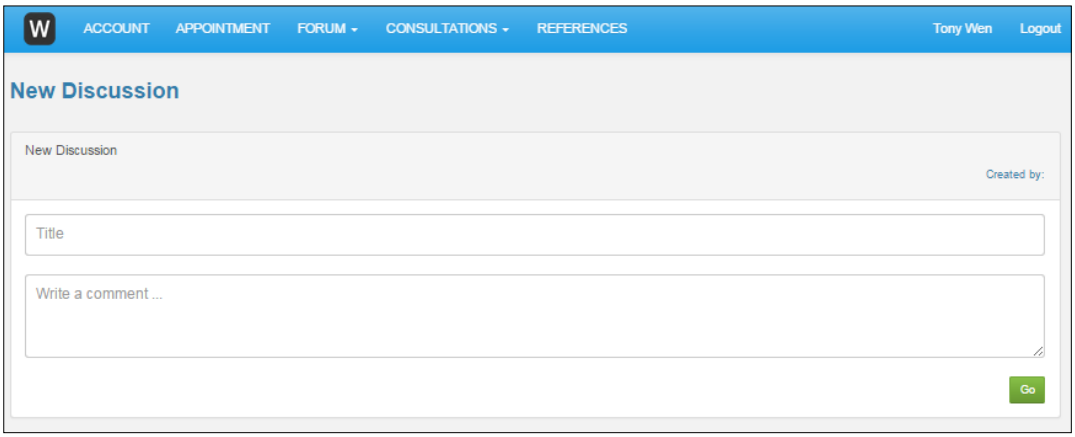


Figure 51 – WebPCS Create New Discussion page

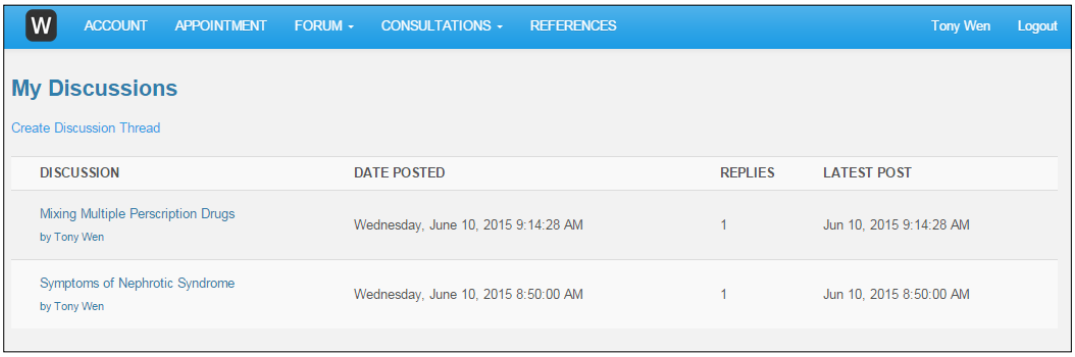


Figure 52 – WebPCS My Discussions page

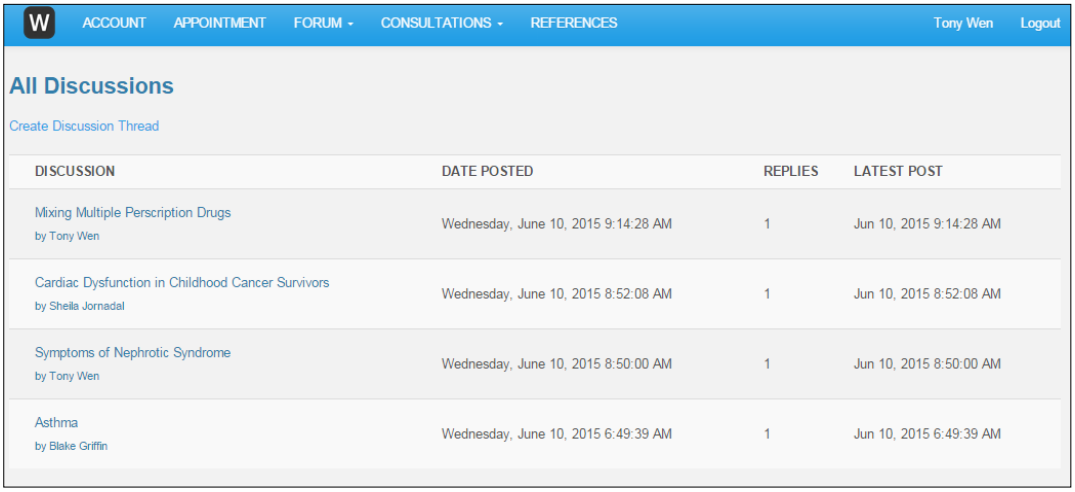


Figure 53 – WebPCS All Discussions page

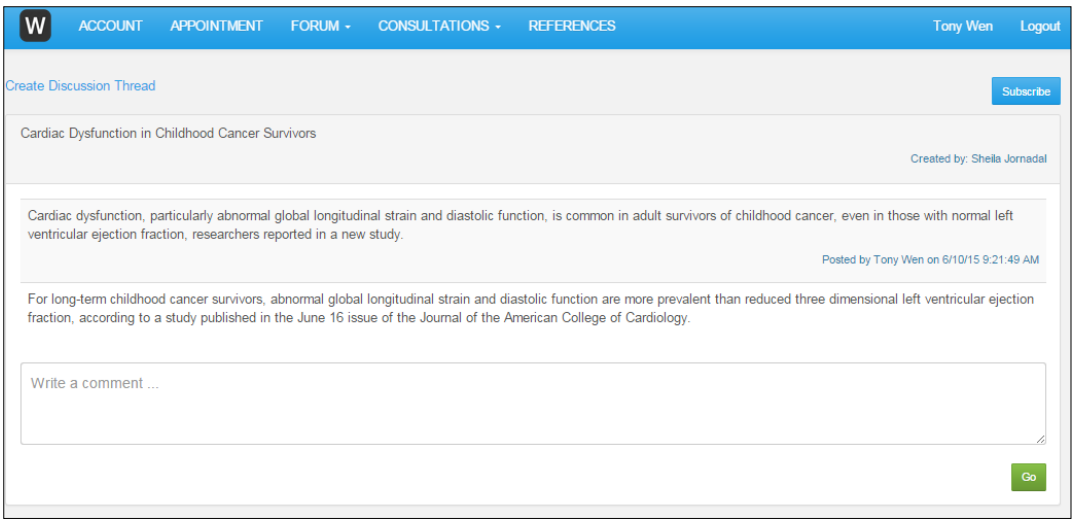


Figure 54 – WebPCS Thread Viewing

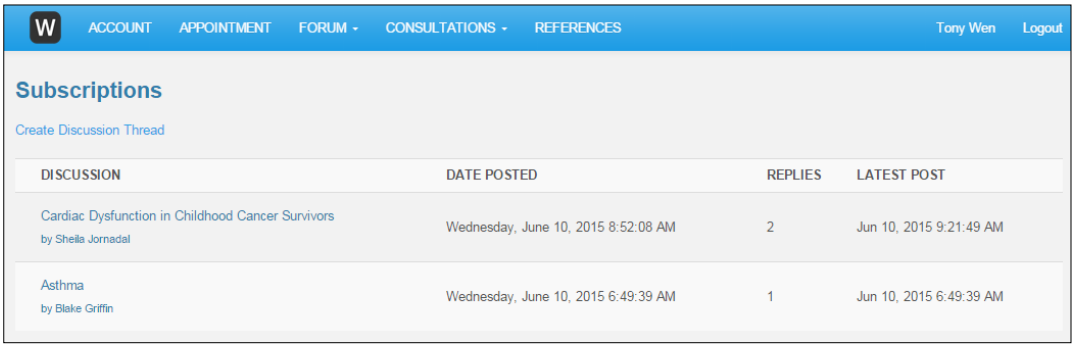


Figure 55 – WebPCS My Subscription page

The Create Consultation page allows a physician to book appointment with another. The requesting physician has to select first from the Specialty dropdown. The Find a Doctor button will display all the physicians with that specialty. Clicking on the names will display a calendar where schedule of both the requestor and consultant are plotted. This makes the checking of the availability of both physicians easier. See Figure 56.

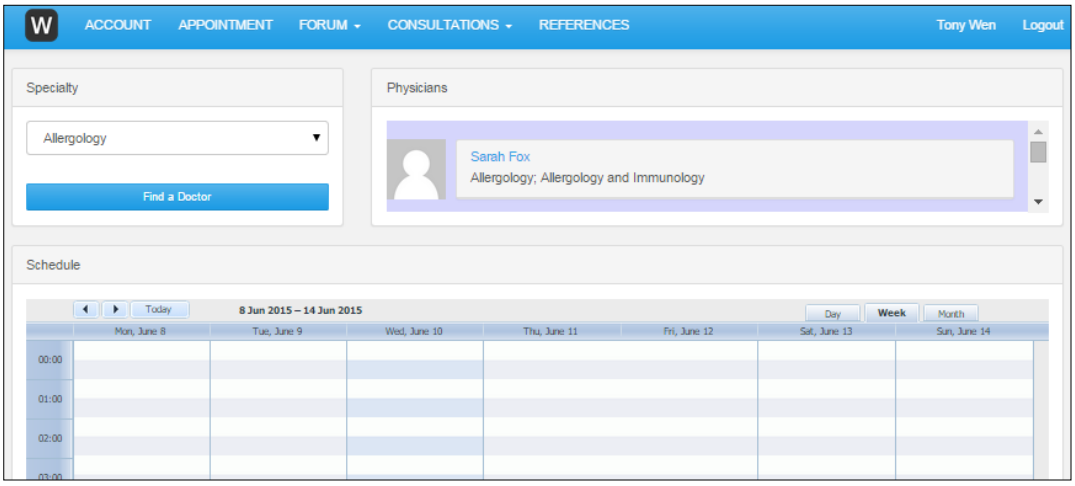


Figure 56 – WebPCS Specialty and Physician Selection

The calendar used has an intuitive clickable interface. To book an appointment, the requestor has to double-click on his desired schedule. A window for New Appointment then appears which allows the requestor to set date and time of appointment as shown in Figure 57. Clicking on the Save button will display the modal window for Clinical Questions which allows the requestor to select the clinical case information form he will use for consultation. See Figure 58.

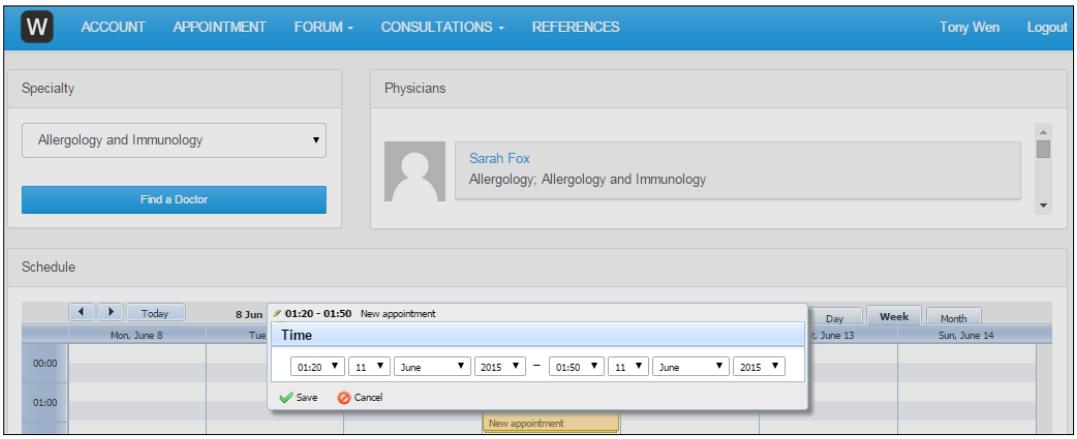


Figure 57 – WebPCS Modal Window for New Appointment

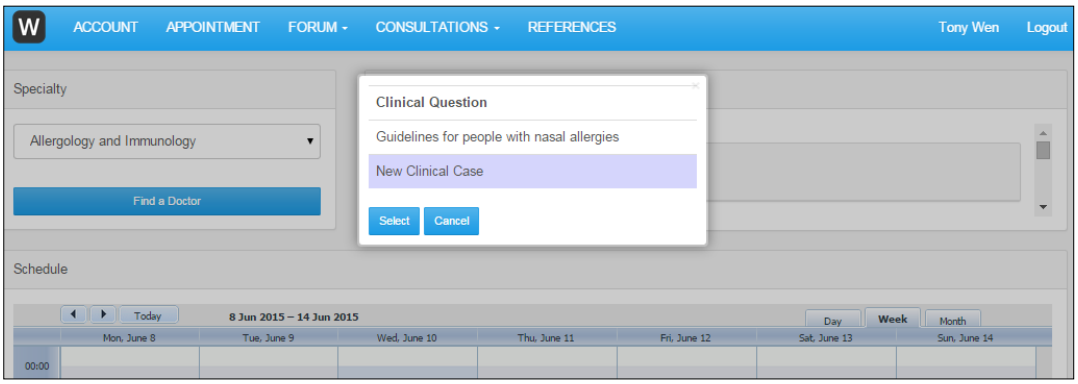


Figure 58 – WebPCS Modal Window for Clinical Questions

If requestor has cases that have not been resolved in past consultations, they will be shown in the above modal window. This allows him to make use of existing details and save time filling out the form. The fields are modifiable. To consult a new case, New Clinical Case option was provided where he can give details from scratch. The form for an existing case is shown in Figure 59. Figure 60 shows the form for a new case.

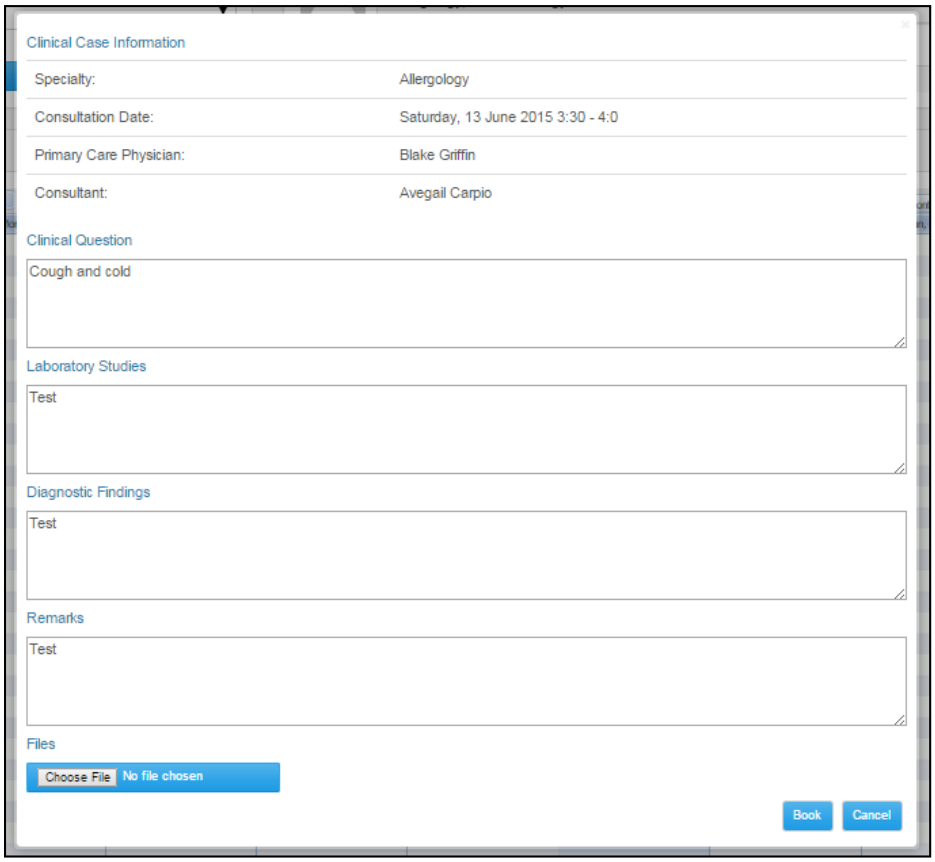


Figure 59 – WebPCS Clinical Case Information Form for an Existing Case

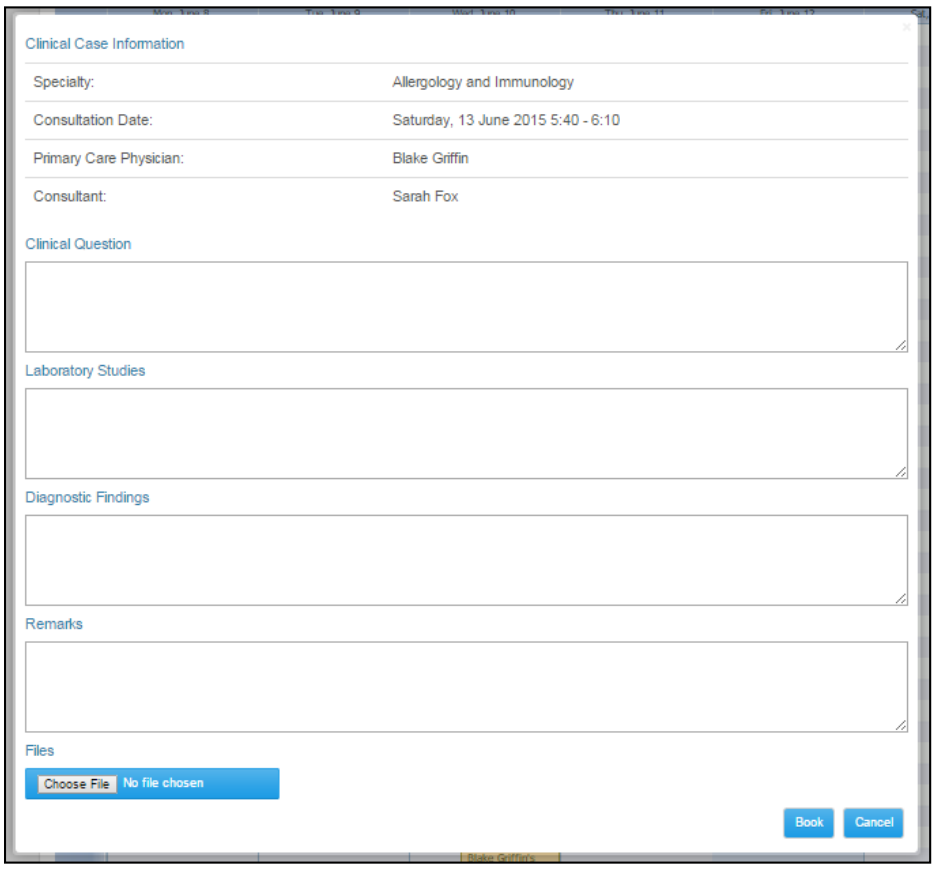


Figure 60 – WebPCS Clinical Case Information Form for a New Case

The Book button saves the appointment and clinical case as shown in Figure 61 and Figure 62, respectively.

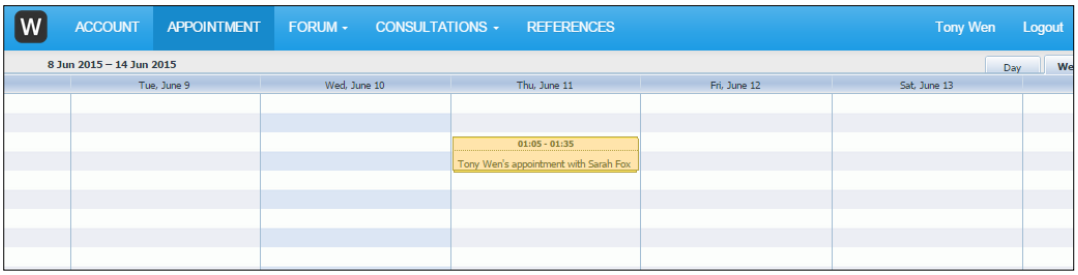


Figure 61 – WebPCS Appointment Saved

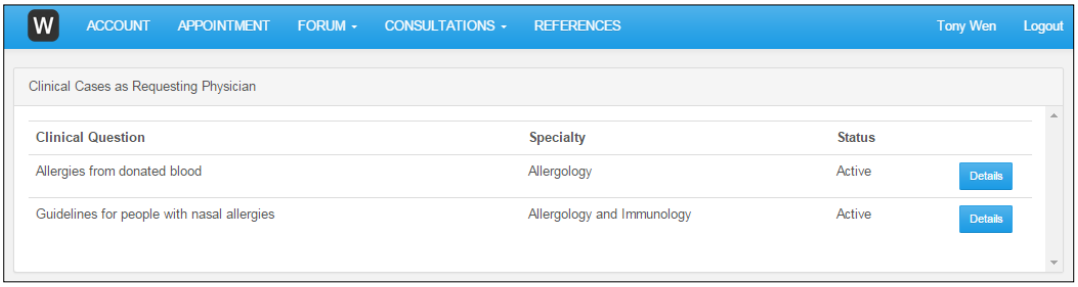


Figure 62 – WebPCS Clinical Case Saved

Clinical Cases page is shown in Figure 63. This page displays all cases of a physician. They are classed as Clinical Cases as Requesting Physician and Clinical Cases as Consultant.

The Status column determines the consultation status for a case. Active case means that a consultation is already set for that particular case. It will remain in Active status until the schedule of consultation comes – which from then on will take the Ongoing status. Case will have Unresolved status if consultation is already done and requestor has not clicked on the Close Clinical Case button yet. This may mean that his question was not resolved and he wants to seek advice to another physician. The Close button will shift the status into Closed.

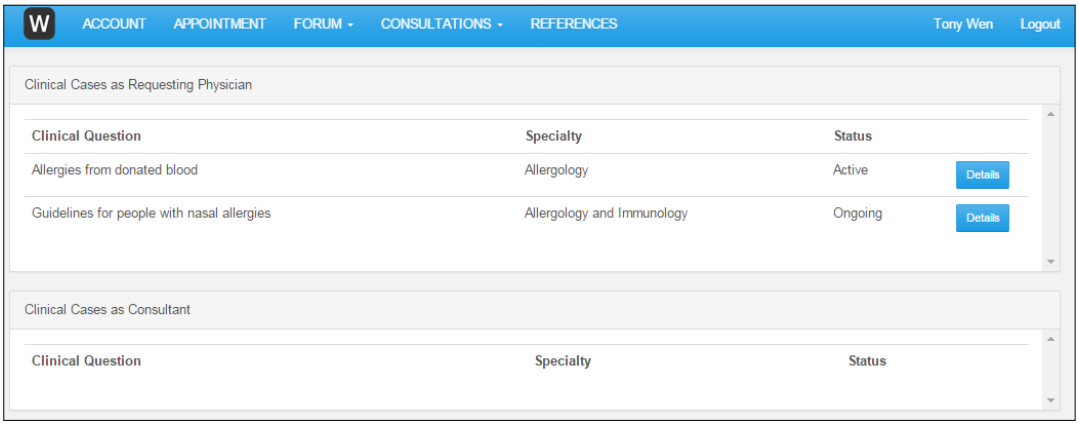


Figure 63 – WebPCS Clinical Cases page

The Details button displays the modal window for Clinical Case Information. For an Active case, View Chat Message History button can be found. Clicking on this will lead to the messenger. Figure 64 shows the view of the chat box for a new case. This view is applicable both to the requestor and consultant.

Figure 64 – WebPCS Chat Box for a New Case

However, perspective of chat box for an unresolved case varies according to viewer. Figure 65 shows the chat box view of a requestor. He can see the complete message history. He can see all the messages sent by all past consultants.

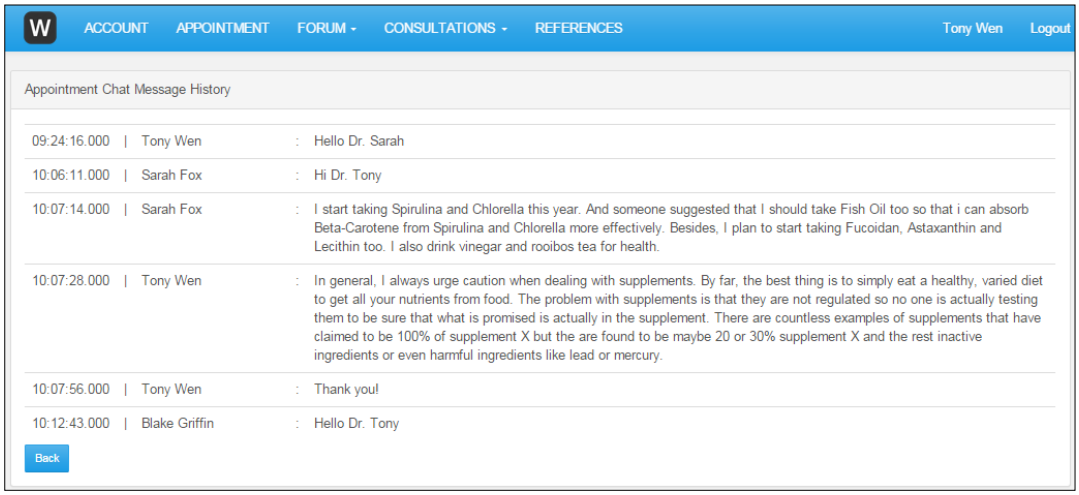


Figure 65 – WebPCS Chat Box for an Unresolved Case (Requestor)

For the consultant, Figure 66 shows his view of the chat box for an unresolved case. He can only view the messages sent by the requestor, but not those from past consultants.

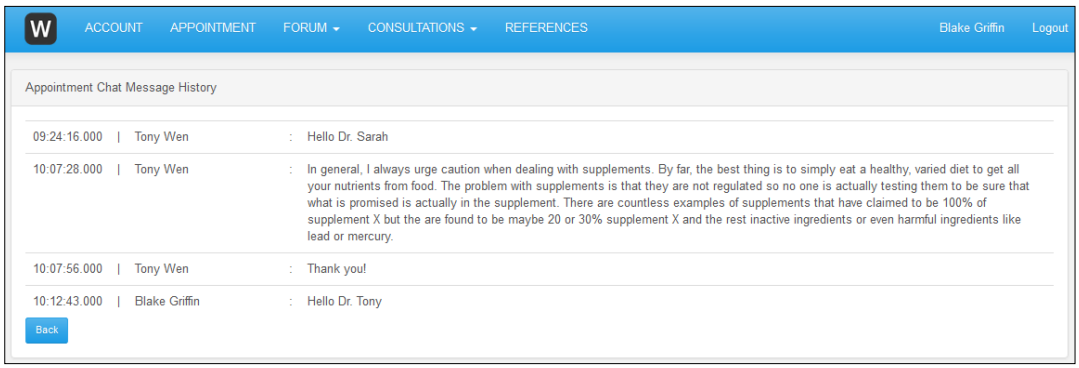


Figure 66 – WebPCS Chat Box for an Unresolved Case (Consultant)

The view of the chat box for an Ongoing consultation is shown in Figure 67. Clicking on the Terminate button will end consultation and mark case as Unresolved. Clicking on the Close button will end consultation and mark case as Closed.

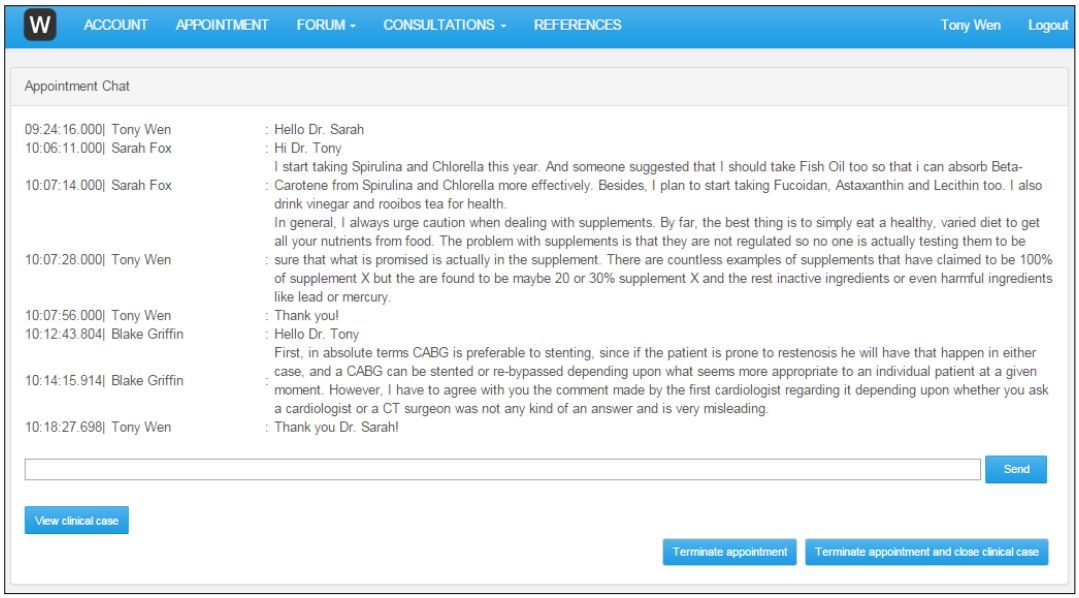


Figure 67 – WebPCS Chat Box for an Ongoing Consultation

A physician can generate two types of consultation reports – a Requesting Physician report and a Consultant report. Figure 68 shows an example of the latter. An example of the former is shown in Figure 69. Clicking on the Export PDF report will save the report in PDF.

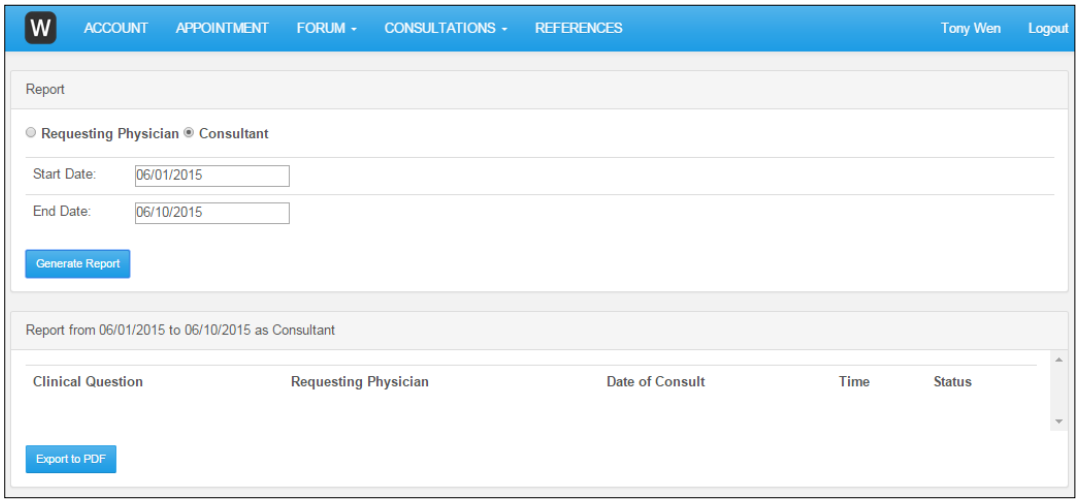


Figure 68 – WebPCS Consultant Report

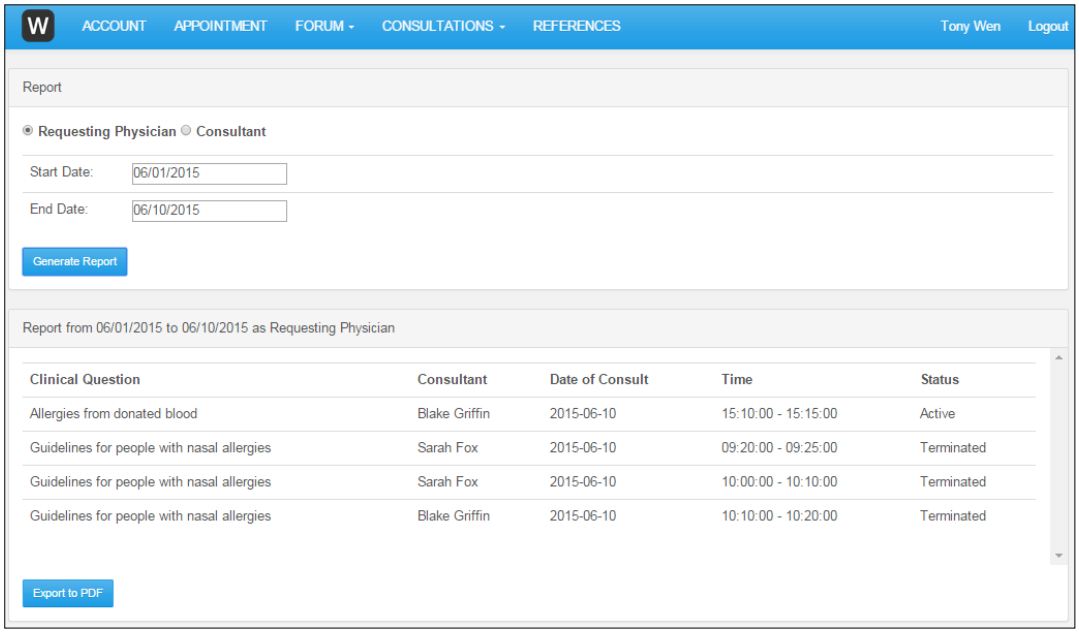


Figure 69 – WebPCS Requesting Physician Report

Available online references in the system can be found on References page shown in Figure 70.

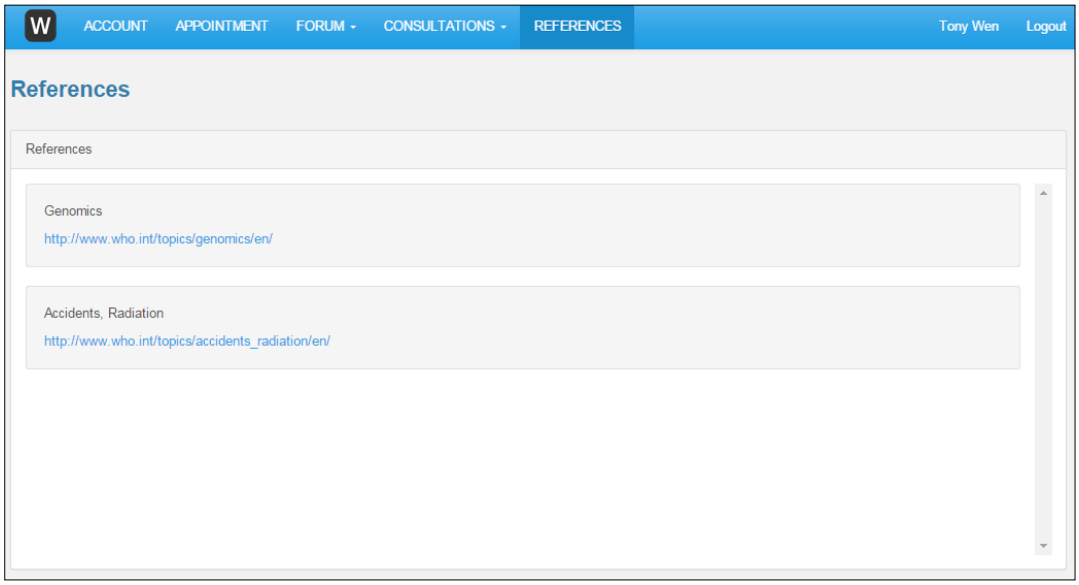


Figure 70 – WebPCS Online References page

Figure 71 shows the Dashboard for admin access level. The admin dashboard keeps shows the Consultation Trends of all physicians in the system presented in bar chart.

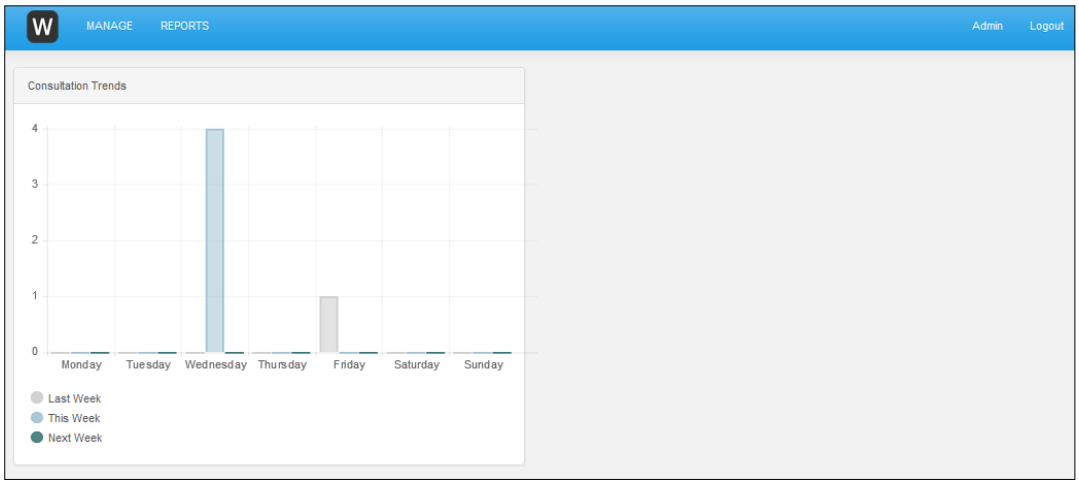


Figure 71 – WebPCS Dashboard for Admin Access Level

Admin can view all system users in Users page shown in Figure 72. This page allows the admin to remove users from the system. He can also update here the first and last name fields. Clicking on the Add New Admin button will display a form below shown in Figure 73 for the creation of a new admin account.

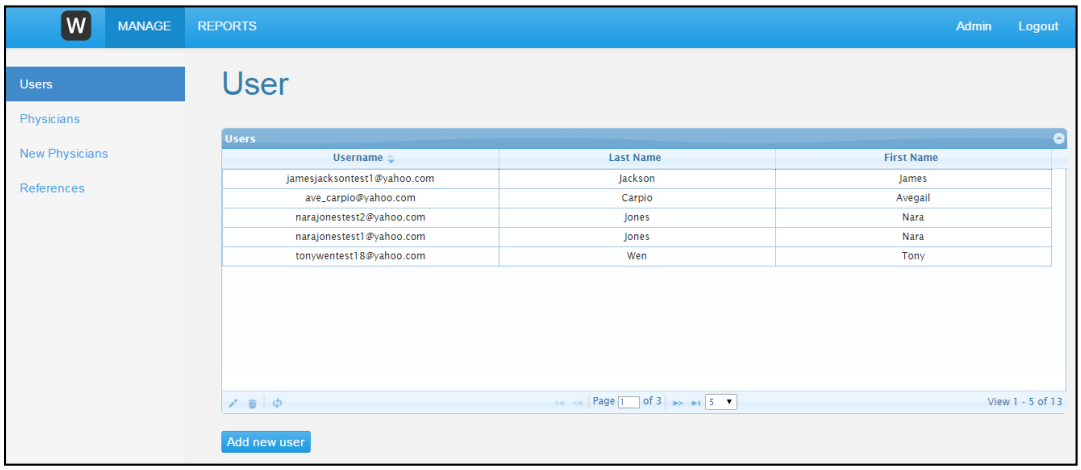


Figure 72 – WebPCS User page

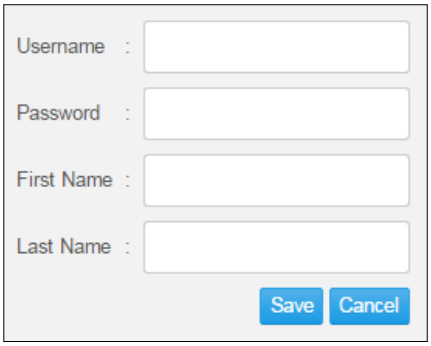


Figure 73 – WebPCS Add New Admin form

Figure 74 shows the Physicians page where admin can view the table of physicians. Clicking on a physician entry will allow the admin to update information as shown in Figure 75. Admin can also create a physician account by clicking on the Add New Physician button. The Add New Physician form is shown in Figure 76.

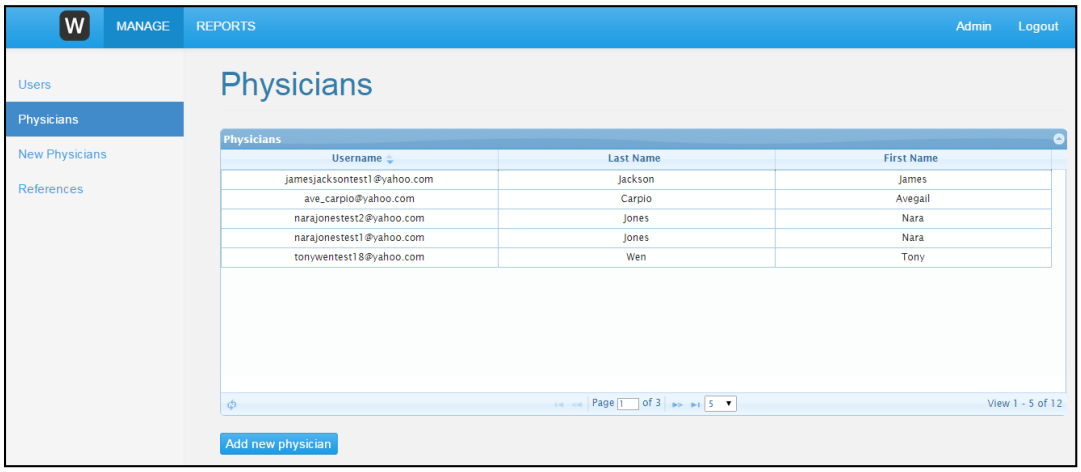


Figure 74 – WebPCS Physicians page

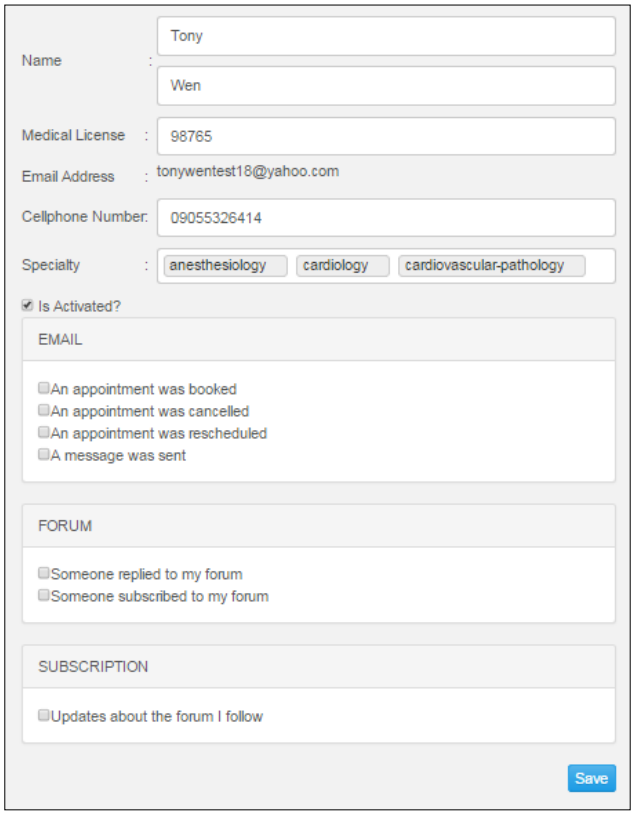


Figure 75 – WebPCS Physician Information Updated

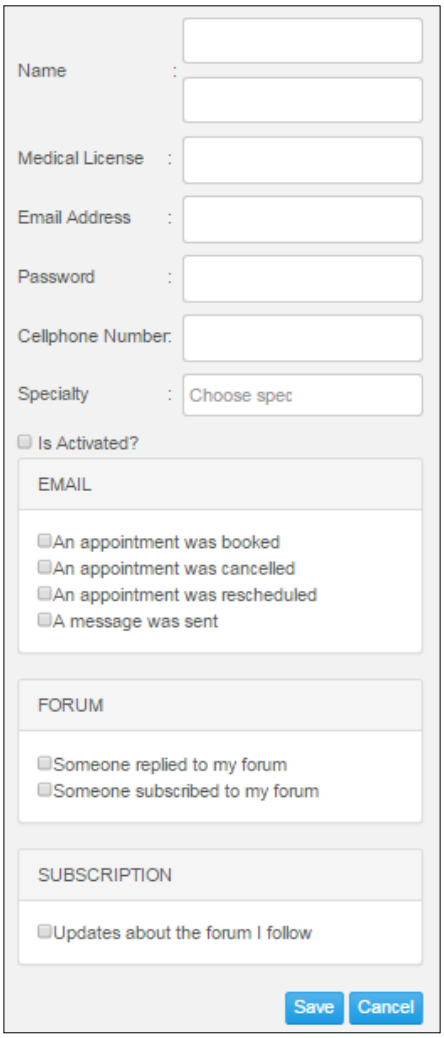


Figure 76 – WebPCS Add New Physician form

Figure 77 shows New Physicians page with pending account requests. Clicking on an entry allow the admin to approve or decline request as shown in Figure 78.

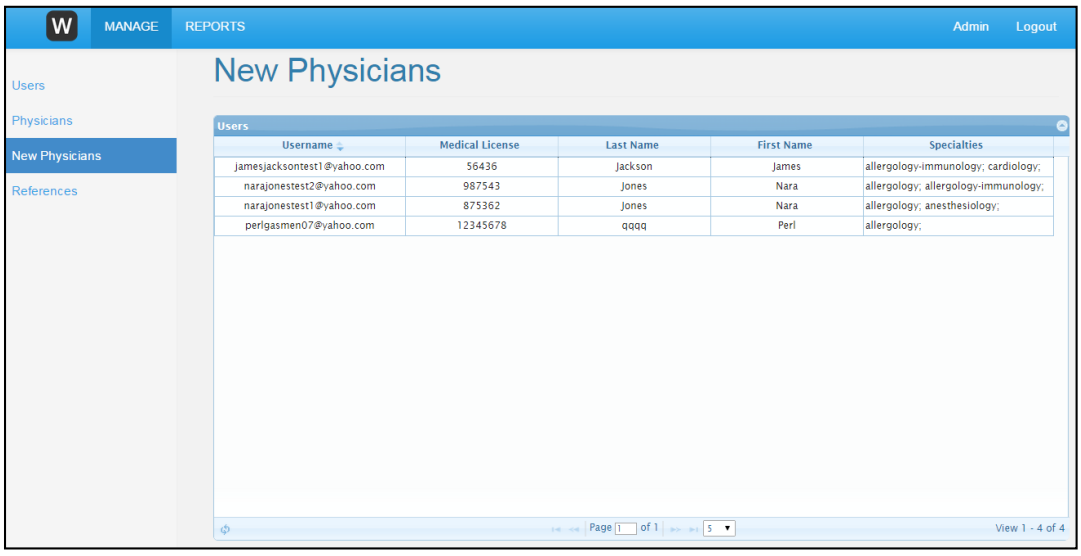


Figure 77 – WebPCS New Physician page

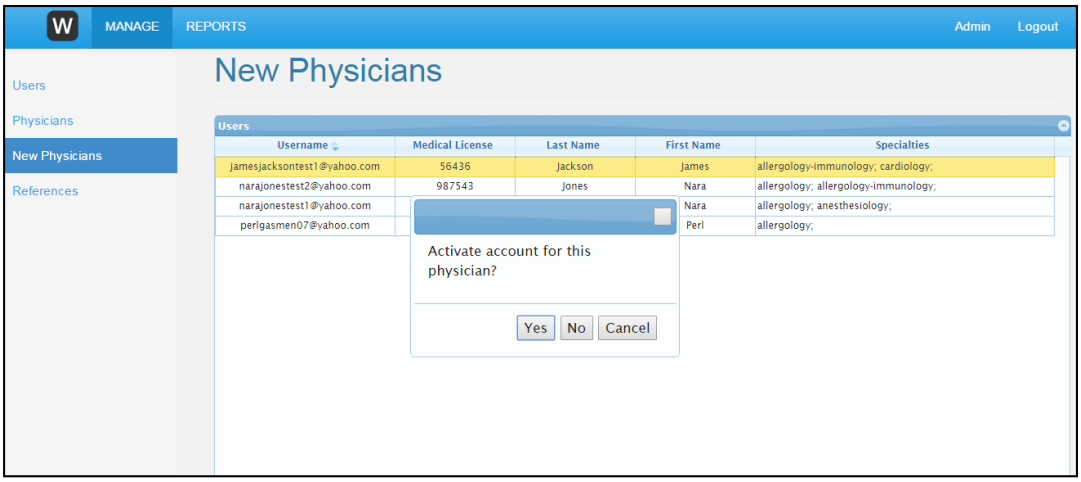


Figure 78 – WebPCS Account Request Confirmation

Admin can manage references in References page shown in Figure 79. Clicking on an entry will allow the admin to update the reference details. He can also remove references in the system. He can add new ones by clicking on Add References button. Figure 80 shows the Add References form.

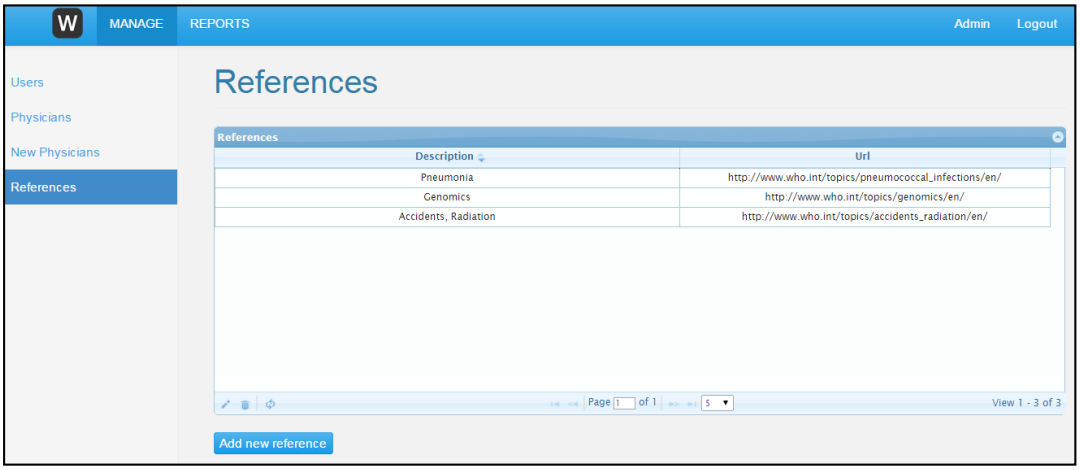


Figure 79 – WebPCS References page

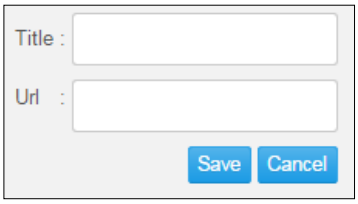


Figure 80 – WebPCS Add References form

6.2 E-mail Notification Screenshots

After successful registration in the system, user will receive a confirmation message via e-mail. An example is shown in Figure 81.

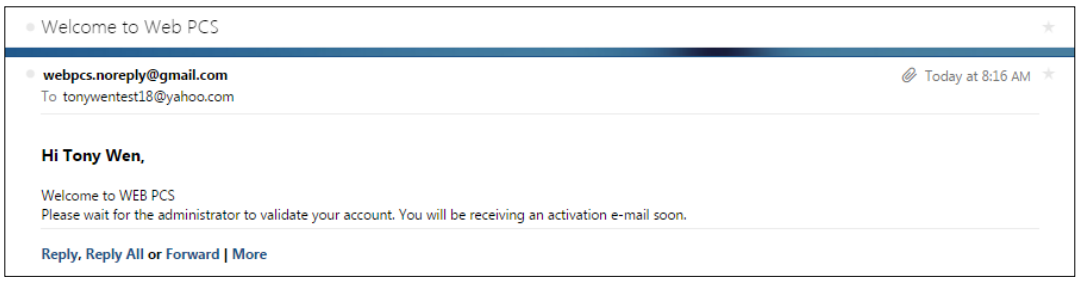


Figure 81 – WebPCS Successful Registration E-mail Message

User will receive a confirmation message via e-mail once admin approves or denies his request. Examples are shown in Figure 82 and Figure 83, respectively.

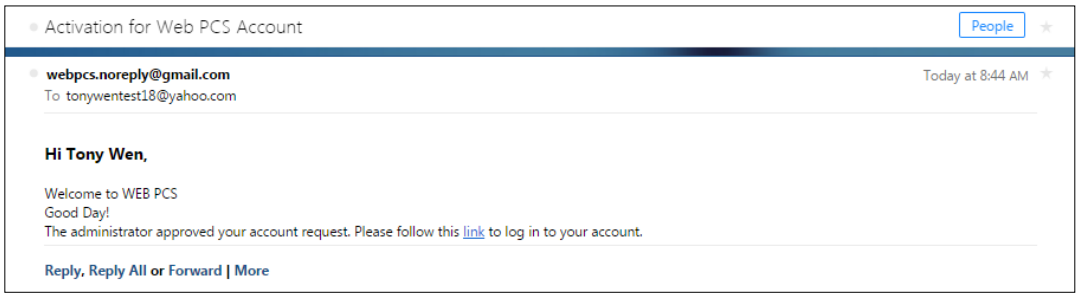


Figure 82 – WebPCS Request Approved E-mail Message

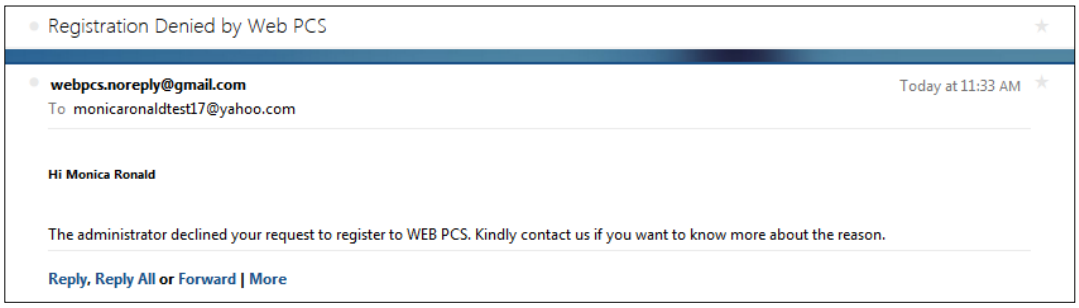


Figure 83 – WebPCS Request Denied E-mail Message

For a password reset request, system will send the user his new password via e-mail as shown in Figure 84.

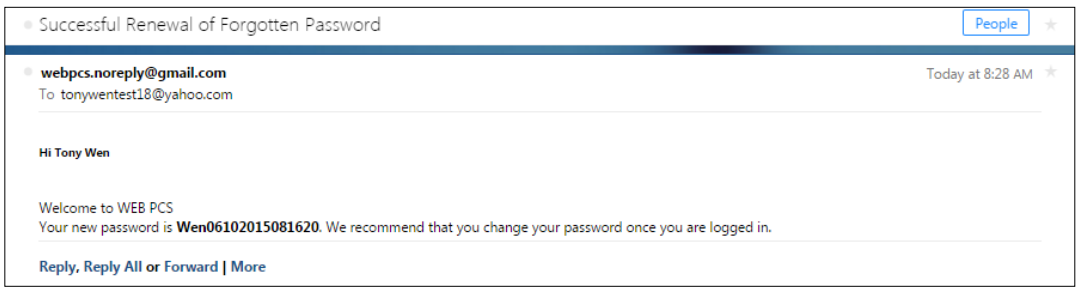


Figure 84 – WebPCS Password Reset E-mail Message

Both the requesting physician and consultant can receive a notification for a booked appointment via e-mail and SMS. Example is shown in Figure 85.

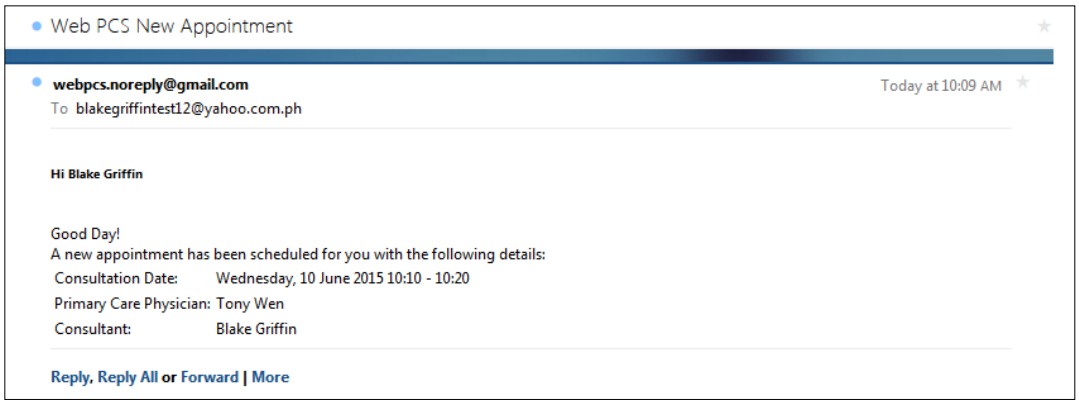


Figure 85 – WebPCS Booked Appointment E-mail Notification

Any update on schedule of appointment will activate sending of notifications to both requesting physician and consultant. Example of e-mail notification is shown Figure 86.

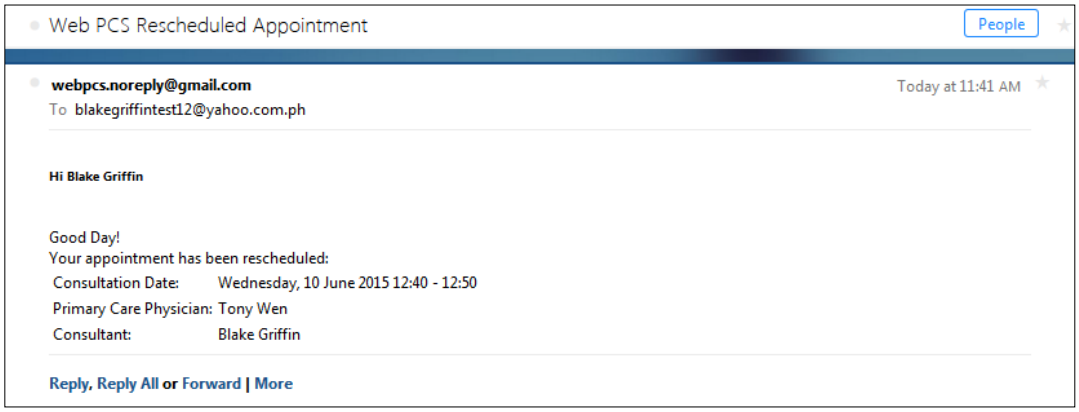


Figure 86 – WebPCS Rescheduled Appointment E-mail Notification

Cancellation of appointment will set off notice via e-mail and SMS to both requesting physician and consultant. Example is shown in Figure 87.

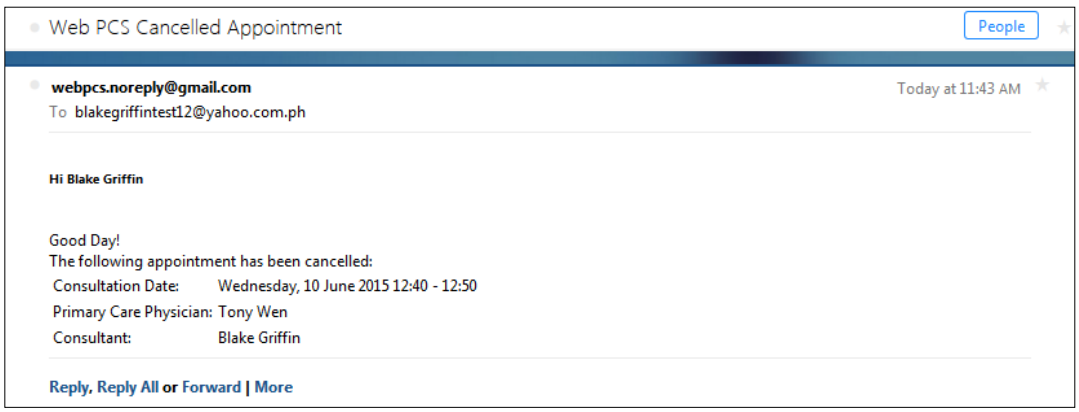


Figure 87 – WebPCS Canceled Appointment E-mail Notification

The forum owner can be notified via e-mail when someone subscribed to one of his forums. An example is shown in Figure 88.

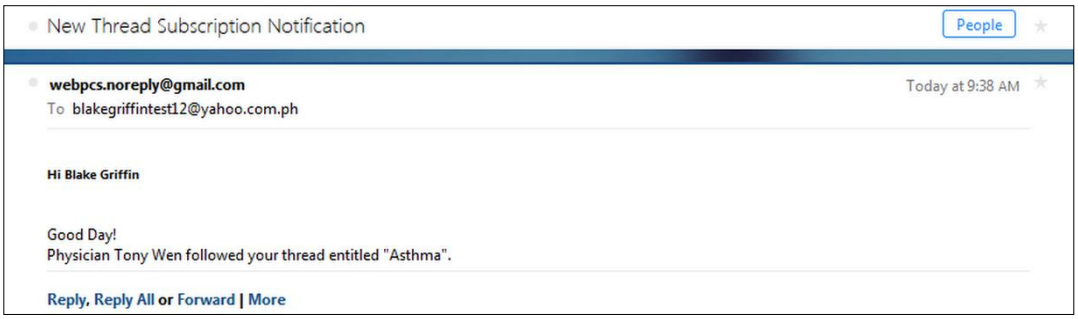


Figure 88 – WebPCS Forum Subscriber E-mail Message

The forum owner can be notified via e-mail when someone posted comments to one of his forums. An example is shown in Figure 89.

