

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. (<i>candidate</i>)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Aldrich Colin K. Co, M.Sc. (<i>candidate</i>)	_____	_____
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.

<hr/> Ma. Sheila A. Magboo, M.Sc. Unit Head Mathematical and Computing Sciences Unit Department of Physical Sciences and Mathematics	<hr/> 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

Acceptance Sheet	i
Abstract	i
List of Figures	iv
List of Tables	vi
I. Introduction	1
A. Background of the Study	1
B. Statement of the Problem	5
C. Objectives of the Study	5
D. Significance of the Project	8
E. Scope and Limitations	9
F. Assumptions	10
II. Review of Related Literature	11
III. Theoretical Framework	15
IV. Design and Implementation	23
V. Architecture	36
A. System Architecture	36
B. Technical Architecture	37
VI. Results	38
A. General View	38
B. Scrum Role View	40
C. Manage Accounts	49
VII. Discussions	52

VIII. Conclusions	54
IX. Recommendations	56
X. Bibliography	57
XI. Appendix	60
A. Source Code	60
XII. Acknowledgement	129

List of Figures

1	Scrum	16
2	Product Backlog	20
3	Sprint Backlog	21
4	Burndown Chart	22
5	Context Diagram of Scrum Project Planner Primer	23
6	Use Case Diagram for Scrum Master and Team Member	25
7	Use Case Diagram for Product Owner, Guest User and Administrator	26
8	Overview	27
9	Create Project	28
10	View Project	29
11	Daily Scrum Meeting and Burndown Chart	30
12	Create Account	31
13	Entity Relationship Diagram	32
14	System Architecture	36
15	Home Page	38
16	Projects Page	38
17	New Project Form	39
18	Invitation to Members and Product Owner	39
19	Projects page with prompt	40
20	List of all the Sprint	40
21	Sprint Backlog	41
22	Indicating no tasks for the Sprint (Scrum Master)	41
23	Add the planned tasks (Scrum Master)	42
24	Activities for task	42
25	Preside the daily scrum meeting (Scrum Master)	43
26	List of Scrum Meetings of the Sprint	43
27	Scrum Meeting Details	44

28	Scrum Board	44
29	Burndown Chart	45
30	Product Backlog	45
31	Sprint Review	46
32	Product Backlog	46
33	Add new user story	47
34	Edit user stories	47
35	List of Sprints (Product Owner)	48
36	New Sprint	48
37	Sprint Review by Product Owner	49
38	Show submenu for edit profile and change password	49
39	Edit profile	50
40	Change password	50
41	Manage Accounts	51

List of Tables

1	accounts Table	33
2	projects Table	33
3	members Table	33
4	userstories Table	33
5	sprints Table	34
6	tasks Table	34
7	activities Table	34
8	sprintreview Table	34
9	scrummeeting Table	35

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 ad-

dress 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 burn-down 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]

There's also a study that tested the ability of Scrum in terms of Global Software Development. There's 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 can't 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 there's 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 13 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]

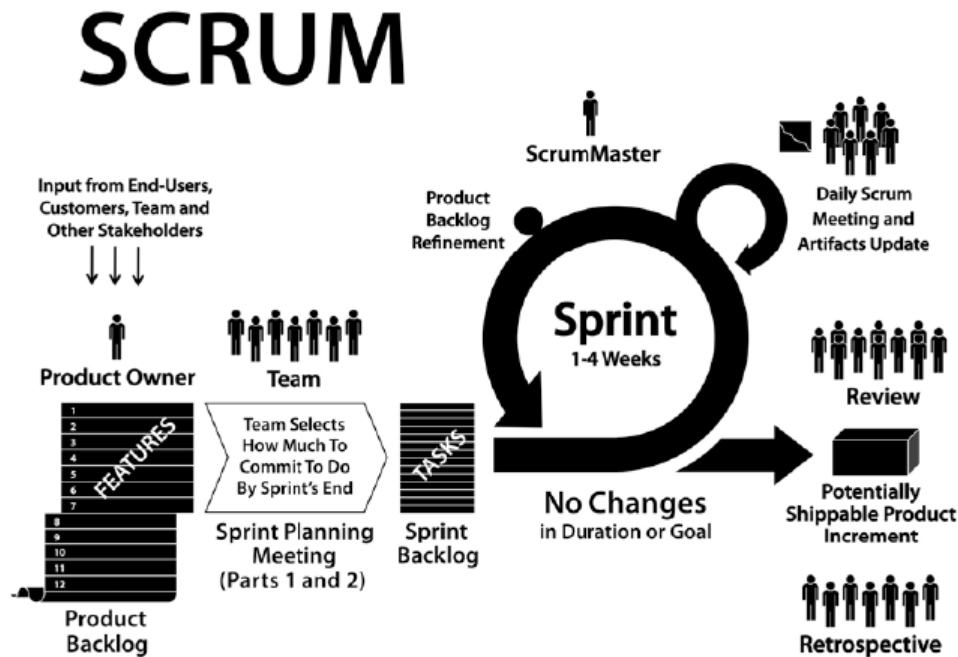


Figure 1: Scrum

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 Back-

log 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 there's 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 there's an ongoing sprint. Prioritizing user stories is the main responsibility of the Product Owner but there's no such technique defined in Scrum for prioritizing things. Usually it's 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, it's 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 they're 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 they're 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. It's 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 team's 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 there's 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.

Item	Details (wiki URL)	Priority	Estimate of Value	Initial Estimate of Effort	New Estimates of Effort Remaining as of Sprint...					
					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 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.

Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	New Estimates of Effort Remaining as of Day...					
				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.

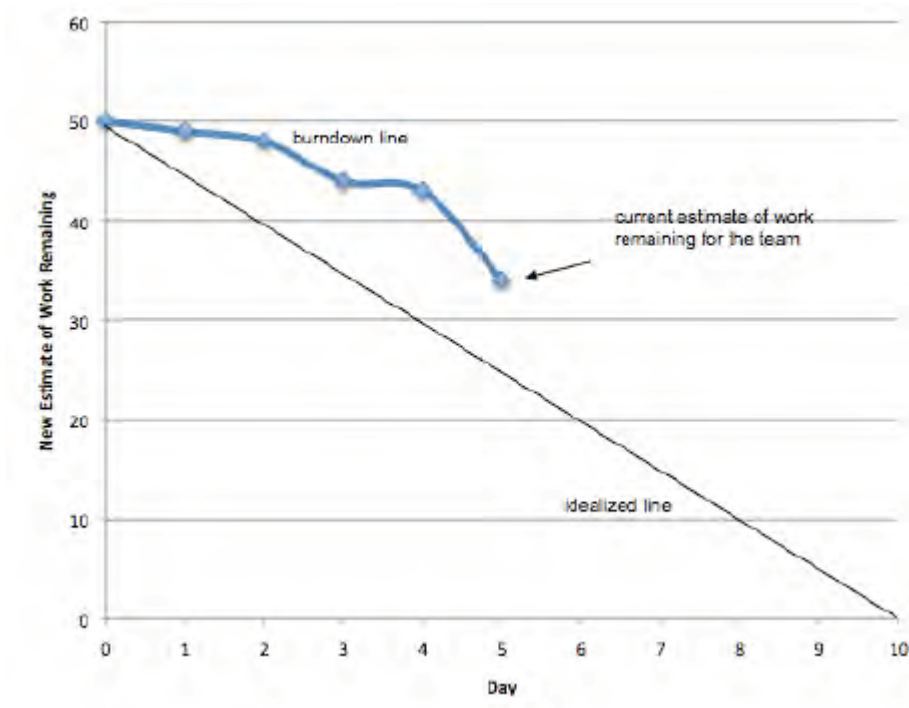


Figure 4: Burndown Chart

- **Scrum Board**

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

IV. Design and Implementation

1. Context Diagram

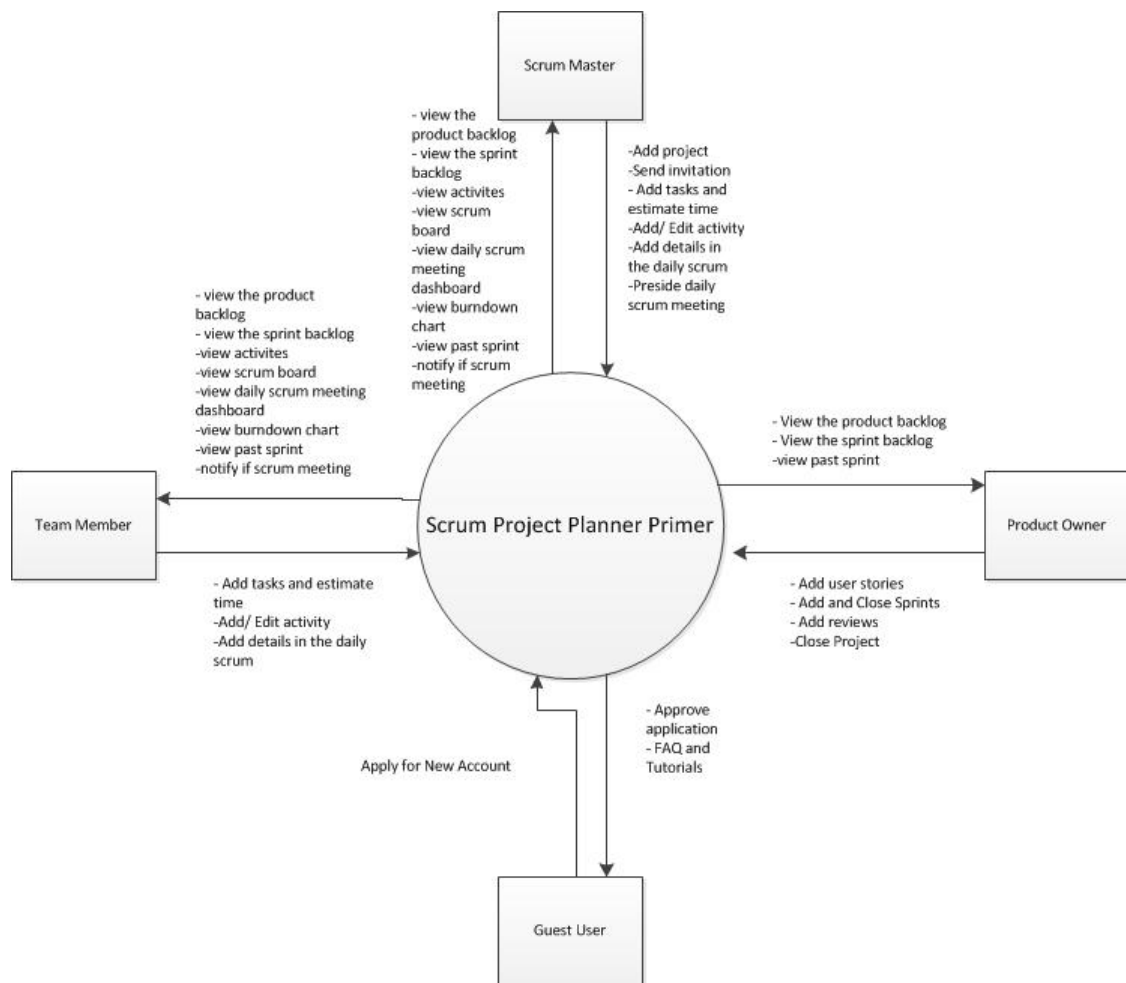


Figure 5: Context Diagram of Scrum Project Planner Primer

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.

velopment team can use the tool as a progress tracking and participate in Scrum activities.

