University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Scrum Project Planner Primer - a software engineering tool for university setting

A special problem in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Alyra Y. Escotido

April 2014

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 agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Scrum Project Planner Primer - a software engineering tool for university setting" prepared and submitted by Alyra Y. Escotido in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Bernie B. Terrado
Adviser

EXAMINERS:

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

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.
Dean
College of Arts and Sciences

Abstract

Agile-Scrum methodology is developed to address issues on immediate deadlines of the projects, requirements changes and problems in communication with the customers. It is a software methodology that deals with repetitive work cycles and is iterative and is now popular to different software companies. Students can also use Scrum for their software projects. Issues like short duration for the project, not fixed requirement specifications, and weekly updates or milestones to the clients or instructor are common to every student team. These issues are parallel on what Scrum methodology can solve. In this study, a collaboration tool that adapts the Agile-Scrum methodology and is designed as a tool suitable for the university environment was developed.

Keywords: Agile, Scrum, Software Development, Software Engineering Tool

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Software development methodologies (SDM) are reference models that correspond to the status of a developing software projects. These methodologies determine the organization of the development of a certain project by means of its frameworks. Different software development methodologies mainly provide the activities for the management of projects. These methodologies are used to prevent the occurrence of project failures. [1]

The Waterfall is the traditional methodology in software development which follows a sequential life cycle. This methodology begins in a thorough planning of the details, requirements and design of the end product. Tasks are immediately identified from the start and all the work needed are organized thru the use of software tool such as Microsoft Project which provides Gantt Chart. Each task was estimated in the planning phase and everything is well documented. Throughout the development process, everything was executed according to what is planned until the production of the end product. [2]

The greatest strength of following the Waterfall is how organized and logical it is. Everything are well documented and planned from the start. The problem with this is that all good ideas should be written down at the beginning but in reality all good ideas are developed throughout the whole process. Documenting all the requirements in a detailed manner is a must. But once the development team starts to read for example the requirements specification, most of the time misunderstanding rises between what the customers want and what the developers understand. Customers may expect the initial things they asked for but the end product is not according to what they desire. Also, unanticipated problems are more likely to happen during the development of the product because it is difficult to plan uncertain things from the start. [2]

Agile, another example of SDM, mostly emerges these past few years to address issues on immediate deadlines of the projects, project requirements changes and problems in communication with the customers. It was modeled to be a software development methodology that was based on flexibility, communication skill within the team and the customer, self-organization of the development team, and the ability to adapt and respond to changes in an instant [3]. Agile is a software methodology that deals with repetitive work cycles and is iterative [4].

Agile methodologies make the project to be assessed during the whole lifecycle process. This made possible through the use of cadences of work that is called iterations or sprints in which the team should possibly deliver a shippable product increment. This approach can greatly reduce time and cost to market. Since the teams work cycle is bounded by certain number of weeks, it is easy for the stakeholders to define the latest date for product release. Agile ensures a teams work will never be wasted and it always preserves a products critical market relevance. [5]

Scrum is the most popular Agile method. The idea was inspired from the game of Rugby in which a self-organizing team works together towards the development of a product. Companies including Yahoo!, Microsoft, Google, Lockheed Martin, Motorola, SAP, Cisco, GE, CapitalOne and the US Federal Reserve use Scrum. Many teams using Scrum report significant changes in productivity. [2]

Being a framework of Agile, Scrum is incremental in nature with certain duration around one to four weeks. These incremental durations are called Sprints. There are three important roles that suffice the Scrum. The Product Owner, usually the customer, is the one who handles the requirements and is responsible in writing it in a Product Backlog where each requirement is written as a User Story. The one who does the development of the project is the Development Team. The Scrum Master is part of the Development Team who monitors them along the development period and serves as the bridge between the team and to other external roles to resolve issues. Every sprint planning, the team estimates the time to complete a certain task through the process called Poker Planning. Estimated hours were according to the set of Fibonacci numbers. A daily Scrum Meeting is hosted by the Scrum Master and is being conducted by the Scrum Team where each of the members should answer the three questions (What he/she have done today; what he/she will do next; and what are the impediments that hinder him/her to finish a certain story). After each sprint, a demo of the project will be held and a retrospective will follow right after. Sprint retrospective is a gathering of the Scrum Team where they will tackle about the happenings during the latest Sprint. Sprint Burndown Chart is used to represent the actual progress of the project and the difference from the baseline. [6]

Global Software Development (GSD) is the new trend in IT industry nowadays where development teams are scattered across cities and some by countries. GSD suggests reducing time to market, increasing productivity, improving quality and gaining cost effectiveness and efficiency. Stakeholders are particularly from different countries and do have own culture. Basically, they are geographically separated and some experiencing different time zones. [4]

Because of GSD, problems in terms of team communication and coordination possibly immersed. It is obvious that in Agile methodologies, daily communication within the team and with the customer is advocacy of it thus its very crucial in the projects success. Even though these issues bound, using Agile methodologies and Scrum gather enough interest to be used in industry [6]. Because of its goals and focus, agile methodologies, especially Scrum, have become popular on software development nowadays [7]. Today, Scrum is widely used among software development organizations, especially for those whose goal is to increase the projects success rate [8]. Agile methodologies supporters cite that a software development team can produce quality software that responds to the early deadline demand by the customer and can acknowledge changes skillfully, by working in small and manageable work packages and delivering it in incremental manner [9]. On a recent study, it was found out that Scrum used in GSD can improve communication, trust, motivation and product quality. The study also said that in industry nowadays, invoke that Scrum practices provide better communication and collaboration among the team, thus clearly indicating the assurance of producing quality software [4].

Students can also use Scrum for their software projects. Issues like short duration for the project, not fixed requirement specifications, and weekly updates or milestones to the clients or instructor are common to every student team. Because of timely updates and tight deadlines, some students tend to develop everyday even for a short span of time to avoid cramming. These issues are parallel on what Scrum methodology can solve but it is still difficult to evaluate in a student group because of wide spatial distribution that they rarely work together in same location [10] and every member has also their own schedules and responsibilities in other subject matter.

For every software development methodology to be followed, it is essential to have a collaboration tool that will help them to adapt to the certain model and at the same time will serve as storage for every requirement specification, iteration planning and progress tracking and reporting [11]. These tools are also important to teams that are not collocated.

Agile-Scrum is not an exception to have tools that will aide its users since in the first place Agile emphasizes Individuals and interaction over tools and processes. These tools help the development teams to attain their goal by having a clear proof of what requirements do customers expected ordered by priority. Scrum tools that have all the basic and special features and are commercially available such as Rally, VersionOne and Jira are now widely used by different organizations. [12]

Although these existing tools may fulfill what any Scrum team needs, it is designed for a professional environment. Because of the variety of features available, these tools bring complexity to most of students due to unfamiliarity to software engineering tools. Students who just want to apply scrum for their software development projects may not use what all features available but only the necessities that will satisfy their needs. Another is there should be a tool that will not just accompany them towards the development but also guide them on how to correctly adapt to the whole process.

B. Statement of the Problem

It is an edge for undergraduate students to learn and experience Scrum because of its goals that are in line of the issues a student team always experience. Unfortunately it is still difficult to apply in an academic setting due to its requirements in team collaboration unlike in a professional environment because of wide spatial distribution of students.

Scrum for its easy implementation in academic setting needs an end-to-end tooling support since collaboration is organized thru these tools but the existing tools that are commercially available is generalized and not designed for the university environment. Because most of the software development tools are designed for the professionals, results showed that applying and adjusting to it may take longer time because of the complexity of the tools. Another is these existing tools do not support reviews for each work package that is highly needed for the quality of the project of the students. Students need an immediate feedback from their instructors or clients before they invest a great amount of work because it will utilize their time and effort in finishing the project. [13]

C. Objectives of the Study

General:

To develop a special tool that will be modeled for the implementation of Scrum Methodology designed for student developers.

Specific:

The system aims to help the users to apply the practices of Scrum by providing infotips or tooltips that will explain and define a specific scrum term or process. Also, the system will be designed in sequential manner on how Scrum works.

The following are the user types of the tool including their specific privileges.

1. Allows the Scrum Master to:
   (a) Update project
   (b) Send invitation to members and product owner via email
   (c) View of notification of confirmation of the members
   (d) View the product backlog
   (e) View the sprint backlog with burndown chart and scrum board
   (f) Add tasks under each user stories in the sprint backlog given the time estimated.
   (g) Add accomplished activity done for the day
   (h) Initiate the daily scrum meeting
   (i) Add details in the daily scrum meeting dashboard
   (j) Receive notification of the upcoming sprint review/project closure
   (k) View past sprints with reviews
   (l) Update password/account
2. Allows each member of the team to:
   (a) Receive the invitation from the Scrum Master in email
   (b) Accept the invitation
      i. If there is an existing account, sign in then automatically be a member of the project team
      ii. Otherwise, register then automatically be a member of the project team
   (c) View the product backlog
   (d) View the sprint backlog with burndown chart and scrum board
   (e) Add tasks under each user stories in the sprint backlog.

   (f) Add accomplished activity done for the day

   (g) Receive notification for daily scrum meeting

   (h) Participate in the daily scrum meeting by adding details in the daily scrum meeting dashboard

   (i) Receive notification of the upcoming sprint review

   (j) View past sprints with reviews

   (k) Update password/account

3. Allows the Product Owner to:

   (a) Update the product backlog

   (b) Open the sprint by indicating the user stories needed for the current sprint and set date for sprint review

   (c) View the sprint backlog with burndown chart and scrum board

   (d) Perform Sprint Review then automatically closes the sprint

   (e) Close Project

   (f) Update password/account

4. Allows a Guest User to:

   (a) Sign up for new user account

   (b) View Tutorials/FAQ of Business Processes of Scrum Methodology

5. Allows System Administrator to:

   (a) Add a new account

   (b) Edit an account

   (c) Deactivate an account

D. Significance of the Project

If applying Scrum in university, tools for creation and planning of works intended for a software project are important to use. Important aspects that a software engineering tool designed for student teams are stated. First is the ability of the tool that allows the team to collaborate and set as storage for the requirements. Next is the versioning that describes the ability of the tool to track historical changes of each task or work package. Another is work product support refers to the ability of the tool to categorize and record each work package. Fourth is the availability of the tool to apply in each computer machine. The last one is the ability of the tool to record the reviews and evaluation from the product owner in each user stories. [13]

For the project team, this collaboration tool will help them to organize and thoroughly list the tasks needed since there will be a storage for the requirements. The tool will enable them to be more interactive on what the other team members are doing by participating in the daily scrum even though they are not collocated. It is also important to each of the member to know the current progress of the project so that they can tell by themselves if they can complete the tasks given the deadline of the sprint. By adding the Product Owner, or the Customer/Client of the project, as an additional user type, the project team is now more aware to take note of the reviews. Customer satisfaction will increase its rate because the customers can now testify the changes they want in the project. With the reviews, changes can be easily adapted as the creation of tasks will be designed toward the customers preferences. Immediate reviews can also minimize their time and effort to develop the project because no great amount of work will be wasted.

For the customers or clients of the project being the Product Owner, the existence of the tool helps them to be more aware on whats going on through the team. They can monitor if the project is progressing or not. Setting user stories every Sprint means that the product owner is the one who drives how the project is progressing which is important for the acceptance of the project. The product owner can also sets realistic expectation on what to deliver given a certain period of time because of the awareness of the projects progress every Sprint. Also, by providing the reviews, the product owner can be positive on what he/she really wants to expect to be the outcome of the project.

The tool will also help the students that will use the tool to be more familiarize to some useful terms and to understand well the whole process because tooltips will be provided. The tool will also be modeled according to the flow of Scrum and each process will have an overview explanation and will be presented sequentially and in cycle. With these, students will not just experience Scrum but also will help them to know how Scrum be applied correctly.

Overall, the tool will utilize the complexity and sophistication of existing tools and will cover all the said aspects for Scrum to be easily applied in an academic setting. It will be a simple tool that is straightforward in application of Scrum that will cover the key roles. The tool will give them access to experience scrum feasibly during undergraduate years.

E. Scope and Limitations

1. The tool will not ask for the fixed time each individual can commit each day because of the variety of own schedules.

2. The tool will include the following process only: Daily Scrum Meeting, for Sprint Retrospective and Sprint Planning is feasible to execute in personal and physical meetings.

3. The tool will not include the capability to support the demonstration of the project every Sprint Review.

4. There should only be one Product Owner per project and one permanent Scrum Master.

F. Assumptions

1. Product owner reviews for user stories are after or during the sprint review which is after or during the day the sprint ends.

2. Scrum master knows the right email addresses of the members and product owner.

3. Users have a preliminary background about software engineering.

4. Choice of programming language and any other external software to be used should be indicated as a specification of the project

II. Review of Related Literature

To execute different development practices, Software Development usually depends on the use of tool. Project Management Tools are not new to all of us and it is obvious that many of them are commercially available online. Many IT companies used these available tools for organized collaboration and some of these tools are encourage using in project management course [14].

In a brief comparative review about software project management tools, it was shown that many of them are more geared for the use of project managers only and few for the collaboration of team. @Task, Clarizen and Gemini are the ones that do have the most number of features. It includes the features time tracking, tasks management, version control and milestones, earned value analysis, critical path method and most of them are web-based. These tools are designed for the integration of project management processes. Jira is a project management tool that combines issue tracking and Agile project management. [15]

Scrum is an Agile framework which means that it adapts the agile practices including the roles that are used in software development processes and is incremental in nature. Its basic principles are having self-organizing teams, responds to changes and able to deliver a shippable products in short time. It has been observed that Scrum is gaining attention in industry nowadays especially for those companies that aims to increase the success rate of their projects by applying these software development processes specifically Agile-Scrum. [8]

To design a specific program for a certain environment, there are some contexts that are needed to consider first for the environment specificities. In [16], a design for an agile tool for a leading software engineering company in agile software development was proposed. They decided to use the CMMI-like presentation for the visualization of the tasks. CMMI lists all the tasks and activities in sequential and tabular form. The company still relies on self-intuition in planning and use burndown chart for tracking the overview progress. Additionally, a good agile metric should measure the outcome, follows trends, provides section for conversation and encourages better quality.

In another comparative review of common Agile tools, it was found out that Rally and VersionOne provide the most number of features. VersionOne has the ability to track dependent stories, defect management and has a robust planning activities. It also supports collaboration of cross-functional teams but it has a complex user interface in which higher learning curve is needed. Rally includes integrated defect management but also has a complex user interface. [12]

Since the focus of the study is to design an appropriate tool for the university settings, several studies that involve the application of agile methodologies are reviewed for the reference. Different institutions tested the capability to apply agile methodologies given the university environment. With these studies, problems and several observations can be attained.

In Vienna University of Technology [10], they evaluate how suitable agile methodologies in student groups by applying it in their Software Engineering Seminar course. Student groups are composed of Software Engineering and Internet Computing master students. All roles are assigned to each of the members of each team. Each Student Groups are required to present how they adapt their team to Scrum and implement it. Results showed that the issues in applying Agile Methodologies are mainly about the Daily Meetings and Communication. Problems arose due to lack of communication and small team members. To address the problems that came out, students adjust and provide possible solutions. For communication issues, each team used Skype which is an internet based voice communication. Minutes of the meeting are passed thru email. Requirements priority issues were addressed by letting each team decides the prioritization of each task.

In Universidade Federal do Amazonas, a federal university in Brazil, another case study has been conducted, where the students involved are Major in Informatics and Electrical Engineering. Students were again grouped and assigned a certain project. Scrum process were presented and taught to the students in the course. On the other hand, due to some circumstances that in a university environment limits, especially the daily meetings, the team decided to use email as a basis of communication. In terms of tracking the progress of the project, each team used Google Groups and Google Docs for sharing. They signified that these tools were helpful because it allows documents to be accessed and modified by each member at the same time and it also provides an instant communication despite the physical distance. With the use of these tools, despite the problems encountered, applying Scrum in academic context shows positive results in their studies. [17]

Theres also a study that tested the ability of Scrum in terms of Global Software Development. Theres collaboration among the students from USA, India and Senegal that developed a mobile application together. The study pointed out the experience in using Scrum in Global Software Development with the use of an end-to-end tooling. They used the IBM RTC environment provided by the researchers. It is a collaboration tool use for tracking the project progress and is integrated in the eclipse IDE. Communications and meetings are done thru Google Mail and Google Chat since chat functionality is not included in RTC and wikis for planning and sharing documents. Scrum processes were adjusted and followed with the use of the tool. For the result, the student developers said that the collaboration is impossible without the tooling but RTC is quite difficult to use because they cant adjust the schedules and it took time for them to be familiarized. [6]

In the collaboration study between the University of Victoria and Aalto University for the application of GSD [18], students were exposed to different development tools to implement Scrum. They used the IBM RTC environment and used Google Hangouts for the communication. The project finished in time but the team experienced difficulties during development because of different time zones that affected their daily meetings even though theres a tool online.

Another, the University Of Maryland University College used Agile Methods for their capstone course in software engineering last 2008. With the short training in agile methods, skills in developing a website and the use of collaboration tools, all the teams successfully pulled-off the e-commerce websites in l3 weeks. They used the Version One, an agile project management tool for team collaboration. All the teams had lack of interaction with their customers resulting into lack in customer feedback of the features thus, somewhat fail in meeting the exact requirements of the project. [19]

For the final project in a Virtual University of Pakistan, a hybrid model of Agile-Scrum approach and the traditional Waterfall has been proposed. The students chose the project management tools based on their needs in the functionalities that will cover the communication problems, tracking of the product log, and management of sprint tasks. The proposed model will facilitate the development and management of the project. It allows the students to mandatory record their daily tasks in the tool. It was shown that breaking down the tasks in smaller parts became a way for the students to better understand the requirements of the project. Also, smaller tasks helped in the easy and fair assignment among the members. [20]

From the said studies it was concluded that student teams faced problems especially in the communication and feature feedback part. Customer feedback is needed to attain Customer Satisfaction that is the standard for the quality of the product. Since one of the purposes of producing projects in university is to be prepared for the expectation of the industry, Customer Satisfaction issue should be taken into account. The techniques used to develop software have highly impact on customer directly thus it is critical to study how these techniques affect the customer. Customer involvement will help in meeting the product requirements. [10]

A collaboration tool for Scrum methodology that will sum up all the needed features of student developers will be developed. These are the collaboration through daily scrum meeting, tasks management, requirements storage and an additional account for the Customers for their reviews on each user stories.

III. Theoretical Framework

1. **Software Engineering Tools**

   Software engineering and development tools are automated tools that are used in the whole software life cycle processes. These tools allow a software developer to be more focused on developing rather than be bothered on the non-technical side of the processes. Tools are usually designed for a certain software development method thus it tends to apply the methods more systematically and easy by reducing the work of applying the method manually. [21]

   (a) **Software Requirements Tools** - these tools are developed for the storage of the requirements that were validated to be included in the project. Requirements should also be traceable along the life cycle processes.

   (b) **Software Engineering Management Tools** - these tools include the project planning and tracking. Project planning and tracking includes project effort measurement and the schedule of the project.

   (c) **Software Engineering Processes Tools** – these tools include the process modeling tools that are used to model and apply software engineering processes.

   (d) **Software Quality Tools** - these tools include the review and audit tools.

2. **Agile Methodology**

   Agile methodologies are software development processes that support the agile philosophy. It is a technique that is based on incremental process [18]. In 2001, Agile Manifesto was proposed by enthusiasts in the field. It describes the foundations of agile methods that were stated as follows: [22]

   - Individuals and interactions over processes and tools;

- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

3. **Scrum**

Scrum mainly tackles the use of iterative development by implementing incremental durations called Sprints. Sprints are timeboxed which means that they cannot be extended even if the work required is done or not. At the beginning of each Sprint the Scrum team is committed to complete these requirements, given and prioritized by the Product Owner, each sprint. At the end of each Sprint, there will be a Sprint Review together with the Product Owner and some stakeholders. The team will demonstrate what is built in the whole Sprint. The reviews in each Sprint Review will be incorporated to the next Sprint. For Scrum considered to be done, the product produced should be fully integrated, tested and potentially shippable or a definition of done given by the Product Owner should be available. [2]

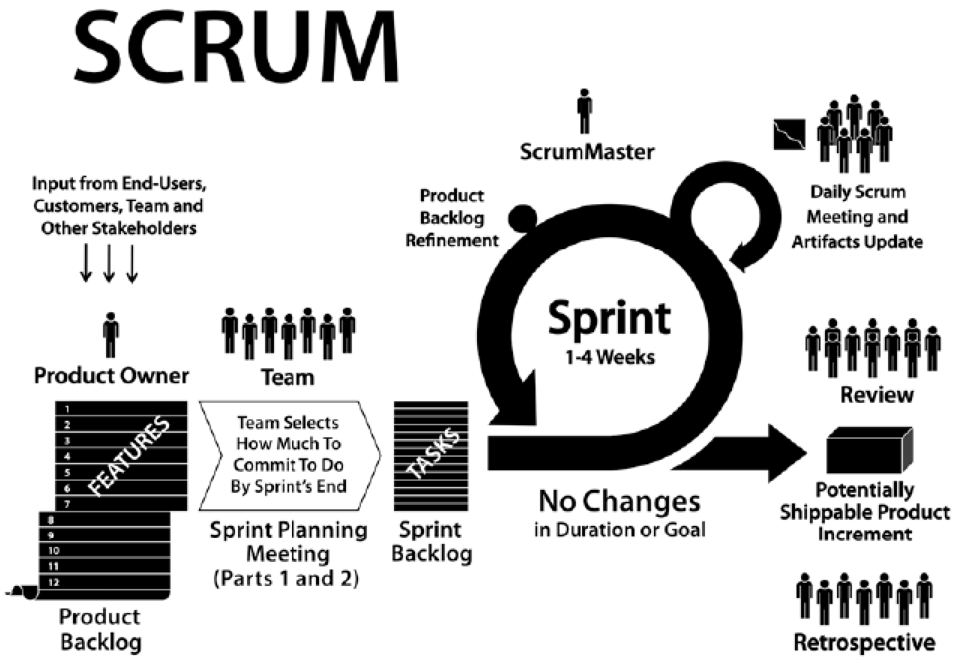

SCRUM
ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Product Owner
Team
Sprint
1-4 Weeks
Review
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
TASKS
Sprint Planning Meeting (Parts 1 and 2)
Sprint Backlog
No Changes in Duration or Goal
Potentially Shippable Product Increment
Product Backlog
Retrospective

Figure 1: Scrum

There are three important roles in Scrum that makes the Scrum Team. These are:

- **Product Owner**

  Product Owner is responsible in providing the projects requirements and maintains it by storing it as user stories in a Product Backlog. The Product Owner is also the one who prioritizes each user stories needed in each sprint and continues reprioritizing everything every new Sprint. In some cases Product Owner and Customer is only one person. In Scrum, Product Owner is the one who is responsible for the value of work.

- **Development Team**

  The development team is the one who develops the product the Product Owner needs. It is cross-functional and self-organizing. Cross-functional means that the team is composed of skilled persons needed to develop the product. Self-organizing means that the team can manage its own with a very high degree of autonomy and accountability. Each member of the team is the one who chooses what task/s they will be doing and how much time they need to deliver the committed tasks.

- **Scrum Master**

  Scrum Master is a member of the Development Team and helps them to attain business value. He/She is the bridge between the team and the Product Owner. Scrum Master is not a manager but instead the one who helps both the team and Product Owner to understand each others needs and reports. The Scrum Master is the one who facilitates the process, helps the team to easily organize and manage itself.

The first step in Scrum is listing of the Product Owner of all the requirements or the product vision. Afterwards, the Product Backlog will be available for the list of all the requirements ordered by prioritization. The Product Backlog includes all the user stories primarily the customer features, improvement goals, and, possibly known defects. It is continuously updated by the Product Owner every time theres a new idea, changes from the customer, or impediments that hinder the specific user story from completing. But these changes cannot be applied immediately especially if theres an ongoing sprint. Prioritizing user stories is the main responsible of the Product Owner but theres no such technique defined in Scrum for prioritizing things. Usually its done by Expert Judgment, self-intuition or through dependency of each feature.

There are meetings and key phases that are required in Scrum. These are:

- **Sprint Planning**
  - **Sprint Planning I**

    This part focuses on what output will the Product Owner want for the upcoming sprint. The Product Owner will set duration for the Sprint. This meeting is where the Product Owner and the Team will discuss what output is needed at the end of the Sprint. The Product Owner will present the User Stories needed for the sprint and will store it in the backlog.
  - **Sprint Planning II**

    - After the first part of the Sprint Planning, the team will now plan and decide how much amount of work will be devoted in that sprint. The team may choose what user stories they will work on for the sprint if they think that the current user stories are too many for the sprint duration. In this case, its a more reliable commitment since the last word is up to the team. Usually, the team will also plan how many hours they can devote for the whole Sprint. After determining the capacity, the team will now decompose the user stories into tasks. The team will now estimate how many hours they think they will finish a certain task. One technique for estimating is the Poker Planning, where each team member has cards that contain the values of Fibonacci numbers indicating the number of hours.

- **Daily Scrum Meeting** - The daily scrum meeting happens when the team started the sprint. This is a short (10-15 mins) meeting that happens every workday given a certain time. In this meeting, each member needs to report and answer three important questions to the whole team. These are:
  (a) What have you accomplished in the last scrum meeting?
  (b) What are you planning to do or finish for the next scrum meeting?
  (c) If possible, is there any impediment during developing a certain task?
- **Updating Sprint Backlog and Burndown Chart** - Since development teams are self-organizing, it is important to track what theyre doing. At the end of each working day, each member of the team will update the Sprint Backlog how many hours they committed to the task theyre currently doing. Following this update, the total hours will be plotted on the Sprint Burndown Chart. This chart overviews the current progress of the team. Its a downward sloping graph that is trajectory to reach zero effort remaining by the last day of the Sprint. The important point of the burndown chart is to show the teams progress towards the goal.
- **Sprint Review**

  Sprint Review happens after the Sprint. In this part, the team will discuss the output of the sprint to the Product Owner. The Product Owner will now judge if theres a need for change or any revisions. The Scrum Master guides the team on what tasks are done for the Sprint. Tasks that are not yet done during the sprint will be in the Product

Backlog again and will be reprioritized by the Product Owner. Other stakeholders can also attend the Sprint Review for their suggestions.

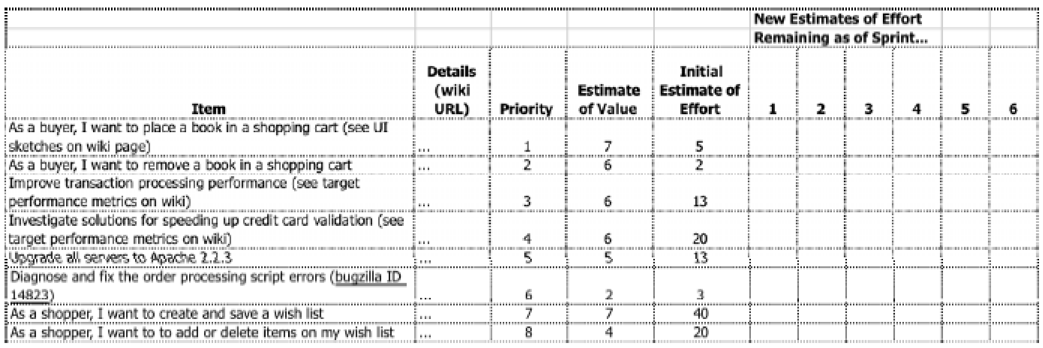
- **Updating Sprint Backlog**

  At this point for sure there are changes or addition in the user stories in the product backlog. The Product Owner is responsible in updating the Product Backlog.

Aside from these practices, there are also some terminologies and tools that are encountered in Scrum framework.

- **Product Backlog**

  It contains the list of overall requirements written as user stories provided by the Product Owner. The list may change over time due to some reasons.



					New Estimates of Effort Remaining as of Sprint...					
Item	Details (wiki URL)	Priority	Estimate of Value	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	...	1	7	5						
As a buyer, I want to remove a book in a shopping cart	...	2	6	2						
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13						
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20						
Upgrade all servers to Apache 2.2.3	...	5	5	13						
Diagnose and fix the order processing script errors (bugzilla ID 14823)	...	6	2	3						
As a shopper, I want to create and save a wish list	...	7	7	40						
As a shopper, I want to to add or delete items on my wish list	...	8	4	20						

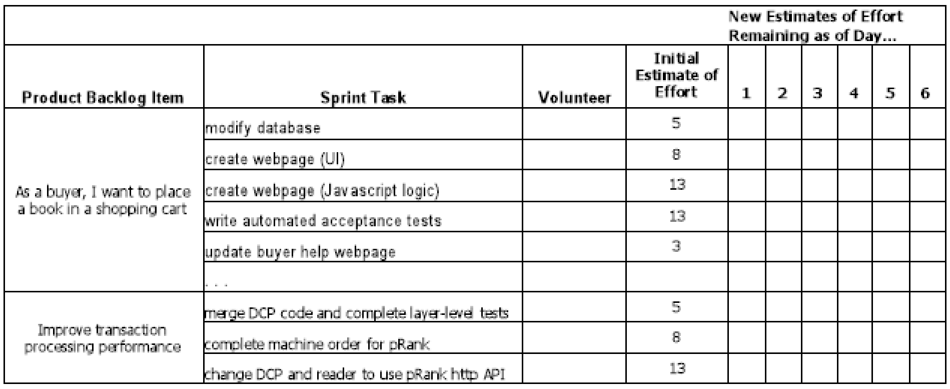
Figure 2: Product Backlog

- **Product Backlog Item/ User Story**

  A project requirement that is a functional requirement, non-functional requirement, or issue. A user story is provided by the product owner.

- **Sprint Backlog**

  It contains the list of the tasks for the whole Sprint. Tasks are created based on each of the user stories. Each of the tasks includes the one who volunteers to do the task and the estimated time planned by the team to finish it.



				New Estimates of Effort Remaining as of Day...					
Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

Figure 3: Sprint Backlog

- **Sprint Backlog Task**

  A task that the team defined based on a specific user story in the product backlog.

- **Burndown Chart**

  The chart which shows the trend of work to the remaining time in a Sprint. Trend will be based on the Sprint Backlog where the vertical axis refers to the amount of work/effort left versus the amount of days left before the closing of the sprint.

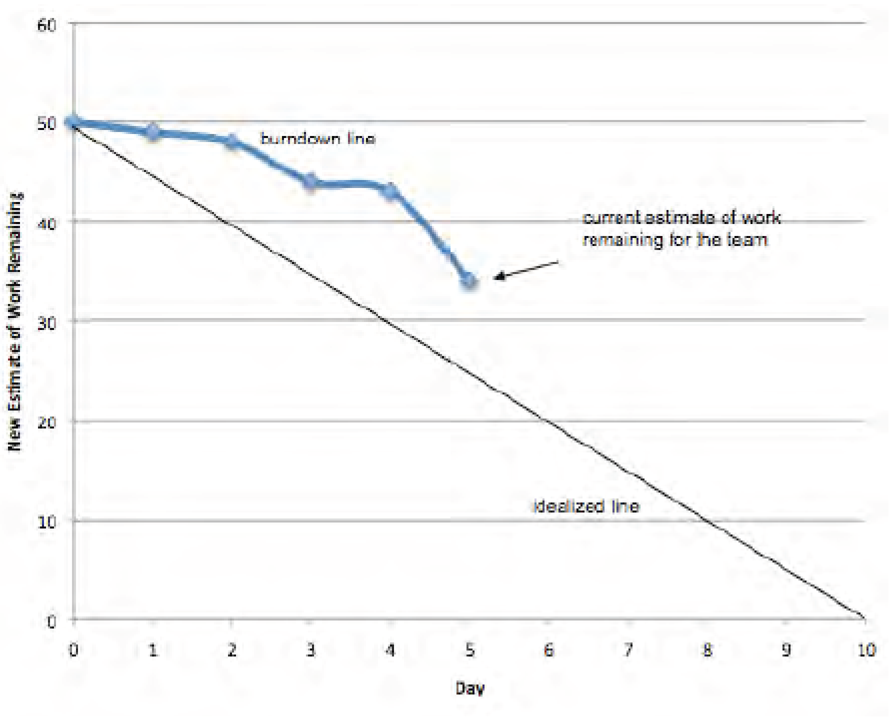


Figure 4: Burndown Chart

- **Scrum Board**

  A visual task-tracking tool that contains the current tasks usually written in post-its and is grouped thru the table that labeled Not yet started, In Progress and Completed.

IV. Design and Implementation

1. **Context Diagram**

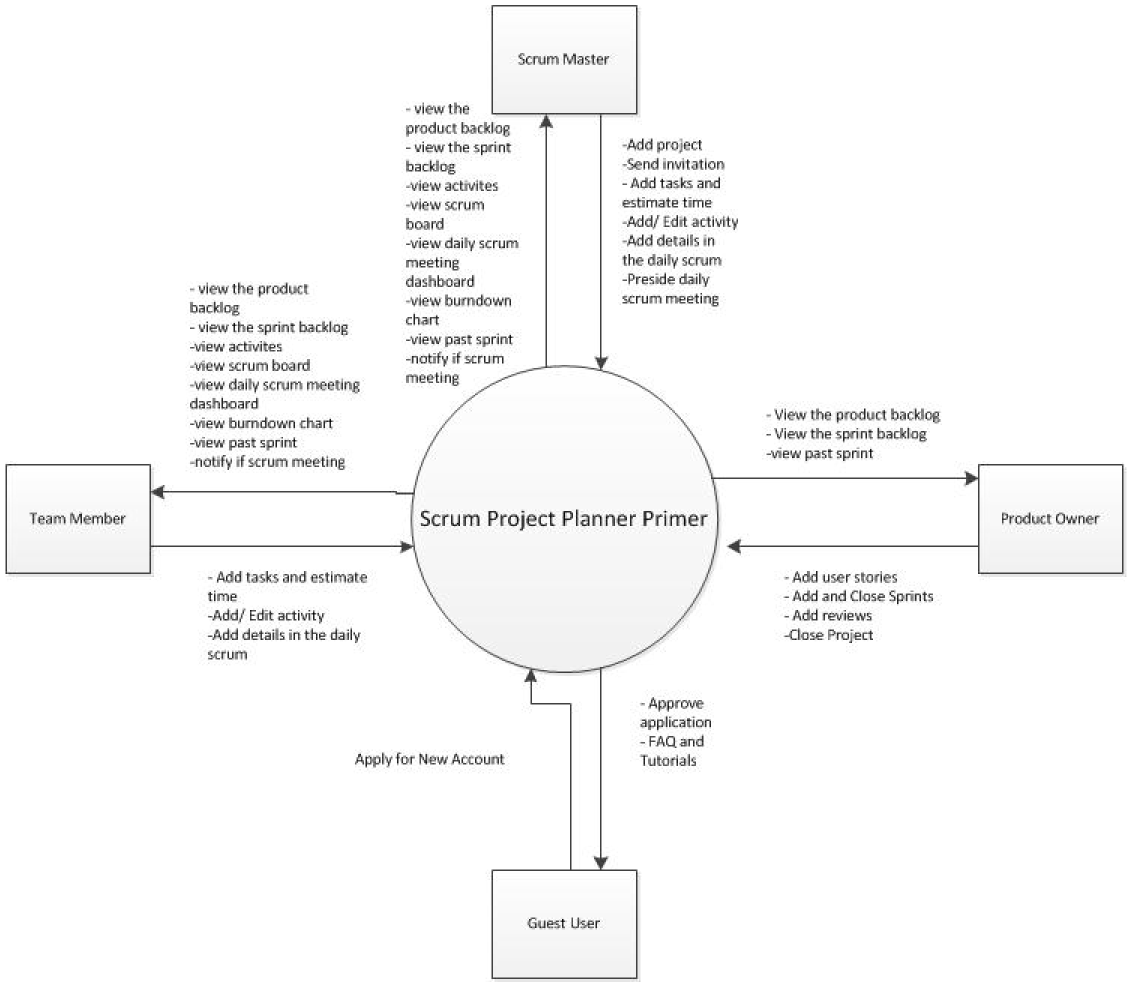


Figure 5: Context Diagram of Scrum Project Planner Primer

The context diagram shown above represents the overview of the tool. The project manager will add a project then he/she will send an invitation to his/her own members including the product owner through the system. After the members received the invitation and register to the system, the project manager will be notified for those who already joined.

Before a new sprint opens, the product owner is responsible in updating the product backlog by adding the requirements named as user stories. Then he/she will select the desired user stories need to be done for the upcoming sprint. After selecting the desired user stories, the product will set the duration of the sprint specifically the date of the sprint review.

The team will now create the tasks and will plan the estimated hours through poker planning. After the tasks were created and added in the sprint backlog, the team can now start the development.

On a daily basis, each day the team will update the tool on what are their current development in each task they handle. Every update of activities will reflect on the baseline of the tasks as well as the formulation of the burndown chart. For overview, the system will also provide a scrum board to know if which tasks are not yet done. The system will notify the members if the project manager will now preside the scrum meeting in which they will answer the three fundamental questions in the dashboard.

The product owner will provide a sprint review about each accomplished tasks for the whole sprint and after this, the sprint will automatically be closed. The product will have an option to add a new sprint or close the current project.

2. **Use Case Diagram**

The system has 5 users namely: Scrum Master, Team Member, Product Owner, Public/Guest User and Administrator.