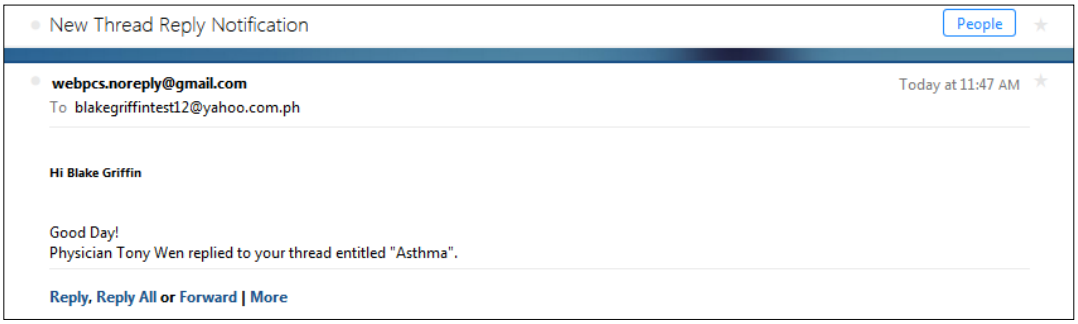


Figure 89 – WebPCS Reply to Forum E-mail Message

VII. DISCUSSION

WebPCS is an application which enables consultation between physicians regarding patient care. It was developed to address the limitations of the current medical consultation schemes such as delay in referral, reduced opportunities for curbside consultation, inability to produce report in a telephone consultation, inconvenient transfer of medical information from a physical record to an e-mail system, and the need for specialized equipment to assist telemedicine. With WebPCS, physicians have the tools they need to improve productivity.

WebPCS has two types of users – physician and system administrator.

The system administrator manages the entire system. He can view all user accounts and update their details. He can create new accounts and delete existing ones. He is also the one who approves and declines account requests. He is able to see the tally of consultations performed by physicians using the system and generate a report. He is also in control of the online references for the physicians.

The physician performs the main functionalities of the system. He has the capability to book appointment and consult online with physicians in a variety of specialties. Reports can be generated for the consultations he performed using the system. He can join in online discussions where he can meet other physicians and talk about their clinical interests. Also, he has access to the online references.

WebPCS implemented several features expedient to consultation.

Online scheduling is made easier by enabling a calendar. The rich set of features of the calendar allows physicians to quickly manage their appointments. It can be displayed in three views – Day, Week, and Month. Physicians can navigate through different views using tabs at the top of the calendar.

The system has displayed two flat calendars. First is a personal calendar where a physician can monitor all of his appointments. Clicking on an appointment box will open a lightbox which contains the appointment details. Only Time section can be updated while the rest of the sections are set to read-only mode. Second is configured for the scheduling service that will be provided by the system. The calendar allows a requesting physician to see open appointment times of other physicians and book right away. He just needs to click on the calendar his desired appointment schedule and follow the quick prompts to book an appointment.

The system allows a physician to send appointment notifications via e-mail and/or SMS. After the requesting physician has the appointment booked, he and the consultant will

receive e-mail and/or SMS message to advise them of their new appointment. In addition to e-mail and/or SMS notification, WebPCS can send both physicians reminders via SMS some time prior to the start of their appointment, ranging from 1 week to 5 minutes. These are to ensure no one forgets their appointment. These features also track the appointments easier and help to reduce the number of no-shows.

WebPCS enables the requesting physician to submit to the consultant his advanced work on a case before the consultation. He can fill out the case information form online and submit it in advance. A mechanism to save the details for his future appointments was also implemented. This allows the consultant to better understand the patient's medical history and any previous investigative work done by the requesting physician before consultation. This also saves both physicians time on making decisions and can increase the legibility and accuracy of the records.

WebPCS allows the system administrator and physicians to generate reports which can then be exported to a printable format such as PDF. A physician can gain insight on his own consultation activities while the system administrator can gain insight on consultation activities of all physicians over a specific period of time.

VIII. CONCLUSION

WebPCS is an application designed to aid medical consultations. Several tools were implemented to address the limitations of the existing consultation models, and improve the communication between physicians and the coordination of care in the country through developing a web system.

WebPCS can be accessed by a system administrator. This type of user is in charge of the entire system. He is the one who approves and declines account requests. He can create new accounts and delete existing ones. He can also update account details. He is able to see the tally of consultations performed by all physicians using the system and able to generate a report. He is also in control of the online references for the physicians.

WebPCS can also be accessed by physicians. They are the ones who implement the main functionalities of the system. A physician has the capability to book appointment and consult online with other physicians in a variety of specialties. Reports can be generated for the consultations he performed using the system. He can join in online discussions where he can meet other physicians and talk about their clinical interests. Also, he has access to the online references.

An abundance of features ensures that a physician can make his online booking and consultation work the way he wants them to work. Online booking makes use of a calendar which allows physicians to see the schedule of other physicians. A personal calendar allows them to quickly manage their appointments. Appointment updates send off notifications via e-mail and/or SMS. WebPCS implemented such tool to track appointments easier and help reduce the number of no-shows. Upon booking, clinical case details are already collected so that physicians can review the medical history of the patient and any previous investigative work done before consultation. The system generates reports for the consultation activities of all physicians over a specific period of time. Online discussions and online references are also available in the system to support and improve care coordination.

IX. RECOMMENDATION

WebPCS can further be enhanced in several ways. Currently, it doesn't offer ways to connect the schedule of a physician to other calendar programs. Synchronization will allow him to define complex schedule. He can either let the availability of a schedule depend on a Google Calendar or publish appointment information to a calendar program like Outlook on his computer or a phone.

WebPCS allows a physician to create only one booking at a time. A repeat booking could be implemented so that the physician doesn't have to create each booking separately. For this, he will now have the option to repeat his booking daily, weekly, monthly by date or monthly by day, all until a particular end date.

WebPCS could implement a video communication to discuss medical concerns and facilitate information exchange between physicians at some distances from each other. This provides direct communication between physicians regardless of their current location.

Because of the growing demand for mobile healthcare services, WebPCS could shift to a mobile application which will still provide all features of the computer application yet will give an exceptional consultation experience to physicians with just few taps on screen.

X. BIBLIOGRAPHY

[1] Slama, T. (2012). *The Curbside Consultation*. Retrieved January 28, 2014, from http://www2.medicine.wisc.edu/home/files/8._Curbside_Consults_Thomas_G._Slama.pdf

[2] ACOG Committee Opinion (2007, Reaffirmed 2013). Seeking and Giving Consultation. *International Journal of Gynecology and Obstetrics: The Official Organ of the International Federation of Gynecology and Obstetrics*, *73*(1): 81-85.

[3] Perley, C. (2006). Physician Use of the Curbside Consultation to Address Information Needs: Report on a Collective Case Study. *Journal of the Medical Library Association*, *94*(2): 137-144.

[4] Horner, K., Wagner, E., and Tufano, J. (2011). Electronic Consultations between Primary and Specialty Care Clinicians: Early Insights. *The Commonwealth Fund*, *23*: 1-14.

[5] Cusack, C., Pan, E., Hook, J., Vincent, A., Kaelber, D., Bates, D., and Middleton, B. (2007). *The Value of Provider-to-Provider Telehealth Technologies*. Charlestown, MA: Healthcare Information and Management System Society.

[6] Department of Health (2012). The Philippine Health Systems at a Glance. *National Objectives for Health 2011-2016*. Retrieved February 7, 2014, from www.doh.gov.ph/sites/default/files/3%20Chapter1.pdf

[7] The Canadian Medical Protective Association (2013). Rural Practice - Strategies to Reduce Medico-Legal Risk. Retrieved from https://oplfrpd5.cmpa-acpm.ca/en/duties-and-responsibilities/-/asset_publisher/bFaUiyQG069N/content/rural-practice-%E2%80%94-strategies-to-reduce-medico-legal-risk

[8] Liddy, C., Rowan, M., Afkham, A., Maranger, J., and Keely, E. (2013). Building Access to Specialist Care through E-consultation. *Open Medicine*, *7*(1).

[9] Scherpbier-de Haan, N., van Gelder, V., Weel, C., Vervoort, G., Wetzels, J., and de Grauw, W. (2013). Initial Implementation of a Web-Based Consultation Process for Patients with Chronic Kidney Disease. *Annals Family of Medicine*, *11*(2).

[10] Caffery, L. and Smith, A. (2010). A Literature Review of E-mail-based Telemedicine. *IOS Press*, *161*: 20-34.

[11] Ali, E. (2008). *Twenty Years with Teledermatology in North Norway*. Master Thesis, Institute of Clinical Medicine, Faculty of Medicine, University of Tromso, Norway.

[12] Hjelm, N. (2005). Benefits and Drawbacks of Telemedicine. *J Telemed Telecare*, *11*(2): 60-70.

[13] Shdefat, A. (2010). *Medical E-consultation System*. Master Thesis, Faculty of Information Technology, University Utara Malaysia, Malaysia.

[14] Bluteau, J., Kitahara, I., Kameda, Y., Noma, H., Kogure, K., and Ohta, Y. (2005). Visual Support for Medical Communication by using Projector-Based Augmented Reality and Thermal Markers. *Association for Computing Machinery*, *ICAT 2005*, 98-105.

[15] Saurage Marketing Research (2011). *Online Access to Doctors*. Retrieved from http://www.saurageresearch.com/healthcare-key-findings-novemberdecember-2011/#5

[16] Wei, L. (2008). *E-health Consultation System*. Undergraduate Thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia.

[17] EliteHealth Website. Retrieved from http://www.elitehealth.com/medical_econsultation.php

[18] Consult a Doctor Website. Retrieved from www.consultadr.com

[19] Gardiner, C., Gott, M., and Ingleton, C. (2012). Factors Supporting Good Partnership Working between Generalist and Specialist Palliative Care Services: A Systematic Review. *British Journal of General Practice*, *62*(598): e353-e362.

[20] Lin, C. (2012). Improving Care Coordination in the Specialty Referral Process between Primary and Specialty Care. *N C Medical Journal*, *73*(1): 61-62.

[21] O'Malley, A. and Reschovsky, J. (2011). Referral and Consultation Communication between Primary Care and Specialist Physicians. *Arch Intern Med*, *171*(1): 56-65.

[22] Esquivel, A., Sittig, D., Murph. D., and Singh, D. (2012). Improving the Effectiveness of Electronic Health Record-Based Referral Processes. *BioMed Central Medical Informatics and Decision Making*, *12*:107.

[23] Kim-Hwang, J., Chen, A., Bell, D., Guzman, D., Yee, H., and Kushel, M. (2010). Evaluating Electronic Referrals for Specialty Care at a Public Hospital. *J Gen Intern Med*, *25*(10): 1123-1128.

[24] Wager, K., Lee, F., and Glaser, J. (2013). *Health Care Information Systems: A Practical Approach for Health Care Management Third Edition*. San Francisco, CA: Jossey-Bass.

[25] Mawell Website. Retrieved from http://www.mawell.com/web/page.aspx?refid=788

[26] Luk, R., Ho, M., and Aoki, P. (2008). Asynchronous Remote Medical Consultation for Ghana. *Computing Research Repository*, abs/0801.1927.

[27] Ickenstein, G., Groß, S., Tenckhoff, D., Hausn, P., Becker, U., Klisch, J., and Isenmann, S. (2010). An Empirical Analysis of the Current Need for

Teleneuromedical Care in German Hospitals without Neurology Departments. *International Journal of Telemedicine and Applications*. DOI: 10.1155/2010/916868.

[28] eConsult Website. Retrieved from http://econsultla.com/home

[29] Peter, K. (2006). Coding Consultation E/M Services Correctly. *Journal of AHIMA 77*, *10*: 70-72.

[30] Cohn, S. (2003). The Role of the Medical Consultant. *The Medical Clinics of North America*, *87*: 1-6.

[31] Canadian Medical Association (2007). *Patient-centered Collaborative Care*. Retrieved December 8, 2013, from http://fhs.mcmaster.ca/surgery/documents/CollaborativeCareBackgrounderRevised.pdf

[32] Department of Health Profile. Retrieved from http://www.doh.gov.ph/node/97.html

[33] Department of Health Deployment Programs. Retrieved from http://www.doh.gov.ph/content/what-are-deployment-programs.html

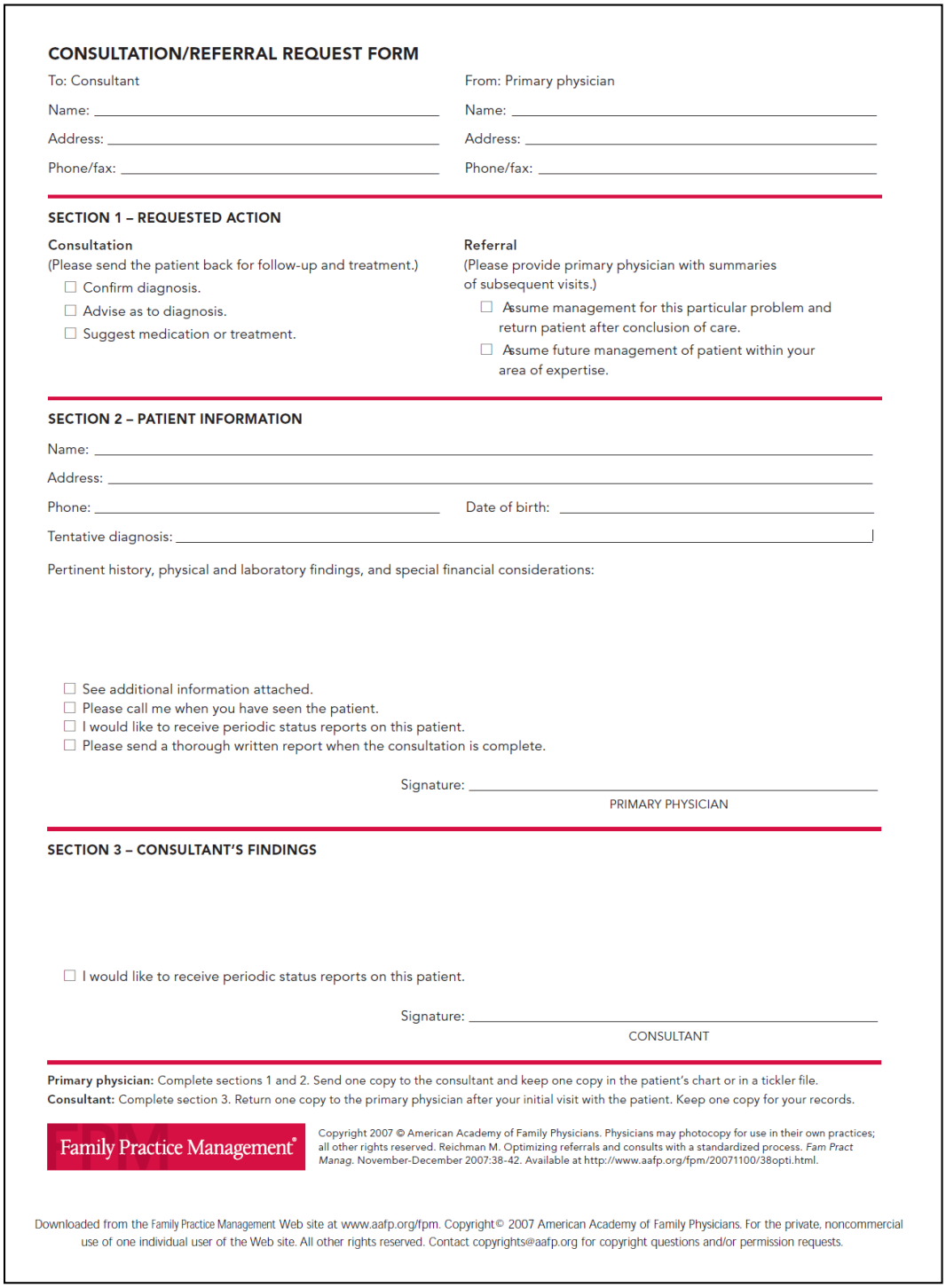
[34] Consultation and Referral Request Form provided by the American Academy of Family Physician. Retrieved from http://www.aafp.org/fpm/2007/1100/p38.html

[35] List of Medical Specialties. Retrieved from http://thefilipinodoctor.com/Medical_Specialties.php

XI. APPENDIX

11.1 Form

The WebPCS clinical case form is based on the *Consultation / Referral Request Form* provided by the *American Academy of Family Physician* [34].



CONSULTATION/REFERRAL REQUEST FORM

To: Consultant

Name: _____

Address: _____

Phone/fax: _____

From: Primary physician

Name: _____

Address: _____

Phone/fax: _____

SECTION 1 – REQUESTED ACTION

Consultation
(Please send the patient back for follow-up and treatment.)

- ☐ Confirm diagnosis.
- ☐ Advise as to diagnosis.
- ☐ Suggest medication or treatment.

Referral
(Please provide primary physician with summaries of subsequent visits.)

- ☐ Assume management for this particular problem and return patient after conclusion of care.
- ☐ Assume future management of patient within your area of expertise.

SECTION 2 – PATIENT INFORMATION

Name: _____

Address: _____

Phone: _____ Date of birth: _____

Tentative diagnosis: _____

Pertinent history, physical and laboratory findings, and special financial considerations:

- ☐ See additional information attached.
- ☐ Please call me when you have seen the patient.
- ☐ I would like to receive periodic status reports on this patient.
- ☐ Please send a thorough written report when the consultation is complete.

Signature: _____
PRIMARY PHYSICIAN

SECTION 3 – CONSULTANT'S FINDINGS

- ☐ I would like to receive periodic status reports on this patient.

Signature: _____
CONSULTANT

Primary physician: Complete sections 1 and 2. Send one copy to the consultant and keep one copy in the patient's chart or in a tickler file.
Consultant: Complete section 3. Return one copy to the primary physician after your initial visit with the patient. Keep one copy for your records.

Family Practice Management®

Figure 90 – AAFP Consultation / Referral Request Form

11.2 Source Codes

- **web-pcs/src/org.leaf.cms.action/BaseAction.java**

```
package org.leaf.cms.action;

import java.io.InputStream;
import java.util.Map;

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

import org.apache.struts2.interceptor.ServletRequestAware;
import org.apache.struts2.interceptor.ServletResponseAware;
import org.apache.struts2.interceptor.SessionAware;

import com.opensymphony.xwork2.ActionSupport;

public abstract class BaseAction extends ActionSupport
    implements SessionAware, ServletRequestAware, ServletResponseAware {

    private static final long serialVersionUID = 1L;
    protected HttpServletRequest request;
    protected HttpServletResponse response;
    protected Map<String, Object> session;

    protected InputStream inputStream;

    public void setServletResponse(HttpServletResponse servletResponse) {
            this.response = servletResponse;
    }

    public void setServletRequest(HttpServletRequest servletRequest) {
            this.request = servletRequest;
    }

    public void setSession(Map<String, Object> sessionMap) {
            this.session = sessionMap;
    }

    public InputStream getInputStream() {
            return inputStream;
    }

    public void setInputStream(InputStream inputStream) {
            this.inputStream = inputStream;
    }

}
```

- **web-pcs/src/org.leaf.cms.action/BaseLoggedInAction.java**

```
package org.leaf.cms.action;

import org.leaf.cms.action.interfaces.CurrentLoggedInUserAware;
import org.leaf.cms.model.User;

public abstract class BaseLoggedInAction extends BaseAction
    implements CurrentLoggedInUserAware {

    private static final long serialVersionUID = 1L;
    protected User currentLoggedInUser;

    public User getCurrentLoggedInUser()
    {
            return currentLoggedInUser;
    }

    public void putCurrentLoggedInUser(User user)
    {
            this.currentLoggedInUser = user;
    }

}
```

- **web-pcs/src/org.leaf.cms.action/BaseLoggedInActionWithPaging.java**

```
package org.leaf.cms.action;

public abstract class BaseLoggedInActionWithPaging<T> extends BaseLoggedInAction{

    private static final long serialVersionUID = 1L;

    protected int page;
    protected int rows;
    protected String[] sidx;
    protected String[] sord;

    public int getPage() {
            return page;
    }
    public void setPage(int currentPage) {
            this.page = currentPage;
    }

    public int getRows() {
            return rows;
    }
    public void setRows(int itemsPerPage) {
            this.rows = itemsPerPage;
    }
    public String[] getSidx() {
            return sidx;
    }
    public void setSidx(String[] sidx) {
            this.sidx = sidx;
    }
    public String[] getSord() {
            return sord;
    }
    public void setSord(String[] sord) {
            this.sord = sord;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.account/UserAccountAction.java**

```
package org.leaf.cms.action.account;

import org.apache.log4j.Logger;
import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.action.interfaces.ConfigurationAware;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.enums.SubscriptionDigestTypeEnum;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.model.User;
import org.leaf.cms.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;

public class UserAccountAction extends BaseLoggedInAction implements
ConfigurationAware {
    private static final Logger _logger = Logger.getLogger(UserAccountAction.class);

    private static final String SEND_WHEN_BOOKED = "sendWhenBooked";
    private static final String SEND_WHEN_CANCELLED = "sendWhenCancelled";
    private static final String SEND_WHEN_RESCHEDULED = "sendWhenRescheduled";
    private static final String SEND_WHEN_MESSAGE_SENT = "sendWhenMessageSent";
    private static final String SEND_ON_FORUM_REPLY = "sendOnForumReply";
    private static final String SEND_ON_FORUM_SUBSCRIBED = "sendOnForumSubscribed";
    private static final String SEND_SUBSCRIPTION_UPDATES =
"sendSubscriptionUpdates";
    private static final String TRIGGER_SMS_REMINDER = "triggerSmsReminder";
    private static final long serialVersionUID = 1L;
    private UserService userService;
    private User user;
    private String subscriptionDigestType;

    @Autowired
    public UserAccountAction(@Qualifier("userService") UserService userService) {
```

```
        this.userService = userService;
    }

    private Configuration configuration;

    @Override
    public Configuration getConfiguration() {
        return configuration;
    }

    @Override
    public void setConfiguration(Configuration configuration) {
        this.configuration = configuration;
    }

    @Override
    public String execute() {
        user = getCurrentLoggedInUser();
        return SUCCESS;
    }

    public String saveAccountConfiguration() {
        User user = null;
        if ((user = getCurrentLoggedInUser()) == null) {
            addActionError(ApplicationConstants.ERR_MISSING_PARAMETER);
            return ERROR;
        }
        user.setSendOnForumReply(request.getParameter(SEND_ON_FORUM_REPLY) != null);
        user.setSendOnForumSubscribed(request.getParameter(SEND_ON_FORUM_SUBSCRIBED)
!= null);

        user.setTriggerSmsReminder(request.getParameter(TRIGGER_SMS_REMINDER) !=
null);


user.setSendSubscriptionUpdates(request.getParameter(SEND_SUBSCRIPTION_UPDATES) !=
null);
        user.setSendWhenBooked(request.getParameter(SEND_WHEN_BOOKED) != null);
        user.setSendWhenCancelled(request.getParameter(SEND_WHEN_CANCELLED) != null);
        user.setSendWhenMessageSent(request.getParameter(SEND_WHEN_MESSAGE_SENT) !=
null);
        user.setSendWhenResceduled(request.getParameter(SEND_WHEN_RESCHEDULED) !=
null);


user.setSubscriptionDigestType(SubscriptionDigestTypeEnum.valueOf(subscriptionDigestT
ype));
        user = userService.saveUser(user);
        if (user == null || user.getId() == null) {
            addActionError(ApplicationConstants.ERR_FAILED_SAVE_ACCOUNT_SETTINGS);
            return ERROR;
        }
        addActionMessage(ApplicationConstants.SUC_SAVE_ACCOUNT_SETTINGS);
        return SUCCESS;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String getSubscriptionDigestType() {
        return subscriptionDigestType;
    }

    public void setSubscriptionDigestType(String subscriptionDigestType) {
        this.subscriptionDigestType = subscriptionDigestType;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.chat/ChatAction.java**

```
package org.leaf.cms.action.chat;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.model.ChatMessage;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.service.ChatService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.dto.ChatRoom;
import org.springframework.beans.factory.annotation.Autowired;

public class ChatAction extends BaseLoggedInAction {

    private static final long serialVersionUID = 1L;

    private ChatService chatService;
    private PhysicianService physicianService;
    private List<ChatRoom> chatRooms;
    private Long chatRoomId;
    private List<ChatMessage> chatRoomMessages;
    private String chatMessage;
    private int messageIndex;
    private Long clinicalCaseId;
    private int fetchedMessagesCount;
    private Long requestingPhysicianId;
    private Long consultantId;

    @Autowired
    public ChatAction(ChatService chatService, PhysicianService physicianService) {
        this.chatService = chatService;
        this.physicianService = physicianService;
    }

    public String userOngoingAppointments() {
        Physician physician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
        chatRooms = chatService.getChatRooms(physician);
        return SUCCESS;
    }

    public String chatRoom() {
        ChatRoom chatRoom = chatService.getChatroom(chatRoomId);
        requestingPhysicianId = chatRoom.getRequestingPhysicianId();
        consultantId = chatRoom.getConsultantId();
        return SUCCESS;
    }

    public String forceUpdateOfChatRooms() {
        chatService.forceUpdateChatRooms();
        return SUCCESS;
    }

    public String getUnfetchedMessages() {
        try {
            chatRoomMessages = new ArrayList<>(chatService.getChatRecords(chatRoomId,
messageIndex));
            fetchedMessagesCount = chatRoomMessages.size();
            filterChatRoomMessages();
        } catch (NullPointerException e) {
            return ERROR;
        }
        return SUCCESS;
    }

    private void filterChatRoomMessages() {
        User user = getCurrentLoggedInUser();
        if (consultantId.longValue() == user.getId()) {
            List<Integer> chatMessagesToBeFiltered = new ArrayList<>();
            int ctr = 0;
            for (ChatMessage chatMessage : chatRoomMessages) {
```

```
                Physician physician = chatMessage.getPhysician();
                if (requestingPhysicianId.longValue() != physician.getId() &&
consultantId.longValue() != physician.getId()) {
                    chatMessagesToBeFiltered.add(ctr);
                }
                ctr++;
            }
            Collections.reverse(chatMessagesToBeFiltered);
            for (int index : chatMessagesToBeFiltered) {
                chatRoomMessages.remove(index);
            }
        }
    }

    public String sendMessage() {
        Physician physician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
        chatService.recordChat(chatRoomId, physician, chatMessage);
        return NONE;
    }

    public String terminateAppointment() {
        chatService.endChatRoom(chatRoomId, false);
        return SUCCESS;
    }

    public String terminateAppointmentAndCloseClinicalCase() {
        chatService.endChatRoom(chatRoomId, true);
        return SUCCESS;
    }

    public String chatRoomMessageHistory() {
        chatRoomMessages = new
ArrayList<>(chatService.getChatRecordsHistory(clinicalCaseId));
        if (requestingPhysicianId != null && consultantId != null) {
            filterChatRoomMessages();
        }
        return SUCCESS;
    }

    public List<ChatRoom> getChatRooms() {
        return chatRooms;
    }

    public void setChatRooms(List<ChatRoom> chatRooms) {
        this.chatRooms = chatRooms;
    }

    public Long getChatRoomId() {
        return chatRoomId;
    }

    public void setChatRoomId(Long chatRoomId) {
        this.chatRoomId = chatRoomId;
    }

    public List<ChatMessage> getChatRoomMessages() {
        return chatRoomMessages;
    }

    public void setChatRoomMessages(List<ChatMessage> chatRoomMessages) {
        this.chatRoomMessages = chatRoomMessages;
    }

    public String getChatMessage() {
        return chatMessage;
    }

    public void setChatMessage(String chatMessage) {
        this.chatMessage = chatMessage;
    }

    public int getMessageIndex() {
        return messageIndex;
    }
```

```
    public void setMessageIndex(int messageIndex) {
        this.messageIndex = messageIndex;
    }

    public Long getClinicalCaseId() {
        return clinicalCaseId;
    }

    public void setClinicalCaseId(Long clinicalCaseId) {
        this.clinicalCaseId = clinicalCaseId;
    }

    public int getFetchedMessagesCount() {
        return fetchedMessagesCount;
    }

    public void setFetchedMessagesCount(int fetchedMessagesCount) {
        this.fetchedMessagesCount = fetchedMessagesCount;
    }

    public long getRequestingPhysicianId() {
        return requestingPhysicianId;
    }

    public void setRequestingPhysicianId(long requestingPhysicianId) {
        this.requestingPhysicianId = requestingPhysicianId;
    }

    public long getConsultantId() {
        return consultantId;
    }

    public void setConsultantId(long consultantId) {
        this.consultantId = consultantId;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.configuration/ConfigurationAction.java**

```
package org.leaf.cms.action.configuration;

import org.apache.log4j.Logger;
import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.action.interfaces.ConfigurationAware;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.service.ConfigurationService;
import org.springframework.beans.factory.annotation.Autowired;

public class ConfigurationAction extends BaseLoggedInAction implements
ConfigurationAware {
    private static final Logger _logger =
Logger.getLogger(ConfigurationAction.class);

    private static final long serialVersionUID = 1L;
    private Configuration configuration;

    private String connectionString;
    private String username;
    private String password;
    private String smsTemplate;

    private ConfigurationService configurationService;

    @Autowired
    public ConfigurationAction(ConfigurationService configurationService) {
        this.configurationService = configurationService;
    }

    @Override
    public String execute() {
        _logger.info("Method Executed: " + (configuration == null));
        return SUCCESS;
    }

    public String save() {
```

```
            _logger.info("Saving Configuration...");
            Configuration configuration = getConfiguration();
            configuration.setIsSMSEnabled((request.getParameter("smsEnabled") != null));
            configuration.setSmsTemplate(smsTemplate);
            configurationService.saveConfiguration(configuration);
            return SUCCESS;
        }

        @Override
        public Configuration getConfiguration() {
            return configuration;
        }

        @Override
        public void setConfiguration(Configuration configuration) {
            this.configuration = configuration;
        }

        public String getConnectionString() {
            return connectionString;
        }

        public void setConnectionString(String connectionString) {
            this.connectionString = connectionString;
        }

        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;
        }

        public String getSmsTemplate() {
                return smsTemplate;
        }

        public void setSmsTemplate(String smsTemplate) {
                this.smsTemplate = smsTemplate;
        }

    }
```

- **web-pcs/src/org.leaf.cms.action.consultation/ConsultationAction.java**

```
    package org.leaf.cms.action.consultation;

    import java.io.File;
    import java.util.ArrayList;
    import java.util.HashMap;
    import java.util.List;
    import java.util.Map;

    import org.joda.time.DateTime;
    import org.leaf.cms.action.BaseLoggedInAction;
    import org.leaf.cms.action.planner.MessageStore;
    import org.leaf.cms.constants.ApplicationConstants;
    import org.leaf.cms.enums.ClinicalCaseStatusEnum;
    import org.leaf.cms.model.Appointment;
    import org.leaf.cms.model.ClinicalCase;
    import org.leaf.cms.model.Configuration;
    import org.leaf.cms.model.FileManagement;
    import org.leaf.cms.model.Physician;
    import org.leaf.cms.model.Specialty;
    import org.leaf.cms.model.User;
    import org.leaf.cms.service.AppointmentService;
    import org.leaf.cms.service.ChatService;
```

```
import org.leaf.cms.service.ClinicalCaseService;
import org.leaf.cms.service.ConfigurationService;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.FileManagementService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.SMSEntryService;
import org.leaf.cms.service.SpecialtyService;
import org.leaf.cms.service.dto.ChatRoom;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;

import com.dhtmlx.planner.DHXPlanner;
import com.dhtmlx.planner.DHXSkin;
import com.dhtmlx.planner.controls.DHXLightboxTime;
import com.dhtmlx.planner.data.DHXDataFormat;
import com.dhtmlx.planner.data.DHXDataLoader.DHXDynLoadingMode;
import com.dhtmlx.planner.extensions.DHXExtension;

public class ConsultationAction extends BaseLoggedInAction {

    private static final long serialVersionUID = 1L;
    private SpecialtyService specialtyService;
    private AppointmentService appointmentService;
    private PhysicianService physicianService;
    private ClinicalCaseService clinicalCaseService;
    private FileManagementService fileManagementService;
    private EmailService emailService;
    private ConfigurationService configurationService;
    private SMSEntryService smsEntryService;
    private ChatService chatService;
    private List<Specialty> specialtyList;
    private Long specialtyId;
    private int physicianPage;
    private Page<Physician> physiciansOfSpecificSpecialty;
    private Long consultantId;
    private Long requestingPhysicianId;
    private MessageStore messageStore = new MessageStore();
    private List<ClinicalCase> clinicalCaseAvailable = new ArrayList<>();
    private Long clinicalCaseId;
    private Long startDate;
    private Long endDate;
    private String specialtyDescription;
    private String consultationDateString;
    private String requestingPhysicianName;
    private String consultantName;
    private String question;
    private String patientLabExam;
    private String patientDiagnosis;
    private String remarks;
    private List<FileManagement> clinicalCaseFiles = new ArrayList<>();
    private List<Long> fileIdsForDeletion = new ArrayList<>();
    private List<File> upload = new ArrayList<>();
    private List<String> uploadFileName = new ArrayList<>();
    private List<String> uploadContentType = new ArrayList<>();
    private List<ClinicalCase> userCreatedClinicalCases = new ArrayList<>();
    private List<String> consultantNames = new ArrayList<>();
    private List<ClinicalCase> clinicalCasesAsConsultant = new ArrayList<>();
    private List<String> requestingPhysicianNames = new ArrayList<>();
    private String status;
    private Long chatRoomId;
    private Long appointmentId;
    private int notificationSmsTime;

    @Autowired
    public ConsultationAction(
                            SpecialtyService specialtyService,
                            AppointmentService appointmentService,
                            PhysicianService physicianService,
                            ClinicalCaseService clinicalCaseService,
                            FileManagementService fileManagementService,
                            EmailService emailService,
                            ConfigurationService configurationService,
                            SMSEntryService smsEntryService,
                            ChatService chatService) {
        this.specialtyService = specialtyService;
        this.appointmentService = appointmentService;
```

```
        this.physicianService = physicianService;
        this.clinicalCaseService = clinicalCaseService;
        this.fileManagementService = fileManagementService;
        this.emailService = emailService;
        this.configurationService = configurationService;
        this.smsEntryService = smsEntryService;
        this.chatService = chatService;
    }

    public String createAppointment() throws Exception {
        specialtyList = specialtyService.getSpecialty();
        return SUCCESS;
    }

    public String loadPhysicians() {
        Specialty specialty = specialtyService.findSpecialty(specialtyId);
        setPhysiciansOfSpecificSpecialty(physicianService.getPhysicians(specialty,
physicianPage, ApplicationConstants.DEFAULT_ITEMS_PER_PAGE));
        return SUCCESS;
    }

    public String initializeConsultationPlanner() throws Exception {
        DHXPlanner planner = new DHXPlanner("./codebase-modified/", DHXSkin.GLOSSY);

        planner.lightbox.clear();

        DHXLightboxTime time = new DHXLightboxTime("time", "Time");
        planner.lightbox.add(time);

        planner.config.setDetailsOnCreate(true);
        planner.config.setDragCreate(false);
        planner.config.setDragLightbox(false);
        planner.config.setDragMove(false);
        planner.config.setDragResize(false);
        planner.config.setWideForm(false);
        planner.config.setScrollHour(8);
        planner.config.setEventDuration(30);
        planner.config.setShowLoading(true);
        planner.config.setSeparateShortEvents(true);
        planner.extensions.add(DHXExtension.QUICK_INFO);
        planner.extensions.add(DHXExtension.READONLY);
        planner.extensions.add(DHXExtension.COLLISION);
        planner.extensions.add(DHXExtension.CONTAINER_AUTORESIZE);

        planner.data.enableDynamicLoading(DHXDynLoadingMode.week);
        planner.data.dataprocessor.setURL("consultations-appointment-
consultantEvents.do?consultantId=" + consultantId);
        planner.load("consultations-appointment-consultantEvents.do?consultantId=" +
consultantId, DHXDataFormat.JSON);
        messageStore.setPlanner(planner.render());

        return SUCCESS;
    }

    public String consultantEvents() throws Exception {
        Physician requestingPhysician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
        Physician consultant = physicianService.findPhysician(consultantId);
        ConsultationEventsManager events = new ConsultationEventsManager(request,
appointmentService, requestingPhysician, consultant);
        messageStore.setData(events.run());
        return SUCCESS;
    }

    public String selectClinicalCase() {
        Specialty specialty = specialtyService.findSpecialty(specialtyId);
        clinicalCaseAvailable =
clinicalCaseService.getClinicalCasesThatCanHaveAnAppointment(getCurrentLoggedInUser()
, specialty);
        return SUCCESS;
    }

    public String finalizeConsultation() {
        Specialty specialtyData = specialtyService.findSpecialty(specialtyId);
        specialtyDescription = specialtyData.getDescription();
```

```
        DateTime start = new DateTime(startDate);
        DateTime end = new DateTime(endDate);
        consultationDateString = consultationDateStringBuilder(start, end);

        Physician requestingPhysician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
        requestingPhysicianName = requestingPhysician.getFullName();

        Physician consultant = physicianService.findPhysician(consultantId);
        consultantName = consultant.getFullName();

        if (clinicalCaseId != -1) {
            ClinicalCase clinicalCase =
clinicalCaseService.getClinicalCase(clinicalCaseId);
            question = clinicalCase.getQuestion();
            patientLabExam = clinicalCase.getPatientLabExam();
            patientDiagnosis = clinicalCase.getPatientDiagnosis();
            remarks = clinicalCase.getRemarks();
            clinicalCaseFiles = fileManagementService.getFiles(clinicalCase);
        }

        return SUCCESS;
    }

    private String consultationDateStringBuilder(DateTime start, DateTime end) {
        StringBuilder stringBuilder = new StringBuilder();
        if (start.getDayOfYear() == end.getDayOfYear()) {
            stringBuilder.append(start.dayOfWeek().getAsText())
                         .append(", ")
                         .append(start.dayOfMonth().getAsText())
                         .append(' ')
                         .append(start.monthOfYear().getAsText())
                         .append(' ')
                         .append(start.year().getAsText())
                         .append(' ')
                         .append(start.hourOfDay().getAsText())
                         .append(':')
                         .append(start.minuteOfHour().getAsText())
                         .append(" - ")
                         .append(end.hourOfDay().getAsText())
                         .append(':')
                         .append(end.minuteOfHour().getAsText());
        } else {
            stringBuilder.append(start.dayOfMonth().getAsText())
                         .append(' ')
                         .append(start.monthOfYear().getAsText())
                         .append(' ')
                         .append(start.year().getAsText())
                         .append(' ')
                         .append(start.hourOfDay().getAsText())
                         .append(':')
                         .append(start.minuteOfHour().getAsText())
                         .append(" - ")
                         .append(end.dayOfMonth().getAsText())
                         .append(' ')
                         .append(end.monthOfYear().getAsText())
                         .append(' ')
                         .append(end.year().getAsText())
                         .append(' ')
                         .append(end.hourOfDay().getAsText())
                         .append(':')
                         .append(end.minuteOfHour().getAsText());
        }
        return stringBuilder.toString();
    }

    public String saveConsultation() {
        ClinicalCase clinicalCase = null;
        if (clinicalCaseId == -1) {
            clinicalCase = new ClinicalCase();
            clinicalCase.setCreatedBy(currentLoggedInUser);
            Specialty specialty = specialtyService.findSpecialty(specialtyId);
            clinicalCase.setSpecialty(specialty);
        } else {
            clinicalCase = clinicalCaseService.getClinicalCase(clinicalCaseId);
        }
```

```
        clinicalCase.setQuestion(question);
        clinicalCase.setPatientDiagnosis(patientDiagnosis);
        clinicalCase.setPatientLabExam(patientLabExam);
        clinicalCase.setRemarks(remarks);

        Appointment appointment = new Appointment();
        appointment.setClinicalCase(clinicalCase);

        Physician requestingPhysician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
        appointment.setRequestingPhysician(requestingPhysician);

        Physician consultant = physicianService.findPhysician(consultantId);
        appointment.setConsultant(consultant);

        DateTime start = new DateTime(startDate);
        appointment.setConsultationDate(start.toLocalDate());
        appointment.setConsultationStart(start);

        DateTime end = new DateTime(endDate);
        appointment.setConsultationEnd(end);

        if (clinicalCaseId == -1) {
            appointment = appointmentService.createAppointment(appointment, true);
        } else {
            appointment = appointmentService.createAppointment(appointment, false);
        }
        appointmentId = appointment.getId();

        List<Long> filteredFileIdsForDeletion = new ArrayList<>();
        for (Long fileId : fileIdsForDeletion) {
            if (fileId != null) {
                filteredFileIdsForDeletion.add(fileId);
            }
        }
        fileManagementService.deleteFiles(filteredFileIdsForDeletion);
        fileManagementService.uploadFiles(upload, uploadFileName, uploadContentType,
appointment.getClinicalCase(), getCurrentLoggedInUser());

        consultationDateString = consultationDateStringBuilder(start, end);
        requestingPhysicianName = requestingPhysician.getFullName();
        consultantName = consultant.getFullName();

        if (requestingPhysician.getSendWhenBooked()) {
            Map<String, Object> requestingPhysicianHashMap = new HashMap<>();
            requestingPhysicianHashMap.put("name", requestingPhysicianName);
            requestingPhysicianHashMap.put("consultationDateString",
consultationDateString);
            requestingPhysicianHashMap.put("requestingPhysicianName",
requestingPhysicianName);
            requestingPhysicianHashMap.put("consultantName", consultantName);
            emailService.sendEmail(
                                "newappointment.html",
                                "Web PCS New Appointment",
                                requestingPhysician.getEmail(),
                                requestingPhysicianHashMap,
                                null,
                                null);
        }

        if (consultant.getSendWhenBooked()) {
            Map<String, Object> consultantHashMap = new HashMap<>();
            consultantHashMap.put("name", consultantName);
            consultantHashMap.put("consultationDateString", consultationDateString);
            consultantHashMap.put("requestingPhysicianName",
requestingPhysicianName);
            consultantHashMap.put("consultantName", consultantName);
            emailService.sendEmail("newappointment.html", "Web PCS New Appointment",
consultant.getEmail(), consultantHashMap, null, null);
        }

        Configuration configuration = configurationService.mainConfiguration();
        if (configuration.getIsSMSEnabled()) {
            Map<String, Object> requestingPhysicianSmsHashMap = new HashMap<>();
            requestingPhysicianSmsHashMap.put("name", requestingPhysicianName);
```

```
            requestingPhysicianSmsHashMap.put("consultationDateString",
consultationDateString);
            requestingPhysicianSmsHashMap.put("otherName", consultantName);
            smsEntryService.saveMessageToBeSent("newappointment_sms.vm",
requestingPhysician.getCellphoneNumber(), requestingPhysicianSmsHashMap);

            Map<String, Object> consultantSmsHashMap = new HashMap<>();
            consultantSmsHashMap.put("name", consultantName);
            consultantSmsHashMap.put("consultationDateString",
consultationDateString);
            consultantSmsHashMap.put("otherName", requestingPhysicianName);
            smsEntryService.saveMessageToBeSent("newappointment_sms.vm",
consultant.getCellphoneNumber(), consultantSmsHashMap);
        }

        return SUCCESS;
    }

    public String viewClinicalCases() {

setUserCreatedClinicalCases(clinicalCaseService.getClinicalCases(getCurrentLoggedInUs
er()));

        Physician consultant =
physicianService.findPhysician(getCurrentLoggedInUser().getId());

setClinicalCasesAsConsultant(clinicalCaseService.getClinicalCasesOfAnAppointmentConsu
ltant(consultant));

        for (ClinicalCase clinicalCase : userCreatedClinicalCases) {
            if
(ClinicalCaseStatusEnum.ACTIVE.getStatus().equalsIgnoreCase(clinicalCase.getStatus())
                    ||
ClinicalCaseStatusEnum.ONGOING.getStatus().equalsIgnoreCase(clinicalCase.getStatus())
) {
                Appointment appointment =
appointmentService.findAppointment(clinicalCase);
                Physician physician = appointment.getConsultant();
                consultantNames.add(physician.getFullName());
            } else {

                 consultantNames.add("");
            }
        }

        for (ClinicalCase clinicalCase : clinicalCasesAsConsultant) {
            if
(ClinicalCaseStatusEnum.ACTIVE.getStatus().equalsIgnoreCase(clinicalCase.getStatus())
                    ||
ClinicalCaseStatusEnum.ONGOING.getStatus().equalsIgnoreCase(clinicalCase.getStatus())
) {
                Appointment appointment =
appointmentService.findAppointment(clinicalCase);
                Physician physician = appointment.getRequestingPhysician();
                requestingPhysicianNames.add(physician.getFullName());
            } else {

                requestingPhysicianNames.add("");
            }
        }

        return SUCCESS;
    }

    public String viewClinicalCaseDetails() {
        ClinicalCase clinicalCase =
clinicalCaseService.getClinicalCase(clinicalCaseId);
        question = clinicalCase.getQuestion();
        patientLabExam = clinicalCase.getPatientLabExam();
        patientDiagnosis = clinicalCase.getPatientDiagnosis();
        remarks = clinicalCase.getRemarks();
        status = clinicalCase.getStatus();
        clinicalCaseFiles = fileManagementService.getFiles(clinicalCase);

        if (ClinicalCaseStatusEnum.ACTIVE.getStatus().equalsIgnoreCase(getStatus())
```

```
                ||
ClinicalCaseStatusEnum.ONGOING.getStatus().equalsIgnoreCase(getStatus())) {
            Appointment appointment =
appointmentService.findAppointment(clinicalCase);

            Specialty specialty = clinicalCase.getSpecialty();
            specialtyDescription = specialty.getDescription();

            DateTime start = appointment.getConsultationStart();
            DateTime end = appointment.getConsultationEnd();
            consultationDateString = consultationDateStringBuilder(start, end);

            Physician requestingPhysician = appointment.getRequestingPhysician();
            requestingPhysicianName = requestingPhysician.getFullName();

            Physician consultant = appointment.getConsultant();
            consultantName = consultant.getFullName();

            requestingPhysicianId = requestingPhysician.getId();
            consultantId = consultant.getId();

            User currentLoggedInUser = getCurrentLoggedInUser();
            if (currentLoggedInUser.getId() == requestingPhysician.getId()) {
                if (!appointment.getIsSeenRequestingPhysician()) {

appointmentService.markAppointmentAsSeenByRequestingPhyisician(appointment);
                }
            } else if (currentLoggedInUser.getId() == consultant.getId()) {
                if (!appointment.getIsSeenConsultant()) {

appointmentService.markAppointmentAsSeenByConsultant(appointment);
                }
            }
        }

        if (ClinicalCaseStatusEnum.ONGOING.getStatus().equalsIgnoreCase(getStatus()))
{
            ChatRoom chatRoom = chatService.getChatroom(clinicalCaseId);
            chatRoomId = chatRoom.getId();
        }

        return SUCCESS;
    }

    public String updateClinicalCaseDetails() {
        ClinicalCase clinicalCase =
clinicalCaseService.getClinicalCase(clinicalCaseId);
        clinicalCase.setQuestion(question);
        clinicalCase.setPatientDiagnosis(patientDiagnosis);
        clinicalCase.setPatientLabExam(patientLabExam);
        clinicalCase.setRemarks(remarks);

        List<Long> filteredFileIdsForDeletion = new ArrayList<>();
        for (Long fileId : fileIdsForDeletion) {
            if (fileId != null) {
                filteredFileIdsForDeletion.add(fileId);
            }
        }
        fileManagementService.deleteFiles(filteredFileIdsForDeletion);
        fileManagementService.uploadFiles(upload, uploadFileName, uploadContentType,
clinicalCase, getCurrentLoggedInUser());

        return SUCCESS;
    }

    public String closeClinicalCase() {
        clinicalCaseService.closeClinicalCase(clinicalCaseId);
        return SUCCESS;
    }

    public String saveSmsNotificationSettings() {
        appointmentService.setAppointmentNotificationSchedule(appointmentId,
notificationSmsTime);
        return NONE;
    }
```

```
    public List<Specialty> getSpecialtyList() {
        return specialtyList;
    }

    public void setSpecialtyList(List<Specialty> specialtyList) {
        this.specialtyList = specialtyList;
    }

    public Long getSpecialtyId() {
        return specialtyId;
    }

    public void setSpecialtyId(Long specialtyId) {
        this.specialtyId = specialtyId;
    }

    public int getPhysicianPage() {
        return physicianPage;
    }

    public void setPhysicianPage(int physicianPage) {
        this.physicianPage = physicianPage;
    }

    public Page<Physician> getPhysiciansOfSpecificSpecialty() {
        return physiciansOfSpecificSpecialty;
    }

    public void setPhysiciansOfSpecificSpecialty(Page<Physician>
physiciansOfSpecificSpecialty) {
        this.physiciansOfSpecificSpecialty = physiciansOfSpecificSpecialty;
    }

    public Long getConsultantId() {
        return consultantId;
    }

    public void setConsultantId(Long consultantId) {
        this.consultantId = consultantId;
    }

    public MessageStore getMessageStore() {
        return messageStore;
    }

    public void setMessageStore(MessageStore messageStore) {
        this.messageStore = messageStore;
    }

    public List<ClinicalCase> getClinicalCaseAvailable() {
        return clinicalCaseAvailable;
    }

    public void setClinicalCaseAvailable(List<ClinicalCase> clinicalCaseAvailable) {
        this.clinicalCaseAvailable = clinicalCaseAvailable;
    }

    public Long getClinicalCaseId() {
        return clinicalCaseId;
    }

    public void setClinicalCaseId(Long clinicalCaseId) {
        this.clinicalCaseId = clinicalCaseId;
    }

    public String getSpecialtyDescription() {
        return specialtyDescription;
    }

    public void setSpecialtyDescription(String specialtyDescription) {
        this.specialtyDescription = specialtyDescription;
    }

    public String getConsultationDateString() {
        return consultationDateString;
    }
```

```
public void setConsultationDateString(String consultationDateString) {
    this.consultationDateString = consultationDateString;
}

public String getRequestingPhysicianName() {
    return requestingPhysicianName;
}

public void setRequestingPhysicianName(String requestingPhysicianName) {
    this.requestingPhysicianName = requestingPhysicianName;
}

public String getConsultantName() {
    return consultantName;
}

public void setConsultantName(String consultantName) {
    this.consultantName = consultantName;
}

public Long getStartDate() {
    return startDate;
}

public void setStartDate(Long startDate) {
    this.startDate = startDate;
}

public Long getEndDate() {
    return endDate;
}

public void setEndDate(Long endDate) {
    this.endDate = endDate;
}

public String getQuestion() {
    return question;
}

public void setQuestion(String question) {
    this.question = question;
}

public String getPatientLabExam() {
    return patientLabExam;
}

public void setPatientLabExam(String patientLabExam) {
    this.patientLabExam = patientLabExam;
}

public String getPatientDiagnosis() {
    return patientDiagnosis;
}

public void setPatientDiagnosis(String patientDiagnosis) {
    this.patientDiagnosis = patientDiagnosis;
}

public String getRemarks() {
    return remarks;
}

public void setRemarks(String remarks) {
    this.remarks = remarks;
}

public List<FileManagement> getClinicalCaseFiles() {
    return clinicalCaseFiles;
}

public void setClinicalCaseFiles(List<FileManagement> clinicalCaseFiles) {
    this.clinicalCaseFiles = clinicalCaseFiles;
}
```

```
    public List<Long> getFileIdsForDeletion() {
        return fileIdsForDeletion;
    }

    public void setFileIdsForDeletion(List<Long> fileIdsForDeletion) {
        this.fileIdsForDeletion = fileIdsForDeletion;
    }

    public List<File> getUpload() {
        return upload;
    }

    public void setUpload(List<File> upload) {
        this.upload = upload;
    }

    public List<String> getUploadFileName() {
        return uploadFileName;
    }

    public void setUploadFileName(List<String> uploadFileName) {
        this.uploadFileName = uploadFileName;
    }

    public List<String> getUploadContentType() {
        return uploadContentType;
    }

    public void setUploadContentType(List<String> uploadContentType) {
        this.uploadContentType = uploadContentType;
    }

    public List<ClinicalCase> getUserCreatedClinicalCases() {
        return userCreatedClinicalCases;
    }

    public void setUserCreatedClinicalCases(List<ClinicalCase>
userCreatedClinicalCases) {
        this.userCreatedClinicalCases = userCreatedClinicalCases;
    }

    public List<ClinicalCase> getClinicalCasesAsConsultant() {
        return clinicalCasesAsConsultant;
    }

    public void setClinicalCasesAsConsultant(List<ClinicalCase>
clinicalCasesAsConsultant) {
        this.clinicalCasesAsConsultant = clinicalCasesAsConsultant;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public Long getChatRoomId() {
        return chatRoomId;
    }

    public void setChatRoomId(Long chatRoomId) {
        this.chatRoomId = chatRoomId;
    }

    public int getNotificationSmsTime() {
        return notificationSmsTime;
    }

    public void setNotificationSmsTime(int notificationSmsTime) {
        this.notificationSmsTime = notificationSmsTime;
    }

    public Long getAppointmentId() {
```

```
        return appointmentId;
    }

    public void setAppointmentId(Long appointmentId) {
        this.appointmentId = appointmentId;
    }

    public List<String> getConsultantNames() {
        return consultantNames;
    }

    public void setConsultantNames(List<String> consultantNames) {
        this.consultantNames = consultantNames;
    }

    public List<String> getRequestingPhysicianNames() {
        return requestingPhysicianNames;
    }

    public void setRequestingPhysicianNames(List<String> requestingPhysicianNames) {
        this.requestingPhysicianNames = requestingPhysicianNames;
    }

    public Long getRequestingPhysicianId() {
        return requestingPhysicianId;
    }

    public void setRequestingPhysicianId(Long requestingPhysicianId) {
        this.requestingPhysicianId = requestingPhysicianId;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.consultation/ConsultationEventsManager.java**

```
package org.leaf.cms.action.consultation;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.joda.time.LocalDate;
import org.leaf.cms.action.planner.AppointmentEvent;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.Physician;
import org.leaf.cms.service.AppointmentService;

import com.dhtmlx.planner.DHXEv;
import com.dhtmlx.planner.DHXEventsManager;
import com.dhtmlx.planner.DHXStatus;

public class ConsultationEventsManager extends DHXEventsManager {

    private AppointmentService appointmentService;
    private Physician requestingPhysician;
    private Physician consultant;

    public ConsultationEventsManager(HttpServletRequest request) {
        this(request, null, null, null);
    }

    public ConsultationEventsManager(
                                    HttpServletRequest request,
                                    AppointmentService appointmentService,
                                    Physician requestingPhysician,
                                    Physician consultant) {
        super(request);
        this.appointmentService = appointmentService;
        this.requestingPhysician = requestingPhysician;
        this.consultant = consultant;
    }

    @Override
    public Iterable<DHXEv> getEvents() {
```

```
        Date from = getFilterFrom();
        Date to = getFilterTo();
        List<Appointment> appointments;
        if (from == null || to == null) {
            appointments = appointmentService.getAppointments(requestingPhysician,
consultant);
        } else {
            LocalDate start = new LocalDate(from);
            LocalDate end = new LocalDate(to);
            appointments = appointmentService.getAppointments(requestingPhysician,
consultant, start, end);
        }

        List<DHXEv> events = new ArrayList<>();
        for (Appointment appointment : appointments) {
            AppointmentEvent event = new AppointmentEvent();
            event.setId(appointment.getId().intValue());
            event.setStart_date(appointment.getConsultationStart().toDate());
            event.setEnd_date(appointment.getConsultationEnd().toDate());

            StringBuilder text = new StringBuilder();
            text.append(appointment.getRequestingPhysician().getFullName())
                .append("'s appointment with ")
                .append(appointment.getConsultant().getFullName());
            event.setText(text.toString());

            event.setReadonly(Boolean.TRUE);
            events.add(event);
        }
        return events;
    }

    @Override
    public DHXStatus saveEvent(DHXEv event, DHXStatus status) {
        if (DHXStatus.DELETE.compareTo(status) == 0) {
            return DHXStatus.ERROR;
        }
        return status;
    }

    @Override
    public DHXEv createEvent(String id, DHXStatus status) {
        AppointmentEvent newEvent = new AppointmentEvent();
        return newEvent;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.consultation/ConsultationFileDownloadAction.java**

```
package org.leaf.cms.action.consultation;

import java.io.File;

import org.apache.commons.io.FileUtils;
import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.model.FileManagement;
import org.leaf.cms.service.FileManagementService;
import org.springframework.beans.factory.annotation.Autowired;

public class ConsultationFileDownloadAction extends BaseLoggedInAction {

    private static final long serialVersionUID = 1L;

    private FileManagementService fileManagementService;
    private Long fileId;
    private String contentType;
    private String fileName;

    @Autowired
    public ConsultationFileDownloadAction(FileManagementService
fileManagementService) {
        this.fileManagementService = fileManagementService;
    }
```

```
    @Override
    public String execute() throws Exception {
        FileManagement fileManagement = fileManagementService.getFile(fileId);
        contentType = fileManagement.getContentType();
        fileName = fileManagement.getName();
        File fileToDownload = fileManagement.getSavedFile();
        if (!fileToDownload.exists()) {
            return ERROR;
        }
        inputStream = FileUtils.openInputStream(fileToDownload);
        return SUCCESS;
    }

    public Long getFileId() {
        return fileId;
    }

    public void setFileId(Long fileId) {
        this.fileId = fileId;
    }

    public String getContentType() {
        return contentType;
    }

    public void setContentType(String contentType) {
        this.contentType = contentType;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.dashboard/DashboardAction.java**

```
package org.leaf.cms.action.dashboard;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.joda.time.DateTime;
import org.joda.time.LocalDate;
import org.json.simple.JSONObject;
import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.action.interfaces.ConfigurationAware;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.DiscussionThreadService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.UserService;
import org.leaf.cms.service.dto.ConsultationTrends;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.domain.Page;

public class DashboardAction extends BaseLoggedInAction implements ConfigurationAware
{

    private static final Logger _logger = Logger.getLogger(DashboardAction.class);
    private static final long serialVersionUID = 1L;
    private Configuration configuration;
    private UserService userService;
    private AppointmentService appointmentService;
```

```
    private DiscussionThreadService discussionThreadService;
    private PhysicianService physicianService;

    private Page<DiscussionThread> discussionThreadPage;
    private Page<Appointment> appointmentPage;
    private Page<Physician> physicianPage;
    private List<String> appointmentShortDetails;

    private String consultationTrends;

    private Long newThreadCount;
    private Long newAppointmentCount;

    @Autowired
    public DashboardAction(
      @Qualifier("userService") UserService userService,
      @Qualifier("discussionService") DiscussionThreadService
discussionThreadService,
      @Qualifier("appointmentService") AppointmentService appointmentService,
      @Qualifier("physicianService") PhysicianService physicianService) {
        this.userService = userService;
        this.discussionThreadService = discussionThreadService;
        this.appointmentService = appointmentService;
        this.physicianService = physicianService;
    }

    @Override
    public String execute() {
        return SUCCESS;
    }

    @Override
    public Configuration getConfiguration() {
        return configuration;
    }

    @Override
    public void setConfiguration(Configuration configuration) {
        this.configuration = configuration;
    }

    public String getDiscussionThread() {
        Integer page = Integer.parseInt(request.getParameter("page"));
        setDiscussionThreadPage(discussionThreadService.findAllWithPaging(page ==
null ? 1 : page, ApplicationConstants.DEFAULT_ITEMS_PER_PAGE));
        return SUCCESS;
    }

    public String getConsultations() {
        Integer page = Integer.parseInt(request.getParameter("page"));
        User currentLoggedInUser = getCurrentLoggedInUser();
        appointmentPage = appointmentService.findAppointment(
                                                            currentLoggedInUser,
                                                            page == null ? 1 : page,

ApplicationConstants.DEFAULT_ITEMS_PER_PAGE);
        List<Appointment> content = appointmentPage.getContent();
        appointmentShortDetails = new ArrayList<>();
        for (Appointment appointment : content) {
            StringBuilder stringBuilder = new StringBuilder();
            LocalDate consultationDate = appointment.getConsultationDate();
            DateTime consultationStart = appointment.getConsultationStart();
            DateTime consultationEnd = appointment.getConsultationEnd();
            stringBuilder.append(consultationDate.toString("dd MMM yyyy"))
                         .append(' ')
                         .append(consultationStart.toString("HH:mm"))
                         .append(" - ")
                         .append(consultationEnd.toString("HH:mm"))
                         .append(" Appointment with ");
            Physician requestingPhysician = appointment.getRequestingPhysician();
            Physician consultant = appointment.getConsultant();
            if (currentLoggedInUser.getId() != requestingPhysician.getId()) {
                stringBuilder.append(requestingPhysician.getFullName());
            } else if (currentLoggedInUser.getId() != consultant.getId()) {
                stringBuilder.append(consultant.getFullName());
            }
```

```
            appointmentShortDetails.add(stringBuilder.toString());
        }
        return SUCCESS;
    }

    public String getNotifications() {
        setNewThreadCount(discussionThreadService.countByIsSeenFalse());
        _logger.info("Discussion Threads: " + getNewThreadCount());

setNewAppointmentCount(appointmentService.countUnseenAppointments(getCurrentLoggedInU
ser()));
        _logger.info("Appointments: " + getNewThreadCount());
        return SUCCESS;
    }

    public String getPhysicians() {
        Integer page = Integer.parseInt(request.getParameter("page"));
        setPhysicianPage(physicianService.findAllActivatedWithPaging(page == null ? 1
: page, ApplicationConstants.DEFAULT_ITEMS_PER_PAGE));
        return SUCCESS;
    }

    @SuppressWarnings("unchecked")
    public String consultationTrends() {
        User currentLoggedInUser = getCurrentLoggedInUser();
        Map<String, ConsultationTrends> consultationTrendsMap = null;
        if
(currentLoggedInUser.getUserType().equalsIgnoreCase(User.class.getSimpleName())) {
            consultationTrendsMap = appointmentService.getConsultationTrends();
        } else {
            consultationTrendsMap =
appointmentService.getConsultationTrends(currentLoggedInUser);
        }
        JSONObject jsonConsultationTrends = new JSONObject();

        ConsultationTrends lastWeekConsultationTrends =
consultationTrendsMap.get("lastWeek");
        jsonConsultationTrends.put("monday1",
lastWeekConsultationTrends.getMonday());
        jsonConsultationTrends.put("tuesday1",
lastWeekConsultationTrends.getTuesday());
        jsonConsultationTrends.put("wednesday1",
lastWeekConsultationTrends.getWednesday());
        jsonConsultationTrends.put("thursday1",
lastWeekConsultationTrends.getThursday());
        jsonConsultationTrends.put("friday1",
lastWeekConsultationTrends.getFriday());
        jsonConsultationTrends.put("saturday1",
lastWeekConsultationTrends.getSaturday());
        jsonConsultationTrends.put("sunday1",
lastWeekConsultationTrends.getSunday());

        ConsultationTrends thisWeekConsultationTrends =
consultationTrendsMap.get("thisWeek");
        jsonConsultationTrends.put("monday2",
thisWeekConsultationTrends.getMonday());
        jsonConsultationTrends.put("tuesday2",
thisWeekConsultationTrends.getTuesday());
        jsonConsultationTrends.put("wednesday2",
thisWeekConsultationTrends.getWednesday());
        jsonConsultationTrends.put("thursday2",
thisWeekConsultationTrends.getThursday());
        jsonConsultationTrends.put("friday2",
thisWeekConsultationTrends.getFriday());
        jsonConsultationTrends.put("saturday2",
thisWeekConsultationTrends.getSaturday());
        jsonConsultationTrends.put("sunday2",
thisWeekConsultationTrends.getSunday());

        ConsultationTrends nextWeekConsultationTrends =
consultationTrendsMap.get("nextWeek");
        jsonConsultationTrends.put("monday3",
nextWeekConsultationTrends.getMonday());
        jsonConsultationTrends.put("tuesday3",
nextWeekConsultationTrends.getTuesday());
```

```
        jsonConsultationTrends.put("wednesday3",
nextWeekConsultationTrends.getWednesday());
        jsonConsultationTrends.put("thursday3",
nextWeekConsultationTrends.getThursday());
        jsonConsultationTrends.put("friday3",
nextWeekConsultationTrends.getFriday());
        jsonConsultationTrends.put("saturday3",
nextWeekConsultationTrends.getSaturday());
        jsonConsultationTrends.put("sunday3",
nextWeekConsultationTrends.getSunday());

        consultationTrends = jsonConsultationTrends.toJSONString();
        return SUCCESS;
    }

    public Page<Physician> getPhysicianPage() {
        return physicianPage;
    }

    public void setPhysicianPage(Page<Physician> physicianPage) {
        this.physicianPage = physicianPage;
    }

    public Page<DiscussionThread> getDiscussionThreadPage() {
        return discussionThreadPage;
    }

    public void setDiscussionThreadPage(Page<DiscussionThread> discussionThreadPage)
{
        this.discussionThreadPage = discussionThreadPage;
    }

    public Page<Appointment> getAppointmentPage() {
        return appointmentPage;
    }

    public void setAppointmentPage(Page<Appointment> appointmentPage) {
        this.appointmentPage = appointmentPage;
    }

    public Long getNewThreadCount() {
        return newThreadCount;
    }

    public void setNewThreadCount(Long newThreadCount) {
        this.newThreadCount = newThreadCount;
    }

    public void setConsultationTrends(String consultationTrends) {
        this.consultationTrends = consultationTrends;
    }

    public String getConsultationTrends() {
        return consultationTrends;
    }

    public List<String> getAppointmentShortDetails() {
        return appointmentShortDetails;
    }

    public void setAppointmentShortDetails(List<String> appointmentShortDetails) {
        this.appointmentShortDetails = appointmentShortDetails;
    }

    public Long getNewAppointmentCount() {
        return newAppointmentCount;
    }

    public void setNewAppointmentCount(Long newAppointmentCount) {
        this.newAppointmentCount = newAppointmentCount;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.discussion/DiscussionAction.java**