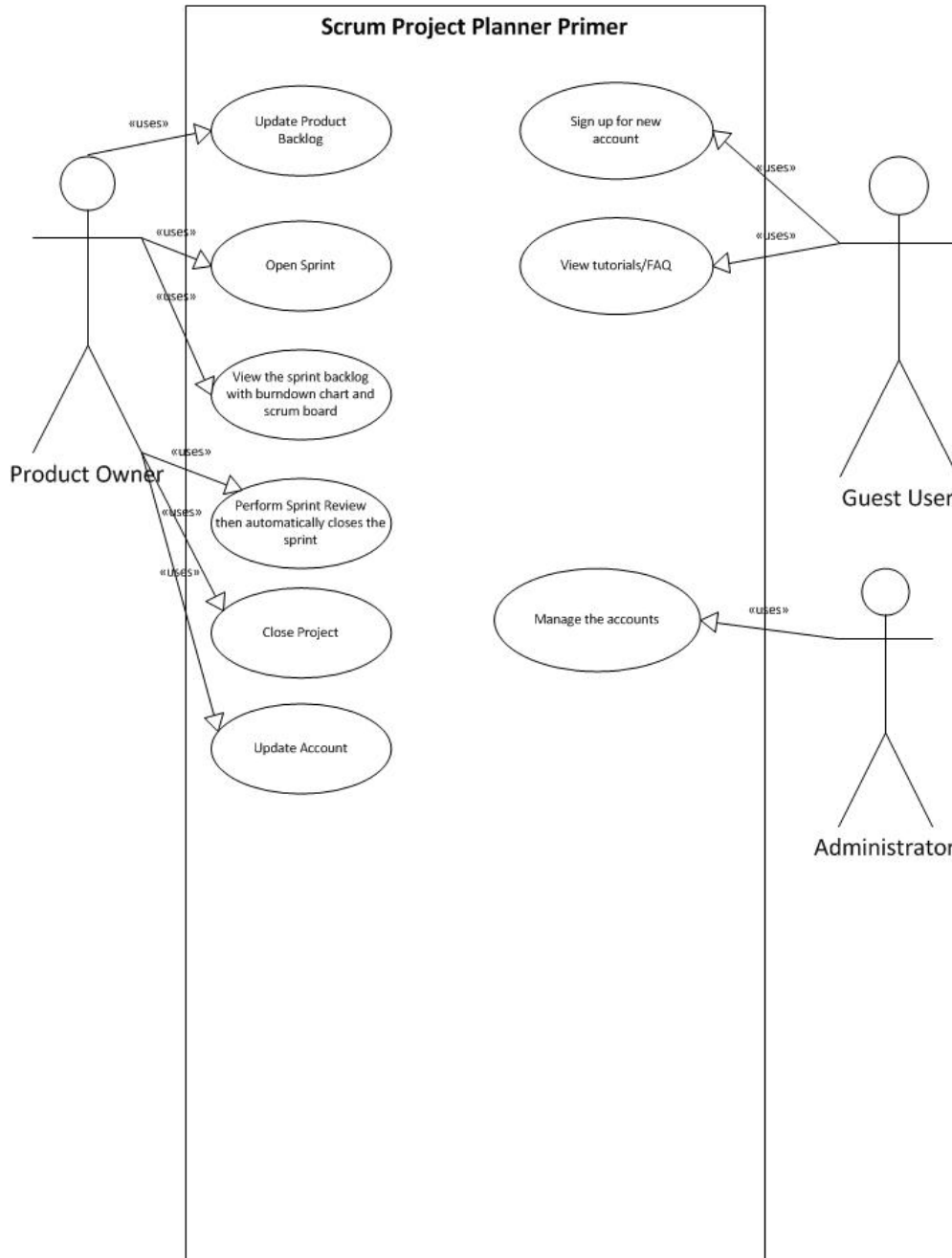


Figure 7: Use Case Diagram for Product Owner, Guest User and Administrator

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

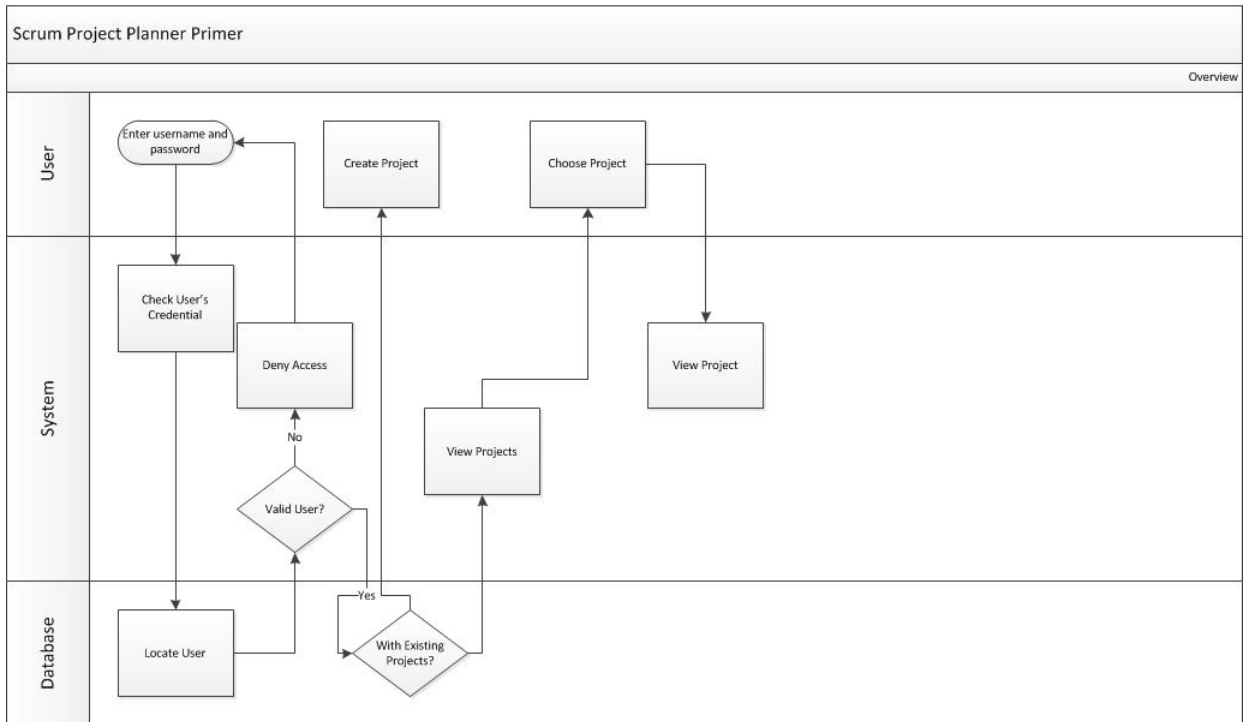


Figure 8: Overview

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

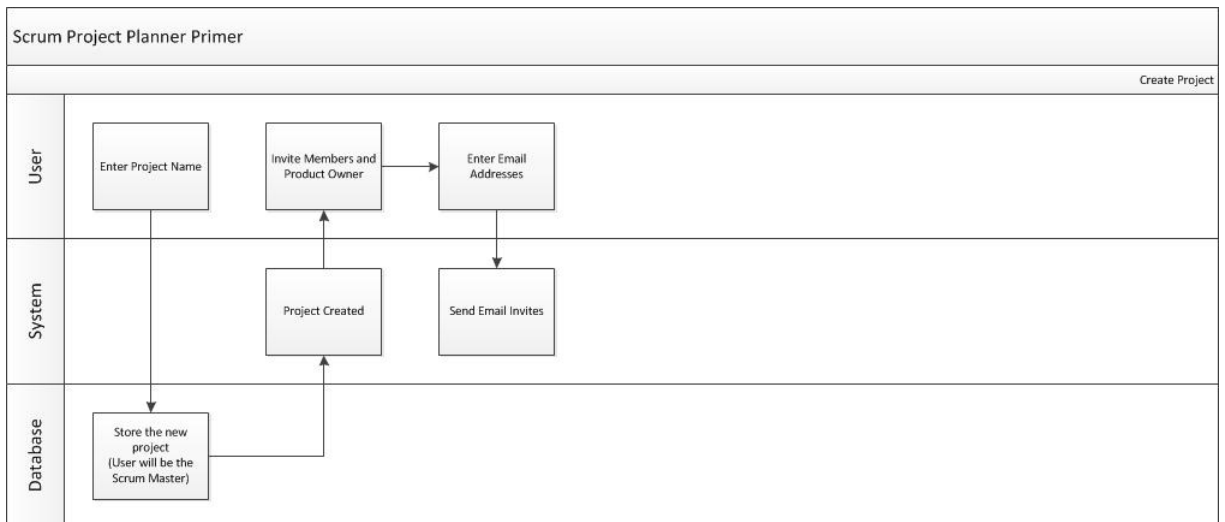


Figure 9: Create Project

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.

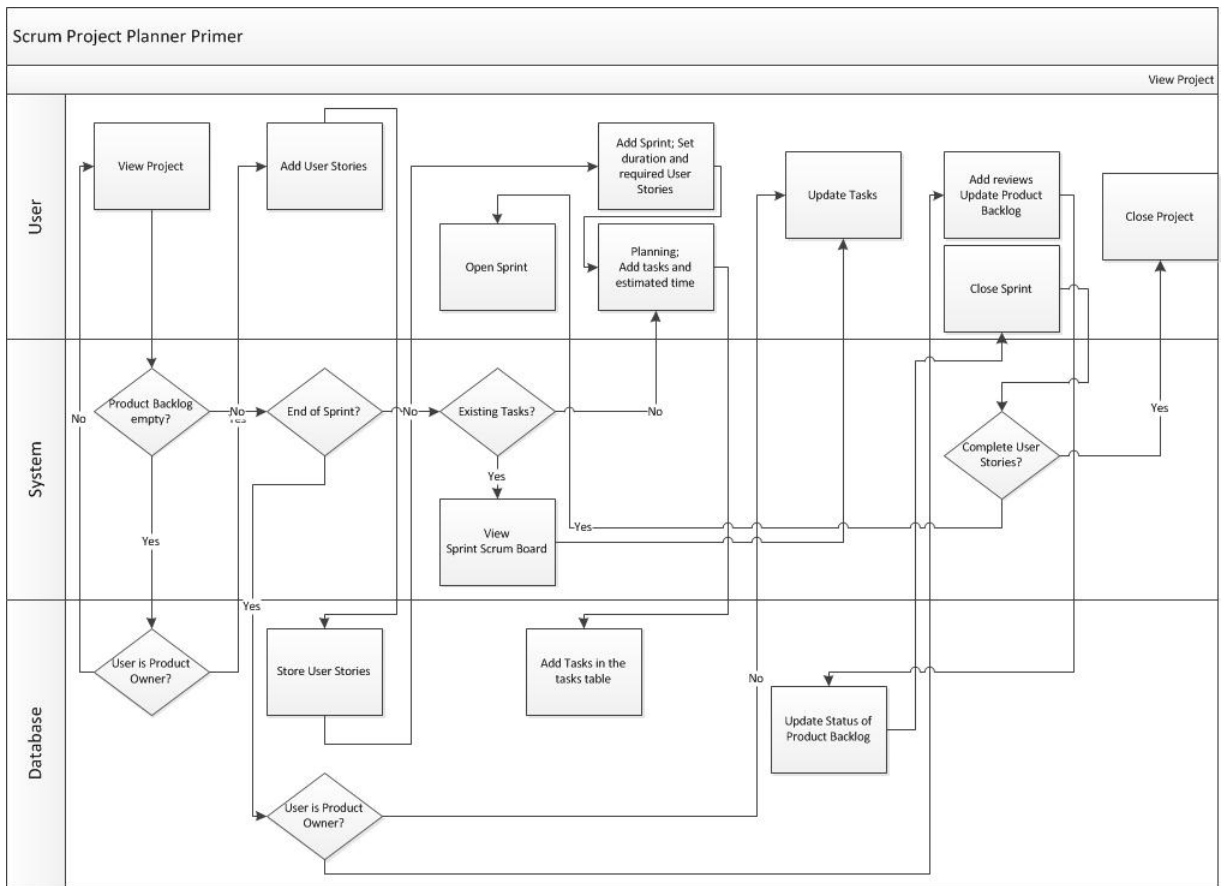


Figure 10: View Project

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.