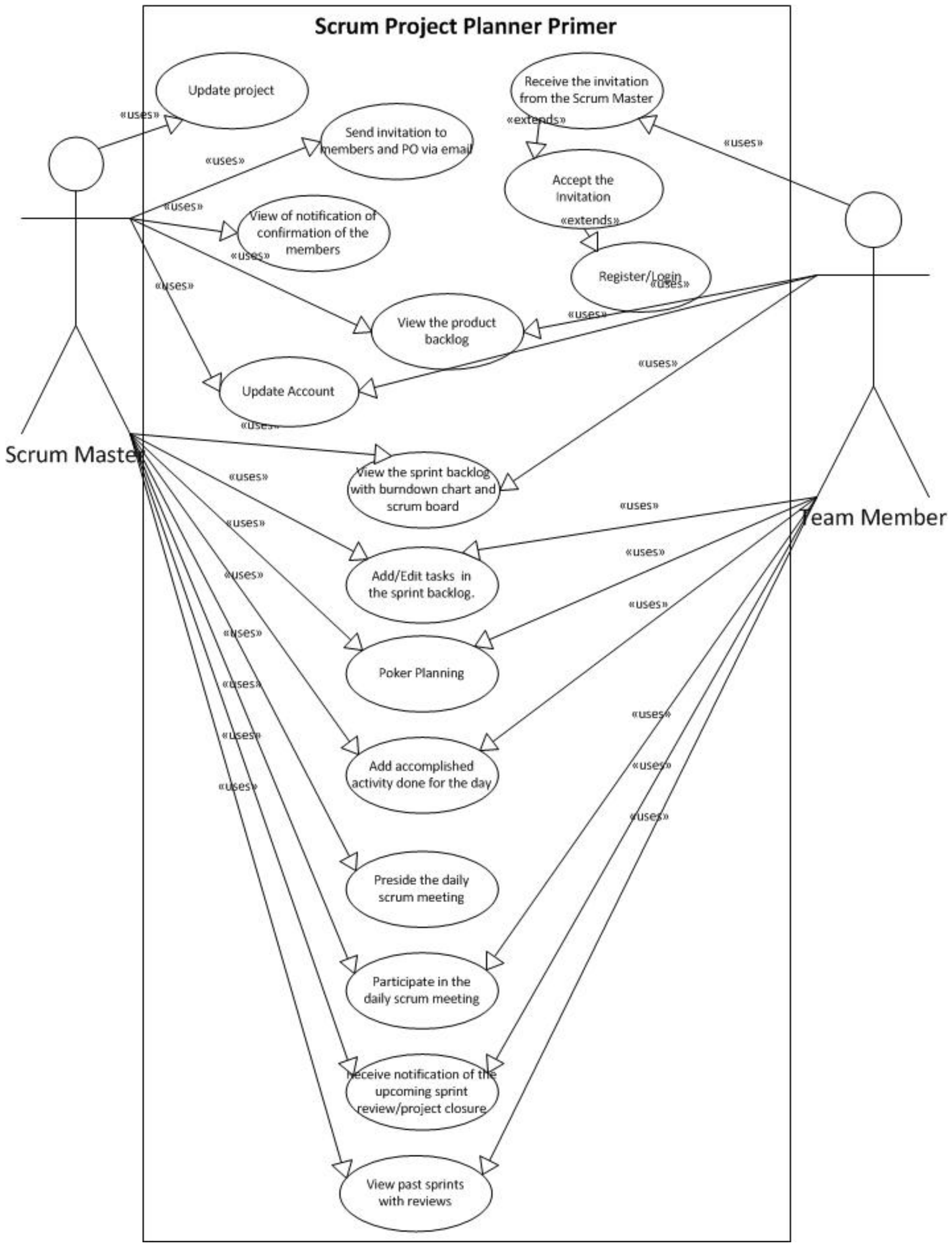

Scrum Project Planner Primer
Update project
Send invitation to members and PO via email
View of notification of confirmation of the members
View the product backlog
Update Account
Receive the invitation from the Scrum Master
Accept the Invitation
Register/Login
View the sprint backlog with burndown chart and scrum board
Add/Edit tasks in the sprint backlog.
Poker Planning
Add accomplished activity done for the day
Preside the daily scrum meeting
Participate in the daily scrum meeting
Receive notification of the upcoming sprint review/project closure
View past sprints with reviews
«uses»
«extends»
Scrum Master
Team Member

Figure 6: Use Case Diagram for Scrum Master and Team Member

The Scrum Master is also a member of the team but the difference is that they are the one who initiate on creating the project and inviting his/her own members including their product owner. All the members of the development team can use the tool as a progress tracking and participate in Scrum activities.

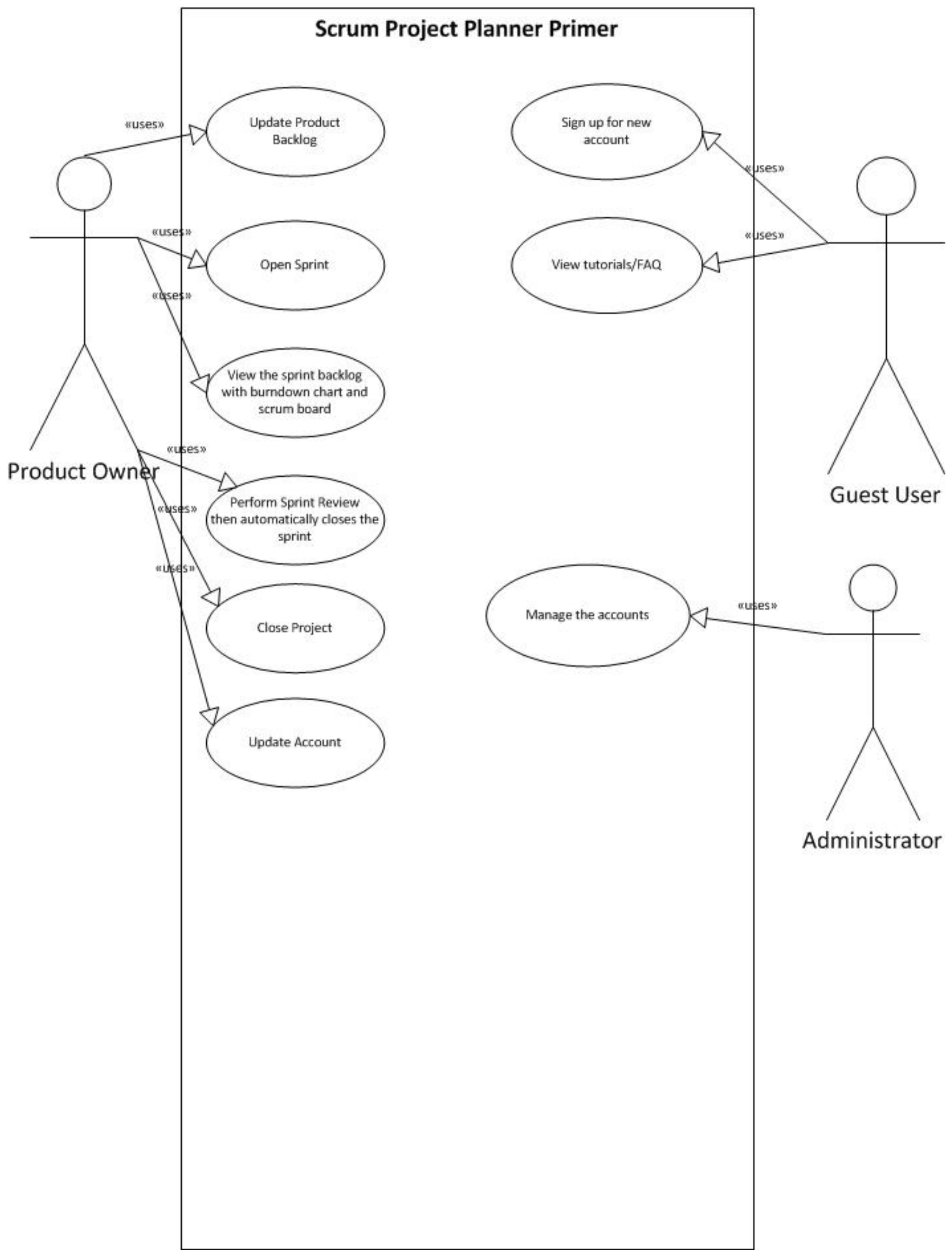


Figure 7: Use Case Diagram for Product Owner, Guest User and Administrator

The Product Owner is the one who provides the project requirements by storing it in the Product Backlog. He/She is also responsible in adding a sprint and selecting the desired user stories to be completed. His/her primary responsibility is to perform a Sprint Review and Closing the Project.

A guest user can sign up a new account or view the tutorials and frequently asked questions page which includes the some brief lecture about Scrum. System Administrator is responsible in managing the accounts.

3. **Flowchart with Swimlanes**

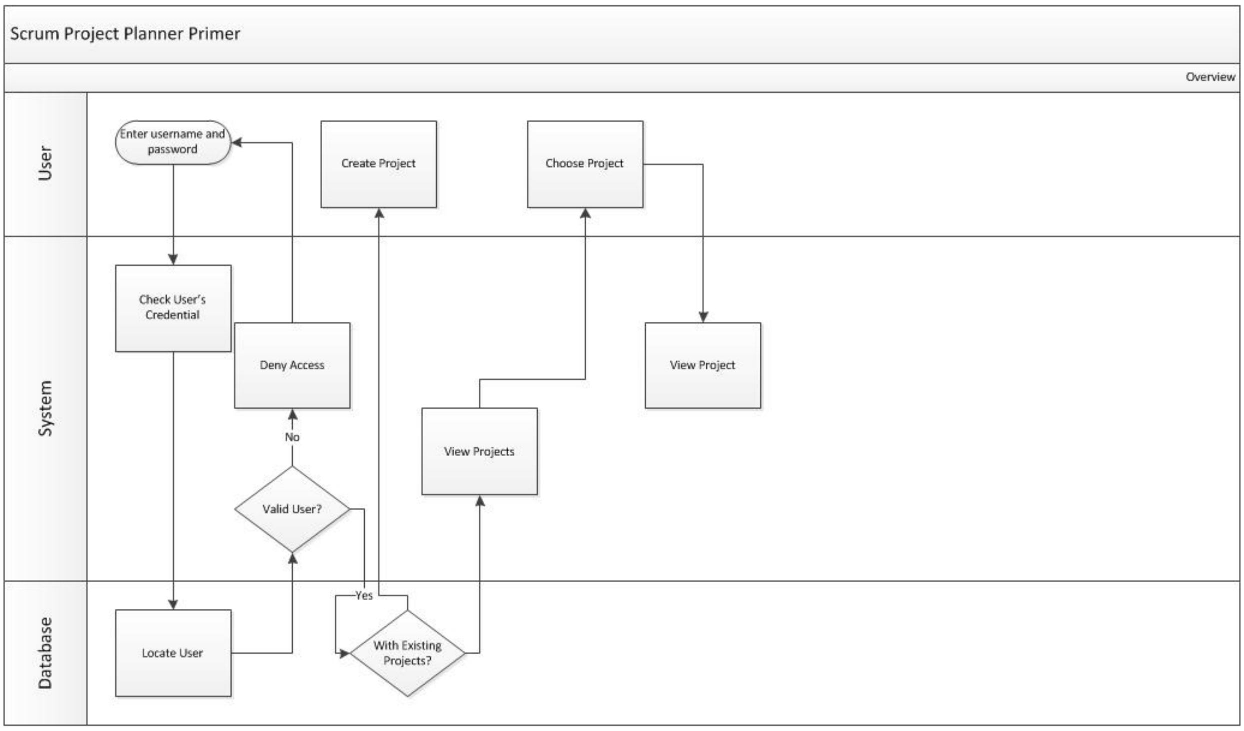


Figure 8: Overview

The user should enter his/her username and password first. The system will check if the input credentials are correct. If its correct, the user can view all his/her projects and choose what project he/she wants to enter. Otherwise, the user will be redirected to the log-in page of the tool and will be prompted Invalid Username or Password

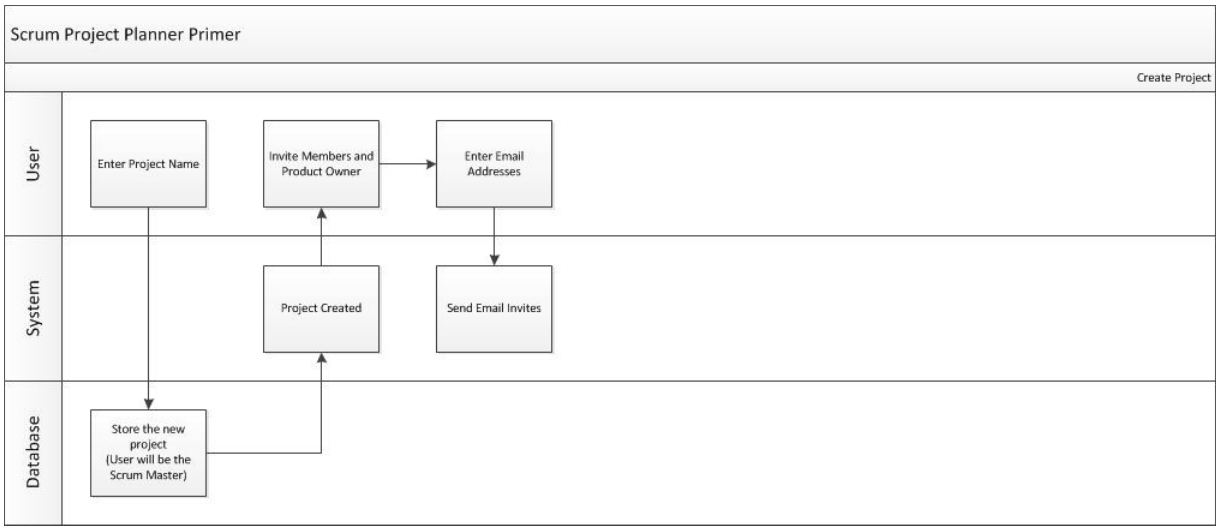

Scrum Project Planner Primer
Create Project
User
System
Database
Enter Project Name
Invite Members and Product Owner
Enter Email Addresses
Project Created
Send Email Invites
Store the new project (User will be the Scrum Master)

Figure 9: Create Project

If the user has no existing projects, he/she can add a new project. Note that the one who created the project will be the Scrum Master. After the project has been created, the user, now the Scrum Master, should enter the email addresses of the members of the project teams as well as the product owner. The system will send an invitation link to them.

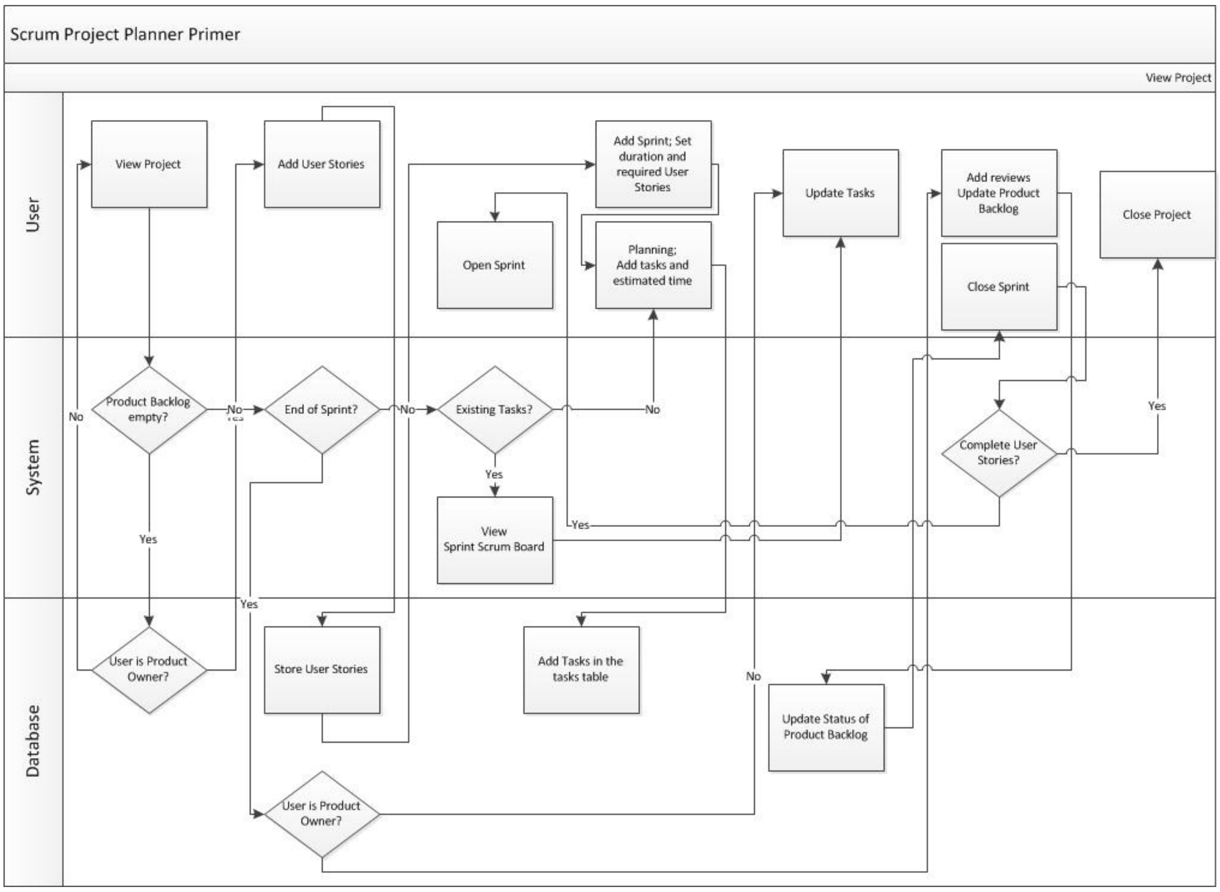


Figure 10: View Project

If the user chooses a project and that project does not have user stories yet, the Product Owner assigned should add user stories first before the team can proceed. Also, the Product Owner is the one who is responsible for adding a new sprint and setting its duration and filing up the user stories needed to accomplish for the sprint. After filling the Product Backlog, the team will now gather the required user stories and create tasks for each of it. After planning and the tasks are now ready, the team can now update daily the progress of the tasks in the sprint backlog. At the end of each sprint, the product owner should provide reviews and will decide if the certain user story was completed or not. The product owner is also the one who closes the sprint. And this process is repeated every iterations. The product owner is also the one who closes the project.

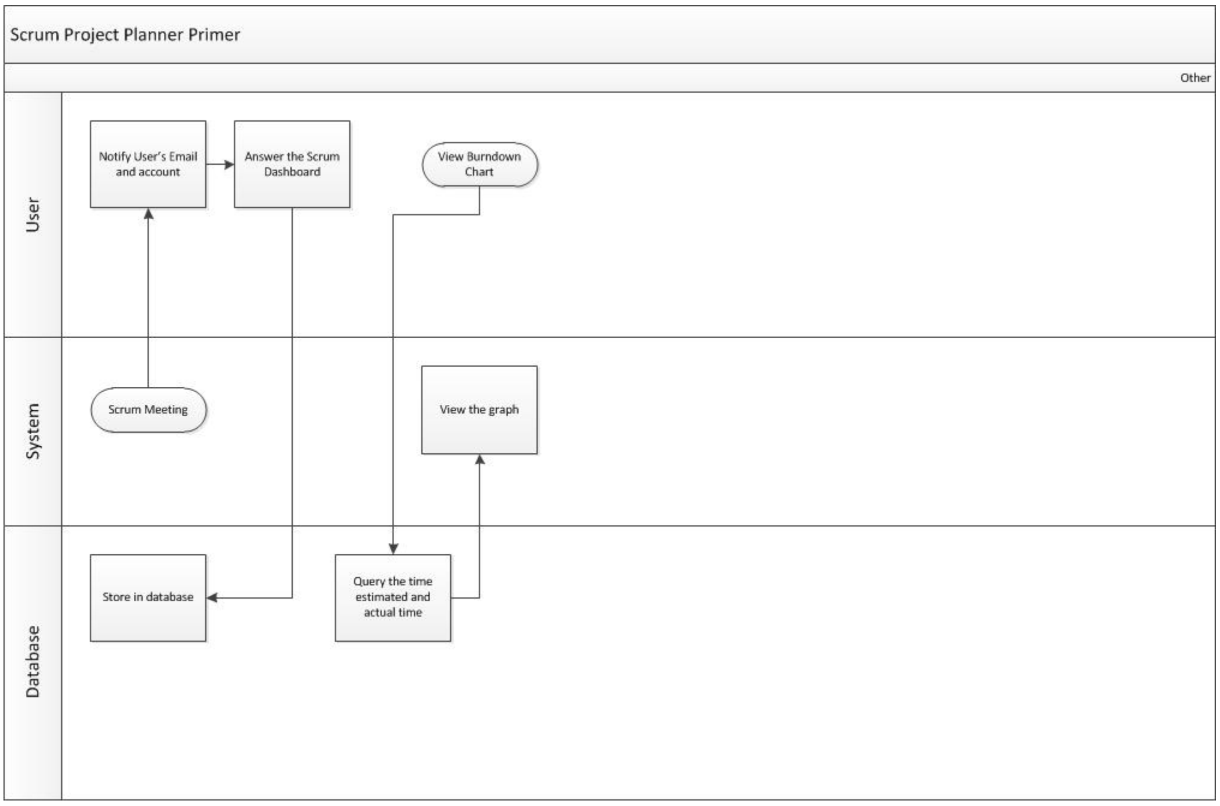

Scrum Project Planner Primer
Other
User
System
Database
Notify User's Email and account
Answer the Scrum Dashboard
View Burndown Chart
Scrum Meeting
View the graph
Store in database
Query the time estimated and actual time

Figure 11: Daily Scrum Meeting and Burndown Chart

Other feature of the tool is the daily scrum meeting where each of the members of the team will answer the three questions.Members can also view the Burndown Chart generated through the program.

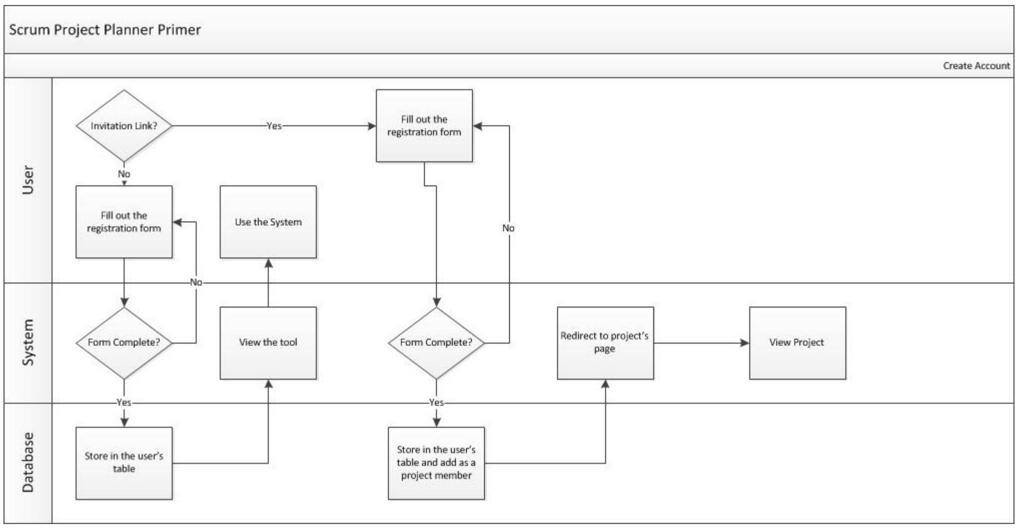

Scrum Project Planner Primer
Create Account
User
System
Database
Invitation Link?
Yes
No
Fill out the registration form
Use the System
Fill out the registration form
No
No
Form Complete?
View the tool
Form Complete?
Redirect to project's page
View Project
Yes
Yes
Store in the user's table
Store in the user's table and add as a project member

Figure 12: Create Account

For those who do not have an account but was given an invitation, after registering, the new user will also be added automatically in the project in which his/her invitation came from. Otherwise, regular registration will be followed.

4. **Entity Relationship Diagram**

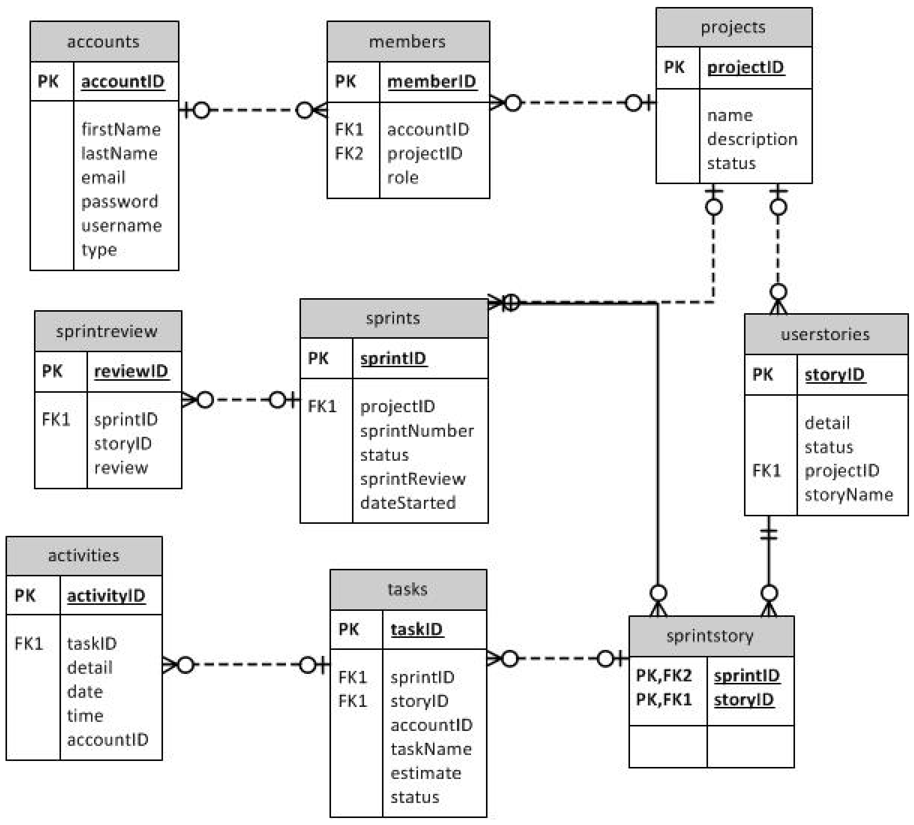


Figure 13: Entity Relationship Diagram

5. **Data Dictionary**

Attribute	**Data Type**	**Description**
accountID	int(11)	Primary key of accounts table
firstName	varchar(30)	First name of the user
lastName	varchar(30)	Last name of the user
email	varchar(30)	Email address of the user
password	varchar(100)	Encrypted password of the user
type	enum('admin', 'regular')	User type

Table 1: accounts Table

Attribute	**Data Type**	**Description**
projectID	int(11)	Primary key for projects table
name	varchar(50)	Project name
description	text	Project description
status	varchar(20)	Project status

Table 2: projects Table

Attribute	**Data Type**	**Description**
memberID	int(11)	Primary key for members
accountID	int(11)	Foreign key from account
projectID	int(11)	Foreign key from projects
role	enum('scrummaster', 'teammember', 'productowner')	Role in the project

Table 3: members Table

Attribute	**Data Type**	**Description**
storyID	int(11)	Primary key of userstories table
projectID	int(11)	Foreign key from projects table
detail	text	User story detail
storyName	varchar(50)	User story name
status	varchar(50)	User story status

Table 4: userstories Table

Attribute	**Data Type**	**Description**
sprintID	int(11)	Primary key for sprints table
projectID	int(11)	Foreign key from projects table
sprintNumber	int(11)	Sprint number of the project
status	varchar(20)	Sprint's status
sprintReview	date	Date of Sprint Review
dateStarted	date	Date that the Sprint Started

Table 5: sprints Table

Attribute	**Data Type**	**Description**
taskID	int(11)	Primary key of tasks table
sprintID	int(11)	Foreign key from sprints
storyID	int(11)	Foreign key from userstories
accountID	int(11)	AccountID of Volunteer
taskName	varchar(50)	Name of task
status	varchar(50)	Task's status
estimate	int(11)	Estimate of number of hours

Table 6: tasks Table

Attribute	**Data Type**	**Description**
activityID	int(11)	Primary key of activities table
detail	text	Activity' details
taskID	int(11)	Foreign key from tasks
date	date	Date of the activity
time	int(11)	Actual Number of hours
accountID	int(11)	Account ID of the doer of the activity

Table 7: activities Table

Attribute	**Data Type**	**Description**
reviewID	int(11)	Primary key of sprintreview
sprintID	int(11)	Foreign key from sprints
storyID	int(11)	Foreign key from userstories
review	text	Sprint review for certain user story

Table 8: sprintreview Table

Attribute	**Data Type**	**Description**
meetingID	int(11)	Primary key for scrummeeting
sprintID	int (11)	Foreign key from Sprints
accountID	int(11)	Account ID
accomplishment	text	Accomplishments
plan	text	Plan for next meeting
impediment	text	Current impediments
date	date	Date of Scrum Meeting

Table 9: scrummeeting Table

V. Architecture

A. System Architecture

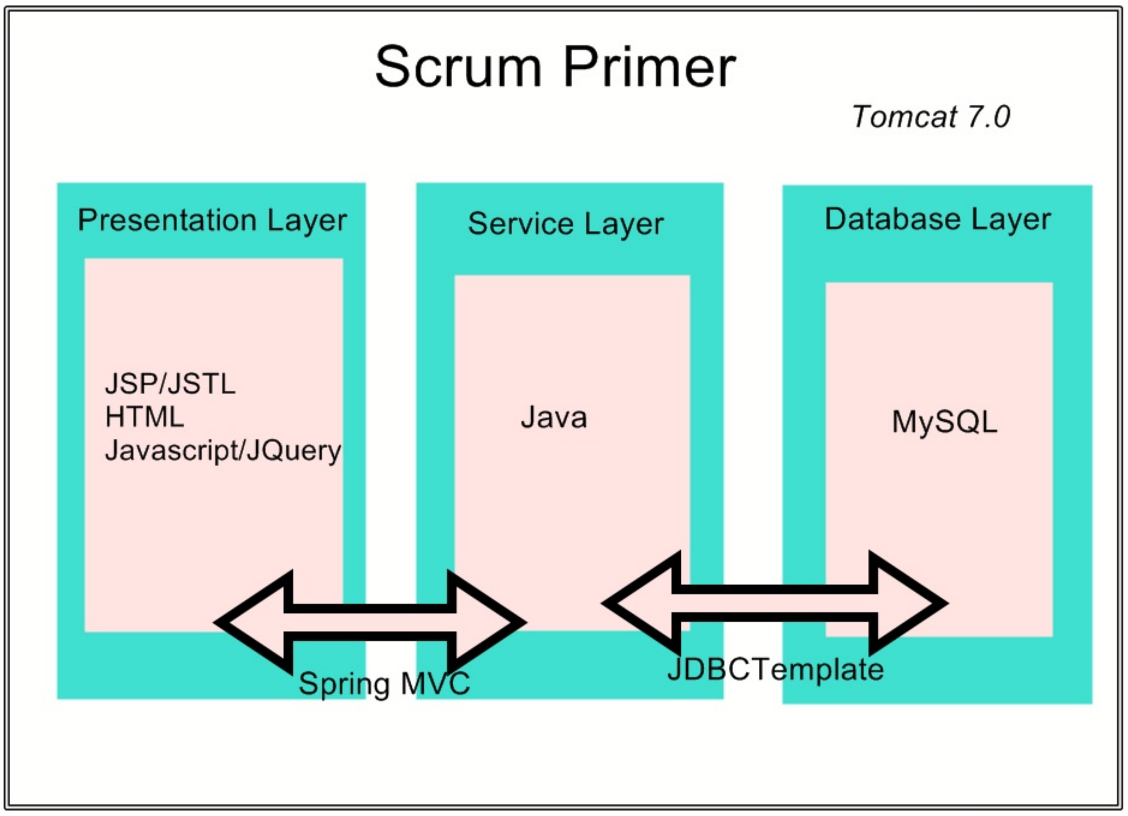


Figure 14: System Architecture

Scrum Project Planner Primer is a web-based tool that is composed of the three basic domains shown in the figure. For its presentation layer, HTML, JSP/JSTL, Javascript and CSS are used to encode the web interface. The three domains are interconnected through the Spring MVC framework. It is a framework for Java web applications. Unlike the usual java webapp in which different servlets are mapped, Spring has only one dispatcher servlet that contains the different annotations needed and handles the mapping of various pages.The database layer is made up of MySQL and is connected by JDBC.

B. Technical Architecture

Hardware requirements for the server:

- 1 GHz processor or higher
- At least 1GB of free disk space
- 1GB RAM or higher

Hardware requirements for the client:

- PC with Intel Pentium 4 or later
- At least 20MB of free disk space
- 128MB RAM or higher

The following software is needed by the server in order to run the Scrum Project Planner Primer

- Tomcat 7.0

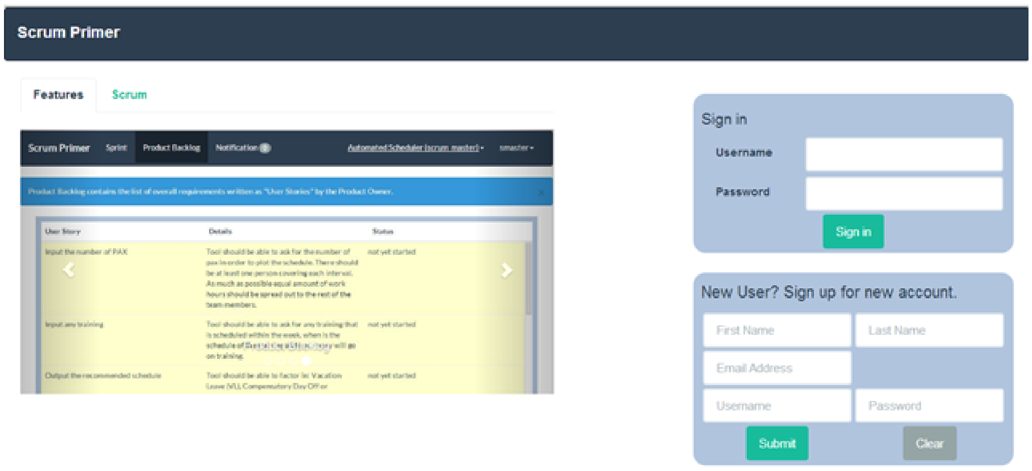
The client must have the following in order to run the tool

- Java Runtime Environment 6
- Web browser(Google Chrome 22.0.1229.94, Mozilla Firefox 16.0.1, Opera 12.02 or other browsers that support HTML5)

VI. Results

A. General View

1. Home Page

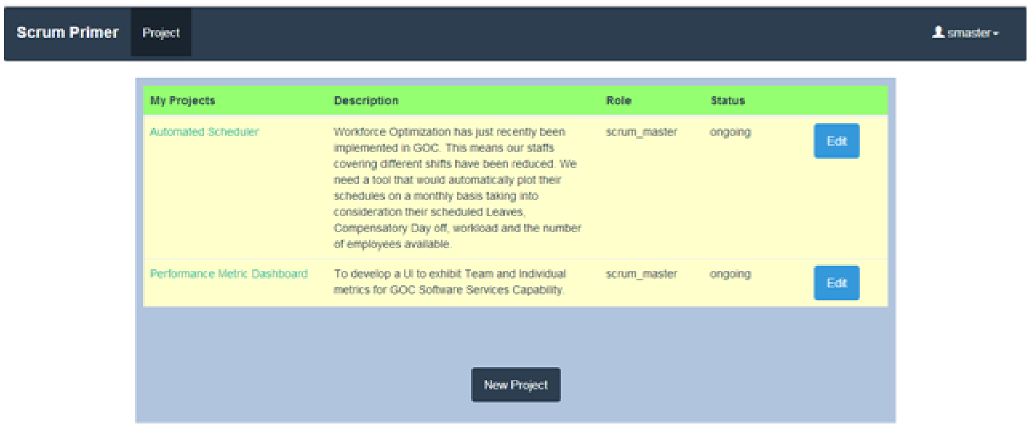


Product Backlog contains the list of overall requirements written as "User Stories" by the Product Owner.

User Story	Details	Status
Input the number of PAX	Tool should be able to ask for the number of pax in order to plot the schedule. There should be at least one person covering each interval. As much as possible equal amount of work hours should be spread out to the rest of the team members.	not yet started
Input any training	Tool should be able to ask for any training that is scheduled within the week, when is the schedule of training and who will go on training.	not yet started
Output the recommended schedule	Tool should be able to factor in: Vacation Leave (VL), Compensatory Day Off or	not yet started

Figure 15: Home Page

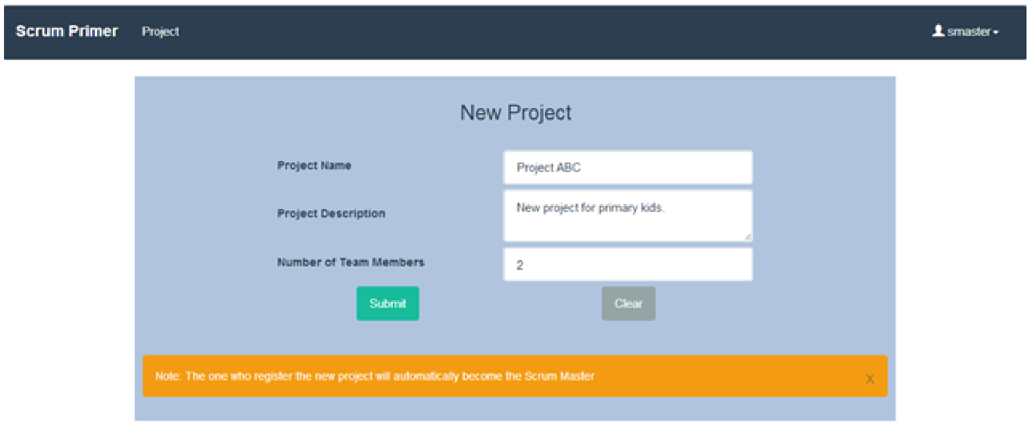
The home page of Scrum Project Planner Primer is displayed once the site is loaded. The user has an option to register or to login if he/she has an account.