```
package org.leaf.cms.action.discussion;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import org.joda.time.DateTime;
import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.DiscussionThreadReply;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.SMSEntry;
import org.leaf.cms.model.Subscription;
import org.leaf.cms.service.DiscussionThreadService;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.SMSEntryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;

public class DiscussionAction extends BaseLoggedInAction {

    private static final long serialVersionUID = 1L;
    private DiscussionThreadService discussionThreadService;
    private SMSEntryService smsEntryService;
    private EmailService emailService;

    private static final Logger _logger = Logger.getLogger(DiscussionAction.class);
    private Long id;
    private String title;
    private String message;

    private String filter;

    private static final String USER = "user";
    private static final String ALL = "all";
    private static final String SUBSCRIPTIONS = "subscriptions";
    private static final int ITEMS_PER_PAGE = 5;
    private DiscussionThread discussionThread;
    private DiscussionThreadReply discussionThreadReply;

    private Page<DiscussionThread> discussionThreads;
    private PhysicianService physicianService;

    private int currentPage;

    @Autowired
    public DiscussionAction(DiscussionThreadService discussionThreadService,
            SMSEntryService smsEntryService, EmailService emailService,
PhysicianService physicianService) {
        this.discussionThreadService = discussionThreadService;
        this.smsEntryService = smsEntryService;
        this.emailService = emailService;
        this.physicianService = physicianService;
    }

    @Override
    public String execute() {
        if (id != null) {
            discussionThread = discussionThreadService.findThreadById(id);
            if(!discussionThread.getIsSeen()){
            discussionThread.setIsSeen(Boolean.TRUE);
            discussionThread =
discussionThreadService.saveDiscussionThread(discussionThread);
            }
        } else {
            if (StringUtils.isNotEmpty(filter)) {
                if (filter.equalsIgnoreCase(SUBSCRIPTIONS)) {
```

```
                    discussionThreads =
discussionThreadService.findAllBySubscriptionFollowerWithPaging(getCurrentLoggedInUse
r(), currentPage == 0 ? 1 : currentPage, ITEMS_PER_PAGE);
                } else if (filter.equalsIgnoreCase(USER)) {
                    discussionThreads =
discussionThreadService.findAllByCreatedByWithPaging(getCurrentLoggedInUser(),
currentPage == 0 ? 1 : currentPage, ITEMS_PER_PAGE);
                } else {
                    discussionThreads =
discussionThreadService.findAllValidWithPaging(currentPage == 0 ? 1 : currentPage,
ITEMS_PER_PAGE);
                }
            } else {
                discussionThreads = discussionThreadService.findAllValidWithPaging(1,
ITEMS_PER_PAGE);
            }
        }
        return SUCCESS;
    }

    public String subscribe(){
    if(id == null){
            addActionError("Discussion Thread Id should not be null");
    }
            discussionThread = discussionThreadService.findThreadById(id);
            Subscription subscription = new Subscription();
            subscription.setFollower(getCurrentLoggedInUser());
            subscription.setThread(discussionThread);
            subscription.setCreatedOn(new DateTime());
            subscription.setUpdatedOn(new Date());
            subscription.setIsValid(Boolean.TRUE);
            if(discussionThreadService.subscribe(subscription)){
                    if(discussionThread.getCreatedBy().getSendOnForumSubscribed()){
                            Physician physician =
physicianService.findPhysician(discussionThread.getCreatedBy().getId());
                            Map<String, Object> hashMap = new HashMap<>();
                hashMap.put("name", discussionThread.getCreatedBy().getFullName());
                hashMap.put("subscriber", getCurrentLoggedInUser().getFullName());
                hashMap.put("title", discussionThread.getTitle());
                emailService.sendEmail("subscription_html.vm", "New Thread
Subscription Notification", physician.getEmail() , hashMap, null, null);
                    }

            }
            else{
                    addActionError("An error occurred");
            }
            return SUCCESS;
    }

    public String unsubscribe(){
    if(id == null){
            addActionError("Discussion Thread Id should not be null");
    }
            discussionThread = discussionThreadService.findThreadById(id);
    discussionThreadService.unsubscribe(discussionThread, getCurrentLoggedInUser());
    addActionMessage("Successfully unsubscribed to thread.");
            return SUCCESS;
    }

    public String delete(){
    if(id == null){
            addActionError("Discussion Thread Id should not be null");
    }
    discussionThread = discussionThreadService.findThreadById(id);
    discussionThreadService.delete(discussionThread);
    addActionMessage("Successfully deleted discussion thread");
    return "delete";
    }

    public String save() {
        if (id != null) {
            setDiscussionThread(discussionThreadService.findThreadById(id));
        } else {
            setDiscussionThread(new DiscussionThread());
            getDiscussionThread().setCreatedBy(getCurrentLoggedInUser());
```

```
            getDiscussionThread().setCreatedOn(new DateTime());
            getDiscussionThread().setIsValid(Boolean.TRUE);
        }
        getDiscussionThread().setTitle(title);
        if
((setDiscussionThread(discussionThreadService.saveDiscussionThread(getDiscussionThrea
d())) == null) {
            addActionError(ApplicationConstants.ERR_FAILED_SAVE_DISCUSSION_THREAD);
            return ERROR;
        }
        addActionMessage(ApplicationConstants.SUC_SAVE_DISCUSSION_THREAD);
        return SUCCESS;
    }

    public String saveInitialThread() {
        _logger.info("Saving initial discussion thread");
        _logger.info("Title: " + title);
        _logger.info("Message: " + message);
        discussionThread = new DiscussionThread();
        discussionThread.setCreatedBy(getCurrentLoggedInUser());
        discussionThread.setCreatedOn(new DateTime());
        discussionThread.setIsValid(Boolean.TRUE);
        discussionThread.setTitle(title);

        discussionThreadReply = new DiscussionThreadReply();
        discussionThreadReply.setCreatedOn(new DateTime());
        discussionThreadReply.setIsValid(Boolean.TRUE);
        discussionThreadReply.setPostedBy(getCurrentLoggedInUser());
        discussionThreadReply.setMessage(message);

        if ((discussionThread =
discussionThreadService.saveInitialDiscussionThread(discussionThread,
discussionThreadReply)) == null) {
            addActionError(ApplicationConstants.ERR_FAILED_SAVE_DISCUSSION_THREAD);
            return ERROR;
        }
        _logger.info("Successfully saved Initial Discussion Thread");
        SMSEntry smsEntry = new SMSEntry();

smsEntry.setRecipient(discussionThreadReply.getPostedBy().getCellphoneNumber());
        smsEntry.setMessage("Successfully created a new discussion thread");
        smsEntry.setCreatedOn(new DateTime());
        smsEntry.setIsValid(Boolean.TRUE);
        smsEntry.setIsSent(Boolean.FALSE);
        smsEntry.setUpdatedOn(new Date());
        smsEntryService.save(smsEntry);
        addActionMessage(ApplicationConstants.SUC_SAVE_DISCUSSION_THREAD);
        return SUCCESS;
    }

    public String saveReply() {
        _logger.info("Adding Reply to discussion id: " + id);
        _logger.info("Reply: " + message);
        if (StringUtils.isEmpty(message) || id == null) {
            addActionError(ApplicationConstants.ERR_FAILED_SAVE_DISCUSSION_REPLY);
            return ERROR;
        }

        discussionThread = discussionThreadService.findThreadById(id);

        discussionThreadReply = new DiscussionThreadReply();
        discussionThreadReply.setCreatedOn(new DateTime());
        discussionThreadReply.setIsValid(Boolean.TRUE);
        discussionThreadReply.setPostedBy(getCurrentLoggedInUser());
        discussionThreadReply.setMessage(message);
        discussionThreadReply.setThread(discussionThread);
        if ((discussionThreadReply =
discussionThreadService.saveDiscussionThreadReply(discussionThreadReply)) == null) {
            addActionError(ApplicationConstants.ERR_FAILED_SAVE_DISCUSSION_REPLY);
            return ERROR;
        }
        id = discussionThread.getId();

        if(discussionThread.getCreatedBy().getSendOnForumReply() &&
discussionThreadReply.getPostedBy() != discussionThread.getCreatedBy()){
```

```
            Physician physician =
physicianService.findPhysician(discussionThread.getCreatedBy().getId());
                    Map<String, Object> hashMap = new HashMap<>();
                    hashMap.put("subscriber",
discussionThreadReply.getPostedBy().getFullName());
            hashMap.put("name", discussionThread.getCreatedBy().getFullName());
            hashMap.put("title", discussionThread.getTitle());
            emailService.sendEmail("newreply_html.vm", "New Thread Reply
Notification", physician.getEmail() , hashMap, null, null);
            }

        if(discussionThread.getSubscriptions() != null &&
discussionThread.getSubscriptions().size() > 0){
            for(Subscription subs :  discussionThread.getSubscriptions()){
                    if(subs.getFollower().getSendSubscriptionUpdates() &&
discussionThreadReply.getPostedBy().getId() != subs.getFollower().getId()){
                    Physician physician =
physicianService.findPhysician(subs.getFollower().getId());
                    if(physician != null){
                            Map<String, Object> hashMap = new HashMap<>();
                       hashMap.put("name", subs.getFollower().getFullName());
                       hashMap.put("title", discussionThread.getTitle());
                       try {
                                                   hashMap.put("url", "http://"
+InetAddress.getLocalHost().getHostAddress() + ":8080/discussions.do?id=" +
discussionThread.getId());
                                           } catch (UnknownHostException e) {
                                                   e.printStackTrace();
                                           }
                       emailService.sendEmail("subscriptionupdate_html.vm", "Thread
Update for " + discussionThread.getTitle(), physician.getEmail() , hashMap, null,
null);
                    }
                    }
            }
        }

        addActionMessage(ApplicationConstants.SUC_SAVE_DISCUSSION_REPLY);
        return SUCCESS;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public DiscussionThread getDiscussionThread() {
        return discussionThread;
    }

    public DiscussionThread setDiscussionThread(DiscussionThread discussionThread) {
        this.discussionThread = discussionThread;
        return discussionThread;
    }

    public DiscussionThreadReply getDiscussionThreadReply() {
        return discussionThreadReply;
```

```
    }

    public void setDiscussionThreadReply(DiscussionThreadReply discussionThreadReply)
{
        this.discussionThreadReply = discussionThreadReply;
    }

    public Page<DiscussionThread> getDiscussionThreads() {
        return discussionThreads;
    }

    public void setDiscussionThreads(Page<DiscussionThread> discussionThreads) {
        this.discussionThreads = discussionThreads;
    }

    public String getFilter() {
        return filter;
    }

    public void setFilter(String filter) {
        this.filter = filter;
    }

    public int getCurrentPage() {
            return currentPage;
    }

    public void setCurrentPage(int currentPage) {
            this.currentPage = currentPage;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.interfaces/ConfigurationAware.java**

```
package org.leaf.cms.action.interfaces;

import org.leaf.cms.model.Configuration;

public interface ConfigurationAware
{
    public Configuration getConfiguration() ;
    public void setConfiguration(Configuration configuration);
}
```

- **web-pcs/src/org.leaf.cms.action.interfaces/CurrentLoggedInUserAware.java**

```
package org.leaf.cms.action.interfaces;

import org.leaf.cms.model.User;

public interface CurrentLoggedInUserAware
{
    public User getCurrentLoggedInUser() ;
    public void putCurrentLoggedInUser(User user) ;
}
```

- **web-pcs/src/org.leaf.cms.action.interfaces/UriAware.java**

```
package org.leaf.cms.action.interfaces;

public interface UriAware {
    public void setUri(String uri);
}
```

- **web-pcs/src/org.leaf.cms.action.login/LoginAction.java**

```
package org.leaf.cms.action.login;

import java.io.ByteArrayInputStream;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
```

```
import java.util.Set;

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

import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import org.apache.struts2.ServletActionContext;
import org.joda.time.DateTime;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.leaf.cms.action.BaseAction;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.leaf.cms.model.User;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.SpecialtyService;
import org.leaf.cms.service.UserService;
import org.leaf.cms.util.EncryptionUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;

import com.opensymphony.xwork2.Action;

public class LoginAction extends BaseAction {

    public static final Logger _logger = Logger.getLogger(LoginAction.class);
    private static final long serialVersionUID = 1L;

    private String username;
    private String password;
    private Long id;

    private String firstName;
    private String lastName;
    private String email;
    private Long medicalLicense;
    private List<Specialty> specialtyList;
    private Long specialtyId;
    private String mobileNumber;

    private UserService userService;

    private PhysicianService physicianService;

    private EmailService emailService;

    private SpecialtyService specialtyService;

    private String action;

    private String verificationCode;
    private List<Long> specialtyIds;

    @Autowired
    public LoginAction(
                        @Qualifier("userService") UserService userService,
                        @Qualifier("physicianService") PhysicianService
physicianService,
                        @Qualifier("emailService") EmailService emailService,
                        @Qualifier("specialtyService") SpecialtyService
specialtyService) {
        this.userService = userService;
        this.physicianService = physicianService;
        this.emailService = emailService;
        this.specialtyService = specialtyService;
    }

    public String login() throws NoSuchAlgorithmException {

        if (StringUtils.isEmpty(getUsername()) && StringUtils.isEmpty(getPassword()))
{
            return ERROR;
        }
```

```
        User user = userService.findUser(username,
EncryptionUtil.encryptPassword(password));
        if (user == null ) {
            System.out.println("USER IS NULL");
            addActionError("Username and Password did not match");
            return ERROR;
        }
        if(!user.isValid() ){
            addActionError("Failed to login.");
            return ERROR;
        }

        if(user instanceof Physician){
            Physician physician = physicianService.findByEmail(user.getUsername());
            if (physician == null || !physician.getIsActivited() ||
!physician.getIsValid()) {
                    addActionError("Failed to login.");
                    return ERROR;
            }
        }

        if(user.getUserType().equalsIgnoreCase("Physician")){
            action = "dashboard";
        }
        else{
            action ="dashboard";
        }

        session.put(ApplicationConstants.SESSION_USER_ID, user.getId());
        return SUCCESS;
    }

    public String logoff() {
        HttpServletRequest request = ServletActionContext.getRequest();
        HttpSession session = request.getSession(true);
        session.invalidate();
        return Action.LOGIN;
    }

    @SuppressWarnings("unchecked")
    public String smstest() {
        System.out.println(request.getParameter("action"));
        System.out.println(request.getParameter("message"));

        JSONObject obj = new JSONObject();
        if (request.getParameter("action").equalsIgnoreCase("incoming")) {
            JSONArray events = new JSONArray();
            JSONObject event = new JSONObject();
            event.put("event", "send");
            JSONArray messagesArray = new JSONArray();
            JSONObject message = new JSONObject();
            message.put("id", "sms_id");// for retrieval purposes
            message.put("to", "09178962106");
            message.put("message", "Forwarded message : " +
request.getParameter("message"));
            messagesArray.add(message);
            event.put("messages", messagesArray);
            events.add(event);
            obj.put("events", events);
            inputStream = new ByteArrayInputStream(obj.toJSONString().getBytes());
            return SUCCESS;
        } else {
            obj.put("d", "success");
            inputStream = new ByteArrayInputStream(obj.toJSONString().getBytes());
            return SUCCESS;
        }

    }

    public String registration() {
        specialtyList = specialtyService.getSpecialty();
        return SUCCESS;
    }

    public String forgotPassword(){
    return SUCCESS;
```

```
    }

    public String renewPassword() throws NoSuchAlgorithmException{
    Physician physician = null;

    if((physician = physicianService.findByEmailAndMedLicenseId(email,
medicalLicense)) != null){
            String newPassword = physician.getLastName() + "" +
physician.getCreatedOn().toString("MMddyyyyhhmmss");
            physician.setPassword(EncryptionUtil.encryptPassword(newPassword));
            physicianService.save(physician);
            Map<String, Object> hashMap = new HashMap<>();
            hashMap.put("name", physician.getFirstName() + " " +
physician.getLastName());
            hashMap.put("password", newPassword);
            emailService.sendEmail("forgotpassword_html.vm", "Successful Renewal of
Forgotten Password", physician.getEmail(), hashMap, null, null);
            addActionMessage(ApplicationConstants.SUC_CHANGE_PASSWORD);
            return SUCCESS;
    }
    addActionError(ApplicationConstants.ERR_CHANGE_PASSWORD);

            return ERROR;
    }

    public String register() {
    specialtyList = specialtyService.getSpecialty();
        try {
            if (physicianService.countByEmail(email) > 0) {
                addActionError(ApplicationConstants.ERR_USERNAME_MUST_BE_UNIQUE);
                return ERROR;
            }

            if (physicianService.countByMedLicenseId(medicalLicense) > 0){
            addActionError(ApplicationConstants.ERR_MEDLICENSEID_MUST_BE_UNIQUE);
            return ERROR;
            }

            Physician physician = new Physician();
            physician.setEmail(email);
            physician.setFirstName(firstName);
            physician.setLastName(lastName);
            physician.setMedLicenseId(medicalLicense);
            physician.setCreatedOn(new DateTime());
            physician.setIsValid(Boolean.TRUE);
            physician.setIsActivited(Boolean.FALSE);
            physician.setVerificationCode(EncryptionUtil.encryptPassword(email));
            physician.setPassword(EncryptionUtil.encryptPassword(password));
            physician.setUsername(email);
            physician.setCellphoneNumber(mobileNumber);

            Set<Specialty> specialties = new HashSet<Specialty>();

            for(Long id: specialtyIds){
            specialties.add(specialtyService.findSpecialty(id));
            }

            physician.setSpecialties(specialties);

            if ((physician = physicianService.save(physician)) == null) {
                addActionError(ApplicationConstants.ERR_FAILED_SAVE_PHYSICIAN);
                return Action.SUCCESS;
            }
            Map<String, Object> hashMap = new HashMap<>();
            hashMap.put("name", physician.getFirstName() + " " +
physician.getLastName());
            emailService.sendEmail("registration_html.vm", "Welcome to WebPCS",
physician.getEmail(), hashMap, null, null);
            addActionMessage(ApplicationConstants.SUC_SAVE_PHYSICIAN);
        } catch (Exception e) {
            _logger.error(e.getMessage());
        }
        return Action.SUCCESS;
    }

    public String verify() {
```

```
        if (verificationCode == null || id == null) {
            addActionError(ApplicationConstants.ERR_FAILED_ACTIVATE_PHYSICIAN);
        }

        if (physicianService.activateAccount(id, verificationCode) == null) {
            addActionError(ApplicationConstants.ERR_FAILED_ACTIVATE_PHYSICIAN);
        }

        return Action.SUCCESS;
    }

    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;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Long getMedicalLicense() {
        return medicalLicense;
    }

    public void setMedicalLicense(Long medicalLicense) {
        this.medicalLicense = medicalLicense;
    }

    public List<Specialty> getSpecialtyList() {
        return specialtyList;
    }

    public void setSpecialtyList(List<Specialty> specialtyList) {
        this.specialtyList = specialtyList;
    }

    public Long getSpecialtyId() {
        return specialtyId;
    }

    public void setSpecialtyId(Long specialtyId) {
        this.specialtyId = specialtyId;
    }

    public String getVerificationCode() {
```

```
        return verificationCode;
    }

    public void setVerificationCode(String verificationCode) {
        this.verificationCode = verificationCode;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getMobileNumber() {
            return mobileNumber;
    }

    public void setMobileNumber(String mobileNumber) {
            this.mobileNumber = mobileNumber;
    }

    public List<Long> getSpecialtyIds() {
            return specialtyIds;
    }

    public void setSpecialtyIds(List<Long> specialtyIds) {
            this.specialtyIds = specialtyIds;
    }

    public String getAction() {
            return action;
    }

    public void setAction(String action) {
            this.action = action;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.manage/AbstractFileMaintenanceClass.java**

```
/*** Eclipse Class Decompiler plugin, copyright (c) 2012 Chao Chen
(cnfree2000@hotmail.com) ***/

package org.leaf.cms.action.manage;

import java.io.ByteArrayInputStream;
import java.io.Serializable;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import javax.servlet.ServletContext;

import org.apache.struts2.util.ServletContextAware;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.leaf.cms.action.BaseLoggedInActionWithPaging;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseId;
import org.leaf.cms.model.repository.BaseJpaRepository;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.domain.Sort.Order;

import com.opensymphony.xwork2.Preparable;

public abstract class AbstractFileMaintenanceClass<T extends BaseId<ID>, ID extends
Serializable, D extends BaseJpaRepository<T, ID>> extends
        BaseLoggedInActionWithPaging<T> implements Preparable, ServletContextAware {
```

```
private static final long serialVersionUID = 1L;
protected ID id;
protected T entity;
protected D repository;
protected JSONObject jsonResult;
protected String oper;
protected Map<String, String> filters;
protected ServletContext servletContext;

protected String validateAction() {
    return "success";
}

protected abstract void initializeEntity();

protected Map<String, String> prepareFilter() {
    return null;
}

@Override
public void prepare() {
    initializeEntity();
}

protected void afterSave() {
}

protected void beforeSave() {
}

protected String validateSave() {
    return "success";
}

protected void afterDelete() {
}

protected void beforeDelete() {
}

public AbstractFileMaintenanceClass(D repository) {
    this.repository = repository;
}

public String save() {
    String validActionStatus = validateAction();
    if (!validActionStatus.equalsIgnoreCase("success")) {
        return validActionStatus;
    }
    String validToSaveStatus = validateSave();
    if (!validToSaveStatus.equalsIgnoreCase("success")) {
        return validToSaveStatus;
    }
    if (getOper().equalsIgnoreCase("del")) {
        return delete();
    }
    beforeSave();
    repository.save(getEntity());
    afterSave();
    return "success";
}

public String delete() {
    String validActionStatus = validateAction();
    if (!validActionStatus.equalsIgnoreCase("success")) {
        return validActionStatus;
    }
    if (getEntity().getId() != null) {
        beforeDelete();
        //repository.delete(getEntity());
        afterDelete();
    } else {
        addActionError("Id was not provided");
        return "error";
    }
```

```
        return "success";
    }

    public ID getId() {
        return id;
    }

    public void setId(ID id) {
        this.id = id;
    }

    public List<T> getEntities() {
        return repository.findAll();
    }

    @SuppressWarnings("unchecked")
    public String getEntitiesWithPaging() {
        String validActionStatus = validateAction();
        if (!validActionStatus.equalsIgnoreCase("success")) {
            return validActionStatus;
        }
        jsonResult = new JSONObject();
        List<Order> orderList = new ArrayList<>();
        for (int ctr = 0; ctr < sidx.length && ctr < sord.length; ctr++) {
            Order order = new Order(Direction.valueOf(sord[ctr].toUpperCase()),
sidx[ctr]);
            orderList.add(order);
        }
        Sort sort = null;
        if (!orderList.isEmpty()) {
            sort = new Sort(orderList);
        }
        PageRequest pageRequest = new PageRequest(page - 1, rows, sort);
        Page<T> repositoryPages = repository.findByIsValidTrue(pageRequest);
        if (repositoryPages != null) {
            JSONArray jsonArray = new JSONArray();
            T entity;
            for (Iterator<T> iterator = repositoryPages.iterator();
iterator.hasNext(); jsonArray.add(((JsonConvertible) entity).toJSONObject())) {
                entity = iterator.next();
            }

            jsonResult.put("rows", jsonArray);
            jsonResult.put("total", repositoryPages.getTotalElements() / rows + 1);
            jsonResult.put("records", repositoryPages.getTotalElements());
            jsonResult.put("page", page);
        } else {
            jsonResult.put("d", "ERROR");
        }
        inputStream = new
ByteArrayInputStream(jsonResult.toJSONString().getBytes(Charset.forName("UTF-8")));
        return "success";
    }

    public T getEntity() {
        return entity;
    }

    public void setEntity(T entity) {
        this.entity = entity;
    }

    public String getOper() {
        return oper != null ? oper : "";
    }

    public void setOper(String oper) {
        this.oper = oper;
    }

    @Override
    public void setServletContext(ServletContext context) {
        servletContext = context;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.manage/NewPhysicianAccountsAction.java**

```
package org.leaf.cms.action.manage;

import java.io.ByteArrayInputStream;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.repository.PhysicianRepository;
import org.leaf.cms.model.repository.UserRepository;
import org.leaf.cms.service.EmailService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.domain.Sort.Order;

public class NewPhysicianAccountsAction extends
AbstractFileMaintenanceClass<Physician, Long, PhysicianRepository>{

    private static final long serialVersionUID = 1L;
    private EmailService emailService;
    private UserRepository userRepository;

    @Autowired
    public NewPhysicianAccountsAction(PhysicianRepository physicianRepository,
                    UserRepository userRepository,
                    @Qualifier("emailService") EmailService emailService){
            super(physicianRepository);
            this.emailService = emailService;
            this.userRepository = userRepository;

    }

    @Override
    @SuppressWarnings("unchecked")
    public String getEntitiesWithPaging() {
            String validActionStatus = validateAction();
        if (!validActionStatus.equalsIgnoreCase("success")) {
            return validActionStatus;
        }
        jsonResult = new JSONObject();
        List<Order> orderList = new ArrayList<>();
        for (int ctr = 0; ctr < sidx.length && ctr < sord.length; ctr++) {
            Order order = new Order(Direction.valueOf(sord[ctr].toUpperCase()),
sidx[ctr]);
            orderList.add(order);
        }
        Sort sort = null;
        if (!orderList.isEmpty()) {
            sort = new Sort(orderList);
        }
        PageRequest pageRequest = new PageRequest(page - 1, rows, sort);
        Page<Physician> repositoryPages =
repository.findByIsActivitedFalseAndIsValidTrueAndActivationDateIsNull(pageRequest);
        if (repositoryPages != null) {
            JSONArray jsonArray = new JSONArray();
            Physician entity;
            for (Iterator<Physician> iterator = repositoryPages.iterator();
iterator.hasNext(); jsonArray.add(((JsonConvertible) entity).toJSONObject())) {
                entity = iterator.next();
            }

            jsonResult.put("rows", jsonArray);
            jsonResult.put("total", repositoryPages.getTotalElements() / rows + 1);
            jsonResult.put("records", repositoryPages.getTotalElements());
```

```
            jsonResult.put("page", page);
        } else {
            jsonResult.put("d", "ERROR");
        }
        inputStream = new
ByteArrayInputStream(jsonResult.toJSONString().getBytes(Charset.forName("UTF-8")));
        return "success";
    }

    public String activate(){
            if(id != null){
                    Physician physician = repository.findOne(id);
                    if(physician != null){
                            physician.setIsActivited(Boolean.TRUE);
                            repository.save(physician);
                            Map<String, Object> hashMap = new HashMap<>();
                hashMap.put("url", "http://localhost:8080/register-verify.do?" +
"verificationCode=" + physician.getVerificationCode() + "&id="
                        + physician.getId());
                hashMap.put("name", physician.getFirstName() + " " +
physician.getLastName());
                emailService.sendEmail("accountactivation_html.vm", "Activation for
Web PCS Account", physician.getEmail(), hashMap, null, null);
                    }
            }

            return SUCCESS;
    }

    public String deactivate(){
            if(id != null){
                    Physician physician = repository.findOne(id);
                    if(physician != null){
                            physician.setIsValid(Boolean.FALSE);
                            repository.save(physician);
                            /*userRepository.delete(id);
                            repository.delete(physician);*/
                            Map<String, Object> hashMap = new HashMap<>();
                hashMap.put("name", physician.getFirstName() + " " +
physician.getLastName());
            emailService.sendEmail("registrationfailed_html.vm", "Registration Denied
by Web PCS", physician.getEmail(), hashMap, null, null);
                    }
            }
            return SUCCESS;
    }

    public String execute(){
            return SUCCESS;
    }

    @Override
    protected void initializeEntity() {

    }
}
```

- **web-pcs/src/org.leaf.cms.action.manage/PhysicianAction.java**

```
package org.leaf.cms.action.manage;

import java.security.NoSuchAlgorithmException;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import org.joda.time.DateTime;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.leaf.cms.model.repository.PhysicianRepository;
import org.leaf.cms.service.SpecialtyService;
import org.leaf.cms.util.EncryptionUtil;
import org.springframework.beans.factory.annotation.Autowired;
```

```
public class PhysicianAction extends AbstractFileMaintenanceClass<Physician, Long,
PhysicianRepository>{

    private List<Long> specialtyIds;
    private SpecialtyService specialtyService;
    private String password;

    @Autowired
    public PhysicianAction(PhysicianRepository repository, SpecialtyService
specialtyService) {
        super(repository);
        this.specialtyService = specialtyService;
    }

    private static final long serialVersionUID = 1L;

    @Override
    protected void initializeEntity() {
        if (getId() == null) {
            setEntity(new Physician());
            entity.setCreatedOn(new DateTime());
        } else {
            entity = repository.findOne(getId());

        }
    }

    @Override
    protected void beforeSave() {
        super.beforeSave();
        try {
            Set<Specialty> specialties = new HashSet<Specialty>();
            for(Long id: specialtyIds){
            specialties.add(specialtyService.findSpecialty(id));
            }
            entity.setUsername(entity.getEmail());
            if(entity.getId() == null){
            entity.setPassword(EncryptionUtil.encryptPassword(entity.getPassword()));
            }
            entity.setSpecialties(specialties);
            entity.setIsActivited(request.getParameter("isActivited") != null );
            entity.setSendOnForumReply(request.getParameter("sendOnForumReply") !=
null );

entity.setSendOnForumSubscribed(request.getParameter("sendOnForumSubscribed") != null
);

entity.setSendSubscriptionUpdates(request.getParameter("sendSubscriptionUpdates") !=
null );
            entity.setSendWhenBooked(request.getParameter("sendWhenBooked") != null
);
            entity.setSendWhenCancelled(request.getParameter("sendWhenCancelled") !=
null );
            entity.setSendWhenMessageSent(request.getParameter("sendWhenMessageSent")
!= null );
            entity.setSendWhenResceduled(request.getParameter("sendWhenResceduled")
!= null );
            entity.setIsValid(Boolean.TRUE);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected String validateSave() {
    if (entity.getId() == null && repository.countByEmail(entity.getEmail()) > 0) {
            addActionError(ApplicationConstants.ERR_USERNAME_MUST_BE_UNIQUE);
            return ERROR;
        }

        if (entity.getId() == null &&
repository.countByMedLicenseId(entity.getMedLicenseId()) > 0){
            addActionError(ApplicationConstants.ERR_MEDLICENSEID_MUST_BE_UNIQUE);
            return ERROR;
        }
    return super.validateSave();
```

```
    }

    @Override
    public String execute() {
    if(id != null){
            entity = repository.findOne(id);
    }
        return SUCCESS;
    }

    public String changePassword() throws NoSuchAlgorithmException{
    if(id != null && password != null){
            entity = repository.findOne(id);
            entity.setPassword(EncryptionUtil.encryptPassword(password));
            repository.save(entity);
            addActionMessage("Successfully change password");
    }
    return SUCCESS;
    }


    public List<Specialty> getSpecialtyList() {
            return specialtyService.getSpecialty();
    }

    public List<Long> getSpecialtyIds() {
            return specialtyIds;
    }

    public void setSpecialtyIds(List<Long> specialtyIds) {
            this.specialtyIds = specialtyIds;
    }

    public String getPassword() {
            return password;
    }

    public void setPassword(String password) {
            this.password = password;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.manage/UserAction.java**

```
package org.leaf.cms.action.manage;

import java.security.NoSuchAlgorithmException;

import org.joda.time.DateTime;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.model.repository.UserRepository;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.util.EncryptionUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.util.StringUtils;

public class UserAction extends AbstractFileMaintenanceClass<User, Long,
UserRepository> {

    private String firstName;
    private String lastName;
    private PhysicianService physicianService;
    private Boolean sendMail;
    private EmailService emailService;

    @Autowired
    public UserAction(UserRepository repository, PhysicianService physicianService,
EmailService emailService) {
        super(repository);
        this.physicianService = physicianService;
        this.emailService = emailService;
    }
```

```
    private static final long serialVersionUID = 1L;

    @Override
    protected void initializeEntity() {
        if (getId() == null) {
            setEntity(new User());
            entity.setCreatedOn(new DateTime());
        } else {
            entity = repository.findOne(getId());
            if (!getOper().equalsIgnoreCase("del")) {
                entity.setFirstName(firstName);
                entity.setLastName(lastName);
            }
        }
    }

    @Override
    protected String validateAction() {
        if ((getOper() == null || getOper().equalsIgnoreCase("")) && entity.getId()
== null) {
            User user = repository.findByUsername(entity.getUsername());
            if (user != null) {
                addActionError("Username is not unique.");
                return "error";
            }
        }
        return super.validateAction();
    }

    @Override
    protected void beforeSave() {
        super.beforeSave();
        if (!StringUtils.isEmpty(entity.getPassword()) && entity.getId() == null) {
            try {

entity.setPassword(EncryptionUtil.encryptPassword(entity.getPassword()));
                entity.setIsValid(Boolean.TRUE);
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    protected void afterDelete() {
        super.afterDelete();
        if (getSendMail()) {
        }
    }

    @Override
    public String execute() {
        return SUCCESS;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public String delete() {
        if (entity instanceof Physician) {
            Physician physician = physicianService.findPhysician(entity.getId());
            physician.setIsActivited(false);
```

```
            physician.setIsValid(false);
            physicianService.save(physician);
        } else {
            entity.setIsValid(Boolean.FALSE);
            repository.save(entity);
        }
        return "success";
    }

    public Boolean getSendMail() {
        if (sendMail == null) {
            return Boolean.FALSE;
        }
        return sendMail;
    }

    public void setSendMail(Boolean sendMail) {
        this.sendMail = sendMail;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.planner/AppointmentEvent.java**

```
package org.leaf.cms.action.planner;

import com.dhtmlx.planner.DHXEvent;

public class AppointmentEvent extends DHXEvent {

    public Boolean readonly = Boolean.FALSE;
    public String specialty;
    public String question;
    public String patientLabExam;
    public String patientDiagnosis;
    public String remarks;
    public String status;
    public String requestingPhysician;
    public String consultant;

    public Boolean getReadonly() {
        return readonly;
    }

    public void setReadonly(Boolean readonly) {
        this.readonly = readonly;
    }

    public void setReadonly(String readonly) {
        this.readonly = Boolean.valueOf(readonly);
    }

    public String getSpecialty() {
        return specialty;
    }

    public void setSpecialty(String specialty) {
        this.specialty = specialty;
    }

    public String getQuestion() {
        return question;
    }

    public void setQuestion(String question) {
        this.question = question;
    }

    public String getPatientLabExam() {
        return patientLabExam;
    }

    public void setPatientLabExam(String patientLabExam) {
        this.patientLabExam = patientLabExam;
    }

    public String getPatientDiagnosis() {
```

```
        return patientDiagnosis;
    }

    public void setPatientDiagnosis(String patientDiagnosis) {
        this.patientDiagnosis = patientDiagnosis;
    }

    public String getRemarks() {
        return remarks;
    }

    public void setRemarks(String remarks) {
        this.remarks = remarks;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getRequestingPhysician() {
        return requestingPhysician;
    }

    public void setRequestingPhysician(String requestingPhysician) {
        this.requestingPhysician = requestingPhysician;
    }

    public String getConsultant() {
        return consultant;
    }

    public void setConsultant(String consultant) {
        this.consultant = consultant;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.planner/DHXLightBoxLabel.java**

```
package org.leaf.cms.action.planner;

import com.dhtmlx.planner.controls.DHXLightboxField;

public class DHXLightBoxLabel extends DHXLightboxField {

    public DHXLightBoxLabel(String name, String label) {
        super(name, label);
        setType("label");
    }

}
```

- **web-pcs/src/org.leaf.cms.action.planner/MessageStore.java**

```
package org.leaf.cms.action.planner;

public class MessageStore {

    private String planner;
    private String data;

    public String getPlanner() {
        return planner;
    }

    public void setPlanner(String planner) {
        this.planner = planner;
    }

    public String getData() {
        return data;
    }
```

```
    public void setData(String data) {
        this.data = data;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.planner/PlannerAction.java**

```
package org.leaf.cms.action.planner;

import org.leaf.cms.action.BaseLoggedInAction;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.ConfigurationService;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.SMSEntryService;
import org.springframework.beans.factory.annotation.Autowired;

import com.dhtmlx.planner.DHXPlanner;
import com.dhtmlx.planner.DHXSkin;
import com.dhtmlx.planner.controls.DHXLightboxTime;
import com.dhtmlx.planner.data.DHXDataFormat;
import com.dhtmlx.planner.data.DHXDataLoader.DHXDynLoadingMode;
import com.dhtmlx.planner.extensions.DHXExtension;

public class PlannerAction extends BaseLoggedInAction {

    private static final long serialVersionUID = -296013827668835974L;
    private AppointmentService appointmentService;
    private PhysicianService physicianService;
    private EmailService emailService;
    private ConfigurationService configurationService;
    private SMSEntryService smsEntryService;
    private MessageStore messageStore = new MessageStore();

    @Autowired
    public PlannerAction(
                        AppointmentService appointmentService,
                        PhysicianService physicianService,
                        EmailService emailService,
                        ConfigurationService configurationService,
                        SMSEntryService smsEntryService) {
        this.appointmentService = appointmentService;
        this.physicianService = physicianService;
        this.emailService = emailService;
        this.configurationService = configurationService;
        this.smsEntryService = smsEntryService;
    }

    public String initializePlanner() throws Exception {
        DHXPlanner planner = new DHXPlanner("./codebase/", DHXSkin.GLOSSY);

        planner.lightbox.clear();

        DHXLightBoxLabel specialty = new DHXLightBoxLabel("specialty", "Specialty");
        specialty.setHeight(25);
        planner.lightbox.add(specialty);

        DHXLightBoxLabel requestingPhysician = new
DHXLightBoxLabel("requestingPhysician", "Primary Care Physician");
        requestingPhysician.setHeight(25);
        planner.lightbox.add(requestingPhysician);

        DHXLightBoxLabel consultant = new DHXLightBoxLabel("consultant",
"Consultant");
        consultant.setHeight(25);
        planner.lightbox.add(consultant);

        DHXLightBoxLabel question = new DHXLightBoxLabel("question", "Clinical
Question");
        question.setHeight(50);
        planner.lightbox.add(question);

        DHXLightBoxLabel patientLabExam = new DHXLightBoxLabel("patientLabExam",
"Laboratory Studies");
        patientLabExam.setHeight(50);
```

```
        planner.lightbox.add(patientLabExam);

        DHXLightBoxLabel patientDiagnosis = new DHXLightBoxLabel("patientDiagnosis",
"Diagnostic Findings");
        patientDiagnosis.setHeight(50);
        planner.lightbox.add(patientDiagnosis);

        DHXLightBoxLabel remarks = new DHXLightBoxLabel("remarks", "Remarks");
        remarks.setHeight(50);
        planner.lightbox.add(remarks);

        DHXLightBoxLabel status = new DHXLightBoxLabel("status", "Status");
        status.setHeight(25);
        planner.lightbox.add(status);

        DHXLightboxTime time = new DHXLightboxTime("time", "Time");
        planner.lightbox.add(time);

        planner.config.setEditOnCreate(false);
        planner.config.setDblClickCreate(false);
        planner.config.setDetailsOnCreate(false);
        planner.config.setDragCreate(false);
        planner.config.setDragLightbox(false);
        planner.config.setDragMove(false);
        planner.config.setDragResize(false);
        planner.config.setWideForm(false);
        planner.config.setScrollHour(8);
        planner.config.setEventDuration(30);
        planner.config.setShowLoading(true);
        planner.config.setSeparateShortEvents(true);
        planner.extensions.add(DHXExtension.QUICK_INFO);
        planner.extensions.add(DHXExtension.READONLY);
        planner.extensions.add(DHXExtension.COLLISION);
        planner.extensions.add(DHXExtension.CONTAINER_AUTORESIZE);

        planner.data.enableDynamicLoading(DHXDynLoadingMode.week);
        planner.data.dataprocessor.setURL("events.do");
        planner.load("events.do", DHXDataFormat.JSON);
        messageStore.setPlanner(planner.render());
        return SUCCESS;
    }

    public String events() throws Exception {PlannerEventsManager events = new
PlannerEventsManager(request, appointmentService, physicianService,
getCurrentLoggedInUser(), emailService, configurationService, smsEntryService);
        messageStore.setData(events.run());
        return SUCCESS;
    }

    public MessageStore getMessageStore() {
        return messageStore;
    }

    public void setMessageStore(MessageStore messageStore) {
        this.messageStore = messageStore;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.planner/PlannerEventsManager.java**

```
package org.leaf.cms.action.planner;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.joda.time.DateTime;
import org.joda.time.LocalDate;
import org.leaf.cms.enums.ClinicalCaseStatusEnum;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ClinicalCase;
```

```
import org.leaf.cms.model.Configuration;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.ConfigurationService;
import org.leaf.cms.service.EmailService;
import org.leaf.cms.service.PhysicianService;
import org.leaf.cms.service.SMSEntryService;

import com.dhtmlx.planner.DHXEv;
import com.dhtmlx.planner.DHXEventsManager;
import com.dhtmlx.planner.DHXStatus;

public class PlannerEventsManager extends DHXEventsManager {

    private AppointmentService appointmentService;
    private PhysicianService physicianService;
    private User user;
    private EmailService emailService;
    private ConfigurationService configurationService;
    private SMSEntryService smsEntryService;

    public PlannerEventsManager(HttpServletRequest request) {
        this(request, null, null, null, null, null, null);
    }

    public PlannerEventsManager(
                                HttpServletRequest request,
                                AppointmentService appointmentService,
                                PhysicianService physicianService,
                                User user,
                                EmailService emailService,
                                ConfigurationService configurationService,
                                SMSEntryService smsEntryService) {
        super(request);
        this.appointmentService = appointmentService;
        this.physicianService = physicianService;
        this.user = user;
        this.emailService = emailService;
        this.configurationService = configurationService;
        this.smsEntryService = smsEntryService;
    }

    @Override
    public Iterable<DHXEv> getEvents() {
        Date from = getFilterFrom();
        Date to = getFilterTo();
        List<Appointment> appointments;
        Physician physician = physicianService.findPhysician(user.getId());
        if (from == null || to == null) {
            appointments = appointmentService.getAppointments(physician, physician);
        } else {
            LocalDate start = new LocalDate(from);
            LocalDate end = new LocalDate(to);
            appointments = appointmentService.getAppointments(physician, physician,
start, end);
        }

        List<DHXEv> events = new ArrayList<>();
        for (Appointment appointment : appointments) {
            AppointmentEvent event = new AppointmentEvent();
            event.setId(appointment.getId().intValue());
            event.setStart_date(appointment.getConsultationStart().toDate());
            event.setEnd_date(appointment.getConsultationEnd().toDate());

            ClinicalCase clinicalCase = appointment.getClinicalCase();
            event.setSpecialty(clinicalCase.getSpecialty().getDescription());
            event.setQuestion(clinicalCase.getQuestion());
            event.setPatientLabExam(clinicalCase.getPatientLabExam());
            event.setPatientDiagnosis(clinicalCase.getPatientDiagnosis());
            event.setRemarks(clinicalCase.getRemarks());
            event.setStatus(clinicalCase.getStatus());

            Physician requestingPhysician = appointment.getRequestingPhysician();
            event.setRequestingPhysician(requestingPhysician.getFullName());
```

```
            Physician consultant = appointment.getConsultant();
            event.setConsultant(consultant.getFullName());

            StringBuilder text = new StringBuilder();
            text.append(requestingPhysician.getFullName()).append("'s appointment
with ").append(consultant.getFullName());
            event.setText(text.toString());

            if (!appointment.isValid() ||
!ClinicalCaseStatusEnum.ACTIVE.getStatus().equalsIgnoreCase(clinicalCase.getStatus())
) {
                event.setReadonly(true);
            }

            events.add(event);
        }
        return events;
    }

    @Override
    public DHXStatus saveEvent(DHXEv event, DHXStatus status) {
        AppointmentEvent appointmentEvent = (AppointmentEvent) event;
        switch (status) {
        case UPDATE: {
            DateTime consultationStart = new
DateTime(appointmentEvent.getStart_date());
            DateTime consultationEnd = new DateTime(appointmentEvent.getEnd_date());
            Appointment appointment =
appointmentService.findAppointment(appointmentEvent.getId().longValue());
            if (appointment.getConsultationStart().isEqual(consultationStart) &&
appointment.getConsultationEnd().isEqual(consultationEnd)) {
                return DHXStatus.ERROR;
            } else {
                appointment.setConsultationDate(consultationStart.toLocalDate());
                appointment.setConsultationStart(consultationStart);
                appointment.setConsultationEnd(consultationEnd);
                appointmentService.rescheduleAppointment(appointment);
            }

            Physician requestingPhysician = appointment.getRequestingPhysician();
            Physician consultant = appointment.getConsultant();
            String consultationDateString =
consultationDateStringBuilder(consultationStart, consultationEnd);

            if (requestingPhysician.getSendWhenRescheduled()) {
                Map<String, Object> requestingPhysicianHashMap = new HashMap<>();
                requestingPhysicianHashMap.put("name",
requestingPhysician.getFullName());
                requestingPhysicianHashMap.put("consultationDateString",
consultationDateString);
                requestingPhysicianHashMap.put("requestingPhysicianName",
requestingPhysician.getFullName());
                requestingPhysicianHashMap.put("consultantName",
consultant.getFullName());
                emailService.sendEmail(
                                        "rescheduleappointment.html",
                                        "Web PCS Rescheduled Appointment",
                                        requestingPhysician.getEmail(),
                                        requestingPhysicianHashMap,
                                        null,
                                        null);
            }

            if (consultant.getSendWhenRescheduled()) {
                Map<String, Object> consultantHashMap = new HashMap<>();
                consultantHashMap.put("name", consultant.getFullName());
                consultantHashMap.put("consultationDateString",
consultationDateString);
                consultantHashMap.put("requestingPhysicianName",
requestingPhysician.getFullName());
                consultantHashMap.put("consultantName", consultant.getFullName());
                emailService.sendEmail(
                                        "rescheduleappointment.html",
                                        "Web PCS Rescheduled Appointment",
                                        consultant.getEmail(),
                                        consultantHashMap,
```

```
                                    null,
                                    null);
            }

            Configuration configuration = configurationService.mainConfiguration();
            if (configuration.getIsSMSEnabled()) {
                Map<String, Object> requestingPhysicianSmsHashMap = new HashMap<>();
                requestingPhysicianSmsHashMap.put("name",
requestingPhysician.getFullName());
                requestingPhysicianSmsHashMap.put("consultationDateString",
consultationDateString);
                requestingPhysicianSmsHashMap.put("otherName",
consultant.getFullName());
                smsEntryService.saveMessageToBeSent(
                                                    "rescheduleappointment_sms.vm",

requestingPhysician.getCellphoneNumber(),
                                                    requestingPhysicianSmsHashMap);

                Map<String, Object> consultantSmsHashMap = new HashMap<>();
                consultantSmsHashMap.put("name", consultant.getFullName());
                consultantSmsHashMap.put("consultationDateString",
consultationDateString);
                consultantSmsHashMap.put("otherName",
requestingPhysician.getFullName());
                smsEntryService.saveMessageToBeSent("rescheduleappointment_sms.vm",
consultant.getCellphoneNumber(), consultantSmsHashMap);
            }
        }
            break;
        case INSERT:
            break;
        case DELETE: {
            Boolean readonly = appointmentEvent.getReadonly();
            if (readonly) {
                return DHXStatus.ERROR;
            }
            Appointment appointment =
appointmentService.findAppointment(appointmentEvent.getId().longValue());
            Physician requestingPhysician = appointment.getRequestingPhysician();
            Physician consultant = appointment.getConsultant();
            String consultationDateString =
consultationDateStringBuilder(appointment.getConsultationStart(),
appointment.getConsultationEnd());


appointmentService.cancelAppointment(appointmentEvent.getId().longValue());

            if (requestingPhysician.getSendWhenCancelled()) {
                Map<String, Object> requestingPhysicianHashMap = new HashMap<>();
                requestingPhysicianHashMap.put("name",
requestingPhysician.getFullName());
                requestingPhysicianHashMap.put("consultationDateString",
consultationDateString);
                requestingPhysicianHashMap.put("requestingPhysicianName",
requestingPhysician.getFullName());
                requestingPhysicianHashMap.put("consultantName",
consultant.getFullName());
                emailService.sendEmail(
                                    "cancelappointment.html",
                                    "Web PCS Cancelled Appointment",
                                    requestingPhysician.getEmail(),
                                    requestingPhysicianHashMap,
                                    null,
                                    null);
            }

            if (consultant.getSendWhenCancelled()) {
                Map<String, Object> consultantHashMap = new HashMap<>();
                consultantHashMap.put("name", consultant.getFullName());
                consultantHashMap.put("consultationDateString",
consultationDateString);
                consultantHashMap.put("requestingPhysicianName",
requestingPhysician.getFullName());
                consultantHashMap.put("consultantName", consultant.getFullName());
                emailService.sendEmail(
```

```
                                        "cancelappointment.html",
                                        "Web PCS Cancelled Appointment",
                                        consultant.getEmail(),
                                        consultantHashMap,
                                        null,
                                        null);
                }

                Configuration configuration = configurationService.mainConfiguration();
                if (configuration.getIsSMSEnabled()) {
                    Map<String, Object> requestingPhysicianSmsHashMap = new HashMap<>();
                    requestingPhysicianSmsHashMap.put("name",
requestingPhysician.getFullName());
                    requestingPhysicianSmsHashMap.put("consultationDateString",
consultationDateString);
                    requestingPhysicianSmsHashMap.put("otherName",
consultant.getFullName());
                    smsEntryService.saveMessageToBeSent(
                                                            "cancelappointment_sms.vm",

requestingPhysician.getCellphoneNumber(),
                                                            requestingPhysicianSmsHashMap);

                    Map<String, Object> consultantSmsHashMap = new HashMap<>();
                    consultantSmsHashMap.put("name", consultant.getFullName());
                    consultantSmsHashMap.put("consultationDateString",
consultationDateString);
                    consultantSmsHashMap.put("otherName",
requestingPhysician.getFullName());
                    smsEntryService.saveMessageToBeSent("cancelappointment_sms.vm",
consultant.getCellphoneNumber(), consultantSmsHashMap);
                }
            }
                break;
            case ERROR:
                break;
            case UNKNOWN:
                break;
            default:
                break;
            }
            return status;
        }

        private String consultationDateStringBuilder(DateTime start, DateTime end) {
            StringBuilder stringBuilder = new StringBuilder();
            if (start.getDayOfYear() == end.getDayOfYear()) {
                stringBuilder.append(start.dayOfWeek().getAsText())
                             .append(", ")
                             .append(start.dayOfMonth().getAsText())
                             .append(' ')
                             .append(start.monthOfYear().getAsText())
                             .append(' ')
                             .append(start.year().getAsText())
                             .append(' ')
                             .append(start.hourOfDay().getAsText())
                             .append(':')
                             .append(start.minuteOfHour().getAsText())
                             .append(" - ")
                             .append(end.hourOfDay().getAsText())
                             .append(':')
                             .append(end.minuteOfHour().getAsText());
            } else {
                stringBuilder.append(start.dayOfMonth().getAsText())
                             .append(' ')
                             .append(start.monthOfYear().getAsText())
                             .append(' ')
                             .append(start.year().getAsText())
                             .append(' ')
                             .append(start.hourOfDay().getAsText())
                             .append(':')
                             .append(start.minuteOfHour().getAsText())
                             .append(" - ")
                             .append(end.dayOfMonth().getAsText())
                             .append(' ')
                             .append(end.monthOfYear().getAsText())
```

```
                        .append(' ')
                        .append(end.year().getAsText())
                        .append(' ')
                        .append(end.hourOfDay().getAsText())
                        .append(':')
                        .append(end.minuteOfHour().getAsText());
            }
            return stringBuilder.toString();
        }

        @Override
        public DHXEv createEvent(String id, DHXStatus status) {
            AppointmentEvent newEvent = new AppointmentEvent();
            return newEvent;
        }

    }
```

- **web-pcs/src/org.leaf.cms.action.profile/ProfileAction.java**

```
    package org.leaf.cms.action.profile;

    import java.io.File;
    import java.util.ArrayList;
    import java.util.Date;
    import java.util.HashSet;
    import java.util.List;
    import java.util.Set;

    import org.apache.commons.io.FileUtils;
    import org.joda.time.DateTime;
    import org.leaf.cms.action.BaseLoggedInAction;
    import org.leaf.cms.action.interfaces.ConfigurationAware;
    import org.leaf.cms.constants.ApplicationConstants;
    import org.leaf.cms.model.ClinicInfo;
    import org.leaf.cms.model.Configuration;
    import org.leaf.cms.model.Physician;
    import org.leaf.cms.model.Specialty;
    import org.leaf.cms.service.PhysicianService;
    import org.leaf.cms.service.SpecialtyService;
    import org.slf4j.Logger;
    import org.slf4j.LoggerFactory;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.beans.factory.annotation.Qualifier;
    import org.springframework.beans.factory.annotation.Value;
    import org.springframework.data.domain.Page;

    import com.opensymphony.xwork2.Preparable;

    public class ProfileAction extends BaseLoggedInAction implements ConfigurationAware,
    Preparable{

        private static final Logger _logger =
    LoggerFactory.getLogger(ProfileAction.class);
        private static final long serialVersionUID = 1L;
        private Configuration configuration;
        private PhysicianService physicianService;

        private Long id;
        private Physician physician;
        private List<Physician> physicianList;

        private String firstName;
        private String lastName;
        private String cellphoneNumber;
        private Long specialtyId;
        private List<Long> specialtyIds;
        private List<Specialty> specialtyList;


        private Long medLicenseId;
        private Page<Physician> physicianPage;

        private String[] clinicName;
        private String[] clinicAddress;
        private String[] clinicTelNo;
```

```
    private File image;
    private String imageFileName;
    private String imageContentType;

    @Value("${images.files.location}")
    private String filesLocation;
    private SpecialtyService specialtyService;

    @Autowired
    public ProfileAction(@Qualifier("physicianService") PhysicianService
physicianService,
                         @Qualifier("specialtyService") SpecialtyService
specialtyService) {
            this.physicianService = physicianService;
            this.specialtyService = specialtyService;

    }

    @Override
    public void prepare() throws Exception {
            this.specialtyList = specialtyService.getSpecialty();
    }

    @Override
    public Configuration getConfiguration() {
            return this.configuration;
    }

    public String execute(){

    setPhysician(physicianService.findPhysician(getCurrentLoggedInUser().getId()));
            return SUCCESS;
    }

    public String edit(){

    setPhysician(physicianService.findPhysician(getCurrentLoggedInUser().getId()));
            return INPUT;
    }

    public String removeProfileImage(){
            if(id == null){
                    addActionError("Id should not be null");
                    return ERROR;
            }
            physician = physicianService.findPhysician(id);
            physician.setImageUrl(null);
            physicianService.save(physician);
            addActionMessage("Successfully remove profile image");
            return SUCCESS;
    }

    public String save(){
            physician =
physicianService.findPhysician(getCurrentLoggedInUser().getId());
            physician.setCellphoneNumber(cellphoneNumber);
            physician.setFirstName(firstName);
            physician.setLastName(lastName);
            /*physician.setMedLicenseId(medLicenseId);*/
            Set<Specialty> specialties = new HashSet<Specialty>();

        for(Long id: specialtyIds){
            specialties.add(specialtyService.findSpecialty(id));
        }

        physician.setSpecialties(specialties);

            if(image != null){
                     File filesDirectory = new
File(request.getSession().getServletContext().getRealPath("images/"));
                     if (!filesDirectory.exists()) {
                filesDirectory.mkdir();
            }

                    File file = image;
```

```
            String fileName = imageFileName;

            File savedFile = null;
            do {
                savedFile = new File(filesDirectory, fileName);
            } while (savedFile.exists());

            try {
                FileUtils.moveFile(file, savedFile);
            } catch (Exception e) {
                if (_logger.isErrorEnabled()) {
                    _logger.error(e.getMessage(), e);
                }
            }

            physician.setImageUrl("images/" + fileName);
            }

            List<ClinicInfo> infos = new ArrayList<ClinicInfo>();
            if(clinicName !=  null && clinicAddress != null && clinicTelNo != null
&& clinicName.length > 0 && clinicAddress.length > 0 && clinicTelNo.length > 0){

                    for(int i = 0 ; i < clinicName.length ; i ++){
                            String name = clinicName[i];
                            String address = clinicAddress[i];
                            String telNo = clinicTelNo[i];
                            ClinicInfo info = new ClinicInfo();
                            info.setName(name);
                            info.setAddress(address);
                            info.setTelNo(telNo);
                            info.setCreatedOn(new DateTime());
                            info.setIsValid(Boolean.TRUE);
                            info.setPhysician(physician);
                            info.setUpdatedOn(new Date());
                            infos.add(info);
                    }
            }
            physicianService.saveWithClinicInfo(physician, infos);
            return SUCCESS;

    }

    public String view(){
            if(getId() != null){
                    setPhysician(physicianService.findPhysician(getId()));
            }
            else{
                    addActionError(ApplicationConstants.ERR_MISSING_PARAMETER);
            }

            return "view";
    }

    public String profiles(){
            return SUCCESS;
    }

    public String loadphysicians(){
            String page = request.getParameter("page");
            physicianPage = physicianService.findAllActivatedWithPaging(page == null
? 1 : Integer.parseInt(page), ApplicationConstants.DEFAULT_ITEMS_PER_PAGE);
            return SUCCESS;

    }

    @Override
    public void setConfiguration(Configuration configuration) {
            this.configuration = configuration;
    }

    public List<Physician> getPhysicianList() {
            return physicianList;
    }

    public void setPhysicianList(List<Physician> physicianList) {
            this.physicianList = physicianList;
```