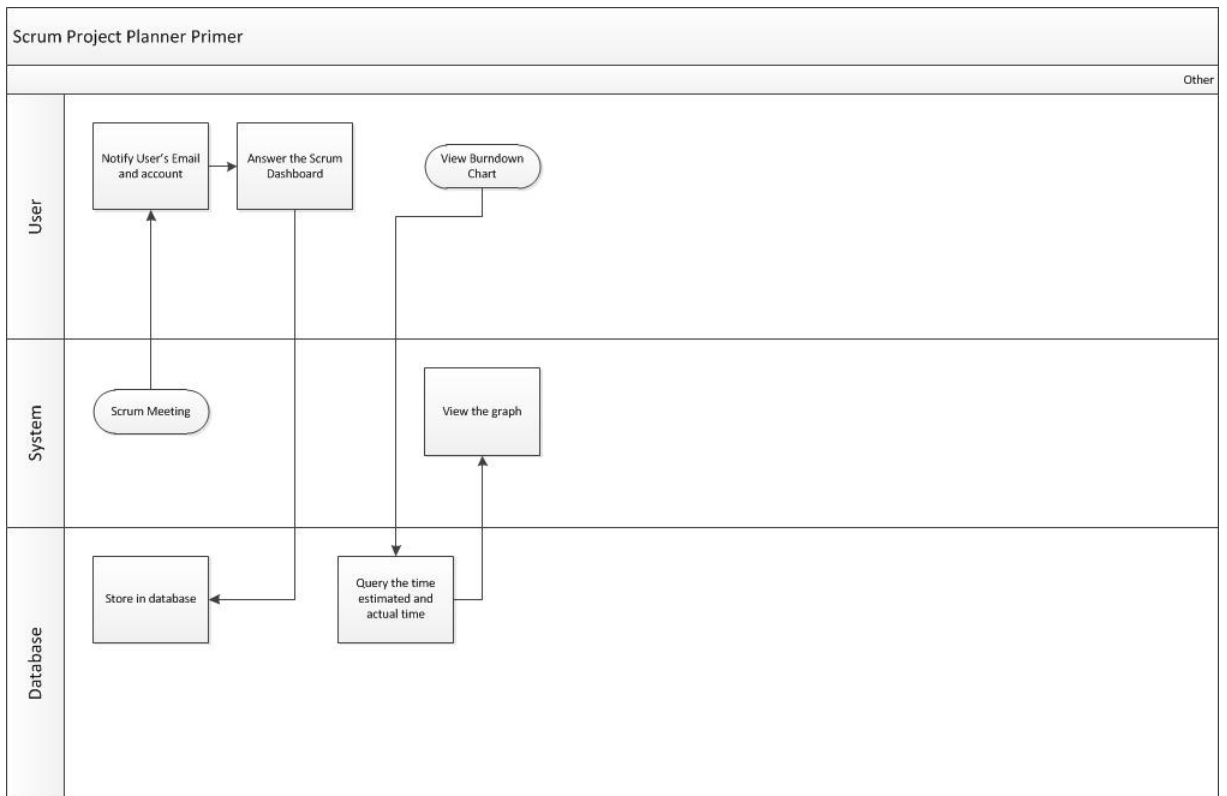


Figure 11: Daily Scrum Meeting and Burndown Chart

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.

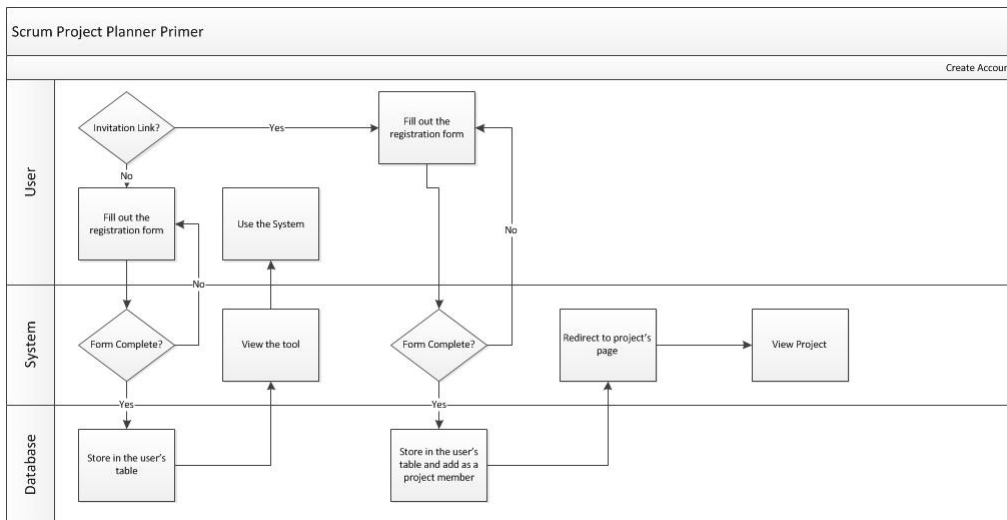


Figure 12: Create Account

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

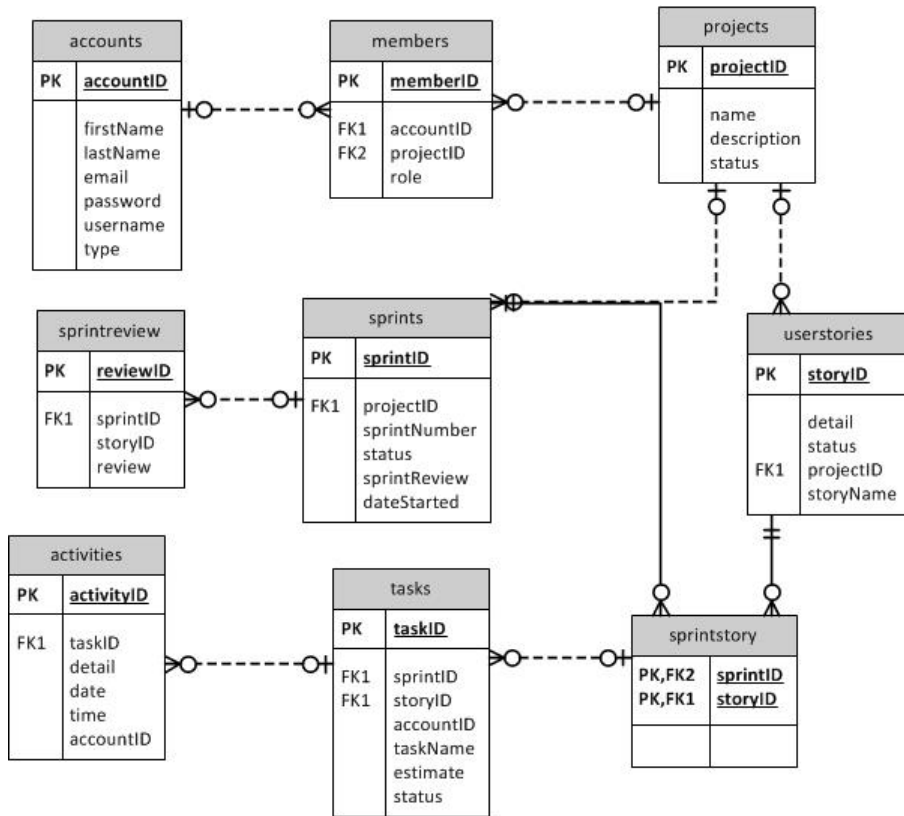


Figure 13: Entity Relationship Diagram

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

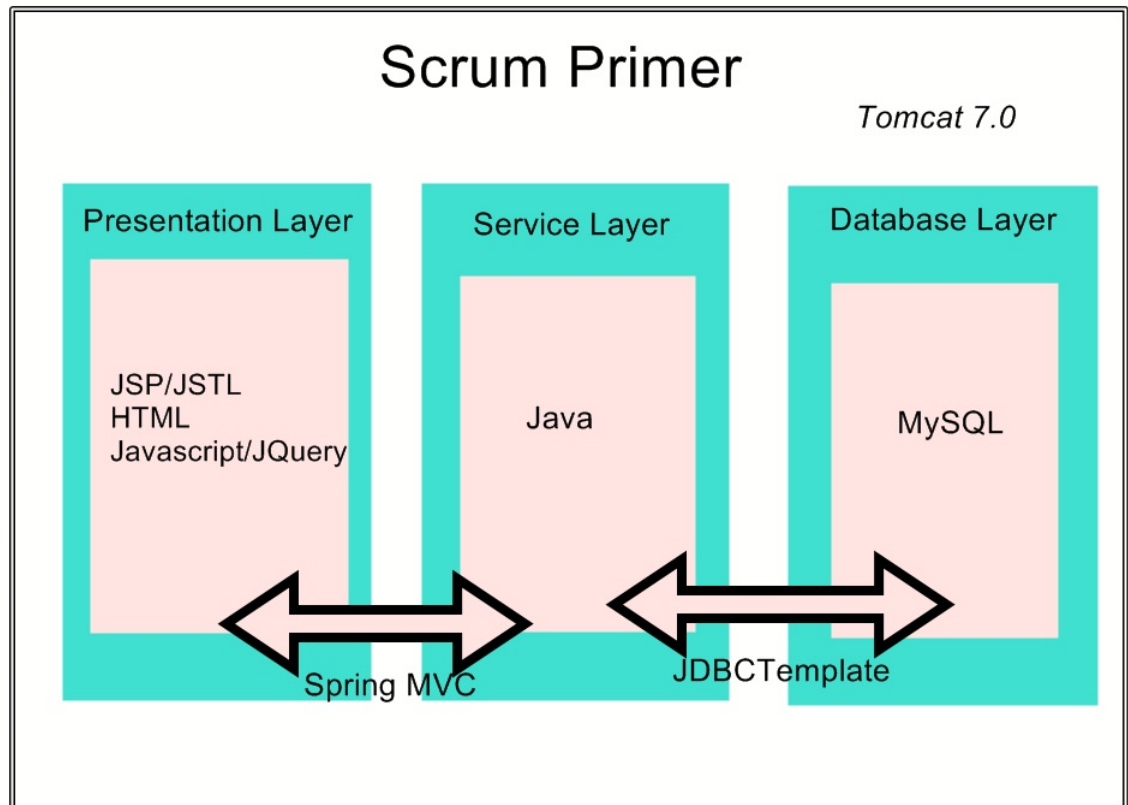


Figure 14: System Architecture

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

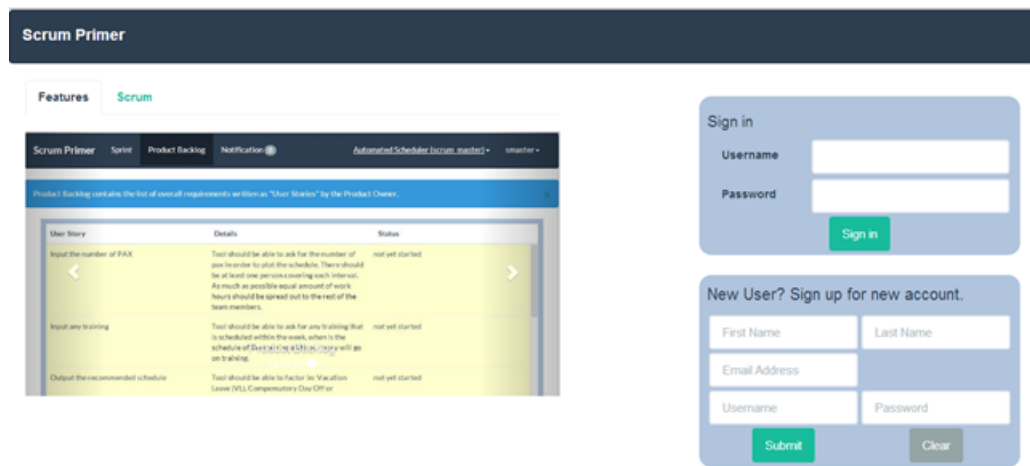


Figure 15: Home Page

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.

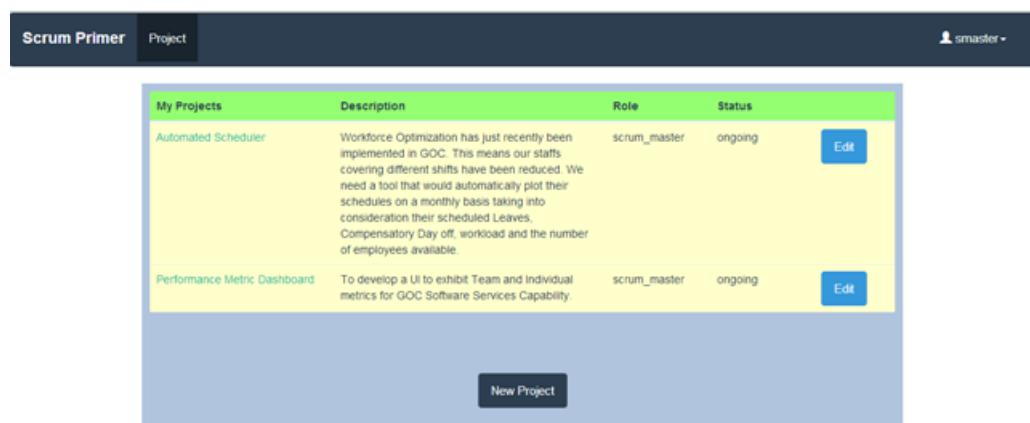


Figure 16: Projects Page

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.