My Projects	Description	Role	Status
Automated Scheduler	Workforce Optimization has just recently been implemented in GOC. This means our staffs covering different shifts have been reduced. We need a tool that would automatically plot their schedules on a monthly basis taking into consideration their scheduled Leaves, Compensatory Day off, workload and the number of employees available.	scrum_master	ongoing
Performance Metric Dashboard	To develop a UI to exhibit Team and Individual metrics for GOC Software Services Capability.	scrum_master	ongoing

Figure 16: Projects Page

After the user logged in, the page will redirect to the projects page where all projects involving the user are listed.In the projects page, the user's role per different projects are also listed. An option for creating new project is also available.


Scrum Primer Project

smaster

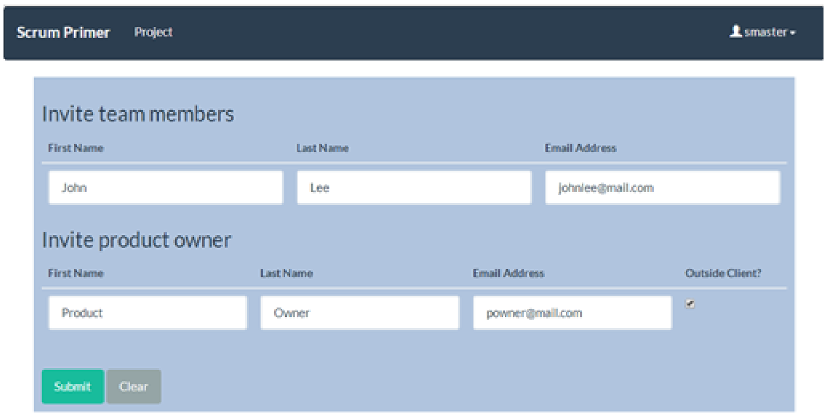
New Project

Project Name	Project ABC
Project Description	New project for primary kids.
Number of Team Members	2

Submit Clear

Note: The one who register the new project will automatically become the Scrum Master x

Figure 17: New Project Form


Scrum Primer Project

smaster

Invite team members

First Name	Last Name	Email Address
John	Lee	johnlee@mail.com

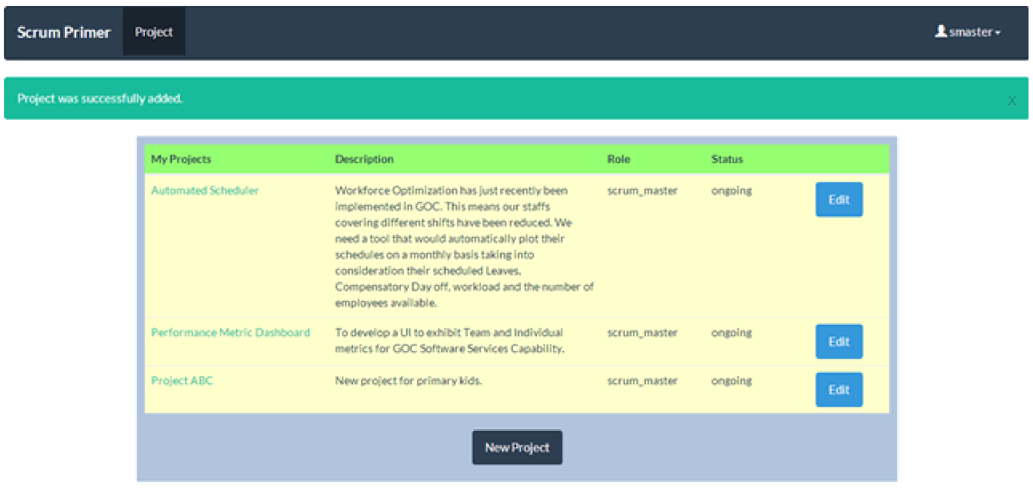
Invite product owner

First Name	Last Name	Email Address	Outside Client?
Product	Owner	powner@mail.com	✓

Submit Clear

Figure 18: Invitation to Members and Product Owner

It is noted that the one who will create the project will automatically become the scrum master. Aside from filling out the project details, the scrum master should also send invitation to his members including the product owner. Scrum master should know if the product owner of their team is a member of their domain.


Scrum Primer Project smaster

Project was successfully added.

My Projects	Description	Role	Status	
Automated Scheduler	Workforce Optimization has just recently been implemented in GOC. This means our staffs covering different shifts have been reduced. We need a tool that would automatically plot their schedules on a monthly basis taking into consideration their scheduled Leaves, Compensatory Day off, workload and the number of employees available.	scrum_master	ongoing	Edit
Performance Metric Dashboard	To develop a UI to exhibit Team and Individual metrics for GOC Software Services Capability.	scrum_master	ongoing	Edit
Project ABC	New project for primary kids.	scrum_master	ongoing	Edit

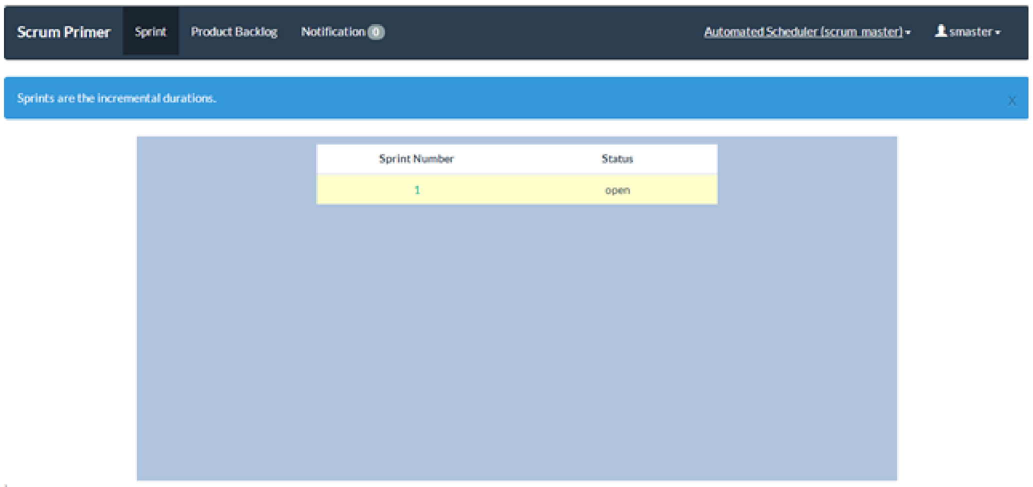
New Project

Figure 19: Projects page with prompt

After creating the new project, it will redirect again to the projects page including the newly added project. Each project name are links to the project's contents. The scrum master can edit the project details.

B. Scrum Role View

1. Scrum Master and Team Member

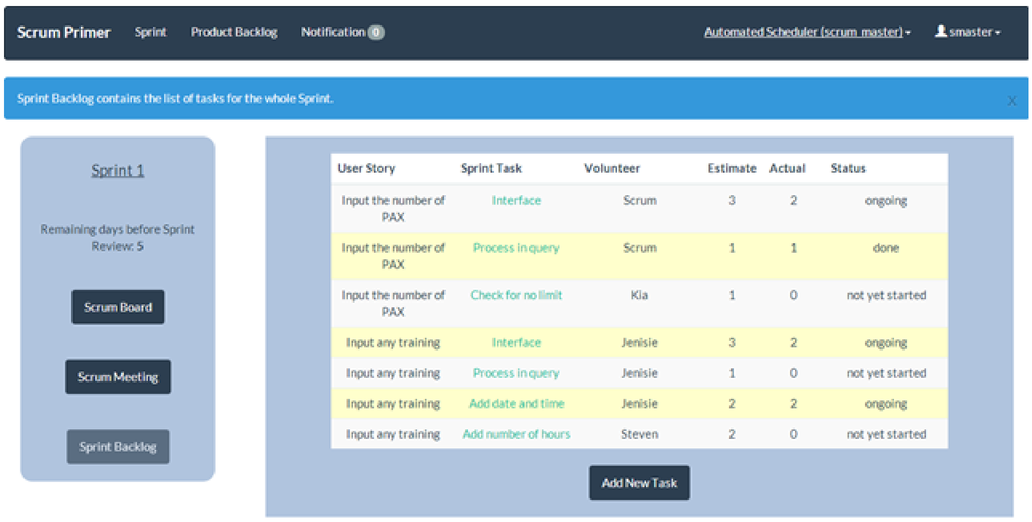

Scrum Primer Sprint Product Backlog Notification 0 Automated Scheduler (scrum master) smaster

Sprints are the incremental durations.

Sprint Number	Status
1	open

Figure 20: List of all the Sprint

A scrum master or a team member will be redirected to sprints page in which all the sprints of the project are listed. In the navigation bar, the username, the project and the user's role are indicated.



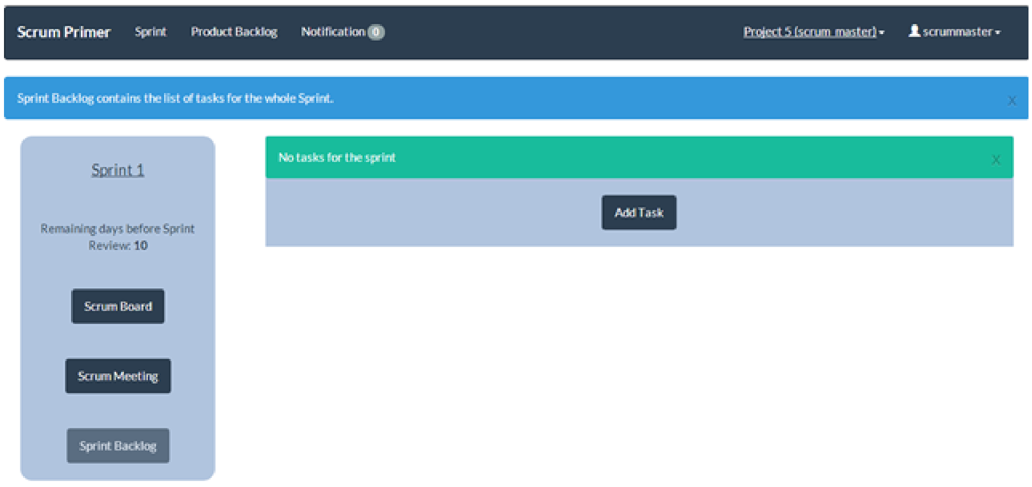
Sprint Backlog contains the list of tasks for the whole Sprint.

Remaining days before Sprint Review: 5

User Story	Sprint Task	Volunteer	Estimate	Actual	Status
Input the number of PAX	Interface	Scrum	3	2	ongoing
Input the number of PAX	Process in query	Scrum	1	1	done
Input the number of PAX	Check for no limit	Kia	1	0	not yet started
Input any training	Interface	Jenisie	3	2	ongoing
Input any training	Process in query	Jenisie	1	0	not yet started
Input any training	Add date and time	Jenisie	2	2	ongoing
Input any training	Add number of hours	Steven	2	0	not yet started

Figure 21: Sprint Backlog

The user can view the sprint backlog – list of all tasks, where he/she can view the activities under a certain task and he/she can also add a new task.

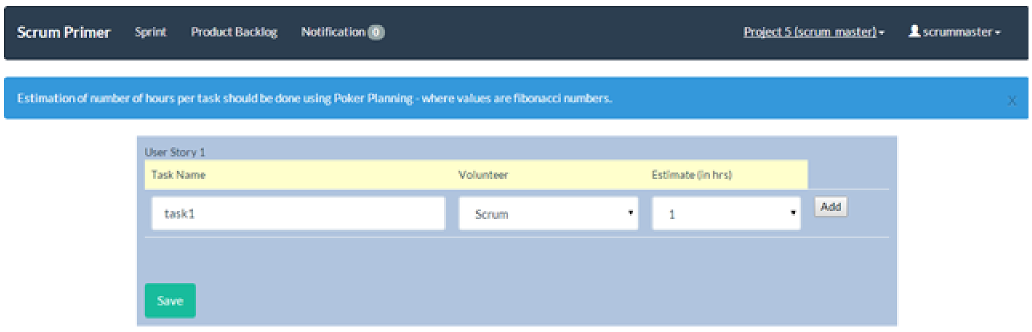


Sprint Backlog contains the list of tasks for the whole Sprint.

Remaining days before Sprint Review: 10

No tasks for the sprint

Figure 22: Indicating no tasks for the Sprint (Scrum Master)


Scrum Primer Sprint Product Backlog Notification 0 Project 5 (scrum master) scrummaster

Estimation of number of hours per task should be done using Poker Planning - where values are fibonacci numbers.

User Story 1

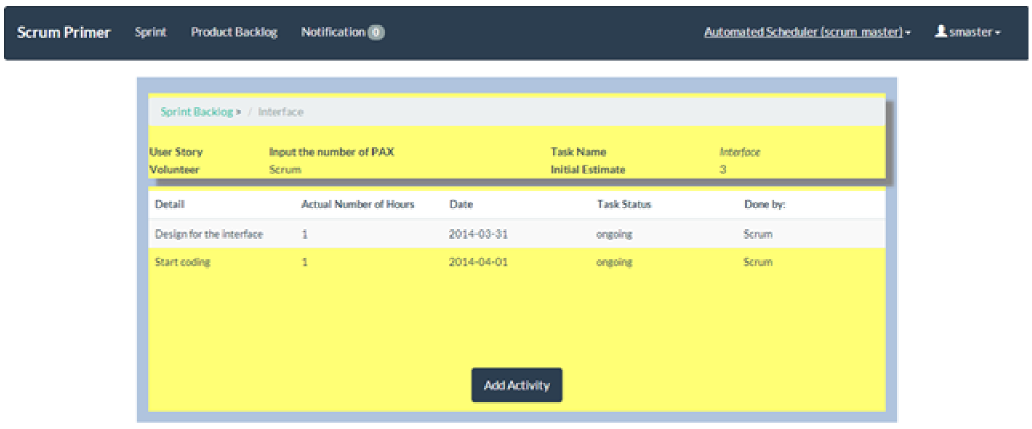
Task Name	Volunteer	Estimate (in hrs)
task1	Scrum	1

Add

Save

Figure 23: Add the planned tasks (Scrum Master)

If there's no planned tasks, the scrum master will add the tasks given the estimated number of hours. Estimation is based on poker planning where choice of estimates are fibonacci numbers.


Scrum Primer Sprint Product Backlog Notification 0 Automated Scheduler (scrum master) smaster

Sprint Backlog » / Interface

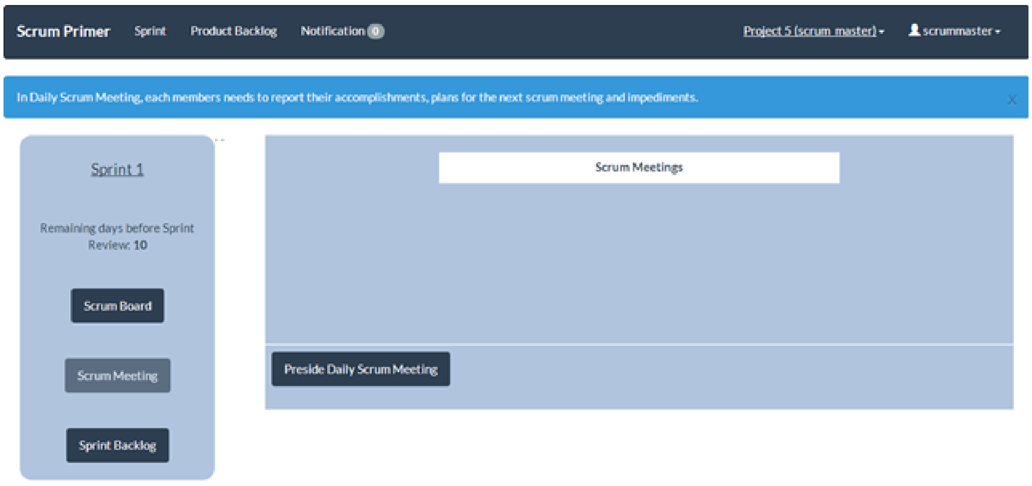
User Story	Input the number of PAX	Task Name	*Interface*
Volunteer	Scrum	Initial Estimate	3

Detail	Actual Number of Hours	Date	Task Status	Done by:
Design for the interface	1	2014-03-31	ongoing	Scrum
Start coding	1	2014-04-01	ongoing	Scrum

Add Activity

Figure 24: Activities for task

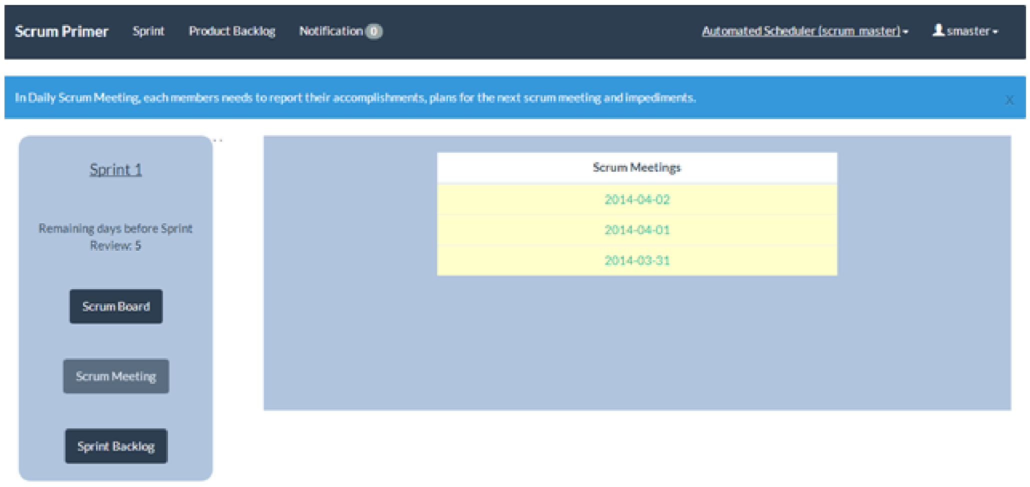
Under the activity page, the user can also add an activity for the certain day.



In Daily Scrum Meeting, each members needs to report their accomplishments, plans for the next scrum meeting and impediments.

Remaining days before Sprint Review: 10

Figure 25: Preside the daily scrum meeting (Scrum Master)



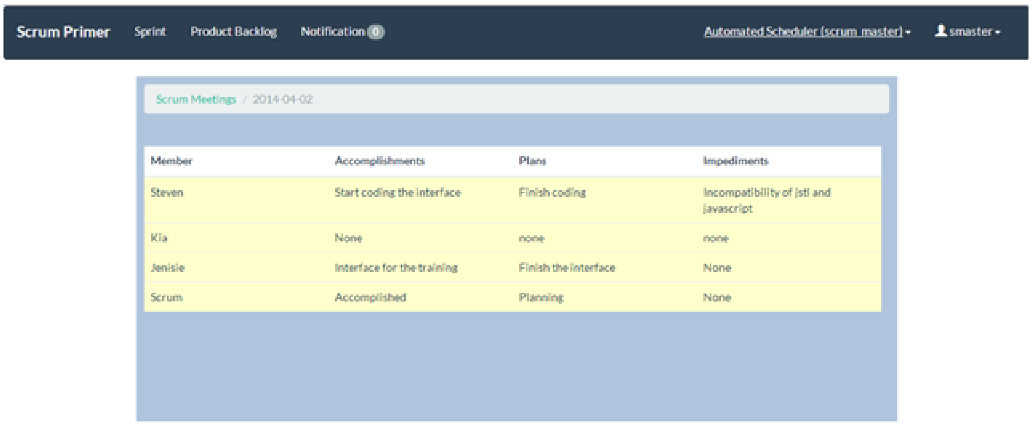
In Daily Scrum Meeting, each members needs to report their accomplishments, plans for the next scrum meeting and impediments.

Remaining days before Sprint Review: 5

Scrum Meetings
2014-04-02
2014-04-01
2014-03-31

Figure 26: List of Scrum Meetings of the Sprint

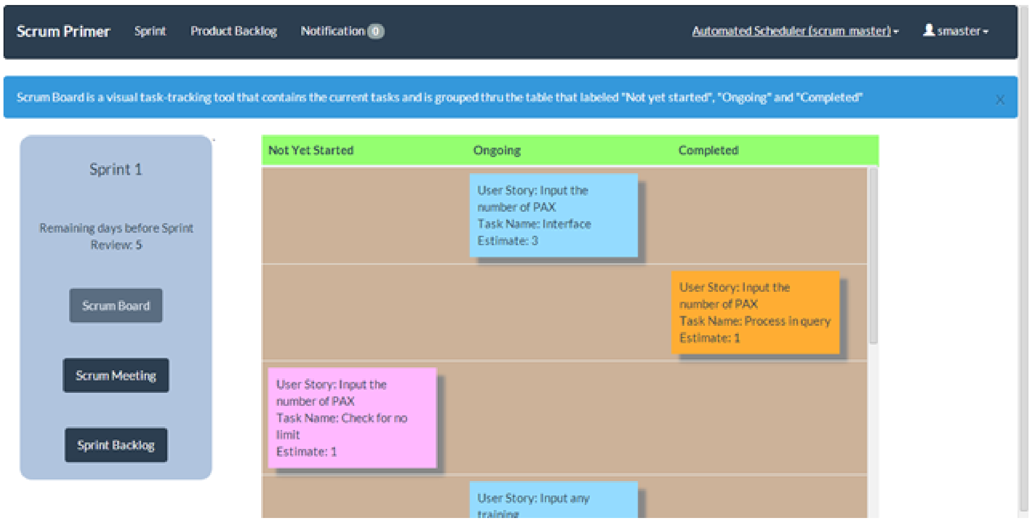
The user can also view the daily scrum meetings for the sprint. The scrum master is the one who will preside it. All the members of the development team will answer the three preliminary questions.


Scrum Primer
Sprint
Product Backlog
Notification 0
Automated Scheduler (scrum master)
smaster
Scrum Meetings / 2014-04-02

Member	Accomplishments	Plans	Impediments
Steven	Start coding the interface	Finish coding	Incompatibility of jstl and javascript
Kia	None	none	none
Jenisie	Interface for the training	Finish the interface	None
Scrum	Accomplished	Planning	None

Figure 27: Scrum Meeting Details

The answers of the members who participated in the meeting are stored per day.

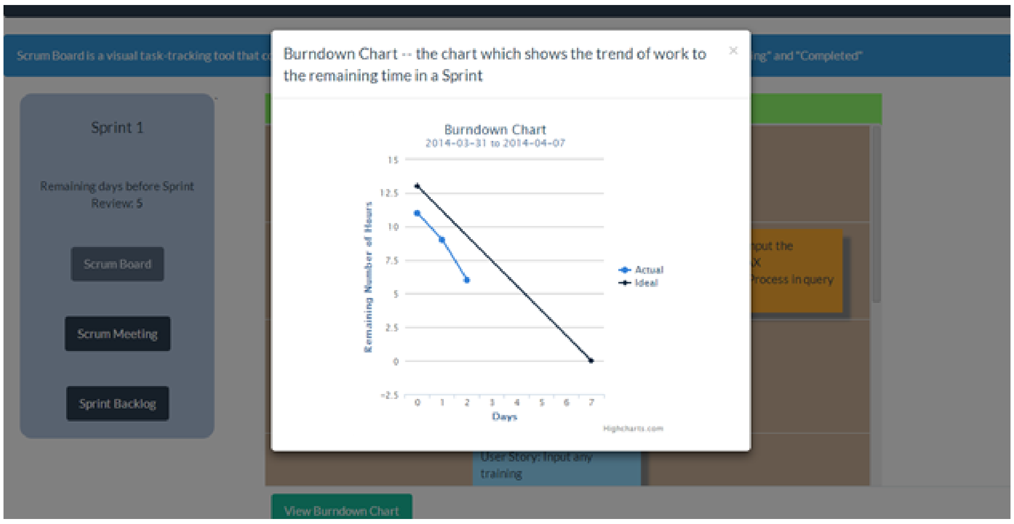


Scrum Board is a visual task-tracking tool that contains the current tasks and is grouped thru the table that labeled "Not yet started", "Ongoing" and "Completed"

Remaining days before Sprint Review: 5

Not Yet Started	Ongoing	Completed
	User Story: Input the number of PAX Task Name: Interface Estimate: 3	
		User Story: Input the number of PAX Task Name: Process in query Estimate: 1
User Story: Input the number of PAX Task Name: Check for no limit Estimate: 1		
	User Story: Input any training	

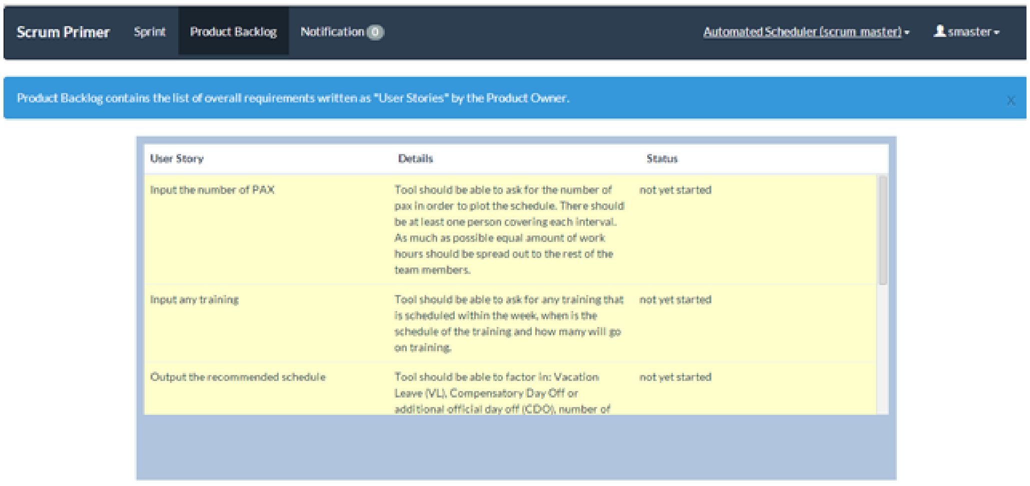
Figure 28: Scrum Board



Burndown Chart -- the chart which shows the trend of work to the remaining time in a Sprint

Figure 29: Burndown Chart

The user can also view the Scrum Board and the Burndown Chart. Scrum board groups the tasks according to its status. Burndown chart represents the trend of work against the estimated time.

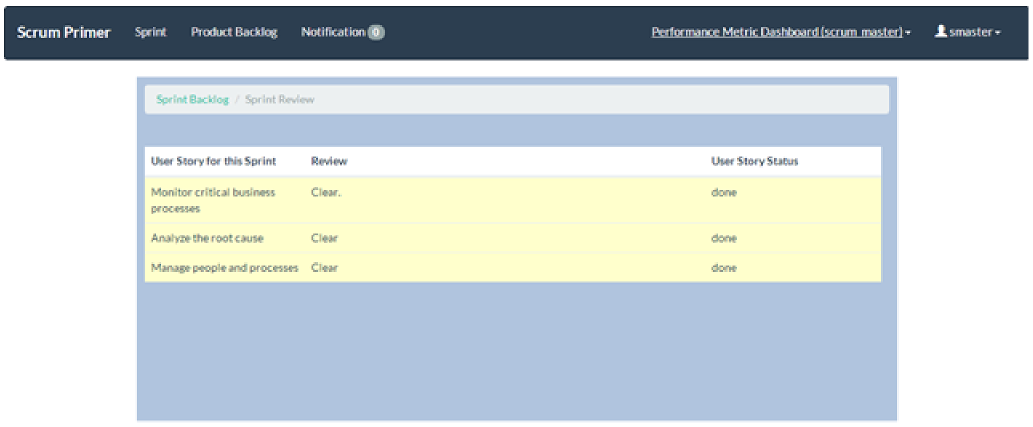


Product Backlog contains the list of overall requirements written as "User Stories" by the Product Owner.

User Story	Details	Status
Input the number of PAX	Tool should be able to ask for the number of pax in order to plot the schedule. There should be at least one person covering each interval. As much as possible equal amount of work hours should be spread out to the rest of the team members.	not yet started
Input any training	Tool should be able to ask for any training that is scheduled within the week, when is the schedule of the training and how many will go on training.	not yet started
Output the recommended schedule	Tool should be able to factor in: Vacation Leave (VL), Compensatory Day Off or additional official day off (CDO), number of	not yet started

Figure 30: Product Backlog

The user can also view the Product Backlog where all the requirements provided by the Product Owner are listed.

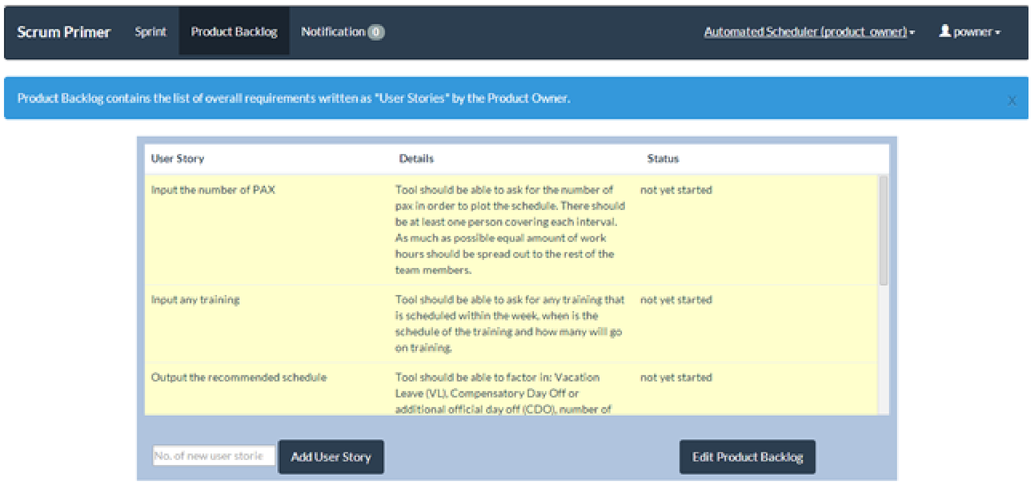

Scrum Primer
Sprint
Product Backlog
Notification 0
Performance Metric Dashboard (scrum_master)
smaster
Sprint Backlog / Sprint Review

User Story for this Sprint	Review	User Story Status
Monitor critical business processes	Clear.	done
Analyze the root cause	Clear	done
Manage people and processes	Clear	done

Figure 31: Sprint Review

For closed sprints, user can also view the sprint backlog, scrum board and sprint's daily scrum meetings. Alongside with the closed sprints are Product Owner's sprint reviews.

2. **Product Owner**

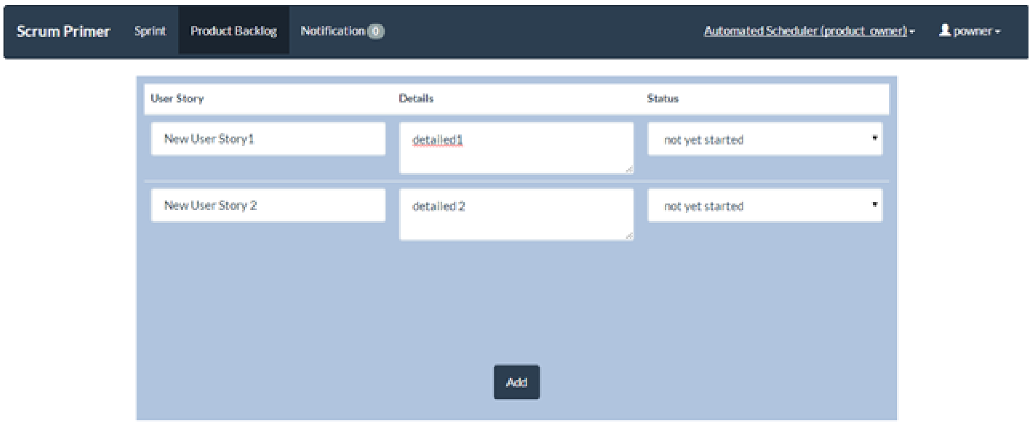


Product Backlog contains the list of overall requirements written as "User Stories" by the Product Owner.

User Story	Details	Status
Input the number of PAX	Tool should be able to ask for the number of pax in order to plot the schedule. There should be at least one person covering each interval. As much as possible equal amount of work hours should be spread out to the rest of the team members.	not yet started
Input any training	Tool should be able to ask for any training that is scheduled within the week, when is the schedule of the training and how many will go on training.	not yet started
Output the recommended schedule	Tool should be able to factor in: Vacation Leave (VL), Compensatory Day Off or additional official day off (CDO), number of	not yet started

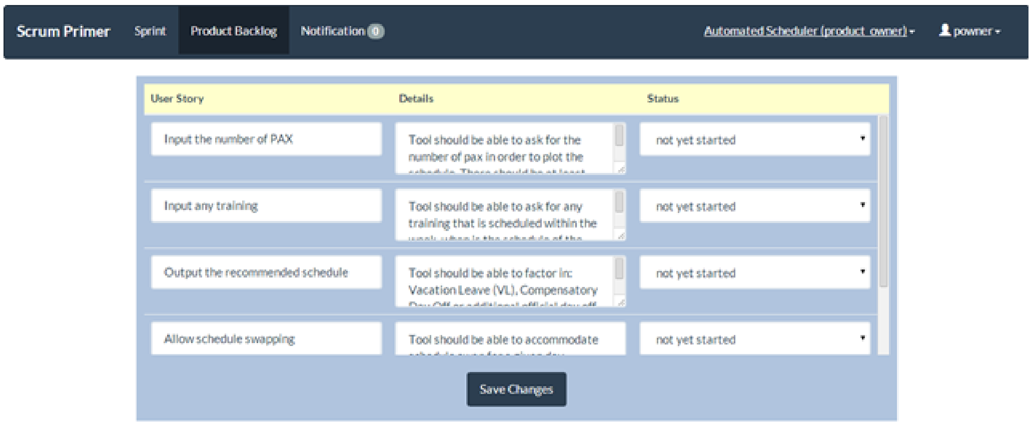
Figure 32: Product Backlog

The product owner will be redirected to the product backlog page where all the specification requirements are listed.



User Story	Details	Status
New User Story1	detailed1	not yet started
New User Story 2	detailed 2	not yet started

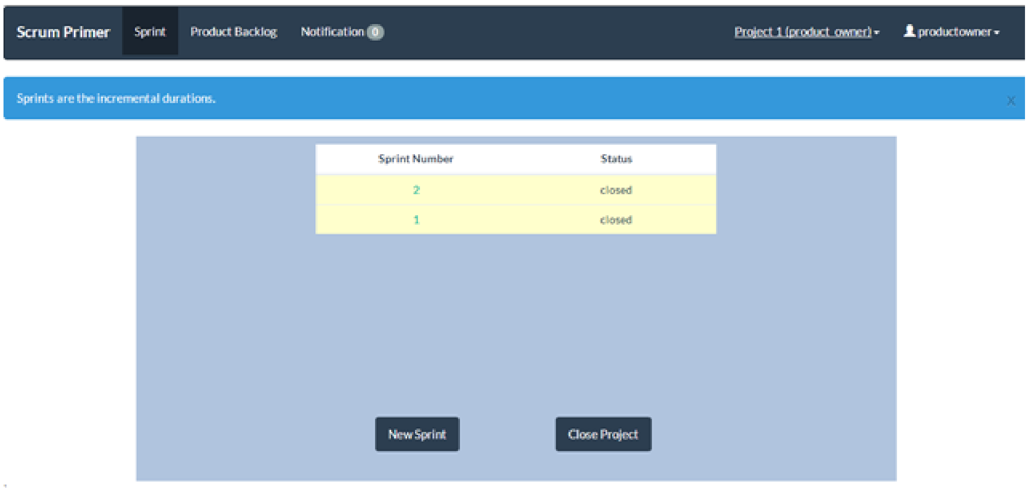
Figure 33: Add new user story



User Story	Details	Status
Input the number of PAX	Tool should be able to ask for the number of pax in order to plot the	not yet started
Input any training	Tool should be able to ask for any training that is scheduled within the	not yet started
Output the recommended schedule	Tool should be able to factor in: Vacation Leave (VL), Compensatory	not yet started
Allow schedule swapping	Tool should be able to accommodate	not yet started

Figure 34: Edit user stories

The product owner can add user stories or edit them.