```
}

public Page<Physician> getPhysicianPage() {
        return physicianPage;
}

public void setPhysicianPage(Page<Physician> physicianPage) {
        this.physicianPage = physicianPage;
}

public Physician getPhysician() {
        return physician;
}

public void setPhysician(Physician physician) {
        this.physician = physician;
}

public Long getId() {
        return id;
}

public void setId(Long id) {
        this.id = id;
}

public String[] getClinicName() {
        return clinicName;
}

public void setClinicName(String[] clinicName) {
        this.clinicName = clinicName;
}

public String[] getClinicAddress() {
        return clinicAddress;
}

public void setClinicAddress(String[] clinicAddress) {
        this.clinicAddress = clinicAddress;
}

public String[] getClinicTelNo() {
        return clinicTelNo;
}

public void setClinicTelNo(String[] clinicTelNo) {
        this.clinicTelNo = clinicTelNo;
}

public Long getSpecialtyId() {
        return specialtyId;
}

public void setSpecialtyId(Long specialtyId) {
        this.specialtyId = specialtyId;
}

public String getFirstName() {
        return firstName;
}

public void setFirstName(String firstName) {
        this.firstName = firstName;
}

public String getLastName() {
        return lastName;
}

public void setLastName(String lastName) {
        this.lastName = lastName;
}

public String getCellphoneNumber() {
        return cellphoneNumber;
```

```
    }

    public void setCellphoneNumber(String cellphoneNumber) {
            this.cellphoneNumber = cellphoneNumber;
    }

    public File getImage() {
            return image;
    }

    public void setImage(File image) {
            this.image = image;
    }

    public String getImageFileName() {
            return imageFileName;
    }

    public void setImageFileName(String imageFileName) {
            this.imageFileName = imageFileName;
    }

    public String getImageContentType() {
            return imageContentType;
    }

    public void setImageContentType(String imageContentType) {
            this.imageContentType = imageContentType;
    }

    public List<Long> getSpecialtyIds() {
            return specialtyIds;
    }

    public void setSpecialtyIds(List<Long> specialtyIds) {
            this.specialtyIds = specialtyIds;
    }

    public List<Specialty> getSpecialtyList() {
            return specialtyList;
    }

    public void setSpecialtyList(List<Specialty> specialtyList) {
            this.specialtyList = specialtyList;
    }

    public Long getMedLicenseId() {
            return medLicenseId;
    }

    public void setMedLicenseId(Long medLicenseId) {
            this.medLicenseId = medLicenseId;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.references/ReferenceAction.java**

```
package org.leaf.cms.action.references;

import java.util.Date;

import org.joda.time.DateTime;
import org.leaf.cms.action.interfaces.ConfigurationAware;
import org.leaf.cms.action.manage.AbstractFileMaintenanceClass;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.model.Reference;
import org.leaf.cms.model.repository.ReferenceRepository;
import org.leaf.cms.service.ReferenceService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.domain.Page;

public class ReferenceAction  extends AbstractFileMaintenanceClass<Reference, Long,
ReferenceRepository> implements ConfigurationAware{
```

```
    private String title;
    private String description;
    private String url;


    private Page<Reference> referencePage;
    private ReferenceService referenceService;

    @Autowired
    public ReferenceAction(
                    @Qualifier("referenceRepository") ReferenceRepository
referenceRepository,
                    @Qualifier("referenceService") ReferenceService referenceService
    )
    {
            super(referenceRepository);
            this.referenceService = referenceService;
    }

    @Override
    protected void initializeEntity() {
            if (getId() == null) {
            setEntity(new Reference());
            entity.setCreatedOn(new DateTime());
            entity.setIsValid(Boolean.TRUE);
        } else {
            entity = repository.findOne(getId());
            entity.setTitle(getTitle());
            entity.setDescription(getDescription());
            entity.setUrl(getUrl());
            entity.setUpdatedOn(new Date());
        }
    }

    @Override
    public String delete() {
        String validActionStatus = validateAction();
        if (!validActionStatus.equalsIgnoreCase("success")) {
            return validActionStatus;
        }
        if (getEntity().getId() != null) {
            beforeDelete();
            repository.delete(getEntity());
            afterDelete();
        } else {
            addActionError("Id was not provided");
            return "error";
        }
        return "success";
    }

    public String execute(){
            return SUCCESS;
    }

    public String view(){
            return "view";
    }

    public String loadReferences(){
            System.out.println("Method executed");

            Integer page = Integer.parseInt(request.getParameter("page"));
            referencePage = referenceService.findAllWithPaging(page == null ? 1 :
page, ApplicationConstants.DEFAULT_ITEMS_PER_PAGE);
            System.out.println("Showing Records: " + referencePage.getSize());
            return SUCCESS;
    }
    private static final long serialVersionUID = 1L;

    private Configuration configuration;

    @Override
    public Configuration getConfiguration() {
            return configuration;
```

```
    }

    @Override
    public void setConfiguration(Configuration configuration) {
            this.configuration = configuration;
    }

    public String getTitle() {
            return title;
    }

    public void setTitle(String title) {
            this.title = title;
    }


    public String getDescription() {
            return description;
    }

    public void setDescription(String description) {
            this.description = description;
    }

    public String getUrl() {
            return url;
    }

    public void setUrl(String url) {
            this.url = url;
    }

    public Page<Reference> getReferencePage() {
            return referencePage;
    }

    public void setReferencePage(Page<Reference> referencePage) {
            this.referencePage = referencePage;
    }

    }
```

- **web-pcs/src/org.leaf.cms.action.report/ReportAction.java**

```
package org.leaf.cms.action.report;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.geom.Rectangle2D;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import jersey.repackaged.com.google.common.collect.Lists;

import org.apache.commons.io.FileUtils;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.chart.renderer.category.StandardBarPainter;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.category.DefaultCategoryDataset;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;
import org.json.simple.JSONObject;
import org.leaf.cms.action.BaseLoggedInAction;
```

```
import org.leaf.cms.enums.ClinicalCaseStatusEnum;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.PhysicianService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import com.itextpdf.awt.PdfGraphics2D;
import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Element;
import com.itextpdf.text.Font;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.PdfContentByte;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfTemplate;
import com.itextpdf.text.pdf.PdfWriter;

public class ReportAction extends BaseLoggedInAction {

    private static final Logger LOGGER = LoggerFactory.getLogger(ReportAction.class);

    private static final long serialVersionUID = 1L;

    private static final Font TITLE_FONT = new Font(Font.FontFamily.HELVETICA, 16,
Font.BOLD);
    private static final Font BODY_FONT = new Font(Font.FontFamily.HELVETICA, 8,
Font.NORMAL);
    private static final String REPORT_PREFIX = "REPORT";
    private static final String REPORT_SUFFIX = ".pdf";

    private PhysicianService physicianService;
    private AppointmentService appointmentService;
    private Connection connection;
    private String startDate;
    private String endDate;
    private String dataReport;
    private List<Appointment> appointments = new ArrayList<>();
    private String contentType;
    private String fileName;
    private String selectionString;
    private Physician physician;

    @Autowired
    public ReportAction(PhysicianService physicianService, AppointmentService
appointmentService) {
        this.physicianService = physicianService;
        this.appointmentService = appointmentService;
    }

    public String generateDataReportOfRequestingPhysician() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        physician = physicianService.findPhysician(getCurrentLoggedInUser().getId());

        appointments =
appointmentService.getAppointmentsAsRequestingPhysician(physician,
parsedStartDate.toLocalDate(), parsedEndDate.toLocalDate());

        for (Appointment appointment : appointments) {
            ClinicalCase clinicalCase = appointment.getClinicalCase();
            if
(clinicalCase.getStatus().equalsIgnoreCase(ClinicalCaseStatusEnum.ONGOING.getStatus()
)
```

```
                    ||
clinicalCase.getStatus().equalsIgnoreCase(ClinicalCaseStatusEnum.ACTIVE.getStatus()))
{
                if (!appointment.getIsValid()) {
                    ClinicalCase tempClinicalCase = new ClinicalCase();

tempClinicalCase.setPatientDiagnosis(clinicalCase.getPatientDiagnosis());

tempClinicalCase.setPatientLabExam(clinicalCase.getPatientLabExam());
                    tempClinicalCase.setQuestion(clinicalCase.getQuestion());
                    tempClinicalCase.setRemarks(clinicalCase.getRemarks());
                    tempClinicalCase.setSpecialty(clinicalCase.getSpecialty());

tempClinicalCase.setStatus(ClinicalCaseStatusEnum.TERMINATED.getStatus());
                    appointment.setClinicalCase(tempClinicalCase);
                }
            }
        }
        return SUCCESS;
    }

    public String generateDataReportOfRequestingPhysicianInPdf() {
        generateDataReportOfRequestingPhysician();
        try {
            Document document = new Document();
            File tempFile = File.createTempFile(REPORT_PREFIX, REPORT_SUFFIX);
            PdfWriter.getInstance(document, new FileOutputStream(tempFile));
            document.open();

            createPdfMetadata(document, "Requesting Physician Report");

            Paragraph preface = new Paragraph();

            String title = new StringBuilder().append("Report from ")
                                              .append(startDate)
                                              .append(" to ")
                                              .append(endDate)
                                              .append(" as Requesting Physician")
                                              .toString();
            Paragraph prefaceTitle = new Paragraph(title, TITLE_FONT);
            prefaceTitle.setAlignment(Paragraph.ALIGN_CENTER);
            preface.add(prefaceTitle);

            preface.add(new Paragraph(" "));

            String name = new StringBuilder().append("Physician Name:
").append(physician.getFullName()).toString();
            Paragraph prefaceName = new Paragraph(name, BODY_FONT);
            prefaceName.setAlignment(Paragraph.ALIGN_LEFT);
            preface.add(prefaceName);

            StringBuilder specialties = new StringBuilder("Specialty: ");
            Set<Specialty> specialtiesSet = physician.getSpecialties();
            Iterator<Specialty> specialtiesIterator = specialtiesSet.iterator();
            while (specialtiesIterator.hasNext()) {
                specialties.append(specialtiesIterator.next().getDescription());
                if (specialtiesIterator.hasNext()) {
                    specialties.append("; ");
                }
            }
            Paragraph prefaceSpecialties = new Paragraph(specialties.toString(),
BODY_FONT);
            prefaceName.setAlignment(Paragraph.ALIGN_LEFT);
            preface.add(prefaceSpecialties);

            preface.add(new Paragraph(" "));

            document.add(preface);

            PdfPTable table = new PdfPTable(5);
            table.setWidthPercentage(100f);
            table.setWidths(new float[] { 35f, 25f, 15f, 15f, 10f });

            PdfPCell header1 = new PdfPCell(new Phrase("Clinical Question",
BODY_FONT));
            header1.setHorizontalAlignment(Element.ALIGN_CENTER);
```

```
            table.addCell(header1);

            PdfPCell header2 = new PdfPCell(new Phrase("Consultant", BODY_FONT));
            header2.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header2);

            PdfPCell header3 = new PdfPCell(new Phrase("Date of Consult",
BODY_FONT));
            header3.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header3);

            PdfPCell header4 = new PdfPCell(new Phrase("Time", BODY_FONT));
            header4.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header4);

            PdfPCell header5 = new PdfPCell(new Phrase("Status", BODY_FONT));
            header5.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header5);

            table.setHeaderRows(1);

            for (Appointment appointment : appointments) {
                ClinicalCase clinicalCase = appointment.getClinicalCase();
                table.addCell(new Phrase(clinicalCase.getQuestion(), BODY_FONT));
                table.addCell(new Phrase(appointment.getConsultant().getFullName(),
BODY_FONT));
                PdfPCell cell3 = new PdfPCell(new
Phrase(appointment.getConsultationDate().toString(), BODY_FONT));
                cell3.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell3);
                PdfPCell cell4 = new PdfPCell(new
Phrase(appointment.getConsultationStart().toLocalTime().toString("HH:mm:ss") + " - "
                        +
appointment.getConsultationEnd().toLocalTime().toString("HH:mm:ss"), BODY_FONT));
                cell4.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell4);
                PdfPCell cell5 = new PdfPCell(new Phrase(clinicalCase.getStatus(),
BODY_FONT));
                cell5.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell5);
            }
            document.add(table);

            document.close();

            inputStream = FileUtils.openInputStream(tempFile);
            fileName = tempFile.getName();
            contentType = "application/pdf";
        } catch (FileNotFoundException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (DocumentException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (IOException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        }
        return SUCCESS;
    }

    public String generateDataReportOfConsultant() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        physician = physicianService.findPhysician(getCurrentLoggedInUser().getId());

        appointments = appointmentService.getAppointmentsAsConsultant(physician,
parsedStartDate.toLocalDate(), parsedEndDate.toLocalDate());

        for (Appointment appointment : appointments) {
```

```
            ClinicalCase clinicalCase = appointment.getClinicalCase();
            if
(clinicalCase.getStatus().equalsIgnoreCase(ClinicalCaseStatusEnum.ONGOING.getStatus()
)
                    ||
clinicalCase.getStatus().equalsIgnoreCase(ClinicalCaseStatusEnum.ACTIVE.getStatus()))
{
                if (!appointment.getIsValid()) {
                    ClinicalCase tempClinicalCase = new ClinicalCase();

tempClinicalCase.setPatientDiagnosis(clinicalCase.getPatientDiagnosis());

tempClinicalCase.setPatientLabExam(clinicalCase.getPatientLabExam());
                    tempClinicalCase.setQuestion(clinicalCase.getQuestion());
                    tempClinicalCase.setRemarks(clinicalCase.getRemarks());
                    tempClinicalCase.setSpecialty(clinicalCase.getSpecialty());

tempClinicalCase.setStatus(ClinicalCaseStatusEnum.TERMINATED.getStatus());
                    appointment.setClinicalCase(tempClinicalCase);
                }
            }
        }
        return SUCCESS;
    }

    public String generateDataReportOfConsultantInPdf() {
        generateDataReportOfConsultant();
        try {
            Document document = new Document();
            File tempFile = File.createTempFile(REPORT_PREFIX, REPORT_SUFFIX);
            PdfWriter.getInstance(document, new FileOutputStream(tempFile));
            document.open();

            createPdfMetadata(document, "Consultant Report");

            Paragraph preface = new Paragraph();

            String title = new StringBuilder().append("Report from ")
                                              .append(startDate)
                                              .append(" to ")
                                              .append(endDate)
                                              .append(" as Consultant")
                                              .toString();
            Paragraph prefaceTitle = new Paragraph(title, TITLE_FONT);
            prefaceTitle.setAlignment(Paragraph.ALIGN_CENTER);
            preface.add(prefaceTitle);

            preface.add(new Paragraph(" "));

            String name = new StringBuilder().append("Physician Name:
").append(physician.getFullName()).toString();
            Paragraph prefaceName = new Paragraph(name, BODY_FONT);
            prefaceName.setAlignment(Paragraph.ALIGN_LEFT);
            preface.add(prefaceName);

            StringBuilder specialties = new StringBuilder("Specialty: ");
            Set<Specialty> specialtiesSet = physician.getSpecialties();
            Iterator<Specialty> specialtiesIterator = specialtiesSet.iterator();
            while (specialtiesIterator.hasNext()) {
                specialties.append(specialtiesIterator.next().getDescription());
                if (specialtiesIterator.hasNext()) {
                    specialties.append("; ");
                }
            }
            Paragraph prefaceSpecialties = new Paragraph(specialties.toString(),
BODY_FONT);
            prefaceName.setAlignment(Paragraph.ALIGN_LEFT);
            preface.add(prefaceSpecialties);

            preface.add(new Paragraph(" "));

            document.add(preface);

            PdfPTable table = new PdfPTable(5);
            table.setWidthPercentage(100f);
            table.setWidths(new float[] { 35f, 25f, 15f, 15f, 10f });
```

```
            PdfPCell header1 = new PdfPCell(new Phrase("Clinical Question",
BODY_FONT));
            header1.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header1);

            PdfPCell header2 = new PdfPCell(new Phrase("Requesting Physician",
BODY_FONT));
            header2.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header2);

            PdfPCell header3 = new PdfPCell(new Phrase("Date of Consult",
BODY_FONT));
            header3.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header3);

            PdfPCell header4 = new PdfPCell(new Phrase("Time", BODY_FONT));
            header4.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header4);

            PdfPCell header5 = new PdfPCell(new Phrase("Status", BODY_FONT));
            header5.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header5);

            table.setHeaderRows(1);

            for (Appointment appointment : appointments) {
                ClinicalCase clinicalCase = appointment.getClinicalCase();
                table.addCell(new Phrase(clinicalCase.getQuestion(), BODY_FONT));
                table.addCell(new
Phrase(appointment.getRequestingPhysician().getFullName(), BODY_FONT));
                PdfPCell cell3 = new PdfPCell(new
Phrase(appointment.getConsultationDate().toString(), BODY_FONT));
                cell3.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell3);
                PdfPCell cell4 = new PdfPCell(new
Phrase(appointment.getConsultationStart().toLocalTime().toString("HH:mm:ss") + " - "
                        +
appointment.getConsultationEnd().toLocalTime().toString("HH:mm:ss"), BODY_FONT));
                cell4.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell4);
                PdfPCell cell5 = new PdfPCell(new Phrase(clinicalCase.getStatus(),
BODY_FONT));
                cell5.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell5);
            }
            document.add(table);

            document.close();

            inputStream = FileUtils.openInputStream(tempFile);
            fileName = tempFile.getName();
            contentType = "application/pdf";
        } catch (FileNotFoundException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (DocumentException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (IOException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        }
        return SUCCESS;
    }

    @SuppressWarnings("unchecked")
    public String generateAdminDataReportOfRequestingPhysician() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);
```

```
        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfRequestingPhysicians(

parsedStartDate.toLocalDate(),

parsedEndDate.toLocalDate());
        JSONObject jsonDataReport = new JSONObject();
        int ctr = 1;
        for (Object[] data : dataReports) {
            jsonDataReport.put("lastname" + ctr, data[0]);
            jsonDataReport.put("firstname" + ctr, data[1]);
            jsonDataReport.put("count" + ctr, data[2]);
            ctr++;
        }
        dataReport = jsonDataReport.toJSONString();
        return SUCCESS;
    }

    public String generateAdminDataReportOfRequestingPhysicianInBarChartPdf() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfRequestingPhysicians(

parsedStartDate.toLocalDate(),

parsedEndDate.toLocalDate());
        generateBarChartPdf(dataReports, "Requesting Physician");
        return SUCCESS;
    }

    public String generateAdminDataReportOfRequestingPhysicianInTablePdf() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfRequestingPhysicians(

parsedStartDate.toLocalDate(),

parsedEndDate.toLocalDate());
        try {
            Document document = new Document();
            File tempFile = File.createTempFile(REPORT_PREFIX, REPORT_SUFFIX);
            PdfWriter.getInstance(document, new FileOutputStream(tempFile));
            document.open();

            createPdfMetadata(document, "Requesting Physician Report");

            Paragraph preface = new Paragraph();
            String title = "Requesting Physician report between " + startDate + " to
" + endDate;
            Paragraph prefaceTitle = new Paragraph(title, TITLE_FONT);
            prefaceTitle.setAlignment(Paragraph.ALIGN_CENTER);
            preface.add(prefaceTitle);
            preface.add(new Paragraph(" "));
            document.add(preface);

            PdfPTable table = new PdfPTable(2);
            table.setWidthPercentage(100f);
            table.setWidths(new float[] { 90f, 10f });

            PdfPCell header1 = new PdfPCell(new Phrase("Physician", BODY_FONT));
            header1.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header1);

            PdfPCell header2 = new PdfPCell(new Phrase("Count", BODY_FONT));
            header2.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header2);

            table.setHeaderRows(1);
```

```
            for (Object[] data : dataReports) {
                table.addCell(new Phrase(data[0] + ", " + data[1], BODY_FONT));
                PdfPCell cell2 = new PdfPCell(new Phrase(data[2].toString(),
BODY_FONT));
                cell2.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell2);
            }
            document.add(table);

            document.close();

            inputStream = FileUtils.openInputStream(tempFile);
            fileName = tempFile.getName();
            contentType = "application/pdf";
        } catch (FileNotFoundException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (DocumentException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (IOException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        }
        return SUCCESS;
    }

    @SuppressWarnings("unchecked")
    public String generateAdminDataReportOfConsultant() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfConsultants(parsedStartDate.toLocalDate(),
parsedEndDate.toLocalDate());
        JSONObject jsonDataReport = new JSONObject();
        int ctr = 1;
        for (Object[] data : dataReports) {
            jsonDataReport.put("lastname" + ctr, data[0]);
            jsonDataReport.put("firstname" + ctr, data[1]);
            jsonDataReport.put("count" + ctr, data[2]);
            ctr++;
        }
        dataReport = jsonDataReport.toJSONString();
        return SUCCESS;
    }

    public String generateAdminDataReportOfConsultantInBarChartPdf() {
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfConsultants(parsedStartDate.toLocalDate(),
parsedEndDate.toLocalDate());
        generateBarChartPdf(dataReports, "Consultant");
        return SUCCESS;
    }

    private void generateBarChartPdf(List<Object[]> dataReports, String subject) {
        try {
            Document document = new Document();
            File tempFile = File.createTempFile(REPORT_PREFIX, REPORT_SUFFIX);
            PdfWriter writer = PdfWriter.getInstance(document, new
FileOutputStream(tempFile));
            document.open();

            createPdfMetadata(document, subject + " Report");
```

```
            String title = subject + " report between " + startDate + " to " +
endDate;

            List<List<Object[]>> dataReportsPartition = Lists.partition(dataReports,
22);

            boolean newpage = false;
            for (List<Object[]> dataReportsList : dataReportsPartition) {
                if (newpage) {
                    document.newPage();
                }
                PdfContentByte contentByte = writer.getDirectContent();
                contentByte.saveState();
                float width = PageSize.A4.getWidth();
                float height = 34 * (dataReportsList.size() + 2);

                PdfTemplate bar = contentByte.createTemplate(width, height);
                Graphics2D g2d2 = new PdfGraphics2D(bar, width, height);
                Rectangle2D r2d2 = new Rectangle2D.Double(0, 0, width, height);

                DefaultCategoryDataset dataset = new DefaultCategoryDataset();
                for (Object[] data : dataReportsList) {
                    dataset.addValue(Integer.parseInt(data[2].toString()), "count",
data[0] + ", " + data[1]);
                }

                JFreeChart barChart = ChartFactory.createBarChart(null, null, null,
dataset, PlotOrientation.HORIZONTAL, false, false, false);
                TextTitle textTitle = new TextTitle(title);
                barChart.setTitle(textTitle);
                barChart.setBackgroundPaint(Color.WHITE);
                barChart.setBorderVisible(false);
                CategoryPlot categoryPlot = barChart.getCategoryPlot();
                NumberAxis numberAxis = (NumberAxis) categoryPlot.getRangeAxis();
                numberAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
                categoryPlot.setBackgroundPaint(Color.WHITE);
                categoryPlot.setRangeGridlinePaint(Color.WHITE);
                categoryPlot.setOutlineVisible(false);
                BarRenderer barRenderer = (BarRenderer) categoryPlot.getRenderer();
                barRenderer.setSeriesPaint(0, new Color(151, 187, 205, (int) (0.5f *
255)));
                barRenderer.setSeriesOutlinePaint(0, new Color(151, 187, 205, (int)
(0.8f * 255)));
                barRenderer.setDrawBarOutline(true);
                barRenderer.setShadowVisible(false);
                barRenderer.setGradientPaintTransformer(null);
                barRenderer.setBarPainter(new StandardBarPainter());
                barChart.draw(g2d2, r2d2);
                g2d2.dispose();
                contentByte.addTemplate(bar, 0, PageSize.A4.getHeight() - height -
10);
                contentByte.restoreState();
                newpage = true;
            }

            document.close();

            inputStream = FileUtils.openInputStream(tempFile);
            fileName = tempFile.getName();
            contentType = "application/pdf";
        } catch (FileNotFoundException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (DocumentException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (IOException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        }
    }

    public String generateAdminDataReportOfConsultantInTablePdf() {
```

```
        DateTimeFormatter formatter = DateTimeFormat.forPattern("MM/dd/yyyy");

        DateTime parsedStartDate = DateTime.parse(startDate, formatter);
        DateTime parsedEndDate = DateTime.parse(endDate, formatter);

        List<Object[]> dataReports =
appointmentService.getAppointmentReportOfConsultants(parsedStartDate.toLocalDate(),
parsedEndDate.toLocalDate());
        try {
            Document document = new Document();
            File tempFile = File.createTempFile(REPORT_PREFIX, REPORT_SUFFIX);
            PdfWriter.getInstance(document, new FileOutputStream(tempFile));
            document.open();

            createPdfMetadata(document, "Consultant Report");

            Paragraph preface = new Paragraph();
            String title = "Consultant report between " + startDate + " to " +
endDate;
            Paragraph prefaceTitle = new Paragraph(title, TITLE_FONT);
            prefaceTitle.setAlignment(Paragraph.ALIGN_CENTER);
            preface.add(prefaceTitle);
            preface.add(new Paragraph(" "));
            document.add(preface);

            PdfPTable table = new PdfPTable(2);
            table.setWidthPercentage(100f);
            table.setWidths(new float[] { 90f, 10f });

            PdfPCell header1 = new PdfPCell(new Phrase("Physician", BODY_FONT));
            header1.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header1);

            PdfPCell header2 = new PdfPCell(new Phrase("Count", BODY_FONT));
            header2.setHorizontalAlignment(Element.ALIGN_CENTER);
            table.addCell(header2);

            table.setHeaderRows(1);

            for (Object[] data : dataReports) {
                table.addCell(new Phrase(data[0] + ", " + data[1], BODY_FONT));
                PdfPCell cell2 = new PdfPCell(new Phrase(data[2].toString(),
BODY_FONT));
                cell2.setHorizontalAlignment(Element.ALIGN_CENTER);
                table.addCell(cell2);
            }
            document.add(table);

            document.close();

            inputStream = FileUtils.openInputStream(tempFile);
            fileName = tempFile.getName();
            contentType = "application/pdf";
        } catch (FileNotFoundException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (DocumentException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        } catch (IOException e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug(e.getMessage(), e);
            }
        }
        return SUCCESS;
    }

    private void createPdfMetadata(Document document, String subject) {
        document.addTitle(subject);
        document.addAuthor("Web-PCS");
        document.addCreator("Web-PCS");
        document.addCreationDate();
        document.addSubject(subject);
    }
```

```
    public Connection getConnection() {
        return connection;
    }

    public void setConnection(Connection connection) {
        this.connection = connection;
    }

    public String getStartDate() {
        return startDate;
    }

    public void setStartDate(String startDate) {
        this.startDate = startDate;
    }

    public String getEndDate() {
        return endDate;
    }

    public void setEndDate(String endDate) {
        this.endDate = endDate;
    }

    public String getDataReport() {
        return dataReport;
    }

    public void setDataReport(String dataReport) {
        this.dataReport = dataReport;
    }

    public List<Appointment> getAppointments() {
        return appointments;
    }

    public void setAppointments(List<Appointment> appointments) {
        this.appointments = appointments;
    }

    public String getContentType() {
        return contentType;
    }

    public void setContentType(String contentType) {
        this.contentType = contentType;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public String getSelectionString() {
        return selectionString;
    }

    public void setSelectionString(String selectionString) {
        this.selectionString = selectionString;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.sms/SMSPushNotificationAction.java**

```
package org.leaf.cms.action.sms;

import java.io.ByteArrayInputStream;
import java.util.Iterator;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.leaf.cms.action.BaseAction;
```

```
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.SMSEntry;
import org.leaf.cms.service.SMSEntryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.domain.Page;

public class SMSPushNotificationAction extends BaseAction{

    private static final long serialVersionUID = 1L;
    private SMSEntryService smsEntryService;

    @Autowired
    public SMSPushNotificationAction(
                    @Qualifier("smsEntryService") SMSEntryService smsEntryService
    )
    {
            this.smsEntryService = smsEntryService;
    }

    @SuppressWarnings("unchecked")
    public String inquireSMS() {
            System.out.println("========STARTING SMS=========");
        System.out.println(request.getParameter("action"));
        System.out.println(request.getParameter("message"));

        String action = request.getParameter("action");
        if(action.equalsIgnoreCase("send_status")){
            System.out.println("SEND STATUS");
            Long id = Long.parseLong(request.getParameter("id"));
            SMSEntry smsEntry = smsEntryService.findById(id);
            if(smsEntry != null){
                    smsEntry.setIsSent(Boolean.TRUE);
                    smsEntryService.save(smsEntry);
            }
        }
        Page<SMSEntry> page = smsEntryService.findAllWithPaging(1,
ApplicationConstants.DEFAULT_ITEMS_PER_PAGE);
        JSONObject obj = new JSONObject();
        if (page != null) {
            SMSEntry smsEntry;

            JSONArray events = new JSONArray();
            JSONObject event = new JSONObject();
            JSONArray messagesArray = new JSONArray();
            event.put("event", "send");

            for (Iterator<SMSEntry> iterator = page.iterator(); iterator.hasNext(); )
{
                smsEntry = iterator.next();

                JSONObject message = new JSONObject();
                message.put("id", smsEntry.getId().toString());// for retrieval
purposes
                message.put("to", smsEntry.getRecipient());
                message.put("message", smsEntry.getMessage());
                messagesArray.add(message);
            }

            event.put("messages", messagesArray);
            events.add(event);
            obj.put("events", events);
            inputStream = new ByteArrayInputStream(obj.toJSONString().getBytes());
        }
        else{
            obj.put("d", "success");
            inputStream = new ByteArrayInputStream(obj.toJSONString().getBytes());
            return SUCCESS;
        }

        return SUCCESS;

    }
}
```

- **web-pcs/src/org.leaf.cms.action.interfaces/Id.java**

```
package org.leaf.cms.commons.interfaces;

import java.io.Serializable;

public interface Id<T extends Serializable>
{
    /**
     * Get the id of the entity
     *
     * @return the id of the entity
     */
    public T getId();

    /**
     * Set the id of the entity
     *
     * @param id the id to be set.
     */
    public void setId(T id);
}
```

- **web-pcs/src/org.leaf.cms.action.interfaces/JsonConvertible.java**

```
package org.leaf.cms.commons.interfaces;

import org.json.simple.JSONObject;

public interface JsonConvertible {
    public JSONObject toJSONObject();
}
```

- **web-pcs/src/org.leaf.cms.action.model/BaseEntity.java**

```
package org.leaf.cms.commons.model;

import java.io.Serializable;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.MappedSuperclass;
import javax.persistence.Transient;

import org.hibernate.annotations.Type;
import org.joda.time.DateTime;

@MappedSuperclass
public abstract class BaseEntity<ID extends Serializable> extends BaseId<ID> {
    private Boolean isValid;
    private DateTime createdOn;

    @Basic
    @Column(name = "valid", nullable = false)
    public Boolean getIsValid() {
        if (isValid == null) {
            return Boolean.FALSE;
        }
        return isValid;
    }

    public void setIsValid(Boolean isValid) {
        this.isValid = isValid;
    }

    @Transient
    public Boolean isValid() {
        return getIsValid();
    }

    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    @Column(name = "created_on", nullable = false)
    public DateTime getCreatedOn() {
        return createdOn;
    }
```

```
    public void setCreatedOn(DateTime createdOn) {
        this.createdOn = createdOn;
    }

    @Override
    public String toString() {
        return "BaseEntity [id=" + id + ", isValid=" + isValid + ", createdOn=" +
createdOn + "]";
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/BaseId.java**

```
package org.leaf.cms.commons.model;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class EntityList<T> implements Iterable<T>{

    private static final long serialVersionUID = 1L;

    private List<T> list;

    private int total;

    @Override
    public String toString()
    {
            String strlist = (this != null) ? this.toString() : "[]";
            StringBuilder sb = new StringBuilder(100 + strlist.length());

            sb.append("Total: " + getTotal()).append(", Size: " +
getList().size()).append(", List: ").append(
                            strlist);

            return sb.toString();
    }

    public int getTotal() {
            return total;
    }

    public void setTotal(int total) {
            this.total = total;
    }

    public Iterator<T> iterator() {
            return getList().iterator();
    }

    public List<T> getList() {
            if (list == null)
            {
                    list = new ArrayList<T>(5);
            }
            return list;
    }

    public void setList(List<T> list) {
            this.list = list;
    }

    public int getSize()
    {
            return getList().size();
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/BaseSequentialUpdatableEntity.java**

```
package org.leaf.cms.commons.model;
```

```
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.MappedSuperclass;



@MappedSuperclass
public abstract class BaseSequentialUpdatableEntity<ID extends Serializable> extends
BaseUpdatableEntity<ID> {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id", nullable = false)
    @Override
    public ID getId() {
        return id;
    }

    @Override
    public void setId(ID id) {
        this.id = id;
    }

    @Override
    public String toString() {
        return "BaseSequentialUpdatableEntity [id=" + id + "]";
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/BaseUpdatableEntity.java**

```
package org.leaf.cms.commons.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.MappedSuperclass;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Version;

@MappedSuperclass
public abstract class BaseUpdatableEntity<ID extends Serializable> extends
BaseEntity<ID> {
    private Date updatedOn;

    @Version
    @Temporal(value = TemporalType.TIMESTAMP)
    @Column(name = "updated_on", nullable = false)
    public Date getUpdatedOn() {
        return updatedOn;
    }

    public void setUpdatedOn(Date updatedOn) {
        this.updatedOn = updatedOn;
    }

    @Override
    public String toString() {
        return "BaseUpdatableEntity [id=" + id + ", updatedOn=" + updatedOn + "]";
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/EntityList.java**

```
package org.leaf.cms.commons.model;

import java.util.ArrayList;
import java.util.Iterator;
```

```
import java.util.List;

public class EntityList<T> implements Iterable<T>{

    private static final long serialVersionUID = 1L;

    private List<T> list;

    private int total;

    @Override
    public String toString()
    {
            String strlist = (this != null) ? this.toString() : "[]";
            StringBuilder sb = new StringBuilder(100 + strlist.length());

            sb.append("Total: " + getTotal()).append(", Size: " +
getList().size()).append(", List: ").append(
                            strlist);

            return sb.toString();
    }

    public int getTotal() {
            return total;
    }

    public void setTotal(int total) {
            this.total = total;
    }

    public Iterator<T> iterator() {
            return getList().iterator();
    }

    public List<T> getList() {
            if (list == null)
            {
                    list = new ArrayList<T>(5);
            }
            return list;
    }

    public void setList(List<T> list) {
            this.list = list;
    }

    public int getSize()
    {
            return getList().size();
    }

}
```

- **web-pcs/src/org.leaf.cms.action.config/ApplicationConfig.java**

```
package org.leaf.cms.config;

import java.util.HashSet;
import java.util.Set;

import javax.ws.rs.core.Application;

import org.glassfish.jersey.media.multipart.MultiPartFeature;

public class ApplicationConfig extends Application {

    public Set<Class<?>> getClasses() {
        final Set<Class<?>> resources = new HashSet<Class<?>>();

        resources.add(MultiPartFeature.class);

        return resources;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.constants/ApplicationConstants.java**

```
package org.leaf.cms.constants;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

import org.apache.commons.io.IOUtils;
import org.leaf.cms.util.ClassHelper;

public class ApplicationConstants
{
    public static final String SESSION_USER_ID = "userid";
    private static final String PROPERTIES_FILE_NAME = "conf/application.properties";

    public static final String DEFAULT_CONNECTION_STRING;
    public static final String DEFAULT_USERNAME;
    public static final String DEFAULT_PASSWORD;
    public static final String DEFAULT_REST_CENTRAL_IP;

    public static final int DEFAULT_ITEMS_PER_PAGE = 5;

    public static final String ERR_MISSING_PARAMETER = "Failed to complete
transaction. Missing parameter.";
    public static final String ERR_USERNAME_MUST_BE_UNIQUE = "Username must be
unique.";
    public static final String ERR_MEDLICENSEID_MUST_BE_UNIQUE = "Medical License ID
must be unique.";
    public static final String ERR_FAILED_SAVE_PHYSICIAN = "Failed to register
physician.";
    public static final String ERR_FAILED_ACTIVATE_PHYSICIAN = "Failed to activitate
physician account.";
    public static final String SUC_SAVE_PHYSICIAN = "Created physician successfully.
Please wait for Administrator's confirmation.";
    public static final String SUC_ACTIVATE_PHYSICIAN = "Successfully activated
physician account.";
    public static final String ERR_FAILED_SAVE_DISCUSSION_THREAD = "Failed to save
discussion thread.";
    public static final String ERR_FAILED_SAVE_DISCUSSION_REPLY = "Failed to save
discussion reply.";
    public static final String SUC_SAVE_DISCUSSION_THREAD = "You have successfully
saved the discussion thread.";
    public static final String SUC_SAVE_DISCUSSION_REPLY = "You have successfully
added your reply.";
    public static final String ERR_FAILED_SAVE_ACCOUNT_SETTINGS = "Failed to save
user account settings.";
    public static final String SUC_SAVE_ACCOUNT_SETTINGS = "You have successfully
saved your user account settings.";
    public static final String SUC_CHANGE_PASSWORD = "An email will be sent with your
new password.";
    public static final String ERR_CHANGE_PASSWORD = "An error occurred in your
request. Either the username or the medical license Id was not found.";

    public static final String INIT_DISCUSSION_THREAD_SMS_TEMPLATE;

    static {
final InputStream inputStream =
ClassHelper.getResourceAsStream(PROPERTIES_FILE_NAME);

            if (inputStream == null) {throw new
IllegalStateException(PROPERTIES_FILE_NAME + " is missing."); }

            final Properties property = new Properties();
            try {property.load(inputStream);
            } catch (IOException e) {
                    throw new RuntimeException("Failed to read property file!", e);
            } finally {IOUtils.closeQuietly(inputStream);
            }

            DEFAULT_CONNECTION_STRING =
property.getProperty("default.mssql.db.connection.string");
            DEFAULT_USERNAME = property.getProperty("default.mssql.db.username");
            DEFAULT_PASSWORD = property.getProperty("default.mssql.db.password");
            DEFAULT_REST_CENTRAL_IP = property.getProperty("default.rest.central");
```

```
            INIT_DISCUSSION_THREAD_SMS_TEMPLATE =
property.getProperty("init.discussion.thread.sms");
    }
}
```

- **web-pcs/src/org.leaf.cms.action.enums/ClinicalCaseStatusEnum.java**

```
package org.leaf.cms.enums;

public enum ClinicalCaseStatusEnum {
    ACTIVE("Active"), ONGOING("Ongoing"), TERMINATED("Terminated"), CLOSED("Closed");

    ClinicalCaseStatusEnum(String status) {
        this.status = status;
    }

    private final String status;

    public String getStatus() {
        return status;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.enums/SubscriptionDigestTypeEnum.java**

```
package org.leaf.cms.enums;

public enum SubscriptionDigestTypeEnum {
    DAILY("daily"),
    WEEKLY("weekly");

    SubscriptionDigestTypeEnum(String name){
            this.name = name;
    }
    private final String name;

    public String getName() {
            return name;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.interceptor/BaseInterceptor.java**

```
package org.leaf.cms.interceptor;

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

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

public abstract class BaseInterceptor implements Interceptor{


private static final long serialVersionUID = 5233823346000885591;

    public void init() {
    }

    public void destroy() {
    }

    public abstract String intercept(ActionInvocation invocation) throws Exception;

    /**
     *
     * @return HttpServletRequest
     */
    protected HttpServletRequest getHttpServletRequest() {
            return ServletActionContext.getRequest();
    }

    /**
```

```
     *
     * @return HttpServletResponse
     */
    protected HttpServletResponse getHttpServletResponse() {
            return ServletActionContext.getResponse();
    }

    /**
     *
     * @return HttpSession
     */
    protected HttpSession getHttpSession() {
            return getHttpServletRequest().getSession();
    }
}
```

- **web-pcs/src/org.leaf.cms.action.interceptor/ConfigurationInterceptor.java**

```
package org.leaf.cms.interceptor;

import org.leaf.cms.action.interfaces.ConfigurationAware;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.service.ConfigurationService;

import com.opensymphony.xwork2.ActionInvocation;

public class ConfigurationInterceptor extends BaseInterceptor {

    private ConfigurationService configurationService;

    public ConfigurationInterceptor(ConfigurationService configurationService) {
        this.configurationService = configurationService;
    }

    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        final Object action = invocation.getAction();
        if (action instanceof ConfigurationAware) {
            Configuration configuration = configurationService.mainConfiguration();
            ((ConfigurationAware) action).setConfiguration(configuration);
        }
        return invocation.invoke();
    }

}
```

- **web-pcs/src/org.leaf.cms.action.interceptor/UserInterceptor.java**

```
package org.leaf.cms.interceptor;

import javax.servlet.http.HttpSession;

import org.leaf.cms.action.interfaces.CurrentLoggedInUserAware;
import org.leaf.cms.constants.ApplicationConstants;
import org.leaf.cms.model.User;
import org.leaf.cms.service.UserService;

import com.opensymphony.xwork2.ActionInvocation;

public class UserInterceptor extends BaseInterceptor {

    private static final long serialVersionUID = 1L;
    private UserService userService;

    public UserInterceptor(UserService userService) {
        this.userService = userService;
    }

    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        final Object action = invocation.getAction();
        final HttpSession session = getHttpSession();

        if (action instanceof CurrentLoggedInUserAware) {
            if( session.getAttribute(ApplicationConstants.SESSION_USER_ID) == null){
                    return "login";
```

```
            }

            User user = userService.findUser((Long)
session.getAttribute(ApplicationConstants.SESSION_USER_ID));
            if (user == null) {
                return "login";
            }
            ((CurrentLoggedInUserAware) action).putCurrentLoggedInUser(user);
        }
        return invocation.invoke();
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/Appointment.java**

```
package org.leaf.cms.model;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Type;
import org.hibernate.annotations.Where;
import org.joda.time.DateTime;
import org.joda.time.LocalDate;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name = "appointment")
@Table(name = Appointment.TABLE_NAME)
public class Appointment extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {
    public static final String TABLE_NAME = "appointment";

    private ClinicalCase clinicalCase;
    private Physician requestingPhysician;
    private Physician consultant;
    private LocalDate consultationDate;
    private DateTime consultationStart;
    private DateTime consultationEnd;
    private DateTime notificationSmsSchedule;
    private Boolean notificationSmsScheduleSent;
    private Boolean isSeenRequestingPhysician;
    private Boolean isSeenConsultant;

    @ManyToOne(targetEntity = ClinicalCase.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "clinical_case", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public ClinicalCase getClinicalCase() {
        return clinicalCase;
    }

    public void setClinicalCase(ClinicalCase clinicalCase) {
        this.clinicalCase = clinicalCase;
    }

    @ManyToOne(targetEntity = Physician.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "requesting_physician", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public Physician getRequestingPhysician() {
        return requestingPhysician;
    }

    public void setRequestingPhysician(Physician requestingPhysician) {
        this.requestingPhysician = requestingPhysician;
    }
```

```
    @ManyToOne(targetEntity = Physician.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "consultant", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public Physician getConsultant() {
        return consultant;
    }

    public void setConsultant(Physician consultant) {
        this.consultant = consultant;
    }

    @Column(name = "consultation_date")
    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentLocalDate")
    public LocalDate getConsultationDate() {
        return consultationDate;
    }

    public void setConsultationDate(LocalDate consultationDate) {
        this.consultationDate = consultationDate;
    }

    @Column(name = "consultation_start_date")
    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    public DateTime getConsultationStart() {
        return consultationStart;
    }

    public void setConsultationStart(DateTime consultationStart) {
        this.consultationStart = consultationStart;
    }

    @Column(name = "consultation_end_date")
    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    public DateTime getConsultationEnd() {
        return consultationEnd;
    }

    public void setConsultationEnd(DateTime consultationEnd) {
        this.consultationEnd = consultationEnd;
    }

    @Column(name = "notification_sms_schedule")
    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    public DateTime getNotificationSmsSchedule() {
        return notificationSmsSchedule;
    }

    public void setNotificationSmsSchedule(DateTime notificationSmsSchedule) {
        this.notificationSmsSchedule = notificationSmsSchedule;
    }

    @Basic
    @Column(name = "notification_sms_schedule_sent")
    public Boolean getNotificationSmsScheduleSent() {
        if (notificationSmsScheduleSent == null) {
            return Boolean.FALSE;
        }
        return notificationSmsScheduleSent;
    }

    public void setNotificationSmsScheduleSent(Boolean notificationSmsScheduleSent) {
        this.notificationSmsScheduleSent = notificationSmsScheduleSent;
    }

    @Override
    public JSONObject toJSONObject() {
        JSONObject json = new JSONObject();
        return json;
    }

    @Override
    public String toString() {
        return "Appointment [clinicalCase=" + clinicalCase + ", requestingPhysician="
+ requestingPhysician + ", consultant=" + consultant
```

```
                + ", consultationDate=" + consultationDate + ", consultationStart=" +
consultationStart + ", consultationEnd=" + consultationEnd
                + ", notificationSmsSchedule=" + notificationSmsSchedule + ",
notificationSmsScheduleSent=" + notificationSmsScheduleSent + "]";
    }

    @Basic
    @Column(name = "is_seen_requesting_phyisician")
    public Boolean getIsSeenRequestingPhysician() {
        if (isSeenRequestingPhysician == null)
            return Boolean.FALSE;
        return isSeenRequestingPhysician;
    }

    public void setIsSeenRequestingPhysician(Boolean isSeenRequestingPhysician) {
        this.isSeenRequestingPhysician = isSeenRequestingPhysician;
    }

    @Basic
    @Column(name = "is_seen_consultant")
    public Boolean getIsSeenConsultant() {
        if (isSeenConsultant == null)
            return Boolean.FALSE;
        return isSeenConsultant;
    }

    public void setIsSeenConsultant(Boolean isSeenConsultant) {
        this.isSeenConsultant = isSeenConsultant;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/ChatMessage.java**

```
package org.leaf.cms.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name = "chatMessage")
@Table(name = ChatMessage.TABLE_NAME)
public class ChatMessage extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    public static final String TABLE_NAME = "chat_message";

    private ClinicalCase clinicalCase;
    private String message;
    private Physician physician;

    @Override
    public JSONObject toJSONObject() {
        JSONObject json = new JSONObject();
        return json;
    }

    @ManyToOne(targetEntity = ClinicalCase.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "clinical_case", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public ClinicalCase getClinicalCase() {
        return clinicalCase;
    }

    public void setClinicalCase(ClinicalCase clinicalCase) {
```

```
        this.clinicalCase = clinicalCase;
    }

    @Column(name = "message", columnDefinition = "text")
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    @ManyToOne(targetEntity = Physician.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "physician", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public Physician getPhysician() {
        return physician;
    }

    public void setPhysician(Physician physician) {
        this.physician = physician;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/ClinicalCase.java**

```
package org.leaf.cms.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name = "clinicalCase")
@Table(name = ClinicalCase.TABLE_NAME)
public class ClinicalCase extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    public static final String TABLE_NAME = "clinical_case";

    private User createdBy;
    private Specialty specialty;
    private String question;
    private String patientLabExam;
    private String patientDiagnosis;
    private String remarks;
    private String status;

    @Override
    public JSONObject toJSONObject() {
        JSONObject json = new JSONObject();
        return json;
    }

    @ManyToOne(targetEntity = User.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "created_by_id")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public User getCreatedBy() {
        return createdBy;
    }

    public void setCreatedBy(User createdBy) {
        this.createdBy = createdBy;
    }
```

```
    @ManyToOne(targetEntity = Specialty.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "specialty")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public Specialty getSpecialty() {
        return specialty;
    }

    public void setSpecialty(Specialty specialty) {
        this.specialty = specialty;
    }

    @Column(name = "question", columnDefinition = "text")
    public String getQuestion() {
        return question;
    }

    public void setQuestion(String question) {
        this.question = question;
    }

    @Column(name = "patient_lab_exam", columnDefinition = "text")
    public String getPatientLabExam() {
        return patientLabExam;
    }

    public void setPatientLabExam(String patientLabExam) {
        this.patientLabExam = patientLabExam;
    }

    @Column(name = "patient_diagnosis", columnDefinition = "text")
    public String getPatientDiagnosis() {
        return patientDiagnosis;
    }

    public void setPatientDiagnosis(String patientDiagnosis) {
        this.patientDiagnosis = patientDiagnosis;
    }

    @Column(name = "remarks", columnDefinition = "text")
    public String getRemarks() {
        return remarks;
    }

    public void setRemarks(String remarks) {
        this.remarks = remarks;
    }

    @Column(name = "status")
    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    @Override
    public String toString() {
        return "ClinicalCase [id=" + id + ", createdBy=" + createdBy + ", specialty="
+ specialty + ", question=" + question + ", patientLabExam="
                + patientLabExam + ", patientDiagnosis=" + patientDiagnosis + ",
remarks=" + remarks + ", status=" + status + "]";
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/ClinicInfo.java**

```
package org.leaf.cms.model;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
```

```
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="clinicInfo")
@Table(name=ClinicInfo.TABLE_NAME)
public class ClinicInfo extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible{

    public final static String TABLE_NAME = "clinic_info";

    private Physician physician;

    private String name;
    private String address;
    private String telNo;

    @Override
    public JSONObject toJSONObject() {
            return null;
    }

    @Basic
    @Column(name="name")
    public String getName() {
            return name;
    }

    public void setName(String name) {
            this.name = name;
    }

    @Basic
    @Column(name="address")
    public String getAddress() {
            return address;
    }

    public void setAddress(String address) {
            this.address = address;
    }

    @Basic
    @Column(name="tel_no")
    public String getTelNo() {
            return telNo;
    }

    public void setTelNo(String telNo) {
            this.telNo = telNo;
    }

    @ManyToOne(fetch=FetchType.LAZY, targetEntity=Physician.class)
    @JoinColumn(name = "physician_id")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public Physician getPhysician() {
            return physician;
    }

    public void setPhysician(Physician physician) {
            this.physician = physician;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/Configuration.java**

```
package org.leaf.cms.model;
```

```
import java.beans.Transient;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.hibernate.annotations.Type;
import org.joda.time.DateTime;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;
import org.leaf.cms.constants.ApplicationConstants;

@Entity(name = "configuration")
@Table(name = Configuration.TABLE_NAME)
public class Configuration extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    public static final String DEFAULT_CONNECTION_STRING =
ApplicationConstants.DEFAULT_CONNECTION_STRING;
    public static final String DEFAULT_USERNAME =
ApplicationConstants.DEFAULT_USERNAME;
    public static final String DEFAULT_PASSWORD =
ApplicationConstants.DEFAULT_PASSWORD;
    public static final String DEFAULT_REST_CENTRAL_IP =
ApplicationConstants.DEFAULT_REST_CENTRAL_IP;

    public static final String TABLE_NAME = "configuration";
    private String connectionString;
    private String username;
    private String password;
    private Boolean isActive;
    private String centralIp;
    private DateTime scanTime;

    private String smsTemplate;
    private Boolean isSMSEnabled;

    public Configuration() {
        this.connectionString = DEFAULT_CONNECTION_STRING;
        this.username = DEFAULT_USERNAME;
        this.password = DEFAULT_PASSWORD;
        this.centralIp = DEFAULT_REST_CENTRAL_IP;
        this.isActive = Boolean.TRUE;
        this.scanTime = new DateTime(0L);
        this.isSMSEnabled = true;
    }

    @Basic
    @Column(name = "connection_string")
    public String getConnectionString() {
        return connectionString;
    }

    public void setConnectionString(String connectionString) {
        this.connectionString = connectionString;
    }

    @Basic
    @Column(name = "username")
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Basic
    @Column(name = "password")
    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    @Transient
    @SuppressWarnings("unchecked")
    public JSONObject toJSONObject() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("id", this.getId());
        jsonObject.put("connectionString", this.getConnectionString());
        jsonObject.put("username", this.getUsername());
        jsonObject.put("password", this.getPassword());
        return jsonObject;
    }

    @Basic
    @Column(name = "is_active")
    public Boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(Boolean isActive) {
        this.isActive = isActive;
    }

    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    @Column(name = "scan_time", nullable = false)
    public DateTime getScanTime() {
        return scanTime;
    }

    public void setScanTime(DateTime scanTime) {
        this.scanTime = scanTime;
    }

    @Basic
    @Column(name = "central_ip")
    public String getCentralIp() {
        return centralIp;
    }

    public void setCentralIp(String centralIp) {
        this.centralIp = centralIp;
    }

    @Override
    public String toString() {
        return "Configuration [id=" + id + ", connectionString=" + connectionString +
", username=" + username + ", password=" + password
                + ", isActive=" + isActive + ", centralIp=" + centralIp + ",
scanTime=" + scanTime + "]";
    }

    @Column(name = "sms_template")
    public String getSmsTemplate() {
        return smsTemplate;
    }

    public void setSmsTemplate(String smsTemplate) {
        this.smsTemplate = smsTemplate;
    }

    @Basic
    @Column(name = "is_sms_enabled")
    public Boolean getIsSMSEnabled() {
        return isSMSEnabled;
    }

    public void setIsSMSEnabled(Boolean isSMSEnabled) {
        this.isSMSEnabled = isSMSEnabled;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/DiscussionThread.java**

```
package org.leaf.cms.model;

import java.util.List;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.OrderBy;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="discussionThread")
@Table(name=DiscussionThread.TABLE_NAME)
public class DiscussionThread extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible{
    public static final String TABLE_NAME = "discussion_thread";

    private String title;
    private User createdBy;
    private Boolean isSeen;

    private List<DiscussionThreadReply> discussionThreadReplies;
    private List<Subscription> subscriptions;

    @Basic
    @Column(name="title")
    public String getTitle() {
            return title;
    }

    public void setTitle(String title) {
            this.title = title;
    }

    @Override
    @SuppressWarnings("unchecked")
    public JSONObject toJSONObject() {
            JSONObject json = new JSONObject();
            json.put("title", getTitle());
            json.put("createdBy", getCreatedBy().getFullName());
            DiscussionThreadReply firstReply ;
            List<DiscussionThreadReply> replies = getDiscussionThreadReplies();
            if(replies != null && replies.size() > 0){
                    firstReply = getDiscussionThreadReplies().get(0);
                    json.put("info", firstReply.getMessage());
            }
            return json;
    }

    @ManyToOne(targetEntity = User.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "created_by")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public User getCreatedBy() {
            return createdBy;
    }

    public void setCreatedBy(User createdBy) {
            this.createdBy = createdBy;
    }

    @OneToMany(mappedBy="thread", fetch=FetchType.LAZY)
    @Where(clause=" valid = 1 ")
    @OrderBy(value=" createdOn DESC ")
```

```
    @NotFound(action=NotFoundAction.IGNORE)
    public List<DiscussionThreadReply> getDiscussionThreadReplies() {
            return discussionThreadReplies;
    }

    public void setDiscussionThreadReplies(List<DiscussionThreadReply>
discussionThreadReplies) {
            this.discussionThreadReplies = discussionThreadReplies;
    }

    @Basic
    @Column(name="is_seen")
    public Boolean getIsSeen() {
            if(isSeen == null)
                    return Boolean.FALSE;
            return isSeen;
    }

    public void setIsSeen(Boolean isSeen) {
            this.isSeen = isSeen;
    }

    @OneToMany(mappedBy="thread", fetch=FetchType.LAZY)
    @Where(clause=" valid = 1 ")
    @OrderBy(value=" createdOn DESC ")
    @NotFound(action=NotFoundAction.IGNORE)
    public List<Subscription> getSubscriptions() {
            return subscriptions;
    }

    public void setSubscriptions(List<Subscription> subscriptions) {
            this.subscriptions = subscriptions;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/DiscussionThreadReply.java**

```
package org.leaf.cms.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="discussionThreadReply")
@Table(name=DiscussionThreadReply.TABLE_NAME)
public class DiscussionThreadReply extends BaseSequentialUpdatableEntity<Long>
implements JsonConvertible{
    public static final String TABLE_NAME = "discussion_thread_reply";
    private DiscussionThread thread;
    private String message;
    private User postedBy;

    @ManyToOne(targetEntity = DiscussionThread.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "thread_id")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public DiscussionThread getThread() {
            return thread;
    }
    public void setThread(DiscussionThread thread) {
            this.thread = thread;
    }

    @ManyToOne(targetEntity = User.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "posted_by")
```

```
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public User getPostedBy() {
            return postedBy;
    }
    public void setPostedBy(User postedBy) {
            this.postedBy = postedBy;
    }

    @Column(name="message", columnDefinition="text")
    public String getMessage() {
            return message;
    }
    public void setMessage(String message) {
            this.message = message;
    }
    @Override
    public JSONObject toJSONObject() {
            return null;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.model/FileManagement.java**

```
package org.leaf.cms.model;

import java.io.File;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Type;
import org.hibernate.annotations.Where;
import org.joda.time.DateTime;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name = "file")
@Table(name = FileManagement.TABLE_NAME)
public class FileManagement extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    public static final String TABLE_NAME = "file";

    private ClinicalCase clinicalCase;
    private String name;
    private String contentType;
    private String location;
    private User uploadedBy;
    private DateTime dateUploaded;
    private File savedFile;

    @ManyToOne(targetEntity = ClinicalCase.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "clinical_case", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public ClinicalCase getClinicalCase() {
        return clinicalCase;
    }

    public void setClinicalCase(ClinicalCase clinicalCase) {
        this.clinicalCase = clinicalCase;
    }

    @Column(name = "name", columnDefinition = "text", nullable = false)
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    @Column(name = "content_type", columnDefinition = "text", nullable = false)
    public String getContentType() {
        return contentType;
    }

    public void setContentType(String contentType) {
        this.contentType = contentType;
    }

    @Column(name = "location", columnDefinition = "text", nullable = false)
    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    @ManyToOne(targetEntity = User.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "uploaded_by", nullable = false)
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public User getUploadedBy() {
        return uploadedBy;
    }

    public void setUploadedBy(User uploadedBy) {
        this.uploadedBy = uploadedBy;
    }

    @Column(name = "date_uploaded", nullable = false)
    @Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
    public DateTime getDateUploaded() {
        return dateUploaded;
    }

    public void setDateUploaded(DateTime dateUploaded) {
        this.dateUploaded = dateUploaded;
    }

    @Transient
    public File getSavedFile() {
        return savedFile;
    }

    public void setSavedFile(File savedFile) {
        this.savedFile = savedFile;
    }

    @Override
    public JSONObject toJSONObject() {
        JSONObject json = new JSONObject();
        return json;
    }

    @Override
    public String toString() {
        return "File [clinicalCase=" + clinicalCase + ", name=" + name + ",
contentType=" + contentType + ", location=" + location + ", uploadedBy="
                + uploadedBy + ", dateUploaded=" + dateUploaded + "]";
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/Reference.java**

```
package org.leaf.cms.model;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
```

```
import javax.persistence.Table;

import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="reference")
@Table(name=Reference.TABLE_NAME)
public class Reference extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible{

    public static final String TABLE_NAME = "reference";

    private String title;
    private String description;
    private String url;

    @Override
    @SuppressWarnings("unchecked")
    public JSONObject toJSONObject() {
            JSONObject json = new JSONObject();

            json.put("title", getTitle());
            json.put("description", getDescription());
            json.put("url", getUrl());
            return json;
    }


    @Basic
    @Column(name="description", columnDefinition="text")
    public String getDescription() {
            return description;
    }

    public void setDescription(String description) {
            this.description = description;
    }
    @Basic
    @Column(name="url")
    public String getUrl() {
            return url;
    }

    public void setUrl(String url) {
            this.url = url;
    }

    @Basic
    @Column(name="title", columnDefinition="text")
    public String getTitle() {
            return title;
    }

    public void setTitle(String title) {
            this.title = title;

    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/SMSEntry.java**

```
package org.leaf.cms.model;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="smsEntry")
@Table(name=SMSEntry.TABLE_NAME)
```

```
public class SMSEntry extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible{

    public static final String TABLE_NAME = "sms_entry";

    private String message;
    private String recipient;
    private Boolean isSent;

    @Override
    public JSONObject toJSONObject() {
            return null;
    }

    @Basic
    @Column(columnDefinition="text", name="message")
    public String getMessage() {
            return message;
    }

    public void setMessage(String message) {
            this.message = message;
    }

    @Basic
    @Column(name="recipient")
    public String getRecipient() {
            return recipient;
    }

    public void setRecipient(String recipient) {
            this.recipient = recipient;
    }

    @Basic
    @Column(name="is_sent")
    public Boolean getIsSent() {
            return isSent;
    }

    public void setIsSent(Boolean isSent) {
            this.isSent = isSent;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/Specialty.java**

```
package org.leaf.cms.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name = "specialty")
@Table(name = Specialty.TABLE_NAME)
public class Specialty extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    public static final String TABLE_NAME = "specialty";

    private String name;
    private String description;
    private Set<Physician> physicians;

    @Override
    public JSONObject toJSONObject() {
```

```
        return null;
    }

    @Basic
    @Column(name = "name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Basic
    @Column(name = "description", columnDefinition = "text")
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    @ManyToMany(mappedBy = "specialties")
    public Set<Physician> getPhysicians() {
        return this.physicians == null ? new HashSet<Physician>() : physicians;
    }

    public void setPhysicians(Set<Physician> physicians) {
        this.physicians = physicians;
    }
}
```

- **web-pcs/src/org.leaf.cms.action.model/Subscription.java**

```
package org.leaf.cms.model;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.NotFound;
import org.hibernate.annotations.NotFoundAction;
import org.hibernate.annotations.Where;
import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;

@Entity(name="subscription")
@Table(name=Subscription.TABLE_NAME)
public class Subscription extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible{
    public static final String TABLE_NAME = "subscription";

    private DiscussionThread thread;
    private User follower;


    @Override
    public JSONObject toJSONObject() {
            return null;
    }

    @ManyToOne(targetEntity = DiscussionThread.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "thread_id")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public DiscussionThread getThread() {
            return thread;
    }

    public void setThread(DiscussionThread thread) {
            this.thread = thread;
    }
```