The screenshot shows a web interface for creating a new project. At the top, there is a dark blue navigation bar with the text 'Scrum Primer Project' on the left and a user profile 'srmaster' on the right. The main content area has a light blue background and is titled 'New Project'. It contains three input fields: 'Project Name' with the value 'Project ABC', 'Project Description' with the value 'New project for primary kids.', and 'Number of Team Members' with the value '2'. Below these fields are two buttons: a green 'Submit' button and a grey 'Clear' button. At the bottom of the form, there is a yellow banner with the text: 'Note: The one who register the new project will automatically become the Scrum Master'.

Figure 17: New Project Form

The screenshot shows a web interface for inviting team members and a product owner. At the top, there is a dark blue navigation bar with the text 'Scrum Primer Project' on the left and a user profile 'srmaster' on the right. The main content area has a light blue background and is titled 'Invite team members'. It contains three input fields: 'First Name' with the value 'John', 'Last Name' with the value 'Lee', and 'Email Address' with the value 'johnlee@mail.com'. Below these fields are two buttons: a green 'Submit' button and a grey 'Clear' button. The 'Invite product owner' section has four input fields: 'First Name' with the value 'Product', 'Last Name' with the value 'Owner', 'Email Address' with the value 'powner@mail.com', and 'Outside Client?' with a checked checkbox. Below these fields are two buttons: a green 'Submit' button and a grey 'Clear' button.

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.

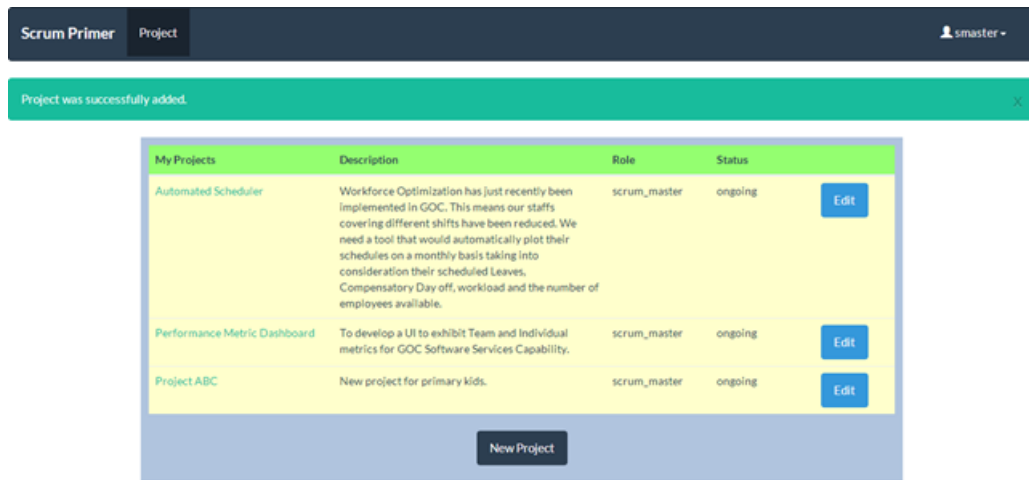


Figure 19: Projects page with prompt

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

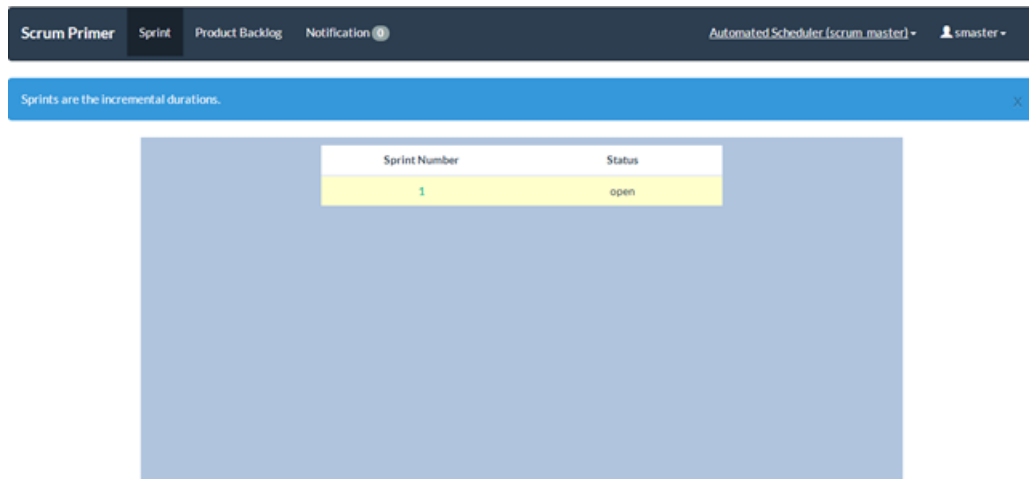





Figure 20: List of all the Sprint

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.

Scrum Primer Sprint Product Backlog Notification  Automated Scheduler (scrum_master) -  smaster -

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

Sprint 1

Remaining days before Sprint Review: 5

[Scrum Board](#)

[Scrum Meeting](#)



[Sprint Backlog](#)


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	Kla	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

[Add New Task](#)

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.

Scrum Primer Sprint Product Backlog Notification  Project 5 (scrum_master) -  scrummaster -

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


Sprint 1

Remaining days before Sprint Review: 10

[Scrum Board](#)

[Scrum Meeting](#)

[Sprint Backlog](#)

No tasks for the sprint 

[Add Task](#)

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

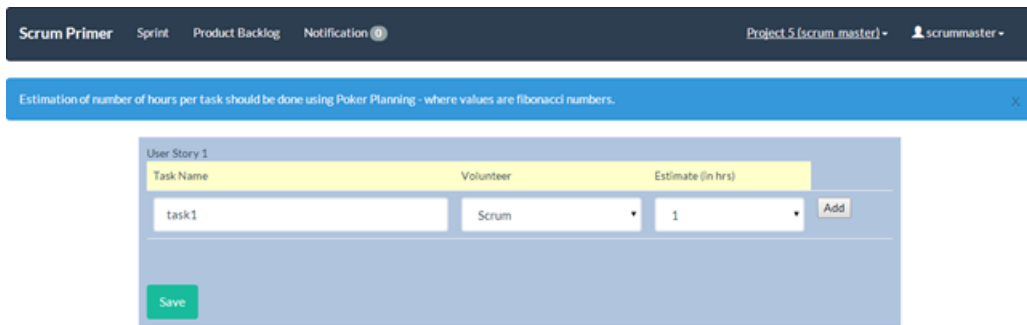


Figure 23: Add the planned tasks (Scrum Master)

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.

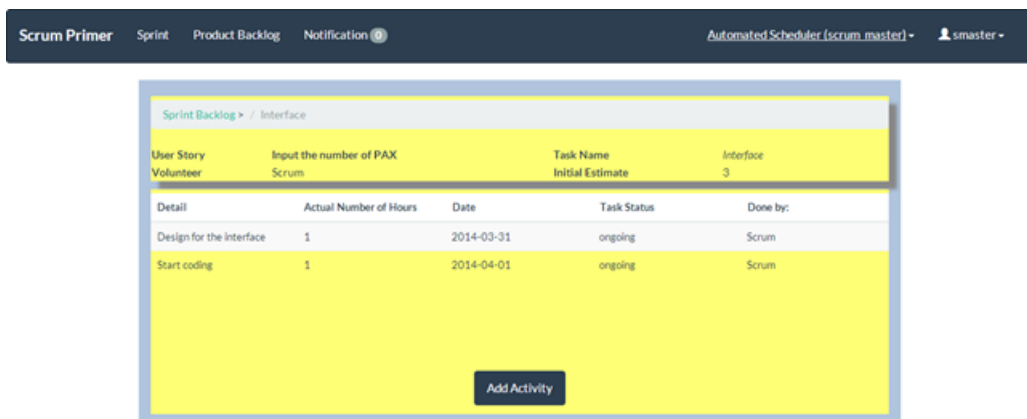


Figure 24: Activities for task

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

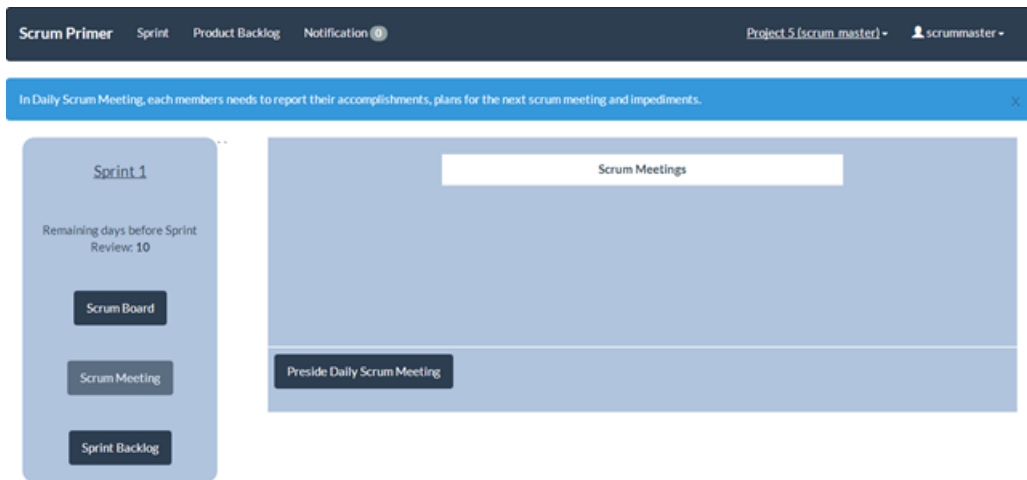


Figure 25: Preside the daily scrum meeting (Scrum Master)

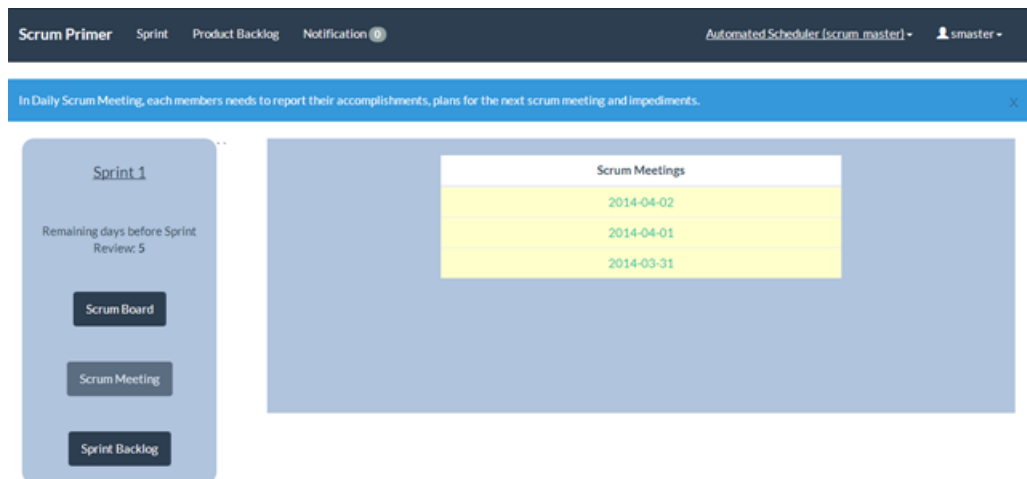


Figure 26: List of Scrum Meetings of the Sprint

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.