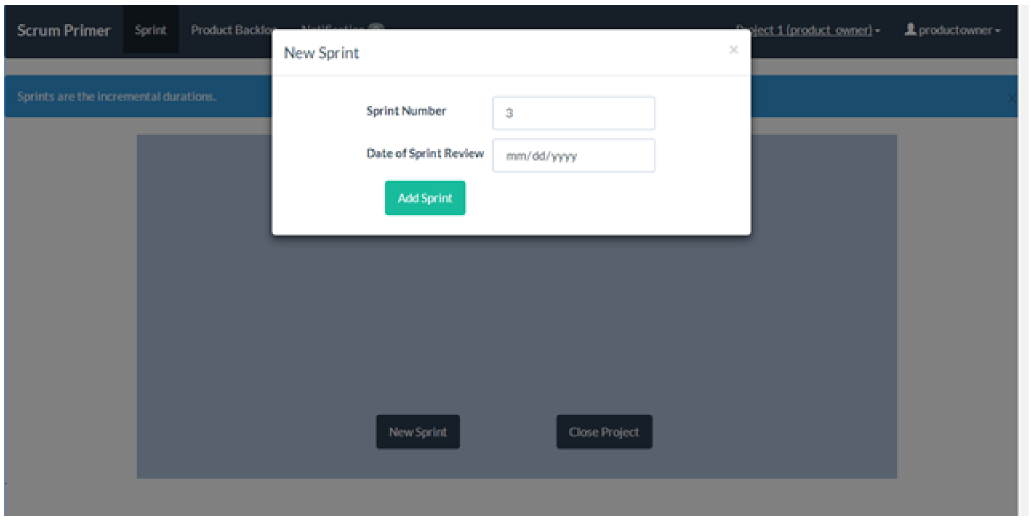


Sprints are the incremental durations.

Sprint Number	Status
2	closed
1	closed

Figure 35: List of Sprints (Product Owner)

In the sprints page, a product owner has an option to add a sprint or close the project.

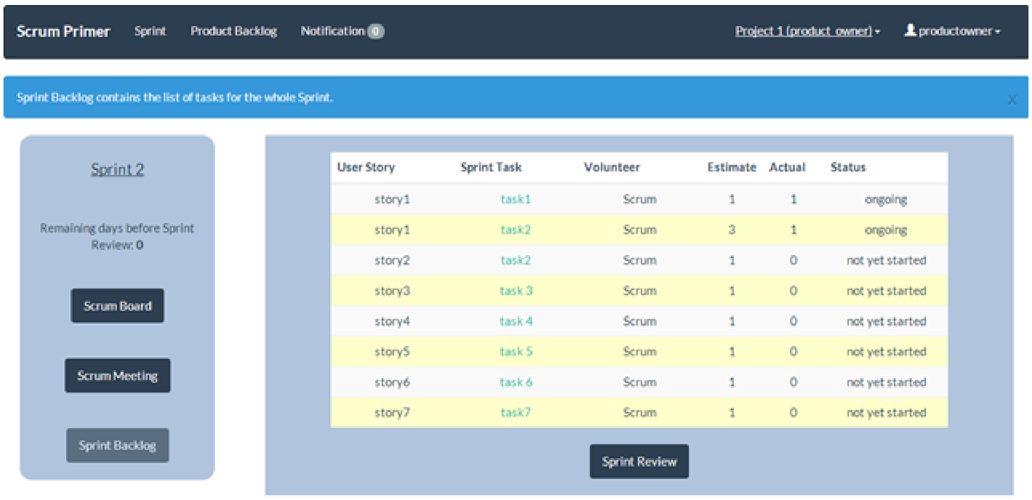


Sprint Number: 3

Date of Sprint Review: mm/dd/yyyy

Figure 36: New Sprint

Product owner is also responsible in opening a new sprint and setting its date of sprint review.



Sprint Backlog contains the list of tasks for the whole Sprint.

Remaining days before Sprint Review: 0

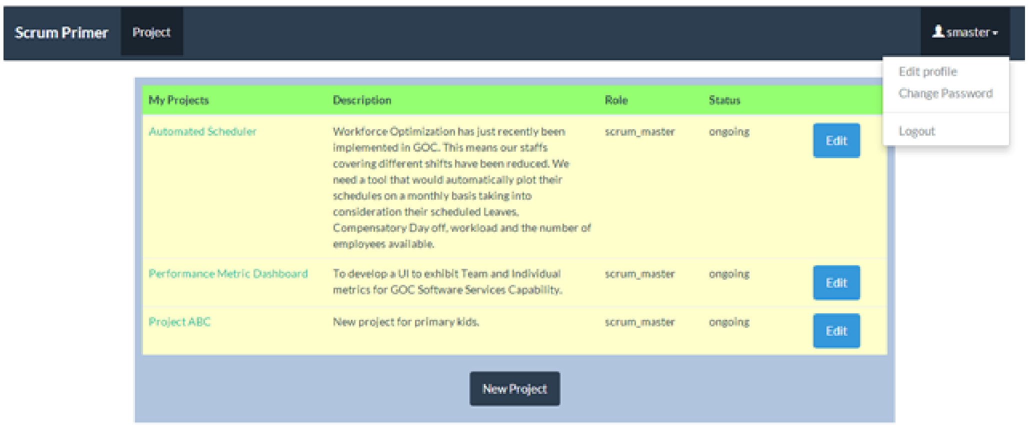
User Story	Sprint Task	Volunteer	Estimate	Actual	Status
story1	task1	Scrum	1	1	ongoing
story1	task2	Scrum	3	1	ongoing
story2	task2	Scrum	1	0	not yet started
story3	task 3	Scrum	1	0	not yet started
story4	task 4	Scrum	1	0	not yet started
story5	task 5	Scrum	1	0	not yet started
story6	task 6	Scrum	1	0	not yet started
story7	task7	Scrum	1	0	not yet started

Figure 37: Sprint Review by Product Owner

Just like the scrum master and team member, a product owner can also view the sprint backlog, daily scrum meeting, scrum board and burndown chart. When the date of sprint of review comes, the product owner is responsible in providing the reviews for the work done and closing the sprint.

C. Manage Accounts

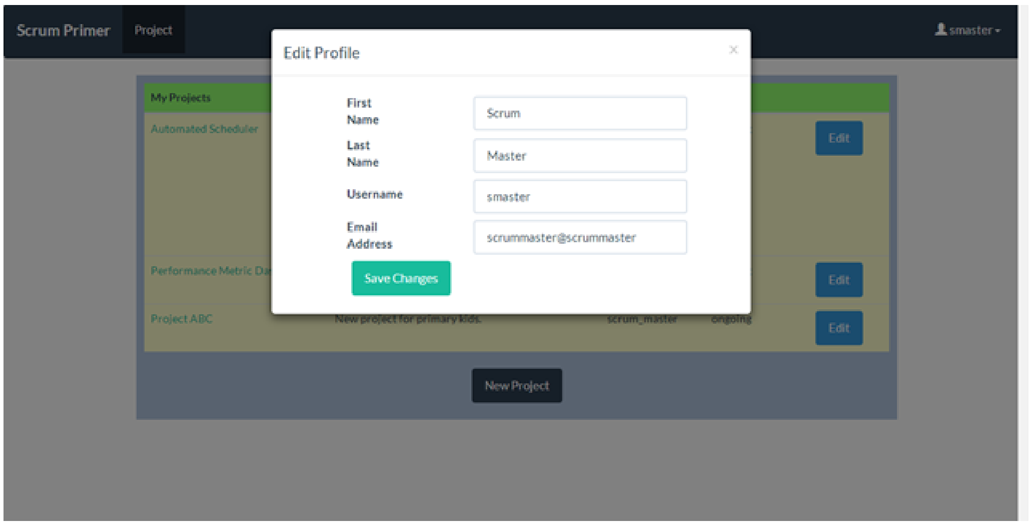
1. User's Edit Profile



My Projects	Description	Role	Status	
Automated Scheduler	Workforce Optimization has just recently been implemented in GOC. This means our staffs covering different shifts have been reduced. We need a tool that would automatically plot their schedules on a monthly basis taking into consideration their scheduled Leaves, Compensatory Day off, workload and the number of employees available.	scrum_master	ongoing	Edit
Performance Metric Dashboard	To develop a UI to exhibit Team and Individual metrics for GOC Software Services Capability.	scrum_master	ongoing	Edit
Project ABC	New project for primary kids.	scrum_master	ongoing	Edit

Figure 38: Show submenu for edit profile and change password

Every user has a privilege to edit his/her profile under their username.

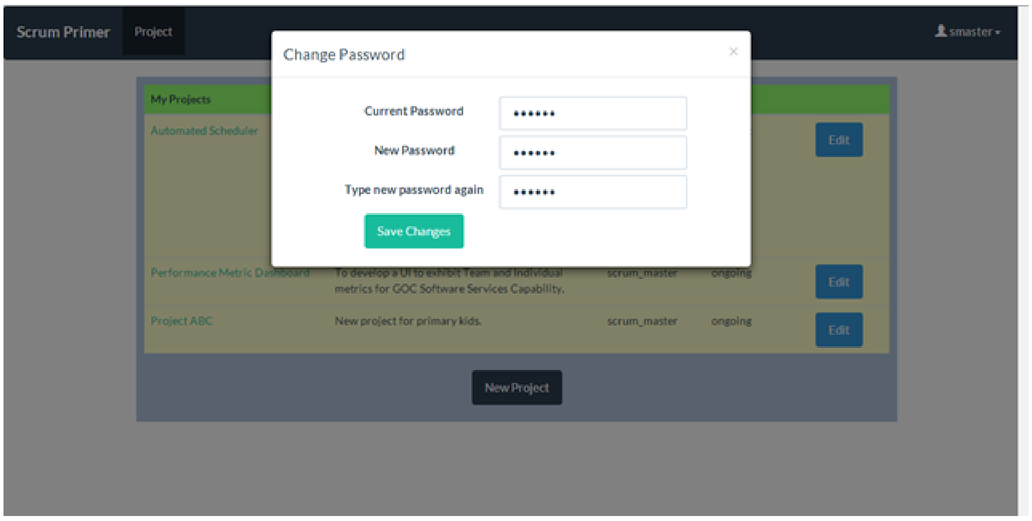


Edit Profile

First Name	Scrum
Last Name	Master
Username	smaster
Email Address	scrummaster@scrummaster

Save Changes

Figure 39: Edit profile


Scrum Primer
Project
smaster
My Projects
Automated Scheduler
Performance Metric Dashboard
To develop a UI to exhibit Team and Individual metrics for GOC Software Services Capability.
scrum_master
ongoing
Project ABC
New project for primary kids.
scrum_master
ongoing
Edit
New Project

Change Password

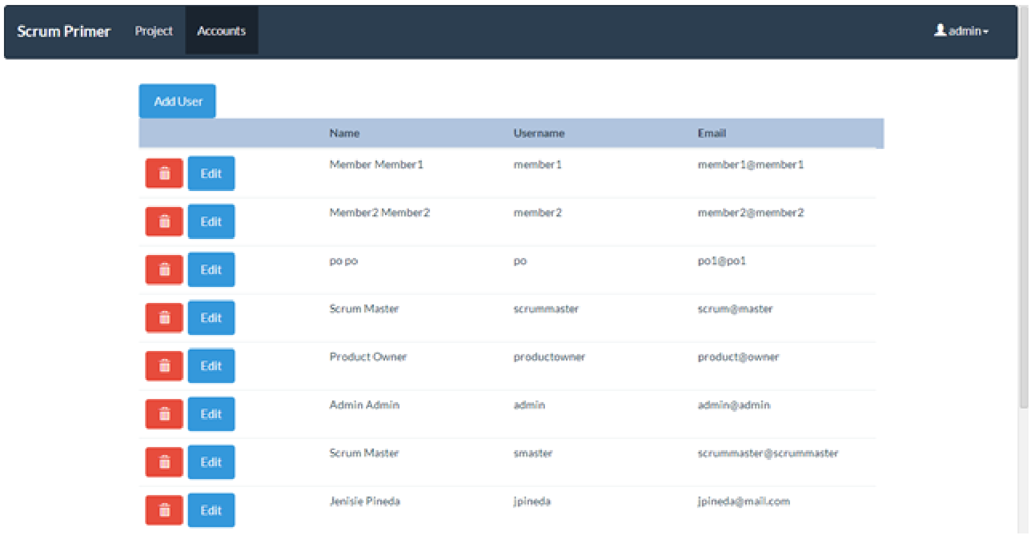
Current Password	••••••
New Password	••••••
Type new password again	••••••

Save Changes

Figure 40: Change password

2. **Administrator**

An administrator has a capability to add, edit and delete account.



	Name	Username	Email
Edit	Member Member1	member1	member1@member1
Edit	Member2 Member2	member2	member2@member2
Edit	po po	po	po1@po1
Edit	Scrum Master	scrummaster	scrum@master
Edit	Product Owner	productowner	product@owner
Edit	Admin Admin	admin	admin@admin
Edit	Scrum Master	smaster	scrummaster@scrummaster
Edit	Jenisie Pineda	jpineda	jpineda@mail.com

Figure 41: Manage Accounts

VII. Discussions

The Scrum Project Planner Primer is a web-based end-to-end tooling support for student groups in which complexity of existing systems are reduced. This tool features the nature of Scrum as a framework of Agile and at the same time supports the basic needs of student groups for a project tool. The tool has 2 types of user, the regular users and the administrators. Regular users may vary into three roles Scrum Master, Team Member and Product Owner, depending on the stand of the user in his/her own project. These roles are the main roles in scrum that represent the Scrum Team. The tool covers the different scrum tools and practices including the product backlog, the sprints, sprint backlog with activities, scrum meetings, scrum board, burndown chart and sprint reviews.

One of the main features of the tool is the involvement of the product owners that represents the clients or the instructors since they are the one who provide the requirements specifications of a project. With their involvement they can monitor the overview progress of the project by viewing the sprint backlogs, scrum board and scrum meetings. There is also a much clearer distinction of the requirements since it is stored in a detailed manner. Product owner is also the one who decides what user stories they want. This may imply that the product owner is at stake on what path the project is heading to.

The tool was developed as a Java Dynamic Web Application using the Spring MVC Framework. Since Spring is an application framework that enables to build sturdy web applications. It features many modules that supports a developer to focus on application-level business thus developing the tool is uncomplicated in terms of deployment environment [23]. Another advantage of using Spring Framework is the use of JdbcTemplate class which is a class in the JDBC core package. JdbcTemplate covers one of the common errors in using JDBC which is closing the connection [24].

One problem with using Java Web Applications is that JSTL is not compatible with Javascript. Passing the values to the scripts is tedious since JSTL core functionalities is not working when merging it with Javascript.

Since its purpose is to minimize the functionalities of the tools that are patterned for professional environment, the tool does not ensure if students may consequently understand the whole Scrum process in a strictly ruled manner. It is a collaborating tool adjusted for the use of student groups. The adjustment deviates the precise process of Scrum defined in different textbooks like the fixed devoted time of a person per day in which is not possible for a student since there are not full-time developers. Also, the tool diverge from the rule of some processes like the scrum meeting should be done in a fixed time every day and theres a fixed time scale for a Sprint.

VIII. Conclusions

Scrum Project Planner Primer is a tool that encompasses the core process of Scrum that is applicable to student groups. The featured functionalities of the tool pertains to the important aspects that a software engineering tool for student teams which are, the ability of the tool that allows the team to collaborate and set as storage for the requirements, the versioning that describes the ability of the tool to track historical changes of each task or work package, the work product support refers to the ability of the tool to categorize and record each work package, the availability of the tool to apply in each computer machine and the ability of the tool to record the reviews and evaluation from the product owner in each user stories[19].

The tool allows the three scrum roles to interact with one another as team. Once the project was created, the Product Owner is responsible in providing the requirements specifications by adding user stories in the product backlog and opens a new sprint given his own preferred user stories to be included in that certain sprint.

After a new sprint was opened, the Scrum Master will now indicate the planned tasks that need to be finished by the end of the sprint. Estimate is needed because it defines the complexity of a certain task and since the developing team of scrum is self-organizing, listing of volunteers is also included. During the sprint, every activity done by the developing team is stored in the sprint backlog. Activities are updated until a certain task is done.

The Scrum Master always initiate a daily scrum meeting by notifying all the members to participate. Scrum meeting details represent the summary of accomplishments in that day, the plans for the next day, and if there are any impediments along the way that stops the member to pursue his ongoing work.

The progress of each work can be overviewed through the use of scrum board and burndown chart. These two scrum tools enable the tasks to be visually grouped according to its status. The trend shown in burndown chart represents the time devoted by the team to finish their tasks in a sprint.

The team will be notified if the sprint review is upcoming. On the day of the sprint review, the product owner will list his comments regarding the outcome of the user story. He will also decide if the user story is still ongoing or considered to be done. After giving the sprint review, the sprint will automatically close. The product owner will have an option if he wants to open a new sprint or just close the project. All the members have an access to closed sprints and projects therefore they can also view the historical changes by then.

The system administrator is a special user in which he/she can add, edit or deactivate a certain account. He can also create another administrator.

With this tool, the student groups can now apply scrum feasibly. Functionalities included are useful to each role type. The tool may help each of the member of the scrum team to track each other's activities and they can also monitor the progress of the project. The tool can help the scrum team to achieve the project's success.

IX. Recommendations

Scrum Project Planner Primer is a simplified version of a scrum collaborative tool. It mainly points out the main process of Scrum with some constraints since the developers are just students and not full time workers.

The tool will be more beneficial to students if this will be integrated to the IDE used by the developers since it will serve as a repository at the same time. Aside from integrating it to the IDE, providing a war room enables the Scrum Team to conduct meeting even in physical absence.

X. Bibliography

[1] S. Dyck and T. Majchrzak, “Identifying common characteristics in fundamental, integrated, and agile software development methodologies,” in *45th Hawaiian International Conference on System Sciences*, 2012.

[2] e. P. Deemer, “The scrum primer v1.2.” `http://www.goodagile.com/scrumprimer/`. [Accessed: Oct, 2013].

[3] C. Peixoto, “Human-computer interface expert system for agile methods,” in *31st Int. Conf. on Information Technology Interfaces*, 2009.

[4] P. B. E. Hossain and R.Jeffery, “Towards an understanding of tailoring scrum in global software development: A multi-case study,” in *ICSSP*, 2011.

[5] “Agile methodology.” `{http://agilemethodology.org/}`. [Accessed: Oct, 2013].

[6] C. Scharff and O. Gotel, “Transitioning to distributed development in students global software development projects: The role of agile methodologies and end-to-end tooling,” in *Fifth International Conference on Software Engineering Advances*, 2010.

[7] H. A. M. Perkusich and A. Perkusich, “A model to detect problems on scrum-based software development projects,” in *SAC*, 2013.

[8] F. d. S. A. Franca and L. de Sousa Mariz, “An empirical study on the relationship between the use of agile practices and the success of scrum projects,” *ACM*, 2010.

[9] P. Grisham and D. Perry, “Customer relationships and extreme programming,” *Human and Social Factors of Software Engineering (HSSE)*, 2005.

[10] C. D. W. Schramm and T. Grechenig, “Issues and mitigation strategies when using agile industrial software development processes in student software engineering projects,” in *IEEE Africon*, 2011.

[11] M. Dubakov, "Agile tools. the good, the bad and the ugly. - targetprocess." `www.targetprocess.com/download/whitepaper/agiletools.pdf`. [Accessed: Oct, 2013].

[12] K. Erickson, "A comparative look at top agile tools." `https://www.captechconsulting.com/blog/kevin-erickson/comparative-look-top-agile-tools`. [Accessed: Oct, 2013].

[13] D. Hart, "Supporting agile processes in software engineering courses," in *CCSC North Eastern Conference*, 2010.

[14] F. Correia and A. Aguiar, "Software knowledge capture and acquisition: Tool support for agile settings," in *Fourth International Conference on Software Engineering Advances*, 2009.

[15] A. Mishra and D. Mishra, "Software project management tools: A brief comparative view," *ACM SIGSOFT Software Engineering Notes*, 2013.

[16] O. Ktata and G. Levesque, "Designing and implementing a measurement program for scrum teams: What do agile developers really need and want?," 2010.

[17] e. L. Pinto, "On the use of scrum for the management of practcal projects in graduate courses," in *39th ASEE/IEEE Frontiers in Education Conference*, 2009.

[18] D. D. et.al, "Teaching a globally distributed project course using scrum practices," *IEEE*, 2012.

[19] D. Rico and H. Sayani, "Use of agile methods in software engineering education," in *Agile Conference*, 2009.

[20] M. Ashraf and S. Rana, "Agile model adaptation for e-learning students final-year project," in *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, 2012.

[21] e. A. Abran, “Software engineering book of knowledge.” www.swebok.org/. [Accessed: Oct, 2013].

[22] “Manifesto for agile software development.” http://agilemanifesto.org/iso/en/. [Accessed: Oct, 2013].

[23] “Introduction to spring.” http://projects.spring.io/spring-framework/. [Accessed: April,2014].

[24] “Data access using jdbc.” http://docs.spring.io/spring/docs/2.0.8/reference/jdbc.html. [Accessed: April,2014].

XI. Appendix

A. Source Code

1. Models and DAOs

Listing 1: Activity.java

```java
/**
 * Activity.java
 *
 * Model for the object Activity
 */
package com.scrumprimer.model;

public class Activity {
        int activityID;
        String detail;
        int taskID;
        String date;
        int time;
        String taskStatus;
        String doer;
        public String getDoer() {
                return doer;
        }
        public void setDoer(String doer) {
                this.doer = doer;
        }
        public int getAccountID() {
                return accountID;
        }
        public void setAccountID(int accountID) {
                this.accountID = accountID;
        }
        int accountID;
        public int getActivityID() {
                return activityID;
        }
        public void setActivityID(int activityID) {
                this.activityID = activityID;
        }
        public String getDetail() {
                return detail;
        }
        public void setDetail(String detail) {
                this.detail = detail;
        }
        public int getTaskID() {
                return taskID;
        }
        public void setTaskID(int taskID) {
                this.taskID = taskID;
        }
        public String getDate() {
                return date;
        }
        public void setDate(String date) {
                this.date = date;
        }
        public int getTime() {
                return time;
        }
        public void setTime(int time) {
                this.time = time;
        }
        public String getTaskStatus() {
                return taskStatus;
        }
        public void setTaskStatus(String taskStatus) {
                this.taskStatus = taskStatus;
        }


}
```

Listing 2: ActivityRowMapper.java

```java
/**
 * ActivityRowMapper.java
 *
 * Row mapper for activities table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;
```

```

public class ActivityRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                Activity activity = new Activity();
                activity.setActivityID(rs.getInt("activityID"));
                activity.setDate(rs.getString("date"));
                activity.setDetail(rs.getString("detail"));
                activity.setTaskID(rs.getInt("taskID"));
                activity.setTime(rs.getInt("time"));
                activity.setAccountID(rs.getInt("accountID"));

                return activity;
        }

}
```

Listing 3: Dummy.java

```
/**
 * Dummy.java
 *
 * Model for object Dummy -- for invite model
 */
package com.scrumprimer.model;

public class Dummy {
        String firstName;
        String lastName;
        String role;
        String email;
        int projectID;

        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 getRole() {
                return role;
        }
        public void setRole(String role) {
                this.role = role;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public int getProjectID() {
                return projectID;
        }
        public void setProjectID(int projectID) {
                this.projectID = projectID;
        }

}
```

Listing 4: DummyRowMapper.java

```
/**
 * DummyRowMapper.java
 *
 * Row Mapper for invites table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class DummyRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                Dummy dummy = new Dummy();
                dummy.setEmail(rs.getString("email"));
                dummy.setFirstName(rs.getString("firstName"));
                dummy.setLastName(rs.getString("lastName"));
                dummy.setProjectID(rs.getInt("projectID"));
                dummy.setRole(rs.getString("role"));
                return dummy;
        }

}
```

Listing 5: Meeting.java

```
/**
 * Meeting.java
 *
 * Model for object Meeting - contains the scrum meeting details
 */

package com.scrumprimer.model;

public class Meeting {
        String date;
        int meetingID;
        int sprintID;
        String accomplishment;
        String plan;
        String impediment;
        int accountID;
        String member;
        public String getMember() {
                return member;
        }
        public void setMember(String member) {
                this.member = member;
        }
        public int getAccountID() {
                return accountID;
        }
        public void setAccountID(int accountID) {
                this.accountID = accountID;
        }
        public String getDate() {
                return date;
        }
        public void setDate(String date) {
                this.date = date;
        }
        public int getMeetingID() {
                return meetingID;
        }
        public void setMeetingID(int meetingID) {
                this.meetingID = meetingID;
        }
        public int getSprintID() {
                return sprintID;
        }
        public void setSprintID(int sprintID) {
                this.sprintID = sprintID;
        }
        public String getAccomplishment() {
                return accomplishment;
        }
        public void setAccomplishment(String accomplishment) {
                this.accomplishment = accomplishment;
        }
        public String getPlan() {
                return plan;
        }
        public void setPlan(String plan) {
                this.plan = plan;
        }
        public String getImpediment() {
                return impediment;
        }
        public void setImpediment(String impediment) {
                this.impediment = impediment;
        }



}
```

Listing 6: Meetingform.java

```
/**
 * MeetingForm.java
 *
 * Model for multiple Meeting objects
 */
package com.scrumprimer.model;

import java.util.ArrayList;

public class MeetingForm {
        ArrayList<Review> reviews;

        public ArrayList<Review> getReviews() {
                return reviews;
        }

        public void setReviews(ArrayList<Review> reviews) {
                this.reviews = reviews;
        }

}
```

Listing 7: MeetingRowMapper.java

```
/**
 * MeetingRowMapper.java
 *
 * Row Mapper for meetings table
 */
package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class MeetingRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                Meeting meeting = new Meeting();
                meeting.setAccomplishment(rs.getString("accomplishment"));
                meeting.setImpediment(rs.getString("impediment"));
                meeting.setMeetingID(rs.getInt("meetingID"));
                meeting.setPlan(rs.getString("plan"));
                meeting.setSprintID(rs.getInt("sprintID"));
                meeting.setAccountID(rs.getInt("accountID"));
                meeting.setDate(rs.getString("date"));
                return meeting;
        }


}
```

Listing 8: Member.java

```
/**
 * Member.java
 *
 * Model for object Member -- project's member
 */

package com.scrumprimer.model;

public class Member {
        int memberID;
        int projectID;
        String projectName;
        String role;
        int accountID;
        String name;
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public int getAccountID() {
                return accountID;
        }
        public void setAccountID(int accountID) {
                this.accountID = accountID;
        }
        public int getMemberID() {
                return memberID;
        }
        public void setMemberID(int memberID) {
                this.memberID = memberID;
        }

        public int getProjectID() {
                return projectID;
        }
        public void setProjectID(int projectID) {
                this.projectID = projectID;
        }
        public String getProjectName() {
                return projectName;
        }
        public void setProjectName(String projectName) {
                this.projectName = projectName;
        }
        public String getRole() {
                return role;
        }
        public void setRole(String role) {
                this.role = role;
        }
}
```

Listing 9: Notification.java

```
/**
 * Notification.java
 *
 * Model for object Notification
 */

package com.scrumprimer.model;

public class Notification {
        int notifID;
        int accountID;
        int projectID;
        String type;
        String date;
        String newMember;
        int sprintID;
        public String getNewMember() {
                return newMember;
        }
        public void setNewMember(String newMember) {
                this.newMember = newMember;
        }
        public String getDate() {
                return date;
        }
        public int getSprintID() {
                return sprintID;
        }
        public void setSprintID(int sprintID) {
                this.sprintID = sprintID;
        }
        public void setDate(String date) {
                this.date = date;
        }
        String status;
        public String getStatus() {
                return status;
        }
        public void setStatus(String status) {
                this.status = status;
        }
        public int getNotifID() {
                return notifID;
        }
        public void setNotifID(int notifID) {
                this.notifID = notifID;
        }
        public int getAccountID() {
                return accountID;
        }
        public void setAccountID(int accountID) {
                this.accountID = accountID;
        }
        public int getProjectID() {
                return projectID;
        }
        public void setProjectID(int projectID) {
                this.projectID = projectID;
        }
        public String getType() {
                return type;
        }
        public void setType(String type) {
                this.type = type;
        }

}
```

Listing 10: NotificationDAO.java

```
/**
 * NotificationDAO.java
 *
 */

package com.scrumprimer.model;

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

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;

import com.scrumprimer.utility.NotificationUtility;

public class NotificationDAO {

        @Autowired
        private DataSource dataSource;
        private JdbcTemplate jdbcTemplateObject;

        public void insertNotification(Notification notif, int receiverID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "INSERT INTO notifications  (type, accountID, projectID,
                      receiverID, sprintID) VALUES (?,?,?,?,?)";
                jdbcTemplateObject.update(query, new Object[]{notif.getType(), notif.
                    getAccountID(), notif.getProjectID(), receiverID, notif.getSprintID
                    ()});
        }

        @SuppressWarnings("unchecked")
        public List<Notification> getNotification(int projectID, int receiverID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "UPDATE notifications SET status = ? WHERE projectID = ?
                    and receiverID = ?";
                jdbcTemplateObject.update(query, new Object[]{"read", projectID,
                    receiverID});
                query = "SELECT * FROM notifications WHERE status = ? and projectID = ?
                    and receiverID = ? ORDER BY notifID DESC";
                ArrayList<Notification> notifications = new ArrayList<Notification>();
                notifications = (ArrayList<Notification>) jdbcTemplateObject.query(query
                    , new Object[]{"read", projectID, receiverID}, new
                    NotificationRowMapper());
                return notifications;
        }

        public int getNumOfNewNotifs(int projectID, int receiverID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM notifications WHERE status = ? and
                    projectID = ? AND receiverID = ?";
                @SuppressWarnings("deprecation")
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{"unread",
                    projectID, receiverID});
                return row;
        }

        public boolean isSprintReview(int sprintID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM notifications WHERE type = ? and
                    sprintID = ?";
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{
                    NotificationUtility.SPRINT_REVIEW, sprintID});
                if(row == 0){
                        return false;
                }
                return true;
        }
}
```

Listing 11: NotificationRowMapper.java

```
/**
 * NotificationRowMapper.java
 *
 * Row mapper for notification table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class NotificationRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                Notification notif = new Notification();
                notif.setAccountID(rs.getInt("accountID"));
                notif.setNotifID(rs.getInt("notifID"));
                notif.setProjectID(rs.getInt("projectID"));
                notif.setType(rs.getString("type"));
                notif.setDate(rs.getString("date"));
                notif.setStatus(rs.getString("status"));
                notif.setSprintID(rs.getInt("sprintID"));
                return notif;
        }

}
```

Listing 12: Project.java

```
/**
 * Project.java
 *
 * Model for object Project
 *
 */

package com.scrumprimer.model;

public class Project {

        int projectID;
        String projectName;
        String projectDescription;
        int numberOfTM;
        int numberOfPO;
        String status;
        public String getStatus() {
                return status;
        }
        public void setStatus(String status) {
                this.status = status;
        }
        public int getNumberOfTM() {
                return numberOfTM;
        }
        public void setNumberOfTM(int numberOfTM) {
                this.numberOfTM = numberOfTM;
        }
        public int getNumberOfPO() {
                return numberOfPO;
        }
        public void setNumberOfPO(int numberOfPO) {
                this.numberOfPO = numberOfPO;
        }
        public int getProjectID() {
                return projectID;
        }
        public void setProjectID(int projectID) {
                this.projectID = projectID;
        }
        public String getProjectName() {
                return projectName;
        }
        public void setProjectName(String projectName) {
                this.projectName = projectName;
        }
        public String getProjectDescription() {
                return projectDescription;
        }
        public void setProjectDescription(String projectDescription) {
                this.projectDescription = projectDescription;
        }
}
```

Listing 13: ProjectDAO.java

```
/**
 * ProjectDAO.java
 *
 */

package com.scrumprimer.model;

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

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.support.rowset.SqlRowSet;
import org.springframework.security.authentication.encoding.PasswordEncoder;

import com.mysql.jdbc.ResultSet;
import com.scrumprimer.model.User;
import com.scrumprimer.model.UserRowMapper;
import com.scrumprimer.utility.DateHandler;
import com.scrumprimer.utility.StatusUtility;


public class ProjectDAO {

        @Autowired
        private DataSource dataSource;
        private JdbcTemplate jdbcTemplateObject;
         @Autowired
        private PasswordEncoder passwordEncoder;
         @Autowired
         private UserDAO userDAO;

         public List<Member> getProject(int accountID){
                        jdbcTemplateObject = new JdbcTemplate(dataSource);
                        String query = "SELECT * FROM members WHERE accountID = ?";
                        List<Member> members = jdbcTemplateObject.query(query, new
                             Object[]{accountID}, new ProjectRowMapper());
                        return members;
         }

         public List<UserStory> getUserStory(int projectID){
                 jdbcTemplateObject = new JdbcTemplate(dataSource);
                 String query = "SELECT * FROM userStories WHERE projectID = ?";
                 List<UserStory> userStories = jdbcTemplateObject.query(query,new Object
                      []{projectID}, new UserStoryMapper());
                 return userStories;
         }

         public List<UserStory> getOngoingUserStory(int projectID){
                 jdbcTemplateObject = new JdbcTemplate(dataSource);
                 String query = "SELECT * FROM userStories WHERE projectID = ? and (
                      status = ? OR status = ?)";
                 List<UserStory> userStories = jdbcTemplateObject.query(query,new Object
                      []{projectID, StatusUtility.ONGOING, StatusUtility.NOT_STARTED},
                      new UserStoryMapper());
```

```
                return userStories;
        }

        public boolean isUserStoryExisting(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM userStories WHERE projectID = ?";
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{projectID}
                    );
                if(row == 0)
                        return false;
            else
                return true;

        }

        public boolean isTaskExisting(int sprintID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM tasks WHERE sprintID = ?";
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{sprintID})
                    ;
                if(row == 0)
                        return false;
                else
                        return true;

        }

        public void updateUserStories(List<UserStory> userStories){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                for(UserStory story : userStories){
                        String query = "UPDATE userstories SET storyName = ?, detail =
                            ?, status = ? WHERE storyID = ?";
                        jdbcTemplateObject.update(query, new Object[]{story.
                            getStoryName(), story.getDetail(), story.getStatus(),
                            story.getStoryID()});
                }

        }

        public void insertUserStories(List<UserStory> userStories){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                for(UserStory story: userStories){
                        String query = "INSERT INTO userStories (storyName, detail,
                            projectID, status) VALUES (?, ?, ?, ?)";
                        jdbcTemplateObject.update(query, new Object[]{story.
                            getStoryName(), story.getDetail(), story.getProjectID(),
                            story.getStatus()});
                }
        }

        public String getProjectName(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM projects WHERE projectID = ?";
                Project project = jdbcTemplateObject.queryForObject(query, new Object
                    []{projectID}, new ProjectDetailsRowMapper());
                return project.getProjectName();
        }



        public List<Dummy> getPendingProject(String email){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM invite where email = ?";
                List<Dummy> dummies = jdbcTemplateObject.query(query, new Object[]{
                    email}, new DummyRowMapper());
                return dummies;
        }

        public String getRole(int accountID, int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM members WHERE accountID = ? and projectID
                    =  ?";
                Member member = jdbcTemplateObject.queryForObject(query, new Object[]{
                    accountID, projectID}, new ProjectRowMapper());
                return member.getRole();
        }


        public boolean isAccountHasProject(int accountID){
                        jdbcTemplateObject = new JdbcTemplate(dataSource);
                        String query = "SELECT COUNT(*) FROM members WHERE accountID =
                            ?";
                        int result = jdbcTemplateObject.queryForInt(query, new Object[]{
                            accountID});

                        if(result == 0){
                                return false;
                        }
                        else{
                                return true;
                        }
        }

        public List<Project> getProjectDetails(List<Member> members){
                        jdbcTemplateObject = new JdbcTemplate(dataSource);
                        List<Project> projects = new ArrayList<Project>();
                        String query = "SELECT * FROM projects where projectID = ?";
                        for(Member member : members){
                                projects.add((Project) jdbcTemplateObject.queryForObject
                                    (query, new Object[]{member.getProjectID()}, new
                                    ProjectDetailsRowMapper()));
                        }

                        return projects;
        }

        public Project getProjectByProjectID(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM projects where projectID = ?";
                Project project = jdbcTemplateObject.queryForObject(query, new Object
                    []{projectID}, new ProjectDetailsRowMapper());
                return project;

        }

        public int addNewProject(Project project){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                project.setStatus(StatusUtility.ONGOING);
                String query = "INSERT INTO projects (name, description, status) VALUES
                     (?, ?, ?)";
                jdbcTemplateObject.update(query, new Object[]{project.getProjectName(),
                     project.getProjectDescription(), project.getStatus()});
                query = "SELECT * FROM projects where name = ?";
                Project project2 = jdbcTemplateObject.queryForObject(query, new Object
                    []{project.getProjectName()}, new ProjectDetailsRowMapper());
                return project2.getProjectID();
        }

        public void updateProject(Project project){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "UPDATE projects SET name=?, description=? WHERE
                    projectID = ?";
                jdbcTemplateObject.update(query, new Object[]{project.getProjectName(),
                     project.getProjectDescription(), project.getProjectID()});
                return ;
        }

        public void addMember(Member member){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "INSERT INTO members (projectID, role, accountID) VALUES
                     (?, ?, ?)";
                jdbcTemplateObject.update(query, new Object[]{member.getProjectID(),
                    member.getRole(), member.getAccountID()});
        }

        public boolean isCurrentSprintExist(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM sprints WHERE projectID = ? and
                    status = ?";
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{projectID,
                     StatusUtility.SPRINT_OPEN});
                if(row == 1){
                        return true;
                }
                else{
                        return false;
                }
        }

        public boolean isSprintExist(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT COUNT(*) FROM sprints WHERE projectID = ?";
                int row = jdbcTemplateObject.queryForInt(query, new Object[]{projectID
                    });
                if(row == 0){
                        return false;
                }
                else
                        return true;
        }

        public Sprint getCurrentSprint(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM sprints WHERE projectID = ? and status =
                    ?";
                Sprint sprint = jdbcTemplateObject.queryForObject(query, new Object[]{
                    projectID, StatusUtility.SPRINT_OPEN}, new SprintRowMapper());
                return sprint;

        }

        public List<Sprint> getAllSprint(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM sprints WHERE projectID = ? ORDER BY
                    sprintNumber DESC";
                List<Sprint> allSprint = jdbcTemplateObject.query(query, new Object[]{
                    projectID}, new SprintRowMapper());
                return allSprint;

        }

        public void insertSprint(Sprint sprint){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "INSERT INTO sprints (sprintNumber, status, projectID,
                    sprintReview) VALUES (?, ?, ?, ?)";
                jdbcTemplateObject.update(query, new Object[]{sprint.getSprintNumber(),
                     sprint.getStatus(), sprint.getProjectID(), sprint.getSprintReview
                ()});

        }

        public int getLatestSprintNumber(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                if(!isSprintExist(projectID)){
                        return 0;
                }
                else{
                        String query = "SELECT MAX(sprintNumber) FROM sprints WHERE
                            projectID = ?";
                        int max = jdbcTemplateObject.queryForInt(query, new Object[]{
                            projectID});
                        return max;
                }
        }

        public void insertSprintStories(List<SprintStory> sprintStories){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "INSERT INTO sprintstory (sprintID, storyID) VALUES
                    (?,?)";
                for(SprintStory sprintStory : sprintStories){
                        if(sprintStory.getSprintID() != 0 && sprintStory.getStoryID()
                            != 0)
                        jdbcTemplateObject.update(query, new Object[]{sprintStory.
                            getSprintID(), sprintStory.getStoryID()});
                }
        }

        public UserStory getUserStoryByID(int storyID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM userstories WHERE storyID = ?";
                UserStory story = jdbcTemplateObject.queryForObject(query, new Object
                    []{storyID}, new UserStoryMapper());
                return story;
        }

        public List<UserStory> getSprintStories(int sprintID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM sprintstory WHERE sprintID = ?";
                List<SprintStory> sprintStories = jdbcTemplateObject.query(query, new
                    Object[]{sprintID}, new SprintStoryRowMapper());
                List<UserStory> stories = new ArrayList<UserStory>();
                for(SprintStory sprint : sprintStories){
                        stories.add(getUserStoryByID(sprint.getStoryID()));
                }
                return stories;
        }

        public void insertTasks(List<Task> tasks){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "INSERT INTO tasks (sprintID, storyID, accountID,
                    taskName, estimate, status) VALUES (?, ?, ?, ?, ?, ?)";
                for(Task task : tasks){
                jdbcTemplateObject.update(query, task.getSprintID(), task.getStoryID(),
                     task.getAccountID(), task.getTaskName(), task.getEstimate(), task
                    .getStatus());
                }
        }

        public List<Task> getTasks(int sprintID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * FROM tasks where sprintID = ?";
                List<Task> tasks = jdbcTemplateObject.query(query, new Object[]{
                    sprintID}, new TaskRowMapper());
                return tasks;
        }

        public List<Member> projectMembers(int projectID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "Select * FROM members WHERE projectID=?";
                List<Member> members = jdbcTemplateObject.query(query, new Object[]{
                    projectID}, new ProjectRowMapper());
                for(Member mem : members){
                        mem.setName(userDAO.getUser2(mem.getAccountID()).getFirstName()
                            );
                }
                return members;
        }

        public Task getTaskViaTaskID(int taskID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * from tasks WHERE taskID = ?";
                Task task = jdbcTemplateObject.queryForObject(query, new Object[]{
                    taskID}, new TaskRowMapper());
                return task;
        }



        public List<Activity> getAllActivities(int taskID){
                jdbcTemplateObject = new JdbcTemplate(dataSource);
                String query = "SELECT * from activities WHERE taskID = ?";
                List<Activity> activities = jdbcTemplateObject.query(query, new Object
                    []{taskID}, new ActivityRowMapper());
                return activities;