```
    @ManyToOne(targetEntity = User.class, fetch = FetchType.LAZY)
    @JoinColumn(name = "follower")
    @Where(clause = " valid = 1 ")
    @NotFound(action = NotFoundAction.IGNORE)
    public User getFollower() {
            return follower;
    }


    public void setFollower(User follower) {
            this.follower = follower;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.model/User.java**

```
package org.leaf.cms.model;

import javax.persistence.Access;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.json.simple.JSONObject;
import org.leaf.cms.commons.interfaces.JsonConvertible;
import org.leaf.cms.commons.model.BaseSequentialUpdatableEntity;
import org.leaf.cms.enums.SubscriptionDigestTypeEnum;

@Entity(name = "User")
@Table(name = User.TABLE_NAME)
@Access(value = javax.persistence.AccessType.PROPERTY)
@Inheritance(strategy = InheritanceType.JOINED)
public class User extends BaseSequentialUpdatableEntity<Long> implements
JsonConvertible {

    private static final long serialVersionUID = 4695129428919955634L;

    public static final String TABLE_NAME = "user";

    private String username;
    private String password;
    private String firstName;
    private String lastName;

    private String imageUrl;

    private Boolean sendWhenBooked;
    private Boolean sendWhenCancelled;
    private Boolean sendWhenRescheduled;
    private Boolean sendWhenMessageSent;

    private Boolean sendOnForumReply;
    private Boolean sendOnForumSubscribed;

    private Boolean sendSubscriptionUpdates;

    private Boolean triggerSmsReminder;

    private String cellphoneNumber;

    private SubscriptionDigestTypeEnum subscriptionDigestType;

    @Basic
    @Column(name = "username", nullable = false, updatable = false, unique = true)
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
```

```
        this.username = username;
    }

    @Basic
    @Column(name = "password", nullable = false)
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Basic
    @Column(name = "firstname")
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    @Basic
    @Column(name = "lastname")
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    @SuppressWarnings("unchecked")
    public JSONObject toJSONObject() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("id", this.getId());
        jsonObject.put("firstName", this.getFirstName());
        jsonObject.put("lastName", this.getLastName());
        jsonObject.put("username", this.getUsername());
        return jsonObject;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", username=" + username + ", password=" + password
+ ", firstName=" + firstName + ", lastName=" + lastName + "]";
    }

    @Transient
    public String getFullName() {
        return firstName + " " + lastName;
    }

    @Basic
    @Column(name = "send_when_booked")
    public Boolean getSendWhenBooked() {
        if (sendWhenBooked == null)
            return Boolean.FALSE;
        return sendWhenBooked;
    }

    public void setSendWhenBooked(Boolean sendWhenBooked) {
        this.sendWhenBooked = sendWhenBooked;
    }

    @Basic
    @Column(name = "send_when_cancelled")
    public Boolean getSendWhenCancelled() {
        if (sendWhenCancelled == null)
            return Boolean.FALSE;
        return sendWhenCancelled;
    }

    public void setSendWhenCancelled(Boolean sendWhenCancelled) {
```

```
        this.sendWhenCancelled = sendWhenCancelled;
    }

    @Basic
    @Column(name = "send_when_rescheduled")
    public Boolean getSendWhenRescheduled() {
        if (sendWhenRescheduled == null)
            return Boolean.FALSE;
        return sendWhenRescheduled;
    }

    public void setSendWhenRescheduled(Boolean sendWhenRescheduled) {
        this.sendWhenRescheduled = sendWhenRescheduled;
    }

    @Basic
    @Column(name = "send_when_message_sent")
    public Boolean getSendWhenMessageSent() {
        if (sendWhenMessageSent == null)
            return Boolean.FALSE;
        return sendWhenMessageSent;
    }

    public void setSendWhenMessageSent(Boolean sendWhenMessageSent) {
        this.sendWhenMessageSent = sendWhenMessageSent;
    }

    @Basic
    @Column(name = "send_on_forum_reply")
    public Boolean getSendOnForumReply() {
        if (sendOnForumReply == null)
            return Boolean.FALSE;
        return sendOnForumReply;
    }

    public void setSendOnForumReply(Boolean sendOnForumReply) {
        this.sendOnForumReply = sendOnForumReply;
    }

    @Basic
    @Column(name = "send_on_forum_subscribed")
    public Boolean getSendOnForumSubscribed() {
        if (sendOnForumSubscribed == null)
            return Boolean.FALSE;
        return sendOnForumSubscribed;
    }

    public void setSendOnForumSubscribed(Boolean sendOnForumSubscribed) {
        this.sendOnForumSubscribed = sendOnForumSubscribed;
    }

    @Basic
    @Column(name = "send_subscription_updates")
    public Boolean getSendSubscriptionUpdates() {
        if (sendSubscriptionUpdates == null)
            return Boolean.FALSE;
        return sendSubscriptionUpdates;
    }

    public void setSendSubscriptionUpdates(Boolean sendSubscriptionUpdates) {
        this.sendSubscriptionUpdates = sendSubscriptionUpdates;
    }

    @Enumerated(EnumType.STRING)
    @Column(name = "subscription_digest_type")
    public SubscriptionDigestTypeEnum getSubscriptionDigestType() {
        return subscriptionDigestType;
    }

    public void setSubscriptionDigestType(SubscriptionDigestTypeEnum
subscriptionDigestType) {
        this.subscriptionDigestType = subscriptionDigestType;
    }

    @Transient
    public String getUserType() {
```

```
        if (this instanceof Physician) {
            return Physician.class.getSimpleName();
        }
        return User.class.getSimpleName();
    }

    @Basic
    @Column(name = "image_url")
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    @Basic
    @Column(name = "cellphone_number")
    public String getCellphoneNumber() {
        return cellphoneNumber;
    }

    public void setCellphoneNumber(String cellphoneNumber) {
        this.cellphoneNumber = cellphoneNumber;
    }

    @Basic
    @Column(name = "trigger_sms_reminder")
    public Boolean getTriggerSmsReminder() {
        if (triggerSmsReminder == null)
            return Boolean.FALSE;
        return triggerSmsReminder;
    }

    public void setTriggerSmsReminder(Boolean triggerSmsReminder) {
        this.triggerSmsReminder = triggerSmsReminder;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.repository/AppointmentRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.joda.time.DateTime;
import org.joda.time.LocalDate;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;

public interface AppointmentRepository extends BaseJpaRepository<Appointment, Long> {

    public List<Appointment>
findByConsultationStartBeforeAndConsultationEndAfterAndIsValid(DateTime before,
DateTime after, boolean isValid);

    public List<Appointment> findByConsultationEndBeforeAndIsValid(DateTime before,
boolean isValid);

    public Page<Appointment> findByRequestingPhysician(User requestingPhysician,
Pageable page);

    @Query("SELECT a FROM appointment a WHERE a.requestingPhysician = ?1 OR
a.requestingPhysician = ?2 OR a.consultant = ?1 OR a.consultant = ?2")
    public List<Appointment>
findAppointmentOfRequestingPhysicianAndConsultant(Physician requestingPhysician,
Physician consultant);
```

```
    @Query("SELECT a FROM appointment a WHERE a.requestingPhysician = ?1 OR
a.requestingPhysician = ?2 OR a.consultant = ?1 OR a.consultant = ?2 AND
a.consultationDate BETWEEN ?3 AND ?4")
    public List<Appointment> findAppointmentOfRequestingPhysicianAndConsultant(

Physician requestingPhysician,

Physician consultant,

LocalDate start,

LocalDate end);

    @Query("SELECT a.clinicalCase FROM appointment a INNER JOIN a.clinicalCase cc
WHERE a.consultant = ?1 AND (cc.status = 'Active' OR cc.status = 'Ongoing') AND
a.isValid = true")
    public List<ClinicalCase>
findActiveAndOngoingClinicalCasesFromAppointmentOfAConsultant(Physician consultant);

    public Appointment findByClinicalCaseAndIsValid(ClinicalCase clinicalCase,
boolean isValid);

    @Query("SELECT p.lastName, p.firstName, (SELECT count(*) FROM appointment a WHERE
a.requestingPhysician = p AND a.consultationDate >= ?1 AND a.consultationDate <= ?2)
FROM Physician p ORDER BY p.lastName ASC")
    public List<Object[]> countAppointmentAsRequestingPhysician(LocalDate startDate,
LocalDate endDate);

    @Query("SELECT p.lastName, p.firstName, (SELECT count(*) FROM appointment a WHERE
a.consultant = p AND a.consultationDate >= ?1 AND a.consultationDate <= ?2) FROM
Physician p ORDER BY p.lastName ASC")
    public List<Object[]> countAppointmentAsConsultant(LocalDate startDate, LocalDate
endDate);

    public int countByConsultationDate(LocalDate consultationDate);

    @Query("SELECT count(a) FROM appointment a WHERE a.consultationDate = ?1 AND
(a.requestingPhysician = ?2 or a.consultant = ?2)")
    public int countByConsultationDate(LocalDate consultationDate, User physician);

    public List<Appointment>
findByNotificationSmsScheduleAndNotificationSmsScheduleSentNot(

DateTime notificationSmsSchedule,

boolean notificationSmsScheduleSent);

    @Query("SELECT a FROM appointment a WHERE a.requestingPhysician = ?1 AND
a.consultationDate >= ?2 AND a.consultationDate <= ?3 ORDER BY a.consultationDate
ASC")
    public List<Appointment> findAppointmentOfRequestingPhysician(Physician
requestingPhysician, LocalDate startDate, LocalDate endDate);

    @Query("SELECT a FROM appointment a WHERE a.consultant = ?1 AND
a.consultationDate >= ?2 AND a.consultationDate <= ?3 ORDER BY a.consultationDate
ASC")
    public List<Appointment> findAppointmentOfConsultant(Physician consultant,
LocalDate startDate, LocalDate endDate);

    @Query("SELECT a FROM appointment a WHERE (a.requestingPhysician = ?1 OR
a.consultant = ?1) AND a.isValid = true ORDER BY a.createdOn DESC")
    public Page<Appointment> findAppointmentOfPhysician(User physician, Pageable
page);

    public Long countByRequestingPhysicianAndIsSeenRequestingPhysicianFalse(User
physician);

    public Long countByConsultantAndIsSeenConsultantFalse(User physician);
}
```

- **web-pcs/src/org.leaf.cms.action.repository/BaseJpaRepository.java**

```
package org.leaf.cms.model.repository;

import java.io.Serializable;
```

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.NoRepositoryBean;

@NoRepositoryBean
public interface BaseJpaRepository<T,ID extends Serializable> extends
JpaRepository<T, ID>{
    public Page<T> findByIsValidTrue(Pageable pageable);
}
```

- **web-pcs/src/org.leaf.cms.action.repository/ChatMessagesRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.ChatMessage;
import org.leaf.cms.model.ClinicalCase;

public interface ChatMessagesRepository extends BaseJpaRepository<ChatMessage, Long>
{

    public List<ChatMessage> findByClinicalCaseOrderByCreatedOnAsc(ClinicalCase
clinicalCase);

}
```

- **web-pcs/src/org.leaf.cms.action.repository/ClinicalCaseRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Specialty;
import org.leaf.cms.model.User;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;

public interface ClinicalCaseRepository extends BaseJpaRepository<ClinicalCase, Long>
{

    @Modifying
    @Query("UPDATE clinicalCase c SET c.status = ?2 WHERE c.id = ?1")
    public int updateStatus(Long id, String status);

    public List<ClinicalCase> findByCreatedByAndStatusAndSpecialty(User createdBy,
String status, Specialty specialty);

    public List<ClinicalCase> findByCreatedBy(User createdBy);

}
```

- **web-pcs/src/org.leaf.cms.action.repository/ClinicInfoRepository.java**

```
package org.leaf.cms.model.repository;

import org.leaf.cms.model.ClinicInfo;

public interface ClinicInfoRepository extends BaseJpaRepository<ClinicInfo, Long>
{

}
```

- **web-pcs/src/org.leaf.cms.action.repository/ConfigurationRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.Configuration;

public interface ConfigurationRepository extends BaseJpaRepository<Configuration,
Long> {
```

```
    public List<Configuration> findByIsValidTrue();
}
```

- **web-pcs/src/org.leaf.cms.action.repository/DiscussionThreadReplyRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.DiscussionThreadReply;

public interface DiscussionThreadReplyRepository extends
BaseJpaRepository<DiscussionThreadReply, Long>{
    public List<DiscussionThread> findByIsValidTrue();
}
```

- **web-pcs/src/org.leaf.cms.action.repository/DiscussionThreadRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.User;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;

public interface DiscussionThreadRepository extends
BaseJpaRepository<DiscussionThread, Long> {

    @Query("SELECT d, MAX(r.createdOn) as latestReply FROM discussionThread d LEFT
JOIN d.discussionThreadReplies r WHERE d.isValid = true GROUP BY d.id ORDER BY
latestReply desc")
    public List<Object[]> findValidDiscussionThreads();

    @Query("SELECT d, MAX(r.createdOn) as latestReply FROM discussionThread d LEFT
JOIN d.discussionThreadReplies r WHERE d.isValid = true GROUP BY d.id ORDER BY
latestReply desc")
    public Page<Object[]> findValidDiscussionThreads(Pageable pageable);

    @Override
    public Page<DiscussionThread> findByIsValidTrue(Pageable pageable);

    @Query("SELECT d, MAX(r.createdOn) as latestReply FROM discussionThread d LEFT
JOIN d.discussionThreadReplies r WHERE d.isValid = true AND d.createdBy = ?1 GROUP BY
d.id ORDER BY latestReply desc")
    public List<Object[]> findValidDiscussionThreads(User createdBy);

    @Query("SELECT d, MAX(r.createdOn) as latestReply FROM discussionThread d LEFT
JOIN d.discussionThreadReplies r WHERE d.isValid = true AND d.createdBy = ?1 GROUP BY
d.id ORDER BY latestReply desc")
    public Page<Object[]> findValidDiscussionThreads(User createdBy,Pageable
pageable);

    public Long countByIsSeenFalseAndIsValidTrue(); }
```

- **web-pcs/src/org.leaf.cms.action.repository/FileManagementRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.FileManagement;

public interface FileManagementRepository extends BaseJpaRepository<FileManagement,
Long> {

    public List<FileManagement> findByClinicalCase(ClinicalCase clinicalCase);

}
```

- **web-pcs/src/org.leaf.cms.action.repository/PhysicianRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;


public interface PhysicianRepository extends BaseJpaRepository<Physician, Long> {

    public Long countByEmail(String email);

    public Long countByMedLicenseId(Long medLicenseId);

    public Physician findByEmailAndMedLicenseId(String email, Long medLicenseId);

    public Physician findByVerificationCode(String verificationCode);

    public Physician findByIdAndVerificationCode(Long id, String verificationCode);

    public Physician findByEmail(String email);

    @Query("SELECT p FROM Physician p INNER JOIN p.specialties s WHERE s = ?1 AND
p.isActivited = true")
    public List<Physician> physiciansOfSpecificSpecialty(Specialty specialty);

    public Page<Physician>
findByIsActivitedFalseAndIsValidTrueAndActivationDateIsNull(Pageable pageable);

    public Page<Physician> findByIsActivitedFalse(Pageable pageable);

    public Page<Physician> findByIsActivitedTrue(Pageable pageable);

    @Query("SELECT p FROM Physician p INNER JOIN p.specialties s WHERE s = ?1 AND
p.isActivited = true")
    public Page<Physician> physiciansOfSpecificSpecialty(Specialty specialty,
Pageable pageable);
}
```

- **web-pcs/src/org.leaf.cms.action.repository/ReferenceRepository.java**

```
package org.leaf.cms.model.repository;

import org.leaf.cms.model.Reference;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

public interface ReferenceRepository extends BaseJpaRepository<Reference, Long>{
    public Page<Reference> findByIsValidTrue(Pageable pageable);
}
```

- **web-pcs/src/org.leaf.cms.action.repository/SMSEntryRepository.java**

```
package org.leaf.cms.model.repository;

import org.leaf.cms.model.SMSEntry;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

public interface SMSEntryRepository extends BaseJpaRepository<SMSEntry, Long>{
    public Page<SMSEntry> findByIsSentFalse(Pageable page);
}
```

- **web-pcs/src/org.leaf.cms.action.repository/SpecialtyRepository.java**

```
package org.leaf.cms.model.repository;

import org.leaf.cms.model.Specialty;
```

```
public interface SpecialtyRepository extends BaseJpaRepository<Specialty, Long> {

    public Specialty findByName(String name);

}
```

- **web-pcs/src/org.leaf.cms.action.repository/SubscriptionRepository.java**

```
package org.leaf.cms.model.repository;

import java.util.List;

import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.Subscription;
import org.leaf.cms.model.User;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;

public interface SubscriptionRepository extends BaseJpaRepository<Subscription, Long>
{

    @Query("SELECT s, MAX(r.createdOn) as latestReply FROM subscription s LEFT JOIN
s.thread d LEFT JOIN d.discussionThreadReplies r WHERE s.isValid = true and
s.follower = ?1 GROUP BY s.id ORDER BY latestReply desc")
    public List<Object[]> subscriptionsOfFollower(User follower);

    @Query("SELECT s, MAX(r.createdOn) as latestReply FROM subscription s LEFT JOIN
s.thread d LEFT JOIN d.discussionThreadReplies r WHERE s.isValid = true and
s.follower = ?1 GROUP BY s.id ORDER BY latestReply desc")
    public Page<Object[]> subscriptionsOfFollower(User follower, Pageable pageable);

    public Subscription findByThreadAndFollower(DiscussionThread thread, User
follower);

    public List<Subscription> findByThread(DiscussionThread thread);

}
```

- **web-pcs/src/org.leaf.cms.action.repository/UserRepository.java**

```
package org.leaf.cms.model.repository;

import org.leaf.cms.model.User;

public interface UserRepository extends BaseJpaRepository<User, Long> {

    public User findByUsername(String username);

    public User findByUsernameAndPassword(String username, String password);

}
```

- **web-pcs/src/org.leaf.cms.action.service/AppointmentService.java**

```
package org.leaf.cms.service;

import java.util.List;
import java.util.Map;

import org.joda.time.DateTime;
import org.joda.time.LocalDate;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.service.dto.ConsultationTrends;
import org.springframework.data.domain.Page;

public interface AppointmentService {

    public List<Appointment> getAppointments(Physician requestingPhysician, Physician
consultant);
```

```
    public List<Appointment> getAppointments(Physician requestingPhysician, Physician
consultant, LocalDate start, LocalDate end);

    public List<Appointment> getAppointments(DateTime before, DateTime after);

    public Appointment createAppointment(Appointment appointment, boolean
isNewClinicalCase);

    public Appointment rescheduleAppointment(Appointment appointment);

    public void cancelAppointment(Long id);

    public void cancelAppointment(Appointment appointment);

    public Appointment terminateAppointment(Appointment appointment);

    public Appointment closeAppointment(Appointment appointment);

    public Page<Appointment> findAppointment(User physician, int pageNumber, int
itemsPerPage);

    public Appointment findAppointment(Long id);

    public Appointment findAppointment(ClinicalCase clinicalCase);

    public List<Object[]> getAppointmentReportOfRequestingPhysicians(LocalDate
startDate, LocalDate endDate);

    public List<Object[]> getAppointmentReportOfConsultants(LocalDate startDate,
LocalDate endDate);

    public Map<String, ConsultationTrends> getConsultationTrends();

    public Map<String, ConsultationTrends> getConsultationTrends(User physician);

    public void setAppointmentNotificationSchedule(Long appointmentId, int
minutesBeforeAppointment);

    public List<Appointment> getUpcomingAppointmentsToBeNotified(DateTime now);

    public void saveAppointments(List<Appointment> appointments);

    public List<Appointment> getAppointmentsAsRequestingPhysician(Physician
physician, LocalDate startDate, LocalDate endDate);

    public List<Appointment> getAppointmentsAsConsultant(Physician physician,
LocalDate startDate, LocalDate endDate);

    public List<Appointment> getActiveAppointmentsBefore(DateTime before);

    public Long countUnseenAppointments(User physician);

    public void markAppointmentAsSeenByRequestingPhyisician(Appointment appointment);

    public void markAppointmentAsSeenByConsultant(Appointment appointment);

}
```

- **web-pcs/src/org.leaf.cms.action.service/ChatService.java**

```
package org.leaf.cms.service;

import java.util.List;

import org.leaf.cms.model.ChatMessage;
import org.leaf.cms.model.Physician;
import org.leaf.cms.service.dto.ChatRoom;

public interface ChatService {

    public void forceUpdateChatRooms();

    public void notifyUpcomingAppointments();

    public List<ChatRoom> getChatRooms(Physician physician);
```

```
    public ChatRoom getChatroom(Long chatRoomId);

    public void recordChat(Long chatRoomId, Physician physician, String message);

    public List<ChatMessage> getChatRecordsHistory(Long clinicalCaseId);

    public List<ChatMessage> getChatRecords(Long chatRoomId, int
lastChatMessageIndex);

    public void endChatRoom(Long chatRoomId, boolean isClinicalCaseToBeClosed);

}
```

- **web-pcs/src/org.leaf.cms.action.service/ClinicalCaseService.java**

```
package org.leaf.cms.service;

import java.util.List;

import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.leaf.cms.model.User;

public interface ClinicalCaseService {

    public ClinicalCase saveClinicalCase(ClinicalCase clinicalCase);

    public ClinicalCase createClinicalCase(ClinicalCase clinicalCase);

    public ClinicalCase reactivateClinicalCase(ClinicalCase clinicalCase);

    public void reactivateClinicalCase(Long id);

    public void terminateClinicalCase(Long id);

    public void closeClinicalCase(Long id);

    public void ongoingClinicalCase(Long id);

    public ClinicalCase getClinicalCase(Long id);

    public List<ClinicalCase> getClinicalCasesThatCanHaveAnAppointment(User user,
Specialty specialty);

    public List<ClinicalCase> getClinicalCases(User user);

    public List<ClinicalCase> getClinicalCasesOfAnAppointmentConsultant(Physician
consultant);

}
```

- **web-pcs/src/org.leaf.cms.action.service/ConfigurationService.java**

```
package org.leaf.cms.service;

import org.leaf.cms.model.Configuration;

public interface ConfigurationService {

    public Configuration mainConfiguration();

    public void saveConfiguration(Configuration configuration);

}
```

- **web-pcs/src/org.leaf.cms.action.service/DiscussionThreadService.java**

```
package org.leaf.cms.service;

import java.util.List;

import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.DiscussionThreadReply;
import org.leaf.cms.model.Subscription;
```

```
import org.leaf.cms.model.User;
import org.springframework.data.domain.Page;

public interface DiscussionThreadService {

    public DiscussionThread findThreadById(Long id);

    public DiscussionThread saveDiscussionThread(DiscussionThread discussionThread);

    public DiscussionThreadReply saveDiscussionThreadReply(
                    DiscussionThreadReply discussionThreadReply);

    public DiscussionThread saveInitialDiscussionThread(
                    DiscussionThread discussionThread,
                    DiscussionThreadReply discussionThreadReply);

    public List<DiscussionThread> findAllValid();

    public List<DiscussionThread> findAllByCreatedBy(User createdBy);

    public List<DiscussionThread> findAllBySubscriptionFollower(User follower);

    public Page<DiscussionThread> findAllWithPaging(int pageNumber, int
itemsPerPage);

    public Long countByIsSeenFalse();

    public Boolean subscribe(Subscription subscription);

    public void unsubscribe(DiscussionThread thread, User follower);

    public void delete(DiscussionThread discussionThread);

    public Page<DiscussionThread> findAllByCreatedByWithPaging(User createdBy,
                    int pageNumber, int itemsPerPage);

    public Page<DiscussionThread> findAllValidWithPaging(int pageNumber,
                    int itemsPerPage);

    public Page<DiscussionThread> findAllBySubscriptionFollowerWithPaging(
                    User follower, int pageNumber, int itemsPerPage);

}
```

- **web-pcs/src/org.leaf.cms.action.service/EmailService.java**

```
package org.leaf.cms.service;

import java.util.Map;

public interface EmailService {

    public void sendEmail(
                        String templateLocation,
                        String subject,
                        String recipientAddress,
                        Map<String, Object> variables,
                        Map<String, String> fileSystemInlineVariables,
                        Map<String, String> classpathInlineVariables);

}
```

- **web-pcs/src/org.leaf.cms.action.service/FileManagementService.java**

```
package org.leaf.cms.service;

import java.io.File;
import java.util.List;

import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.FileManagement;
import org.leaf.cms.model.User;

public interface FileManagementService {
```

```
    public void uploadFiles(List<File> files, List<String> fileNames, List<String>
contentTypes, ClinicalCase clinicalCase, User uploadedBy);

    public void deleteFiles(List<Long> fileIdsForDeletion);

    public FileManagement getFile(Long id);

    public List<FileManagement> getFiles(ClinicalCase clinicalCase);

}
```

- **web-pcs/src/org.leaf.cms.action.service/PhysicianService.java**

```
package org.leaf.cms.service;

import java.util.List;

import org.leaf.cms.model.ClinicInfo;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.Specialty;
import org.springframework.data.domain.Page;

public interface PhysicianService {

    public List<Physician> getPhysicians();

    public List<Physician> getPhysicians(Specialty specialty);

    public Page<Physician> getPhysicians(Specialty specialty, int pageNumber, int
itemsPerPage);

    public Physician findPhysician(Long id);

    public Long countByEmail(String email);

    public Physician save(Physician physician);

    public Physician activateAccount(Long id, String verificationCode);

    public Physician findByEmail(String email);

    public Page<Physician> findAllWithPaging(int pageNumber, int itemsPerPage);

    public void saveWithClinicInfo(Physician physician, List<ClinicInfo> clinics);

    public Page<Physician> findAllActivatedWithPaging(int pageNumber, int
itemsPerPage);

    public Long countByMedLicenseId(Long medLicenseId);

    public void delete(Physician physician);

    public Physician findByEmailAndMedLicenseId(String email, Long medLicenseId);
}
```

- **web-pcs/src/org.leaf.cms.action.service/ReferenceService.java**

```
package org.leaf.cms.service;

import org.leaf.cms.model.Reference;
import org.springframework.data.domain.Page;

public interface ReferenceService {

    public Page<Reference> findAllWithPaging(int pageNumber, int itemsPerPage); }
```

- **web-pcs/src/org.leaf.cms.action.service/SMSEntryService.java**

```
package org.leaf.cms.service;

import java.util.Map;
import org.leaf.cms.model.SMSEntry;
import org.springframework.data.domain.Page;

public interface SMSEntryService {
```

```
    public Page<SMSEntry> findAllWithPaging(int pageNumber, int itemsPerPage);

    public SMSEntry saveMessageToBeSent(String templateLocation, String recipient,
Map<String, Object> variables);
    public SMSEntry save(SMSEntry entry);
    public SMSEntry findById(Long id); }
```

- **web-pcs/src/org.leaf.cms.action.service/SpecialtyService.java**

```
package org.leaf.cms.service;

import java.util.List;
import org.leaf.cms.model.Specialty;

public interface SpecialtyService {

    public List<Specialty> getSpecialty();

    public Specialty findSpecialty(String name);

    public Specialty findSpecialty(Long id); }
```

- **web-pcs/src/org.leaf.cms.action.service/UserService.java**

```
package org.leaf.cms.service;

import org.leaf.cms.model.User;

public interface UserService {

    public User findUser(String username, String password);

    public User findUser(Long userId);

    public User saveUser(User user); }
```

- **web-pcs/src/org.leaf.cms.action.service.dto/ChatRoom.java**

```
package org.leaf.cms.service.dto;

import java.util.List;

import org.joda.time.DateTime;
import org.leaf.cms.model.ChatMessage;

public class ChatRoom {

    private Long id;
    private List<ChatMessage> chatMessages;
    private Long requestingPhysicianId;
    private Long consultantId;
    private DateTime consultationEnd;
    private Long appointmentId;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public List<ChatMessage> getChatMessages() {
        return chatMessages;
    }

    public void setChatMessages(List<ChatMessage> chatMessages) {
        this.chatMessages = chatMessages;
    }

    public Long getRequestingPhysicianId() {
        return requestingPhysicianId;
    }
```

```
    public void setRequestingPhysicianId(Long requestingPhysicianId) {
        this.requestingPhysicianId = requestingPhysicianId;
    }

    public Long getConsultantId() {
        return consultantId;
    }

    public void setConsultantId(Long consultantId) {
        this.consultantId = consultantId;
    }

    public DateTime getConsultationEnd() {
        return consultationEnd;
    }

    public void setConsultationEnd(DateTime consultationEnd) {
        this.consultationEnd = consultationEnd;
    }

    public Long getAppointmentId() {
        return appointmentId;
    }

    public void setAppointmentId(Long appointmentId) {
        this.appointmentId = appointmentId;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.dto/ConsultationTrends.java**

```
package org.leaf.cms.service.dto;

public class ConsultationTrends {

    private int sunday;
    private int monday;
    private int tuesday;
    private int wednesday;
    private int thursday;
    private int friday;
    private int saturday;

    public int getSunday() {
        return sunday;
    }

    public void setSunday(int sunday) {
        this.sunday = sunday;
    }

    public int getMonday() {
        return monday;
    }

    public void setMonday(int monday) {
        this.monday = monday;
    }

    public int getTuesday() {
        return tuesday;
    }

    public void setTuesday(int tuesday) {
        this.tuesday = tuesday;
    }

    public int getWednesday() {
        return wednesday;
    }

    public void setWednesday(int wednesday) {
        this.wednesday = wednesday;
    }
```

```
    public int getThursday() {
        return thursday;
    }

    public void setThursday(int thursday) {
        this.thursday = thursday;
    }

    public int getFriday() {
        return friday;
    }

    public void setFriday(int friday) {
        this.friday = friday;
    }

    public int getSaturday() {
        return saturday;
    }

    public void setSaturday(int saturday) {
        this.saturday = saturday;
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/AppointmentServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.joda.time.DateTime;
import org.joda.time.DateTimeConstants;
import org.joda.time.LocalDate;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.User;
import org.leaf.cms.model.repository.AppointmentRepository;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.ClinicalCaseService;
import org.leaf.cms.service.dto.ConsultationTrends;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.Assert;

@Service("appointmentService")
public class AppointmentServiceImpl implements AppointmentService {

    @Autowired
    private AppointmentRepository appointmentRepository;

    @Autowired
    private ClinicalCaseService clinicalCaseService;

    @Override
    public List<Appointment> getAppointments(Physician requestingPhysician, Physician
consultant) {
        return
appointmentRepository.findAppointmentOfRequestingPhysicianAndConsultant(requestingPhy
sician, consultant);
    }

    @Override
    public List<Appointment> getAppointments(Physician requestingPhysician, Physician
consultant, LocalDate start, LocalDate end) {
```

```
        return
appointmentRepository.findAppointmentOfRequestingPhysicianAndConsultant(requestingPhy
sician, consultant, start, end);
    }

    @Override
    public List<Appointment> getAppointments(DateTime before, DateTime after) {
        return
appointmentRepository.findByConsultationStartBeforeAndConsultationEndAfterAndIsValid(
before, after, true);
    }

    public Page<Appointment> findWithPaging(int page, int itemsPerPage) {
        Pageable pageRequest = new PageRequest(page - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
        return appointmentRepository.findAll(pageRequest);
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public Appointment createAppointment(Appointment appointment, boolean
isNewClinicalCase) {
        ClinicalCase clinicalCase = appointment.getClinicalCase();
        if (isNewClinicalCase) {
            ClinicalCase savedClinicalCase =
clinicalCaseService.createClinicalCase(clinicalCase);
            appointment.setClinicalCase(savedClinicalCase);
        } else {
            clinicalCaseService.reactivateClinicalCase(clinicalCase.getId());
        }
        appointment.setCreatedOn(DateTime.now());
        appointment.setIsValid(true);
        return appointmentRepository.save(appointment);
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public Appointment rescheduleAppointment(Appointment appointment) {
        return appointmentRepository.save(appointment);
    }

    @Override
    public void cancelAppointment(Long id) {
        Appointment appointment = appointmentRepository.findOne(id);
        if (appointment != null) {
            cancelAppointment(appointment);
        }
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public void cancelAppointment(Appointment appointment) {
        ClinicalCase clinicalCase = appointment.getClinicalCase();
        clinicalCaseService.terminateClinicalCase(clinicalCase.getId());
        appointmentRepository.delete(appointment);
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public Appointment terminateAppointment(Appointment appointment) {
        ClinicalCase clinicalCase = appointment.getClinicalCase();
        clinicalCaseService.terminateClinicalCase(clinicalCase.getId());
        appointment.setIsValid(false);
        return appointmentRepository.save(appointment);
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public Appointment closeAppointment(Appointment appointment) {
        ClinicalCase clinicalCase = appointment.getClinicalCase();
        clinicalCaseService.closeClinicalCase(clinicalCase.getId());
        appointment.setIsValid(false);
        return appointmentRepository.save(appointment);
    }

    @Override
```

```
    public Page<Appointment> findAppointment(User physician, int pageNumber, int
itemsPerPage) {
        Assert.notNull(physician, "Physician should not be null");
        Pageable pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
        return appointmentRepository.findAppointmentOfPhysician(physician,
pageRequest);
    }

    @Override
    public Appointment findAppointment(Long id) {
        return appointmentRepository.findOne(id);
    }

    @Override
    public Appointment findAppointment(ClinicalCase clinicalCase) {
        return appointmentRepository.findByClinicalCaseAndIsValid(clinicalCase,
true);
    }

    @Override
    public List<Object[]> getAppointmentReportOfRequestingPhysicians(LocalDate
startDate, LocalDate endDate) {
        return appointmentRepository.countAppointmentAsRequestingPhysician(startDate,
endDate);
    }

    @Override
    public List<Object[]> getAppointmentReportOfConsultants(LocalDate startDate,
LocalDate endDate) {
        return appointmentRepository.countAppointmentAsConsultant(startDate,
endDate);
    }

    @Override
    public Map<String, ConsultationTrends> getConsultationTrends() {
        DateTime now = DateTime.now();
        LocalDate today = now.toLocalDate();
        int currentDayOfWeek = today.getDayOfWeek();
        int statisticsToday = appointmentRepository.countByConsultationDate(today);

        ConsultationTrends thisWeekTrends = new ConsultationTrends();
        recordConsultationTrend(thisWeekTrends, currentDayOfWeek, statisticsToday);

        LocalDate localDate = today.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics =
appointmentRepository.countByConsultationDate(localDate);
            recordConsultationTrend(thisWeekTrends, dayOfWeek, statistics);
            localDate = localDate.minusDays(1);
        }

        localDate = today.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics =
appointmentRepository.countByConsultationDate(localDate);
            recordConsultationTrend(thisWeekTrends, dayOfWeek, statistics);
            localDate = localDate.plusDays(1);
        }

        Map<String, ConsultationTrends> trends = new HashMap<String,
ConsultationTrends>();
        trends.put("thisWeek", thisWeekTrends);

        LocalDate dayLastWeek = today.minusWeeks(1);
        int statisticsLastWeekOnSameWeekDay =
appointmentRepository.countByConsultationDate(dayLastWeek);

        ConsultationTrends lastWeekTrends = new ConsultationTrends();
        recordConsultationTrend(lastWeekTrends, currentDayOfWeek,
statisticsLastWeekOnSameWeekDay);

        LocalDate localDateLastWeek = dayLastWeek.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateLastWeek);
```

```
            recordConsultationTrend(lastWeekTrends, dayOfWeek, statistics);
            localDateLastWeek = localDateLastWeek.minusDays(1);
        }

        localDateLastWeek = dayLastWeek.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateLastWeek);
            recordConsultationTrend(lastWeekTrends, dayOfWeek, statistics);
            localDateLastWeek = localDateLastWeek.plusDays(1);
        }

        trends.put("lastWeek", lastWeekTrends);

        LocalDate dayNextWeek = today.plusWeeks(1);
        int statisticsNextWeekOnSameWeekDay =
appointmentRepository.countByConsultationDate(dayNextWeek);

        ConsultationTrends nextWeekTrends = new ConsultationTrends();
        recordConsultationTrend(nextWeekTrends, currentDayOfWeek,
statisticsNextWeekOnSameWeekDay);

        LocalDate localDateNextWeek = dayNextWeek.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateNextWeek);
            recordConsultationTrend(nextWeekTrends, dayOfWeek, statistics);
            localDateNextWeek = localDateNextWeek.minusDays(1);
        }

        localDateNextWeek = dayNextWeek.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateNextWeek);
            recordConsultationTrend(nextWeekTrends, dayOfWeek, statistics);
            localDateNextWeek = localDateNextWeek.plusDays(1);
        }

        trends.put("nextWeek", nextWeekTrends);

        return trends;
    }

    @Override
    public Map<String, ConsultationTrends> getConsultationTrends(User physician) {
        DateTime now = DateTime.now();
        LocalDate today = now.toLocalDate();
        int currentDayOfWeek = today.getDayOfWeek();
        int statisticsToday = appointmentRepository.countByConsultationDate(today,
physician);

        ConsultationTrends thisWeekTrends = new ConsultationTrends();
        recordConsultationTrend(thisWeekTrends, currentDayOfWeek, statisticsToday);

        LocalDate localDate = today.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics = appointmentRepository.countByConsultationDate(localDate,
physician);
            recordConsultationTrend(thisWeekTrends, dayOfWeek, statistics);
            localDate = localDate.minusDays(1);
        }

        localDate = today.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics = appointmentRepository.countByConsultationDate(localDate,
physician);
            recordConsultationTrend(thisWeekTrends, dayOfWeek, statistics);
            localDate = localDate.plusDays(1);
        }

        Map<String, ConsultationTrends> trends = new HashMap<String,
ConsultationTrends>();
        trends.put("thisWeek", thisWeekTrends);

        LocalDate dayLastWeek = today.minusWeeks(1);
```

```
        int statisticsLastWeekOnSameWeekDay =
appointmentRepository.countByConsultationDate(dayLastWeek, physician);

        ConsultationTrends lastWeekTrends = new ConsultationTrends();
        recordConsultationTrend(lastWeekTrends, currentDayOfWeek,
statisticsLastWeekOnSameWeekDay);

        LocalDate localDateLastWeek = dayLastWeek.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateLastWeek, physician);
            recordConsultationTrend(lastWeekTrends, dayOfWeek, statistics);
            localDateLastWeek = localDateLastWeek.minusDays(1);
        }

        localDateLastWeek = dayLastWeek.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateLastWeek, physician);
            recordConsultationTrend(lastWeekTrends, dayOfWeek, statistics);
            localDateLastWeek = localDateLastWeek.plusDays(1);
        }

        trends.put("lastWeek", lastWeekTrends);

        LocalDate dayNextWeek = today.plusWeeks(1);
        int statisticsNextWeekOnSameWeekDay =
appointmentRepository.countByConsultationDate(dayNextWeek, physician);

        ConsultationTrends nextWeekTrends = new ConsultationTrends();
        recordConsultationTrend(nextWeekTrends, currentDayOfWeek,
statisticsNextWeekOnSameWeekDay);

        LocalDate localDateNextWeek = dayNextWeek.minusDays(1);
        for (int dayOfWeek = currentDayOfWeek - 1; dayOfWeek > 0; dayOfWeek--) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateNextWeek, physician);
            recordConsultationTrend(nextWeekTrends, dayOfWeek, statistics);
            localDateNextWeek = localDateNextWeek.minusDays(1);
        }

        localDateNextWeek = dayNextWeek.plusDays(1);
        for (int dayOfWeek = currentDayOfWeek + 1; dayOfWeek < 8; dayOfWeek++) {
            int statistics =
appointmentRepository.countByConsultationDate(localDateNextWeek, physician);
            recordConsultationTrend(nextWeekTrends, dayOfWeek, statistics);
            localDateNextWeek = localDateNextWeek.plusDays(1);
        }

        trends.put("nextWeek", nextWeekTrends);

        return trends;
    }

    private void recordConsultationTrend(ConsultationTrends consultationTrends, int
dayOfWeek, int statistics) {
        switch (dayOfWeek) {
        case DateTimeConstants.MONDAY:
            consultationTrends.setMonday(statistics);
            break;
        case DateTimeConstants.TUESDAY:
            consultationTrends.setTuesday(statistics);
            break;
        case DateTimeConstants.WEDNESDAY:
            consultationTrends.setWednesday(statistics);
            break;
        case DateTimeConstants.THURSDAY:
            consultationTrends.setThursday(statistics);
            break;
        case DateTimeConstants.FRIDAY:
            consultationTrends.setFriday(statistics);
            break;
        case DateTimeConstants.SATURDAY:
            consultationTrends.setSaturday(statistics);
            break;
        case DateTimeConstants.SUNDAY:
```

```
            consultationTrends.setSunday(statistics);
            break;
        }
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public void setAppointmentNotificationSchedule(Long appointmentId, int
minutesBeforeAppointment) {
        if (minutesBeforeAppointment != -1) {
            Appointment appointment = appointmentRepository.findOne(appointmentId);
            DateTime consultationStart = appointment.getConsultationStart();
            DateTime notificationSmsSchedule =
consultationStart.minusMinutes(minutesBeforeAppointment);
            appointment.setNotificationSmsSchedule(notificationSmsSchedule);
            appointment.setNotificationSmsScheduleSent(false);
            appointmentRepository.save(appointment);
        }
    }

    @Override
    public List<Appointment> getUpcomingAppointmentsToBeNotified(DateTime now) {
        return
appointmentRepository.findByNotificationSmsScheduleAndNotificationSmsScheduleSentNot(
now, true);
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public void saveAppointments(List<Appointment> appointments) {
        appointmentRepository.save(appointments);
    }

    @Override
    public List<Appointment> getAppointmentsAsRequestingPhysician(Physician
physician, LocalDate startDate, LocalDate endDate) {
        return appointmentRepository.findAppointmentOfRequestingPhysician(physician,
startDate, endDate);
    }

    @Override
    public List<Appointment> getAppointmentsAsConsultant(Physician physician,
LocalDate startDate, LocalDate endDate) {
        return appointmentRepository.findAppointmentOfConsultant(physician,
startDate, endDate);
    }

    @Override
    public List<Appointment> getActiveAppointmentsBefore(DateTime before) {
        return appointmentRepository.findByConsultationEndBeforeAndIsValid(before,
true);
    }

    @Override
    public Long countUnseenAppointments(User physician) {
        return
appointmentRepository.countByRequestingPhysicianAndIsSeenRequestingPhysicianFalse(phy
sician)
                +
appointmentRepository.countByConsultantAndIsSeenConsultantFalse(physician);
    }

    @Override
    public void markAppointmentAsSeenByRequestingPhyisician(Appointment appointment)
{
        appointment.setIsSeenRequestingPhysician(true);
        appointmentRepository.save(appointment);
    }

    @Override
    public void markAppointmentAsSeenByConsultant(Appointment appointment) {
        appointment.setIsSeenConsultant(true);
        appointmentRepository.save(appointment);
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/ChatServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import org.joda.time.DateTime;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.model.ChatMessage;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.Physician;
import org.leaf.cms.model.repository.ChatMessagesRepository;
import org.leaf.cms.service.AppointmentService;
import org.leaf.cms.service.ChatService;
import org.leaf.cms.service.ClinicalCaseService;
import org.leaf.cms.service.SMSEntryService;
import org.leaf.cms.service.dto.ChatRoom;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Async;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service("chatService")
@EnableScheduling
public class ChatServiceImpl implements ChatService {

    @Autowired
    private ChatMessagesRepository chatMessagesRepository;

    @Autowired
    private AppointmentService appointmentService;

    @Autowired
    private ClinicalCaseService clinicalCaseService;

    @Autowired
    private SMSEntryService smsEntryService;

    private Map<Long, ChatRoom> chatRooms = new HashMap<>();

    @Override
    @Scheduled(cron = "0 0/1 * * * ?")
    @Transactional(rollbackFor = Exception.class)
    public void forceUpdateChatRooms() {
        synchronized (chatRooms) {
            DateTime now = DateTime.now();
            List<Long> toBeRemoved = new ArrayList<>();
            for (Entry<Long, ChatRoom> chatRoom : chatRooms.entrySet()) {
                ChatRoom room = chatRoom.getValue();
                DateTime consultationEnd = room.getConsultationEnd();
                if (now.isAfter(consultationEnd)) {
                    toBeRemoved.add(chatRoom.getKey());
                }
            }
            for (Long id : toBeRemoved) {
                ChatRoom chatRoom = chatRooms.remove(id);
                Appointment appointment =
appointmentService.findAppointment(chatRoom.getAppointmentId());
                appointment.setIsSeenRequestingPhysician(true);
                appointment.setIsSeenConsultant(true);
                appointmentService.terminateAppointment(appointment);
            }

            List<Appointment> appointments = appointmentService.getAppointments(now,
now);
            for (Appointment appointment : appointments) {
                ClinicalCase clinicalCase = appointment.getClinicalCase();
                Long chatRoomId = clinicalCase.getId();
                if (!chatRooms.containsKey(chatRoomId)) {
                    clinicalCaseService.ongoingClinicalCase(chatRoomId);
```

```
                    ChatRoom chatRoom = new ChatRoom();
                    chatRoom.setId(clinicalCase.getId());
                    chatRoom.setConsultationEnd(appointment.getConsultationEnd());

chatRoom.setRequestingPhysicianId(appointment.getRequestingPhysician().getId());
                    chatRoom.setConsultantId(appointment.getConsultant().getId());
                    chatRoom.setAppointmentId(appointment.getId());

                    List<ChatMessage> messages =
chatMessagesRepository.findByClinicalCaseOrderByCreatedOnAsc(clinicalCase);
                    chatRoom.setChatMessages(messages);

                    chatRooms.put(chatRoomId, chatRoom);
                }
            }
        }
    }

    @Override
    @Scheduled(cron = "0 0/1 * * * ?")
    public void notifyUpcomingAppointments() {
        notifyUpcomingAppointmentsAsync();
    }

    @Async
    private void notifyUpcomingAppointmentsAsync() {
        DateTime now = DateTime.now();
        DateTime truncatedDateTime = now.secondOfMinute().setCopy(0);
        truncatedDateTime = truncatedDateTime.millisOfSecond().setCopy(0);
        List<Appointment> upcomingAppointmentsToBeNotified =
appointmentService.getUpcomingAppointmentsToBeNotified(truncatedDateTime);
        for (Appointment appointment : upcomingAppointmentsToBeNotified) {
            Physician requestingPhysician = appointment.getRequestingPhysician();
            Physician consultant = appointment.getConsultant();

            String consultationDateString =
consultationDateStringBuilder(appointment.getConsultationStart(),
appointment.getConsultationEnd());

            Map<String, Object> requestingPhysicianMap = new HashMap<>();
            requestingPhysicianMap.put("name", requestingPhysician.getFullName());
            requestingPhysicianMap.put("consultationDateString",
consultationDateString);
            requestingPhysicianMap.put("otherName", consultant.getFullName());
            smsEntryService.saveMessageToBeSent("notifyappointment_sms.vm",
requestingPhysician.getCellphoneNumber(), requestingPhysicianMap);

            Map<String, Object> consultantMap = new HashMap<>();
            consultantMap.put("name", consultant.getFullName());
            consultantMap.put("consultationDateString", consultationDateString);
            consultantMap.put("otherName", requestingPhysician.getFullName());
            smsEntryService.saveMessageToBeSent("notifyappointment_sms.vm",
consultant.getCellphoneNumber(), consultantMap);

            appointment.setNotificationSmsScheduleSent(true);
        }
        appointmentService.saveAppointments(upcomingAppointmentsToBeNotified);
    }

    private String consultationDateStringBuilder(DateTime start, DateTime end) {
        StringBuilder stringBuilder = new StringBuilder();
        if (start.getDayOfYear() == end.getDayOfYear()) {
            stringBuilder.append(start.dayOfWeek().getAsText())
                    .append(", ")
                    .append(start.dayOfMonth().getAsText())
                    .append(' ')
                    .append(start.monthOfYear().getAsText())
                    .append(' ')
                    .append(start.year().getAsText())
                    .append(' ')
                    .append(start.hourOfDay().getAsText())
                    .append(':')
                    .append(start.minuteOfHour().getAsText())
                    .append(" - ")
                    .append(end.hourOfDay().getAsText())
```

```
                        .append(':')
                        .append(end.minuteOfHour().getAsText());
            } else {
                stringBuilder.append(start.dayOfMonth().getAsText())
                        .append(' ')
                        .append(start.monthOfYear().getAsText())
                        .append(' ')
                        .append(start.year().getAsText())
                        .append(' ')
                        .append(start.hourOfDay().getAsText())
                        .append(':')
                        .append(start.minuteOfHour().getAsText())
                        .append(" - ")
                        .append(end.dayOfMonth().getAsText())
                        .append(' ')
                        .append(end.monthOfYear().getAsText())
                        .append(' ')
                        .append(end.year().getAsText())
                        .append(' ')
                        .append(end.hourOfDay().getAsText())
                        .append(':')
                        .append(end.minuteOfHour().getAsText());
            }
            return stringBuilder.toString();
        }

        @Override
        public List<ChatRoom> getChatRooms(Physician physician) {
            List<ChatRoom> userCurrentChatRooms = new ArrayList<>();
            for (Entry<Long, ChatRoom> chatRoom : chatRooms.entrySet()) {
                ChatRoom room = chatRoom.getValue();
                long requestingPhysicianId = room.getRequestingPhysicianId();
                long consultantId = room.getConsultantId();
                long physicianId = physician.getId();
                if (physicianId == requestingPhysicianId || physicianId == consultantId)
{
                    userCurrentChatRooms.add(room);
                }
            }
            return userCurrentChatRooms;
        }

        @Override
        public ChatRoom getChatroom(Long chatRoomId) {
            return chatRooms.get(chatRoomId);
        }

        @Override
        @Transactional(rollbackFor = Exception.class)
        public void recordChat(Long chatRoomId, Physician physician, String message) {
            ChatRoom chatRoom = chatRooms.get(chatRoomId);
            ClinicalCase clinicalCase = clinicalCaseService.getClinicalCase(chatRoomId);

            ChatMessage chatMessage = new ChatMessage();
            chatMessage.setClinicalCase(clinicalCase);
            chatMessage.setPhysician(physician);
            chatMessage.setMessage(message);
            chatMessage.setIsValid(true);
            chatMessage.setCreatedOn(DateTime.now());
            chatMessagesRepository.save(chatMessage);

            chatRoom.getChatMessages().add(chatMessage);
        }

        @Override
        public List<ChatMessage> getChatRecordsHistory(Long clinicalCaseId) {
            ClinicalCase clinicalCase =
clinicalCaseService.getClinicalCase(clinicalCaseId);
            return
chatMessagesRepository.findByClinicalCaseOrderByCreatedOnAsc(clinicalCase);
        }

        @Override
        public List<ChatMessage> getChatRecords(Long chatRoomId, int
lastChatMessageIndex) {
            ChatRoom chatRoom = chatRooms.get(chatRoomId);
```

```
            if (chatRoom == null) {
                throw new NullPointerException("Chat room does not exist.");
            } else {
                List<ChatMessage> chatMessages = chatRoom.getChatMessages();
                int indexFrom = lastChatMessageIndex + 1;
                int indexTo = chatMessages.size();
                return chatMessages.subList(indexFrom, indexTo);
            }
        }

        @Override
        @Transactional(rollbackFor = Exception.class)
        public void endChatRoom(Long chatRoomId, boolean isClinicalCaseToBeClosed) {
            ChatRoom chatRoom = chatRooms.remove(chatRoomId);
            Appointment appointment =
    appointmentService.findAppointment(chatRoom.getAppointmentId());
            appointment.setIsSeenRequestingPhysician(true);
            appointment.setIsSeenConsultant(true);
            if (isClinicalCaseToBeClosed) {
                appointmentService.closeAppointment(appointment);
            } else {
                appointmentService.terminateAppointment(appointment);
            }
        }

    }
```

- **web-pcs/src/org.leaf.cms.action.service.impl/ClinicalCaseServiceImpl.java**

```
    package org.leaf.cms.service.impl;

    import java.util.List;

    import org.joda.time.DateTime;
    import org.leaf.cms.enums.ClinicalCaseStatusEnum;
    import org.leaf.cms.model.ClinicalCase;
    import org.leaf.cms.model.Physician;
    import org.leaf.cms.model.Specialty;
    import org.leaf.cms.model.User;
    import org.leaf.cms.model.repository.AppointmentRepository;
    import org.leaf.cms.model.repository.ClinicalCaseRepository;
    import org.leaf.cms.service.ClinicalCaseService;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Service;
    import org.springframework.transaction.annotation.Transactional;

    @Service("clinicalCaseService")
    public class ClinicalCaseServiceImpl implements ClinicalCaseService {

        @Autowired
        private ClinicalCaseRepository clinicalCaseRepository;

        @Autowired
        private AppointmentRepository appointmentRepository;

        @Override
        public ClinicalCase saveClinicalCase(ClinicalCase clinicalCase) {
            return clinicalCaseRepository.save(clinicalCase);
        }

        @Override
        @Transactional(rollbackFor = Exception.class)
        public ClinicalCase createClinicalCase(ClinicalCase clinicalCase) {
            clinicalCase.setCreatedOn(DateTime.now());
            clinicalCase.setIsValid(true);
            clinicalCase.setStatus(ClinicalCaseStatusEnum.ACTIVE.getStatus());
            return clinicalCaseRepository.save(clinicalCase);
        }

        @Override
        @Transactional(rollbackFor = Exception.class)
        public ClinicalCase reactivateClinicalCase(ClinicalCase clinicalCase) {
            clinicalCase.setStatus(ClinicalCaseStatusEnum.ACTIVE.getStatus());
            return clinicalCaseRepository.save(clinicalCase);
        }
```

```
    @Override
    @Transactional(rollbackFor = Exception.class)
    public void reactivateClinicalCase(Long id) {
        clinicalCaseRepository.updateStatus(id,
ClinicalCaseStatusEnum.ACTIVE.getStatus());
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public void terminateClinicalCase(Long id) {
        clinicalCaseRepository.updateStatus(id,
ClinicalCaseStatusEnum.TERMINATED.getStatus());
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public void closeClinicalCase(Long id) {
        clinicalCaseRepository.updateStatus(id,
ClinicalCaseStatusEnum.CLOSED.getStatus());
    }

    @Override
    @Transactional(rollbackFor = Exception.class)
    public void ongoingClinicalCase(Long id) {
        clinicalCaseRepository.updateStatus(id,
ClinicalCaseStatusEnum.ONGOING.getStatus());
    }

    @Override
    public ClinicalCase getClinicalCase(Long id) {
        return clinicalCaseRepository.findOne(id);
    }

    @Override
    public List<ClinicalCase> getClinicalCasesThatCanHaveAnAppointment(User user,
Specialty specialty) {
        return clinicalCaseRepository.findByCreatedByAndStatusAndSpecialty(user,
ClinicalCaseStatusEnum.TERMINATED.getStatus(), specialty);
    }

    @Override
    public List<ClinicalCase> getClinicalCases(User user) {
        return clinicalCaseRepository.findByCreatedBy(user);
    }

    @Override
    public List<ClinicalCase> getClinicalCasesOfAnAppointmentConsultant(Physician
consultant) {
        return
appointmentRepository.findActiveAndOngoingClinicalCasesFromAppointmentOfAConsultant(c
onsultant);
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/ClinicalInfoServiceImpl.java**

```
package org.leaf.cms.service.impl;

import org.leaf.cms.model.repository.ClinicInfoRepository;
import org.leaf.cms.service.ClinicInfoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service("clinicInfoService")
public class ClinicInfoServiceImpl implements ClinicInfoService{

    @Autowired
    private ClinicInfoRepository clinicInfoRepository;
}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/ConfigurationServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.Collections;
```

```
import java.util.List;

import org.joda.time.DateTime;
import org.leaf.cms.model.Configuration;
import org.leaf.cms.model.repository.ConfigurationRepository;
import org.leaf.cms.service.ConfigurationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service("configurationService")
public class ConfigurationServiceImpl implements ConfigurationService {

    @Autowired
    private ConfigurationRepository configurationRepository;

    @Override
    public Configuration mainConfiguration() {
        List<Configuration> configurations =
configurationRepository.findByIsValidTrue();
        Configuration configuration = null;
        if (!configurations.isEmpty()) {
            Collections.shuffle(configurations);
            configuration = configurations.get(0);
        }
        if (configuration == null) {
            configuration = new Configuration();
        }
        return configuration;
    }

    @Override
    public void saveConfiguration(Configuration configuration) {
        if (configuration.getId() == null) {
            configuration.setCreatedOn(new DateTime());
            configuration.setIsValid(Boolean.TRUE);
        }
        configurationRepository.save(configuration);
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/DiscussionThreadServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.ArrayList;
import java.util.List;

import javax.transaction.Transactional;

import org.leaf.cms.model.DiscussionThread;
import org.leaf.cms.model.DiscussionThreadReply;
import org.leaf.cms.model.Subscription;
import org.leaf.cms.model.User;
import org.leaf.cms.model.repository.DiscussionThreadReplyRepository;
import org.leaf.cms.model.repository.DiscussionThreadRepository;
import org.leaf.cms.model.repository.SubscriptionRepository;
import org.leaf.cms.service.DiscussionThreadService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageImpl;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;

public class DiscussionThreadServiceImpl implements DiscussionThreadService {

    @Autowired
    private DiscussionThreadRepository discussionThreadRepository;

    @Autowired
    private DiscussionThreadReplyRepository discussionThreadReplyRepository;

    @Autowired
    private SubscriptionRepository subscriptionRepository;

    @Override
```

```
    public DiscussionThread findThreadById(Long id) {
        return discussionThreadRepository.findOne(id);
    }

    @Override
    public DiscussionThread saveDiscussionThread(DiscussionThread discussionThread) {
        return discussionThreadRepository.saveAndFlush(discussionThread);
    }

    @Override
    public DiscussionThreadReply saveDiscussionThreadReply(DiscussionThreadReply
discussionThreadReply) {
        return discussionThreadReplyRepository.saveAndFlush(discussionThreadReply);
    }

    @Override
    @Transactional(rollbackOn = Exception.class)
    public DiscussionThread saveInitialDiscussionThread(DiscussionThread
discussionThread, DiscussionThreadReply discussionThreadReply) {
        if ((discussionThread = discussionThreadRepository.save(discussionThread)) !=
null) {
            discussionThreadReply.setThread(discussionThread);
            discussionThreadReplyRepository.save(discussionThreadReply);
        }
        return discussionThread;
    }

    @Override
    public List<DiscussionThread> findAllValid() {
        List<Object[]> discussionThreadsQuery =
discussionThreadRepository.findValidDiscussionThreads();
        List<DiscussionThread> discussionThreads = new ArrayList<>();
        for (Object[] data : discussionThreadsQuery) {
            discussionThreads.add((DiscussionThread) data[0]);
        }
        return discussionThreads;
    }

    @Override
    public Page<DiscussionThread> findAllValidWithPaging(int pageNumber, int
itemsPerPage){
    PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage);
    Page<Object[]> result =
discussionThreadRepository.findValidDiscussionThreads(pageRequest);
    List<DiscussionThread> discussionThreads = new ArrayList<DiscussionThread>();
    for (Object[] data : result) {
            discussionThreads.add((DiscussionThread) data[0]);
        }

    Page<DiscussionThread> page = new PageImpl<DiscussionThread>(discussionThreads,
pageRequest, result.getTotalElements());
    return page;
    }

    @Override
    public List<DiscussionThread> findAllByCreatedBy(User createdBy) {
        List<Object[]> discussionThreadsQuery =
discussionThreadRepository.findValidDiscussionThreads(createdBy);
        List<DiscussionThread> discussionThreads = new ArrayList<>();
        for (Object[] data : discussionThreadsQuery) {
            discussionThreads.add((DiscussionThread) data[0]);
        }
        return discussionThreads;
    }

    @Override
    public Page<DiscussionThread> findAllByCreatedByWithPaging(User createdBy, int
pageNumber, int itemsPerPage){
    PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage);
    Page<Object[]> result =
discussionThreadRepository.findValidDiscussionThreads(createdBy, pageRequest);
    List<DiscussionThread> discussionThreads = new ArrayList<DiscussionThread>();
    for (Object[] data : result) {
            discussionThreads.add((DiscussionThread) data[0]);
        }
```

```
    Page<DiscussionThread> page = new PageImpl<DiscussionThread>(discussionThreads,
pageRequest, result.getTotalElements());
    return page;
    }

    @Override
    public Page<DiscussionThread> findAllWithPaging(int pageNumber, int itemsPerPage)
{
        PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
        return discussionThreadRepository.findByIsValidTrue(pageRequest);
    }

    @Override
    public List<DiscussionThread> findAllBySubscriptionFollower(User follower) {
    List<Object[]> subscriptionsQuery =
subscriptionRepository.subscriptionsOfFollower(follower);
        List<Subscription> subscriptions = new ArrayList<>();
        for (Object[] data : subscriptionsQuery) {
            subscriptions.add((Subscription) data[0]);
        }
        List<DiscussionThread> threads = new ArrayList<DiscussionThread>();
        if (subscriptions != null && subscriptions.size() > 0) {
            for (Subscription subscription : subscriptions) {
                threads.add(subscription.getThread());
            }
        }
        return threads;
    }

    @Override
    public Page<DiscussionThread> findAllBySubscriptionFollowerWithPaging(User
follower, int pageNumber, int itemsPerPage) {
    PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage);
    Page<Object[]> subscriptionsQuery =
subscriptionRepository.subscriptionsOfFollower(follower, pageRequest);
        List<Subscription> subscriptions = new ArrayList<>();
        for (Object[] data : subscriptionsQuery) {
            subscriptions.add((Subscription) data[0]);
        }
        List<DiscussionThread> threads = new ArrayList<DiscussionThread>();
        if (subscriptions != null && subscriptions.size() > 0) {
            for (Subscription subscription : subscriptions) {
                threads.add(subscription.getThread());
            }
        }

        Page<DiscussionThread> page = new PageImpl<DiscussionThread>(threads,
pageRequest, subscriptionsQuery.getTotalElements());
        return page;
    }

    @Override
    public Long countByIsSeenFalse() {
        return discussionThreadRepository.countByIsSeenFalseAndIsValidTrue();
    }

    @Override
    public Boolean subscribe(Subscription subscription) {
        Subscription subs = subscriptionRepository.save(subscription);
        return subs != null;
    }

    @Override
    public void unsubscribe(DiscussionThread thread, User follower) {
        Subscription sub = subscriptionRepository.findByThreadAndFollower(thread,
follower);
        subscriptionRepository.delete(sub);

    }

    @Override
    public void delete(DiscussionThread discussionThread) {
        discussionThread.setIsValid(Boolean.FALSE);
        discussionThreadRepository.save(discussionThread);
```

```
        List<Subscription> subs =
subscriptionRepository.findByThread(discussionThread);
        if (subs != null && subs.size() > 0) {
            for (Subscription sub : subs) {
                subscriptionRepository.delete(sub);
            }
        }

    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/EmailServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.Map;
import java.util.Map.Entry;

import javax.mail.internet.MimeMessage;

import org.apache.velocity.app.VelocityEngine;
import org.leaf.cms.service.EmailService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.FileSystemResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.ui.velocity.VelocityEngineUtils;

@Service("emailService")
public class EmailServiceImpl implements EmailService {

    private static final Logger LOGGER =
LoggerFactory.getLogger(EmailServiceImpl.class);

    @Autowired
    private JavaMailSender emailSender;

    @Autowired
    private VelocityEngine velocityEngine;

    @Override
    public void sendEmail(
                          final String templateLocation,
                          final String subject,
                          final String recipientAddress,
                          final Map<String, Object> variables,
                          final Map<String, String> fileSystemInlineVariables,
                          final Map<String, String> classpathInlineVariables) {
        Thread sendEmailThread = new Thread(new Runnable() {

                    @Override
                    public void run() {
                            try {
                    MimeMessage mimeMessage = emailSender.createMimeMessage();
                    MimeMessageHelper messageHelper = new
MimeMessageHelper(mimeMessage, true);
                    messageHelper.setTo(recipientAddress);
                    messageHelper.setSubject(subject);

                    String emailText =
VelocityEngineUtils.mergeTemplateIntoString(velocityEngine, templateLocation, "UTF-
8", variables);
                    messageHelper.setText(emailText, true);

                    if (fileSystemInlineVariables != null) {
                        for (Entry<String, String> fileSystemInlineVariable :
fileSystemInlineVariables.entrySet()) {
                            FileSystemResource fileSystemResource1 = new
FileSystemResource(fileSystemInlineVariable.getValue());

messageHelper.addInline(fileSystemInlineVariable.getKey(), fileSystemResource1);
```

```
                        }
                    }

                    if (classpathInlineVariables != null) {
                        for (Entry<String, String> classpathInlineVariable :
classpathInlineVariables.entrySet()) {
                            ClassPathResource fileSystemResource2 = new
ClassPathResource(classpathInlineVariable.getValue());
                            messageHelper.addInline(classpathInlineVariable.getKey(),
fileSystemResource2);
                        }
                    }

                    emailSender.send(mimeMessage);
                } catch (Exception e) {
                    if (LOGGER.isErrorEnabled()) {
                        LOGGER.error(e.getMessage(), e);
                    }
                }
                    }
            });
        sendEmailThread.start();
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/FileManagementServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

import org.apache.commons.io.FileUtils;
import org.joda.time.DateTime;
import org.leaf.cms.model.ClinicalCase;
import org.leaf.cms.model.FileManagement;
import org.leaf.cms.model.User;
import org.leaf.cms.model.repository.FileManagementRepository;
import org.leaf.cms.service.FileManagementService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service("fileManagementService")
public class FileManagementServiceImpl implements FileManagementService {

    private static final Logger LOGGER =
LoggerFactory.getLogger(FileManagementServiceImpl.class);

    @Autowired
    private FileManagementRepository fileManagementRepository;

    @Value("${clinicalcase.files.location}")
    private String filesLocation;

    @Override
    @Transactional
    public void uploadFiles(List<File> files, List<String> fileNames, List<String>
contentTypes, ClinicalCase clinicalCase, User uploadedBy) {
        List<FileManagement> recordsToSave = new ArrayList<>();
        DateTime now = DateTime.now();
        File filesDirectory = new File(filesLocation);
        if (!filesDirectory.exists()) {
            filesDirectory.mkdir();
        }
        for (int ctr = 0; ctr < files.size(); ctr++) {
            File file = files.get(ctr);
            String fileName = fileNames.get(ctr);
            String contentType = contentTypes.get(ctr);
```

```
                String tempFileName = UUID.randomUUID().toString() + ".tmp";
                File savedFile = null;
                do {
                    savedFile = new File(filesDirectory, tempFileName);
                } while (savedFile.exists());

                try {
                    FileUtils.moveFile(file, savedFile);
                } catch (Exception e) {
                    if (LOGGER.isErrorEnabled()) {
                        LOGGER.error(e.getMessage(), e);
                    }
                }

                FileManagement fileManagement = new FileManagement();
                fileManagement.setClinicalCase(clinicalCase);
                fileManagement.setName(fileName);
                fileManagement.setContentType(contentType);
                fileManagement.setLocation(savedFile.getAbsolutePath());
                fileManagement.setUploadedBy(uploadedBy);
                fileManagement.setDateUploaded(now);
                fileManagement.setCreatedOn(now);
                fileManagement.setIsValid(true);

                recordsToSave.add(fileManagement);
            }
            fileManagementRepository.save(recordsToSave);
        }

        @Override
        @Transactional
        public void deleteFiles(List<Long> fileIdsForDeletion) {
            for (Long fileId : fileIdsForDeletion) {
                FileManagement fileManagement = fileManagementRepository.findOne(fileId);
                File savedFile = new File(fileManagement.getLocation());
                savedFile.delete();
                fileManagementRepository.delete(fileManagement);
            }
        }

        @Override
        public FileManagement getFile(Long id) {
            FileManagement fileManagement = fileManagementRepository.findOne(id);
            File savedFile = new File(fileManagement.getLocation());
            fileManagement.setSavedFile(savedFile);
            return fileManagement;
        }

        @Override
        public List<FileManagement> getFiles(ClinicalCase clinicalCase) {
            List<FileManagement> fileManagementList =
    fileManagementRepository.findByClinicalCase(clinicalCase);
            for (FileManagement fileManagement : fileManagementList) {
                File savedFile = new File(fileManagement.getLocation());
                fileManagement.setSavedFile(savedFile);
            }
            return fileManagementList;
        }

    }
```