Member	Accomplishments	Plans	Impediments
Steven	Start coding the interface	Finish coding	Incompatibility of jstl and javascript
Kia	None	none	none
Jenisle	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"

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

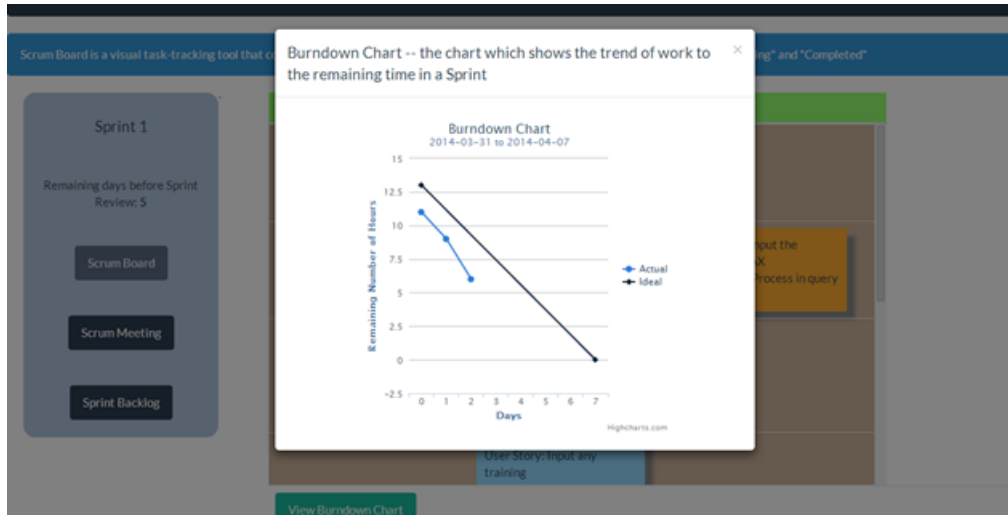


Figure 29: Burndown Chart

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.

Scrum Primer Sprint Product Backlog Notification ⓘ Automated Scheduler (scrum_master) smaster

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.

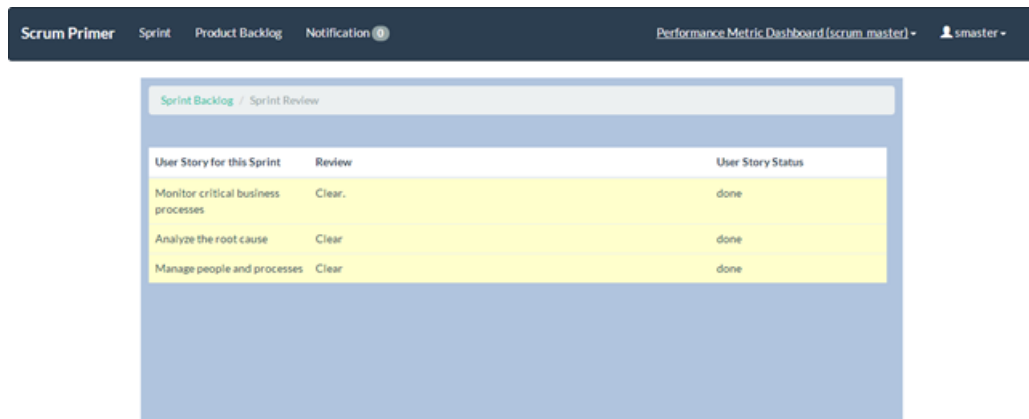


Figure 31: Sprint Review

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

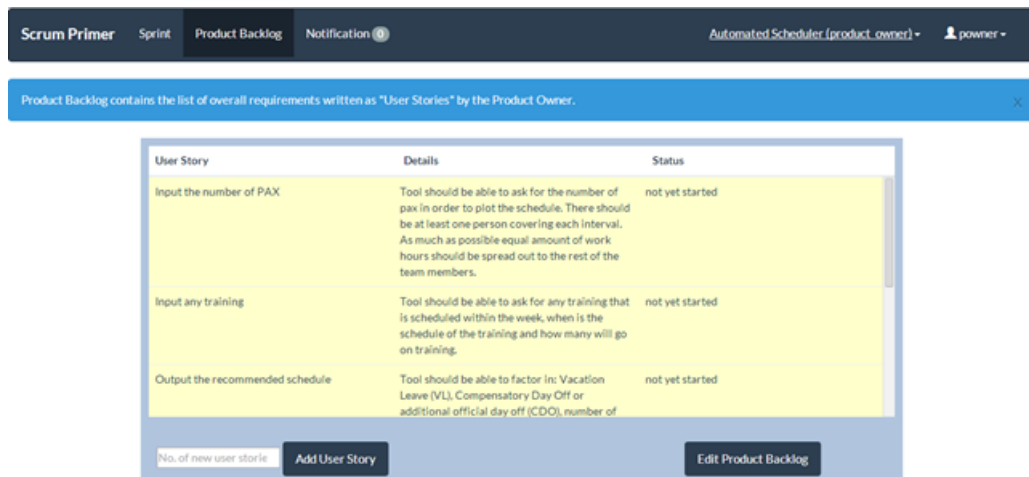


Figure 32: Product Backlog

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

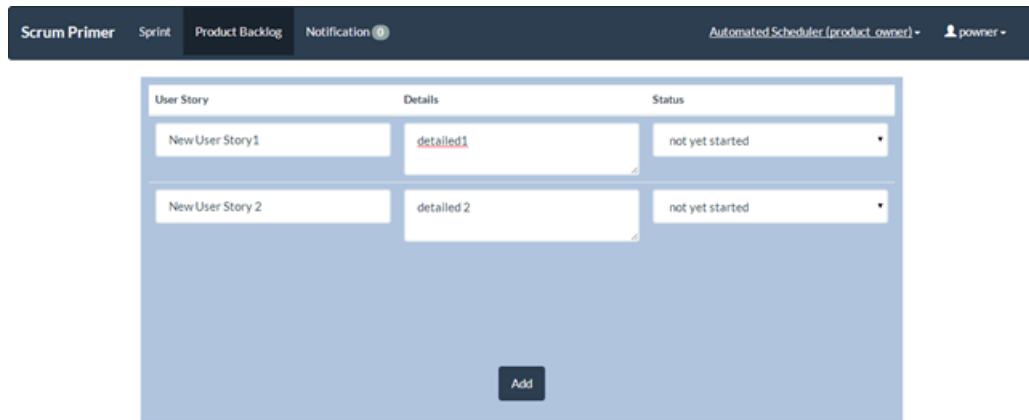


Figure 33: Add new user story

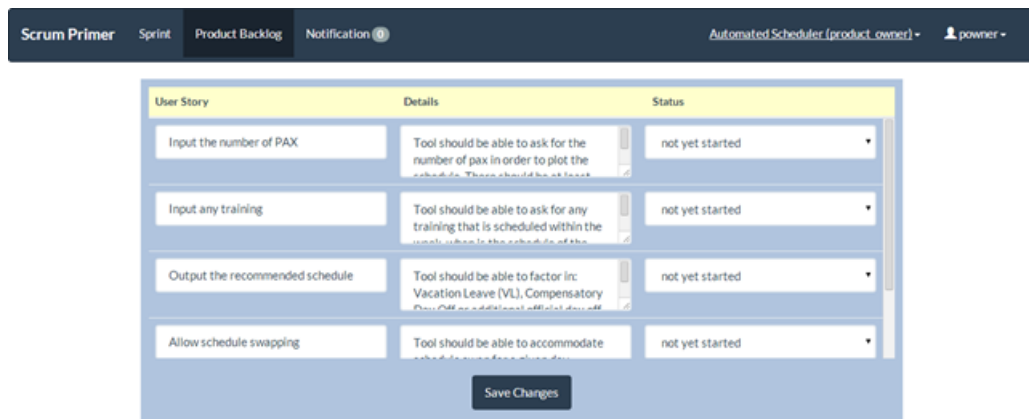


Figure 34: Edit user stories

The product owner can add user stories or edit them.

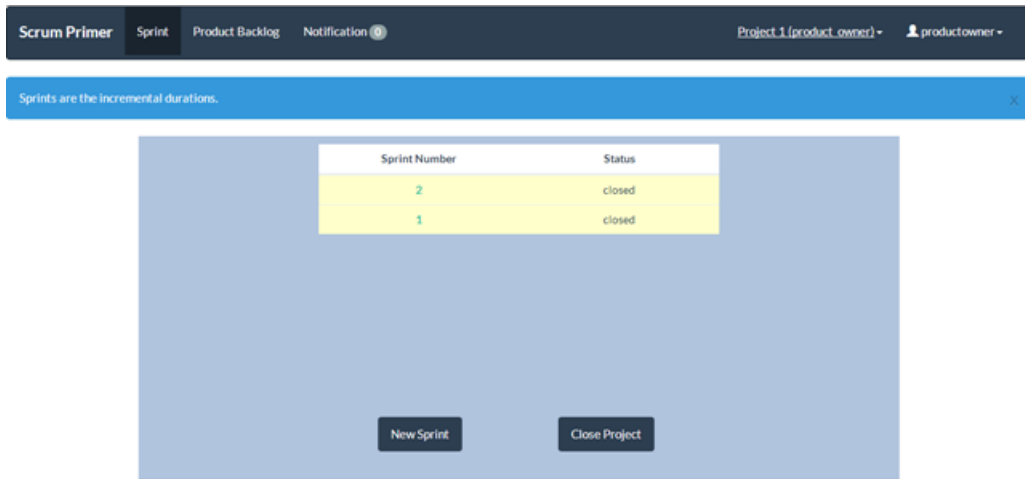


Figure 35: List of Sprints (Product Owner)