```

```
	}

	public void insertActivity(Activity activity){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "INSERT INTO activities (detail, taskID, date, time,
		     accountID) VALUES (?, ?, ?, ?, ?)";
		jdbcTemplateObject.update(query, new Object[]{activity.getDetail(),
		     activity.getTaskID(), activity.getDate(), activity.getTime(),
		     activity.getAccountID()});
		return;
	}

	public void updateTaskStatus(String status, int taskID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "UPDATE tasks SET status=? WHERE taskID = ? ";
		jdbcTemplateObject.update(query, new Object[]{status, taskID});
	}

	public Sprint getSprintBySprintID(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM sprints WHERE sprintID = ?";
		Sprint sprint = jdbcTemplateObject.queryForObject(query, new Object[]{
		     sprintID}, new SprintRowMapper());
		return sprint;
	}

	public boolean isActivityExisting(int taskID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT COUNT(*) FROM activities WHERE taskID = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{taskID});
		if(row == 0){
			return false;
		}
		else{
			return true;
		}
	}

	public int getTotalActivityPerDay(int taskID, String date){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM activities WHERE taskID = ? AND date =
		     ?";

		if(isActivityExisting(taskID)){
		List<Activity> activities = jdbcTemplateObject.query(query, new Object
		     []{taskID, date}, new ActivityRowMapper());
		int sum = 0;
		for(Activity act : activities){
			sum += act.getTime();
		}

		return sum;
		}

		else{
			return 0;
		}
	}

	public int getTotalEstimate(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM tasks WHERE sprintID = ?";
		List<Task> tasks = jdbcTemplateObject.query(query,new Object[]{sprintID
		     }, new TaskRowMapper());
		int sum = 0;
		for(Task task : tasks){
		      sum += task.getEstimate();
		}
		return sum;
	}

	public void insertNotif(Notification notif){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "INSERT INTO notifications (accountID, projectID, type)
		     VALUES (?, ?, ?)";
		jdbcTemplateObject.update(query, new Object[]{notif.getAccountID(),
		     notif.getProjectID(), notif.getType()});
		return;
	}

	public List<Sprint> getAllOpenSprint(){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM sprints WHERE status = ?";
		List<Sprint> sprint = jdbcTemplateObject.query(query, new Object[]{"
		     open"}, new SprintRowMapper());
		return sprint;
	}



	public List<Meeting> getScrumMeeting(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM scrummeeting WHERE sprintID = ? GROUP BY
		     date ORDER BY meetingID DESC";
		List<Meeting> meeting = jdbcTemplateObject.query(query, new Object[]{
		     sprintID}, new MeetingRowMapper());
		return meeting;
	}

	public List<Meeting> getMeetingDetail(int sprintID, String date){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM scrummeeting WHERE sprintID = ? AND date
			= ? ORDER BY meetingID DESC";
		List<Meeting> meeting = jdbcTemplateObject.query(query, new Object[]{
			sprintID, date}, new MeetingRowMapper());
		return meeting;
	}

	public boolean isScrumMeeting(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{sprintID})
			;
		if(row == 0){
			return false;
		}
		return true;
	}

	public boolean isCurrentScrumMeeting(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?
			AND date = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{sprintID,
			DateHandler.getCurrentDate()});
		if(row == 0)
			return false;
		return true;
	}

	public boolean isScrumMeetingAnswered(int sprintID, int accountID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?
			AND date = ? AND accountID = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{sprintID,
			DateHandler.getCurrentDate(), accountID});
		if(row == 0)
			return false;
		return true;

	}

	public void insertScrumMeeting(Meeting meeting){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "INSERT INTO scrummeeting (date, accountID,
			accomplishment, plan, impediment, sprintID) VALUES (?,?,?,?,?,?)";
		jdbcTemplateObject.update(query,new Object[]{meeting.getDate(), meeting
			.getAccountID(), meeting.getAccomplishment(), meeting.getPlan(),
			meeting.getImpediment(), meeting.getSprintID() });
		return ;
	}

	public boolean isReviewExisting(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT COUNT(*) from sprintreview WHERE sprintID = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{sprintID})
			;
		if(row == 0)
			return false;
		return true;

	}

	public List<Review> getSprintReview(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * from sprintreview WHERE sprintID = ?";
		List<Review> reviews = jdbcTemplateObject.query(query,new Object[]{
			sprintID}, new ReviewRowMapper());
		return reviews;
	}

	public void insertSprintReview(Review review){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "INSERT INTO sprintreview (sprintID, storyID, review)
			VALUES (?,?,?)";
		jdbcTemplateObject.update(query, new Object[]{review.getSprintID(),
			review.getStoryID(), review.getReview()});
		return;
	}

	public void updateStoryStatus(String status, int storyID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "UPDATE userstories SET status = ? WHERE storyID = ?";
		jdbcTemplateObject.update(query, new Object[]{status, storyID});
		return;
	}

	public void closeSprint(int sprintID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "UPDATE sprints SET status = ? WHERE sprintID = ?";
		jdbcTemplateObject.update(query, new Object[]{StatusUtility.CLOSED,
			sprintID});
		return;
	}

	public void closeProject(int projectID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "UPDATE projects SET status = ? WHERE projectID = ?";
```

```
                    jdbcTemplateObject.update(query, new Object[]{StatusUtility.CLOSED,
                        projectID});
                    return;
            }



}
```

Listing 14: ProjectDetailsRowMapper.java

```
/**
 * ProjectDetailsRowMapper.java
 *
 * Row mapper for projects table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;


public class ProjectDetailsRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
                // TODO Auto-generated method stub
                Project project = new Project();
                project.setProjectID(rs.getInt("projectID"));
                project.setProjectName(rs.getString("name"));
                project.setProjectDescription(rs.getString("description"));
                project.setStatus(rs.getString("status"));
                return project;
        }

}
```

Listing 15: ProjectRowMapper.java

```
/**
 * ProjectRowMapper.java
 *
 * Row mapper for projects table
 *
 */
package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class ProjectRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
                Member member = new Member();
                member.setMemberID(rs.getInt("memberID"));
                member.setProjectID(rs.getInt("projectID"));
                member.setRole(rs.getString("role"));
                member.setAccountID(rs.getInt("accountID"));
                // TODO Auto-generated method stub
                return member;
        }

}
```

Listing 16: Review.java

```
/**
 * Review.java
 *
 * Model for object Review
 */

package com.scrumprimer.model;

public class Review {
        int reviewID;
        int sprintID;
        int storyID;
        String review;
        String status;
        String storyName;
        public int getReviewID() {
                return reviewID;
        }
        public void setReviewID(int reviewID) {
                this.reviewID = reviewID;
        }
        public int getSprintID() {
```

```
		return sprintID;
	}
	public void setSprintID(int sprintID) {
		this.sprintID = sprintID;
	}
	public int getStoryID() {
		return storyID;
	}
	public void setStoryID(int storyID) {
		this.storyID = storyID;
	}
	public String getReview() {
		return review;
	}
	public void setReview(String review) {
		this.review = review;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public String getStoryName() {
		return storyName;
	}
	public void setStoryName(String storyName) {
		this.storyName = storyName;
	}

}
```

Listing 17: ReviewRowMapper.java

```
/**
 * ReviewRowMapper.java
 *
 * Row mapper for reviews table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class ReviewRowMapper implements RowMapper {

	@Override
	public Object mapRow(ResultSet rs, int row) throws SQLException {
		// TODO Auto-generated method stub
		Review review = new Review();
		review.setReview(rs.getString("review"));
		review.setReviewID(rs.getInt("reviewID"));
		review.setSprintID(rs.getInt("sprintID"));
		review.setStoryID(rs.getInt("storyID"));
		return review;
	}


}
```

Listing 18: Sprint.java

```
/**
 * Sprint.java
 *
 * Model for object Sprint
 */
package com.scrumprimer.model;

import java.util.Date;

public class Sprint {
	int sprintID;
	int projectID;
	int sprintNumber;
	String status;
	String sprintReview;
	String dateStarted;
	public String getDateStarted() {
		return dateStarted;
	}
	public void setDateStarted(String dateStarted) {
		this.dateStarted = dateStarted;
	}
	public String getSprintReview() {
		return sprintReview;
	}
	public void setSprintReview(String sprintReview) {
		this.sprintReview = sprintReview;
	}
	public int getSprintID() {
		return sprintID;
	}
	public void setSprintID(int sprintID) {
		this.sprintID = sprintID;
	}
	public int getProjectID() {
		return projectID;
	}
	public void setProjectID(int projectID) {
		this.projectID = projectID;
	}
	public int getSprintNumber() {
		return sprintNumber;
	}
	public void setSprintNumber(int sprintNumber) {
		this.sprintNumber = sprintNumber;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}

}
```

Listing 19: SprintRowMapper.java

```
/**
 * SprintRowMapper.java
 *
 * Row mapper for sprints table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class SprintRowMapper implements RowMapper {

	@Override
	public Object mapRow(ResultSet rs, int row) throws SQLException {
		// TODO Auto-generated method stub
		Sprint sprint = new Sprint();
		sprint.setProjectID(rs.getInt("projectID"));
		sprint.setSprintID(rs.getInt("sprintID"));
		sprint.setSprintNumber(rs.getInt("sprintNumber"));
		sprint.setStatus(rs.getString("status"));
		sprint.setSprintReview(rs.getString("sprintReview"));
		sprint.setDateStarted(rs.getString("dateStarted"));
		return sprint;
	}

}
```

Listing 20: SprintStory.java

```
/**
 * SprintStory.java
 *
 * Model for object SprintStory
 */

package com.scrumprimer.model;

public class SprintStory {
	int sprintID;
	int storyID;
	public int getSprintID() {
		return sprintID;
	}
	public void setSprintID(int sprintID) {
		this.sprintID = sprintID;
	}
	public int getStoryID() {
		return storyID;
	}
	public void setStoryID(int storyID) {
		this.storyID = storyID;
	}
}
```

Listing 21: SprintStoryRowMapper.java

```
/**
 * SprintStoryRowMapper.java
 *
 * Row mapper for sprintstory table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
```

```java
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class SprintStoryRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                SprintStory sprint = new SprintStory();
                sprint.setSprintID(rs.getInt("sprintID"));
                sprint.setStoryID(rs.getInt("storyID"));
                return sprint;
        }

}
```

Listing 22: TaskForm.java

```java
/**
 * TaskForm.java
 *
 * Model for multiple tasks object
 */

package com.scrumprimer.model;

import org.springframework.util.AutoPopulatingList;

public class TaskForm {
        private AutoPopulatingList<Task> tasks;

        public AutoPopulatingList<Task> getTasks() {
                return tasks;
        }

        public void setTasks(AutoPopulatingList<Task> tasks) {
                this.tasks = tasks;
        }
}
```

Listing 23: TaskRowMapper.java

```java
/**
 * TaskRowMapper.java
 *
 * Row mapper for tasks table
 */

package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class TaskRowMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int row) throws SQLException {
                // TODO Auto-generated method stub
                Task task = new Task();
                task.setTaskName(rs.getString("taskName"));
                task.setEstimate(rs.getInt("estimate"));
                task.setSprintID(rs.getInt("sprintID"));
                task.setTaskID(rs.getInt("taskID"));
                task.setStatus(rs.getString("status"));
                task.setStoryID(rs.getInt("storyID"));
                task.setAccountID(rs.getInt("accountID"));

                return task;
        }

}
```

Listing 24: User.java

```java
/**
 * User.java
 *
 * Model for object User
 */

package com.scrumprimer.model;

public class User {
        private String firstName;
        private String lastName;
        private String email;
        private String password;
        private Integer userID;
        private String username;
        private String newPassword;
        private String newPassword2;
        private String role;
        private String userType;

        public String getUserType() {
                return userType;
        }
        public void setUserType(String userType) {
                this.userType = userType;
        }
        public String getRole() {
                return role;
        }
        public void setRole(String role) {
                this.role = role;
        }
        public String getNewPassword() {
                return newPassword;
        }
        public void setNewPassword(String newPassword) {
                this.newPassword = newPassword;
        }
        public String getNewPassword2() {
                return newPassword2;
        }
        public void setNewPassword2(String newPassword2) {
                this.newPassword2 = newPassword2;
        }
        public String getFirstName() {
                return firstName;
        }
        public void setFirstName(String firstName) {
                this.firstName = firstName;
        }
        public String getUsername() {
                return username;
        }
        public void setUsername(String username) {
                this.username = username;
        }
        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 String getPassword() {
                return password;
        }
        public void setPassword(String password) {
                this.password = password;
        }
        public Integer getUserID() {
                return userID;
        }
        public void setUserID(Integer userID) {
                this.userID = userID;
        }


}
```

Listing 25: UserDAO.java

```
/**
 * UserDAO.java
 */
package com.scrumprimer.model;

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

import javax.servlet.http.HttpSession;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.support.rowset.SqlRowSet;
import org.springframework.security.authentication.encoding.PasswordEncoder;

import com.mysql.jdbc.ResultSet;
import com.scrumprimer.model.User;
import com.scrumprimer.model.UserRowMapper;
import com.scrumprimer.utility.SessionUtility;

public class UserDAO {

        @Autowired
        private DataSource dataSource;
        private JdbcTemplate jdbcTemplateObject;
         @Autowired
        private PasswordEncoder passwordEncoder;
         @Autowired
```

```
     private ProjectDAO projectDAO;

    public boolean insertUser(User newUser){
        jdbcTemplateObject = new JdbcTemplate(dataSource);
        String password = passwordEncoder.encodePassword(newUser.getPassword(), null);

        if(isUsernameExist(newUser.getUsername())){
            return false;
        }

      String SQL = "insert into accounts (username, password, firstName, lastName, email, type) values (?, ?, ?, ?, ?,?)";

      jdbcTemplateObject.update( SQL,new Object[]{newUser.getUsername(), password, newUser.getFirstName(), newUser.getLastName(), newUser.getEmail(), newUser.getUserType()});
       return true;

    }

    public void addInvite(ArrayList<User> users, HttpSession session){
        jdbcTemplateObject = new JdbcTemplate(dataSource);

        for(User user : users){
            if(isEmailExist(user.getEmail())){
                String query = "SELECT * FROM accounts WHERE email = ?";
                User user2 = jdbcTemplateObject.queryForObject(query, new Object[]{user.getEmail()}, new UserRowMapper());
                Member member = new Member();
                member.setAccountID(user2.getUserID());
                member.setProjectID((Integer) (session.getAttribute(SessionUtility.SESSION_PROJECT)));
                member.setRole(user.getRole());

                projectDAO.addMember(member);
            }
            else{
                String query = "INSERT into invite (firstName, lastName, projectID, email, role) values (?, ?, ?, ?, ?)";
                jdbcTemplateObject.update(query, new Object[]{user.getFirstName(), user.getLastName(), session.getAttribute(SessionUtility.SESSION_PROJECT), user.getEmail(), user.getRole()});
            }
        }



    }

    public User getUser(User loginUser){
        jdbcTemplateObject = new JdbcTemplate(dataSource);
        String password = passwordEncoder.encodePassword(loginUser.getPassword(), null);

        jdbcTemplateObject = new JdbcTemplate(dataSource);

        String query = "Select * from accounts where username = ? and password = ?";


        User user = jdbcTemplateObject.queryForObject(query, new Object[]{loginUser.getUsername(), password}, new UserRowMapper());
        return user;
    }

    public User getUser2(Integer accountID){
        jdbcTemplateObject = new JdbcTemplate(dataSource);

        String query = "Select * from accounts where accountID = ?";


        User user = jdbcTemplateObject.queryForObject(query, new Object[]{accountID}, new UserRowMapper());
        return user;
    }

    public int getUserIDViaEmail(String email){
        jdbcTemplateObject = new JdbcTemplate(dataSource);
        String query = "Select * from accounts where email = ?";


        User user = jdbcTemplateObject.queryForObject(query, new Object[]{email}, new UserRowMapper());
        return user.getUserID();
    }
    public boolean isAccountExist(String username, String password){
        jdbcTemplateObject = new JdbcTemplate(dataSource);
        String encodePassword = passwordEncoder.encodePassword(password, null);
        System.out.println(encodePassword);

        String query = "Select COUNT(*) from accounts where username = ? and password = ?";
        @SuppressWarnings("deprecation")
        int row = jdbcTemplateObject.queryForInt(query, new Object[]{username, encodePassword});
        System.out.println(row);
```

```
		if(row != 1)
			return false;
		else
			return true;
	}

	public boolean isUsernameExist(String username){
		jdbcTemplateObject = new JdbcTemplate(dataSource);

		String query = "Select COUNT(*) from accounts where username = ?";
		@SuppressWarnings("deprecation")
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{username});
		System.out.println(row);
		if(row != 1)
			return false;
		else
			return true;
	}

	public boolean isEmailExist(String email){
		jdbcTemplateObject = new JdbcTemplate(dataSource);

		String query = "Select COUNT(*) from accounts where email = ?";
		@SuppressWarnings("deprecation")
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{email});
		if(row != 1)
			return false;
		else
			return true;
	}

	public boolean isEmailInvite(String email){
		jdbcTemplateObject = new JdbcTemplate(dataSource);

		String query = "Select COUNT(*) from invite where email = ?";
		int row = jdbcTemplateObject.queryForInt(query, new Object[]{email});
		if(row == 0)
			return false;
		else
			return true;
	}


	public void updateUser(User user){
		jdbcTemplateObject = new JdbcTemplate(dataSource);

		String query = "UPDATE accounts SET username = ?, firstName = ?, email =
			?, lastName = ? WHERE accountID = ?";
		jdbcTemplateObject.update(query, new Object[]{user.getUsername(), user.
			getFirstName(), user.getEmail(), user.getLastName(), user.getUserID
			()});
		return;
	}

	public void updatePassword(User user){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String encodePassword = passwordEncoder.encodePassword(user.
			getNewPassword(), null);
		String query = "UPDATE accounts SET password = ? WHERE accountID = ?";
		jdbcTemplateObject.update(query, new Object[]{encodePassword, user.
			getUserID()});
		return;
	}

	public List<User> getAllUser(){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "SELECT * FROM accounts";
		List<User> users = jdbcTemplateObject.query(query,new UserRowMapper());
		return users;
	}

	public void deleteUser(int accountID){
		jdbcTemplateObject = new JdbcTemplate(dataSource);
		String query = "DELETE FROM accounts where accountID = ?";
		jdbcTemplateObject.update(query, accountID);
		return;
	}

}
```

Listing 26: UserForm.java

```
/**
 * UserForm.java
 *
 * Model for multiple user object
 */
package com.scrumprimer.model;

import java.util.ArrayList;

public class UserForm {

	private ArrayList<User> users;

	public ArrayList<User> getUsers() {
		return users;
	}

	public void setUsers(ArrayList<User> users) {
		this.users = users;
	}

}
```

Listing 27: UserRowMapper.java

```
/**
 * UserRowMapper.java
 *
 * Row mapper for users table
 */
package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class UserRowMapper implements RowMapper {

	@Override
	public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
		// TODO Auto-generated method stub
		User user = new User();
		user.setUserID(rs.getInt("accountID"));
		user.setPassword(rs.getString("password"));
		user.setUsername(rs.getString("username"));
		user.setEmail(rs.getString("email"));
		user.setFirstName(rs.getString("firstName"));
		user.setLastName(rs.getString("lastName"));
		user.setUserType(rs.getString("type"));
		return user;
	}

}
```

Listing 28: UserSprintForm.java

```
/**
 * UserSprintForm.java
 *
 * Model for multiple sprintstory object
 */
package com.scrumprimer.model;

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

public class UserSprintForm {

	ArrayList<SprintStory> sprintStories;

	public ArrayList<SprintStory> getSprintStories() {
		return sprintStories;
	}

	public void setSprintStories(ArrayList<SprintStory> sprintStories) {
		this.sprintStories = sprintStories;
	}
}
```

Listing 29: UserStory.java

```
/**
 * UserStory.java
 *
 * Model for object UserStory
 */
package com.scrumprimer.model;

public class UserStory {
	private int storyID;
	private String detail;
	private String storyName;
	private String status;
	private int projectID;
	public int getStoryID() {
		return storyID;
	}
	public void setStoryID(int storyID) {
		this.storyID = storyID;
	}
	public String getDetail() {
		return detail;
	}
	public void setDetail(String detail) {
		this.detail = detail;
	}
	public String getStoryName() {
		return storyName;
	}
```

```java
        public void setStoryName(String storyName) {
                this.storyName = storyName;
        }
        public String getStatus() {
                return status;
        }
        public void setStatus(String status) {
                this.status = status;
        }
        public int getProjectID() {
                return projectID;
        }
        public void setProjectID(int projectID) {
                this.projectID = projectID;
        }



}
```

Listing 30: UserStoryForm.java

```java
/**
 * UserStoryForm.java
 *
 * Model for userstory object
 */
package com.scrumprimer.model;

import java.util.ArrayList;

import org.springframework.util.AutoPopulatingList;

public class UserStoryForm {

        private ArrayList<UserStory> stories;

        public ArrayList<UserStory> getStories() {
                return stories;
        }

        public void setStories(ArrayList<UserStory> stories) {
                this.stories = stories;
        }




}
```

Listing 31: UserStoryMapper.java

```java
/**
 * UserStoryMapper.java
 *
 * Row mapper for userstories table
 */
package com.scrumprimer.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class UserStoryMapper implements RowMapper {

        @Override
        public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
                UserStory story = new UserStory();
                story.setDetail(rs.getString("detail"));
                story.setProjectID(rs.getInt("projectID"));
                story.setStatus(rs.getString("status"));
                story.setStoryID(rs.getInt("storyID"));
                story.setStoryName(rs.getString("storyName"));
                return story;

        }

}
```

2. **Controllers**

Listing 32: AdminController.java

```java
/**
 * AdminController.java
 *
 * Controller for all Admin functionalities. Involve view accounts, delete accounts and
    update account
 *
 */
package com.scrumprimer.controller;

```

```java
import java.util.List;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.security.authentication.encoding.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.scrumprimer.model.User;
import com.scrumprimer.model.UserDAO;
import com.scrumprimer.utility.SessionUtility;

@Controller
@SessionAttributes
@EnableWebMvc



public class AdminController {
        @Autowired
        UserDAO userDAO;
         @Autowired
        private PasswordEncoder passwordEncoder;

         @RequestMapping("/Accounts")
         public ModelAndView viewAccounts(HttpSession session){
                 ModelAndView view = new ModelAndView("accounts");
                         view.addObject("edituser", new User());
                         view.addObject("edituser2", new User());
                         List<User> users = userDAO.getAllUser();
                         view.addObject("users", users);
                         view.addObject("newUser", new User());
                         view.addObject("edit", new User());
                         view.addObject("activeaccounts", "active");
                         return view;
         }

         @RequestMapping("/deleteAccount")
         public String deleteAccounts(HttpSession session, @RequestParam("aid")int
              accountID, RedirectAttributes map){
                 userDAO.deleteUser(accountID);
                 map.addFlashAttribute("successfuldelete", "Account has been
                      successfully deleted");
                 return "redirect:Accounts";
         }

         @RequestMapping("/editAccount")
         public String editAccount(HttpSession session, @RequestParam("aid")int
              accountID, RedirectAttributes map){
                 User updateUser = userDAO.getUser2(accountID);
                 map.addFlashAttribute("updateuser", updateUser);
                 map.addFlashAttribute("update", "yes");
                 return "redirect:Accounts";
         }

         @RequestMapping("/updateAccount")
         public String updateAccount(HttpSession session, @ModelAttribute("edit")User
              user, RedirectAttributes map){
                 userDAO.updateUser(user);
                 map.addFlashAttribute("updateaccount", "Account has been successfully
                      updated");

                 return "redirect:Accounts";

         }

         @RequestMapping("/addAccount")
         public String addAccount(HttpSession session, @ModelAttribute("newUser")User
              user, RedirectAttributes map){
                 userDAO.insertUser(user);
                 map.addFlashAttribute("addaccount", "Account has been successfully
                      added");

                 return "redirect:Accounts";
         }
}
```

Listing 33: LoginController.java

```java
/**
 * LoginController.java
 *
 * Controller for logging in and logout.
 *
 */
package com.scrumprimer.controller;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.security.authentication.encoding.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.scrumprimer.model.User;
import com.scrumprimer.model.UserDAO;
import com.scrumprimer.utility.SessionUtility;

@Controller
@SessionAttributes
@EnableWebMvc

public class LoginController {
         @Autowired
        UserDAO userDAO;
          @Autowired
         private PasswordEncoder passwordEncoder;

         @RequestMapping(value = "/Login", method = RequestMethod.POST)
         public String addUser(@ModelAttribute("login")
                                                         User login, BindingResult result
                                                              , HttpSession session,
                                                              RedirectAttributes map) {

                 //userDAO.getUser(login);
                 if(userDAO.isAccountExist(login.getUsername(), login.getPassword())){
                         User user = new User();
                         user = userDAO.getUser(login);

                         session.setAttribute(SessionUtility.USERNAME, user.getUsername()
                              );
                         session.setAttribute(SessionUtility.SESSION_ID, user.getUserID()
                              );
                         session.setAttribute(SessionUtility.SESSION_FIRSTNAME, user.
                              getFirstName());
                         session.setAttribute(SessionUtility.SESSION_LASTNAME, user.
                              getLastName());
                         session.setAttribute(SessionUtility.SESSION_EMAIL, user.getEmail
                              ());
                         session.setAttribute(SessionUtility.PASSWORD, login.getPassword
                              ());

                         if(user.getUserType().equalsIgnoreCase("admin") ){
                                 session.setAttribute(SessionUtility.SESSION_USERTYPE,
                                      user.getUserType());
                                 return "redirect:Accounts";
                         }
                         return "redirect:Project";
                 }
                 else{
                         map.addFlashAttribute("errorlogin", "Invalid username and
                              password");
                         return "redirect:Home";
                 }
         }

         @RequestMapping("/Logout")
         public String logout(HttpSession session){
                 session.invalidate();
                 return "redirect:Home";
         }
}
```

Listing 34: ProjectController.java

```
/**
 * ProjectController.java
 *
 * Controller for project's page.
 *
 */
package com.scrumprimer.controller;

import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;




import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.scrumprimer.model.Member;
import com.scrumprimer.model.NotificationDAO;
import com.scrumprimer.model.Project;
import com.scrumprimer.model.ProjectDAO;
import com.scrumprimer.model.User;
import com.scrumprimer.model.UserDAO;
import com.scrumprimer.model.UserForm;
import com.scrumprimer.utility.EmailNotification;
import com.scrumprimer.utility.RoleUtility;
import com.scrumprimer.utility.SessionUtility;

@Controller
@SessionAttributes


public class ProjectController {
        @Autowired
        UserDAO userDAO;

        @Autowired
        ProjectDAO projectDAO;

        @Autowired
        NotificationDAO notifDAO;


        @RequestMapping("/Project")
        public ModelAndView viewProject(HttpSession session){
                ModelAndView project = new ModelAndView("projects");
                project.addObject("edituser", new User());
                project.addObject("edituser2", new User());
                List<Member> projects = new ArrayList<Member>();
                List<Project> projectDetails = new ArrayList<Project>();

                if(!projectDAO.isAccountHasProject((Integer)session.getAttribute(
                     SessionUtility.SESSION_ID))){
                        project.addObject("noexisting","No existing projects");
                }
                else{
                         projects = projectDAO.getProject((Integer) session.getAttribute
                             (SessionUtility.SESSION_ID));
                         projectDetails = projectDAO.getProjectDetails(projects);

                }
                project.addObject("projects", projects);
                project.addObject("projectDetails", projectDetails);
                project.addObject("newProject", new Project());
                project.addObject("activeproject", "active");
                return project;
        }

        @RequestMapping("/editProject")
        public String editProject(HttpSession session, @RequestParam("pid")int projectID
             , RedirectAttributes map){
                Project updateProject = projectDAO.getProjectByProjectID(projectID);
                map.addFlashAttribute("updateProject", updateProject);
                map.addFlashAttribute("update", "yes");
                return "redirect:Project";
        }

        @RequestMapping("/updateProject")
        public String updateProject(HttpSession session, @ModelAttribute("newProject")
             Project newProject, BindingResult result, RedirectAttributes map){
                projectDAO.updateProject(newProject);
                map.addFlashAttribute("projectupdated", "Project has been updated");
                return "redirect:Project";
        }

        @RequestMapping("/viewProject")
        public String view_project(HttpSession session, @RequestParam("pid")int
             projectID, RedirectAttributes map){
                session.setAttribute(SessionUtility.SESSION_PROJECT, projectID);
                String projectName = projectDAO.getProjectName(projectID);
                session.setAttribute(SessionUtility.SESSION_PROJECT_NAME, projectName);
                String role = projectDAO.getRole((Integer) session.getAttribute(
                     SessionUtility.SESSION_ID), projectID);
                session.setAttribute(SessionUtility.SESSION_ROLE, role);
                session.setAttribute(SessionUtility.SESSION_PROJECTSTATUS, projectDAO.
                     getProjectByProjectID((Integer)session.getAttribute(SessionUtility.
                     SESSION_PROJECT)).getStatus());
                if(projectDAO.isCurrentSprintExist((Integer)session.getAttribute(
                        SessionUtility.SESSION_PROJECT)))
                        map.addFlashAttribute("currentSprint", "true");
                if(role.equalsIgnoreCase(RoleUtility.PO)){
                        return "redirect:viewProductBacklog";
                }
                else{

                        return "redirect:viewAllSprint";
                }

        }

        @RequestMapping("/closeProject")
        public String closeProject(HttpSession session){
                projectDAO.closeProject((Integer)session.getAttribute(SessionUtility.
                    SESSION_PROJECT));
                return "redirect:/Project";
        }

        @RequestMapping("/newProject")
        public ModelAndView newProject(HttpSession session){
                ModelAndView newProject = new ModelAndView("add_project");
                newProject.addObject("newProject", new Project());
                newProject.addObject("edituser", new User());
                newProject.addObject("edituser2", new User());
                return newProject;
        }

        @RequestMapping("/inviteMembers")
        public ModelAndView addProject(@ModelAttribute("newProject")
        Project newProject, BindingResult result,HttpSession session){
                int projectid = projectDAO.addNewProject(newProject);
                Member newMember = new Member();
                newMember.setAccountID((Integer)session.getAttribute(SessionUtility.
                    SESSION_ID));
                newMember.setProjectID(projectid);
                newMember.setRole(RoleUtility.SM);
                projectDAO.addMember(newMember);
                ModelAndView members = new ModelAndView("new_member");
                members.addObject("numberOfTM", newProject.getNumberOfTM());
                members.addObject("numberOfPO", newProject.getNumberOfPO());
                members.addObject("projectID", projectid);
                members.addObject("newMember", new Member());
                members.addObject("edituser", new User());
                members.addObject("edituser2", new User());
                ArrayList<User> team_members = new ArrayList<User>();
                for(int i = 0; i < newProject.getNumberOfTM()+newProject.getNumberOfPO()
                    ; i++){
                        team_members.add(new User());
                }
                UserForm teamForm = new UserForm();
                teamForm.setUsers(team_members);
                members.addObject("teamForm", teamForm);

                session.setAttribute(SessionUtility.SESSION_PROJECT, projectid);
                session.setAttribute(SessionUtility.SESSION_PROJECT_NAME, newProject.
                    getProjectName());
                return members;
        }

        @RequestMapping("/addMembers")
        public String addMembers(@ModelAttribute("teamForm")UserForm user, BindingResult
             result,HttpSession session, RedirectAttributes map){
                ArrayList<User> team_members = user.getUsers();
                for(User user2: team_members){
                        System.out.println(user2.getEmail());
                }
        EmailNotification invite = new EmailNotification();
        if(invite.emailInvite(team_members))
                map.addFlashAttribute("invite","Invitation sent!" );

        userDAO.addInvite(team_members, session);


        map.addFlashAttribute("addproject", "Project was successfully added.");
                return "redirect:Project";
        }



}
```

Listing 35: PublicController.java

```
/**
 * PublicController.java
 *
 * Controller for public users
 *
 */
package com.scrumprimer.controller;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;

import com.scrumprimer.model.User;

@Controller
@SessionAttributes

public class PublicController {
        @RequestMapping("/Home")
        public ModelAndView viewHomeRegular(){
                ModelAndView homeView = new ModelAndView("home_regular");
                homeView.addObject("login", new User());
                homeView.addObject("user", new User());
                return homeView;
        }

}
```