- **web-pcs/src/org.leaf.cms.action.service.impl/PhysicianServiceImpl.java**

```
    package org.leaf.cms.service.impl;

    import java.util.List;

    import javax.transaction.Transactional;

    import org.joda.time.DateTime;
    import org.leaf.cms.model.ClinicInfo;
    import org.leaf.cms.model.Physician;
    import org.leaf.cms.model.Specialty;
    import org.leaf.cms.model.repository.ClinicInfoRepository;
    import org.leaf.cms.model.repository.PhysicianRepository;
    import org.leaf.cms.service.PhysicianService;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.domain.Sort.Order;
import org.springframework.stereotype.Service;

@Service("physicianService")
public class PhysicianServiceImpl implements PhysicianService {

    @Autowired
    private PhysicianRepository physicianRepository;

    @Autowired
    private ClinicInfoRepository clinicInfoRepository;

    @Override
    public List<Physician> getPhysicians() {
        return physicianRepository.findAll();
    }

    @Override
    public List<Physician> getPhysicians(Specialty specialty) {
        return physicianRepository.physiciansOfSpecificSpecialty(specialty);
    }

    @Override
    public Page<Physician> getPhysicians(Specialty specialty, int pageNumber, int
itemsPerPage) {
        Order lastNameOrder = new Order(Direction.ASC, "lastName");
        Order firstNameOrder = new Order(Direction.ASC, "firstName");
        Sort sort = new Sort(lastNameOrder, firstNameOrder);
        PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
sort);
        return physicianRepository.physiciansOfSpecificSpecialty(specialty,
pageRequest);
    }

    @Override
    public Physician findPhysician(Long id) {
        return physicianRepository.findOne(id);
    }

    @Override
    public Long countByEmail(String email) {
        return physicianRepository.countByEmail(email);
    }

    @Override
    public Long countByMedLicenseId(Long medLicenseId){
    return physicianRepository.countByMedLicenseId(medLicenseId);
    }

    @Override
    public Physician findByEmail(String email) {
        return physicianRepository.findByEmail(email);
    }

    @Override
    public Physician save(Physician physician) {
        return physicianRepository.saveAndFlush(physician);
    }

    @Override
    public Physician activateAccount(Long id, String verificationCode) {
        Physician physician = physicianRepository.findByIdAndVerificationCode(id,
verificationCode);
        if (physician == null) {
            return null;
        }
        physician.setActivationDate(new DateTime());
        physician.setIsActivited(Boolean.TRUE);
        return physicianRepository.saveAndFlush(physician);
    }
```

```
    @Override
    public Page<Physician> findAllWithPaging(int pageNumber, int itemsPerPage) {
        PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
        return physicianRepository.findAll(pageRequest);
    }

    @Override
    @Transactional
    public void saveWithClinicInfo(Physician physician, List<ClinicInfo> clinics) {
            if((physician = physicianRepository.save(physician)) != null) {
                    if(physician.getClinics() != null &&
physician.getClinics().size() > 0){
                            clinicInfoRepository.delete(physician.getClinics());
                    }
                    physician.setClinics(clinics);
                    physicianRepository.saveAndFlush(physician);
            }
    }

    @Override
    public Page<Physician> findAllActivatedWithPaging(int pageNumber,
                    int itemsPerPage) {
            PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
            return physicianRepository.findByIsActivitedTrue(pageRequest);
    }

    @Override
    public void delete(Physician physician){
            physicianRepository.delete(physician);
    }

    @Override
    public Physician findByEmailAndMedLicenseId(String email, Long medLicenseId){
            return physicianRepository.findByEmailAndMedLicenseId(email,
medLicenseId);
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/ReferenceServiceImpl.java**

```
package org.leaf.cms.service.impl;

import org.leaf.cms.model.Reference;
import org.leaf.cms.model.repository.ReferenceRepository;
import org.leaf.cms.service.ReferenceService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

@Service("referenceService")
public class ReferenceServiceImpl implements ReferenceService{

    @Autowired
    public ReferenceRepository referenceRepository;

    @Override
    public Page<Reference> findAllWithPaging(int pageNumber, int itemsPerPage){
            PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
            return referenceRepository.findByIsValidTrue(pageRequest);
    }
}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/SMSEntryServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.Map;

import org.apache.velocity.app.VelocityEngine;
```

```
import org.joda.time.DateTime;
import org.leaf.cms.model.SMSEntry;
import org.leaf.cms.model.repository.SMSEntryRepository;
import org.leaf.cms.service.SMSEntryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.ui.velocity.VelocityEngineUtils;

@Service("smsEntryService")
public class SMSEntryServiceImpl implements SMSEntryService {

    @Autowired
    private SMSEntryRepository smsEntryRepository;

    @Autowired
    private VelocityEngine velocityEngine;

    @Override
    public Page<SMSEntry> findAllWithPaging(int pageNumber, int itemsPerPage) {
        PageRequest pageRequest = new PageRequest(pageNumber - 1, itemsPerPage,
Sort.Direction.DESC, "createdOn");
        return smsEntryRepository.findByIsSentFalse(pageRequest);
    }

    @Override
    public SMSEntry saveMessageToBeSent(String templateLocation, String recipient,
Map<String, Object> variables) {
        String message = VelocityEngineUtils.mergeTemplateIntoString(velocityEngine,
templateLocation, "UTF-8", variables);
        SMSEntry smsEntry = new SMSEntry();
        smsEntry.setIsSent(false);
        smsEntry.setMessage(message);
        smsEntry.setRecipient(recipient);
        smsEntry.setCreatedOn(DateTime.now());
        smsEntry.setIsValid(true);
        return smsEntryRepository.save(smsEntry);
    }

    @Override
    public SMSEntry save(SMSEntry entry){
            return smsEntryRepository.save(entry);
    }

    @Override
    public SMSEntry findById(Long id){
            return smsEntryRepository.findOne(id);
    }
}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/SpecialtyServiceImpl.java**

```
package org.leaf.cms.service.impl;

import java.util.List;

import org.leaf.cms.model.Specialty;
import org.leaf.cms.model.repository.SpecialtyRepository;
import org.leaf.cms.service.SpecialtyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service("specialtyService")
public class SpecialtyServiceImpl implements SpecialtyService {

    @Autowired
    private SpecialtyRepository specialtyRepository;

    @Override
    public List<Specialty> getSpecialty() {
        return specialtyRepository.findAll();
    }

    @Override
```

```
    public Specialty findSpecialty(String name) {
        return specialtyRepository.findByName(name);
    }

    @Override
    public Specialty findSpecialty(Long id) {
        return specialtyRepository.findOne(id);
    }

}
```

- **web-pcs/src/org.leaf.cms.action.service.impl/UserServiceImpl.java**

```
package org.leaf.cms.service.impl;

import org.leaf.cms.model.User;
import org.leaf.cms.model.repository.UserRepository;
import org.leaf.cms.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service("userService")
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public User findUser(String username, String password) {
        return userRepository.findByUsernameAndPassword(username, password);
    }

    @Override
    public User findUser(Long userId) {
        return userRepository.findOne(userId);
    }

    @Override
    public User saveUser(User user){
    return user = userRepository.saveAndFlush(user);
    }

}
```

- **web-pcs/src/org.leaf.cms.util/ClassHelper.java**

```
package org.leaf.cms.util;

import java.io.InputStream;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.net.URL;

public class ClassHelper
{
    public static <T> Class<T> getGenericParameter(Class<?> clazz, int
parameterIndex)
    {
            Class<T> parameter = null;
            Type superClass = clazz.getGenericSuperclass();

            while (!ParameterizedType.class.isInstance(superClass) && superClass !=
null)
            { superClass = (superClass instanceof Class<?>) ? ((Class<?>)
superClass).getGenericSuperclass() : null; }

            if (superClass instanceof ParameterizedType)
            {
                    ParameterizedType paramType = (ParameterizedType) superClass;
                    Type arguments[] = paramType.getActualTypeArguments();

                    if (arguments != null && arguments.length > parameterIndex) {
                            Type type = arguments[parameterIndex];

                            if (type instanceof Class)
```

```
                                {
                                        parameter = cast(type);
                                } } }

                if (parameter == null)
                {
                        System.out.println("The extended class " + clazz + " must set
the generic paramters.");
                }
                        return parameter;
        }

        @SuppressWarnings("unchecked")
        public static <T> T cast(Object object)
        {
                return (T) object;
        }

        public static URL getResource(String name) {
                ClassLoader classLoader = Thread.currentThread()
                                .getContextClassLoader();
                return classLoader.getResource(name);
        }

        public static InputStream getResourceAsStream(String name)
        {
                final ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
                return classLoader.getResourceAsStream(name);
        }
}
```

- **web-pcs/src/org.leaf.cms.util/EncryptionUtil.java**

```
package org.leaf.cms.util;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class EncryptionUtil {

        public static void main(String[] args) throws NoSuchAlgorithmException {
                System.out.println(encryptPassword("q"));
        }

        public static String encryptPassword(String passwordToHash) throws
NoSuchAlgorithmException{
                return get_SHA_256_SecurePassword(passwordToHash, "E@stL@ndP4Is");
        }

        public static String get_SHA_1_SecurePassword(String passwordToHash, String salt)
        {
            String generatedPassword = null;
            try {
                MessageDigest md = MessageDigest.getInstance("SHA-1");
                md.update(salt.getBytes());
                byte[] bytes = md.digest(passwordToHash.getBytes());
                StringBuilder sb = new StringBuilder();
                for(int i=0; i< bytes.length ;i++)
                {
                    sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
                }
                generatedPassword = sb.toString();
            }
            catch (NoSuchAlgorithmException e)
            {
                e.printStackTrace();
            }
            return generatedPassword;
        }

        public static String get_SHA_256_SecurePassword(String passwordToHash, String
salt)
        {
```

```
       String generatedPassword = null;
        try {
            // Create MessageDigest instance for MD5
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            //Add password bytes to digest
            md.update(salt.getBytes());
            //Get the hash's bytes
            byte[] bytes = md.digest(passwordToHash.getBytes());
            //This bytes[] has bytes in decimal format;
            //Convert it to hexadecimal format
            StringBuilder sb = new StringBuilder();
            for(int i=0; i< bytes.length ;i++)
            {
                sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
            }
            //Get complete hashed password in hex format
            generatedPassword = sb.toString();
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return generatedPassword;
    }

    public static String get_SHA_384_SecurePassword(String passwordToHash, String
salt)
    {
    String generatedPassword = null;
        try {
            // Create MessageDigest instance for MD5
            MessageDigest md = MessageDigest.getInstance("SHA-384");
            //Add password bytes to digest
            md.update(salt.getBytes());
            //Get the hash's bytes
            byte[] bytes = md.digest(passwordToHash.getBytes());
            //This bytes[] has bytes in decimal format;
            //Convert it to hexadecimal format
            StringBuilder sb = new StringBuilder();
            for(int i=0; i< bytes.length ;i++)
            {
                sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
            }
            //Get complete hashed password in hex format
            generatedPassword = sb.toString();
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return generatedPassword;
    }

    private static String get_SHA_512_SecurePassword(String passwordToHash, String
salt)
    {
        String generatedPassword = null;
        try {
            // Create MessageDigest instance for MD5
            MessageDigest md = MessageDigest.getInstance("SHA-512");
            //Add password bytes to digest
            md.update(salt.getBytes());
            //Get the hash's bytes
            byte[] bytes = md.digest(passwordToHash.getBytes());
            //This bytes[] has bytes in decimal format;
            //Convert it to hexadecimal format
            StringBuilder sb = new StringBuilder();
            for(int i=0; i< bytes.length ;i++)
            {
                sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
            }
            //Get complete hashed password in hex format
            generatedPassword = sb.toString();
        }
        catch (NoSuchAlgorithmException e) {
```

```
            e.printStackTrace();
        }
        return generatedPassword;
    }

    private static String getSalt() throws NoSuchAlgorithmException
    {
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        byte[] salt = new byte[16];
        sr.nextBytes(salt);
        return salt.toString();
    }
}
```

- **web-pcs/src/org.leaf.cms.util/FileUtil.java**

```
package org.leaf.cms.util;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.List;
import java.util.zip.ZipEntry;
import java.util.zip.ZipFile;
import java.util.zip.ZipOutputStream;

import javax.servlet.ServletContext;

import org.apache.struts2.ServletActionContext;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class FileUtil {
private static Logger logger = LoggerFactory.getLogger(FileUtil.class);

    public static void copyFile(File source, File destination) {
            FileInputStream input = null;
            FileOutputStream output = null;
            System.out.println("destination: " + destination.getAbsolutePath());
            try {
                    if(!destination.exists()) {
                            destination.createNewFile();
                    }
                    input = new FileInputStream(source);
                    output = new FileOutputStream(destination);

                    byte[] buf = new byte[1024];
                    int len;
                    while((len = input.read(buf)) > 0) {
                            output.write(buf, 0, len);
                    }

            }
            catch(Exception e) {
                    logger.error("failed to copy file " + source.getAbsolutePath() +
" to " +
                                    destination.getAbsolutePath(), e);
            }
            finally {
                    try {
                            if(output != null) {
                                    output.close();
                            }

                            if(input != null) {
                                    input.close();
                            }
```

```
                }
                catch(IOException ioe) {}
            }
    }

    public static boolean createDirectory(String path) {
            File file = new File(path.replaceAll(" ", "_"));
            return file.mkdir();
    }

    public static boolean deleteFile(String filepath) {
            logger.debug("DELETING: " + filepath);
            File file = new File(filepath);
            return file.delete();
    }

    public static String insertPostfix(String filename, String postfix) {
            String[] file = filename.split("[.]");
            StringBuffer sb = new StringBuffer();
            sb.append(file[0]);
            sb.append("-");
            sb.append(postfix);
            sb.append(".");
            sb.append(file[1]);
            return sb.toString();
    }

    public static String replaceExtension(String filename, String newExtension) {
            String[] filenameSeparated = filename.split("[.]");
            filenameSeparated[filenameSeparated.length-1] = newExtension;
            StringBuffer sb = new StringBuffer();
            for(String s: filenameSeparated) {
                    sb.append(s + ".");
            }
            sb.setLength(sb.length()-1);
            return sb.toString();
    }

    public static String getExtension(String filename)
    {
            String[] filenameSeparated = filename.split("[.]");
            logger.debug("get extension: " +
filenameSeparated[filenameSeparated.length-1]);
            if(filenameSeparated.length > 1)
            {
                    return filenameSeparated[filenameSeparated.length-1];
            }
            else
            {
                    return "";
            }
    }

    public static final String getFileName(String file)
    {
            String fileName = file.split("\\\\")[file.split("\\\\").length - 1];

            if(fileName.lastIndexOf(".") > -1)
            {
                    return fileName.substring(0, fileName.lastIndexOf("." +
getExtension(file)));
            }
            else
            {
                    return fileName;
            }
    }

    public static final String getFileNameWithExtension(String file)
    {
            return file.split("\\\\")[file.split("\\\\").length - 1];
    }

    protected static final File getFolder()
    {
            return getFolder(ServletActionContext.getServletContext());
```

```
    }

    protected static final File getFolder(ServletContext context)
    {
            return getFolder(context.getRealPath(""));
    }

    public static final File getFolder(String parentRealPath)
    {
            final File file = new File(parentRealPath, "file");

            return file;
    }

    public static final File getDirectory(String childDirectory)
    {
            File targetPath = new File(getFolder(), childDirectory);
            if(!targetPath.exists())
            {
                    System.out.println("TARGET PATH DOES NOT EXIST!");
                    targetPath.mkdirs();
            }
            return targetPath;
    }

    public static final File getDirectory(ServletContext context, String
childDirectory)
    {
            File targetPath = new File(getFolder(context), childDirectory);
            if(!targetPath.exists())
            {
                    System.out.println("TARGET PATH DOES NOT EXIST!");
                    targetPath.mkdirs();
            }
            return targetPath;
    }

    protected static final File getSystemBackupDirectory()
    {
            return new File("bak");
    }

    public static final File getSystemBackupDirectory(String childDirectory)
    {
            final File dir = new File(getSystemBackupDirectory(), childDirectory);
            if(!dir.exists())
            {
                    dir.mkdirs();
            }
            return dir;
    }

    public static final void zipFile(File file)
    {
            try {
                    ZipOutputStream zip = new ZipOutputStream(new
FileOutputStream(file.getAbsolutePath().replace(FileUtil.getExtension(file.getAbsolut
ePath()), "zip")));

                    zip.setLevel(9);
                    zip.setMethod(ZipOutputStream.DEFLATED);

                    ZipEntry entry = new
ZipEntry(FileUtil.getFileNameWithExtension(file.getAbsolutePath()));
                    entry.setTime(file.lastModified());

                    final int fileLength = (int) file.length();
                    final FileInputStream fis = new FileInputStream(file);
                    final byte [] wholeFile = new byte[fileLength];
                    fis.read(wholeFile, 0, fileLength);
                    fis.close();

                    zip.putNextEntry(entry);
                    zip.write(wholeFile, 0, fileLength);
                    zip.closeEntry();
                    zip.close();
```

```
            } catch(Exception e) {
                    e.printStackTrace();
            }
    }

    public static final File zipFile(List<File> files, File folder,String fileName)
    {
            File zipFile = new File(folder, fileName);
            ZipOutputStream zip = null;
            try {
                    zip = new ZipOutputStream(new FileOutputStream(zipFile));
                    zip.setLevel(9);
                    zip.setMethod(ZipOutputStream.DEFLATED);
                    for(File file : files){
                            ZipEntry entry = new
ZipEntry(FileUtil.getFileNameWithExtension(file.getName()));
                            entry.setTime(file.lastModified());

                            final int fileLength = (int) file.length();
                            final FileInputStream fis = new FileInputStream(file);
                            final byte [] wholeFile = new byte[fileLength];
                            fis.read(wholeFile, 0, fileLength);
                            fis.close();

                            zip.putNextEntry(entry);
                            zip.write(wholeFile, 0, fileLength);
                            zip.closeEntry();
                    }

                    if(files.size() < 1){
                            File messageFile = createMessageTextFile(folder);
                            ZipEntry entry = new
ZipEntry(FileUtil.getFileNameWithExtension(messageFile.getName()));
                            entry.setTime(messageFile.lastModified());

                            final int fileLength = (int) messageFile.length();
                            final FileInputStream fis = new
FileInputStream(messageFile);
                            final byte [] wholeFile = new byte[fileLength];
                            fis.read(wholeFile, 0, fileLength);
                            fis.close();

                            zip.putNextEntry(entry);
                            zip.write(wholeFile, 0, fileLength);
                            zip.closeEntry();
                    }
                    zip.close();
                    return zipFile;
            } catch(Exception e) {
                    e.printStackTrace();
                    return new File(folder, fileName);
            }
    }

    public static final File zipFile(List<File> files, List<String> fileNames, File
folder,String fileName)
    {
            File zipFile = new File(folder, fileName);
            ZipOutputStream zip = null;
            try {
                    zip = new ZipOutputStream(new FileOutputStream(zipFile));
                    zip.setLevel(9);
                    zip.setMethod(ZipOutputStream.DEFLATED);
                    int counter = 0;
                    for(File file : files){
                            ZipEntry entry = new
ZipEntry(FileUtil.getFileNameWithExtension(fileNames.get(counter)));
                            entry.setTime(file.lastModified());

                            final int fileLength = (int) file.length();
                            final FileInputStream fis = new FileInputStream(file);
                            final byte [] wholeFile = new byte[fileLength];
                            fis.read(wholeFile, 0, fileLength);
                            fis.close();

                            zip.putNextEntry(entry);
```

```
                                zip.write(wholeFile, 0, fileLength);
                                zip.closeEntry();
                                counter++;
                        }

                        if(files.size() < 1){
                                File messageFile = createMessageTextFile(folder);
                                ZipEntry entry = new
ZipEntry(FileUtil.getFileNameWithExtension(messageFile.getName()));
                                entry.setTime(messageFile.lastModified());

                                final int fileLength = (int) messageFile.length();
                                final FileInputStream fis = new
FileInputStream(messageFile);
                                final byte [] wholeFile = new byte[fileLength];
                                fis.read(wholeFile, 0, fileLength);
                                fis.close();

                                zip.putNextEntry(entry);
                                zip.write(wholeFile, 0, fileLength);
                                zip.closeEntry();
                        }
                        zip.close();
                        return zipFile;
                } catch(Exception e) {
                        e.printStackTrace();
                        return new File(folder, fileName);
                }
        }

        private static File createMessageTextFile(File folder){
                File file = new File(folder ,"message.txt");
                try{
                        file = new File(folder ,"message.txt");
                        String message = "No excel file was generated";
                        Writer output = new BufferedWriter(new FileWriter(file));
                        output.write(message);
                        output.close();

                }catch(Exception e){
                        e.printStackTrace();
                }
                return file;
        }

        public static String unZip (String inputZip, String destinationDirectory) throws
IOException
        {
                int BUFFER = 2048;
            List zipFiles = new ArrayList();
            File sourceZipFile = new File(inputZip);
            File unzipDestinationDirectory = new File(destinationDirectory);
            unzipDestinationDirectory.mkdir();

            ZipFile zipFile;
            // Open Zip file for reading
            zipFile = new ZipFile(sourceZipFile, ZipFile.OPEN_READ);

            // Create an enumeration of the entries in the zip file
            Enumeration zipFileEntries = zipFile.entries();

            // Process each entry
            while (zipFileEntries.hasMoreElements()) {
                // grab a zip file entry
                ZipEntry entry = (ZipEntry) zipFileEntries.nextElement();

                String currentEntry = entry.getName();

                File destFile = new File(unzipDestinationDirectory, currentEntry);
                destFile = new File(unzipDestinationDirectory, destFile.getName());

                if (currentEntry.endsWith(".zip")) {
                    zipFiles.add(destFile.getAbsolutePath());
                }

                // grab file's parent directory structure
```

```
                File destinationParent = destFile.getParentFile();

                // create the parent directory structure if needed
                destinationParent.mkdirs();

                try {
                    // extract file if not a directory
                    if (!entry.isDirectory()) {
                        BufferedInputStream is =
                                new BufferedInputStream(zipFile.getInputStream(entry));
                        int currentByte;
                        // establish buffer for writing file
                        byte data[] = new byte[BUFFER];

                        // write the current file to disk
                        FileOutputStream fos = new FileOutputStream(destFile);
                        BufferedOutputStream dest =
                                new BufferedOutputStream(fos, BUFFER);

                        // read and write until last byte is encountered
                        while ((currentByte = is.read(data, 0, BUFFER)) != -1) {
                            dest.write(data, 0, currentByte);
                        }
                        dest.flush();
                        dest.close();
                        is.close();
                    }
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }
            zipFile.close();

            for (Iterator iter = zipFiles.iterator(); iter.hasNext();) {
                String zipName = (String)iter.next();
                unZip(
                    zipName,
                    destinationDirectory +
                        File.separatorChar +
                        zipName.substring(0,zipName.lastIndexOf(".zip"))
                );
            }
            return "success";

        }
    }
```

- **web-pcs/src/org.leaf.cms.util/HMAC.java**

```
    package org.leaf.cms.util;

    import java.io.UnsupportedEncodingException;
    import java.util.Arrays;
    import java.util.Iterator;
    import java.util.List;
    import java.util.Map;
    import java.util.TreeMap;

    import javax.crypto.Mac;
    import javax.crypto.spec.SecretKeySpec;

    import org.apache.tomcat.util.codec.binary.Base64;

    public class HMAC
    {
        private static final String UTF8_CHARSET = "UTF-8";
        private static final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

        public static Mac getMac(String secretKey)
        {
                try {
                        SecretKeySpec secretKeySpec = new
    SecretKeySpec(secretKey.getBytes(UTF8_CHARSET), HMAC_SHA256_ALGORITHM);
            Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
            mac.init(secretKeySpec);
            return mac;
```

```
            } catch(Exception e) {
                    return null;
            }
    }

    public static String sign(String secretKey, String stringToSign)
    {
        Mac mac = getMac(secretKey);
        if(mac == null)
        {
              return null;
        }
        else
        {
              String signature = null;
          byte[] data;
          byte[] rawHmac;
          try {
              data = stringToSign.getBytes(UTF8_CHARSET);
            rawHmac = mac.doFinal(data);
            Base64 encoder = new Base64();
            signature = new String(encoder.encode(rawHmac));
          } catch(UnsupportedEncodingException e) {
              throw new RuntimeException(UTF8_CHARSET + " is unsupported!", e);
          }
          return signature;
        }
    }

   public static String sign(String secretKey, Map<String, String[]> parameterMap)
    {
      return sign(secretKey, arrangeParameters(parameterMap));
    }

   public static final String arrangeParameters(Map<String, String[]> parameters)
    {
      parameters = new TreeMap<String, String[]>(parameters);

      final StringBuffer arrangedParameters = new StringBuffer();
              final Iterator<Map.Entry<String, String[]>> iterator =
parameters.entrySet().iterator();

              while(iterator.hasNext())
              {
                      final Map.Entry<String, String[]> entry = iterator.next();
                      final List<String> valueList = Arrays.asList(entry.getValue());
                      final Iterator<String> valueListIterator = valueList.iterator();

                      while(valueListIterator.hasNext())
                      {

      arrangedParameters.append(entry.getKey()).append("=").append(valueListIterator.ne
xt());
                              if(valueListIterator.hasNext())
                              {
                                      arrangedParameters.append("&");
                              }
                      }

                      if(iterator.hasNext()) arrangedParameters.append("&");
              }

              return arrangedParameters.toString();
    }

    public static void main(String ... args)
    {
      System.out.println("hmac: " + sign("t1c]c3Tj0]\\[]Es",
"companyId=3&userId=582"));
    }
}
```

- **web-pcs/src/org.leaf.cms.util/ImageUtil.java**

```
package org.leaf.cms.util;
```

```
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.imageio.ImageIO;

import org.apache.commons.io.FileUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ImageUtil {
private static Logger _logger = LoggerFactory.getLogger(ImageUtil.class);

    public static File generateJpegImage(String input, String output) {

            output = FileUtil.replaceExtension(output, "jpg");

            try {
                    File newFile = new File(output);
                    File old = new File(input);

                    FileUtil.copyFile(old, newFile);
                    FileUtil.deleteFile(input);

                    return newFile;
            }
            catch (Exception e)
            {
                    _logger.error("Unable to generate jpeg image. ", e);
            }
            return null;
    }

    /**
     * Reads an image in a file and resizes the image.
     * Image will only be created if it's larger than the given <code>maxDim</code>
     *
     * @param orig The name of image file.
     * @param thumb The name of thumbnail file.
     * @param maxDim The width and height of the thumbnail must be maxDim pixels or
less.
     *
     * @author Jay Q.
     */
    public static void scaleImage(String origImageFile, String thumb,
                    float maxDim) throws FileNotFoundException,      IOException {

            BufferedImage inImage = ImageIO.read(new File(origImageFile));
            int width = inImage.getWidth(null);
            int height = inImage.getHeight(null);

            if (width <= maxDim) {

                    if(_logger.isDebugEnabled()) _logger.debug("image is smaller
than or equal to desired size.");

                    FileUtils.copyFile(new File(origImageFile), new File(thumb));
                    return;
            }

            // Determine the scale.
            double scale = (double) maxDim / (double) height;

            if (width > height) {
                    scale = (double) maxDim / (double) width;
            }

            // Determine size of new image.
            // One of them should equal maxDim.
            int scaledW = width;
```

```
            int scaledH = height;

            if (width >= maxDim) {
                    scaledW = (int) (scale * width);
                    scaledH = (int) (scale * height);
            }

            // Create an image buffer in which to paint on.
            BufferedImage outImage = null;
            Image src = null;
            try {
                    outImage = new BufferedImage(scaledW, scaledH,
BufferedImage.TYPE_INT_RGB);
                    src = inImage.getScaledInstance(scaledW, scaledH,
Image.SCALE_SMOOTH);
            } catch (Exception e) {
                    System.out.println("height: " + height);
                    System.out.println("width: " + width);
                     e.printStackTrace();
                    return;
            }

            // Set the scale.
            AffineTransform tx = new AffineTransform();

            // Paint image.
            Graphics2D g2d = outImage.createGraphics();
            g2d.drawImage(src, tx, null);
            g2d.dispose();

            // JPEG-encode the image and write to file.
            OutputStream os = new FileOutputStream(thumb);
            ImageIO.write(outImage, "jpeg", os);

            os.close();
            inImage.flush();
            inImage = null;
            outImage = null;
            g2d = null;
    }
}
```

- **web-pcs/src/org.leaf.cms.utility/HttpUtility.java**

```
package org.leaf.cms.utility;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.lang3.StringUtils;

public class HttpUtility
{
    /**
     * Get the uri from the request. This method removes the context path.
     *
     * @param request the request
     * @return the uri
     */
    public static String getServletPath(HttpServletRequest request)
    {
            String servletPath = request.getServletPath();

            if (StringUtils.isNotEmpty(servletPath))
            {
                    return servletPath;
            }

            final String contextPath = request.getContextPath();
            final String requestUri = request.getRequestURI();

            int startIndex = contextPath.equals("") ? 0 : contextPath.length();

            return requestUri.substring(startIndex);
    }

  /**
```

```
     * Retrieves the current request servlet path. Deals with differences between
servlet specs (2.2 vs 2.3+)
     *
     * @param request the request
     * @return the servlet path
     */
    public static String getServletPathWithPathInfo(HttpServletRequest request)
    {
            String servletPath = request.getServletPath();

            if (StringUtils.isNotEmpty(servletPath))
            {
                    return servletPath;
            }

            final String contextPath = request.getContextPath();
            final String requestUri = request.getRequestURI();
            final String pathInfo = request.getPathInfo();

            int startIndex = contextPath.equals("") ? 0 : contextPath.length();
            int endIndex = pathInfo == null ? requestUri.length() :
requestUri.lastIndexOf(pathInfo);

            if (startIndex > endIndex)
            {
                    endIndex = startIndex;
            }

            return requestUri.substring(startIndex, endIndex);
    }

}
```

- **web-pcs/src/org.leaf.cms.ws.rest.client/MemberClient.java**

```
package org.leaf.cms.ws.rest.client;

import java.io.File;

import org.leaf.cms.action.interfaces.UriAware;

public interface MemberClient extends UriAware {

    public String save(String cardNo, String referenceNumber, File memberImg, File
memberSign, File memberThumb);

}
```

- **web-pcs/src/org.leaf.cms.ws.rest.client.impl/AbstractClient.java**

```
package org.leaf.cms.ws.rest.client.impl;

import java.net.URI;
import java.util.Arrays;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation.Builder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.UriBuilder;

import org.glassfish.jersey.client.ClientConfig;
import org.glassfish.jersey.media.multipart.MultiPartFeature;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;

public abstract class AbstractClient {

    protected static final DateTimeFormatter FORMATTER = DateTimeFormat
                    .forPattern("MMddYYYYHHmmss");

    private String headerNamePublicKey;
```

```
    private String headerNameSharedSecret;

    private String parameterNameDateRequest;

    private String privateKey;

    private String publicKey;

    private String uri;

    protected final URI getBaseUri() {
            return UriBuilder.fromUri(uri).build();
    }

    protected final WebTarget getService() {
            /*final ClientConfig clientConfig = new DefaultClientConfig();
            final Client client = Client.create(clientConfig);*/
            final ClientConfig clientConfig = new ClientConfig();
            final Client client =
ClientBuilder.newClient(clientConfig).register(MultiPartFeature.class);

            return client.target(getBaseUri());
    }

    protected final WebTarget appendParameters(WebTarget service,
                    Map<String, String[]> parameterMap) {
            if (parameterMap != null) {
                    for (Map.Entry<String, String[]> entry :
parameterMap.entrySet()) {
                            final List<String> valueList =
Arrays.asList(entry.getValue());
                            final Iterator<String> iterator = valueList.iterator();

                            while (iterator.hasNext()) {
                                    service = service.queryParam(entry.getKey(),
iterator
                                                    .next());
                            }
                    }
            }
            return service;
    }

    protected final Builder appendHeaderPublicKey(
                    WebTarget service) {
            return service.request().header(headerNamePublicKey, publicKey);
    }

    protected final Form createForm(Map<String, String[]> parameterMap) {
            final Form form = new Form();

            if (parameterMap != null) {
                    for (Map.Entry<String, String[]> entry :
parameterMap.entrySet()) {
                            final List<String> valueList =
Arrays.asList(entry.getValue());
                            final Iterator<String> iterator = valueList.iterator();

                            while (iterator.hasNext()) {
                                    form.param(entry.getKey(), iterator.next());
                            }
                    }
            }

            return form;
    }

    public String getUri() {
            return uri;
    }

    public void setUri(String uri) {
            this.uri = uri;
    }
```

```
    public String getHeaderNamePublicKey() {
            return headerNamePublicKey;
    }

    public void setHeaderNamePublicKey(String headerNamePublicKey) {
            this.headerNamePublicKey = headerNamePublicKey;
    }

    public String getHeaderNameSharedSecret() {
            return headerNameSharedSecret;
    }

    public void setHeaderNameSharedSecret(String headerNameSharedSecret) {
            this.headerNameSharedSecret = headerNameSharedSecret;
    }

    public String getParameterNameDateRequest() {
            return parameterNameDateRequest;
    }

    public void setParameterNameDateRequest(String parameterNameDateRequest) {
            this.parameterNameDateRequest = parameterNameDateRequest;
    }

    public String getPrivateKey() {
            return privateKey;
    }
    public void setPrivateKey(String privateKey) {
            this.privateKey = privateKey;
    }

    public String getPublicKey() {
            return publicKey;
    }

    public void setPublicKey(String publicKey) {
            this.publicKey = publicKey;
    }

}
```

- **web-pcs/src/org.leaf.cms.ws.rest.client.impl/MemberClientImpl.java**

```
package org.leaf.cms.ws.rest.client.impl;

import java.io.File;
import java.util.Map;
import java.util.TreeMap;

import javax.servlet.http.HttpServletResponse;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation.Builder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import org.glassfish.jersey.media.multipart.FormDataMultiPart;
import org.glassfish.jersey.media.multipart.file.FileDataBodyPart;
import org.joda.time.DateTime;
import org.leaf.cms.ws.rest.client.MemberClient;

public class MemberClientImpl extends AbstractClient implements MemberClient
{

    @Override
    public String save(String cardNo, String referenceNumber, File memberImg, File
memberSign, File memberThumb) {
            try {
                    final Map<String, String[]> parameterMap = new TreeMap<String,
String[]>();
                    parameterMap.put(getParameterNameDateRequest(),
                                    new String[] { FORMATTER.print(new DateTime())
});

                    FormDataMultiPart multiPart = new FormDataMultiPart();
                multiPart.setMediaType(MediaType.MULTIPART_FORM_DATA_TYPE);
```

```
                    FileDataBodyPart fileDataBodyPart = new
FileDataBodyPart("memberImg", memberImg, MediaType.APPLICATION_OCTET_STREAM_TYPE);
                    FileDataBodyPart fileDataBodyPart2 = new
FileDataBodyPart("signImg", memberSign, MediaType.APPLICATION_OCTET_STREAM_TYPE);
                    FileDataBodyPart fileDateBodyPart3 = new
FileDataBodyPart("thumbImg", memberThumb, MediaType.APPLICATION_OCTET_STREAM_TYPE);

                    multiPart.field("cardNumber", cardNo);
                    multiPart.field("referenceNumber", referenceNumber);
                    multiPart.bodyPart(fileDataBodyPart);
                    multiPart.bodyPart(fileDataBodyPart2);
                    multiPart.bodyPart(fileDateBodyPart3);

                    final String mediaType = MediaType.APPLICATION_XML;

                    WebTarget webTarget =
getService().path("rest").path("member").path("pushnotification");
                    webTarget = appendParameters(webTarget, parameterMap);
                    Builder builder = appendHeaderPublicKey(webTarget);
                    final Response clientResponse =
builder.accept(mediaType).post(Entity.entity(multiPart, multiPart.getMediaType()));

                    if (clientResponse.getStatus() == HttpServletResponse.SC_OK
                                    || clientResponse.getStatus() ==
HttpServletResponse.SC_NO_CONTENT) {
                            return clientResponse.readEntity(String.class);
                    } else {
                            return "FAILED";
                    }
            } catch (Exception e) {
                    e.printStackTrace();
                    return "FAILED";
            } } }
```

- **web-pcs/src/org.leaf.filter/StyleFolderFilter.java**

```
package org.leaf.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import org.leaf.cms.utility.HttpUtility;

public class StyleFolderFilter implements Filter
{
private ServletContext context;

    private String toPath;

    private String capturePath;

    public void init(FilterConfig config) throws ServletException
    {
            context = config.getServletContext();

            toPath = config.getInitParameter("toPath");
            capturePath = config.getInitParameter("capturePath");

    }

    public void destroy()
    {
            toPath = null;
            capturePath = null;
            context = null;
    }
```

```
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
                    throws IOException, ServletException
    {
            final HttpServletRequest req = (HttpServletRequest) request;

            final String requestURI = HttpUtility.getServletPath(req);

            final String childRequestURI = requestURI.replaceFirst(capturePath,
"$1");
            final String finalPath = toPath.replace("${1}", childRequestURI);


            context.getRequestDispatcher(finalPath).forward(request, response);
    }
}
```

- **web-pcs/src/org.leaf.initializer/ApplicationInitializer.java**

```
package org.leaf.initializer;

import java.util.List;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;
import org.joda.time.DateTime;
import org.leaf.cms.model.Appointment;
import org.leaf.cms.service.AppointmentService;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;

public class ApplicationInitializer implements ServletContextListener {

    private Logger _logger = Logger.getLogger(ApplicationInitializer.class);

    @Override
    public void contextDestroyed(ServletContextEvent event) {

    }

    @Override
    public void contextInitialized(ServletContextEvent event) {
        _logger.info("Executing Initialization method in ApplicationInitializer");
        final WebApplicationContext context =
WebApplicationContextUtils.getWebApplicationContext(event.getServletContext());
        terminatePastAppointments(context);
        _logger.info("Successfully initialized application");
    }

    private void terminatePastAppointments(WebApplicationContext context) {
        _logger.info("Terminating past appointments...");
        AppointmentService appointmentService = context.getBean("appointmentService",
AppointmentService.class);
        DateTime now = DateTime.now();
        List<Appointment> appointments =
appointmentService.getActiveAppointmentsBefore(now);
        for (Appointment appointment : appointments) {
            appointment.setIsSeenRequestingPhysician(true);
            appointment.setIsSeenConsultant(true);
            appointmentService.terminateAppointment(appointment);
        }
    }

}
```

- **web-pcs/resources/live/conf/application.properties**

```
default.mssql.db.connection.string = jdbc:sqlserver://TIM\\SQLEXPRESS;"
default.mssql.db.username = sa
default.mssql.db.password = root
default.rest.central = http://pnpverification-leafinc.rhcloud.com

clinicalcase.files.location = ../clinical-case-files
images.files.location = ../images
```

```
init.discussion.thread.sms = Successfully created new SMS discussion thread: <title>
```

- **web-pcs/resources/live/conf/dbconn.properties**

```
# dbconn.properties contains properties to be used in spring for database connection.

jndi.datasource.location=java:comp/env/jdbc/datasource
```

- **web-pcs/resources/live/conf/hibernate.properties**

```
#hibernate.session.context.class=thread
hibernate.autocommit=false
hibernate.hbm2ddl.auto=update
hibernate.show_sql=true
#hibernate.transaction.factory=org.hibernate.transaction.JDBCTransactionFactory

# hibernate second level cache
hibernate.cache.provider=org.hibernate.cache.NoCacheProvider
hibernate.cache.use_second_level_cache=false
hibernate.cache.use_structured_entries=false

## Hibernate Dialects
#for sql server
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.sql.dialect=org.hibernate.dialect.SQLServerDialect
#for mysql
#hibernate.dialect=org.hibernate.dialect.MySQLDialect

quartz.memberscheduler.cron=0 * * * * ?
```

- **web-pcs/resources/live/conf/smtp.properties**

```
#GMAIL SMTP
smtp.host=smtp.gmail.com
smtp.port=465
smtp.username=webpcs.noreply@gmail.com
smtp.password=physicians231
smtp.auth=true
smtp.socketfactoryport=25

smtp.ssl.enable=true
```

- **web-pcs/web/WEB-INF/velocity/accountactivation_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
            Good Day!<br><h5></h5>
            <p>
            Welcome to WebPCS! The administrator approved your account request.
Please follow this <a href="$url">link</a> to log in to your account.
            </p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/cancelappointment_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
        <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
        <p>The following appointment has been cancelled:</p>
        <h5></h5>
        <table>
            <tr>
                <td>Consultation Date:</td>
                <td>$consultationDateString</td>
```

```
            </tr>
            <tr>
                <td>Primary Care Physician:</td>
                <td>$requestingPhysicianName</td>
            </tr>
            <tr>
                <td>Consultant:</td>
                <td>$consultantName</td>
            </tr>
        </table>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/forgotpassword_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
            <p>
                    Your new password is <b>$password</b>. We recommend that you
change your password once you have logged in.
            </p>
    </div>
</body>

</html>
```

- **web-pcs/web/WEB-INF/velocity/newappointment_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
            <p>A new appointment has been scheduled for you with the following
details:</p>
                    <h5></h5>

            <table>
                    <tr>
                            <td>Consultation Date:</td>
                            <td>$consultationDateString</td>
                    </tr>
                    <tr>
                            <td>Primary Care Physician:</td>
                            <td>$requestingPhysicianName</td>
                    </tr>
                    <tr>
                            <td>Consultant:</td>
                            <td>$consultantName</td>
                    </tr>
            </table>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/newreply_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
            <p>
```

```
                    Physician $subscriber replied to your thread entitled "$title".
</p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/registration_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
            <p>
            Welcome to WEB PCS! Please wait for the administrator to validate your
account. You will be receiving an activation e-mail soon.
            </p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/registrationfailed_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
            <p>
            The administrator declined your request to register to WebPCS. Kindly
contact us if you want to know about the reason. </p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/rescheduleappointment_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
        <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
        <p>Your appointment has been rescheduled:</p>
        <h5></h5>
        <table>
            <tr>
                <td>Consultation Date:</td>
                <td>$consultationDateString</td>
            </tr>
            <tr>
                <td>Primary Care Physician:</td>
                <td>$requestingPhysicianName</td>
            </tr>
            <tr>
                <td>Consultant:</td>
                <td>$consultantName</td>
            </tr>
        </table>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/subscription_html.vm**

```
<html>
<head>
</head>
<body>
```

```
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
            <p>
                    Physician $subscriber followed your thread entitled "$title".
            </p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/subscriptionupdate_html.vm**

```
<html>
<head>
</head>
<body>
    <div>
            <h3>Hi $name,</h3>
            <h5></h5>
        Good Day!<br><h5></h5>
            <p>
                    New comment added to <a href="$url"> $title</a>.
            </p>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/cancelappointment_sms.vm**

```
Hi, $name. Your appointment with $otherName on $consultationDateString has been
cancelled.
```

- **web-pcs/web/WEB-INF/velocity/newappointment_sms.vm**

```
Hi, $name. You have a new appointment with $otherName on $consultationDateString.
```

- **web-pcs/web/WEB-INF/velocity/notifyappointment_sms.vm**

```
Hi, $name. This is a reminder of your appointment with $otherName on
$consultationDateString.
```

- **web-pcs/web/WEB-INF/velocity/rescheduleappointment_sms.vm**

```
Hi, $name. Your appointment with $otherName has been rescheduled to
$consultationDateString.
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/account/account-form.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="account"/>
<%@include file="../includes/nav.jsp" %>
<div class="container-fluid">
    <h5></h5>
    <div class="container">
    <h3><b>Update your Account</b></h3>
    <%@include file="../includes/errors.jsp" %>
    <h1></h1>
    <form action="account-saveAccountConfiguration.do" method="post">
    <div class="panel panel-default">
    <div class="panel-heading"> EMAIL </div>
    <div class="panel-body">
            <input id="sendWhenBookedInput" type="checkbox" value="1"
name="sendWhenBooked" ${user.sendWhenBooked ? 'checked' : '' }/>
            <label for="sendWhenBookedInput">An appointment was booked</label><br/>
            <input id="sendWhenCancelledInput" type="checkbox" value="1"
name="sendWhenCancelled" ${user.sendWhenCancelled ? 'checked' : '' }/>
            <label for="sendWhenCancelledInput">An appointment was
cancelled</label><br/>
            <input id="sendWhenRescheduledInput" type="checkbox" value="1"
name="sendWhenRescheduled" ${user.sendWhenRescheduled ? 'checked' : '' }/>
            <label for="sendWhenRescheduledInput">An appointment was
rescheduled</label><br/>
    </div>
    </div>
        <div class="panel panel-default">
```

```
    <div class="panel-heading"> SMS </div>
    <div class="panel-body">
        <input id="triggerSmsReminderInput" type="checkbox" value="1"
name="triggerSmsReminder" ${user.triggerSmsReminder ? 'checked' : '' }/>
        <label for="triggerSmsReminderInput">Set SMS Reminder after booking an
appointment</label>
    </div>
    </div>
    <div class="panel panel-default">
    <div class="panel-heading"> FORUM <div>
    <div class="panel-body">
            <input id="sendOnForumReplyInput" type="checkbox" value="1"
name="sendOnForumReply" ${user.sendOnForumReply ? 'checked' : '' }/>
            <label for="sendOnForumReplyInput">Someone replied to my
forum</label><br/>
            <input id="sendOnForumSubscribedInput" type="checkbox" value="1"
name="sendOnForumSubscribed" ${user.sendOnForumSubscribed ? 'checked' : '' }/>
            <label for="sendOnForumSubscribedInput">Someone subscribed to my
forum</label>
    </div>
    </div>

    <div class="panel panel-default">
    <div class="panel-heading"> SUBSCRIPTION <div>
    <div class="panel-body">
            <input id="sendSubscriptionUpdatesInput" type="checkbox" value="1"
name="sendSubscriptionUpdates" ${user.sendSubscriptionUpdates ? 'checked' : '' }/>
            <label for="sendSubscriptionUpdatesInput">Updates about the forum I
follow</label><br/>
            <select name="subscriptionDigestType" style="display: none;">
                    <option value="DAILY">Send as daily digest</option>
                    <option value="WEEKLY">Send as weekly digest</option>
            </select>
    </div>
    </div>
    <div style="float: right;">
            <input class="btn btn-sm btn-primary" type="submit" value="Save Changes">
    </div>
    </form>
    </div>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/appointment-notify-select.jsp**

```
<div class="panel-body">
    Select when to be notified:
    <form id="notificationSelectionForm" action="appointments-notify-save.do">
        <br><input id="appointmentNotifySelectId" name="appointmentId" type="hidden"
value="${appointmentId}">
            <select class="form-control" id="notificationSmsTime"
name="notificationSmsTime" required>
                    <option value="-1">None</option>
                    <option value="0">At time of event</option>
                    <option value="5">5 minutes before</option>
                    <option value="15">15 minutes before</option>
                    <option value="30">30 minutes before</option>
                    <option value="60">1 hour before</option>
                    <option value="120">2 hours before</option>
                    <option value="1440">1 day before</option>
                    <option value="2880">2 days before</option>
                    <option value="10080">1 week before</option>
            </select>
    </form>
</div>
<div class="panel-body">
    <input class="btn btn-sm btn-primary" type="button" value="Save"
onclick="saveNotification()">
    <input class="btn btn-sm btn-primary" type="button" value="Cancel"
onclick="cancelNotification()">
</div>
<script type="text/javascript">
    function saveNotification() {
            var formData = $("#notificationSelectionForm").serialize();
            $.facebox(function() {
                    $.ajax({
```

```
                                url : 'appointments-notify-save.do',
                                type : 'post',
                                data : formData,
                                cache : false,
                        }).done(function(data) {
                                cancelNotification();
                        });
                });
        }
        function cancelNotification() {
                $.facebox.close();
        }
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/appointment-notify.jsp**

```
<div class="panel-body">Do you want to be notified via SMS before the said
appointment?</div>
<div class="panel-body">
    <input class="btn btn-sm btn-primary" type="button" value="Yes"
onclick="selectNotification()">
    <input class="btn btn-sm btn-primary" type="button" value="No"
onclick="cancelNotification()">
</div>
<script type="text/javascript">
function selectNotification() {
    var appointmentId = '${param.appointmentId}';
    $.facebox(function() {
        $.ajax({
            url: 'appointments-notify-select.do',
            type: 'get',
            data: {'appointmentId' : appointmentId},
            cache: false,
        })
        .done(function(data){
            $.facebox(data);
        });
    });
}
function cancelNotification() {
    $.facebox.close();
}

</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/chat-room-clinicalcase-details.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<input id="clinicalCaseIdHidden" type="hidden" name="clinicalCaseId"
value="${clinicalCaseId}">
<div id="appointmentDetailsDiv">
    <table class="table">
        <tr>
            <td>Specialty:</td>
            <td>${specialtyDescription}</td>
        </tr>
        <tr>
            <td>Consultation Date:</td>
            <td>${consultationDateString}</td>
        </tr>
        <tr>
            <td>Primary Care Physician:</td>
            <td>${requestingPhysicianName}</td>
        </tr>
        <tr>
            <td>Consultant:</td>
            <td>${consultantName}</td>
        </tr>
    </table>
</div>

<h5>Clinical Question</h5>
<textarea id="questionTextArea" style="width: 865px; height: 89px;"
name="question">${question}</textarea>
```

```
<h5>Laboratory Studies</h5>
<textarea id="patientLabExamTextArea" style="width: 865px; height: 89px;"
name="patientLabExam">${patientLabExam}</textarea>

<h5>Diagnostic Findings</h5>
<textarea id="patientDiagnosisTextArea" style="width: 865px; height: 89px;"
name="patientDiagnosis">${patientDiagnosis}</textarea>

<h5>Remarks</h5>
<textarea id="remarksTextArea" style="width: 865px; height: 89px;"
name="remarks">${remarks}</textarea>

<h5>Status</h5>
<textarea id="statusTextArea" style="width: 865px; height: 29px;" name="status"
disabled="disabled">${status}</textarea>

<h5>Files</h5>
<div id="filesDiv">
    <s:iterator value="clinicalCaseFiles" var="clinicalCaseFile" status="fileStatus">
        <div style="margin-bottom: 5px;">
            <input id="clinicalCaseFileName<s:property
value="#clinicalCaseFile.id"/>" type="text" value=" <s:property
value="#clinicalCaseFile.name"/>" readonly="readonly" style="min-width:
260px;vertical-align: middle;">
            <input id="fileIdForDeletion<s:property value="#clinicalCaseFile.id"/>"
type="hidden" name="fileIdsForDeletion[<s:property value="#fileStatus.index"/>]">
            <input class="btn btn-sm btn-primary" type="button" value="Download"
onclick="downloadFile(<s:property value="#clinicalCaseFile.id"/>);">
            <input class="btn btn-sm btn-primary" id="deleteFileButton" type="button"
value="Delete" onclick="markFileForDeletion(this, <s:property
value="#clinicalCaseFile.id"/>)">
        </div>
    </s:iterator>
    <div id="fileUploadDiv" style="margin-bottom: 5px;">
        <input class="btn btn-sm btn-primary" style="display: inline-block;"
type="file" name="upload" onchange="addNewInputFile(this);">
    </div>
</div>

<div id="editDiv" style="margin-bottom: 5px;">
    <input class="btn btn-sm btn-primary" type="button" value="Edit Details"
onclick="editDetails()">
</div>
<div id="saveDiv" style="margin-bottom: 5px;">
    <input class="cbox-button btn btn-sm btn-primary" type="button" value="Cancel"
onclick="cancelUpdate()">
    <input class="cbox-button btn btn-sm btn-primary" type="button" value="Save"
onclick="updateDetails()">
</div>

<script type="text/javascript">
    $(document).ready(function() {
        var statusValue = $('[name=status]').val();
        if(statusValue != 'Active' && statusValue != 'Ongoing') {
            $("#appointmentDetailsDiv").hide();
        }
        if(statusValue == 'Closed') {
            $("#editDiv").hide();
        }
        $("#questionTextArea").attr("disabled", "disabled");
        $("#patientLabExamTextArea").attr("disabled", "disabled");
        $("#patientDiagnosisTextArea").attr("disabled", "disabled");
        $("#remarksTextArea").attr("disabled", "disabled");
        $("[id^=deleteFileButton]").hide();
        $("#fileUploadDiv").hide();
        $("#saveDiv").hide();
    });
    function downloadFile(fileId) {
        window.open("/consultations-clinicalcase-file-download.do?fileId=" + fileId,
"_blank", null);
    }
    function editDetails() {
        $("#questionTextArea").removeAttr("disabled");
        $("#patientLabExamTextArea").removeAttr("disabled");
        $("#patientDiagnosisTextArea").removeAttr("disabled");
        $("#remarksTextArea").removeAttr("disabled");
```

```
        $("[id^=deleteFileButton]").show();
        $("#fileUploadDiv").show();
        $("#editDiv").hide();
        $("#saveDiv").show();
    }
    function markFileForDeletion(button, fileId) {
        var fileNameDivId = 'clinicalCaseFileName' + fileId;
        document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"line-through");
        button.value = 'Undo';
        button.onclick = function() {
            unmarkFileForDeletion(this, fileId);
        }
        var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
        document.getElementById(fileIdForDeletionId).value = fileId;
    }
    function unmarkFileForDeletion(button, fileId) {
        var fileNameDivId = 'clinicalCaseFileName' + fileId;
        document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"none");
        button.value = 'Delete';
        button.onclick = function() {
            markFileForDeletion(this, fileId);
        }
        var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
        document.getElementById(fileIdForDeletionId).value = null;
    }
    function addNewInputFile(element) {
            element.onchange = null;
        var deleteButton = document.createElement('input');
        deleteButton.className = 'btn btn-sm btn-primary';
        deleteButton.type = 'button';
        deleteButton.value = 'Delete';
        deleteButton.onclick = function() {
            var parentDiv = this.parentNode;
            document.getElementById('filesDiv').removeChild(parentDiv);
        }
        element.parentNode.appendChild(deleteButton);
        var newUploadFileDiv = document.createElement('div');
        var inputFile = document.createElement('input');
        inputFile.className = 'btn btn-sm btn-primary';
        inputFile.setAttribute('style', 'display: inline-block;margin-right: 4px;');
        inputFile.type = 'file';
        inputFile.name = 'upload';
        inputFile.onchange = function() {
            addNewInputFile(this);
        }
        newUploadFileDiv.appendChild(inputFile);
        newUploadFileDiv.setAttribute('style', 'margin-bottom: 5px;');
        document.getElementById('filesDiv').appendChild(newUploadFileDiv);
    }
    function updateDetails() {
        var formData = new FormData();
        formData.append('clinicalCaseId', $("#clinicalCaseIdHidden").val());
        formData.append('question', $("#questionTextArea").val());
        formData.append('patientDiagnosis', $("#patientDiagnosisTextArea").val());
        formData.append('patientLabExam', $("#patientLabExamTextArea").val());
        formData.append('remarks', $("#remarksTextArea").val());
        formData.append('fileIdsForDeletion',
$("[name^=fileIdsForDeletion]").get(0));
        formData.append('upload', $("[name^=upload]").get(0).files[0]);
        $.facebox(function() {
            $.ajax({
                url: 'appointments-room-clinicalcase-details-update.do',
                type: 'post',
                data: formData,
                cache: false,
                contentType: false,
                processData: false
            })
            .done(function(data){
                $.facebox(data);
            });
        });
    }
    function cancelUpdate() {
```

```
        $("#questionTextArea").prop('disabled', true);
        $("#patientLabExamTextArea").prop('disabled', true);
        $("#patientDiagnosisTextArea").prop('disabled', true);
        $("#remarksTextArea").prop('disabled', true);
        $("[id^=deleteFileButton]").hide();
        $("#fileUploadDiv").hide();
        $("#editDiv").show();
        $("#saveDiv").hide();
    }