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

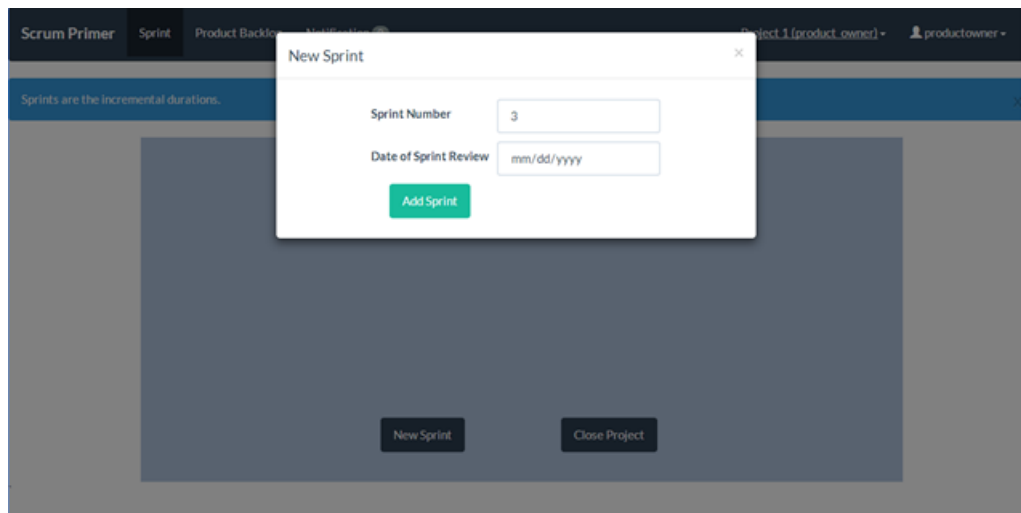


Figure 36: New Sprint

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

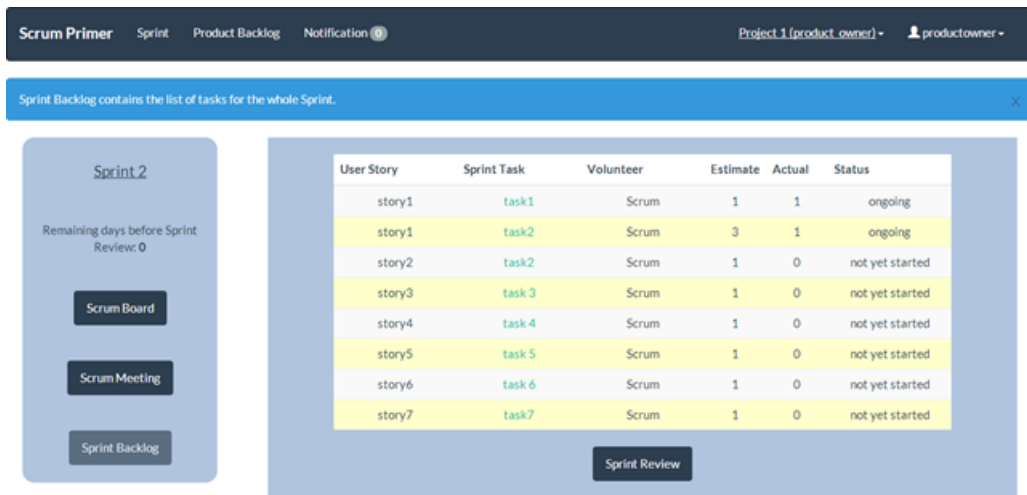


Figure 37: Sprint Review by Product Owner

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

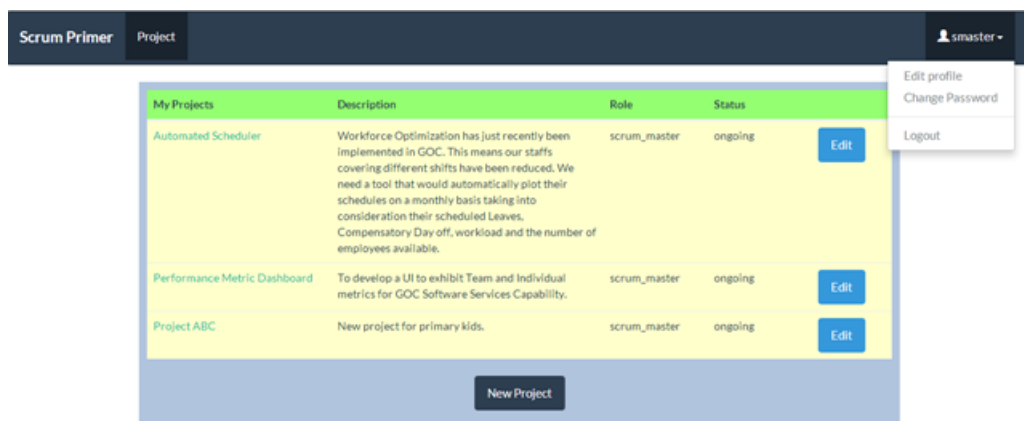


Figure 38: Show submenu for edit profile and change password

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

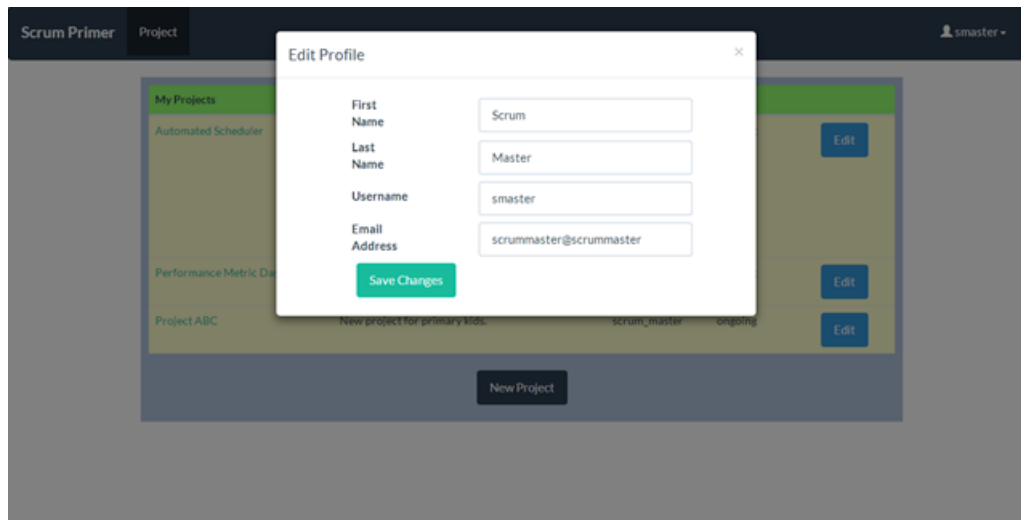


Figure 39: Edit profile

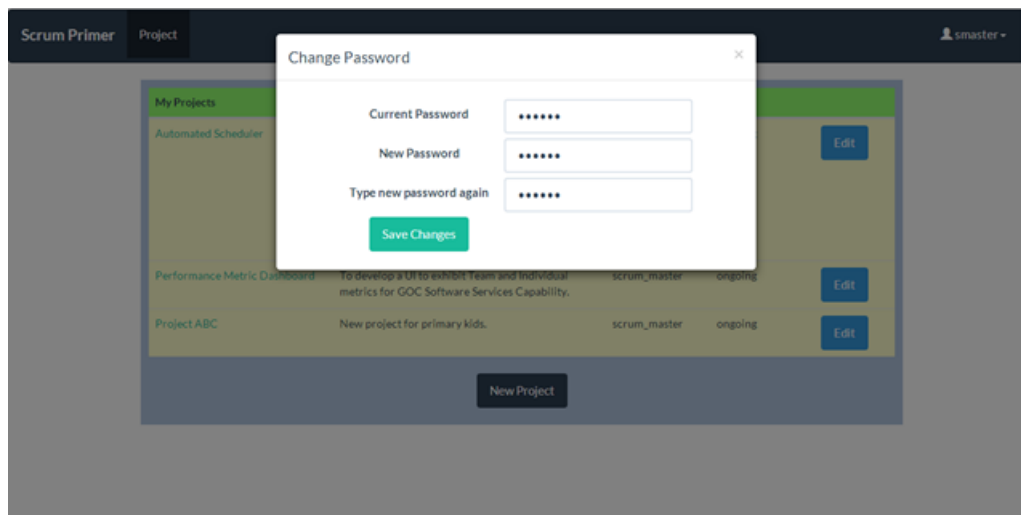


Figure 40: Change password

2. Administrator

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

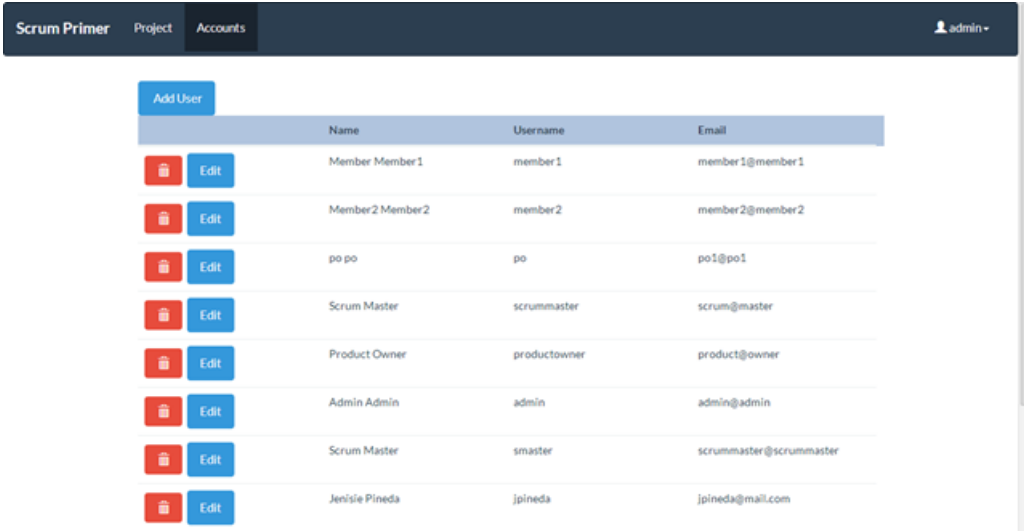


Figure 41: Manage Accounts

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 pertain 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.capttechconsulting.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 practical 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

```
1  /**
2   * Activity.java
3   *
4   * Model for the object Activity
5   */
6  package com.scrumprimer.model;
7
8  public class Activity {
9      int activityID;
10     String detail;
11     int taskID;
12     String date;
13     int time;
14     String taskStatus;
15     String doer;
16     public String getDoer() {
17         return doer;
18     }
19     public void setDoer(String doer) {
20         this.doer = doer;
21     }
22     public int getAccountID() {
23         return accountID;
24     }
25     public void setAccountID(int accountID) {
26         this.accountID = accountID;
27     }
28     int accountID;
29     public int getActivityID() {
30         return activityID;
31     }
32     public void setActivityID(int activityID) {
33         this.activityID = activityID;
34     }
35     public String getDetail() {
36         return detail;
37     }
38     public void setDetail(String detail) {
39         this.detail = detail;
40     }
41     public int getTaskID() {
42         return taskID;
43     }
44     public void setTaskID(int taskID) {
45         this.taskID = taskID;
46     }
47     public String getDate() {
48         return date;
49     }
50     public void setDate(String date) {
51         this.date = date;
52     }
53     public int getTime() {
54         return time;
55     }
56     public void setTime(int time) {
57         this.time = time;
58     }
59     public String getTaskStatus() {
60         return taskStatus;
61     }
62     public void setTaskStatus(String taskStatus) {
63         this.taskStatus = taskStatus;
64     }
65 }
66
67 }
```

Listing 2: ActivityRowMapper.java

```
1  /**
2   * ActivityRowMapper.java
3   *
4   * Row mapper for activities table
5   */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
```

```

13
14 public class ActivityRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Activity activity = new Activity();
20         activity.setActivityID(rs.getInt("activityID"));
21         activity.setDate(rs.getString("date"));
22         activity.setDetail(rs.getString("detail"));
23         activity.setTaskID(rs.getInt("taskID"));
24         activity.setTime(rs.getInt("time"));
25         activity.setAccountID(rs.getInt("accountID"));
26
27         return activity;
28     }
29 }
30

```

Listing 3: Dummy.java

```

1 /**
2  * Dummy.java
3  *
4  * Model for object Dummy -- for invite model
5  */
6 package com.scrumprimer.model;
7
8 public class Dummy {
9     String firstName;
10    String lastName;
11    String role;
12    String email;
13    int projectID;
14
15    public String getFirstName() {
16        return firstName;
17    }
18    public void setFirstName(String firstName) {
19        this.firstName = firstName;
20    }
21    public String getLastName() {
22        return lastName;
23    }
24    public void setLastName(String lastName) {
25        this.lastName = lastName;
26    }
27    public String getRole() {
28        return role;
29    }
30    public void setRole(String role) {
31        this.role = role;
32    }
33    public String getEmail() {
34        return email;
35    }
36    public void setEmail(String email) {
37        this.email = email;
38    }
39    public int getProjectID() {
40        return projectID;
41    }
42    public void setProjectID(int projectID) {
43        this.projectID = projectID;
44    }
45 }
46

```

Listing 4: DummyRowMapper.java

```

1 /**
2  * DummyRowMapper.java
3  *
4  * Row Mapper for invites table
5  */
6
7 package com.scrumprimer.model;
8
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class DummyRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Dummy dummy = new Dummy();
20         dummy.setEmail(rs.getString("email"));
21         dummy.setFirstName(rs.getString("firstName"));
22         dummy.setLastName(rs.getString("lastName"));
23         dummy.setProjectID(rs.getInt("projectID"));
24         dummy.setRole(rs.getString("role"));
25         return dummy;
26     }
27 }
28

```

```
27
28 }
```

Listing 5: Meeting.java

```
1  /**
2   * Meeting.java
3   *
4   * Model for object Meeting – contains the scrum meeting details
5   */
6
7  package com.scrumprimer.model;
8
9  public class Meeting {
10     String date;
11     int meetingID;
12     int sprintID;
13     String accomplishment;
14     String plan;
15     String impediment;
16     int accountID;
17     String member;
18     public String getMember() {
19         return member;
20     }
21     public void setMember(String member) {
22         this.member = member;
23     }
24     public int getAccountID() {
25         return accountID;
26     }
27     public void setAccountID(int accountID) {
28         this.accountID = accountID;
29     }
30     public String getDate() {
31         return date;
32     }
33     public void setDate(String date) {
34         this.date = date;
35     }
36     public int getMeetingID() {
37         return meetingID;
38     }
39     public void setMeetingID(int meetingID) {
40         this.meetingID = meetingID;
41     }
42     public int getSprintID() {
43         return sprintID;
44     }
45     public void setSprintID(int sprintID) {
46         this.sprintID = sprintID;
47     }
48     public String getAccomplishment() {
49         return accomplishment;
50     }
51     public void setAccomplishment(String accomplishment) {
52         this.accomplishment = accomplishment;
53     }
54     public String getPlan() {
55         return plan;
56     }
57     public void setPlan(String plan) {
58         this.plan = plan;
59     }
60     public String getImpediment() {
61         return impediment;
62     }
63     public void setImpediment(String impediment) {
64         this.impediment = impediment;
65     }
66
67
68 }
69 }
```

Listing 6: Meetingform.java

```
1  /**
2   * MeetingForm.java
3   *
4   * Model for multiple Meeting objects
5   */
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9
10 public class MeetingForm {
11     ArrayList<Review> reviews;
12
13     public ArrayList<Review> getReviews() {
14         return reviews;
15     }
16
17     public void setReviews(ArrayList<Review> reviews) {
18         this.reviews = reviews;
19     }
20 }
```

```
20
21 }
```