Listing 36: RegisterController.java

```
/**
 * RegisterController.java
 *
 * Controller for adding new users
 *
 */
package com.scrumprimer.controller;

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

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.security.authentication.encoding.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;



import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.scrumprimer.model.Dummy;
import com.scrumprimer.model.Member;
import com.scrumprimer.model.Notification;
import com.scrumprimer.model.NotificationDAO;
import com.scrumprimer.model.ProjectDAO;
import com.scrumprimer.model.User;
import com.scrumprimer.model.UserDAO;
import com.scrumprimer.utility.NotificationUtility;
import com.scrumprimer.utility.RoleUtility;
import com.scrumprimer.utility.SessionUtility;


@Controller
@SessionAttributes
@EnableWebMvc

public class RegisterController {

        @Autowired
        UserDAO userDAO;

         @Autowired
        private PasswordEncoder passwordEncoder;

         @Autowired
         private ProjectDAO projectDAO;

         @Autowired
         private NotificationDAO notifDAO;


        @RequestMapping(value = "/addUser", method = RequestMethod.POST)
        public String addUser(@ModelAttribute("user")
                                                        User user, BindingResult result,
                                                                RedirectAttributes map) {

                boolean email = userDAO.isEmailExist(user.getEmail());
                user.setUserType("regular");
                boolean success = userDAO.insertUser(user);

                if(email){
                        map.addFlashAttribute("errormessage2", "Email Address is already
                                registered");
```

```
            map.addAttribute("email-taken");
            map.addFlashAttribute("successmessage", null);
            map.addFlashAttribute("errormessage", null);

        }
        else if(!success){
            map.addFlashAttribute("errormessage", "Username is already taken");
            map.addAttribute("username-taken");
            map.addFlashAttribute("successmessage", null);
            map.addFlashAttribute("errormessage2",null);


        }
        else{
            map.addFlashAttribute("successmessage", "Account already made");
            if(userDAO.isEmailInvite(user.getEmail())){
                List<Dummy> projectPending = projectDAO.getPendingProject(user.getEmail());
                int accountID = userDAO.getUserIDViaEmail(user.getEmail());
                for(Dummy dummy : projectPending){
                    Member member = new Member();
                    member.setAccountID(accountID);
                    member.setProjectID(dummy.getProjectID());
                    member.setRole(dummy.getRole());
                    projectDAO.addMember(member);
                    Notification notif = new Notification();
                    notif.setAccountID(accountID);
                    notif.setProjectID(dummy.getProjectID());
                    notif.setType(NotificationUtility.NEW_MEMBER);
                    for(Member member2 : projectDAO.projectMembers(dummy.getProjectID()))
                    if(member2.getRole().equalsIgnoreCase(RoleUtility.SM)){
                        notifDAO.insertNotification(notif, member2.getAccountID());
                    }


                }
            }
            map.addFlashAttribute("errormessage2",null);
            map.addFlashAttribute("errormessage", null);
        }
        return "redirect:Home";
    }

    @RequestMapping(value = "/editUser", method = RequestMethod.POST)
    public String editUser(@ModelAttribute("edituser")
    User editUser, BindingResult result, RedirectAttributes map, HttpSession session) {
        editUser.setUserID((Integer)session.getAttribute(SessionUtility.SESSION_ID));
        userDAO.updateUser(editUser);

        User user = userDAO.getUser2(editUser.getUserID());

        session.setAttribute(SessionUtility.USERNAME, user.getUsername());
        session.setAttribute(SessionUtility.SESSION_ID, user.getUserID());
        session.setAttribute(SessionUtility.SESSION_FIRSTNAME, user.getFirstName());
        session.setAttribute(SessionUtility.SESSION_LASTNAME, user.getLastName());
        session.setAttribute(SessionUtility.SESSION_EMAIL, user.getEmail());

        return "redirect:Project";

    }

    @RequestMapping(value = "/editPassword", method = RequestMethod.POST)
    public String editPassword(@ModelAttribute("edituser2")
    User editUser, BindingResult result, RedirectAttributes map, HttpSession session) {
        editUser.setUserID((Integer)session.getAttribute(SessionUtility.SESSION_ID));
        System.out.println(editUser.getNewPassword());
        userDAO.updatePassword(editUser);
        session.setAttribute(SessionUtility.PASSWORD, editUser.getNewPassword());
        return "redirect:Project";

    }

}
```

Listing 37: ScrumController.java

```
/**
 * ScrumController.java
 *
 * Controller that involves the scrum functionalities.
 */
package com.scrumprimer.controller;

import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.util.AutoPopulatingList;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.scrumprimer.model.Activity;
import com.scrumprimer.model.BCModel;
import com.scrumprimer.model.Meeting;
import com.scrumprimer.model.MeetingForm;
import com.scrumprimer.model.Member;
import com.scrumprimer.model.Notification;
import com.scrumprimer.model.NotificationDAO;
import com.scrumprimer.model.Project;
import com.scrumprimer.model.ProjectDAO;
import com.scrumprimer.model.Review;
import com.scrumprimer.model.Sprint;
import com.scrumprimer.model.SprintStory;
import com.scrumprimer.model.Task;
import com.scrumprimer.model.TaskForm;
import com.scrumprimer.model.User;
import com.scrumprimer.model.UserDAO;
import com.scrumprimer.model.UserForm;
import com.scrumprimer.model.UserSprintForm;
import com.scrumprimer.model.UserStory;
import com.scrumprimer.model.UserStoryForm;
import com.scrumprimer.utility.DateHandler;
import com.scrumprimer.utility.NotificationUtility;
import com.scrumprimer.utility.RoleUtility;
import com.scrumprimer.utility.SessionUtility;
import com.scrumprimer.utility.StatusUtility;

@Controller
@SessionAttributes

public class ScrumController {
        @Autowired
        UserDAO userDAO;

        @Autowired
        ProjectDAO projectDAO;

        @Autowired
        NotificationDAO notifDAO;



        @RequestMapping("/viewAllSprint")
        public ModelAndView viewAllSprint(HttpSession session){
                ModelAndView view = new ModelAndView("all_sprint");
                view.addObject("sprint", "active");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                int sprintNumber = projectDAO.getLatestSprintNumber((Integer)session.
                     getAttribute(SessionUtility.SESSION_PROJECT)) + 1;
                view.addObject("sprintNumber", sprintNumber);
                view.addObject("newSprint", new Sprint());
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                     SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                     SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                     session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                     getAttribute(SessionUtility.SESSION_PROJECT)));
                if(projectDAO.isCurrentSprintExist((Integer)session.getAttribute(
                     SessionUtility.SESSION_PROJECT))){
                        view.addObject("currentSprint", "yes");
                }
                if(!projectDAO.isUserStoryExisting((Integer)session.getAttribute(
                     SessionUtility.SESSION_PROJECT))){
                                view.addObject("nostories", "No User Stories");
                }
                if(projectDAO.isSprintExist((Integer)session.getAttribute(SessionUtility
                        .SESSION_PROJECT))){
                        List<Sprint> allSprints = projectDAO.getAllSprint((Integer)
                            session.getAttribute(SessionUtility.SESSION_PROJECT));
                        view.addObject("allSprints", allSprints );

                }
                else{
                        view.addObject("nosprints", "No Sprints");
                }

                return view;
        }


        @RequestMapping("/viewProductBacklog")
        public ModelAndView viewProductBacklog(HttpSession session){
                ModelAndView view = new ModelAndView("product_backlog");
                view.addObject("productbacklog", "active");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                    SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                    SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                    session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                    getAttribute(SessionUtility.SESSION_PROJECT)));
                if(projectDAO.isUserStoryExisting((Integer)session.getAttribute(
                    SessionUtility.SESSION_PROJECT))){
                        List<UserStory> stories = projectDAO.getUserStory((Integer)
                            session.getAttribute(SessionUtility.SESSION_PROJECT));
                        view.addObject("stories", stories);
                }
                else{
                        view.addObject("nostory", "Empty Product Backlog");
                }
                return view;
        }

        @RequestMapping("/editProductBacklog")
        public ModelAndView editProductBacklog(HttpSession session){
                ModelAndView view = new ModelAndView("edit_product_backlog");
                view.addObject("productbacklog", "active");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                ArrayList<UserStory> stories = new ArrayList<UserStory>();
                UserStoryForm storyForm = new UserStoryForm();
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                    SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                    SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                    session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                    getAttribute(SessionUtility.SESSION_PROJECT)));
                if(projectDAO.isUserStoryExisting((Integer)session.getAttribute(
                    SessionUtility.SESSION_PROJECT))){
                        stories = (ArrayList<UserStory>) projectDAO.getUserStory((
                            Integer)session.getAttribute(SessionUtility.SESSION_PROJECT
                            ));

                }
                storyForm.setStories(stories);
                view.addObject("storyForm", storyForm);
                return view;
        }

        @RequestMapping("/saveProductBacklog")
        public String saveProductBacklog(HttpSession session, @ModelAttribute("storyForm
            ") UserStoryForm storyForm, BindingResult result){
                ArrayList<UserStory> stories = storyForm.getStories();
                for(UserStory story : stories){
                        System.out.println(story.getStoryID());
                }
                projectDAO.updateUserStories(stories);
                return "redirect:viewProductBacklog";
        }

        @RequestMapping("/insertUserStories")
        public String insertUserStories(HttpSession session, @ModelAttribute("newStory")
            UserStoryForm newStory, BindingResult result){
                ArrayList<UserStory> stories = newStory.getStories();
                projectDAO.insertUserStories(stories);
                return "redirect:viewProductBacklog";
        }



        @RequestMapping("/addUserStories")
        public ModelAndView addUserStories(HttpSession session, @RequestParam("
            numberOfStories")int story){
                ModelAndView view = new ModelAndView("add_product_backlog");
                view.addObject("productbacklog", "active");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                view.addObject("numberOfStory", story);
                ArrayList<UserStory> newStories = new ArrayList<UserStory>();
                for(int i = 0; i < story; i++){
```

```
                newStories.add(new UserStory());
            }
            UserStoryForm newUserStories = new UserStoryForm();
            newUserStories.setStories(newStories);
            view.addObject("newStory", newUserStories);
            int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                SessionUtility.SESSION_ID));
            view.addObject("notifs",notifs);
            view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                session.getAttribute(SessionUtility.SESSION_PROJECT)));
            view.addObject("members", projectDAO.projectMembers((Integer)session.
                getAttribute(SessionUtility.SESSION_PROJECT)));
            return view;
    }

    @RequestMapping("/viewSprintReview")
    public ModelAndView viewSprintReview(HttpSession session, @RequestParam("sid")
        int sprintID){
            ModelAndView view = new ModelAndView("sprint_review_view");
            view.addObject("edituser", new User());
            view.addObject("edituser2", new User());
            view.addObject("sprintID",sprintID);
            int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                SessionUtility.SESSION_ID));
            view.addObject("notifs",notifs);
            view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
                getSprintNumber());
            view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                session.getAttribute(SessionUtility.SESSION_PROJECT)));
            view.addObject("members", projectDAO.projectMembers((Integer)session.
                getAttribute(SessionUtility.SESSION_PROJECT)));
            List<Review> reviews = projectDAO.getSprintReview(sprintID);
            for(Review review : reviews){
                review.setStoryName(projectDAO.getUserStoryByID(review.
                    getStoryID()).getStoryName());
                review.setStatus(projectDAO.getUserStoryByID(review.getStoryID()
                    ).getStatus());
            }
            view.addObject("reviews",reviews);
            return view;
    }

    @RequestMapping("/sprintReview")
    public ModelAndView sprintReview(HttpSession session, @RequestParam("sid")int
        sprintID){
            ModelAndView view = new ModelAndView("sprint_review");
            view.addObject("edituser", new User());
            view.addObject("edituser2", new User());
            view.addObject("sprintID",sprintID);
            int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                SessionUtility.SESSION_ID));
            view.addObject("notifs",notifs);
            view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
                getSprintNumber());
            MeetingForm reviews = new MeetingForm();
            List<UserStory> stories = projectDAO.getSprintStories(sprintID);
            ArrayList<Review> review = new ArrayList<Review>();
            for(int i = 0; i < stories.size(); i++){
                Review review2 = new Review();
                review2.setStoryID(stories.get(i).getStoryID());
                review2.setStoryName(stories.get(i).getStoryName());
                review.add(review2);
            }
            reviews.setReviews(review);
            view.addObject("reviewForm", reviews);
            view.addObject("stories", stories);

            return view;
    }

    @RequestMapping("/addReview")
    public String addReview(@ModelAttribute("reviewForm")MeetingForm reviewForm,
        BindingResult result, HttpSession session, @RequestParam("sid")int sprintID
        ){
            List<Review> reviews = reviewForm.getReviews();
            for(Review review : reviews){
                projectDAO.updateStoryStatus(review.getStatus(), review.
                    getStoryID());
                projectDAO.insertSprintReview(review);
            }
            projectDAO.closeSprint(sprintID);
            return "redirect:/viewAllSprint";
    }

    @RequestMapping("/viewSprintBacklog")
    public ModelAndView viewSprintBacklog(HttpSession session, @RequestParam("sid")
        int sprintID){
            ModelAndView view = new ModelAndView("sprint_backlog");
            notifSprintReview();
            view.addObject("edituser", new User());
            view.addObject("edituser2", new User());
            view.addObject("sprintID",sprintID);
            int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                SessionUtility.SESSION_ID));
            view.addObject("notifs",notifs);
```

```
                view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
                     getSprintNumber());
                view.addObject("sprintstatus",projectDAO.getSprintBySprintID(sprintID).
                     getStatus());
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                     session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                     getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("remainingdays", DateHandler.getDaysDifference(
                     DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(
                     sprintID).getSprintReview()));
                view.addObject("newTask", new Task());
                List<UserStory> sprintStories = projectDAO.getSprintStories(sprintID);
                view.addObject("sprintstories", sprintStories);
                if(DateHandler.getDaysDifference(DateHandler.getCurrentDate(),
                     projectDAO.getSprintBySprintID(sprintID).getSprintReview()) == 0){
                        view.addObject("sprintreview", "yes");
                }
                if(!projectDAO.isTaskExisting(sprintID)){
                        view.addObject("pokerplanning", "yes");
                }
                else{
                        List<Task> tasks = projectDAO.getTasks(sprintID);
                        for(Task task : tasks){
                                //System.out.println(task.getStoryID());
                                task.setStoryName(projectDAO.getUserStoryByID(task.
                                     getStoryID()).getStoryName());
                                task.setVolunteer(userDAO.getUser2(task.getAccountID()).
                                     getFirstName());
                                List<Activity> activities = projectDAO.getAllActivities(
                                     task.getTaskID());
                                int sum = 0;
                                for(Activity act : activities){
                                        sum += act.getTime();
                                }
                                task.setActual(sum);
                        }
                        view.addObject("tasks",tasks);
                }
                return view;
        }

        @RequestMapping("/addTask")
        public String addTask(@ModelAttribute("newTask")Task task, BindingResult result,
              HttpSession session, RedirectAttributes map){
                task.setStatus(StatusUtility.NOT_STARTED);
                List<Task> tasks = new ArrayList<Task>();
                tasks.add(task);
                projectDAO.insertTasks(tasks);
                map.addFlashAttribute("taskadded", "New tasks has been added");
                return "redirect:viewSprintBacklog?sid="+task.getSprintID();
        }

        @RequestMapping("/viewScrumMeeting")
        public ModelAndView viewScrumMeeting(HttpSession session, @RequestParam("sid")
             int sprintID){
                ModelAndView view = new ModelAndView("scrummeeting");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                view.addObject("sprintID",sprintID);
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                     SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                     SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
                     getSprintNumber());
                view.addObject("meeting", new Meeting());
                view.addObject("sprintstatus",projectDAO.getSprintBySprintID(sprintID).
                     getStatus());
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                     session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                     getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("remainingdays", DateHandler.getDaysDifference(
                     DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(
                     sprintID).getSprintReview()));

                if(projectDAO.isScrumMeeting(sprintID)){
                        List<Meeting> meetings = projectDAO.getScrumMeeting(sprintID);
                        view.addObject("meetings", meetings);

                        }
                if(!projectDAO.isCurrentScrumMeeting(sprintID)){
                        view.addObject("currentmeeting", "no");
                }
                if(!projectDAO.isScrumMeetingAnswered(sprintID, (Integer)session.
                     getAttribute(SessionUtility.SESSION_ID))){
                                        view.addObject("fillmeeting", "no");
                }

                return view;
        }

        @RequestMapping("/viewScrumMeetingDetail")
        public ModelAndView viewScrumMeetingDetail(HttpSession session, @RequestParam("
             date")String date, @RequestParam("sid")int sprintID){
                ModelAndView view = new ModelAndView("scrummeeting_detail");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                view.addObject("sprintID",sprintID);
                view.addObject("date", date);
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                List<Meeting> details = projectDAO.getMeetingDetail(sprintID, date);
                for(Meeting det : details){
                        det.setMember(userDAO.getUser2(det.getAccountID()).getFirstName());
                }
                view.addObject("details", details);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT)));
                return view;
        }

        @RequestMapping("/addScrumMeeting")
        public String addScrumMeeting(HttpSession session,@ModelAttribute("meeting") Meeting meeting, BindingResult result){
                meeting.setDate(DateHandler.getCurrentDate());
                meeting.setAccountID((Integer)session.getAttribute(SessionUtility.SESSION_ID));
                projectDAO.insertScrumMeeting(meeting);
                Notification notif = new Notification();
                notif.setProjectID((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT));
                notif.setType(NotificationUtility.SCRUM_MEETING);
                notif.setSprintID(meeting.getSprintID());
                if(session.getAttribute(SessionUtility.SESSION_ROLE).equals(RoleUtility.SM)){
                        for(Member member : projectDAO.projectMembers((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT))){
                            if(member.getRole().equalsIgnoreCase(RoleUtility.TM))
                            notifDAO.insertNotification(notif, member.getAccountID());
                        }
                }
                return "redirect:/viewScrumMeeting?sid="+meeting.getSprintID();
        }

        @RequestMapping("/viewScrumBoard")
        public ModelAndView viewSprint(HttpSession session, @RequestParam("sid")int sprintID){
                ModelAndView view = new ModelAndView("sprint");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                view.addObject("sprintID",sprintID);
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).getSprintNumber());
                Sprint sprint = projectDAO.getSprintBySprintID(sprintID);
                String date2 = sprint.getDateStarted();
                sprint.setDateStarted(date2.substring(0,10));
                view.addObject("sprint", sprint);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("remainingdays", DateHandler.getDaysDifference(DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(sprintID).getSprintReview()));
                if(!projectDAO.isTaskExisting(sprintID)){
                        view.addObject("pokerplanning", "yes");
                }
                else{
                        List<Task> tasks = projectDAO.getTasks(sprintID);
                        List<BCModel> burnvalues = new ArrayList<BCModel>();

                        for(Task task : tasks){
                            //System.out.println(task.getStoryID());
                            task.setStoryName(projectDAO.getUserStoryByID(task.getStoryID()).getStoryName());
                            task.setVolunteer(userDAO.getUser2(task.getAccountID()).getFirstName());
                        }
                        int total = projectDAO.getTotalEstimate(sprintID);
                        String date;
                        if(DateHandler.getDaysDifference(DateHandler.getCurrentDate(), sprint.getSprintReview()) == 0){
                            date = sprint.getSprintReview();
                        }
                        else{
                            date = DateHandler.getCurrentDate();
                        }
                        for(int i = 0; i <= DateHandler.getDaysDifference(sprint.getDateStarted(), date); i++){
                            BCModel burn = new BCModel();
                            burn.setDay(i);
                            int sum = 0;
                            for(Task task : tasks){
                                sum += projectDAO.getTotalActivityPerDay(task.getTaskID(), DateHandler.getDayAdded(sprint.getDateStarted(), i));

                            }
                            total = total - sum;
                            burn.setTotal(total);
                            burnvalues.add(burn);
                        }
                        String burns = "["+burnvalues.get(0).getDay()+","+burnvalues.get
                            (0).getTotal()+"]";
                        if(burnvalues.size() > 1){
                        for(int i = 1; i < burnvalues.size(); i++){
                            String concat = ",["+burnvalues.get(i).getDay()+","+
                                burnvalues.get(i).getTotal()+"]";
                            burns = burns + concat;
                        }
                        }

                        view.addObject("tasks",tasks);
                        view.addObject("burnvalues", burns);
                        int totalDays = DateHandler.getDaysDifference(sprint.
                            getDateStarted(), sprint.getSprintReview());
                        view.addObject("totalDays", totalDays);
                        view.addObject("totalEstimate", projectDAO.getTotalEstimate(
                            sprintID));
                }
                return view;
        }



        @ModelAttribute("taskForm")
        public TaskForm getTaskForm(){
                TaskForm taskForm = new TaskForm();
                taskForm.setTasks(new AutoPopulatingList<Task>(Task.class));
                return taskForm;
        }

        @RequestMapping("/pokerPlanning")
        public ModelAndView pokerPlanning(HttpSession session, @RequestParam("sid")int
                sprintID, @RequestParam("rmng") int remaining){
                ModelAndView view = new ModelAndView("poker_planning");
                view.addObject("edituser", new User());
                view.addObject("edituser2", new User());
                view.addObject("sprintID",sprintID);
                int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
                        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
                        SessionUtility.SESSION_ID));
                view.addObject("notifs",notifs);
                TaskForm taskForm = new TaskForm();
                taskForm.setTasks(new AutoPopulatingList<Task>(Task.class));
                view.addObject("taskForm", taskForm);
                List<UserStory> stories = projectDAO.getSprintStories(sprintID);
                List<Member> members = projectDAO.projectMembers((Integer) session.
                        getAttribute(SessionUtility.SESSION_PROJECT));
                for(Member member : members){
                        member.setName(userDAO.getUser2(member.getAccountID()).
                            getFirstName());
                }
                view.addObject("members", members);
                view.addObject("story",stories.get(remaining));
                view.addObject("ts", stories.size());
                view.addObject("remaining", remaining+1);
                view.addObject("project",projectDAO.getProjectByProjectID((Integer)
                        session.getAttribute(SessionUtility.SESSION_PROJECT)));
                view.addObject("members", projectDAO.projectMembers((Integer)session.
                        getAttribute(SessionUtility.SESSION_PROJECT)));

                return view;
        }

        @RequestMapping("/addTasks")
        public String addTasks(HttpSession session, @ModelAttribute("taskForm") TaskForm
             taskForm, BindingResult result, RedirectAttributes map, @RequestParam("
                rmng") int remainingStory, @RequestParam("ts") int totalStory,
                @RequestParam("sid")int sprintID ){
                List<Task> tasks = taskForm.getTasks();
                List<UserStory> stories = projectDAO.getSprintStories(sprintID);
                for(Task task : tasks){
                        task.setSprintID(sprintID);
                        task.setStatus(StatusUtility.NOT_STARTED);
                        task.setStoryID(stories.get(remainingStory-1).getStoryID());
                }
                projectDAO.insertTasks(tasks);
                if(remainingStory < totalStory){
                        return "redirect:/pokerPlanning?sid="+sprintID+"&rmng="+
                            remainingStory;
                }
                else{
                        return "redirect:/viewSprintBacklog?sid="+sprintID;
                }

        }


        @RequestMapping(method = RequestMethod.GET, value="/appendStoryView")
        protected String appendStoryField(@RequestParam Integer fieldId, ModelMap model,
             HttpSession session)
        {
                model.addAttribute("storyNumber", fieldId);
                List<Member> members = projectDAO.projectMembers((Integer) session.
				getAttribute(SessionUtility.SESSION_PROJECT));
			for(Member member : members){
				member.setName(userDAO.getUser2(member.getAccountID()).getFirstName());
				System.out.println(member.getName());
			}
			model.addAttribute("members", members);
			int sprintID = projectDAO.getLatestSprintNumber((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT));
			model.addAttribute("sprintID", sprintID);


			return "addTasksRow";
		}




		@RequestMapping("/addSprint")
		public ModelAndView addSprint(HttpSession session, @ModelAttribute("newSprint") Sprint newSprint, BindingResult result){
			projectDAO.insertSprint(newSprint);
			System.out.println(newSprint.getSprintReview());
			Sprint sprint = projectDAO.getCurrentSprint((Integer) session.getAttribute(SessionUtility.SESSION_PROJECT));
			ModelAndView view = new ModelAndView("add_sprint");
			view.addObject("edituser", new User());
			view.addObject("edituser2", new User());
			List<UserStory> userStories = projectDAO.getOngoingUserStory((Integer) session.getAttribute(SessionUtility.SESSION_PROJECT));
			view.addObject("userStories", userStories);
			view.addObject("sprintID", sprint.getSprintID());
			view.addObject("sprintNumber", newSprint.getSprintNumber());
			UserSprintForm storyForm = new UserSprintForm();
			ArrayList<SprintStory> sprintStories = new ArrayList<SprintStory>();
			for (int i = 0; i < userStories.size(); i++){
				SprintStory sprintStory = new SprintStory();
				sprintStories.add(sprintStory);
			}
			storyForm.setSprintStories(sprintStories);
			view.addObject("sprintForm", storyForm);
			int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(SessionUtility.SESSION_ID));
			view.addObject("notifs",notifs);
			view.addObject("project",projectDAO.getProjectByProjectID((Integer) session.getAttribute(SessionUtility.SESSION_PROJECT)));
			view.addObject("members", projectDAO.projectMembers((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT)));

			return view;

		}

		@RequestMapping("/insertSprintStories")
		public String insertSprintStories(HttpSession session, @ModelAttribute("sprintForm")UserSprintForm sprintForm, BindingResult result, RedirectAttributes map){
			projectDAO.insertSprintStories(sprintForm.getSprintStories());
			map.addFlashAttribute("addsprintstory", "New Sprint Added!");
			return "redirect:viewAllSprint";
		}

		@RequestMapping("/viewActivity")
		public ModelAndView viewActivity(HttpSession session, @RequestParam("tid")int taskID){
			ModelAndView view = new ModelAndView("activity");
			Task task = projectDAO.getTaskViaTaskID(taskID);
			view.addObject("edituser", new User());
			view.addObject("edituser2", new User());
			task.setVolunteer(userDAO.getUser2(task.getAccountID()).getFirstName());
			task.setStoryName(projectDAO.getUserStoryByID(task.getStoryID()).getStoryName());
			view.addObject("sprintstatus",projectDAO.getSprintBySprintID(task.getSprintID()).getStatus());
			view.addObject("task", task);
			view.addObject("datetoday", DateHandler.getCurrentDate());
			String datestarted = projectDAO.getSprintBySprintID(task.getSprintID()).getDateStarted().substring(0, 10);
			view.addObject("datestarted", datestarted);
			if(!projectDAO.isActivityExisting(taskID)){
				view.addObject("noactivity", "yes");
			}
			else{
				List<Activity> activities = projectDAO.getAllActivities(taskID);
				for(Activity act : activities){
					act.setTaskStatus(task.getStatus());
					act.setDoer(userDAO.getUser2(act.getAccountID()).getFirstName());
				}
				view.addObject("activities", activities);
			}
			view.addObject("newActivity", new Activity());
			int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(SessionUtility.SESSION_ID));
			view.addObject("notifs",notifs);
			view.addObject("project",projectDAO.getProjectByProjectID((Integer) session.getAttribute(SessionUtility.SESSION_PROJECT)));
			view.addObject("members", projectDAO.projectMembers((Integer)session.
				getAttribute(SessionUtility.SESSION_PROJECT)));

			return view;

		}

		@RequestMapping("/addActivity")
		public String addActivity(@ModelAttribute("newActivity")Activity activity,
			BindingResult result, HttpSession session){
			projectDAO.insertActivity(activity);
			projectDAO.updateTaskStatus(activity.getTaskStatus(), activity.getTaskID
				());
			return "redirect:viewActivity?tid="+activity.getTaskID();
		}

		@RequestMapping("/viewNotification")
		public ModelAndView viewNotification(HttpSession session){
			ModelAndView view = new ModelAndView("notification");
			List<Notification> notification = notifDAO.getNotification((Integer)
				session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)
				session.getAttribute(SessionUtility.SESSION_ID));
			for(Notification not : notification){
				if(not.getType().equals(NotificationUtility.NEW_MEMBER))
					not.setNewMember(userDAO.getUser2(not.getAccountID()).
						getFirstName());
			}
			view.addObject("edituser", new User());
			view.addObject("edituser2", new User());
			view.addObject("notification", "active");
			view.addObject("notifications", notification);
			int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
				SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
				SessionUtility.SESSION_ID));
			view.addObject("notifs",notifs);
			view.addObject("project",projectDAO.getProjectByProjectID((Integer)
				session.getAttribute(SessionUtility.SESSION_PROJECT)));
			view.addObject("members", projectDAO.projectMembers((Integer)session.
				getAttribute(SessionUtility.SESSION_PROJECT)));

			return view;
		}

		@RequestMapping("/notifSprintReview")
		public void notifSprintReview(){
			List<Sprint> openSprint = projectDAO.getAllOpenSprint();
			for(Sprint sprint : openSprint){
				if(DateHandler.getDaysDifference(DateHandler.getCurrentDate(),
					sprint.getSprintReview()) <= 1){
					Notification notif = new Notification();
					notif.setProjectID(sprint.getProjectID());
					notif.setType(NotificationUtility.SPRINT_REVIEW);
					notif.setSprintID(sprint.getSprintID());
					for(Member member : projectDAO.projectMembers(sprint.
						getProjectID())){
					if(!notifDAO.isSprintReview(sprint.getSprintID()))
						notifDAO.insertNotification(notif, member.
							getAccountID());
					}
				}
			}
		}



}
```

3. **Utilities**

Listing 38: DateHandler.java

```
/**
 * DateHandler.java
 */
package com.scrumprimer.utility;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;


public class DateHandler
{
	/**
	 * Date today in Calendar format
	 */
	private static Calendar TODAY = Calendar.getInstance();


	/**
	 * Format for parsing date Strings
	 */
	private final static SimpleDateFormat DATE_FORMAT = new SimpleDateFormat( "yyyy-MM-
		dd" );
```

```


    /**
     * Subtracts earlier date from today
     *
     * @param earlierDateStr Previous date in String format
     * @return Working days passed from earlierDateStr until today
     */

    public static String getCurrentDate()
    {
        String curDate = DATE_FORMAT.format( TODAY.getTime() );
        return curDate;
    }

    public static int getDaysDifference( String earlierDateStr )
    {

        int daysDifference = 0;

        if( !earlierDateStr.isEmpty() )
        {
            Calendar earlierDate = Calendar.getInstance();
            try {
                          earlierDate.setTime( DATE_FORMAT.parse( earlierDateStr )
                              );
                      } catch (ParseException e) {

                      }

            //-1 to take account the excess milliseconds passed from today, check in
                calculateWorkingDaysDifference() while loop
            daysDifference = calculateWorkingDaysDifference( earlierDate, TODAY ) - 1;
        }
        if(daysDifference < 0)
        {
            daysDifference = 0;
        }

        return daysDifference;

    }

    /**
     * @param d1 string of date to be compared
     * @param d2 string of date to be compared
     * @return true of String d1 is earlier that d2
     */
    public static boolean isEarlier(String d1, String d2)
    {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date date1 = null;
                try {
                        date1 = sdf.parse(d1);
                } catch (ParseException e) {

                }
                Date date2 = null;
                try {
                        date2 = sdf.parse(d2);
                } catch (ParseException e) {

                }

        if( date1.compareTo( date2 ) >= 0 )
                {
                        return false;
                }
                else return true;
    }

    public static String getDayAdded(String dateStr, int num){
        if(!dateStr.isEmpty()){
                Calendar date = Calendar.getInstance();
                try {
                                date.setTime(DATE_FORMAT.parse(dateStr));

                        } catch (ParseException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                date.add(Calendar.DAY_OF_MONTH, num);
                return DATE_FORMAT.format(date.getTime());
        }
        else{
                return null;
        }
    }


    /**
     * overloaded, subtracts earlier date from later date
     *
     * @param earlierDateStr Previous date in String format
     * @param laterDateStr Later date in String format
     * @return Working days passed from earlierDateStr until laterDateStr
     */

    public static int getDaysDifference( String earlierDateStr, String laterDateStr )
    {
        int daysDifference = 0;

        if( !earlierDateStr.isEmpty() && !laterDateStr.isEmpty() )
        {
            Calendar earlierDate = Calendar.getInstance();
            Calendar laterDate = Calendar.getInstance();
            try {
                                earlierDate.setTime( DATE_FORMAT.parse( earlierDateStr )
                                    );
                        laterDate.setTime( DATE_FORMAT.parse( laterDateStr ) );
                    } catch (ParseException e) {

                    }



            daysDifference = calculateWorkingDaysDifference( earlierDate, laterDate );
        }

        return daysDifference;
    }

    /**
     * @param earlierDate Previous date in String format
     * @param laterDate Later date in String format
     * @return Total working days between dates
     */
    private static int calculateWorkingDaysDifference( Calendar earlierDate, Calendar
         laterDate )
    {

        int workDays = 0;


        SimpleDateFormat date = new SimpleDateFormat( "MM-dd" );

        while( earlierDate.getTimeInMillis() < laterDate.getTimeInMillis() )
        {

                    //System.out.println( DATE_FORMAT.format( earlierDate.getTime() ) );
                    workDays++;


            earlierDate.add( Calendar.DAY_OF_MONTH, 1 );
        }

        return workDays;
    }




}
```

Listing 39: EmailNotification.java

```
/**
 * EmailNotification.java
 */
package com.scrumprimer.utility;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

import com.scrumprimer.controller.ProjectController;
import com.scrumprimer.model.User;

public class EmailNotification {

        private  static Properties properties = null;

        static {

        try
        {
                  properties = new Properties();
                  InputStream  inputStream  = ProjectController.class.getClassLoader().
                       getResourceAsStream( "mail.properties" );
                  properties.load(inputStream);
        }
        catch( Exception e )
```

```
            {
              e.printStackTrace();
            }
          }

            public static boolean emailInvite(ArrayList<User> team_members){
                    try{
                            String subject = "Scrum Primer Project Invitation";
                            Session session1 = Session.getInstance( properties, new
                                SocialAuth() );
                    Message msg = new MimeMessage( session1 );
                    InternetAddress from = new InternetAddress( ( String ) properties.get( "
                        mail.login.username" ),
                                ( String ) properties.get( "mail.login.fromname" ) );
                    msg.setFrom( from );
                    InternetAddress[] toAddresses = new InternetAddress[team_members.size()
                        ];
                    int i = 0;
                    for(User user2: team_members){
                            toAddresses[i] = new InternetAddress(user2.getEmail());
                            i++;
                    }
                    msg.setRecipients( Message.RecipientType.TO, toAddresses );
                    msg.setSubject( subject );

                    MimeBodyPart htmlPart = new MimeBodyPart();
                    String body = "<center>Register or login <a href =" +"http://localhost
                        :8080/ScrumPrimer/"+">here to be part of the team </a>";
                    htmlPart.setContent(body, "text/html");
                    Multipart multipart = new MimeMultipart();
                    multipart.addBodyPart(htmlPart);
                    msg.setContent(multipart);
                    Transport.send(msg);
                    return true;
                    } catch(Exception e){
                            return false;
                    }

            }

        static class SocialAuth extends Authenticator
        {

            /**
             * @return
             * @see javax.mail.Authenticator#getPasswordAuthentication()
             */
            @SuppressWarnings( "javadoc" )
            @Override
            protected PasswordAuthentication getPasswordAuthentication()
            {

                return new PasswordAuthentication( ( String ) properties.get( "mail.login.
                    username" ),
                    ( String ) properties.get( "mail.login.password" ) );

            }

        }

}
```

Listing 40: NotificationUtility.java

```
/**
 * NotificationUtility.java
 */
package com.scrumprimer.utility;

public class NotificationUtility {
        public static final String NEW_MEMBER = "new_member";
        public static final String SPRINT_REVIEW = "sprint_review";
        public static final String SCRUM_MEETING = "scrum_meeting";

}
```

Listing 41: RoleUtility.java

```
/**
 * RoleUtility.java
 */
package com.scrumprimer.utility;

public class RoleUtility {
        public static final String SM = "scrum_master";
        public static final String TM = "team_member";
        public static final String PO = "product_owner";

}
```

Listing 42: SessionUtility.java

```
/**
 * SessionUtility.java
 */
package com.scrumprimer.utility;

public class SessionUtility {

    public static final String SESSION_FIRSTNAME = "sessionName";

    public static final String SESSION_LASTNAME = "sessionLast";

    public static final String SESSION_ID = "accountID";

    public static final String SESSION_ROLE = "role";

    public static final String SESSION_PROJECT = "projectID";

    public static final String SESSION_PROJECT_NAME = "projectName";

    public static final String USERNAME = "username";

    public static final String PASSWORD = "password";

    public static final String SESSION_EMAIL = "sessionEmail";

    public static final String SPRINT = "sprintNumber";

    public static final String SESSION_PROJECTSTATUS = "status";

    public static final String SESSION_USERTYPE = "userType";


}
```

Listing 43: StatusUtility.java

```
/**
 * StatusUtility.java
 */
package com.scrumprimer.utility;

public class StatusUtility {
        public static final String ONGOING = "ongoing";
        public static final String CLOSED = "closed";
        public static final String NOT_STARTED = "not yet started";
        public static final String DONE = "done";
        public static final String SPRINT_CLOSE = "closed";
        public static final String SPRINT_OPEN = "open";
}
```