</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/chat-room-end.jsp**

```
<input type="hidden" id="idHistory" value="${chatRoomId}">
<input id="count" type="hidden" value="0">
<script type="text/javascript">
alert("The appointment has ended.");
var id = $("#idHistory").val();
var requestingPhysicianId = $("#requestingPhysicianId").val();
var consultantId = $("#consultantId").val();
window.location.replace("appointments-room-messages-history.do?clinicalCaseId=" + id
+ "&requestingPhysicianId=" + requestingPhysicianId + "&consultantId=" +
consultantId);
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/chat-room-messages-history.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div class="container-fluid">
    <br>
    <div class="container">
        <div class="row">
            <div class="col-sm-12 col-md-12">
                <div class="panel panel-default">
                    <div class="panel-heading">Appointment Chat Message History</div>
                    <div class="panel-body">
                                            <div>
                                                <table class="table">
                                                    <s:iterator
value="chatRoomMessages" var="chatMessages">
                                                        <tr>
                                                            <td style="width:
7%"><s:property value="#chatMessages.createdOn.toLocalTime().toString()" /></td>
                                                            <td style="width:
1%">|</td>
                                                            <td style="width:
15%"><s:property value="#chatMessages.physician.fullName" /></td>
                                                            <td style="width:
1%">:</td>
                                                            <td><s:property
value="#chatMessages.message" /></td>
                                                        </tr>
                                                    </s:iterator>
                                                </table>
                                            </div>
                                        <input class="btn btn-sm btn-primary"
type="button" onclick="goBack()" value="Back">
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<script type="text/javascript">
function goBack() {
    window.location.assign("consultations-clinicalcases.do");
}
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/chat-room-messages.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<s:iterator value="chatRoomMessages" var="chatMessages">
    <tr>
            <td style="width: 7%"><s:property
value="#chatMessages.createdOn.toLocalTime().toString()" /></td>
            <td style="width: 1%">|</td>
            <td style="width: 15%"><s:property
value="#chatMessages.physician.fullName" /></td>
            <td style="width: 1%">:</td>
            <td><s:property value="#chatMessages.message" /></td>
    </tr>
</s:iterator>

<input id="count" type="hidden" value="${fetchedMessagesCount}">
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/chat-room.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div class="container-fluid">
    <br>
    <div class="container">
            <div class="row">
                    <div class="col-sm-12 col-md-12">
                            <div class="panel panel-default">
                                    <div class="panel-heading">Appointment
Chat</div>
                                    <div class="panel-body">
                                            <div style="margin-bottom: 5px;">
                                                <input id="chatRoomId" type="hidden"
name="chatRoomId" value="${chatRoomId}">
                                                <input id="requestingPhysicianId"
type="hidden" name="requestingPhysicianId" value="${requestingPhysicianId}">
                                                <input id="consultantId"
type="hidden" name="consultantId" value="${consultantId}">
                                                <div>
                                                    <table
id="chatRoomMessagesTable" class="table">
                                                    </table>
                                                </div>
                                                <div>
                                                    <input id="chatMessage"
name="chatMessage" type="text" style="display: inline-block;width:93%;">
                                                    <input class="btn btn-sm btn-
primary" type="button" value="Send" onclick="sendMessage()" style="width: 6%;">
                                                </div>
                                            </div>

                                            <div>
                                              <br>  <input class="btn btn-sm btn-
primary" type="button" value="View clinical case" onclick="viewClinicalCase()">
                                            </div>
                                            <div>
                                                <form action="appointments-room-
terminate-clinicalcase-close.do">
                                                    <input type="hidden"
name="chatRoomId" value="${chatRoomId}">
                                                    <input class="cbox-button btn
btn-sm btn-primary" type="submit" value="Terminate appointment and close clinical
case">
                                                </form>
                                                <form action="appointments-room-
terminate.do">
                                    <input type="hidden" name="chatRoomId"
value="${chatRoomId}">
                                    <input class="cbox-button btn btn-sm btn-primary"
type="submit" value="Terminate appointment">
                                </form>
                                            </div>
                                    </div>
                             </div>
                    </div>
            </div>
```

```
    </div>
</div>
<script type="text/javascript">
var updateTimer;
var messageIndex;
jQuery(document).ready(function() {
    messageIndex = -1;
    getMessages();
});
function getMessages() {
    var chatRoomId = $("#chatRoomId").val();
    var requestingPhysicianId = $("#requestingPhysicianId").val();
    var consultantId = $("#consultantId").val();
    $.ajax({
            url: 'appointments-room-getMessages.do',
        type: 'post',
        data: {'chatRoomId' : chatRoomId,
            'requestingPhysicianId' : requestingPhysicianId,
            'consultantId' : consultantId,
            'messageIndex' : messageIndex
        },
        cache: false,
    })
    .done(function(data){
            $("#chatRoomMessagesTable").append(data);
            var count = $("#count");
            messageIndex =  parseInt(count.val()) + messageIndex;
            count.remove();
    });
    updateTimer = setTimeout('getMessages();',2000);
}
function sendMessage() {
    var chatMessageElement = document.getElementById("chatMessage");
    var chatMessage = chatMessageElement.value;
    if(chatMessage == '') {
            alert("You have not entered a message");
    } else {
            clearInterval(updateTimer);
            var chatRoomId = $("#chatRoomId").val();
            $.ajax({
            url: 'appointments-room-sendMessage.do',
            type: 'post',
            data: {'chatRoomId' : chatRoomId,
            'chatMessage' : chatMessage
            },
            cache: false,
        })
        .done(function(){
            chatMessageElement.value = '';
            getMessages();
        });
    }
}
function viewClinicalCase() {
    var id = $("#chatRoomId").val();
    $.facebox(function() {
        $.ajax({
            url: 'appointments-room-clinicalcase-details.do',
            type: 'post',
            data: {'clinicalCaseId' : id},
            cache: false,
        })
        .done(function(data){
            $.facebox(data);
        });
    });
}
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/appointment/ongoing-list.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div>
```

```
    <br>
    <div>
            <s:iterator value="chatRooms" var="room">
                    <s:url action="appointments-room.do" var="chatRoomUrl">
                            <s:param name="chatRoomId">
                                    <s:property value="#room.id" />
                            </s:param>
                    </s:url>
                    <a href="<s:property value="#chatRoomUrl"/>">
                        <s:property
value="#room.appointment.requestingPhysician.fullName" />'s consultation with
<s:property value="#room.appointment.consultant.fullName" />
                    </a>
            </s:iterator>
    </div>

    <form action="appointments-ongoing-forceUpdate.do">
            <div>
                    <input type="submit" value="Force Update">
            </div>
    </form>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/clinicalcase-details.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<input id="clinicalCaseIdHidden" type="hidden" name="clinicalCaseId"
value="${clinicalCaseId}">
<div id="appointmentDetailsDiv">
    <table class="table">
            <tr>
                    <td>Specialty:</td>
                    <td>${specialtyDescription}</td>
            </tr>
            <tr>
                    <td>Consultation Date:</td>
                    <td>${consultationDateString}</td>
            </tr>
            <tr>
                    <td>Primary Care Physician:</td>
                    <td>${requestingPhysicianName}</td>
            </tr>
            <tr>
                    <td>Consultant:</td>
                    <td>${consultantName}</td>
            </tr>
    </table>
</div>

<h5>Clinical Question</h5>
<textarea id="questionTextArea" style="width: 865px; height: 89px;"
name="question">${question}</textarea>

<h5>Laboratory Studies</h5>
<textarea id="patientLabExamTextArea" style="width: 865px; height: 89px;"
name="patientLabExam">${patientLabExam}</textarea>

<h5>Diagnostic Findings</h5>
<textarea id="patientDiagnosisTextArea" style="width: 865px; height: 89px;"
name="patientDiagnosis">${patientDiagnosis}</textarea>

<h5>Remarks</h5>
<textarea id="remarksTextArea" style="width: 865px; height: 89px;"
name="remarks">${remarks}</textarea>

<h5>Status</h5>
<textarea id="statusTextArea" style="width: 865px; height: 29px;" name="status"
disabled="disabled">${status}</textarea>

<h5>Files</h5>
<div id="filesDiv">
    <s:iterator value="clinicalCaseFiles" var="clinicalCaseFile" status="fileStatus">
            <div style="margin-bottom: 5px;">
```

```
                    <input id="clinicalCaseFileName<s:property
value="#clinicalCaseFile.id"/>" type="text" value=" <s:property
value="#clinicalCaseFile.name"/>" readonly="readonly" style="min-width:
260px;vertical-align: middle;">
                    <input id="fileIdForDeletion<s:property
value="#clinicalCaseFile.id"/>" type="hidden" name="fileIdsForDeletion[<s:property
value="#fileStatus.index"/>]">
                    <input class="btn btn-sm btn-primary" type="button"
value="Download" onclick="downloadFile(<s:property value="#clinicalCaseFile.id"/>);">
                    <input class="btn btn-sm btn-primary" id="deleteFileButton"
type="button" value="Delete" onclick="markFileForDeletion(this, <s:property
value="#clinicalCaseFile.id"/>)">
            </div>
    </s:iterator>
    <div id="fileUploadDiv" style="margin-bottom: 5px;">
            <input class="btn btn-sm btn-primary" style="display: inline-block;"
type="file" name="upload" onchange="addNewInputFile(this);">
    </div>
</div>

<div id="editDiv" style="margin-bottom: 5px;">
    <input class="btn btn-sm btn-primary" type="button" value="Edit Details"
onclick="editDetails()">
</div>
<div id="saveDiv" style="margin-bottom: 5px;">
    <input class="cbox-button btn btn-sm btn-primary" type="button" value="Cancel"
onclick="cancelUpdate()">
    <input class="cbox-button btn btn-sm btn-primary" type="button" value="Save"
onclick="updateDetails()">
</div>
<div style="margin-bottom: 50px;"></div>
<div id="closeDiv" style="margin-bottom: 5px;">
    <input class="cbox-button btn btn-sm btn-primary" type="button" value="Close
Clinical Case" onclick="closeClinicalCase()">
</div>
<div id="chatRoomLinkDiv" style="margin-bottom: 5px;">
    <form action="appointments-room.do">
        <input type="hidden" name="chatRoomId" value="${chatRoomId}">
        <input class="btn btn-sm btn-primary" type="submit" value="Go to appointment
chat room">
    </form>
</div>
<div id="chatRoomMessageHistoryLinkDiv" style="margin-bottom: 5px;">
    <form action="appointments-room-messages-history.do">
        <input type="hidden" name="clinicalCaseId" value="${clinicalCaseId}">
        <input id="requestingPhysicianId" type="hidden" name="requestingPhysicianId"
value="${requestingPhysicianId}">
        <input id="consultantId" type="hidden" name="consultantId"
value="${consultantId}">
        <input class="btn btn-sm btn-primary" type="submit" value="View chat message
history">
    </form>
</div>

<script type="text/javascript">
$(document).ready(function() {
    var statusValue = $('[name=status]').val();
    if(statusValue != 'Active' && statusValue != 'Ongoing') {
    $("#appointmentDetailsDiv").hide();
    }
    if(statusValue == 'Closed') {
    $("#editDiv").hide();
    }
    if(statusValue == 'Closed' || statusValue != 'Terminated') {
        $("#closeDiv").hide();
    }
    if(statusValue != 'Ongoing') {
    $("#chatRoomLinkDiv").hide();
    } else {
    $("#chatRoomMessageHistoryLinkDiv").hide();
    }
    $("#questionTextArea").attr("disabled", "disabled");
    $("#patientLabExamTextArea").attr("disabled", "disabled");
    $("#patientDiagnosisTextArea").attr("disabled", "disabled");
    $("#remarksTextArea").attr("disabled", "disabled");
    $("[id^=deleteFileButton]").hide();
```

```
    $("#fileUploadDiv").hide();
    $("#saveDiv").hide();

    var vdc = '${param.vdc}';
    if((statusValue == 'Closed' || statusValue == 'Terminated') && vdc) {
    $("#editDiv").hide();
    $("#closeDiv").hide();
    $("#chatRoomMessageHistoryLinkDiv").hide();
        setTimeout('alert("The appointment has ended");',1000)
    }
});
function downloadFile(fileId) {
    window.open("/consultations-clinicalcase-file-download.do?fileId=" + fileId,
"_blank", null);
}
function editDetails() {
    $("#questionTextArea").removeAttr("disabled");
    $("#patientLabExamTextArea").removeAttr("disabled");
    $("#patientDiagnosisTextArea").removeAttr("disabled");
    $("#remarksTextArea").removeAttr("disabled");
    $("#fileUploadDiv").show();
    $("#editDiv").hide();
    $("#saveDiv").show();
}
function markFileForDeletion(button, fileId) {
    var fileNameDivId = 'clinicalCaseFileName' + fileId;
    document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"line-through");
    button.value = 'Undo';
    button.onclick = function() {
        unmarkFileForDeletion(this, fileId);
    }
    var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
    document.getElementById(fileIdForDeletionId).value = fileId;
}
function unmarkFileForDeletion(button, fileId) {
    var fileNameDivId = 'clinicalCaseFileName' + fileId;
    document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"none");
    button.value = 'Delete';
    button.onclick = function() {
        markFileForDeletion(this, fileId);
    }
    var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
    document.getElementById(fileIdForDeletionId).value = null;
}
function addNewInputFile(element) {
    element.onchange = null;
    var deleteButton = document.createElement('input');
    deleteButton.className = 'btn btn-sm btn-primary';
    deleteButton.type = 'button';
    deleteButton.value = 'Delete';
    deleteButton.onclick = function() {
        var parentDiv = this.parentNode;
        document.getElementById('filesDiv').removeChild(parentDiv);
    }
    element.parentNode.appendChild(deleteButton);
    var newUploadFileDiv = document.createElement('div');
    var inputFile = document.createElement('input');
    inputFile.className = 'btn btn-sm btn-primary';
    inputFile.setAttribute('style', 'display: inline-block;margin-right: 4px;');
    inputFile.type = 'file';
    inputFile.name = 'upload';
    inputFile.onchange = function() {
        addNewInputFile(this);
    }
    newUploadFileDiv.appendChild(inputFile);
    newUploadFileDiv.setAttribute('style', 'margin-bottom: 5px;');
    document.getElementById('filesDiv').appendChild(newUploadFileDiv);
}
function updateDetails() {
    var formData = new FormData();
    formData.append('clinicalCaseId', $("#clinicalCaseIdHidden").val());
    formData.append('question', $("#questionTextArea").val());
    formData.append('patientDiagnosis', $("#patientDiagnosisTextArea").val());
    formData.append('patientLabExam', $("#patientLabExamTextArea").val());
```

```
    formData.append('remarks', $("#remarksTextArea").val());
    formData.append('fileIdsForDeletion', $("[name^=fileIdsForDeletion]").get(0));
    formData.append('upload', $("[name^=upload]").get(0).files[0]);
    $.facebox(function() {
        $.ajax({
            url: 'consultations-clinicalcases-details-update.do',
            type: 'post',
            data: formData,
            cache: false,
            contentType: false,
            processData: false
        })
        .done(function(data){
            $.facebox(data);
        });
    });
}
function cancelUpdate() {
    $("#questionTextArea").prop('disabled', true);
    $("#patientLabExamTextArea").prop('disabled', true);
    $("#patientDiagnosisTextArea").prop('disabled', true);
    $("#remarksTextArea").prop('disabled', true);
    $("[id^=deleteFileButton]").hide();
    $("#fileUploadDiv").hide();
    $("#editDiv").show();
    $("#saveDiv").hide();
}
function closeClinicalCase() {
    var id = $("#clinicalCaseIdHidden").val();
    $.facebox(function() {
        $.ajax({
            url: 'consultations-clinicalcases-details-closecase.do',
            type: 'post',
            data: { 'clinicalCaseId' : id},
            cache: false,
        })
        .done(function(data){
            $.facebox(data);
        });
    });
}
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/clinicalcase.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div class="container-fluid">
    <div class="container" >
    <br>
            <div class="row">
                    <div class="col-sm-12 col-md-12">
                            <div class="panel panel-default">
                                    <div class="panel-heading">Clinical Cases as
Requesting Physician</div>
                                    <div id="" class="infinite-scroll" style="max-
height: 400px;overflow-y: scroll;">
                                    <div class="panel-body">
                                            <table class="table">
                                                    <th align="center"
width="40%">Clinical Question</th>
                                                    <th align="center"
width="25%">Specialty</th>
                                                    <th align="center"
width="10%">Status</th>
                                                    <th width="5%"></th>
                                                    <s:iterator
value="userCreatedClinicalCases"

    var="userCreatedClinicalCasesVar"
status="userCreatedClinicalCasesIteratorStatus">
                                                                    <tr id="case<s:property
value="#userCreatedClinicalCasesVar.id" />">
    <td><s:property value="#userCreatedClinicalCasesVar.question" /></td>
```

```
    <td><s:property value="#userCreatedClinicalCasesVar.specialty.description"
/></td>             <td><s:property value="#userCreatedClinicalCasesVar.status"
/></td>
                                                                    <td><input
class="btn btn-sm btn-primary" type="button" value="Details"
onclick="viewDetails(<s:property value="#userCreatedClinicalCasesVar.id" />)"></td>
                                                                </tr>
                                                        </s:iterator>
                                                </table>
                                                </div>
                                        </div>
                                </div>
                        </div>
                </div>
                <div class="row">
                        <div class="col-sm-12 col-md-12">
                                <div class="panel panel-default">
                                        <div class="panel-heading">Clinical Cases as
Consultant</div>
                                        <div id="" class="infinite-scroll" style="max-
height: 400px;overflow-y: scroll;">
                                        <div class="panel-body">
                                                <table class="table">
                                                        <th align="center"
width="40%">Clinical Question</th>
                                                        <th align="center"
width="25%">Specialty</th>
                                                        <th align="center"
width="10%">Status</th>
                                                        <th width="5%"></th>
                                                        <s:iterator
value="clinicalCasesAsConsultant"

    var="clinicalCasesAsConsultantVar"
status="clinicalCasesAsConsultantVarIteratorStatus">
                                                                <tr id="case<s:property
value="#clinicalCasesAsConsultantVar.id" />">
                                                                        <td><s:property
value="#clinicalCasesAsConsultantVar.question" /></td>
                                                                        <td><s:property
value="#clinicalCasesAsConsultantVar.specialty.description" /></td>

<td><s:property value="#clinicalCasesAsConsultantVar.status" /></td>
                                                                        <td><input
class="btn btn-sm btn-primary" type="button" value="Details"
onclick="viewDetailsConsultant(<s:property value="#clinicalCasesAsConsultantVar.id"
/>)"></td>
                                                                </tr>
                                                        </s:iterator>
                                                </table>
                                                </div>
                                        </div>
                                </div>
                        </div>
                </div>
    </div>
</div>

<script type="text/javascript">
$(document).bind('close.facebox', function() {
    location.reload();
});
function viewDetails(id) {
    $.facebox(function() {
            $.ajax({
                    url: "consultations-clinicalcases-details.do",
                    data: { 'clinicalCaseId' : id},
                    type: 'post',
                    cache: false
            })
            .done(function(data){
                    $.facebox(data);
            });
    });
};
function viewDetailsConsultant(id) {
```

```
    $.facebox(function() {
        $.ajax({
            url: "consultations-clinicalcases-details.do",
            data: {
            'clinicalCaseId' : id,
               'vdc' : true
            },
            type: 'post',
            cache: false
        })
        .done(function(data){
            $.facebox(data);
        });
    });
};
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment-clinicalcase-seletion.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<div>
    <form id="clinicalCaseSelectionForm" action="consultations-appointment-
clinicalcase-finalize.do">
            <input id="specialtyId" type="hidden" name="specialtyId"
value="${specialtyId}">
            <input id="consultantId" type="hidden" name="consultantId"
value="${consultantId}">
            <input id="startDate" type="hidden" name="startDate"
value="${startDate}">
            <input id="endDate" type="hidden" name="endDate" value="${endDate}">
            <input id="clinicalCaseId" type="hidden" name="clinicalCaseId"
value="${clinicalCaseId}">
            <table class="table">
                    <th align="center">Clinical Question</th>
                    <s:iterator value="clinicalCaseAvailable" var="clinicalCaseVar">
                            <tr id="case<s:property value="#clinicalCaseVar.id" />"
onmouseover="changeColor(this, true, <s:property value="#clinicalCaseVar.id" />);"
                                    onmouseout="changeColor(this, false, <s:property
value="#clinicalCaseVar.id" />);" onclick="selectClinicalCase(this, <s:property
value="#clinicalCaseVar.id" />);">
                                    <td><s:property
value="#clinicalCaseVar.question" /></td>
                            </tr>
                    </s:iterator>
                    <tr id="case-1" onmouseover="changeColor(this, true, '-1');"
onmouseout="changeColor(this, false, '-1');" onclick="selectClinicalCase(this, '-
1');">
                            <td>New Clinical Case</td>
                    </tr>
            </table>
            <div>
                    <input class="btn btn-sm btn-primary" type="button"
value="Select" onclick="submitSelection()">
                    <input class="btn btn-sm btn-primary"type="button"
value="Cancel" onclick="cancelSelection()">
            </div>
    </form>
</div>

<script type="text/javascript">
jQuery(document).ready(function() {
    var defaultSelection = document.getElementById('case-1');
    selectClinicalCase(defaultSelection, -1);
});
function changeColor(element, highLight, id) {
    if (highLight) {
        element.style.backgroundColor = '#dcfac9';
    } else {
        var selectedClinicalCaseId = document.getElementById('clinicalCaseId').value;
        if(id == selectedClinicalCaseId) {
            element.style.backgroundColor = '#D5D5FC';
        } else {
            element.style.backgroundColor = 'white';
        }
    }
```

```
}
function selectClinicalCase(element, clinicalCaseId) {
    var clinicalCaseIdElement = document.getElementById('clinicalCaseId');
    var previouslySelected = clinicalCaseIdElement.value;
    var previouslySelectedElementId = 'case' + previouslySelected;
    var previouslySelectedElement =
document.getElementById(previouslySelectedElementId);
    if(previouslySelectedElement != null) {
        changeColor(previouslySelectedElement, false);
    }
    element.style.backgroundColor = '#D5D5FC';
    clinicalCaseIdElement.value = clinicalCaseId;
}
function submitSelection() {
    var formData = $("#clinicalCaseSelectionForm").serialize();
    $.facebox(function() {
        $.ajax({
            url: 'consultations-appointment-clinicalcase-finalize.do',
            type: 'post',
            data: formData,
            cache: false,
        })
        .done(function(data){
            $.facebox(data);
        });
    });
}
function cancelSelection() {
    $.facebox.close();
}
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment-finalize.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<div>
    <h5>Clinical Case Information</h5>
    <form action="consultations-appointment-clinicalcase-save.do" method="post"
enctype="multipart/form-data">
            <input id="specialtyId" type="hidden" name="specialtyId"
value="${specialtyId}">
            <input id="consultantId" type="hidden" name="consultantId"
value="${consultantId}">
            <input id="startDate" type="hidden" name="startDate"
value="${startDate}">
            <input id="endDate" type="hidden" name="endDate" value="${endDate}">
            <input id="clinicalCaseId" type="hidden" name="clinicalCaseId"
value="${clinicalCaseId}">
            <div>
                    <table class="table">
                            <tr>
                                    <td>Specialty:</td>
                                    <td>${specialtyDescription}</td>
                            </tr>
                            <tr>
                                    <td>Consultation Date:</td>
                                    <td>${consultationDateString}</td>
                            </tr>
                            <tr>
                                    <td>Primary Care Physician:</td>
                                    <td>${requestingPhysicianName}</td>
                            </tr>
                            <tr>
                                    <td>Consultant:</td>
                                    <td>${consultantName}</td>
                            </tr>
                    </table>
            </div>

            <h5>Clinical Question</h5>
            <textarea style="width: 865px; height: 89px;"
name="question">${question}</textarea>

            <h5>Laboratory Studies</h5>
            <textarea style="width: 865px; height: 89px;"
name="patientLabExam">${patientLabExam}</textarea>
```

```
            <h5>Diagnostic Findings</h5>
            <textarea style="width: 865px; height: 89px;"
name="patientDiagnosis">${patientDiagnosis}</textarea>

            <h5>Remarks</h5>
            <textarea style="width: 865px; height: 89px;"
name="remarks">${remarks}</textarea>

            <h5>Files</h5>
            <div id="uploadFilesDiv">
                <s:iterator value="clinicalCaseFiles" var="clinicalCaseFile"
status="status">
                <div style="margin-bottom: 5px;">
                    <input id="clinicalCaseFileName<s:property
value="#clinicalCaseFile.id"/>" type="text" value=" <s:property
value="#clinicalCaseFile.name"/>" readonly="readonly" style="min-width:
260px;vertical-align: middle;">
                    <input id="fileIdForDeletion<s:property
value="#clinicalCaseFile.id"/>" type="hidden" name="fileIdsForDeletion[<s:property
value="#status.index"/>]">
                    <input class="btn btn-sm btn-primary" type="button"
value="Download" onclick="downloadFile(<s:property value="#clinicalCaseFile.id"/>);">
                    <input class="btn btn-sm btn-primary" type="button"
value="Delete" onclick="markFileForDeletion(this, <s:property
value="#clinicalCaseFile.id"/>)">
                </div>
                </s:iterator>
                    <div style="margin-bottom: 5px;">
                            <input class="btn btn-sm btn-primary" style="display:
inline-block;" type="file" name="upload" onchange="addNewInputFile(this);">
                    </div>
            </div>

            <div>
                <input class="cbox-button btn btn-sm btn-primary" type="button"
value="Cancel" onclick="cancelAppointment()">
                <input class="cbox-button btn btn-sm btn-primary" type="submit"
value="Book">
            </div>

    </form>

</div>

<script type="text/javascript">
function downloadFile(fileId) {
    window.open("/consultations-clinicalcase-file-download.do?fileId=" + fileId,
"_blank", null);
}
function markFileForDeletion(button, fileId) {
    var fileNameDivId = 'clinicalCaseFileName' + fileId;
    document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"line-through");
    button.value = 'Undo';
    button.onclick = function() {
        unmarkFileForDeletion(this, fileId);
    }
    var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
    document.getElementById(fileIdForDeletionId).value = fileId;
}
function unmarkFileForDeletion(button, fileId) {
    var fileNameDivId = 'clinicalCaseFileName' + fileId;
    document.getElementById(fileNameDivId).style.setProperty("text-decoration",
"none");
    button.value = 'Delete';
    button.onclick = function() {
        markFileForDeletion(this, fileId);
    }
    var fileIdForDeletionId = 'fileIdForDeletion' + fileId;
    document.getElementById(fileIdForDeletionId).value = null;
}
function addNewInputFile(element) {
    element.onchange = null;
    var deleteButton = document.createElement('input');
    deleteButton.className = 'btn btn-sm btn-primary';
```

```
    deleteButton.type = 'button';
    deleteButton.value = 'Delete';
    deleteButton.onclick = function() {
        var parentDiv = this.parentNode;
        document.getElementById('uploadFilesDiv').removeChild(parentDiv);
    }
    element.parentNode.appendChild(deleteButton);
    var newUploadFileDiv = document.createElement('div');
    var inputFile = document.createElement('input');
    inputFile.className = 'btn btn-sm btn-primary';
    inputFile.setAttribute('style', 'display: inline-block;margin-right: 4px;');
    inputFile.type = 'file';
    inputFile.name = 'upload';
    inputFile.onchange = function() {
        addNewInputFile(this);
    }
    newUploadFileDiv.appendChild(inputFile);
    newUploadFileDiv.setAttribute('style', 'margin-bottom: 5px;');
    document.getElementById('uploadFilesDiv').appendChild(newUploadFileDiv);
}
function cancelAppointment() {
    $.facebox.close();
}
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment-loadphysicians.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<div id="physicians" class="infinite-scroll" style="height:99px; max-height: 99px;
overflow-y: scroll;">
    <div class="">
            <a class="jscroll-physicians-next" href="consultations-appointment-
physicians.do?physicianPage=1&specialtyId=${specialtyId}">next</a>
    </div>
</div>

<script type="text/javascript">
    jQuery(document).ready(function() {
        jQuery('#physicians').jscroll({
            loadingHtml: '<img src="images/loading.gif" alt="Loading"/> Loading...',
            nextSelector: 'a.jscroll-physicians-next:last',
            debug: true
        });
    });
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment-physician-schedule.jsp**

```
<%@taglib uri="/struts-tags" prefix="s"%>
<div id="consultationScheduler">
<s:property escape="false" value="messageStore.planner" />
</div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment-physicians.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="/struts-tags" prefix="s"%>
<c:choose>
    <c:when test="${physicianPage == 1 &&
!physiciansOfSpecificSpecialty.hasContent()}">No result found</c:when>
    <c:otherwise>
            <c:forEach items="${physiciansOfSpecificSpecialty.content}"
var="physician">
                <c:if test="${currentLoggedInUser.id != physician.id}">
                        <div id="phys${physician.id}" onmouseover="changeColor(this,
true, ${physician.id});" onmouseout="changeColor(this, false, ${physician.id});"
onclick="loadPhysicianPlanner(this, ${physician.id});">
                            <div class="media">
                                <div class="pull-left">
                                  <br>  <img src="${physician.imageUrl == null ?
'images/default-user-image.png' : physician.imageUrl}" alt="User Image"
style="height: 65px; width: 65px">
                                </div>
                                <div class="media-body">
```

```
                                    <br>
                                        <div class="panel panel-default" style="margin-
right: 14px;">
                                            <div class="panel-heading" style="height:
auto">
                                                <h5 class="media-heading">
                                                    <a href="profile-
view.do?id=${physician.id}" target="_blank"> ${physician.fullName}</a>
                                                </h5>
                                                <c:forEach
items="${physician.specialties}" var="specialty"
varStatus="specialtyVarStatus"><c:out value="${specialty.description}" /><c:if
test="${!specialtyVarStatus.isLast()}">;</c:if> </c:forEach>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </c:if>
                        <c:if test="${currentLoggedInUser.id == physician.id && physicianPage
== 1 && physiciansOfSpecificSpecialty.getTotalElements() == 1}">No result
found</c:if>
                    </c:forEach>
                    <c:if test="${physiciansOfSpecificSpecialty.number + 1 <
physiciansOfSpecificSpecialty.totalPages }">
                        <div class="well bs-component"><a class="jscroll-physicians-next"
href="consultations-appointment-
physicians.do?physicianPage=${physiciansOfSpecificSpecialty.number +
2}&specialtyId=${specialtyId}">next</a></div>
                    </c:if>
                </c:otherwise>
            </c:choose>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/consultation/create-appointment.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div class="container-fluid">
    <div class="container" >
    <br>
            <div class="row">
                    <div class="col-sm-4 col-md-4">
                            <div class="panel panel-default">
                                    <div class="panel-heading">Specialty</div>
                                    <div class="panel-body">
                                            <form id="loadPhysiciansForm"
                                                    action="consultations-
appointment-loadPhysicians.do">
                                                    <div>
                                                            <s:select
list="specialtyList" name="specialtyId" cssClass="form-control" headerKey="-1"
headerValue="" listKey="id" listValue="description" />
                                                    </div>

                                                    <div style="margin-top: 30px;">
                                                            <input class="btn btn-sm
btn-primary btn-block" type="button" value="Find a Doctor"
onclick="loadPhysicians()">
                                                    </div>
                                            </form>
                                    </div>
                            </div>
                    </div>
                    <div class="col-sm-8 col-md-8">
                            <div class="panel panel-default" >
                                    <div class="panel-heading">Physicians</div>
                                    <div id="viewphysicians" class="panel-
body"></div>
                            </div>
                    </div>
            </div>
            <div class="row" >
                    <div class="col-sm-15 col-md-12">
                            <div class="panel panel-default">
                                    <div class="panel-heading" >Schedule</div>
```

```
                                        <div id="planner" class="panel-body"></div>
                                </div>
                        </div>
                </div>
        </div>
</div>
<input id="consultantIdHidden" type="hidden"/>

<script type="text/javascript">
        var selectedPhysician;
        function loadPhysicians() {
                selectedPhysician = null;
                var scheduler = document.getElementById("consultationScheduler");
                if(scheduler != null) {
                        scheduler.remove();
                }
                var formData = $("#loadPhysiciansForm").serialize();
                $.ajax({
                        url : "consultations-appointment-loadPhysicians.do",
                        data : formData,
                        type : 'post',
                        cache : false
                }).done(function(data) {
                        $("#viewphysicians").html(data);
                });
        };
        function changeColor(element, highLight, id) {
                if (highLight) {
                        element.style.backgroundColor = '#dcfac9';
                } else {
                        if (selectedPhysician == element) {
                                element.style.backgroundColor = '#D5D5FC';
                        } else {
                                element.style.backgroundColor = 'white';
                        }
                }
        };
        function loadPhysicianPlanner(element, consultantId) {
                if (selectedPhysician != null) {
                        selectedPhysician.style.backgroundColor = 'white';
                }
                element.style.backgroundColor = '#D5D5FC';
                selectedPhysician = element;
                document.getElementById("consultantIdHidden").value = consultantId;
                $.ajax({
                        url : "consultations-appointment-loadSchedule.do",
                        data : {
                                'consultantId' : consultantId
                        },
                        type : 'post',
                        cache : false
                }).done(function(data) {
                        $("#planner").html(data);
                });
        };
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/dashboard/dashboard.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="dashboard"/>
<%@include file="../includes/nav.jsp" %>
<div class="container-fluid">
<div class="container" >
<br>
        <c:choose>
                <c:when test="${currentLoggedInUser.userType =='Physician'}">
                <div class="row">
                        <div class="col-sm-6 col-md-6">
                                <div class="panel panel-default">
                                <div class="panel-heading">Latest Discussions</div>
                                <div id="latestdiscussions" class="panel-body" >
                                        <div id="discussions" class="infinite-scroll"
style="max-height: 300px;overflow-y: scroll;">
```

```
                                    <div class=""><a class="jscroll-next"
href="dashboard-loaddiscussions.do?page=1">next</a> </div>
                                    </div>
                            </div>
                            </div>
                    </div>

                    <div class="col-sm-6 col-md-6">
                            <div class="panel panel-default" >
                            <div class="panel-heading">Latest Appointments</div>
                            <div id="latestappointments" class="panel-body" >
                                    <div id="appointments" class="infinite-scroll"
style="height: 300px;max-height: 300px;overflow-y: scroll;">
                                    <div class=""><a class="jscroll-appointment-
next" href="dashboard-loadappointments.do?page=1">next</a> </div>
                                    </div>
                            </div>
                            </div>
                    </div>
            </div>
            <div class="row">
                    <div class="col-sm-6 col-md-6">
                            <div class="panel panel-default" >
                            <div class="panel-heading">View Doctors</div>
                            <div id="viewphysicians" class="panel-body" >
                                    <div id="physicians" class="infinite-scroll"
style="max-height: 365px;overflow-y: scroll;">
                                    <div class=""><a class="jscroll-physicians-next"
href="dashboard-loadphysicians.do?page=1">next</a> </div>
                                    </div>
                            </div>
                            </div>
                    </div>

                    <div class="col-sm-6 col-md-6">
                <div class="panel panel-default" >
                <div class="panel-heading">Consultation Trend</div>
                <div id="consultationTrends" class="panel-body" >
                </div>
                </div>
            </div>
            </div>
            </c:when>
            <c:otherwise>
            <div class="row">
                    <div class="col-sm-6 col-md-6">
                <div class="panel panel-default" >
                <div class="panel-heading">Consultation Trends</div>
                <div id="consultationTrends" class="panel-body" >
                </div>
                </div>
            </div>
            </div>
            </c:otherwise>
    </c:choose>

</div><!-- container end -->
</div><!-- container-fluid end-->

<script type="text/javascript">
    jQuery(document).ready(function() {
            jQuery('#discussions').jscroll({
                loadingHtml: '<img src="images/loading.gif" alt="Loading" />
Loading...',
                nextSelector: 'a.jscroll-next:last',
                debug: true
            });

            jQuery('#physicians').jscroll({
                    loadingHtml: '<img src="images/loading.gif" alt="Loading"/>
Loading...',
                    nextSelector: 'a.jscroll-physicians-next:last',
                    debug: true
            });

            jQuery('#appointments').jscroll({
```

```
                    loadingHtml: '<img src="images/loading.gif" alt="Loading"/>
Loading...',
                    nextSelector: 'a.jscroll-appointment-next:last',
                    debug: true
            });
            jQuery.ajax({
            url: 'dashboard-loadtrends.do',
            type: 'GET',
            dataType: 'html'
          }).done(function ( data ) {
              jQuery("#consultationTrends").html(data);
      });
            checkForNotifications();
            setInterval(checkForNotifications,10000);
    });

    function checkForNotifications(){
            jQuery.ajax({
                      url: 'dashboard-loadnotifications.do',
                      type: 'GET',
                      dataType: 'html'
                    }).done(function ( data ) {
                            jQuery("#notifications").html(data);
            });
    }
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/dashboard/loadappointment.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:forEach items="${appointmentShortDetails}" var="appointmentString"
varStatus="status">
    <div class="well bs-component"  style="margin-right: 15px;">
       <p>${appointmentString}</p>
    </div>
</c:forEach>
<c:if test="${appointmentPage.number + 1 < appointmentPage.totalPages }">
    <div class="well bs-component">
            <a class="jscroll-appointment-next" href="dashboard-
loadappointments.do?page=${appointmentPage.number + 2}">next</a>
    </div>
</c:if>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/dashboard/loaddiscussions.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:forEach items="${ discussionThreadPage.content}" var="thread">
    <div class="media" style="margin-right: 15px;">
      <div class="media-body">
        <div class="panel panel-default">
            <div class="panel-heading">
                    <a href="discussions.do?id=${thread.id }">
                     <h5 class="media-heading">${thread.title}</h5>
                </a>
                <p class="text-right">by: ${thread.createdBy.fullName } </p>
            </div>
            </div>
      </div>
    </div>
</c:forEach>
<c:choose>
<c:when test="${discussionThreadPage.number + 1 < discussionThreadPage.totalPages }">
    <div class="panel-footer" style="margin-right: 15px;"><a
href="discussions.do?filter=all"><center>View All Discussions</center></a> </div>
</c:when>
<c:otherwise>
</c:otherwise>
</c:choose>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/dashboard/loadphysicians.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<%@taglib uri="/struts-tags" prefix="s" %>
```

```
<c:forEach items="${ physicianPage.content}" var="physician">
    <c:if test="${currentLoggedInUser.id != physician.id}">
    <div class="media" style="margin-right: 15px;">
      <a class="pull-left" href="profile-view.do?id=${physician.id }">
        <img src="${physician.imageUrl == null ? 'images/default-user-image.png' :
physician.imageUrl}" alt="User Image" style="height: 65px; width: 65px; vertical-
align: middle;">
      </a>
      <div class="media-body">
        <div class="panel panel-default">
            <div class="panel-heading" style="height: auto">
                <h4 class="media-heading"><a href="profile-
view.do?id=${physician.id}">${physician.fullName }</a></h4>
                <c:forEach items="${physician.specialties}" var="specialty"
varStatus="specialtyVarStatus"><c:out value="${specialty.description}" /><c:if
test="${!specialtyVarStatus.isLast()}">;</c:if> </c:forEach>
            </div>
            </div>
      </div>
    </div>
    </c:if>
</c:forEach>
<c:choose>
<c:when test="${physicianPage.number + 1 < physicianPage.totalPages }">
    <div class="panel-footer" style="margin-right: 15px;"><a
href="profiles.do"><center>View All Doctors</center></a> </div>
</c:when>
<c:otherwise>
</c:otherwise>
</c:choose>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/dashboard/loadtrends.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<link href="css/legend.css" rel="stylesheet" type="text/css"  />
<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/chart/Chart.js" type="text/javascript"></script>
<script src="scripts/chart/legend.js" type="text/javascript"></script>
<canvas id="consultationTrendsCanvas"></canvas>
<h2></h2><div id="legendDiv"></div>
<script type="text/javascript">
    jQuery(document).ready(function() {
        var consultationTrends = JSON.parse('<s:property value="consultationTrends"
escapeHtml="false"/>');

        var chartData = {
                    labels: ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"],
                    datasets: [
                       {
                       label: "Last Week",
                       fillColor: "rgba(200,200,200,0.5)",
                           strokeColor: "rgba(200,200,200,0.8)",
                           highlightFill: "rgba(200,200,200,0.75)",
                           highlightStroke: "rgba(200,200,200,1)",
                           data: [
                                        consultationTrends["monday1"],
                                        consultationTrends["tuesday1"],
                                        consultationTrends["wednesday1"],
                                        consultationTrends["thursday1"],
                                        consultationTrends["friday1"],
                                        consultationTrends["saturday1"],
                                        consultationTrends["sunday1"]
                                ]
                       },
                       {
                           label: "This Week",
                           fillColor: "rgba(151,187,205,0.5)",
                           strokeColor: "rgba(151,187,205,0.8)",
                           highlightFill: "rgba(151,187,205,0.75)",
                           highlightStroke: "rgba(151,187,205,1)",
                           data: [
                                     consultationTrends["monday2"],
                                     consultationTrends["tuesday2"],
                                     consultationTrends["wednesday2"],
                                     consultationTrends["thursday2"],
```

```
                                    consultationTrends["friday2"],
                                    consultationTrends["saturday2"],
                                    consultationTrends["sunday2"]
                                ]
                        },
                        {
                            label: "Next Week",
                            fillColor: "rgba(34,102,102,0.5)",
                            strokeColor: "rgba(34,102,102,0.8)",
                            highlightFill: "rgba(34,102,102,0.75)",
                            highlightStroke: "rgba(34,102,102,1)",
                            data: [
                                    consultationTrends["monday3"],
                                    consultationTrends["tuesday3"],
                                    consultationTrends["wednesday3"],
                                    consultationTrends["thursday3"],
                                    consultationTrends["friday3"],
                                    consultationTrends["saturday3"],
                                    consultationTrends["sunday3"]
                                ]
                        }
                    ]
        }

        var ctx =
document.getElementById("consultationTrendsCanvas").getContext("2d");
        var barChart = new Chart(ctx).Bar(chartData, {
            responsive: true
        });
        legend(document.getElementById("legendDiv"), chartData);
    });
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/discussion/discussion-reply-form.jsp**

```
<div class="table-responsive">
    <table>
            <tr>
            <td>
            <textarea name="body" class="form-control" rows="3"></textarea>
            </td>
            </tr>
    </table>
</div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/discussion/discussions-form.jsp**

```
<c:set var="isUpdate" value="${discussionThread.id eq null ? false : true}"/>
<div class="panel panel-default">
<div class="panel-heading">
    <p>${discussionThread.id eq null ? 'New Discussion' : discussionThread.title}</p>
    <div align="right"><h6>Created by: ${discussionThread.createdBy.fullName
}</h6></div>
</div>
<div class="panel-body">
    <form class="form-default" action="${discussionThread.id eq null ? 'discussions-
saveInitialThread' : 'discussions-saveReply'}.do">
    <input type="hidden" value="${discussionThread.id}" name="id" id="id">
    <div align="center">
    <table cellpadding="10" cellspacing="0" width="100%" >
            <c:if test="${!isUpdate}">
            <tr>
            <td>
            <input type="text" class="form-control" name="title" id="title"
placeholder="Title" value="${discussionThread.title }" required/><br>
            </td>
            </tr>
            </c:if>
            <c:if test="${isUpdate}">
            <tr >
            <td>
            <table class="table table-striped">
            <c:forEach items="${discussionThread.discussionThreadReplies }"
var="reply" varStatus="i">
            <tr>
            <td colspan="3">
```

```
            <p>${reply.message }</p>
            <c:if test="${i.count <
fn:length(discussionThread.discussionThreadReplies)}">
            <h6 align="right">Posted by ${reply.postedBy.fullName } on <joda:format
value="${reply.createdOn }" style="SM" /></h6>
            </c:if>
            </td>
            </tr>
            </c:forEach>
            </table>
            </td>
            </tr>
            </c:if>
            <tr>
            <td colspan="3">
            <textarea name="message" class="form-control" rows="3" required
placeholder="Write a comment ..."></textarea>
            </td>
            </tr>
            <tr>
            <td colspan="3" align="right">
            <br><input type="submit" class="btn btn-sm btn-success" value="Go"/>
            </td>
            </tr>
    </table>
    </div>
    </form>
    </div>
</div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/discussion/discussions.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="discussions"/>
<%@include file="../includes/nav.jsp" %>
<div class="container-fluid">
<h5></h5>
    <div class="container">
            <!-- <h1 class="page-header"> -->
            <class="page-header">
                    <c:choose>
                    <c:when test="${param.filter == 'user' }">
                            <h3><b>My Discussions</b></h3>
                            <h3></h3><a href="discussions.do?display=form">Create
Discussion Thread</a>
                    </c:when>
                    <c:when test="${param.filter ==  'subscriptions'}">
                            <h3><b>Subscriptions</b></h3>
                            <h3></h3><a href="discussions.do?display=form">Create
Discussion Thread</a>
                    </c:when>
                    <c:when test="${param.filter == 'all' }">
                            <h3><b>All Discussions</b></h3>
                            <h3></h3><a href="discussions.do?display=form">Create
Discussion Thread</a>
                    </c:when>
                    <c:when test="${param.id != null }">
                            <h3><b></b></h3>
                            <h3></h3><a href="discussions.do?display=form">Create
Discussion Thread</a>
                    </c:when>
                    <c:otherwise>
                            <h3><b>New Discussion</b></h3>
                            <br>
                    </c:otherwise>
                    </c:choose>

                    <c:if test="${discussionThread.id ne null }">
                    <c:set var="canSubscribe" value="true"/>
                    <c:set var="isSubscriber" value="false"/>
                    <c:forEach items="${discussionThread.subscriptions }" var="sub">
                            <c:if test="${sub.follower.id == currentLoggedInUser.id
}">
                                    <c:set var="canSubscribe" value="false"/>
                                    <c:set var="isSubscriber" value="true"/>
                            </c:if>
```

```
                    </c:forEach>
                    <c:if test="${discussionThread.createdBy.id ==
currentLoggedInUser.id }">
                            <c:set var="canSubscribe" value="false"/>
                    </c:if>
                    <c:if test="${canSubscribe }">
                    <form action="discussions-subscribe.do" method="post"
style="float: right;">
                            <input type="hidden" value="${discussionThread.id}"
name="id"/>
                            <input  type="submit" class="btn btn-sm btn-primary "
value="Subscribe"/>
                    </form>
                    <br><br>
                    </c:if>
                    <c:if test="${isSubscriber }">
                    <form action="discussions-unsubscribe.do" method="post"
style="float: right;">
                            <input type="hidden" value="${discussionThread.id}"
name="id"/>
                            <input  type="submit" class="btn btn-sm btn-primary "
value="Unsubscribe"/>
                    </form>
                    <br><br>
                    </c:if>
                    <c:if test="${discussionThread.createdBy.id ==
currentLoggedInUser.id }">
                    <form action="discussions-delete.do" method="post" style="float:
right;">
                            <input type="hidden" value="${discussionThread.id}"
name="id"/>
                            <input type="submit" class="btn btn-sm btn-primary "
value="Delete"/>
                    </form>
                    <br><br>
                    </c:if>
                    </c:if>
            </h1>
            <c:choose>
                    <c:when test="${param.display eq 'form' or discussionThread.id
ne null }">
                       <%@include file="discussions-form.jsp" %>
                    </c:when>
                    <c:otherwise>
                    <br>
                    <br>
                    <table class="table table-striped">
                            <tr>
                                    <th>

                                    </th>
                                    <th>
                                            DISCUSSION
                                    </th>
                                    <th align="center">
                                            DATE POSTED
                                    </th>
                                    <th>
                                            REPLIES
                                    </th>
                                    <th>
                                            LATEST POST
                                    </th>
                            </tr>
                            <c:forEach items="${discussionThreads.content}"
var="thread" >
                                    <tr>
                                            <td  >

                                            </td>
                                            <td>
                                                    <a
href="discussions.do?id=${thread.id }">
                                                            <p><h5> ${thread.title }
</h5></p>
                                                    </a>
```

```
                                                        <h6>by
${thread.createdBy.fullName }</h6>
                                                </td>
                                                <td  ><br>
                                                        <joda:format
value="${thread.createdOn}" style="FM" />
                                                </td>
                                                <td ><br>

    ${fn:length(thread.discussionThreadReplies) }
                                                </td>
                                                <td ><br>
                                                        <joda:format
value="${thread.discussionThreadReplies[0].createdOn }" style="MM" />
                                                </td>
                                        </tr>
                                </c:forEach>
                        </table>
                        <center><%@include file="paging.jsp" %> </center>
                        </c:otherwise>
                </c:choose>
                </class>
    </div>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/discussion/paging.jsp**

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<c:set var="currentIndex" value="${discussionThreads.number + 1 }"/>
<c:url var="firstUrl" value="discussions.do?filter=${request.filter }" />
<c:url var="lastUrl" value="discussions.do?filter=${request.filter
}&currentPage=${discussionThreads.totalPages}" />
<c:url var="prevUrl" value="discussions.do?filter=${request.filter
}&currentPage=${currentIndex - 1}" />
<c:url var="nextUrl" value="discussions.do?filter=${request.filter
}&currentPage=${currentIndex + 1}" />
<div class="container-fluid">

<nav>
    <ul class="pagination" style="text-align: center;" >
        <c:choose>
            <c:when test="${currentIndex == 1 || currentIndex == 0}">
            </c:when>

            <c:otherwise>
             <li><a href="${firstUrl}"aria-label="Previous"><span aria-
hidden="true">&laquo;</a></li>
            </c:otherwise>
        </c:choose>

        <c:forEach var="i" begin="1" end="${discussionThreads.totalPages }" step="1">
            <c:url var="pageUrl" value="discussions.do?filter=${request.filter
}&currentPage=${i}" />
            <c:choose>
                <c:when test="${i == currentIndex}">
                    <li class="active"><a href="${pageUrl}"><b><c:out value="${i}"
/></b><span class="sr-only">(current)</span></a></li>
                </c:when>
                <c:otherwise>
                    <li><a href="${pageUrl}"><c:out value="${i}" /></a></li>
                </c:otherwise>
            </c:choose>
        </c:forEach>

        <c:choose>
            <c:when test="${currentIndex == discussionThreads.totalPages }">

            </c:when>

             <c:when test="${currentIndex -1 == discussionThreads.totalPages}">

            </c:when>

            <c:otherwise>
```

```
                <li><a href="${lastUrl}"aria-label="Previous"><span aria-
hidden="true">&raquo;</a></li>
            </c:otherwise>
        </c:choose>
    </ul>
</nav>
 </div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/includes/errors.jsp**

```
<style type="text/css">
.errors {
  color: #a94442;
  background-color: #f2dede;
  width:45.2%;
  padding: 0px;
  border: 1px solid transparent;
  border-radius: 4px;

}
.errors li{
    list-style-type: none;
}

.success {
color: #3c763d;
  background-color: #dff0d8;
  width:45.2%;
  padding: 0px;
  border: 1px solid transparent;
  border-radius: 4px;

}
.success li{
    list-style-type: none;
}

</style>

<s:if test="hasActionErrors()">
   <div class="errors" role="alert"><h4></h4>
     <s:actionerror/>
   </div><br>
</s:if>
<s:if test="hasActionMessages()">
   <div class="success" role="alert"> <h4></h4>
     <s:actionmessage/>
   </div><br>
</s:if>

<div class="clear"></div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/includes/footer.jsp**

```
<c:if test="${currentLoggedInUser.userType =='Physician'}">
 <br><br>
  <div class="footer">
       <div class="container">
       <br>
       <center>
        <p class="text-muted"> Copyright © 2015 Web PCS </p>
        </center>
      </div>
    </div>
    </c:if>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/includes/header.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
    <%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

```
    <%@taglib uri="http://www.joda.org/joda/time/tags" prefix="joda" %>
    <%@taglib uri="/struts-tags" prefix="s" %>

<!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=UTF-8">
            <title>Web PCS</title>
            <link href="css/ui.jqgrid.css" rel="stylesheet" type="text/css"/>
            <link href="css/jquery-ui.css" rel="stylesheet" type="text/css"/>
            <link href="css/bootstrap/bootstrap-acrulian-theme.css" rel="stylesheet"
type="text/css"/>
            <link href="css/bootstrap/default-dashboard-template.css"
rel="stylesheet" type="text/css"/>
            <link href="css/facebox.css" rel="stylesheet" type="text/css"  />
            <link href="css/chosen.bootstrap.min.css" rel="stylesheet"
type="text/css"/>

            <link rel="icon"
      type="image/png"
      href="images/fav1.png">


            <script src="scripts/jq/jquery-1.11.0.min.js"
type="text/javascript"></script>
            <script src="scripts/jq/jquery-ui.js" type="text/javascript"></script>
            <script src="scripts/jqGrid/jquery.jqGrid.min.js"
type="text/javascript"></script>
            <script src="scripts/jqGrid/grid.locale-en.js"
type="text/javascript"></script>
            <script src="scripts/jScroll/jquery.jscroll.min.js"
type="text/javascript"></script>

            <script src="scripts/facebox/facebox.js" type="text/javascript"></script>
            <script src="scripts/chosen.jquery.min.js"
type="text/javascript"></script>

            <script src="scripts/bootstrap/bootstrap.js"
type="text/javascript"></script>

            <script type="text/javascript">
                    $(document).ready(function() {
                            $('a[rel*=facebox]').facebox();
                    });
                    $.facebox.settings.closeImage =
'/images/facebox/closelabel.png';
                    $.facebox.settings.loadingImage = '/images/facebox/loading.gif';
            </script>
    </head>
    <body>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/includes/nav.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="usersettings"/>
<div class="container">
<%@include file="../includes/nav.jsp" %>
<script type='text/javascript' src='<c:url
value="scripts/ModalChangePassword.js"/>'></script>
<c:set var="isUpdate" value="${(entity.id != null) }"/>
<form action="usersettings-save.do" method="post">
<input type="hidden" name="id" value="${entity.id }"/>
<h3 class="page-header">User Settings</h3>
<table>
    <tr>
            <td>Username</td>
            <td>:</td>
            <td>
                    ${entity.username}
            </td>
    </tr>
    <tr>
            <td colspan="3" align="left"><input type="button" class="btn btn-mini
btn-primary" value="Change Password"
onclick="showChangePasswordModal(${entity.id});"/></td>
```

```
    </tr>
    <tr>
            <td colspan="3"><h3>Personal information</h3></td>
    </tr>
    <tr>
            <td>First Name</td>
            <td>:</td>
            <td><input type="text" class="form-control" name="entity.firstName"
value="${(isUpdate == true)?entity.firstName : '' }"/></td>
    </tr>
    <tr>
            <td>Last Name</td>
            <td>:</td>
            <td><input type="text" class="form-control" name="entity.lastName"
value="${(isUpdate == true)?entity.lastName : '' }"/></td>
    </tr>
    <tr>
            <td colspan="3" align="right">
                    <input type="submit" class="btn btn-mini btn-primary"
value="Save"/>
            </td>
    </tr>
</table>
</form>

<div id="passwordModal">

</div>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/includes/usersettings.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="usersettings"/>
<div class="container">
<%@include file="../includes/nav.jsp" %>
<script type='text/javascript' src='<c:url
value="scripts/ModalChangePassword.js"/>'></script>
<c:set var="isUpdate" value="${(entity.id != null) }"/>
<form action="usersettings-save.do" method="post">
<input type="hidden" name="id" value="${entity.id }"/>
<h3 class="page-header">User Settings</h3>
<table>
    <tr>
            <td>Username</td>
            <td>:</td>
            <td>
                    ${entity.username}
            </td>
    </tr>
    <tr>
            <td colspan="3" align="left"><input type="button" class="btn btn-mini
btn-primary" value="Change Password"
onclick="showChangePasswordModal(${entity.id});"/></td>
    </tr>
    <tr>
            <td colspan="3"><h3>Personal information</h3></td>
    </tr>
    <tr>
            <td>First Name</td>
            <td>:</td>
            <td><input type="text" class="form-control" name="entity.firstName"
value="${(isUpdate == true)?entity.firstName : '' }"/></td>
    </tr>
    <tr>
            <td>Last Name</td>
            <td>:</td>
            <td><input type="text" class="form-control" name="entity.lastName"
value="${(isUpdate == true)?entity.lastName : '' }"/></td>
    </tr>
    <tr>
            <td colspan="3" align="right">
                    <input type="submit" class="btn btn-mini btn-primary"
value="Save"/>
            </td>
```

```
    </tr>
</table>
</form>

<div id="passwordModal">

</div>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/login/accessdenied.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>PNP Warrant</title>
<%@taglib uri="/struts-tags" prefix="s" %>
<link href="css/bootstrap/bootstrap-theme.css" rel="stylesheet" type="text/css"/>
<link href="css/bootstrap/bootstrap.css" rel="stylesheet" type="text/css"/>
<link href="css/login.css" rel="stylesheet" type="text/css"/>

<link rel="icon"
      type="image/png"
      href="images/fav1.png">

<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/bootstrap/bootstrap.js" type="text/javascript"></script>
</head>
<body>

    <div class="container">

                    <input type="hidden" name="event" value="search"/>
            <div class="contentError">

                    <div class="clear"></div>
                    <div class="errorBox" align="center">
                            <h3><img src="images/iDenied.png" align="absmiddle"
/>  Access is Denied!</h3>
                    </div>

                      <p align="center">
                      <input class="button" type="button" value="Back to Main"
onClick="location.href = 'warrant.do'" class="btnLogin" /></a>
                  </p>
                    </div>
            </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/login/error.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>WEB PCS</title>
<%@taglib uri="/struts-tags" prefix="s" %>
<link href="css/bootstrap/bootstrap-theme.css" rel="stylesheet" type="text/css"/>
<link href="css/bootstrap/bootstrap.css" rel="stylesheet" type="text/css"/>
<link href="css/login.css" rel="stylesheet" type="text/css"/>

<link rel="icon"
      type="image/png"
      href="images/fav1.png">

<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/bootstrap/bootstrap.js" type="text/javascript"></script>
</head>
```

```
<body>

    <div class="container">

                    <input type="hidden" name="event" value="search"/>
            <div class="contentError">

                    <div class="clear"></div>
                    <div class="errorBox">
                            <h3><img src="images/iDenied.png" align="absmiddle"
/>  Error - Please contact administrator</h3>
                    </div>

                    <div class="page">
                            <table cellpadding="0" cellspacing="0"   width="96%">
                    <tbody align="left">
                            <tr>
                                    <td><s:property value="exception"/></td>
                            </tr>
                            <tr>
                                    <td><s:property value="exceptionStack"/></td>
                            </tr>
                    </tbody>
                    </table>
                    </div>

                      <p align="center">
                      <input class="button" type="button" value="Back to Login"
onClick="location.href = 'login.do'" class="btnLogin" /></a>
                    </p>
            </div>
    </div>
</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/login/forgotpassword-form.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>Web PCS</title>
<%@taglib uri="/struts-tags" prefix="s" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 <link href="css/bootstrap/bootstrap-acrulian-theme.css" rel="stylesheet"
type="text/css"/>
 <link href="css/bootstrap/default-dashboard-template.css" rel="stylesheet"
type="text/css"/>
<link href="css/login.css" rel="stylesheet" type="text/css"/>
 <link href="css/chosen.min.css" rel="stylesheet" type="text/css"/>

 <link rel="icon"
      type="image/png"
      href="images/fav1.png">

<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/bootstrap/bootstrap.js" type="text/javascript"></script>
<script src="scripts/chosen.jquery.min.js" type="text/javascript"></script>

<style>
body {
    background: #E6E6FA;
 }
</style>
</head>
<body>
 <div class="container-fluid" align="center">
    <div class="container">

 <div align="center" style="background-color: #E6E6FA;" ><a href="login.do"><h2><img
alt="WEB PCS" src="images/oie_transparent.png"/></h2></a>
   <div class="bs-component">
    <div class="well"> <br>
```

```
     <form action="forgotpassword-validate.do" method="post"  class="form-signin"
role="form">

            <%@include file="../includes/errors.jsp" %>

        <input type="text" name="email" class="form-control" placeholder="Email"
value="" required autofocus>
        <h4></h4><input type="text" name="medicalLicense" class="form-control"
placeholder="Medical License" value="" required>
        <h4></h4><button class="btn btn-lg btn-primary btn-block" type="submit">Renew
Password</button>
        <br/>
      </form>
</div>
</div>

</div>
    </div>
</div>
    <form action="login.do" method="post" name="loginForm">
</form>

</body>
</html>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/login/login.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>Web PCS</title>
<%@taglib uri="/struts-tags" prefix="s" %>
<link href="css/bootstrap/bootstrap-theme.css" rel="stylesheet" type="text/css"/>
<link href="css/bootstrap/bootstrap.css" rel="stylesheet" type="text/css"/>
<link href="css/login.css" rel="stylesheet" type="text/css"/>

<link rel="icon"
      type="image/png"
      href="images/fav1.png">

<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/bootstrap/bootstrap.js" type="text/javascript"></script>

<style>
body {
    background: url("images/Background1.jpg");
    background-repeat: no-repeat;
    background-size:cover;
    padding-top:15px;
    padding-left:15px;
 }

div.bgimage{
position: fixed;
padding-top:495px;
padding-left:1070px;

}
a.qqq{
color:#F5F5F5;
padding-top:3px;
font-size: 11px;

}
div.join>a.qqq:hover{
color: #262626;
  text-decoration: none;

}

div.forgot>a.qqq:hover{
color: #262626;
```

```
  text-decoration: none;
}

.btn-login{
position:absolute;
top:15px;
left:513px;
width:5%;
}

.btn-join{
position:absolute;
top:15px;
left:595px;
width:6%;
}

</style>
</head>

 <body>

       <form action="login.do" method="post" name="loginForm">

       <input type="text" name="username" class="form-signin" size=30
placeholder="Username" value="${username }" required autofocus>
       <input type="password" class="form-signin" name="password" size=30
placeholder="Password" value="${password }" required>
        <button class="btn btn-primary btn-login "  type="submit">Log In</button>
         <button class="btn btn-primary btn-login "  type="submit">Log In</button>
        <h5></h5>
         <div class="join"><a href="registration.do" class="btn btn-primary btn-join"
role="button">Join Now</a>
          <div class="forgot"><a class="qqq" href="forgotpassword.do">Forgot
Password?</a></div>
      </form>
      <br>
</body>
<div class="bgimage">
<img alt="WEB PCS" src="images/oie_transparent.png" width=390 height=110/>
 </div>

</html>

<%@include file="../includes/errors.jsp" %>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/login/register-form.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>Web PCS</title>
<%@taglib uri="/struts-tags" prefix="s" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 <link href="css/bootstrap/bootstrap-acrulian-theme.css" rel="stylesheet"
type="text/css"/>
 <link href="css/bootstrap/default-dashboard-template.css" rel="stylesheet"
type="text/css"/>
<link href="css/login.css" rel="stylesheet" type="text/css"/>
 <link href="css/chosen.min.css" rel="stylesheet" type="text/css"/>

 <link rel="icon"
      type="image/png"
      href="images/fav1.png">
<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/bootstrap/bootstrap.js" type="text/javascript"></script>

<script src="scripts/chosen.jquery.min.js" type="text/javascript"></script>
<style>
body {
    background: #E6E6FA;
 }
```

```
</style>
</head>
<body >
<br>
    <div class="container-fluid">
    <div class="container">

    <div align="center" style="background-color: #E6E6FA;" ><a href="login.do"><img
alt="WEB PCS" src="images/oie_transparent.png"/></a>
    <div class="bs-component"><br>
    <div class="well">
     <form action="register-save.do" method="post"  class="form-signin" role="form"
onsubmit="return verifySubmit();">

            <%@include file="../includes/errors.jsp" %>
        <h5>Name</h5>
        <input type="text" name="firstName" class="form-control" placeholder="First"
required autofocus/>
        <h4></h4><input type="text" name="lastName" class="form-control"
placeholder="Last" required/>

        <h5>Medical License</h5>
        <input type="text" id="medicalLicense" maxlength="10" name="medicalLicense"
class="form-control" required/>

        <h5>Email Address</h5>
        <input type="email" name="email" class="form-control" required/>

        <h5>Cellphone Number</h5>
        <input type="text" id="mobileNumber"name="mobileNumber" class="form-control"
placeholder="format: 09XXXXXXXXX" required/>

        <h5>Specialty</h5>
        <select id="specialty" class="chosen-select form-control" multiple
name="specialtyIds" data-placeholder="Choose specialties...">
        <c:forEach items="${specialtyList}"  var="specialty">
            <option value="${specialty.id }">${specialty.description }</option>
        </c:forEach>
        </select>

        <h5>Create Password</h5>
        <input type="password" id="password" name="password" class="form-control"
required/>

        <h5>Confirm Password</h5>
        <input type="password" id="confirmPassword" name="confirmPassword"
class="form-control" required/>
        <h3></h3>
        <button class="btn btn-lg btn-primary btn-block"
type="submit">Register</button>
      </form>
      </div>
      </div>
    </div>
    </div>
    </div>
  </body>

<script type="text/javascript">

    jQuery(document).ready(function(){
            jQuery(".chosen-select").chosen({disable_search_threshold: 10});
    });

    jQuery("#mobileNumber").keypress(function (e) {
         if (e.which != 8 && e.which != 0 && (e.which < 48 || e.which > 57)) {
           return false
        }
      });

    jQuery("#medicalLicense").keypress(function (e) {
         if (e.which != 8 && e.which != 0 && (e.which < 48 || e.which > 57)) {
           return false
        }
      });
```

```
    function verifySubmit(){
            var password = jQuery('#password').val();
            var confirmPassword = jQuery('#confirmPassword').val();
            if(password != confirmPassword){
                    alert('Passwords do not match.');
                    return false;
            }

            var value = Number(jQuery('#medicalLicense').val());
            if (Math.floor(value) != value) {
                    alert('Medical License must be numeric.');
                    return false;
            }

            if(jQuery('[name="specialtyIds"]').val() == null){
                    alert("There should be at least 1 value supplied in specialty
field.")
                    return false;
            }

            var reg = /^\d{11}$/;
            if(jQuery('#mobileNumber').val().length > 0 &&
!reg.test(jQuery('#mobileNumber').val())){
                    alert("Cellphone Number must be 11 numeric characters.");
                    return false;
            }

            return true;
    }
</script>
</html>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/includes/sidebar.jsp**