Listing 7: MeetingRowMapper.java

```
1  /**
2   * MeetingRowMapper.java
3   *
4   * Row Mapper for meetings table
5   */
6  package com.scrumprimer.model;
7
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10
11 import org.springframework.jdbc.core.RowMapper;
12
13 public class MeetingRowMapper implements RowMapper {
14
15     @Override
16     public Object mapRow(ResultSet rs, int row) throws SQLException {
17         // TODO Auto-generated method stub
18         Meeting meeting = new Meeting();
19         meeting.setAccomplishment(rs.getString("accomplishment"));
20         meeting.setImpediment(rs.getString("impediment"));
21         meeting.setMeetingID(rs.getInt("meetingID"));
22         meeting.setPlan(rs.getString("plan"));
23         meeting.setSprintID(rs.getInt("sprintID"));
24         meeting.setAccountID(rs.getInt("accountID"));
25         meeting.setDate(rs.getString("date"));
26         return meeting;
27     }
28
29 }
30 }
```

Listing 8: Member.java

```
1  /**
2   * Member.java
3   *
4   * Model for object Member -- project's member
5   */
6
7  package com.scrumprimer.model;
8
9  public class Member {
10     int memberID;
11     int projectID;
12     String projectName;
13     String role;
14     int accountID;
15     String name;
16     public String getName() {
17         return name;
18     }
19     public void setName(String name) {
20         this.name = name;
21     }
22     public int getAccountID() {
23         return accountID;
24     }
25     public void setAccountID(int accountID) {
26         this.accountID = accountID;
27     }
28     public int getMemberID() {
29         return memberID;
30     }
31     public void setMemberID(int memberID) {
32         this.memberID = memberID;
33     }
34
35     public int getProjectID() {
36         return projectID;
37     }
38     public void setProjectID(int projectID) {
39         this.projectID = projectID;
40     }
41     public String getProjectName() {
42         return projectName;
43     }
44     public void setProjectName(String projectName) {
45         this.projectName = projectName;
46     }
47     public String getRole() {
48         return role;
49     }
50     public void setRole(String role) {
51         this.role = role;
52     }
53 }
```

Listing 9: Notification.java

```
1
```

```

2  /**
3   * Notification.java
4   *
5   * Model for object Notification
6   */
7
8  package com.scrumprimer.model;
9
10 public class Notification {
11     int notifID;
12     int accountID;
13     int projectID;
14     String type;
15     String date;
16     String newMember;
17     int sprintID;
18     public String getNewMember() {
19         return newMember;
20     }
21     public void setNewMember(String newMember) {
22         this.newMember = newMember;
23     }
24     public String getDate() {
25         return date;
26     }
27     public int getSprintID() {
28         return sprintID;
29     }
30     public void setSprintID(int sprintID) {
31         this.sprintID = sprintID;
32     }
33     public void setDate(String date) {
34         this.date = date;
35     }
36     String status;
37     public String getStatus() {
38         return status;
39     }
40     public void setStatus(String status) {
41         this.status = status;
42     }
43     public int getNotifID() {
44         return notifID;
45     }
46     public void setNotifID(int notifID) {
47         this.notifID = notifID;
48     }
49     public int getAccountID() {
50         return accountID;
51     }
52     public void setAccountID(int accountID) {
53         this.accountID = accountID;
54     }
55     public int getProjectID() {
56         return projectID;
57     }
58     public void setProjectID(int projectID) {
59         this.projectID = projectID;
60     }
61     public String getType() {
62         return type;
63     }
64     public void setType(String type) {
65         this.type = type;
66     }
67 }
68 }

```

Listing 10: NotificationDAO.java

```

1  /**
2   * NotificationDAO.java
3   *
4   */
5
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9  import java.util.List;
10
11 import javax.sql.DataSource;
12
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.jdbc.core.JdbcTemplate;
15
16 import com.scrumprimer.utility.NotificationUtility;
17
18 public class NotificationDAO {
19
20     @Autowired
21     private DataSource dataSource;
22     private JdbcTemplate jdbcTemplateObject;
23
24     public void insertNotification(Notification notif, int receiverID){
25         jdbcTemplateObject = new JdbcTemplate(dataSource);
26         String query = "INSERT INTO notifications (type, accountID, projectID,
27             receiverID, sprintID) VALUES (?, ?, ?, ?, ?)";

```



```

27         jdbcTemplateObject.update(query, new Object[]{ notif.getType(), notif.
                getAccountID(), notif.getProjectID(), receiverID, notif.getSprintID
                ()});
28     }
29
30     @SuppressWarnings("unchecked")
31     public List<Notification> getNotification(int projectID, int receiverID){
32         jdbcTemplateObject = new JdbcTemplate(dataSource);
33         String query = "UPDATE notifications SET status = ? WHERE projectID = ?
                and receiverID = ?";
34         jdbcTemplateObject.update(query, new Object[]{"read", projectID,
                receiverID});
35         query = "SELECT * FROM notifications WHERE status = ? and projectID = ?
                and receiverID = ? ORDER BY notifID DESC";
36         ArrayList<Notification> notifications = new ArrayList<Notification>();
37         notifications = (ArrayList<Notification>) jdbcTemplateObject.query(query
                , new Object[]{"read", projectID, receiverID}, new
                NotificationRowMapper());
38         return notifications;
39     }
40
41     public int getNumOfNewNotifs(int projectID, int receiverID){
42         jdbcTemplateObject = new JdbcTemplate(dataSource);
43         String query = "SELECT COUNT(*) FROM notifications WHERE status = ? and
                projectID = ? AND receiverID = ?";
44         @SuppressWarnings("deprecation")
45         int row = jdbcTemplateObject.queryForInt(query, new Object[]{"unread",
                projectID, receiverID});
46         return row;
47     }
48
49     public boolean isSprintReview(int sprintID){
50         jdbcTemplateObject = new JdbcTemplate(dataSource);
51         String query = "SELECT COUNT(*) FROM notifications WHERE type = ? and
                sprintID = ?";
52         int row = jdbcTemplateObject.queryForInt(query, new Object[]{"
                NotificationUtility.SPRINT_REVIEW, sprintID});
53         if(row == 0){
54             return false;
55         }
56         return true;
57     }
58 }

```

Listing 11: NotificationRowMapper.java

```

1  /**
2   * NotificationRowMapper.java
3   *
4   * Row mapper for notification table
5   */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class NotificationRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Notification notif = new Notification();
20         notif.setAccountID(rs.getInt("accountID"));
21         notif.setNotifID(rs.getInt("notifID"));
22         notif.setProjectID(rs.getInt("projectID"));
23         notif.setType(rs.getString("type"));
24         notif.setDate(rs.getString("date"));
25         notif.setStatus(rs.getString("status"));
26         notif.setSprintID(rs.getInt("sprintID"));
27         return notif;
28     }
29 }
30 }

```

Listing 12: Project.java

```

1  /**
2   * Project.java
3   *
4   * Model for object Project
5   *
6   */
7
8  package com.scrumprimer.model;
9
10 public class Project {
11
12     int projectID;
13     String projectName;
14     String projectDescription;
15     int numberOfTM;
16     int numberOfPO;
17     String status;

```

```

18     public String getStatus() {
19         return status;
20     }
21     public void setStatus(String status) {
22         this.status = status;
23     }
24     public int getNumberOfTM() {
25         return numberOfTM;
26     }
27     public void setNumberOfTM(int numberOfTM) {
28         this.numberOfTM = numberOfTM;
29     }
30     public int getNumberOfPO() {
31         return numberOfPO;
32     }
33     public void setNumberOfPO(int numberOfPO) {
34         this.numberOfPO = numberOfPO;
35     }
36     public int getProjectID() {
37         return projectID;
38     }
39     public void setProjectID(int projectID) {
40         this.projectID = projectID;
41     }
42     public String getProjectName() {
43         return projectName;
44     }
45     public void setProjectName(String projectName) {
46         this.projectName = projectName;
47     }
48     public String getProjectDescription() {
49         return projectDescription;
50     }
51     public void setProjectDescription(String projectDescription) {
52         this.projectDescription = projectDescription;
53     }
54 }

```

Listing 13: ProjectDAO.java

```

1  /**
2   * ProjectDAO.java
3   *
4   */
5
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9  import java.util.List;
10 import java.util.Map;
11
12 import javax.sql.DataSource;
13
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.jdbc.core.JdbcTemplate;
16 import org.springframework.jdbc.support.rowset.SqlRowSet;
17 import org.springframework.security.authentication.encoding.PasswordEncoder;
18
19 import com.mysql.jdbc.ResultSet;
20 import com.scrumprimer.model.User;
21 import com.scrumprimer.model.UserRowMapper;
22 import com.scrumprimer.utility.DateHandler;
23 import com.scrumprimer.utility.StatusUtility;
24
25
26 public class ProjectDAO {
27
28     @Autowired
29     private DataSource dataSource;
30     private JdbcTemplate jdbcTemplateObject;
31     @Autowired
32     private PasswordEncoder passwordEncoder;
33     @Autowired
34     private UserDao userDao;
35
36     public List<Member> getProject(int accountID){
37         jdbcTemplateObject = new JdbcTemplate(dataSource);
38         String query = "SELECT * FROM members WHERE accountID = ?";
39         List<Member> members = jdbcTemplateObject.query(query, new Object
40             []{ accountID }, new ProjectRowMapper());
41         return members;
42     }
43
44     public List<UserStory> getUserStory(int projectID){
45         jdbcTemplateObject = new JdbcTemplate(dataSource);
46         String query = "SELECT * FROM userStories WHERE projectID = ?";
47         List<UserStory> userStories = jdbcTemplateObject.query(query, new Object
48             []{ projectID }, new UserStoryMapper());
49         return userStories;
50     }
51
52     public List<UserStory> getOngoingUserStory(int projectID){
53         jdbcTemplateObject = new JdbcTemplate(dataSource);
54         String query = "SELECT * FROM userStories WHERE projectID = ? and (
55             status = ? OR status = ?)";
56         List<UserStory> userStories = jdbcTemplateObject.query(query, new Object
57             []{ projectID, StatusUtility.ONGOING, StatusUtility.NOT_STARTED },
58             new UserStoryMapper());

```

```

54         return userStories;
55     }
56
57     public boolean isUserStoryExisting(int projectID){
58         jdbcTemplateObject = new JdbcTemplate(dataSource);
59         String query = "SELECT COUNT(*) FROM userStories WHERE projectID = ?";
60         int row = jdbcTemplateObject.queryForInt(query, new Object[] { projectID }
61             );
62         if(row == 0)
63             return false;
64     }
65     else
66         return true;
67 }
68
69     public boolean isTaskExisting(int sprintID){
70         jdbcTemplateObject = new JdbcTemplate(dataSource);
71         String query = "SELECT COUNT(*) FROM tasks WHERE sprintID = ?";
72         int row = jdbcTemplateObject.queryForInt(query, new Object[] { sprintID }
73             );
74         if(row == 0)
75             return false;
76     }
77     else
78         return true;
79 }
80
81     public void updateUserStories(List<UserStory> userStories){
82         jdbcTemplateObject = new JdbcTemplate(dataSource);
83         for(UserStory story : userStories){
84             String query = "UPDATE userstories SET storyName = ?, detail =
85                 ?, status = ? WHERE storyID = ?";
86             jdbcTemplateObject.update(query, new Object[] { story.
87                 getStoryName(), story.getDetail(), story.getStatus(),
88                 story.getStoryID() });
89         }
90     }
91
92     public void insertUserStories(List<UserStory> userStories){
93         jdbcTemplateObject = new JdbcTemplate(dataSource);
94         for(UserStory story : userStories){
95             String query = "INSERT INTO userStories (storyName, detail,
96                 projectID, status) VALUES (?, ?, ?, ?)";
97             jdbcTemplateObject.update(query, new Object[] { story.
98                 getStoryName(), story.getDetail(), story.getProjectID(),
99                 story.getStatus() });
100         }
101     }
102
103     public String getProjectName(int projectID){
104         jdbcTemplateObject = new JdbcTemplate(dataSource);
105         String query = "SELECT * FROM projects WHERE projectID = ?";
106         Project project = jdbcTemplateObject.queryForObject(query, new Object
107             [] { projectID }, new ProjectDetailsRowMapper());
108         return project.getProjectName();
109     }
110
111     public List<Dummy> getPendingProject(String email){
112         jdbcTemplateObject = new JdbcTemplate(dataSource);
113         String query = "SELECT * FROM invite where email = ?";
114         List<Dummy> dummies = jdbcTemplateObject.query(query, new Object[] {
115             email }, new DummyRowMapper());
116         return dummies;
117     }
118
119     public String getRole(int accountID, int projectID){
120         jdbcTemplateObject = new JdbcTemplate(dataSource);
121         String query = "SELECT * FROM members WHERE accountID = ? and projectID
122             = ?";
123         Member member = jdbcTemplateObject.queryForObject(query, new Object[] {
124             accountID, projectID }, new ProjectRowMapper());
125         return member.getRole();
126     }
127
128     public boolean isAccountHasProject(int accountID){
129         jdbcTemplateObject = new JdbcTemplate(dataSource);
130         String query = "SELECT COUNT(*) FROM members WHERE accountID =
131             ?";
132         int result = jdbcTemplateObject.queryForInt(query, new Object[] {
133             accountID });
134         if(result == 0){
135             return false;
136         }
137         else{
138             return true;
139         }
140     }
141
142     public List<Project> getProjectDetails(List<Member> members){
143         jdbcTemplateObject = new JdbcTemplate(dataSource);
144         List<Project> projects = new ArrayList<Project>();
145         String query = "SELECT * FROM projects where projectID = ?";
146         for(Member member : members){