4. **Views**

```
<!-- import.jsp -->
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<title>SPP</title>
<link href="<c:url value="/resources/css/bootstrap.css" />" rel="stylesheet" media="
     screen">
<link href="<c:url value="/resources/css/additional.css" />" rel="stylesheet" media="
     screen">


<script type="text/javascript" src="<c:url value="/resources/js/jquery.js" />"></script>
<script type="text/javascript" src="<c:url value="/resources/js/jquery-1.91.js" />"></
     script>
<script src="<c:url value="/resources/js/jquery-ui.js" />"></script>
<script type="text/javascript" src="<c:url value="/resources/js/bootstrap.js" />"></
     script>
<script type="text/javascript" src="<c:url value="/resources/js/highcharts.js" />"></
     script>
```

```
<!-- navbar_regular.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
     pageEncoding="ISO-8859-1"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
     html4/loose.dtd">
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<!-- navbar -->
                <div class = "navbar navbar-default" role = "navigation">
                        <div class = "navbar-header">
                                <center><label class = "navbar-brand"> Scrum Primer </
                                     label></center>
                                <button class="navbar-toggle" data-target="#navbar-
                                     collapse" data-toggle="collapse" type="button">
                                        <span class="sr-only">
                                                Toggle navigation

                                        </span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>

                                </button>



                        </div>
</div>
```

```
<!-- navbar_scrum.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<!-- navbar -->
        <div class = "navbar navbar-default" role = "navigation">
            <div class = "navbar-header">
                <center><label class = "navbar-brand"> Scrum Primer </label></center>
                <button class="navbar-toggle" data-target="#navbar-collapse" data-toggle="collapse" type="button">
                    <span class="sr-only">
                        Toggle navigation

                    </span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>

                </button>



            </div>

            <div class="navbar-collapse collapse" id = "navbar-collapse">
            <ul class="nav navbar-nav">

                    <li class = ${sprint }>
                                <a href = "${contextPath }/viewAllSprint" data-toggle = "tooltip" data-placement = "bottom" title = "Sprint - the incremental durations" >Sprint </a>
                    </li>

                    <li class = "${productbacklog }">
                        <a href = "${contextPath }/viewProductBacklog" data-toggle = "tooltip" data-placement = "bottom" title = "Product Backlog - contains the list of overall requirements written as user stories by the Product Owner">Product Backlog</a>
                    </li>
                    <li class = "${notification }">
                        <a href = "${contextPath }/viewNotification">Notification <span class = "badge">${notifs }</span></a>
                    </li>

            </ul>

            <ul class = "nav navbar-nav navbar-right" style = "margin-right:5px;">
                    <li class="dropdown dropdown-inverse"><a class="dropdown-toggle" data-toggle="dropdown" href="#"><u><c:out value = "${sessionScope.projectName } (${sessionScope.role })"/></u><b class="caret" data-toggle = "tooltip" data-placement = "bottom" title = "-Project Profile, -Back to Projects"></b></a>
                            <ul class = "dropdown-menu">
                            <li><a href = "#projectprofilemodal" data-toggle = "modal">Project Profile</a></li>
                            <li class = "divider"></li>
                            <li><a href = "${contextPath }/Project">All Projects</a></li>
                            </ul>


                        </li>
                        <li class="dropdown dropdown-inverse"><a class="dropdown-toggle" data-toggle="dropdown" href="#"><span class = "glyphicon glyphicon-user"></span>&nbsp;<c:out value = "${sessionScope.username }"/><b class="caret"></b></a>
                            <ul class="dropdown-menu">
                                <li><a href="#profilemodal" data-toggle = "modal">Edit profile</a></li>
                                <li><a href = "#passwordmodal" data-toggle = "modal">
```

```
                                        Change Password</a>
                                        <li class = "divider"></li>
                                        <li><a href="${contextPath }/Logout">Logout</a></li>
                                </ul>

                        </ul>
                </div>
        </div>

        <!-- project profile modal -->
        <div class="modal fade" id = "projectprofilemodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                        <h4 class="modal-title">Project Profile</h4>
                </div>
                <div class="modal-body">
                        <table align = "center" cellpadding = "3px" width = "80%">
                        <tr>
                                <th>Project Name
                                <td>${project.projectName}

                        </tr>
                        <tr>
                                <th>Project Description
                                <td>${project.projectDescription}

                        </tr>
                        <tr>
                                <th>Status
                                <td>${project.status }
                        </tr>
                        </table>
                        <br>
                        <br>
                        <table align = "center" width = "80%" cellpadding = "3px">
                        <tr>
                                <th>Member
                                <th> Role
                        </tr>
                        <c:forEach items = "${members }" var = "member">
                        <tr>
                                <td>${member.name }
                                <td>${member.role }
                        </tr>
                        </c:forEach>

                        </table>
                </div>
                </div>
                </div>
       </div>
        <!-- edit profile modal -->
                <div class="modal fade" id = "profilemodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                        <h4 class="modal-title">Edit Profile</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="editUser" modelAttribute = "edituser" onSubmit = "alertChange();" >
                            <table align = "center" cellpadding = "3px" width = "80%">
                                <tr>
                                        <td align = "center"><form:label path = "firstName" class="col-sm-2 control-label">First Name</form:label>
                                        <td align = "center"><form:input name = "firstname" path="firstName" value = "${sessionScope.sessionName }" class = "form-control" id = "firstname" required = "required" />
                                </tr>
                                <tr>
                                         <td align = "center"><form:label path = "lastName" class="col-sm-2 control-label">Last Name</form:label>
                                        <td align = "center"><form:input name = "lastname" path="lastName" value = "${sessionScope.sessionLast }" class = "form-control" id = "lastname" required = "required" />
                                </tr>


                                <tr>
                                         <td align = "center"><form:label path = "username" class="col-sm-2 control-label">Username</form:label>
                                        <td align = "center"><form:input path="
                                             username" value = "${sessionScope.
                                             username }" class = "form-control"
                                             id = "username" required = "
                                             required"/>
                                </tr>
                                <tr>
                                        <td align = "center"><form:label path =
                                             "email" class="col-sm-2 control-
                                             label">Email Address</form:label>

                                        <td align = "center"><form:input type =
                                             "email" path="email" value = "${
                                             sessionScope.sessionEmail }" class
                                             = "form-control" id = "email"
                                             required = "required" />

                                </tr>

                                <tr>

                                <td align = "center"><button type="submit" class
                                     ="btn btn-success">Save Changes</button>

                        </tr>
                </table>
                </form:form>


                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
                </div><!-- /.modal -->

                <!-- change password modal -->
                <div class="modal fade" id = "passwordmodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                             hidden="true">&times;</button>
                        <h4 class="modal-title">Change Password</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="editPassword" modelAttribute =
                             "edituser2" onsubmit="return validate('${sessionScope.
                             password }');">
                                <table align = "center" cellpadding = "3px" width =
                                     "80%">
                                <tr>
                                        <td align = "center"><form:label path =
                                             "password" class="control-label">
                                             Current Password</form:label>
                                        <td align = "center"><form:input name =
                                             "password" path="password" class =
                                             "form-control" required = "required
                                             " type ="password" id = "pass" />
                                </tr>
                                <tr>
                                         <td align = "center"><form:label path =
                                              "newPassword" class="control-
                                              label">New Password</form:label>
                                        <td align = "center"><form:input name =
                                             "newPassword" path="newPassword"
                                             class = "form-control" id = "first"
                                              required = "required" type = "
                                             password" />
                                </tr>

                                <tr>
                                         <td align = "center"><form:label path =
                                              "newPassword2" class="control-
                                              label">Type new password again</
                                              form:label>
                                        <td align = "center"><form:input path="
                                             newPassword2" class = "form-control
                                             " id = "second" required = "
                                             required" type = "password"/>
                                </tr>
                                <tr>

                                <td align = "center"><button type="submit" class
                                     ="btn btn-success">Save Changes</button>

                        </tr>
                </table>
                </form:form>





                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
                </div><!-- /.modal -->
<script>

function alertChange(){
        alert("Changes has been saved");

}
function validate(password){
var firstInput = document.getElementById("first").value;
var secondInput = document.getElementById("second").value;
var input = document.getElementById("pass").value;


if(firstInput != secondInput)
{
        alert("New passwords did not match");
        return false;

}
if(input != password){
        alert("Invalid password");
        return false;
}

alert("Password has been changed");
return true;

}

</script>
```

```
<!-- navbar_afterlogin.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<c:set var = "currentURL" value = "${pageContext.request.requestURI }"/>
<c:set var = "lol" value = "lol"/>
<!-- navbar -->
        <div class = "navbar navbar-default" role = "navigation">
            <div class = "navbar-header">
                <label class = "navbar-brand"> Scrum Primer </label>
                <button class="navbar-toggle" data-target="#navbar-collapse" data-toggle="collapse" type="button">
                    <span class="sr-only">
                        Toggle navigation

                    </span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>

                </button>



            </div>
            <div class="navbar-collapse collapse" id = "navbar-collapse">
            <ul class="nav navbar-nav">
                <li class="${activeproject }">
                    <a href = "${contextPath}/Project" data-toggle = "tooltip" data-placement = "bottom" title = "list of all projects"> Project </a>
                </li>
                <c:if test = "${sessionScope.userType eq 'admin' }" >
                                <li class = ${activeaccounts }><a href = "${contextPath }/Accounts">Accounts</a> </li>

                    </c:if>
            </ul>

            <ul class = "nav navbar-nav navbar-right" style = "margin-right:5px;">
                        <li class="dropdown dropdown-inverse"><a class="dropdown-toggle" data-toggle="dropdown" href="#"><span class = "glyphicon glyphicon-user"></span>&nbsp;<c:out value = "${sessionScope.username }"/><b class="caret"></b></a>
                            <ul class="dropdown-menu">
                                <li><a href="#profilemodal" data-toggle = "modal">Edit profile</a></li>
                                <li><a href = "#passwordmodal" data-toggle = "modal">Change Password</a>
                                <li class = "divider"></li>
                                <li><a href="${contextPath }/Logout">Logout</a></li>
```

```
                                                    </ul>
                                  </ul>

                        </div>
                </div>


                <!-- edit profile modal -->
                <div class="modal fade" id = "profilemodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                             hidden="true">&times;</button>
                        <h4 class="modal-title">Edit Profile</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="editUser" modelAttribute = "
                             edituser" onSubmit = "alertChange();" >
                                <table align = "center" cellpadding = "3px" width =
                                     "80%">
                                        <tr>
                                                <td align = "center"><form:label path =
                                                     "firstName" class="col-sm-2 control
                                                     -label">First Name</form:label>
                                                <td align = "center"><form:input name =
                                                     "firstname" path="firstName" value
                                                     = "${sessionScope.sessionName }"
                                                     class = "form-control" id = "
                                                     firstname" required = "required" />
                                        </tr>
                                        <tr>
                                                 <td align = "center"><form:label path =
                                                      "lastName" class="col-sm-2
                                                      control-label">Last Name</form:
                                                      label>
                                                <td align = "center"><form:input name =
                                                     "lastname" path="lastName" value =
                                                     "${sessionScope.sessionLast }"
                                                     class = "form-control" id = "
                                                     lastname" required = "required" />
                                        </tr>

                                        <tr>
                                                 <td align = "center"><form:label path =
                                                      "username" class="col-sm-2
                                                      control-label">Username</form:
                                                      label>
                                                <td align = "center"><form:input path="
                                                     username" value = "${sessionScope.
                                                     username }" class = "form-control"
                                                     id = "username" required = "
                                                     required"/>
                                        </tr>
                                        <tr>
                                                <td align = "center"><form:label path =
                                                     "email" class="col-sm-2 control-
                                                     label">Email Address</form:label>

                                                <td align = "center"><form:input type =
                                                     "email" path="email" value = "${
                                                     sessionScope.sessionEmail }" class
                                                     = "form-control" id = "email"
                                                     required = "required" />

                                        </tr>

                                        <tr>

                                        <td align = "center"><button type="submit" class
                                             ="btn btn-success">Save Changes</button>

                                </tr>
                </table>
                </form:form>


                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
                </div><!-- /.modal -->

                <!-- change password modal -->
                <div class="modal fade" id = "passwordmodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                             hidden="true">&times;</button>
                        <h4 class="modal-title">Change Password</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="editPassword" modelAttribute =
                             "edituser2" onsubmit="return validate('${sessionScope.
                             password }');">
                                <table align = "center" cellpadding = "3px" width =
                                     "80%">
                                        <tr>
                                                <td align = "center"><form:label path =
                                                        "password" class="control-label">
                                                        Current Password</form:label>
                                                    <td align = "center"><form:input name =
                                                        "password" path="password" class =
                                                        "form-control" required = "required
                                                        " type ="password" id = "pass" />
                                            </tr>
                                            <tr>
                                                     <td align = "center"><form:label path =
                                                           "newPassword" class="control-
                                                          label">New Password</form:label>
                                                    <td align = "center"><form:input name =
                                                        "newPassword" path="newPassword"
                                                        class = "form-control" id = "first"
                                                         required = "required" type = "
                                                        password" />
                                            </tr>

                                            <tr>
                                                     <td align = "center"><form:label path =
                                                           "newPassword2" class="control-
                                                          label">Type new password again</
                                                          form:label>
                                                    <td align = "center"><form:input path="
                                                        newPassword2" class = "form-control
                                                        " id = "second" required = "
                                                        required" type = "password"/>
                                            </tr>
                                            <tr>

                                            <td align = "center"><button type="submit" class
                                                 ="btn btn-success">Save Changes</button>

                                    </tr>
                        </table>
                        </form:form>





                        </div> <!-- modal-body -->
                        </div><!-- /.modal-content -->
                                </div><!-- /.modal-dialog -->
                        </div><!-- /.modal -->
<script>

function alertChange(){
        alert("Changes has been saved");

}
function validate(password){
var firstInput = document.getElementById("first").value;
var secondInput = document.getElementById("second").value;
var input = document.getElementById("pass").value;


if(firstInput != secondInput)
{
        alert("New passwords did not match");
        return false;

}
if(input != password){
        alert("Invalid password");
        return false;
}

alert("Password has been changed");
return true;

}

</script>
```

```
<!-- accounts.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
<jsp:include page = "navbar_afterlogin.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<c:if test="${update eq 'yes'}">
                <script type="text/javascript">
                           $(window).load(function(){
                          $('#UserEditorModal').modal('show');
                           });
                </script>
        </c:if>
    <c:if test="${successfuldelete != null }">
        <div class="alert alert-success alertMessage">
            <c:out value="${successfuldelete }"/>
            <button type="button" class="close" data-dismiss="alert">x</button>
        </div>
</c:if>
    <c:if test="${updateaccount != null }">
        <div class="alert alert-success alertMessage">
            <c:out value="${updateaccount }"/>
            <button type="button" class="close" data-dismiss="alert">x</button>
        </div>
</c:if>
<div class = "container-project" style = "background-color: transparent;">
    <a href="#AddUserModal" data-toggle="modal"><button class = "btn btn-info">&nbsp;Add User</button></a>
            <br>
            <table class="table" cellpadding = "5px" width = "930px" >
                    <thead>
                        <tr bgcolor = "lightsteelblue" style = "display: block; width: 930px;">
                            <th style = "width: 230px;">&nbsp;</th>
                            <th style = "width: 230px;">Name</th>
                            <th style = "width: 230px;">Username</th>
                            <th style = "width: 230px;">Email</th>

                        </tr>
                    </thead>
                    <tbody style = "display:block; width:930px;">
                        <c:forEach items = "${users }" var = "user">
                        <tr>
                            <td style = "width: 230px;"><a href = "${contextPath }/deleteAccount?aid=${user.userID}" onclick = "return confirm('Are you sure you want to delete?');"><button class = "btn btn-danger"><span class = "glyphicon glyphicon-trash"></span></button></a>
                            &nbsp;<a href = "${contextPath }/editAccount?aid=${user.userID}"><button class = "btn btn-info">Edit</button></a>
                            <td style="width: 230px;"><c:out value = "${user.firstName }"/> <c:out value = "${user.lastName }"/>
                            <td style = "width: 230px;"><c:out value = "${user.username }"/>
                            <td style = "width: 230px;"><c:out value = "${user.email }"/>
                        </tr>
                        </c:forEach>
                    </tbody>
            </table>
</div>

        <!-- edit profile modal -->
        <div class="modal fade" id = "UserEditorModal" tabindex = "-1" >
            <div class="modal-dialog" style = "max-width: 70%;">
        <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
            <h4 class="modal-title">Edit User</h4>
        </div>
        <div class="modal-body">
            <form:form method="post" action="updateAccount" modelAttribute = "edit">
                <table align = "center" cellpadding = "3px" width = "80%">
                    <tr>
                        <td align = "center"><form:label path = "firstName" class="col-sm-2 control-label">First Name</form:label>
                        <td align = "center"><form:input name = "firstname" path="firstName" value = "${updateuser.firstName }" class = "form-control" id = "firstname" required = "required" />
                    </tr>
                    <tr>
                         <td align = "center"><form:label path = "lastName" class="col-sm-2 control-label">Last Name</form:label>
                        <td align = "center"><form:input name = "lastname" path="lastName" value =
                                            "${updateuser.lastName }" class = "
                                            form-control" id = "lastname"
                                            required = "required" />
                                </tr>

                                <tr>
                                        <td align = "center"><form:label path =
                                                "username" class="col-sm-2
                                                control-label">Username</form:
                                                label>
                                        <td align = "center"><form:input path="
                                            username" value = "${updateuser.
                                            username }" class = "form-control"
                                            id = "username" required = "
                                            required"/>
                                </tr>
                                <tr>
                                        <td align = "center"><form:label path =
                                            "email" class="col-sm-2 control-
                                            label">Email Address</form:label>

                                        <td align = "center"><form:input type =
                                            "email" path="email" value = "${
                                            updateuser.email }" class = "form-
                                            control" id = "email" required = "
                                            required" />
                                         <form:input type="hidden" path = "
                                                userID" value = "${updateuser.
                                                userID }"/>
                                </tr>

                                <tr>

                                <td align = "center"><button type="submit" class
                                        ="btn btn-success">Save Changes</button>

                        </tr>
                </table>
                </form:form>


                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
                </div><!-- /.modal -->

                        <!-- edit profile modal -->
                <div class="modal fade" id = "AddUserModal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                                hidden="true">&times;</button>
                        <h4 class="modal-title">Edit User</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="addAccount" modelAttribute = "
                                newUser">
                                <table align = "center" cellpadding = "3px" width =
                                        "80%">
                                <tr>
                                        <td align = "center"><form:label path =
                                            "firstName" class="col-sm-2 control
                                            -label">First Name</form:label>
                                        <td align = "center"><form:input name =
                                            "firstname" path="firstName"  class
                                             = "form-control" id = "firstname"
                                            required = "required" />
                                </tr>
                                <tr>
                                         <td align = "center"><form:label path =
                                                "lastName" class="col-sm-2
                                                control-label">Last Name</form:
                                                label>
                                        <td align = "center"><form:input name =
                                            "lastname" path="lastName"  class =
                                             "form-control" id = "lastname"
                                            required = "required" />
                                </tr>

                                <tr>
                                         <td align = "center"><form:label path =
                                                "username" class="col-sm-2
                                                control-label">Username</form:
                                                label>
                                        <td align = "center"><form:input path="
                                            username"  class = "form-control"
                                            id = "username" required = "
                                            required"/>
                                </tr>
                                <tr>
                                        <td align = "center"><form:label path =
                                            "password" class="col-sm-2 control-
                                            label">Password</form:label>

                                        <td align = "center"><form:input type =
                                            "password" path="password"  class =
                                             "form-control" id = "password"
                                            required = "required" />
```

```

                    </tr>
                    <tr>
                        <td align = "center"><form:label path = "email" class="col-sm-2 control-label">Email Address</form:label>

                        <td align = "center"><form:input type = "email" path="email"  class = "form-control" id = "email" required = "required" />

                    </tr>
                    <tr>
                        <td align = "center"><form:label path = "userType" class="col-sm-2 control-label">Admin?</form:label>
                        <td align = "center"><form:checkbox path = "userType" value = "admin"/>
                    </tr>

                    <tr>

                    <td align = "center"><button type="submit" class="btn btn-success">Save Changes</button>

                </tr>
        </table>
        </form:form>


        </div> <!-- modal-body -->
        </div><!-- /.modal-content -->
            </div><!-- /.modal-dialog -->
        </div><!-- /.modal -->

</body>
</html>
```

```
<!-- activity.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
    <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
    <jsp:include page = "navbar_scrum.jsp" />

    <div class = "container-project">
        <div class = "container-taskdetail">
        <ul class="breadcrumb">
        <li>
    <a href="${contextPath }/viewSprintBacklog?sid=${task.sprintID}">Sprint Backlog</a> <span class="divider">></span>
        </li>
        <li class="active">${task.taskName }</li>
        </ul>

        <table width = "90%">
            <tr>
                <th>User Story
                <th>${task.storyName }
                <th>Task Name
                <td><i>${task.taskName }</i>
            </tr>
            <tr>
                <th>Volunteer
                <td>${task.volunteer }
                <th>Initial Estimate
                <td>${task.estimate }
            </tr>
        </table>
        </div>
        <div class = "container-activity">
            <table class = "table table-striped" cellpadding = "5px" height = "200px" width = "930px">
            <thead>
                <tr bgcolor = "white" style = "display: block; width: 920px;">
                            <th style = "width: 186px;">Detail</th>
                            <th style = "width: 186px;">Actual Number of Hours</th>
                            <th style = "width: 186px;">Date</th>
                            <th style = "width: 186px;">Task Status</th>
                            <th style = "width: 186px;">Done by: </th>
                </tr>
            </thead>

            <tbody style = "display:block; overflow-y:auto; max-height: 200px; width:920px;">
                <c:forEach items = "${activities}" var = "act">
                <tr>
                    <td style = "width: 186px;"><c:out value = "${act.detail }" />
                    <td style = "width: 186px;"><c:out value = "${act.time }" />
                    <td style = "width: 186px;"><c:out value = "${act.date }" />
                    <td style = "width: 186px;"><c:out value = "${act.taskStatus }" />
                    <td style = "width: 186px;"><c:out value = "${act.doer }" />
                </tr>
                </c:forEach>

            </tbody>

        </table>
        <c:if test = "${sessionScope.role eq 'team_member' || sessionScope.role eq 'scrum_master' && sprintstatus == 'open'}">
        <c:if test = "${task.status ne 'done' }">
        <center><a href = "#newactivitymodal" data-toggle = "modal"><button class = "btn btn-primary">Add Activity</button></a></center>
        </c:if>
        </c:if>

        </div>
    </div>

    <!-- newactivitymodal -->
    <div class="modal fade" id = "newactivitymodal" tabindex = "-1" >
            <div class="modal-dialog" style = "max-width: 70%;">
        <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
            <h4 class="modal-title"><c:out value = "New Activity for ${task.taskName}"></c:out></h4>
        </div>
        <div class="modal-body">
        <form:form action = "addActivity" modelAttribute = "newActivity" >
        <table align = "center" cellpadding = "5px">
            <tr>
                <td><form:label path = "detail" class = "control-label">Detail</form:label>
                    <td><form:textarea class = "form-control" path = "detail" required = "required"></form:textarea>
                    <form:input type = "hidden" path = "taskID" value = "${task.taskID }"/>
                    <form:input type = "hidden" path = "accountID" value = "${sessionScope.accountID }"/>
            </tr>
            <tr>
                <td><form:label path = "time" class = "control-label">Actual (no. of hours)</form:label>
                <td><form:input type = "number" path = "time" class = "form-control" required = "required" min = "1"/>
            </tr>
            <tr>
                <td><form:label path = "time" class = "control-label">Date</form:label>
                <td><form:input type = "date" path = "date" class = "form-control" required = "required" min = "${datestarted }" max = "${datetoday }"/>
            </tr>
            <tr>
                <td><form:label path = "taskStatus" class = "control-label">Task Status</form:label>
                <td><form:select path = "taskStatus" class = "form-control" required = "required">
                    <option value = "ongoing">Ongoing
                    <option value = "done">Done
                    </form:select>
            </tr>
            <tr>


                    <td align = "center"><button type="submit" class="btn btn-success">Add Activity</button>

            </tr>

        </table>
        </form:form>
        </div> <!-- modal-body -->
        </div><!-- /.modal-content -->
            </div><!-- /.modal-dialog -->
    </div><!-- /.modal -->

</body>
</html>
```

```
<!-- add_product_backlog.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

 <!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">
<jsp:include page="import.jsp" />
</head>
<body>
<jsp:include page = "navbar_scrum.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<div class="alert alert-info alertMessage">
            User story is a project requirement that can be a functional requirement, non-functional requirement, or issue.
            <button type="button" class="close" data-dismiss="alert">x</button>
    </div>
<div class = "container-project">
    <div  class="table-scroll">
        <form:form action = "insertUserStories" method = "POST" modelAttribute = "newStory">
        <table class = "table" cellpadding = "5px" height = "330px" width = "930px">
            <thead>
                <tr bgcolor = "white" style = "display: block; width: 930px;">
                            <th style = "width: 310px;">User Story</th>
                            <th style = "width: 310px;">Details</th>
                            <th style = "width: 310px;">Status</th>
                </tr>
            </thead>

            <tbody style = "display:block; overflow-y:auto; max-height: 300px; width:930px;">
                <c:forEach items = "${ newStory.stories}" varStatus = "index" >
                <tr>
                    <td style = "width: 310px;"><input name = "stories[${index.index}].storyName" class = "form-control" required = "required" >
                    <td style = "width: 310px;"><textarea name = "stories[${index.index}].detail" class = "form-control" required = "required" ></textarea>
                    <td style = "width: 310px;"><select name = "stories[${index.index}].status" class = "form-control">
                            <option value = "not yet started">not yet started</option>
                        <option value = "ongoing">ongoing</option>
                        <option value = "done">done</option>
                        </select>
                        <input type = "hidden" name = "stories[${index.index }].projectID" value = "${sessionScope.projectID }" />
                </tr>
                </c:forEach>
            </tbody>

        </table>
        <center><button class = "btn btn-primary" type = "submit">Add</button></center>
        </form:form>
    </div>
</div>

</body>
</html>
```

```
<!-- add_project.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

 <!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">
<jsp:include page="import.jsp" />
</head>
<body>
    <jsp:include page = "navbar_afterlogin.jsp" />
<div class = "container-project">
<center><strong><h3>New Project</h3></strong></center>
<div style = "margin-top: 30px;">
<form:form action = "inviteMembers" modelAttribute = "newProject">
    <table align = "center" cellpadding = "3px" width = "600px">
    <tr>
```

```
			<td><form:label path = "projectName" class = "control-label">Project Name</form:label>
			<td><form:input path = "projectName" class = "form-control" required = "required"></form:input>
		</tr>
		<tr>
			<td><form:label path = "projectDescription" class = "control-label">Project Description</form:label>
			<td><form:textarea path = "projectDescription" class = "form-control" required = "required"></form:textarea>

		</tr>
		<tr>
			<td><form:label path = "numberOfTM" class = "control-label">Number of Team Members</form:label>
			<td><form:input type = "number" path = "numberOfTM" class = "form-control" required = "required" ></form:input>
				<form:input type = "hidden" path = "numberOfPO" value = "1" />
		</tr>
		<tr>
			<td align = "center"><button type="submit" class="btn btn-success">Submit</button>
		<td align = "center"><button type="reset" class="btn btn-default">Clear</button>
		</tr>


		</table>
</form:form>
</div>
<br>
<br>
<div class="alert alert-warning alertMessage">
				Note: The one who register the new project will automatically become the Scrum Master
				<button type="button" class="close" data-dismiss="alert">x</button>
</div>
</div>
</body>
</html>
```

```
<!-- add_sprint.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

 <!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">
<jsp:include page="import.jsp" />
</head>
<body>
		<jsp:include page = "navbar_scrum.jsp" />
<div class = "container-project">
<center><strong><h3>User Stories for Sprint ${sprintNumber } </h3></strong></center>
<div style = "margin-top: 30px; margin-left: auto; margin-right: auto; width: 400px;">

<form:form action = "insertSprintStories" modelAttribute = "sprintForm">
		<table class="table" cellpadding = "5px" height = "240px" width = "400px" align = "center">
		<thead style = "display:block; width: 400px; background-color: white">
			<tr>
			<td align = "center" width = "400px">Unfinished User Stories
			</tr>
		</thead>
		<tbody style = "display:block; overflow-y:auto; height: 240px; width:400px;">

		<c:forEach items = "${userStories }" var = "story" varStatus = "index">
			<tr>
			<td width = "400px"><input type = "checkbox" name = "sprintStories[${index.index }].storyID" value = "${story.storyID }" ><c:out value = "${story.storyName }"/>
			<input type = "hidden" name = "sprintStories[${index.index }].sprintID" value = "${sprintID }" />
			</tr>
		</c:forEach>


		</tbody>
		</table>
		<table width = "100%">
		<tr>
		<td align = "center"><button type="submit" class="btn btn-success">Submit</button>
	<td align = "center"><button type="reset" class="btn btn-default">Clear</button>
		</tr>


		</table>
</form:form>
</div>

</div>

</body>
</html>
```

```
<!-- edit_product_backlog.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
<jsp:include page = "navbar_scrum.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<div class = "container-project">
        <div  class="table-scroll">
                <form:form action = "saveProductBacklog" method = "POST" modelAttribute = "storyForm">
                <table class = "table" cellpadding = "5px" height = "330px" width = "930px">
                        <thead>
                                <tr bgcolor = "#ffffcc" style = "display: block; width: 930px;">
                                                                <th style = "width: 310px;">User Story</th>
                                                                <th style = "width: 310px;">Details</th>
                                                                <th style = "width: 310px;">Status</th>
                                </tr>
                        </thead>

                        <tbody style = "display:block; overflow-y:auto; max-height: 300px; width:930px;">
                                <c:forEach items = "${ storyForm.stories}" var = "story" varStatus = "index">
                                <tr>
                                        <td style = "width: 310px;"><input name = "stories[${index.index}].storyName" class = "form-control" required = "required" value = "${story.storyName }">
                                        <td style = "width: 310px;"><textarea name = "stories[${index.index}].detail" class = "form-control" required = "required" ><c:out value = "${story.detail }" /></textarea>
                                        <td style = "width: 310px;"><select name = "stories[${index.index}].status" class = "form-control">
                                                <option value = "${story.status }"><c:out value="${story.status}"/></option>
                                                <optgroup label = "----------">
                                                <option value = "not yet started">not yet started</option>
                                                <option value = "ongoing">ongoing</option>
                                                <option value = "done">done</option>
                                                </optgroup>
                                                </select>
                                        <input type = "hidden" name = "stories[${index.index }].storyID" value = "${story.storyID}" />
                                </tr>
                                </c:forEach>
                        </tbody>

                </table>
                <center><button class = "btn btn-primary" type = "submit">Save Changes</button></center>
                </form:form>
        </div>
</div>

</body>
</html>
```

```
<!-- home_regular.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
        <jsp:include page = "navbar_regular.jsp" />
        <c:if test="${successmessage != null }">
                <div class="alert alert-success alertMessage">
                        <c:out value="${successmessage }"/>
```

```
                      <button type="button" class="close" data-dismiss="alert">x</button>
              </div>
      </c:if>
      <c:if test="${errormessage != null }">
              <div class="alert alert-warning alertMessage">
                      <c:out value="${errormessage }"/>
                      <button type="button" class="close" data-dismiss="alert">x</button>
              </div>
      </c:if>
      <c:if test="${errormessage2 != null }">
              <div class="alert alert-warning alertMessage">
                      <c:out value="${errormessage2 }"/>
                      <button type="button" class="close" data-dismiss="alert">x</button>
              </div>
      </c:if>
      <c:if test="${errorlogin != null }">
              <div class="alert alert-warning alertMessage">
                      <c:out value="${errorlogin }"/>
                      <button type="button" class="close" data-dismiss="alert">x</button>
              </div>
      </c:if>

      <div class = "span1">
      <div class="tabbable">
                                <ul class="nav nav-tabs">
                                  <li class="active"><a href="#features" data-toggle="tab"><strong>Features</strong></strong></a></li>
                                  <li><a href="#about" data-toggle="tab"><strong>Scrum</strong></a></li>
                                </ul>


              <div class= "tab-content">
                      <div id="features" class="tab-pane active">
                      <br>
                              <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
                              <!-- Indicators -->
                              <ol class="carousel-indicators">
                      <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
                      <li data-target="#carousel-example-generic" data-slide-to="1"></li>
                      <li data-target="#carousel-example-generic" data-slide-to="2"></li>
                      <li data-target="#carousel-example-generic" data-slide-to="3"></li>
                              </ol>
                              <!-- Wrapper for slides -->
                              <div class="carousel-inner">
                      <div class="item active">
                      <img src = "<c:url value = "/resources/scrum/scrumboard.JPG"/>">
                      <div class="carousel-caption">
                      Scrum Board
                      </div>
                      </div>

                      <div class="item">
                      <img src = "<c:url value = "/resources/scrum/burndownchart.JPG"/>">
                      <div class="carousel-caption">
                      Burndown Chart
                      </div>
                              </div>

                              <div class="item">
                      <img src = "<c:url value = "/resources/scrum/sprintbacklog.JPG"/>">
                      <div class="carousel-caption">
                      Sprint Backlog
                      </div>
                              </div>

                              <div class="item">
                      <img src = "<c:url value = "/resources/scrum/productbacklog.JPG"/>">
                      <div class="carousel-caption">
                      Product Backlog
                      </div>
                              </div>

                              </div>

                              <!-- Controls -->
<a class="left carousel-control" href="#carousel-example-generic" data-slide="prev">
  <span class="glyphicon glyphicon-chevron-left"></span>
</a>
<a class="right carousel-control" href="#carousel-example-generic" data-slide="next">
  <span class="glyphicon glyphicon-chevron-right"></span>
</a>

</div>
                      </div> <!-- features -->
                      <div id = "about" class = "tab-pane">
```

```
            <p>Scrum mainly tackles the use of iterative development by implementing incremental durations called <i>Sprints</i>.
            Sprints are timeboxed which means that they cannot be extended even if the work required is done or not. At the beginning of
            each Sprint the Scrum Team is committed to complete these requirements, given and prioritized by the Product Owner, each sprint.
            At the end of each Sprint, there will be a Sprint Review together with the Product Owner and some stakeholders. The team will
            demonstrate what is built in the whole Sprint. The reviews in each Sprint Review will be incorporated to the next Sprint. For Scrum considered to be done,
            the product produced should be finally integrated, tested and potentially shippable or a definition of done given by the Product Owner should be available. </p>

            <h4>FAQ</h4>
            <ul>
                <li><u>What is a Scrum Team?</u><br> <br>
                    <p>Scrum team is composed of the <b>Product Owner</b> -- responsible in providing the project's requirements, <b>Development Team</b> -- cross functional and self organizing team that develops the product the Product Owner needs, and <b>Scrum Master</b> -- member of the development team that acts as the bridge between the team and Product Owner.</p>
                <li><u>What is the Product Backlog</u><br><br>
                    <p>Product backlog contains the list of overall requirements written as user stories provided by the Product Owner.
                <li><u>What is a User Story?</u><br><br>
                    <p>User story is a project requirement that can be a functional requirement, non-functional requirement, or issue.
                <li><u>What is the Sprint Backlog?</u><br><br>
                    <p>Sprint backlog contains the list of the tasks for the whole Sprint.</p>
                <li><u>What is the Burndown Chart?</u><br><br>
                    <p>It is the chart which shows the trend of work to the remaining time in a Sprint.</p>
                <li><u>What is a Scrum Board?</u><br><br>
                    <p>A visual task-tracking tool that contains the current tasks usually written in post-its and is grouped thru the table that labeled "Not yet started", "In progress" and "Completed".
            </ul>
        </div> <!-- about -->
    </div><!-- tab-content -->
</div> <!-- tabbable-->
</div>
<div class = "span2">
<div class = "container-login">
    <h4>Sign in</h4>
    <form:form method="post" action="Login" modelAttribute = "login">
        <table align = "center" width = "380px" cellpadding = "3px">
            <tr>
            <td align = "right"><form:label path="username" class="col-sm-2 control-label">Username</form:label>
            <td><form:input class="form-control" path= "username" required = "required" />

            </tr>
            <tr>
            <td align = "right"><form:label path = "password" class="col-sm-2 control-label">Password</form:label>

            <td><form:input type="password" class="form-control" path="password" required = "required" />

             </tr>

            </table>
            <table align = "center" width = "380px">
            <tr>

            <td align = "center"><button type="submit" class="btn btn-success">Sign in</button>

            </tr>
            </table>

            </form:form>

    </div> <!-- container-login -->
<div class = "container-register">
```

```
        New User? Sign up for new account.
        <div style = "padding-top: 10px;">
            <!-- registration form -->
            <form:form method="post" action="addUser" modelAttribute = "user" id = "registration-form">
                <table align = "center" width = "380px" height = "180px" cellpadding = "2px">
                    <tr>
                        <td align = "center"><form:input name = "firstname" path="firstName" placeholder = "First Name" class = "form-control" id = "firstname" required = "required" pattern = "[a-zA-Z0-9]+" />
                        <td align = "center"><form:input name = "lastname" path="lastName" placeholder = "Last Name" class = "form-control" id = "lastname" required = "required" pattern = "[a-zA-Z0-9]+" />
                    </tr>

                    <tr>
                        <td align = "center"><form:input type = "email" path="email" placeholder = "Email Address" class = "form-control" id = "email" required = "required" />

                    </tr>

                    <tr>
                        <td align = "center"><form:input path="username" placeholder="Username" class = "form-control" id = "username" required = "required"/>
                        <td align = "center"><form:input path="password" type = "password" placeholder = "Password" class = "form-control" id = "password" required = "required"/>

                    </tr>

                    <tr>

                    <td align = "center"><button type="submit" class="btn btn-success">Submit</button>
                    <td align = "center"><button type="reset" class="btn btn-default">Clear</button>

                </tr>
        </table>
        </form:form>
    </div>
    </div> <!-- container-register -->
</div>
</body>
</html>
```

```
<!-- new_member.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
    <jsp:include page = "navbar_afterlogin.jsp" />
    <div class = "container-members">
        <form:form action = "addMembers" modelAttribute = "teamForm">
        <h3>Invite team members</h3>
        <table class = "table">
        <thead>
        <tr>
            <th>First Name
            <th>Last Name
            <th>Email Address
        </tr>
        </thead>
        <tbody>
            <c:forEach items = "${teamForm.users }" varStatus = "index" begin= "0" end = "${numberOfTM-1 }">
            <tr>
                <td><input name = "users[${index.index }].firstName" class = "form-control" required = "required" pattern = "[a-zA-Z0-9]+">
                <td><input name = "users[${index.index }].lastName" class = "form-control" required = "required" pattern = "[a-zA-Z0-9]+">
                <td><input type = "email" name = "users[${index.index }].email" class = "form-control" required = "required
```

```
                                        ">
                                <input type = "hidden" value = "team_member" name = "users[${index.index}].role" />
                        </tr>
                        </c:forEach>
                </tbody>

                </table>
                <h3>Invite product owner</h3>
                <table class = "table">
                <thead>
                <tr>
                        <th>First Name
                        <th>Last Name
                        <th>Email Address
                        <th>Outside Client?
                </tr>
                </thead>
                <tbody>
                        <c:forEach items = "${teamForm.users}" varStatus = "index" begin = "${numberOfTM }" end = "${numberOfTM + numberOfPO-1 }">
                        <tr>
                                <td><input name = "users[${index.index}].firstName" class = "form-control" required = "required">
                                <td><input name = "users[${index.index}].lastName" class = "form-control" required = "required">
                                <td><input type ="email" name = "users[${index.index}].email" class = "form-control" required = "required">
                                <td><input type = "checkbox" value = "outsideclient">
                                <input type = "hidden" value = "product_owner" name = "users[${index.index}].role" />
                        </tr>
                        </c:forEach>
                </tbody>

                </table>
                <br>
                <td align = "center"><button type="submit" class="btn btn-success">Submit</button>
        <td align = "center"><button type="reset" class="btn btn-default">Clear</button>
                </form:form>
        </div>
</body>
</html>
```

```
<!-- notification.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
        <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
        <jsp:include page = "navbar_scrum.jsp" />

        <div class = "container-project">
        <table class="table table-striped" cellpadding = "5px" height = "330px" width = "930px" >
                <thead style = "background-color: #ffffcc; width:930px;" >
                        <tr>
                                <th>Notifications
                                <th>&nbsp;
                        </tr>
                </thead>
                <tbody style = "display:block; overflow-y:auto; height: 300px; width:930px;">
                <c:forEach items = "${notifications}" var = "notif">
                <c:if test = "${notif.type == 'new_member'}">
                <tr>
                <td style = "width: 630px;"><c:out value = "${notif.newMember } was added to the group"/>
                <td style = "width: 300px;"><c:out value = "${notif.date }"/>
                </tr>
                </c:if>
                <c:if test = "${notif.type eq 'scrum_meeting'}">
                <tr>
                <td style = "width: 630px;"><a href = "${contextPath }/viewScrumMeeting?sid=${notif.sprintID}"><c:out value = "Please participate the daily scrum meeting" /></a>
                <td style = "width: 300px;"><c:out value = "${notif.date }" />
                </tr>
                </c:if>
                <c:if test = "${notif.type eq 'sprint_review' }">
                <tr>
                <td style = "width: 630px;"><c:out value = "Notice: Upcoming Sprint Review!" />
                <td style = "width: 300px;"><c:out value = "${notif.date }" />
                </tr>
                </c:if>
                </c:forEach>
        </table>
        </div>

</body>
</html>
```

```
<!-- poker_planning.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>

 <!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">
<jsp:include page="import.jsp" />
</head>
<body>
<jsp:include page = "navbar_scrum.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<div class="alert alert-info alertMessage">
                        Estimation of number of hours per task should be done using Poker Planning - where values are fibonacci numbers.
                        <button type="button" class="close" data-dismiss="alert">x</button>
</div>
<div class = "container-members">
        <c:out value = "${story.storyName }"/>
        <form:form method="post" name="classForm" id="classForm" modelAttribute="taskForm" action = "${contextPath }/addTasks?sid=${sprintID}&rmng=${remaining }&ts=${ts}">
        <table class="table" cellpadding = "5px" width = "920px" >
                <thead style = "background-color:#ffffcc;" width = "920px">
                        <tr>
                                <td>Task Name
                                <td>Volunteer
                                <td>Estimate (in hrs)
                        </tr>
                </thead>
                <tbody>
                        <tr>
                                <td><spring:bind path="taskForm.tasks[0].taskName">
                                        <form:input path="${status.expression}" class = "form-control"/>
                                        </spring:bind>


                                <td><spring:bind path="taskForm.tasks[0].accountID">
                                        <select class ="form-control" name = "tasks[0].accountID" >
                                        <c:forEach items = "${members}" var = "member">
                                        <c:if test = "${member.role ne 'product_owner'}">
                                        <option value = "${member.accountID }"><c:out value = "${member.name }"/></option>
                                        </c:if>
                                        </c:forEach>
                                        </select>
                                        </spring:bind>
                                <td><spring:bind path="taskForm.tasks[0].estimate">
                                        <select name = "tasks[0].estimate" class = "form-control">
                                        <option value = "1">1
                                        <option value = "2">2
                                        <option value = "3">3
                                        <option value = "5">5
                                        <option value = "8">8
                                        <option value = "13">13
                                        </select>
                                        </spring:bind>
                                <td><input type="button" id="addTaskButton" value="Add" />

                        </tr>
                        <tr id = "submitRow">
                                <td>&nbsp;</td>
                                <td>&nbsp;</td>
                                <td>&nbsp;</td>
                                <td>&nbsp;</td>
                        </tr>
                </tbody>
        </table>
        <button type="submit" value="Save Class" class = "btn btn-success" >Save</button>
        </form:form>
</div>
<script type="text/javascript">
$(document).ready(function() {
        var storyPosition = 0;
        $("#addTaskButton").click(function() {
                storyPosition++;

                $.get("<%=request.getContextPath()%>/appendStoryView", { fieldId: storyPosition},
                        function(data){
                                $("#submitRow").before(data);
                });
        });
});
</script>
</body>
</html>
```

```
<!-- product_backlog.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
<jsp:include page = "navbar_scrum.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<div class="alert alert-info alertMessage">
                Product Backlog contains the list of overall requirements written as "
                    User Stories" by the Product Owner.
                        <button type="button" class="close" data-dismiss="alert">x</
                            button>
</div>
<div class = "container-project">
        <div  class="table-scroll">
                <table class = "table" cellpadding = "5px" height = "300px" width = "930
                    px">
                        <thead>
                                <tr bgcolor = "white" style = "display: block; width:
                                    930px;">
                                                        <th style = "width: 310px;">User
                                                            Story</th>
                                                        <th style = "width: 310px;">
                                                            Details</th>
                                                        <th style = "width: 310px;">
                                                            Status</th>
                                </tr>
                        </thead>

                        <tbody style = "display:block; overflow-y:auto; max-height: 300
                            px; width:930px; background-color: #ffffcc">
                                <c:forEach items = "${ stories}" var = "story">
                                <tr>
                                        <td style = "width: 310px;"><c:out value = "${
                                            story.storyName }" />
                                        <td style = "width: 310px;"><c:out value = "${
                                            story.detail }" />
                                        <td style = "width: 310px;"><c:out value = "${
                                            story.status }" />
                                </tr>
                                </c:forEach>

                        </tbody>

                </table>
        </div>
                <c:if test = "${nostory != null }">
                                        <center><strong><c:out value="${nostory }"/></
                                            strong></center>
                </c:if>

                <c:if test = "${sessionScope.role eq 'product_owner' && sessionScope.
                    status == 'ongoing' }">
                <table cellpadding = "10px" width = "930px">
                <tr>
                <td><form action = "addUserStories"><input type = "number" class = "
                    form_control" name = "numberOfStories" required = "required"
                    placeholder = "No. of new user stories" min ="1"> <button class = "
                    btn btn-primary" type = "submit" >Add User Story</button></form>

                <td align = "right"><c:if test = "${stories != null }">
                <center><a href = "${pageContext.request.contextPath }/
                    editProductBacklog"><button class = "btn btn-primary">Edit Product
                    Backlog</button></a></center>
                </c:if>
                </tr>
                </table>
                </c:if>
</div>

</body>
</html>
```

```
<!-- projects.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