```
<div class="col-sm-3 col-md-2 sidebar">
  <ul class="nav nav-sidebar">

    <li ${subnav eq 'users' ? 'class="active"' : ''}><a
href="users.do">Users</a></li>
    <li ${subnav eq 'physicians' ? 'class="active"' : '' }><a
href="physicians.do">Physicians</a></li>
    <li ${subnav eq 'newphysicians' ? 'class="active"' : ''}><a
href="newphysicians.do">New Physicians</a></li>
    <li ${subnav eq 'references' ? 'class="active"' : ''}><a
href="references.do">References</a></li>
   </ul>
   </div>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/newphysicians/newphysicians.jsp**

```
<%@include file="../../includes/header.jsp"%>
<c:set var="nav" value="manage"/>
<c:set var="subnav" value="newphysicians"/>
<%@include file="../../includes/nav.jsp" %>
<div class="container-fluid">
<div class="row">
<%@include file="../includes/sidebar.jsp" %>
    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
  <h1 class="page-header">New Physicians</h1>
    <%@include file="../../includes/errors.jsp" %>
    <div class="table-responsive">
    <table id="entityTable"></table>
    <div id="entityTablePaging"></div>
    </div>
</div>
</div>
</div>

<div id="confirmation" style="display: none;">
    <p>Activate account for this physician?</p>
</div>
<script type="text/javascript">
var lastSelectedId;
    function myelem (value, options) {
```

```
      var el = document.createElement("input");
      el.type="text";
      el.value = value;
      el.readonly = true;
      return el;
    }

    function jQMyelem (value, options) {
              return jQuery('<input type="text" value="'+value+'" name="' +
options.name +'" readonly/>');
            }

    function myvalue(elem, operation, value) {
        if(operation === 'get') {
           return $(elem).val();
        } else if(operation === 'set') {
           $('input',elem).val(value);
        }
    }

    jQuery("#entityTable").jqGrid({
            url: "newphysicians-retrieve.do",
            datatype: "json",
            height: 400,
            autowidth: true,
            colNames:['Username', 'Medical License', 'Last Name', 'First Name',
'Specialties'],
            colModel:[
                    {name:'username',index:'username', width:150, align:"center",
editable:false, edittype:'custom', editoptions:{custom_element: jQMyelem,
custom_value:myvalue}},
                    {name:'medLicenseId', index:'medLicenseId', width:100,
align:"center", editable:false, edittype: 'text'},
                    {name:'lastName',index:'lastName', width:100, align:"center",
editable:false, edittype: 'text'},
                    {name:'firstName',index:'firstName', width:100, align:"center" ,
editable:false, edittype: 'text'},
                    {name:'specialties', index:'specialties', width:250}
            ],
            rowNum:5,
            rowList:[5,10,15],
            pager: '#entityTablePaging',
            multisort: true,
            sortname: "id",
        sortorder: "desc",
        viewrecords: true,
            caption: "Users",
            onSelectRow: function(id){

                    jQuery("#confirmation").dialog({
                      buttons: {
                        "Yes": function() {
                            jQuery.get( "newphysicians-activate.do", { id: id} )
                    .done(function( data ) {
                            alert( "Succesfully activated physician.");
                            jQuery("#entityTable").trigger("reloadGrid");

                    });
                            jQuery(this).dialog("close");
                        },
                        "No": function() {
                            jQuery.get( "newphysicians-deactivate.do", {id: id } ,
function( data ){
                                    alert("Successfully deleted physician.")
                                    jQuery("#entityTable").trigger("reloadGrid");

                            });
                            jQuery(this).dialog("close");
                        },
                        "Cancel": function(){jQuery(this).dialog("close");}
                      }
                    });
         }
    })
```

```
    .jqGrid('navGrid','#entityTablePaging',{edit:false,add:false,del:false,search:fal
se},{height: 220,
            closeAfterEdit: true,resize: false,drag:
false,closeOnEscape:true,viewPagerButtons:false});

    function onSaveSuccess(xhr)
    {response = xhr.responseText; if(response == 1) return true; return false;}

    </script>
<%@include file="../../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/physicians/physicians-form.jsp**

```
<c:set var="isUpdate" value="${(entity.id != null) }"/>
<form action="physicians-save.do" method="post" onsubmit="return validateSubmit();">
<input type="hidden" name="id" value="${entity.id }">
<table>
    <tr>
            <td>Name</td>
            <td> : </td>
        <td><input type="text" name="entity.firstName" class="form-control"
value="${isUpdate? entity.firstName: '' }" placeholder="" required />
        <h4></h4><input type="text" name="entity.lastName" class="form-control"
value="${isUpdate? entity.lastName: '' }" placeholder="" required/></td>
    </tr>

    <tr>
    <td><h4></h4>Medical License</td>
    <td><h4></h4>:</td>
    <td><h4></h4><input type="text" id="medLicenseId" maxlength="10"
name="entity.medLicenseId" value="${isUpdate? entity.medLicenseId: '' }" class="form-
control" required/></td>
    </tr>
    <tr>
    <td><h4></h4>Email Address</td>
    <td><h4></h4>:</td>
    <td>
            <c:if test="${isUpdate }">
            ${entity.email}
            </c:if>
            <c:if test="${!isUpdate }">
                    <h4></h4><input type="email" name="entity.email"
value="${isUpdate? entity.email: '' }" class="form-control" required/>
            </c:if>
    </td>
    </tr>
    <c:if test="${!isUpdate}">
    <tr>
            <td><h4></h4>Password</td>
            <td><h4></h4>:</td>
            <td><h4></h4><input type="password" class="form-control"
name="entity.password" value="${isUpdate? entity.password : ''}" required/></td>
    </tr>
    </c:if>
    <tr>
    <td><h4></h4>Cellphone Number</td>
    <td><h4></h4>:</td>
    <td><h4></h4><input type="text" id="mobileNumber"name="entity.cellphoneNumber"
value="${isUpdate? entity.cellphoneNumber: '' }" class="form-control" required/></td>
    </tr>
    <tr>
    <td><h4></h4>Specialty</td>
    <td><h4></h4>:</td>
    <td>
            <h4></h4><select id="specialty" class="form-control chosen-select"
multiple name="specialtyIds" data-placeholder="Choose specialties...">
            <c:forEach items="${specialtyList}"  var="specialty">
            <c:set var="isSelected" value=""/>
            <c:forEach items="${entity.specialties}" var="selectedSS">
                    <c:if test="${selectedSS.id == specialty.id}">
                            <c:set var="isSelected" value="selected"/>
                    </c:if>
            </c:forEach>
```

```
            <option value="${specialty.id }" ${isSelected }>${specialty.name
}</option>
            </c:forEach>
            </select>
        </td>
    </tr>
    <tr>
    <td colspan="3"><h4></h4><h4></h4><input type="checkbox" name="isActivited"
value="true" ${entity.isActivited ? "checked" : "" }> Is Activated?</td>
    </tr>

    <tr>
    <td colspan="3">
    <div class="panel panel-default">
    <div class="panel-heading">
    EMAIL
    </div>
    <div class="panel-body">
            <input type="checkbox" value="true" name="sendWhenBooked"
${entity.sendWhenBooked ? 'checked' : '' }/>An appointment was booked <br/>
            <input type="checkbox" value="true" name="sendWhenCancelled"
${entity.sendWhenCancelled ? 'checked' : '' }/>An appointment was cancelled<br/>
            <input type="checkbox" value="true" name="sendWhenResceduled"
${entity.sendWhenResceduled ? 'checked' : '' }/>An appointment was rescheduled<br/>
            <input type="checkbox" value="true" name="sendWhenMessageSent"
${entity.sendWhenMessageSent ? 'checked' : '' }/>A message was sent
    </div>
    </div>

     <div class="panel panel-default">
    <div class="panel-heading">
    FORUM
    </div>
    <div class="panel-body">
            <input type="checkbox" value="true" name="sendOnForumReply"
${entity.sendOnForumReply ? 'checked' : '' }/>Someone replied to my forum<br/>
            <input type="checkbox" value="true" name="sendOnForumSubscribed"
${entity.sendOnForumSubscribed ? 'checked' : '' }/>Someone subscribed to my forum
    </div>
    </div>

    <div class="panel panel-default">
    <div class="panel-heading">
    SUBSCRIPTION
    </div>
    <div class="panel-body">
            <input type="checkbox" value="true" name="sendSubscriptionUpdates"
${entity.sendSubscriptionUpdates ? 'checked' : '' }/>Updates about the forum I
follow<br/>
            <select name="subscriptionDigestType" style="display: none;">
                    <option value="DAILY">Send as daily digest</option>
                    <option value="WEEKLY">Send as weekly digest</option>
            </select>
    </div>
    </div>
    </td>
    </tr>
    <tr>
            <td colspan="3" align="right">
                    <input class="btn btn-mini btn-primary" type="submit"
value="Save"/>
                    <c:if test="${!isUpdate }">
                    <input class="btn btn-mini btn-primary" type="button"
id="btnClose" value="Cancel"/>
                    </c:if>
            </td>
    </tr>
</table>
</form>
<c:if test="${isUpdate }">
<form action="physicians-changePassword.do" method="post" onsubmit="return
checkPasswordValidity()">
<input type="hidden" name="id" value="${entity.id }">
<table>
<tr>
    <td>
```

```
            New Password
    </td>
    <td> : </td>
    <td>
            <input type="password" id="password" name="password" class="form-control"
required/><h4></h4>
    </td>
    </tr>
    <tr>
    <td>Confirm Password</td>
    <td> : </td>
        <td><input type="password" id="confirmPassword" name="confirmPassword"
class="form-control" required/></td>
</tr>
<tr>
    <td colspan="3"><input class="btn btn-mini btn-primary" type="submit"
value="Change Password"></td>
</tr>
</table>
</form>
</c:if>

<script type="text/javascript">
jQuery(document).ready(function(){
    jQuery(".chosen-select").chosen({disable_search_threshold: 10});
});

function checkPasswordValidity(){
    var password = jQuery('#password').val();
    var confirmPassword = jQuery('#confirmPassword').val();
    if(password != confirmPassword){
            alert('Passwords do not match.');
            return false;
    }
    return true
}

function validateSubmit(){

    if(jQuery('[name="specialtyIds"]').val() == null){
            alert("There should be at least 1 value supplied in specialty field.")
            return false;
    }

    var reg = /^\d{11}$/;
    if(jQuery('#mobileNumber').val().length > 0 &&
!reg.test(jQuery('#mobileNumber').val())){
            alert("Cellphone Number must be 11 numeric characters");
            return false;
    }

    return true;
}

jQuery("#mobileNumber").keypress(function (e) {
    if (e.which != 8 && e.which != 0 && (e.which < 48 || e.which > 57)) {
      return false
   }
 });

jQuery("#medLicenseId").keypress(function (e) {
    if (e.which != 8 && e.which != 0 && (e.which < 48 || e.which > 57)) {
      return false
   }
 });

</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/physicians/physicians.jsp**

```
<%@include file="../../includes/header.jsp"%>
<c:set var="nav" value="manage"/>
<c:set var="subnav" value="physicians"/>
<%@include file="../../includes/nav.jsp" %>
<div class="container-fluid">
<div class="row">
```

```
<%@include file="../includes/sidebar.jsp" %>
    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
    <h1 class="page-header">Physicians</h1>
    <%@include file="../../includes/errors.jsp" %>
    <c:if test="${entity.id == null }">
    <div class="table-responsive">
    <table id="entityTable"></table>
    <div id="entityTablePaging"></div>
    </div>
    </c:if>

    <br/>
    <input id="btn" class="btn btn-mini btn-primary" type="button" value="Add new
physician" />

    <div id="entityform" >
    <%@include file="physicians-form.jsp"%>
    </div>

    </div>
</div>
</div>
<script type="text/javascript">
var lastSelectedId;

    <c:if test="${(entity.id == null) }">
            jQuery("#entityform").hide();
    </c:if>
    <c:if test="${(entity.id != null) }">
            jQuery("#btn").hide();
    </c:if>

    jQuery("#btn").click(function(){
            jQuery("#btn").hide();
            jQuery("#entityform").show();
    });

    jQuery("#btnClose").click(function(){
            jQuery("#btn").show();
            jQuery("#entityform").hide();
    });

    function myelem (value, options) {
      var el = document.createElement("input");
      el.type="text";
      el.value = value;
      el.readonly = true;
      return el;
    }

    function jQMyelem (value, options) {
              return jQuery('<input type="text" value="'+value+'" name="' +
options.name +'" readonly/>');
            }

    function myvalue(elem, operation, value) {
        if(operation === 'get') {
           return $(elem).val();
        } else if(operation === 'set') {
           $('input',elem).val(value);
        }
    }
    <c:if test="${entity.id == null }">
    jQuery("#entityTable").jqGrid({
            url: "physicians-retrieve.do",
            datatype: "json",
            height: 250,
            autowidth: true,
            colNames:['Username', 'Last Name', 'First Name'],
            colModel:[
                    {name:'username',index:'username', width:150, align:"center",
editable:false, edittype:'custom', editoptions:{custom_element: jQMyelem,
custom_value:myvalue}},
                    {name:'lastName',index:'lastName', width:100, align:"center",
editable:true, edittype: 'text'},
```

```
                    {name:'firstName',index:'firstName', width:100, align:"center" ,
editable:true, edittype: 'text'},

            ],
            rowNum:5,
            rowList:[5,10,15],
            pager: '#entityTablePaging',
            multisort: true,
            sortname: "id",
        sortorder: "desc",
        viewrecords: true,
            caption: "Physicians",
            onSelectRow: function(id){
                    window.location = "physicians.do?id="+id;
         }
    })

    .jqGrid('navGrid','#entityTablePaging',{edit:false,add:false,del:false,search:fal
se},{height: 220,
            closeAfterEdit: true,resize: false,drag:
false,closeOnEscape:true,viewPagerButtons:false});

    function onSaveSuccess(xhr)
    {response = xhr.responseText; if(response == 1) return true; return false;}
    </c:if>
    </script>
<%@include file="../../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/references/loadreferences.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:forEach items="${ referencePage.content}" var="reference">
    <div  class="well bs-component" style="margin-right:
15px;"><p>${reference.description}</p><a href="${reference.url}">${reference.url
}</a></div>
</c:forEach>
<c:if test="${referencePage.number + 1 < referencePage.totalPages }">
    <div class="well bs-component" style="margin-right: 15px;"><a class="jscroll-
next" href="references-loadreferences.do?page=${referencePage.number +
2}">next</a></div>
</c:if>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/references/references-form.jsp**

```
<c:set var="isUpdate" value="${(entity.id != null) }"/>
<form action="references-save.do" method="post">
<table>
    <tr>
            <td>Title</td>
            <td> : </td>
            <td><input type="text" class="form-control" name="entity.description"
required="required" value="${isUpdate?entity.description : '' }"/></td>
    </tr>
    <tr>
            <td>Url</td>
            <td> : </td>
            <td> <h4></h4><input type="url" class="form-control" id="url"
name="entity.url" required="required" value="${isUpdate?entity.url : '' }"/></td>
    </tr>
    <tr>
            <td colspan="3" align="right">
                    <h4></h4><input class="btn btn-mini btn-primary" type="submit"
value="Save"/>
                    <c:if test="${!isUpdate }">
                    <input class="btn btn-mini btn-primary" type="button"
id="btnClose" value="Cancel"/>
                    </c:if>
            </td>
    </tr>
</table>
<c:if test="${!isUpdate }">
<script type="text/javascript">
    jQuery("#url").val(' ');
</script>
</c:if>
</form>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/references/references.jsp**

```
<%@include file="../../includes/header.jsp"%>
<c:set var="nav" value="manage"/>
<c:set var="subnav" value="references"/>
<%@include file="../../includes/nav.jsp" %>
<div class="container-fluid">
<div class="row">
<%@include file="../includes/sidebar.jsp" %>
    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
    <h1 class="page-header">References</h1>
    <%@include file="../../includes/errors.jsp" %>
        <div class="table-responsive">
            <table id="entityTable"></table>
            <div id="entityTablePaging"></div>
            </div> <br>
            <input id="btn" class="btn btn-mini btn-primary" type="button" value="Add
new reference" />

            <div id="entityform" >
            <%@include file="references-form.jsp"%>
            </div>
    </div>

</div>
</div>
<script type="text/javascript">
var lastSelectedId;

    <c:if test="${(entity.id == null) }">
            jQuery("#entityform").hide();
    </c:if>
    <c:if test="${(entity.id != null) }">
            jQuery("#btn").hide();
    </c:if>

    jQuery("#btn").click(function(){
            jQuery("#btn").hide();
            jQuery("#entityform").show();
    });

    jQuery("#btnClose").click(function(){
            jQuery("#btn").show();
            jQuery("#entityform").hide();
    });

    function myelem (value, options) {
      var el = document.createElement("input");
      el.type="text";
      el.value = value;
      el.readonly = true;
      return el;
    }

    function jQMyelem (value, options) {
              return jQuery('<input type="text" value="'+value+'" name="' +
options.name +'" readonly/>');
            }

    function myvalue(elem, operation, value) {
        if(operation === 'get') {
           return $(elem).val();
        } else if(operation === 'set') {
           $('input',elem).val(value);
        }
    }

        jQuery("#entityTable").jqGrid({
                    url: "references-retrieve.do",
                    datatype: "json",
                    height: 250,
                    autowidth: true,
                    colNames:['Description', 'Url'],
                    colModel:[
                            {name:'description',index:'description', width:100,
align:"center", editable:true, edittype: 'text'},
```

```
                            {name:'url',index:'url', width:80, align:"center" ,
editable:true, edittype: 'text'},
                    ],
                    rowNum:5,
                    rowList:[5,10,15],
                    pager: '#entityTablePaging',
                    multisort: true,
                    sortname: "id",
                sortorder: "desc",
                viewrecords: true,
                    caption: "References",
                    editurl: "references-save.do",
            })

    .jqGrid('navGrid','#entityTablePaging',{edit:true,add:false,del:true,search:false
},{height: 220,
                closeAfterEdit: true,resize: false,drag:
false,closeOnEscape:true,viewPagerButtons:false});

            function onSaveSuccess(xhr)
            {response = xhr.responseText; if(response == 1) return true; return
false;}
    </script>
    <%@include file="../../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/user/users-form.jsp**

```
<c:set var="isUpdate" value="${(entity.id != null) }"/>
<form action="users-save.do" method="post">
<table>
    <tr>
            <td><h4></h4>Username</td>
            <td><h4></h4>   :  </td>
            <td>
                    <c:choose>
                            <c:when test="${isUpdate }">
                                    ${entity.username}
                            </c:when>
                            <c:otherwise>
                                    <h4></h4><input  class="form-control"
type="text" name="entity.username" required/>
                            </c:otherwise>
                    </c:choose>
            </td>
    </tr>
    <tr>
            <td><h4></h4>Password</td>
            <td><h4></h4>  :</td>
            <td><h4></h4><input type="password" class="form-control"
name="entity.password" value="${isUpdate? entity.password : ''}" required/></td>
    </tr>
    <tr>
            <td><h4></h4>First Name</td>
            <td><h4></h4>  :</td>
            <td><h4></h4><input type="text" class="form-control"
name="entity.firstName" value="${isUpdate?entity.firstName : '' }" required/></td>
    </tr>
    <tr>
            <td><h4></h4>Last Name</td>
            <td><h4></h4>  :</td>
            <td><h4></h4><input type="text" class="form-control"
name="entity.lastName" value="${isUpdate?entity.lastName : '' }" required/></td>
    </tr>
    <tr>
            <td colspan="3" align="right">
                    <h4></h4><input class="btn btn-mini btn-primary" type="submit"
value="Save"/>
                    <c:if test="${!isUpdate }">
                    <input class="btn btn-mini btn-primary" type="button"
id="btnClose" value="Cancel"/>
                    </c:if>
            </td>
    </tr>
</table>
</form>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/manage/user/users.jsp**

```
<%@include file="../../includes/header.jsp"%>
<c:set var="nav" value="manage"/>
<c:set var="subnav" value="users"/>
<%@include file="../../includes/nav.jsp" %>
<div class="container-fluid">
<div class="row">
<%@include file="../includes/sidebar.jsp" %>
    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
    <h1 class="page-header">User</h1>
    <%@include file="../../includes/errors.jsp" %>
    <div class="table-responsive">
    <table id="entityTable"></table>
    <div id="entityTablePaging"></div>
    </div>

    <br/>
    <input id="btn" class="btn btn-mini btn-primary" type="button" value="Add new
user" />

    <div id="entityform" >
            <%@include file="users-form.jsp"%>
    </div>

    </div>
</div>

<script type="text/javascript">
var lastSelectedId;
    <c:if test="${(entity.id == null) }">
            jQuery("#entityform").hide();
    </c:if>
    <c:if test="${(entity.id != null) }">
            jQuery("#btn").hide();
    </c:if>

    jQuery("#btn").click(function(){
            jQuery("#btn").hide();
            jQuery("#entityform").show();
    });

    jQuery("#btnClose").click(function(){
            jQuery("#btn").show();
            jQuery("#entityform").hide();
    });

    function myelem (value, options) {
      var el = document.createElement("input");
      el.type="text";
      el.value = value;
      el.readonly = true;
      return el;
    }

    function jQMyelem (value, options) {
              return jQuery('<input type="text" value="'+value+'" name="' +
options.name +'" readonly/>');
            }
            function myvalue(elem, operation, value) {
        if(operation === 'get') {
           return $(elem).val();
        } else if(operation === 'set') {
           $('input',elem).val(value);
        }
    }
    jQuery("#entityTable").jqGrid({
            url: "users-retrieve.do",
            datatype: "json",
            height: 250,
            autowidth: true,
            colNames:['Username', 'Last Name', 'First Name'],
            colModel:[
                    {name:'username',index:'username', width:150, align:"center",
editable:false, edittype:'custom', editoptions:{custom_element: jQMyelem,
custom_value:myvalue}},
```

```
                    {name:'lastName',index:'lastName', width:100, align:"center",
editable:true, edittype: 'text'},
                    {name:'firstName',index:'firstName', width:80, align:"center" ,
editable:true, edittype: 'text'},
            ],
            rowNum:5,
            rowList:[5,10,15],
            pager: '#entityTablePaging',
            multisort: true,
            sortname: "id",
        sortorder: "desc",
        viewrecords: true,
            caption: "Users",
            editurl: "users-save.do",
    })
.jqGrid('navGrid','#entityTablePaging',{edit:true,add:false,del:true,search:false},{h
eight: 220,
           closeAfterEdit: true,resize: false,drag:
false,closeOnEscape:true,viewPagerButtons:false});

    function onSaveSuccess(xhr)
    {response = xhr.responseText; if(response == 1) return true; return false;}
</script>
<%@include file="../../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/planner/article.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="appointments" />
<%@include file="../includes/nav.jsp"%>
<div class="content" id="content">
    <div id="planner">
            <s:property escape="false" value="messageStore.planner" />
    </div>
</div>
<script type="text/javascript">
    jQuery(document).ready(function() {
            var appointmentId = '${param.appointmentId}';
            if(appointmentId != null && appointmentId != '') {
                    var triggerSmsReminder = <s:property
value="currentLoggedInUser.triggerSmsReminder"/>;
                    if(triggerSmsReminder) {
                            $.facebox(function() {
                            $.ajax({
                                url: "appointments-notify.do",
                                data: { 'appointmentId' : appointmentId},
                                type: 'get',
                                cache: false
                            })
                            .done(function(data){
                                $.facebox(data);
                            });
                        });
                    }
                    var url=document.location.href;
            var urlparts= url.split('?');
            window.history.pushState('',document.title,urlparts[0]);
            }
    });
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/planner/data.jsp**

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<s:property escape="false" value="messageStore.data" />
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/profile/allprofiles.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value=""/>
<%@include file="../includes/nav.jsp" %>
<div class="container-fluid">
<div class="container">
    <div id="physiciansTable" >
```

```
    </div>
    </div>
</div>

<script type="text/javascript">

jQuery(document).ready(function() {
    jQuery.ajax({
              url: 'profiles-loadphysicians.do',
              type: 'GET',
              dataType: 'html'
             }).done(function ( data ) {
                     jQuery("#physiciansTable").html(data);
    });

});

function updateTable(url){
    jQuery.ajax({
              url: url,
              type: 'GET',
              dataType: 'html'
             }).done(function ( data ) {
                     jQuery("#physiciansTable").html(data);
    });
}
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/profile/loadprofiles.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<div class="container-fluid"> <h4></h4>
    <div class="container">
    <h3><b>All Physicians</b></h3>
    <h3></h3>
<table class="table table-striped table-responsive">
   <tr>
    <th></th>
    <th>Name</th>
    <th>Specialty</th>
   </tr>
            <c:forEach items="${physicianPage.content }" var="physician">
      <tr>
       <td></td>
       <td><a href="profile-view.do?id=${physician.id }">${physician.fullName
}</a></td>
         <td>
            <c:forEach items="${physician.specialties}" var="specialty"
varStatus="specialtyVarStatus">
                                      ${specialty.description}<c:if
test="${!specialtyVarStatus.isLast()}">;</c:if>
                           </c:forEach>
         </td>
       </tr>
     </c:forEach>

</table>
</div>
</div>

<center>
<nav>
  <ul class="pagination">
    <li>
    <c:if test="${physicianPage.number - 1 >= 0}">
      <a class="jscroll-physicians-next" href="javascript:void(0)" aria-
label="Previous" onclick="updateTable('profiles-
loadphysicians.do?page=${physicianPage.number }')">
        <span aria-hidden="true">&laquo;</span>
      </a>
      </c:if>
    </li>
    <li>
                <c:forEach begin="1" end="${physicianPage.totalPages }" step="1"
var="counter" >
```

```
                    <a href="javascript:void(0)" onclick="updateTable('profiles-
loadphysicians.do?page=${counter}')">                                <c:choose>
                  <c:when test="${counter == physicianPage.number + 1}">
                                          <b>${counter}</b>
                  </c:when>
                  <c:otherwise>
                     ${counter }
                  </c:otherwise>
                 </c:choose>
                     </a>
              </c:forEach>
    </li>
    <li>
    <c:if test="${physicianPage.number + 1 < physicianPage.totalPages }">
      <a href="javascript:void(0)" aria-label="Next" onclick="updateTable('profiles-
loadphysicians.do?page=${physicianPage.number + 2}')">
        <span aria-hidden="true">&raquo;</span>
      </a>
      </c:if>
    </li>
  </ul>
</nav>
</center>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/profile/profile-edit.jsp**

```
<%@include file="../includes/header.jsp"%>

<c:set var="nav" value="profile"/>
<%@include file="../includes/nav.jsp" %>

<style type="text/css">
.panel
{
  overflow: visible;
}
</style>
<h3></h3>
<div class="container-fluid">
<div class="container">
    <div class="panel panel-default">
    <div class="panel-body">
    <form action="profile-save.do" method="post" enctype="multipart/form-data"
onsubmit="return verifySubmit();">
            <%@include file="../includes/errors.jsp" %>
            <div class="media" style="z-index: -1;">

              <div class="pull-left">
              <h5></h5>
              <center> <img src="${physician.imageUrl == null ? 'images/default-user-
image.png' : physician.imageUrl}" alt="User Image" style="height: 150px; width:
150px"></center>
              <h3></h3>
                    <input type="file" class="btn btn-sm btn-primary btn-block"
name="image" accept="image/gif, image/jpeg, image/jpg" />

                    <c:if test="${physician.imageUrl != null }">

                    <input type="button" class="btn btn-sm btn-primary btn-block"
value="remove" onclick="removeProfilePicture()"/>
                    </c:if>

              </div>
              <div id="physiciandetails" class="media-body">
                <div class="panel panel-default">
                    <div class="panel-heading" style="height: 250px">
                            <h4>${physician.email }</h4>
                        <div class="row">
                        <div class="col-sm-6 col-md-6">
                            <input type="text" name="firstName" class="form-control"
placeholder="First Name" value="${physician.firstName }" required="required"/>
                            </div>
                            <div class="col-sm-6 col-md-6">
                                    <input type="text" name="lastName" class="form-
control" placeholder="Last Name" value="${physician.lastName }" required="required"/>
                        </div>
```

```
                        </div>
                         <h3></h3>
                        <div class="row">
                        <div class="col-sm-6 col-md-6">
                            <%-- <input type="text" name="specialty" class="form-
control" placeholder="Specialty" value="${physician.specialty.description }"
required="required"/>  --%>
                                    <select id="specialties" class="form-control
chosen-select" name="specialtyIds" data-placeholder="Choose specialties..." multiple
>
                                    <c:forEach items="${specialtyList}"
var="specialty">
                                            <c:set var="isSelected" value=""/>
                                            <c:forEach
items="${physician.specialtyIds }" var="sid">
                                                    <c:if test="${sid ==
specialty.id }">
                                                            <c:set var="isSelected"
value="true"/>
                                                    </c:if>
                                            </c:forEach>
                                            <c:choose>
                                                    <c:when test="${isSelected}">
                                                            <option
value="${specialty.id }" selected >${specialty.description }</option>
                                                    </c:when>
                                                    <c:otherwise>
                                                            <option
value="${specialty.id }" >${specialty.description }</option>
                                                    </c:otherwise>
                                            </c:choose>
                                </c:forEach>
                                </select>
                                </div>

                                <div class="col-sm-6 col-md-6">
                                <input type="text" id="cellphoneNumber"
name="cellphoneNumber" class="form-control" placeholder="Cellphone Number"
value="${physician.cellphoneNumber }" required="required"/>
                                </div>
                                </div>
                                <h3></h3>
                                <div class="row">
                                <div class="col-sm-6 col-md-6">
                                        ${physician.medLicenseId }
                                </div>
                                </div>
                        </div>
                        </div>
                  </div>
                </div>


                <div class="panel panel-default" style="z-index: 100">
                <div class="panel-heading">
                   Clinic Information
                </div>
                <div class="panel-body"><h4></h4>
                        <c:set var="totalRows" value="${fn:length(physician.clinics)
}"/>
                        <c:forEach items="${physician.clinics }" var="clinic"
varStatus="counter">
                        <div id="field${counter.count }">
                                <div class="col-sm-4 col-md-4">
                                    <input type="text" name="clinicName" class="form-
control" placeholder="Clinic Name" value="${clinic.name }" required="required"/>
                                </div>
                                <div class="col-sm-4 col-md-4">
                                    <input type="text" name="clinicAddress" class="form-
control" placeholder="Clinic Address" value="${clinic.address }"
required="required"/>
                                </div>
                                <div class="col-sm-3 col-md-3">
                                    <input type="text" name="clinicTelNo" class="form-
control" placeholder="Clinic Landline" value="${clinic.telNo }" required="required"/>
                                </div>
```

```
                            <button id="remove${counter.count }" class="btn btn-
danger remove-me" >-</button></div>  <h4></h4>
                    </c:forEach>
                    <c:if test="${totalRows == 0 }">
                    <div id="field${totalRows + 1 }" >
                            <div class="col-sm-4 col-md-4">
                                <input type="text" name="clinicName" class="form-
control" placeholder="Clinic Name" value="" required="required"/>
                            </div>
                            <div class="col-sm-4 col-md-4">
                                <input type="text" name="clinicAddress" class="form-
control" placeholder="Clinic Address" value="" required="required"/>
                            </div>
                            <div class="col-sm-3 col-md-3">
                                <input type="text" name="clinicTelNo" class="form-
control" placeholder="Clinic Landline" value="" required="required"/>
                            </div>
                            <button id="remove${totalRows+1 }" class="btn btn-danger
remove-me" >-</button><h4></h4>
                    </div>
                    </c:if>
            </div>
            <div class="panel-footer">
                    <button type="button" class="add-more btn btn-default btn-sm">
                            +
                    </button>
            </div>
            </div>
            <div style="width: 100px; float: right;">
                    <button class="btn btn-sm btn-primary btn-block"
type="submit">Save</button>
            </div>
            <input type="hidden" id="count" value=""/>
    </form>
    </div>
    </div>

    <div class="panel panel-default" style="z-index: 100">
            <div class="panel-heading">
                    Change Password
            </div>
            <div class="panel-body">
            <div>
                    <form action="profile-changepassword.do" method="post"
onsubmit="return checkPasswordValidity()">
                    <input type="hidden" name="id" value="${currentLoggedInUser.id
}">
                    <table>
                    <tr>
                            <td>
                                    New Password
                            </td>
                            <td>   :  </td>
                            <td>
                                    <h5></h5><input type="password" id="password"
name="password" class="form-control" required/><h3></h3>
                            </td>

                        </tr>

                        <tr>

                            <td>Confirm Password</td>
                            <td>  :  </td>
                            <td><input type="password" id="confirmPassword"
name="confirmPassword" class="form-control" required/></td>
                    </tr>
                    <tr>
                            <td>
                              
                            </td>
                            <td>

                            </td>
                            <td>
```

```
                        </td>
                    </tr>
                    <tr>
                        <td colspan="3"><input class="btn btn-mini btn-primary"
type="submit" value="Change Password"></td>
                    </tr>
                    </table>
                    </form>
                </div>
            </div>
            </div>
</div>
</div>

<form action="profile-removeProfileImage.do" id="removeProfileImage" method="post">
    <input type="hidden" name="id" value="${physician.id }"/>
</form>

<script type="text/javascript">

jQuery(document).ready(function(){
    jQuery(".chosen-select").chosen({disable_search_threshold: 3});
    jQuery('.remove-me').click(function(e){
        e.preventDefault();
        if(jQuery("div[id^='field']").length == 1){
                    return false;
            }
        var fieldNum = this.id.split('remove')[1];
        var fieldID = "#field" + fieldNum;
        jQuery(this).remove();
        jQuery(fieldID).remove();
        next =
parseInt(jQuery("div[id^='field']:last").attr("id").split("field")[1]);

    });

    function checkPasswordValidity(){
            var password = jQuery('#password').val();
            var confirmPassword = jQuery('#confirmPassword').val();
            if(password != confirmPassword){
                    alert('Passwords do not match.');
                    return false;
            }
            return true
    }

    var next = ${totalRows};
    jQuery('.add-more').click(function(e){
        e.preventDefault();
        var addto = "#field" + next ;
        var addRemove = "#field" + (next);
        next = next + 1;
        var newIn =
        '<div id="field' + next + '" >' +
            '<div class="col-sm-4 col-md-4">' +
        '<input type="text" name="clinicName" class="form-control"
placeholder="Clinic Name" value=""  required="required"/>' +
            '</div>' + '<h4></h4>'+
                    '<div class="col-sm-4 col-md-4">' +
                '<input type="text" name="clinicAddress" class="form-control"
placeholder="Clinic Address" value=""  required="required"/>' +
                    '</div> '+
                    '<div class="col-sm-3 col-md-3">' +
                '<input type="text" name="clinicTelNo" class="form-control"
placeholder="Clinic Landline" value=""  required="required"/> '+
                    '</div>  '+
                    '<button id="remove' + (next) + '" class="btn btn-danger remove-
me" >-</button>' +
            '</div> ';

        var newInput = jQuery(newIn);
        jQuery(addto).after(newInput);
        jQuery("#field" + next).attr('data-source',jQuery(addto).attr('data-
source'));
        jQuery("#count").val(next);
```

```
        jQuery('.remove-me').click(function(e){
            e.preventDefault();
                    if(jQuery("div[id^='field']").length == 1){
                            return false;
                    }
                var fieldNum = this.id.split('remove')[1];
                var fieldID = "#field" + fieldNum;
                jQuery(this).remove();
                jQuery(fieldID).remove();
                next =
parseInt(jQuery("div[id^='field']:last").attr("id").split("field")[1]);
            });
    });

});

function removeProfilePicture(){
    jQuery('#removeProfileImage').submit();
}

function verifySubmit(){

    var reg = /^\d{11}$/;
    if(jQuery('#cellphoneNumber').val().length > 0 &&
!reg.test(jQuery('#cellphoneNumber').val())){
            alert("Cellphone Number must be 11 numeric characters");
            return false;
    }

    var count = jQuery("#specialties :selected").length;
    if(count == 0){
            alert('Please select at least 1 item in specialties');
            return false;
    }

    return true;
}
</script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/profile/profile.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="${ physician.id == currentLoggedInUser.id ? 'profile' :
''}"/>
<%@include file="../includes/nav.jsp" %>
<h3></h3>
<div class="container-fluid">
<div class="container">
            <%@include file="../includes/errors.jsp" %>
            <div class="panel panel-default">
            <div class="panel-body">
            <div class="media">
              <a class="pull-left" href="${physician.id == currentLoggedInUser.id ?
'profile-edit.do' : 'javascript:void(0)' }">
                 <img src="${physician.imageUrl == null ? 'images/default-user-
image.png' : physician.imageUrl}" alt="User Image" style="height: 100px; width:
100px">
              </a>
              <div class="media-body">
                <div class="panel panel-default">
                    <div class="panel-heading" style="height: auto">
                            <h4>${physician.fullName }<c:if test="${physician.id ==
currentLoggedInUser.id }">
                                    <a href="profile-edit.do"><span class="glyphicon
glyphicon-pencil" aria-hidden="true"></span></a>

                                    </c:if> </h4>


                        <div class="row">
                        <div class="col-sm-6 col-md-10">
                            ${physician.email }
                            </div>
                            <div class="col-sm-6 col-md-1">
                                    ${physician.medLicenseId }
```

```
                        </div>
                        </div>

                        <div class="row">
                        <div class="col-sm-6 col-md-10">
                            <c:forEach items="${physician.specialties}"
var="specialty" varStatus="specialtyVarStatus"><c:out
value="${specialty.description}" /><c:if
test="${!specialtyVarStatus.isLast()}">;</c:if> </c:forEach>
                            </div>
                            <div class="col-sm-6 col-md-1">
                            ${physician.cellphoneNumber }
                            </div>
                            </div>
                    </div>
                    </div>
              </div>
            </div>

            <c:if test="${fn:length(physician.clinics)>0}">
            <div class="panel panel-default">
            <div class="panel-heading">
               Clinic Information
            </div>
            <div class="panel-body">
                    <c:forEach items="${physician.clinics }" var="clinic">
                            <div class="panel">
                                    <div class="panel-body">
                                            <div class="col-sm-4 col-md-4">
                                                    ${clinic.name }
                                            </div>
                                            <div class="col-sm-4 col-md-4">
                                                    ${clinic.address }
                                            </div>
                                            <div class="col-sm-4 col-md-4">
                                                    ${clinic.telNo }
                                            </div>
                                    </div>
                            </div>
                    </c:forEach>

            </div>
            </div>
            </c:if>
            </div>
            </div>
</div>
</div>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/references/references.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="references"/>
<%@include file="../includes/nav.jsp" %>
<div class="container-fluid">
<h5></h5>
<div class="container">
   <h3><b>References</b></h3>
    <div class="row">
            <div class="col-xs-15 col-lg-12">
            <h3></h3>
                    <div class="panel panel-default" >
                    <div class="panel-heading" >References</div>
                    <div id="latestreferences" class="panel-body" >

                            <div id="references" class="infinite-scroll"
style="height: 400px; max-height:400px; overflow-y: scroll;">
                            <div class=""><a class="jscroll-next" href="references-
loadreferences.do?page=1" ">next</a> </div>
                            </div>
                    </div>
                    </div>
            </div>
            </div>
</div>
```

```
</div>
<script type="text/javascript">
    jQuery(document).ready(function() {
            jQuery('#references').jscroll({
                loadingHtml: '<img src="images/loading.gif" alt="Loading" />
Loading...',
                nextSelector: 'a.jscroll-next:last',
                debug: true
            });

    });
    </script>
<%@include file="../includes/footer.jsp"%>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/report/report-admin-barchart.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/chart/Chart.js" type="text/javascript"></script>
<script src="scripts/chart/Chart.HorizontalBar.js"></script>
<input id="adminReportStartDate" type="hidden" value="${startDate}">
<input id="adminReportEndDate" type="hidden" value="${endDate}">
<input id="adminReportSelectionString" type="hidden" value="${selectionString}">
<div class="row">
    <div class="col-sm-12 col-md-12">
        <div class="panel panel-default">
            <div class="panel-heading">${selectionString} report between ${startDate}
to ${endDate}</div>
            <div class="panel-body">
                <div style="width: 99%">
                    <canvas id="reportChart"/>
                </div>
                <input class="btn btn-sm btn-primary" type="button"
onclick="exportAdminBarChartReportToPDF()" value="Export to PDF">
            </div>
        </div>
    </div>
</div>
<script type="text/javascript">
    jQuery(document).ready(function() {
        var dataReport = JSON.parse('<s:property value="dataReport"
escapeHtml="false"/>');
        var numberOfDataObjects = Object.keys(dataReport).length / 3;
        var labels = new Array(numberOfDataObjects);
        for(ctr = 0; ctr < numberOfDataObjects; ctr++) {
            var lastname = dataReport["lastname" + (ctr + 1)];
            var firstname = dataReport["firstname" + (ctr + 1)];
            labels[ctr] = lastname + ", " + firstname;
        }
        var dataValues = new Array(numberOfDataObjects);
        for(ctr = 0; ctr < numberOfDataObjects; ctr++) {
            var count = dataReport["count" + (ctr + 1)];
            dataValues[ctr] = count;
        }
        labels.reverse();
        dataValues.reverse();

        var chartData = {
                    labels: labels,
                    datasets: [
                       {
                       label: "Appointment Report",
                       fillColor: "rgba(151,187,205,0.5)",
                           strokeColor: "rgba(151,187,205,0.8)",
                           highlightFill: "rgba(151,187,205,0.75)",
                           highlightStroke: "rgba(151,187,205,1)",
                           data: dataValues
                       }
                    ]
        }

        var ctx = document.getElementById("reportChart").getContext("2d");
        var barChart = new Chart(ctx).HorizontalBar(chartData, {
            responsive: true
        });
    });
```

```
    function exportAdminBarChartReportToPDF() {
        var startDate = $("#adminReportStartDate").val();
        var endDate = $("#adminReportEndDate").val();
        var adminReportSelectionString = $("#adminReportSelectionString").val();
        if(adminReportSelectionString == 'Requesting Physician') {
            window.open("/consultations-reports-requestingphysician-admin-barchart-
pdf.do?startDate=" + startDate + "&endDate=" + endDate, "_blank", null);
        } else if(adminReportSelectionString = 'Consultant') {
            window.open("/consultations-reports-consultant-admin-barchart-
pdf.do?startDate=" + startDate + "&endDate=" + endDate, "_blank", null);
        }
    }
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/report/report-admin-table.jsp**

```
<%@taglib uri="/struts-tags" prefix="s" %>
<script src="scripts/jq/jquery-1.11.0.min.js" type="text/javascript"></script>
<script src="scripts/chart/Chart.js" type="text/javascript"></script>
<input id="adminReportStartDate" type="hidden" value="${startDate}">
<input id="adminReportEndDate" type="hidden" value="${endDate}">
<input id="adminReportSelectionString" type="hidden" value="${selectionString}">
<div class="row">
    <div class="col-sm-12 col-md-12">
        <div class="panel panel-default">
            <div class="panel-heading">${selectionString} report between ${startDate}
to ${endDate}</div>
            <div class="panel-body">
                <table id="adminReportTable" class="table">
                    <th>Physician</th>
                    <th>Count</th>
                </table>
                <input class="btn btn-sm btn-primary" type="button"
onclick="exportAdminTableReportToPDF()" value="Export to PDF">
            </div>
        </div>
    </div>
</div>
<script type="text/javascript">
    jQuery(document).ready(function() {
        var dataReport = JSON.parse('<s:property value="dataReport"
escapeHtml="false"/>');
        var numberOfDataObjects = Object.keys(dataReport).length / 3;
        for(ctr = 0; ctr < numberOfDataObjects; ctr++) {
            var lastname = dataReport["lastname" + (ctr + 1)];
            var firstname = dataReport["firstname" + (ctr + 1)];
            var count = dataReport["count" + (ctr + 1)];

            var physicianCell = document.createElement('td');
            physicianCell.appendChild(document.createTextNode(lastname + ", " +
firstname));

            var countCell = document.createElement('td');
            countCell.appendChild(document.createTextNode(count));

            var row = document.createElement('tr');
            row.appendChild(physicianCell);
            row.appendChild(countCell);


document.getElementById('adminReportTable').firstElementChild.appendChild(row);
        }
    });
    function exportAdminTableReportToPDF() {
        var startDate = $("#adminReportStartDate").val();
        var endDate = $("#adminReportEndDate").val();
        var adminReportSelectionString = $("#adminReportSelectionString").val();
        if(adminReportSelectionString == 'Requesting Physician') {
            window.open("/consultations-reports-requestingphysician-admin-table-
pdf.do?startDate=" + startDate + "&endDate=" + endDate, "_blank", null);
        } else if(adminReportSelectionString = 'Consultant') {
            window.open("/consultations-reports-consultant-admin-table-
pdf.do?startDate=" + startDate + "&endDate=" + endDate, "_blank", null);
        }
    }
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/report/consultant.jsp**

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<input id="consultantReportStartDate" type="hidden" value="${startDate}">
<input id="consultantReportEndDate" type="hidden" value="${endDate}">
<div class="row">
    <div class="col-sm-12 col-md-12">
            <div class="panel panel-default">
                    <div class="panel-heading">Report from ${startDate} to
${endDate} as Consultant</div>
                    <div id="" class="infinite-scroll" style="max-height:
400px;overflow-y: scroll;">
                    <div class="panel-body">
                            <table class="table">
                                <th>Clinical Question</th>
                                <th>Requesting Physician</th>
                                <th>Date of Consult</th>
                                <th>Time</th>
                                <th>Status</th>
                                    <s:iterator value="appointments"
var="appointmentsVar">
                                            <tr>
                                                    <td><s:property
value="#appointmentsVar.clinicalCase.question" /></td>
                                                    <td><s:property
value="#appointmentsVar.requestingPhysician.getFullName()" /></td>
                                                    <td><s:property
value="#appointmentsVar.consultationDate" /></td>
                                                    <td><s:property
value="#appointmentsVar.consultationStart.toLocalTime().toString('HH:mm:ss')" /> -
<s:property
value="#appointmentsVar.consultationEnd.toLocalTime().toString('HH:mm:ss')" /></td>
                                                    <td><s:property
value="#appointmentsVar.clinicalCase.status" /></td>
                                            </tr>
                                    </s:iterator>
                            </table>
                            </div>
                            </div>
                            <div class="panel-body">
                            <input class="btn btn-sm btn-primary" type="button"
onclick="exportConsultantReportToPDF()" value="Export to PDF">

            </div>
    </div>
</div>
<script type="text/javascript">
   function exportConsultantReportToPDF() {
       var startDate = $("#consultantReportStartDate").val();
       var endDate = $("#consultantReportEndDate").val();
       window.open("/consultations-reports-consultant-pdf.do?startDate=" + startDate
+ "&endDate=" + endDate, "_blank", null);
   }
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/report/report-requestingphysician.jsp**

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<input id="requestingPhyisicanReportStartDate" type="hidden" value="${startDate}">
<input id="requestingPhyisicanReportEndDate" type="hidden" value="${endDate}">
<div class="row">
    <div class="col-sm-12 col-md-12">
            <div class="panel panel-default">
                    <div class="panel-heading">Report from ${startDate} to
${endDate} as Requesting Physician</div>
                    <div id="" class="infinite-scroll" style="max-height:
400px;overflow-y: scroll;">
                    <div class="panel-body">
                            <table class="table">
                                <th>Clinical Question</th>
                                <th>Consultant</th>
                                <th>Date of Consult</th>
```

```
                                    <th>Time</th>
                                    <th>Status</th>
                                        <s:iterator value="appointments"
var="appointmentsVar">
                                                <tr>
                                                    <td><s:property
value="#appointmentsVar.clinicalCase.question" /></td>
                                                    <td><s:property
value="#appointmentsVar.consultant.getFullName()" /></td>
                                                    <td><s:property
value="#appointmentsVar.consultationDate" /></td>
                                                    <td><s:property
value="#appointmentsVar.consultationStart.toLocalTime().toString('HH:mm:ss')" /> -
<s:property
value="#appointmentsVar.consultationEnd.toLocalTime().toString('HH:mm:ss')" /></td>
                                                    <td><s:property
value="#appointmentsVar.clinicalCase.status" /></td>
                                                </tr>
                                        </s:iterator>
                                </table>
                                </div>
                                </div>
                                <div class="panel-body"><input class="btn btn-sm btn-
primary" type="button" onclick="exportRequestingPhysicianReportToPDF()" value="Export
to PDF">

            </div>
    </div>
</div>
<script type="text/javascript">
   function exportRequestingPhysicianReportToPDF() {
       var startDate = $("#requestingPhyisicanReportStartDate").val();
       var endDate = $("#requestingPhyisicanReportEndDate").val();
       window.open("/consultations-reports-requestingphysician-pdf.do?startDate=" +
startDate + "&endDate=" + endDate, "_blank", null);
   }
</script>
```

- **web-pcs/web/WEB-INF/velocity/view/jsp/report/report.jsp**

```
<%@include file="../includes/header.jsp"%>
<c:set var="nav" value="consultations" />
<%@include file="../includes/nav.jsp"%>
<div class="container-fluid">
    <div class="container">
    <br>
        <c:choose>
            <c:when test="${currentLoggedInUser.userType == 'Physician'}">
                <div class="row">
                    <div class="col-sm-12 col-md-12">
                        <div class="panel panel-default">
                            <div class="panel-heading">Report</div>
                            <div class="panel-body">
                                <div>
                                    <form id="physicianReportForm">
                                        <input id="physicianReportSelection1"
name="selection" type="radio" onclick="selectReport('consultations-reports-
requestingphysician', 'Requesting Physician')" checked="checked" >
                                        <label
for="physicianReportSelection1">Requesting Physician</label>
                                        <input id="physicianReportSelection2"
name="selection" type="radio" onclick="selectReport('consultations-reports-
consultant', 'Consultant')">
                                        <label
for="physicianReportSelection2">Consultant</label>
                                        <h5></h5>
                                        <table class="table">
                                          <tr>
                                              <td style="width: 10%">Start Date:
</td>
                                              <td><input type="text"
id="physicianStartdatepicker" name="startDate"></td>
                                          </tr>

                                          <tr>
                                               <td style="width: 10%">End Date: </td>
```

```
                                            <td><input type="text"
id="physicianEnddatepicker" name="endDate"></td>
                                              </tr>
                                            </table>
                                            <input class="btn btn-sm btn-primary"
type="button" onclick="generatePhysicianReport()" value="Generate Report">
                                        </form>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                         </c:when>
                         <c:otherwise>
                    <div class="row">
                        <div class="col-sm-12 col-md-12">
                            <div class="panel panel-default">
                                <div class="panel-heading">Report</div>
                                <div class="panel-body">
                                    <div>
                                        <form id="adminReportForm">
                                            <input id="adminReportSelection1"
name="selection" type="radio" onclick="selectReport('consultations-reports-
requestingphysician', 'Requesting Physician')" checked="checked" >
                                            <label for="adminReportSelection1">Requesting
Physician</label>
                                            <input id="adminReportSelection2"
name="selection" type="radio" onclick="selectReport('consultations-reports-
consultant', 'Consultant')">
                                            <label
for="adminReportSelection2">Consultant</label>
                                            <table class="table">
                                              <tr>
                                                  <td style="width: 10%">Start Date:
</td>
                                                  <td><input type="text"
id="adminStartdatepicker" name="startDate"></td>
                                              </tr>
                                              <tr>
                                                  <td style="width: 10%">End Date: </td>
                                                  <td><input type="text"
id="adminEnddatepicker" name="endDate"></td>
                                              </tr>
                                            </table>
                                            <input id="adminViewSelection1" name="view"
type="radio" onclick="selectView('barchart')" checked="checked">
                                            <label for="adminViewSelection1">Bar
Chart</label>
                                            <input id="adminViewSelection2" name="view"
type="radio" onclick="selectView('table')">
                                            <label
for="adminViewSelection2">Table</label>
                                            <input class="btn btn-sm btn-primary"
type="button" onclick="generateAdminReport()" value="Generate Report">
                                        </form>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                         </c:otherwise>
                </c:choose>
                <div id="reportOutput"></div>
        </div>
</div>

<script type="text/javascript">
    var selectedView;
    var selectedReport;
    var selectionString;
    $(function() {
    $("#physicianStartdatepicker").datepicker();
        $("#physicianEnddatepicker").datepicker();
        $("#adminStartdatepicker").datepicker();
        $("#adminEnddatepicker").datepicker();
        selectedReport = "consultations-reports-requestingphysician";
```

```
        selectedView = "barchart";
        selectionString = "Requesting Physician";
    });
    function selectReport(report, opt) {
        selectedReport = report;
        selectionString = opt;
    }
    function selectView(view) {
    selectedView = view;
    }
    function generatePhysicianReport(){
    var startDate = $("#physicianStartdatepicker").val();
        var endDate = $("#physicianEnddatepicker").val();
        if(startDate == "" || endDate == "") {
            alert("Please select date.");
        } else if(!isValidDate(startDate) || !isValidDate(endDate)) {
            alert("Please input a valid date.")
        } else {
            var parsedStartDate = convertStringToDate(startDate);
            var parsedEndDate = convertStringToDate(endDate);
            if(parsedStartDate > parsedEndDate) {
                    alert("Selected start date must not be after the selected end
date.")
            } else {
                $.ajax({
                    url : selectedReport + ".do?selectionString=" + selectionString +
"&",
                    data : $("#physicianReportForm").serialize(),
                    type : 'post',
                    cache : false
                }).done(function(data) {
                    $("#reportOutput").html(data);
                });
            }
        }
    }
    function generateAdminReport() {
    var startDate = $("#adminStartdatepicker").val();
    var endDate = $("#adminEnddatepicker").val();
    if(startDate == "" || endDate == "") {
            alert("Please select date.");
    } else if(!isValidDate(startDate) || !isValidDate(endDate)) {
            alert("Please input a valid date.")
    } else {
            var parsedStartDate = convertStringToDate(startDate);
            var parsedEndDate = convertStringToDate(endDate);
            if(parsedStartDate > parsedEndDate) {
                alert("Selected start date must not be after the selected end date.")
            } else {
                    $.ajax({
                    url : selectedReport + "-admin-" + selectedView +
".do?selectionString=" + selectionString + "&",
                    data : $("#adminReportForm").serialize(),
                    type : 'post',
                    cache : false
                }).done(function(data) {
                    $("#reportOutput").html(data);
                });
            }
    }
    }
    function isValidDate(text) {
        var date = Date.parse(text);
        if (isNaN(date)) {
            return false;
        }
        var comp = text.split('/');
        if (comp.length !== 3) {
            return false;
        }
        var m = parseInt(comp[0], 10);
        var d = parseInt(comp[1], 10);
        var y = parseInt(comp[2], 10);
        var date = new Date(y, m - 1, d);
        return (date.getFullYear() == y && date.getMonth() + 1 == m && date.getDate()
== d);
```

```
    }
    function convertStringToDate(text) {
    var comp = text.split('/');
    var m = parseInt(comp[0], 10);
        var d = parseInt(comp[1], 10);
        var y = parseInt(comp[2], 10);
        return new Date(y, m - 1, d);
    }
</script>
%@include file="../includes/footer.jsp"%
```

XII. ACKNOWLEDGEMENT

Mom and Dad, I do not even know how to thank you for everything you have done and everything you have sacrificed for me. I know I can never square you for being the best parents in the world, but I hope I make you proud. I would not be here if it was not for your endless love and support. It is all for you, everything I do. I love you!

Goldey, thank you for being my partner in crime and for always being the one I can turn to no matter what. Knowing that you will always be on my side and by my side is such a blessing. We both know how much effort we put into this thing. I appreciate how you take time to go with me to school (or elsewhere) and carry loads of paper. Many times we would fight over the laptop ☺ but you always have a patient heart. You have played a role in this achievement, my sister, so thank you! Ate loves you!

Sir Aldrich Co, thank you for all your help and support in setting up my study. You have set an example of excellence as a mentor, even in a short span of time. Your inputs and feedback about my study were very helpful in making it a significant one and I still bring all your lessons with me, every day.

Sir Geof Solano, thank you for being my second adviser. I truly appreciate the time you alloted so that I can finally proceed with my study. I am very lucky to be assisted by you and **Ma'am Bic Amarillo** which led to the approval of my study. Thank you, sir! And I wish you more success in your career.

Ma'am Perl Gasmen, I know that you have a very looong list of advisees, but thank you for having me on board ☺ Seriously, how would I be without you? You only became my adviser in the latter part of the semester, but it was the most crucial one and you helped me get through it. I learned so much from you and I learned everything with a smile on my face. You will always be one of the people whom I will look up to for the rest of my life.

Jeselle, my best friend and my number 1 supporter, thank you for lifting me up and making me believe in myself again whenever I am at my lowest point. You are very patient and you never stopped believing that I could reach my goal. We do not get to spend as much time together, but it helps knowing you are there whenever I need you.

Shoutout to **Zach Marasigan** for lending a hand just before my defense! ☺

Lord, thank you for the grace and guidance. Thank you for helping me stay focused and determined. You are the reason I stand proud today and I have no greater achievement

than being Your warrior and child. I offer this most especially to You because in all things, You shall be glorified.

And thank you, **UP Manila**, for everything. After 6 long years of stay, all I can say is FINALLY! Right at this point in time, I still can't believe I actually pulled it off. It just proves that you really can do whatever you set your mind to if you just try hard enough.