```

```
<!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”>
<jsp:include page=”import.jsp” />
</head>
<body>
<jsp:include page = ”navbar_afterlogin.jsp”/>
<c:set var=”contextPath” value=”${pageContext.request.contextPath}”/>
<c:if test=”${update eq 'yes'}”>
        <script type=”text/javascript”>
             $(window).load(function(){
            $('#ProjectEditorModal').modal('show');
             });
        </script>
</c:if>
<c:if test=”${addproject != null }”>
        <div class=”alert alert-success alertMessage”>
            <c:out value=”${addproject }”/>
            <button type=”button” class=”close” data-dismiss=”alert”>x</button>
        </div>
</c:if>
<c:if test=”${invite != null }”>
        <div class=”alert alert-success alertMessage”>
            <c:out value=”${invite }”/>
            <button type=”button” class=”close” data-dismiss=”alert”>x</button>
        </div>
</c:if>
<div class = ”container-project”>
            <div  class=”table-scroll”>
                <table class=”table” cellpadding = ”5px” height = ”330px” width = ”930px” >
                    <thead>
                        <tr bgcolor = ”#94ff70” style = ”display: block; width: 930px;”>
                            <th style = ”width: 230px;”>My Projects</th>
                            <th style = ”width: 340px;”>Description</th>
                            <th style = ”width: 130px;”>Role</th>
                            <th style = ”width: 130px;”>Status</th>
                            <th style = ”width: 100px;”>&nbsp;</th>
                        </tr>
                    </thead>
                    <tbody style = ”display:block; overflow-y:auto; max-height: 300px; width:930px; background-color:#ffffcc”>

                        <c:forEach var = ”proj” items = ”${projects }”>
                        <c:forEach var = ”projDetail” items = ”${projectDetails }”>
                        <c:if test = ”${proj.projectID eq projDetail.projectID }”>
                        <tr>
                            <td style = ”width: 230px;”><a href = ”${contextPath }/viewProject?pid=${proj.projectID}”><c:out value = ”${projDetail.projectName }”/></a>
                            <td style = ”width: 340px;”><c:out value = ”${projDetail.projectDescription }” />
                            <td style = ”width: 130px;”><c:out value = ”${proj.role }” />
                            <td style = ”width: 130px;”><c:out value = ”${projDetail.status }” />
                            <c:if test = ”${proj.role eq 'scrum_master' }”><td style = ”width: 100px;”><a href = ”${contextPath }/editProject?pid=${proj.projectID}”><button class = ”btn btn-info”>Edit</button></a></c:if>
                        </tr>
                        </c:if>
                        </c:forEach>
                        </c:forEach>
                    </tbody>
                </table>
                <c:if test = ”${noexisting != null }”>
                    <center><strong><c:out value=”${noexisting }”/></strong></center>
                </c:if>
                </div>
                <c:if test = ”${sessionScope.username ne 'powner' }”>
                <center><a href = ”${pageContext.request.contextPath }/newProject”><button class = ”btn btn-primary”>New Project</button></a></center>
```

```
                                    </c:if>
</div>
        <!-- edit profile modal -->
                <div class="modal fade" id = "ProjectEditorModal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                        <h4 class="modal-title">Edit Project</h4>
                </div>
                <div class="modal-body">
                        <form:form method="post" action="updateProject" modelAttribute = "newProject">
                                <table align = "center" cellpadding = "3px" width = "80%">
                                        <tr>
                                                <td align = "center"><form:label path = "projectName" class="col-sm-2 control-label">Project Name</form:label>
                                                <td align = "center"><form:input path="projectName" value = "${updateProject.projectName }" class = "form-control" required = "required" />
                                        </tr>
                                        <tr>
                                                <td align = "center"><form:label path = "projectDescription" class="col-sm-2 control-label">Project Description</form:label>
                                                <td align = "center"><textarea name="projectDescription" class = "form-control" required = "required" ><c:out value = "${updateProject.projectDescription }"/></textarea>
                                                <form:input type = "hidden" path = "projectID" value = "${updateProject.projectID }"/>
                                        </tr>

                                        <tr>

                                        <td align = "center"><button type="submit" class="btn btn-success">Save Changes</button>

                                </tr>
                </table>
                </form:form>


                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
                </div><!-- /.modal -->
</body>
</html>
```

```
<!-- register.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
    <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<jsp:include page="import.jsp" />
</head>
<body >
        <jsp:include page = "navbar_regular.jsp" />


         <div class = "container-register">
         Sign up for new account.
        <div style = "padding-top: 10px;">
        <!-- registration form -->
        <form:form method="post" action="addUser" modelAttribute = "user" id = "registration-form">
                <table align = "center" width = "380px" height = "180px" >
                <tr>
                        <td align = "center"><form:input name = "firstname" path="firstName" placeholder = "First Name" class = "form-control" id = "firstname" required = "required" />
                        <td align = "center"><form:input name = "lastname" path="lastName" placeholder = "Last Name" class = "form-control" id = "lastname" required = "required" />
                </tr>


                <tr>
                        <td align = "center"><form:input type = "email" path="email" placeholder = "Email Address" class = "form-control" id = "email" required = "required" />

                </tr>

        <tr>
            <td align = "center"><form:input path="username" placeholder="Username" class = "form-control" id = "username" required = "required"/>
            <td align = "center"><form:input path="password" type = "password" placeholder = "Password" class = "form-control" id = "password" required = "required"/>

        </tr>


        <tr>

        <td align = "center"><button type="submit" class="btn btn-success">Submit</button>
        <td align = "center"><button type="reset" class="btn btn-default">Clear</button>

    </tr>
    </table>
    </form:form>
    </div>

</div>
</body>
</html>
```

```
<!-- scrummeeting.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
    <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
    <jsp:include page = "navbar_scrum.jsp" />
    <div class="alert alert-info alertMessage">
        In Daily Scrum Meeting, each members needs to report their accomplishments, plans for the next scrum meeting and impediments.
            <button type="button" class="close" data-dismiss="alert">x</button>
</div>
    <div class = "span3">
    <table width = "100%" height = "100%" cellpadding = "20px">
    <tr align = "center">
        <td><h4><u>Sprint <c:out value = "${sprintNumber }" /></u></h4>
    </tr>
    <tr align = "center">
        <td>Remaining days before Sprint Review: <b><c:out value = "${remainingdays }"/></b>
    </tr>
    <tr align = "center">
        <td><a href = "${contextPath }/viewScrumBoard?sid=${sprintID}"><button class = "btn btn-primary">Scrum Board</button></a>
    </tr>
    <tr align = "center">
        <td><a href = "${contextPath }/viewScrumMeeting?sid=${sprintID}"><button class = "btn btn-primary" disabled>Scrum Meeting</button></a>
    </tr>
    <tr align = "center">
        <td><a href = "${contextPath }/viewSprintBacklog?sid=${sprintID}"><button class = "btn btn-primary" >Sprint Backlog</button></a>
    </tr>
    </table>
    </div>
    <div class = "span4">
    <br>
    <table class = "table" cellpadding = "5px" height = "300px" width = "500px" align = "center" style = "margin-left: auto; margin-right: auto;">
        <thead align = "center">
            <tr bgcolor = "white" style = "display: block; width: 500px;">
            <th style = "width: 500px;" align = "center"><center>Scrum Meetings</center>
            </tr>
        </thead>
        <tbody style = "display:block; overflow-y:auto; max-height: 300px; width:500px; background-color: #ffffcc; margin-left: auto; margin-right: auto; ">
        <c:forEach items = "${meetings }" var = "meet">
            <tr>
                <td style ="width: 500px;" align = "center"><a href = "${contextPath }/viewScrumMeetingDetail?sid=${sprintID }&date=${meet.date}"><c:out value = "${meet.date }"></c:out></a>
            </tr>
        </c:forEach>
        </tbody>
        <c:if test = "${sprintstatus == 'open' }">
        <c:if test = "${sessionScope.role eq 'scrum_master' && currentmeeting eq 'no' }">
            <tr>
```

```
                                <td style ="width: 500px;"><a href = "#newmeetingmodal"
                                     data-toggle = "modal"><button class = "btn btn-
                                     primary">Preside Daily Scrum Meeting</button></a>
                        </tr>
                </c:if>
                <c:if test = "${sessionScope.role eq 'team_member' && fillmeeting eq 'no
                     ' && currentmeeting eq null }">
                        <tr>
                                <td style ="width: 500px;"><a href = "#newmeetingmodal2"
                                      data-toggle = "modal"><button class = "btn btn-
                                     primary">Answer Daily Scrum Meeting</button></a>
                        </tr>
                </c:if>
                </c:if>
        </table>
        </div>

        `<!-- newcurrentmodal -->
<div class="modal fade" id = "newmeetingmodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                             hidden="true">&times;</button>
                        <h4 class="modal-title">Daily Scrum Meeting</h4>
                </div>
                <div class="modal-body">
                <form:form action = "addScrumMeeting" modelAttribute = "meeting" >
                <table align = "center" cellpadding = "5px">
                        <tr>
                                <td><form:label path = "accomplishment" class = "control
                                     -label">What have you accomplished in the last
                                     scrum meeting?</form:label>
                                        <form:input type = "hidden" path = "sprintID"
                                             value = "${sprintID }"/>
                        </tr>
                        <tr>
                                        <td><form:textarea class = "form-control"  path
                                             = "accomplishment"></form:textarea>

                        </tr>
                        <tr>
                                <td><form:label path = "plan" class = "control-label">
                                     What are you planning to do/finish for the next
                                     scrum meeting?</form:label>

                        </tr>
                        <tr>
                                        <td><form:textarea class = "form-control"  path
                                             = "plan"></form:textarea>

                        </tr>
                        <tr>
                                <td><form:label path = "impediment" class = "control-
                                     label">Is there any impediment?</form:label>

                        </tr>
                        <tr>
                                        <td><form:textarea class = "form-control"  path
                                             = "impediment"></form:textarea>

                        </tr>


                        <tr>

                                        <td align = "center"><button type="submit" class
                                             ="btn btn-success">Submit</button>

                        </tr>

                </table>
                </form:form>
                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
        `<!-- newcurrentmodal -->
<div class="modal fade" id = "newmeetingmodal2" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                             hidden="true">&times;</button>
                        <h4 class="modal-title">Daily Scrum Meeting</h4>
                </div>
                <div class="modal-body">
                <form:form action = "addScrumMeeting" modelAttribute = "meeting" >
                <table align = "center" cellpadding = "5px">
                        <tr>
                                <td><form:label path = "accomplishment" class = "control
                                     -label">What have you accomplished in the last
                                     scrum meeting?</form:label>
                                        <form:input type = "hidden" path = "sprintID"
                                             value = "${sprintID }"/>
                        </tr>
                        <tr>
                                        <td><form:textarea class = "form-control"  path
                                             = "accomplishment"></form:textarea>
```

```

            </tr>
            <tr>
                <td><form:label path = "plan" class = "control-label">What are you planning to do/finish for the next scrum meeting?</form:label>

            </tr>
            <tr>
                    <td><form:textarea class = "form-control"  path = "plan"></form:textarea>

            </tr>
            <tr>
                <td><form:label path = "impediment" class = "control-label">Is there any impediment?</form:label>

            </tr>
            <tr>
                    <td><form:textarea class = "form-control"  path = "impediment"></form:textarea>

            </tr>


            <tr>

                    <td align = "center"><button type="submit" class="btn btn-success">Submit</button>

            </tr>

        </table>
        </form:form>
        </div> <!-- modal-body -->
        </div><!-- /.modal-content -->
            </div><!-- /.modal-dialog -->
</div><!-- /.modal -->

</body>
</html>
```

```
<!-- scrummeeting_detail.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
    <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
    <jsp:include page = "navbar_scrum.jsp" />

    <div class = "container-project">
    <ul class="breadcrumb">
        <li>
    <a href="${contextPath }/viewScrumMeeting?sid=${sprintID}">Scrum Meetings</a> <span class="divider"></span>
        </li>
        <li class="active">${date}</li>
    </ul>

    <br>
    <table class = "table" cellpadding = "5px" height = "300px" width = "930px">
    <thead>
                <tr bgcolor = "white" style = "display: block; width: 920px;">
                            <th style = "width: 230px;">Member</th>
                            <th style = "width: 230px;">Accomplishments</th>
                            <th style = "width: 230px;">Plans</th>
                            <th style = "width: 230px;">Impediments</th>
                </tr>
    </thead>
    <tbody style = "display:block; overflow-y:auto; max-height: 300px; width:920px; background-color: #ffffcc">
                <c:forEach items = "${details }" var = "detail">
                <tr>
                    <td style = "width: 230px;"><c:out value = "${detail.member }"/>
                    <td style = "width: 230px;"><c:out value = "${detail.accomplishment }"/>
                    <td style = "width: 230px;"><c:out value = "${detail.plan }"/>
                    <td style = "width: 230px;"><c:out value = "${detail.impediment }"/>

                </tr>
                </c:forEach>
    </tbody>
    </table>
    </div>

</body>
</html>
```

```
<!-- sprint.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

 <!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">
<jsp:include page="import.jsp" />
</head>
<body>
        <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
        <jsp:include page = "navbar_scrum.jsp" />
        <div class="alert alert-info alertMessage">
Scrum Board is a visual task-tracking tool that contains the current tasks and is grouped thru the table that labeled "Not yet started", "Ongoing" and "Completed"
                        <button type="button" class="close" data-dismiss="alert">x</button>
        </div>
        <div class = "span3">
        <table width = "100%" height = "100%" cellpadding = "20px">
        <tr align = "center">
                <td><h4>Sprint ${sprintNumber }</h4>
        </tr>
        <tr align = "center">
                <td>Remaining days before Sprint Review: <b><c:out value = "${remainingdays }"/></b>
        </tr>
        <tr align = "center">
                <td><button class = "btn btn-primary" href = "${contextPath }/viewScrumBoard" disabled data-toggle = "tooltip" data-placement = "right" title = "Scrum Board - a visual task-tracking tool that contain the current sprint tasks that is grouped thru the table that labeled 'Not yet Started', 'Ongoing' and 'Completed'">Scrum Board</button>
        </tr>
        <tr align = "center">
                <td><a href = "${contextPath }/viewScrumMeeting?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "right" title = "Scrum Meeting - contains the accomplishments, the plans for the next meeting and the impediments">Scrum Meeting</button></a>
        </tr>
        <tr align = "center">
                <td><a href = "${contextPath }/viewSprintBacklog?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "right" title = "Sprint Backlog - contains the list of the tasks for the whole sprint" >Sprint Backlog</button></a>
        </tr>
        </table>
        </div>
        <div class = "span4" style = "background-color: inherit">
                <table class="table" height = "450px" width = "770px" >
                                        <thead>
                                                <tr bgcolor = "#94ff70" style = "display:block; width: 770px;">
                                                        <th width = "256px">Not Yet Started</th>
                                                        <th width = "256px">Ongoing</th>
                                                        <th width = "256px">Completed</th>

                                                </tr>
                                        </thead>
                                        <tbody style = "display:block; overflow-y: auto; height: 450px; width:770px; background-color:#ccb299">
                                        <c:forEach items = "${tasks }" var = "task">
                                                <c:if test = "${task.status eq 'not yet started' }">
                                                <tr>
                                                <td width = "256px"><div style = "background-color: #ffb8ff; width: 210px; padding: 10px; box-shadow: 10px 10px 5px #888888;">User Story: <c:out value = "${task.storyName }"/> <br/>Task Name: <c:out value = "${task.taskName }"/> <br /> Estimate: <c:out value = "${task.estimate }"/></div>
                                                <td width = "256px">
                                                <td width = "256px">
                                                </tr>
                                                </c:if>

                                                <c:if test = "${task.status eq 'ongoing' }">
                                                <tr>
                                                <td width = "256px">
                                                <td width = "256px"><div style = "background-color: #94dbff; width: 210px; padding: 10px; box-shadow: 10px 10px 5px #888888;">User Story: <c:out value = "${task.storyName }"/><br/>Task Name: <c:out value = "${task.taskName }"/> <br /> Estimate: <c:out value = "${task.estimate }"/></div>
                                                <td width = "256px">
                                                </tr>
                                                </c:if>

                                                <c:if test = "${task.status eq 'done' }">
                                                <tr>
                                                <td width = "256px">
                                                <td width = "256px">
                                                <td width = "256px"><div style = "background-color: #ffad33; width: 210px; padding: 10px; box-shadow: 10px 10px 5px #888888;">User Story: <c:out value = "${task.storyName }" /> <br/>Task Name: <c:out value = "${task.taskName }"/> <br /> Estimate: <c:out value = "${task.estimate }"/></div>
                                                </tr>
                                                </c:if>

                                        </c:forEach>
                                        </tbody>
                                        <tr>
                                                <td align = "justify"><a href = "#burndownchartmodal" data-toggle = "modal"><button class = "btn btn-success" data-toggle = "tooltip" data-placement = "right" title = "Burndown Chart - the chart which shows the trend of work to the remaining time in a Sprint">View Burndown Chart</button></a>
                                        </tr>
                                    </table>
        </div>

        `<!-- burndownchart -->
<div class="modal fade" id = "burndownchartmodal" tabindex = "-1" >
                        <div class="modal-dialog" style = "max-width: 70%;">
                <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                        <h4 class="modal-title">Burndown Chart -- the chart which shows the trend of work to the remaining time in a Sprint</h4>
                </div>
                <div class="modal-body">
                <div id = "container" style="width: 400px; height: 400px; margin: 0 auto">
                </div>
                </div> <!-- modal-body -->
                </div><!-- /.modal-content -->
                        </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
<c:forEach begin = "1" end = "${totalDays }" varStatus = "index">
<c:set var = "xAxis" value = " ${xAxis },${index.index }"/>
</c:forEach>
<script>

$(function () {
    $('#container').highcharts({
        title: {
                text: 'Burndown Chart',
                x: -20 //center
        },
        subtitle: {
                text: '${sprint.dateStarted} to ${sprint.sprintReview}',
                x: -20
        },
        xAxis: {
                        title: {
                        text: 'Days'
                },
                categories: []
        },
        yAxis: {
                title: {
                        text: 'Remaining Number of Hours'
                },
                plotLines: [{
                        value: 0,
                width: 1,
                color: '#808080'
                }]
        },
        tooltip: {
                valueSuffix: 'hours'
        },
        legend: {
            layout: 'vertical',
            align: 'right',
            verticalAlign: 'middle',
            borderWidth: 0
        },
        series: [{
            name: 'Actual',
            data: [${burnvalues}]
        }, {
            name: 'Ideal',
            data: [ [0, ${totalEstimate}], [${totalDays}, 0] ]
        }]
    });
});
</script>
</body>
</html>
```

```
<!-- sprintbacklog.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
        <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
        <jsp:include page = "navbar_scrum.jsp" />
        <div class="alert alert-info alertMessage">
                Sprint Backlog contains the list of tasks for the whole Sprint.
                        <button type="button" class="close" data-dismiss="alert">x</button>
</div>
        <div class = "span3">
        <table width = "100%" height = "100%" cellpadding = "20px">
        <tr align = "center">
                <td><h4><u>Sprint <c:out value = "${sprintNumber }" /></u></h4>
        </tr>
        <tr align = "center">
                <td>Remaining days before Sprint Review: <b><c:out value = "${remainingdays }"/></b>
        </tr>
        <tr align = "center">
                <td><a href = "${contextPath }/viewScrumBoard?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "right" title = "Scrum Board - a visual task-tracking tool that contain the current sprint tasks that is grouped thru the table that labeled 'Not yet Started', 'Ongoing' and 'Completed'">Scrum Board</button></a>
        </tr>
        <tr align = "center">
                <td><a href = "${contextPath }/viewScrumMeeting?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "right" title = "Scrum Meeting - contains the accomplishments, the plans for the next meeting and the impediments">Scrum Meeting</button></a>
        </tr>
        <tr align = "center">
                <td><button class = "btn btn-primary" href = "${contextPath }/viewSprintBacklog" disabled data-toggle = "tooltip" data-placement = "right" title = "Sprint Backlog - contains the list of the tasks for the whole sprint">Sprint Backlog</button>
        </tr>
        </table>
        </div>
        <div class = "span4">
        <c:if test = "${pokerplanning eq 'yes' }">
                        <div class="alert alert-success alertMessage">
                        <c:out value="No tasks for the sprint"/>
                        <button type="button" class="close" data-dismiss="alert">x</button>
                        </div>
                        <c:if test = "${sessionScope.role eq 'scrum_master'}" >
                        <center><a href = "${contextPath }/pokerPlanning?sid=${sprintID}&rmng=0"><button class = "btn btn-primary">Add Task</button></a></center>
                        </c:if>
        </c:if>
        <c:if test = "${pokerplanning eq null }">
        <br>
        <c:if test = "${taskadded != null }">
        <div class="alert alert-success alertMessage">
                        <c:out value="${taskadded }"/>
                        <button type="button" class="close" data-dismiss="alert">x</button>
        </div>
        </c:if>

        <table class = "table table-striped" cellpadding = "5px" width = "770px">
                        <thead align = "center">
```

```
                    <tr bgcolor = "white" style = "display: block; width: 770px;">
                                <th style = "width: 154px;" data-toggle ="tooltip" data-placement = "top" title = "User Story - required features, improvement goals, and, possibly known defects provided by the Product Owner">User Story</th>
                                <th style = "width: 154px;" data-toggle = "tooltip" data-placement = "top" title = "Task -- certain actions that are created to comple a certain user story">Sprint Task</th>
                                <th style = "width: 154px;" >Volunteer</th>
                                <th style = "width: 77px;" >Estimate</th>
                                <th style = "width: 77px;">Actual </th>
                                <th style = "width: 154px;">Status</th>
                    </tr>
                </thead>

                <tbody align = "center" style = "min-height:300px;">
                    <c:forEach items = "${tasks}" var = "task">
                    <tr style = "display:block; width:770px; background-color: #ffffcc; ">
                        <td style = "width: 154px;" ><c:out value = "${task.storyName }"></c:out></td>
                        <td style = "width: 154px;" ><a href = "${contextPath }/viewActivity?tid=${task.taskID}"><c:out value = "${task.taskName}"></c:out></a></td>
                        <td style = "width: 154px;" ><c:out value = "${task.volunteer }"></c:out></td>
                        <td style = "width: 77px;" ><c:out value = "${task.estimate }"></c:out></td>
                        <td style = "width: 77px;" ><c:out value = "${task.actual }"></c:out></td>
                        <td style = "width: 154px;"><c:out value = "${task.status }"></c:out></td>
                    </tr>
                    </c:forEach>

                </tbody>

            </table>
            <c:if test = "${sprintstatus == 'open' && sessionScope.role ne 'product_owner' }">
                <center><a href = "#newtaskmodal" data-toggle = "modal"><button class = "btn btn-primary">Add New Task</button></a></center>
            </c:if>
            <c:if test = "${(sessionScope.role eq 'product_owner') && (sprintreview == 'yes') && (sprintstatus == 'open')}" >
                <center><a href = "${contextPath }/sprintReview?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "top" title = "Sprint Review - list of the comments about the output of the current sprint">Sprint Review</button></a></center>
        </c:if>
        <c:if test = "${ (sprintstatus == 'closed')}" >
            <center><a href = "${contextPath }/viewSprintReview?sid=${sprintID}"><button class = "btn btn-primary" data-toggle = "tooltip" data-placement = "top" title = "Sprint Review - list of the comments about the output of the current sprint">View Sprint Review</button></a></center>
        </c:if>
        </c:if>
        <br>
        </div>

        <!-- newactivitymodal -->
        <div class="modal fade" id = "newtaskmodal" tabindex = "-1" >
                <div class="modal-dialog" style = "max-width: 70%;">
            <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                <h4 class="modal-title">New Task</h4>
            </div>
            <div class="modal-body">
            <form:form action = "addTask" modelAttribute = "newTask" >
            <table align = "center" cellpadding = "5px">
            <tr>
                <td><form:label path = "storyID" class = "control-label">User Story</form:label>
                <td><form:select path = "storyID" class = "form-control" required = "required">
                    <c:forEach items = "${sprintstories }" var = "story">
                    <option value = "${story.storyID }"><c:out value = "${story.storyName }"/></option>
                    </c:forEach>
```

```
                                </form:select>
                    </tr>
                    <tr>
                            <td><form:label path = "taskName" class = "control-label">Task
                                 Name</form:label>
                                            <td><form:input class = "form-control"  path = "
                                                 taskName" required = "required"></form:
                                                 input>

                    </tr>
                    <tr>
                            <td><form:label path = "accountID" class = "control-label">
                                 Volunteer</form:label>
                            <td><form:select path = "accountID" class = "form-control"
                                 required = "required">
                                <c:forEach items = "${members}" var = "member">
                                                    <c:if test = "${member.role ne '
                                                         product_owner' }">
                                                    <option value = "${member.accountID }"><
                                                         c:out value = "${member.name }"/>
                                                    </c:if>
                                            </c:forEach>
                                </form:select>
                    </tr>
                    <tr>
                            <td><form:label path = "estimate" class = "control-label">
                                 Estimate</form:label>
                            <td><form:select path = "estimate" class = "form-control"
                                 required = "required">
                                <option value = "1">1
                                        <option value = "2">2
                                        <option value = "3">3
                                        <option value = "5">5
                                        <option value = "8">8
                                        <option value = "13">13
                                </form:select>
                                <form:input type = "hidden" value = "${sprintID }" path
                                     = "sprintID"/>
                    </tr>
                    <tr>

                                        <td align = "center"><button type="submit" class
                                             ="btn btn-success">Add Activity</button>

                    </tr>
                    </table>
                    </form:form>
                    </div>
                    </div>
                    </div>
        </div>

</body>
</html>
```

```
<!-- sprint_review_view.jsp -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<!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">
<jsp:include page="import.jsp" />
</head>
<body>
        <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
        <jsp:include page = "navbar_scrum.jsp" />

        <div class = "container-project">
                <ul class="breadcrumb">
                <li>
        <a href="${contextPath }/viewSprintBacklog?sid=${sprintID}">Sprint Backlog</a> <
             span class="divider"></span>
                </li>
                <li class="active">Sprint Review</li>
                </ul>

                <br>
                <table class = "table" cellpadding = "5px" height = "300px" width = "920
                     px">
                <thead>
                                <tr bgcolor = "white" style = "display: block; width:
                                     920px;">
                                                        <th style = "width: 200px;">User
                                                             Story for this Sprint</th>
                                                        <th style = "width: 500px;">
                                                             Review</th>
                                                        <th style = "width: 220px;">User
                                                             Story Status</th>



                                </tr>
                </thead>
```

```
                    <tbody style = "display:block; overflow-y:auto; max-height: 280px; width
                         :920px; background-color: #ffffcc">

                                        <c:forEach items = "${reviews }" varStatus = "index" var
                                              = "review" >
                                        <tr>
                                                  <td style = "width: 200px;"><c:out value = "${
                                                        review.storyName }"/>
                                                  <td style = "width: 500px;"><c:out value = "${
                                                        review.review }"/>
                                                  <td style = "width: 220px;"><c:out value = "${
                                                        review.status }"/>

                                        </tr>
                                        </c:forEach>

                    </tbody>
                    </table>

          </div>
</body>
</html>
```

XII. Acknowledgement

MARAMING SALAMAT SA LAHAT!

Unang una. Thank you God! As in thank you. Sa lahat lahat po as in lahat lahat. Thank you sa pagiging confidant, yung unang unang matatakbuhan kapag feeling mo susuko ka na.

Para sa adviser ko, kay Sir Bernie. Hi Sir! Thank you po. Ang daming dapat ipagpasalamat pero isang maraming maraming thank you po talaga!

Sa lahat ng naging prof ko, Thank you very much po! Hindi matatawaran ang lahat ng naituro niyo for the past years. Simula freshie to senior year. Thumbs up po sa inyong lahat!

Syempre, sa pinakamamahal kong mga matatalik na kaibigan ngayong college. Hi Sarah, Celina, Gera at Sherwin! Thank you! Kung hindi dahil sa inyo baka hindi ko naexperience ang social life, pageenjoy at ang pagiging solid at loyal na kaibigan ngayong college. Yung four years na punong puno ng saya kasama niyo.

Dose '10 with affiliates(Stuff, Team Smile, BOR, 09s)!! Thank you sa lahat. Lahat ng bonding. Sana madagdagan pa talaga yun. Salamat at nagkaron ako ng opportunity na lahat kayo eh macoconsider ko talagang mga kabigan. Nafeel ko naman na sa bawat grupo eh madaling makaconnect. Goodluck sating lahat guys!

And finally sa pamilya ko. Sobra sobrang salamat sa full force na suporta. Sa 18-going-to-19 years na pagaaruga ng maigi. Wala pong SP na matatapos kung wala rin po kayo. Thank you.

Pahabol pala! Sa pinakabestfriend ko ngayong college. Salamat kasi 4 years kang anjan. Araw man o gabi. Sa lahat ng MP, papers, pdfs etc. Ikaw pa ang nagpapasaya sakin kapag stressed ako or walang magawa dahil sa mga movies, dramas at anumang shows na meron ka. Pati ba naman tong SP kinaya mo. Kaya bilib ako sayo eh! Kung hindi dahil sayo baka handicapped ako at di makakatapos talaga forever. Super thank you pinakamamahal kong netbook! :)

At muli, Maraming maraming Salamat sa lahat!