```

```

138         projects.add((Project) jdbcTemplateObject.queryForObject
139             (query, new Object []{ member.getProjectID () }, new
140                 ProjectDetailsRowMapper ());
141     }
142     }
143     }
144     public Project getProjectByProjectID(int projectID){
145         jdbcTemplateObject = new JdbcTemplate(dataSource);
146         String query = "SELECT * FROM projects where projectID = ?";
147         Project project = jdbcTemplateObject.queryForObject(query, new Object
148             []{ projectID }, new ProjectDetailsRowMapper ());
149         return project;
150     }
151     }
152     public int addNewProject(Project project){
153         jdbcTemplateObject = new JdbcTemplate(dataSource);
154         project.setStatus(StatusUtility.ONGOING);
155         String query = "INSERT INTO projects (name, description, status) VALUES
156             (?, ?, ?)";
157         jdbcTemplateObject.update(query, new Object []{ project.getProjectName (),
158             project.getProjectDescription (), project.getStatus ()});
159         query = "SELECT * FROM projects where name = ?";
160         Project project2 = jdbcTemplateObject.queryForObject(query, new Object
161             []{ project.getProjectName (), new ProjectDetailsRowMapper ());
162         return project2.getProjectID ();
163     }
164     }
165     public void updateProject(Project project){
166         jdbcTemplateObject = new JdbcTemplate(dataSource);
167         String query = "UPDATE projects SET name=?, description=? WHERE
168             projectID = ?";
169         jdbcTemplateObject.update(query, new Object []{ project.getProjectName (),
170             project.getProjectDescription (), project.getProjectID ()});
171         return ;
172     }
173     }
174     public void addMember(Member member){
175         jdbcTemplateObject = new JdbcTemplate(dataSource);
176         String query = "INSERT INTO members (projectID, role, accountID) VALUES
177             (?, ?, ?)";
178         jdbcTemplateObject.update(query, new Object []{ member.getProjectID (),
179             member.getRole (), member.getAccountID ()});
180     }
181     }
182     public boolean isCurrentSprintExist(int projectID){
183         jdbcTemplateObject = new JdbcTemplate(dataSource);
184         String query = "SELECT COUNT(*) FROM sprints WHERE projectID = ? and
185             status = ?";
186         int row = jdbcTemplateObject.queryForInt(query, new Object []{ projectID,
187             StatusUtility.SPRINT_OPEN});
188         if(row == 1){
189             return true;
190         }
191         else{
192             return false;
193         }
194     }
195     }
196     public boolean isSprintExist(int projectID){
197         jdbcTemplateObject = new JdbcTemplate(dataSource);
198         String query = "SELECT COUNT(*) FROM sprints WHERE projectID = ?";
199         int row = jdbcTemplateObject.queryForInt(query, new Object []{ projectID
200             });
201         if(row == 0){
202             return false;
203         }
204         else
205             return true;
206     }
207     }
208     public Sprint getCurrentSprint(int projectID){
209         jdbcTemplateObject = new JdbcTemplate(dataSource);
210         String query = "SELECT * FROM sprints WHERE projectID = ? and status =
211             ?";
212         Sprint sprint = jdbcTemplateObject.queryForObject(query, new Object []{
213             projectID, StatusUtility.SPRINT_OPEN}, new SprintRowMapper ());
214         return sprint;
215     }
216     }
217     }
218     public List<Sprint> getAllSprint(int projectID){
219         jdbcTemplateObject = new JdbcTemplate(dataSource);
220         String query = "SELECT * FROM sprints WHERE projectID = ? ORDER BY
221             sprintNumber DESC";
222         List<Sprint> allSprint = jdbcTemplateObject.query(query, new Object []{
223             projectID }, new SprintRowMapper ());
224         return allSprint;
225     }
226     }
227     }
228     public void insertSprint(Sprint sprint){
229         jdbcTemplateObject = new JdbcTemplate(dataSource);
230         String query = "INSERT INTO sprints (sprintNumber, status, projectID,
231             sprintReview) VALUES (?, ?, ?, ?)";
232         jdbcTemplateObject.update(query, new Object []{ sprint.getSprintNumber (),
233             sprint.getStatus (), sprint.getProjectID (), sprint.getSprintReview

```

```

        ());
218     }
219
220     public int getLatestSprintNumber(int projectID){
221         jdbcTemplateObject = new JdbcTemplate(dataSource);
222         if(!isSprintExist(projectID)){
223             return 0;
224         }
225         else{
226             String query = "SELECT MAX(sprintNumber) FROM sprints WHERE
227                 projectID = ?";
228             int max = jdbcTemplateObject.queryForInt(query, new Object[]{
                projectID});
229         }
230     }
231
232     public void insertSprintStories(List<SprintStory> sprintStories){
233         jdbcTemplateObject = new JdbcTemplate(dataSource);
234         String query = "INSERT INTO sprintstory (sprintID, storyID) VALUES
235             (?,?)";
236         for(SprintStory sprintStory : sprintStories){
237             if(sprintStory.getSprintID() != 0 && sprintStory.getStoryID()
                != 0)
238                 jdbcTemplateObject.update(query, new Object[]{sprintStory.
                getSprintID(), sprintStory.getStoryID()});
239         }
240     }
241
242     public UserStory getUserStoryByID(int storyID){
243         jdbcTemplateObject = new JdbcTemplate(dataSource);
244         String query = "SELECT * FROM userstories WHERE storyID = ?";
245         UserStory story = jdbcTemplateObject.queryForObject(query, new Object
                []{storyID}, new UserStoryMapper());
246         return story;
247     }
248
249     public List<UserStory> getSprintStories(int sprintID){
250         jdbcTemplateObject = new JdbcTemplate(dataSource);
251         String query = "SELECT * FROM sprintstory WHERE sprintID = ?";
252         List<SprintStory> sprintStories = jdbcTemplateObject.query(query, new
                Object[]{sprintID}, new SprintStoryRowMapper());
253         List<UserStory> stories = new ArrayList<UserStory>();
254         for(SprintStory sprint : sprintStories){
255             stories.add(getUserStoryByID(sprint.getStoryID()));
256         }
257         return stories;
258     }
259
260     public void insertTasks(List<Task> tasks){
261         jdbcTemplateObject = new JdbcTemplate(dataSource);
262         String query = "INSERT INTO tasks (sprintID, storyID, accountID,
                taskName, estimate, status) VALUES (?, ?, ?, ?, ?, ?)";
263         for(Task task : tasks){
264             jdbcTemplateObject.update(query, task.getSprintID(), task.getStoryID(),
                task.getAccountID(), task.getTaskName(), task.getEstimate(), task
                .getStatus());
265         }
266     }
267
268     public List<Task> getTasks(int sprintID){
269         jdbcTemplateObject = new JdbcTemplate(dataSource);
270         String query = "SELECT * FROM tasks where sprintID = ?";
271         List<Task> tasks = jdbcTemplateObject.query(query, new Object[]{
                sprintID}, new TaskRowMapper());
272         return tasks;
273     }
274
275     public List<Member> projectMembers(int projectID){
276         jdbcTemplateObject = new JdbcTemplate(dataSource);
277         String query = "Select * FROM members WHERE projectID=?";
278         List<Member> members = jdbcTemplateObject.query(query, new Object[]{
                projectID}, new ProjectRowMapper());
279         for(Member mem : members){
280             mem.setName(userDAO.getUser2(mem.getAccountID()).getFirstName()
                );
281         }
282         return members;
283     }
284
285     public Task getTaskViaTaskID(int taskID){
286         jdbcTemplateObject = new JdbcTemplate(dataSource);
287         String query = "SELECT * from tasks WHERE taskID = ?";
288         Task task = jdbcTemplateObject.queryForObject(query, new Object[]{
                taskID}, new TaskRowMapper());
289         return task;
290     }
291
292     public List<Activity> getAllActivities(int taskID){
293         jdbcTemplateObject = new JdbcTemplate(dataSource);
294         String query = "SELECT * from activities WHERE taskID = ?";
295         List<Activity> activities = jdbcTemplateObject.query(query, new Object
                []{taskID}, new ActivityRowMapper());
296         return activities;
297     }
298
299
300

```

```

301     }
302
303     public void insertActivity(Activity activity){
304         jdbcTemplateObject = new JdbcTemplate(dataSource);
305         String query = "INSERT INTO activities (detail, taskID, date, time,
306             accountID) VALUES (?, ?, ?, ?, ?)";
307         jdbcTemplateObject.update(query, new Object[]{ activity.getDetail(),
308             activity.getTaskID(), activity.getDate(), activity.getTime(),
309             activity.getAccountID() });
310         return;
311     }
312
313     public void updateTaskStatus(String status, int taskID){
314         jdbcTemplateObject = new JdbcTemplate(dataSource);
315         String query = "UPDATE tasks SET status=? WHERE taskID = ? ";
316         jdbcTemplateObject.update(query, new Object[]{ status, taskID });
317     }
318
319     public Sprint getSprintBySprintID(int sprintID){
320         jdbcTemplateObject = new JdbcTemplate(dataSource);
321         String query = "SELECT * FROM sprints WHERE sprintID = ?";
322         Sprint sprint = jdbcTemplateObject.queryForObject(query, new Object[]{
323             sprintID }, new SprintRowMapper());
324         return sprint;
325     }
326
327     public boolean isActivityExisting(int taskID){
328         jdbcTemplateObject = new JdbcTemplate(dataSource);
329         String query = "SELECT COUNT(*) FROM activities WHERE taskID = ?";
330         int row = jdbcTemplateObject.queryForInt(query, new Object[]{ taskID });
331         if(row == 0){
332             return false;
333         }
334         else{
335             return true;
336         }
337     }
338
339     public int getTotalActivityPerDay(int taskID, String date){
340         jdbcTemplateObject = new JdbcTemplate(dataSource);
341         String query = "SELECT * FROM activities WHERE taskID = ? AND date =
342             ?";
343         if(isActivityExisting(taskID)){
344             List<Activity> activities = jdbcTemplateObject.query(query, new Object
345                 []{ taskID, date }, new ActivityRowMapper());
346             int sum = 0;
347             for(Activity act : activities){
348                 sum += act.getTime();
349             }
350             return sum;
351         }
352         else{
353             return 0;
354         }
355     }
356
357     public int getTotalEstimate(int sprintID){
358         jdbcTemplateObject = new JdbcTemplate(dataSource);
359         String query = "SELECT * FROM tasks WHERE sprintID = ?";
360         List<Task> tasks = jdbcTemplateObject.query(query, new Object[]{ sprintID
361             }, new TaskRowMapper());
362         int sum = 0;
363         for(Task task : tasks){
364             sum += task.getEstimate();
365         }
366         return sum;
367     }
368
369     public void insertNotif(Notification notif){
370         jdbcTemplateObject = new JdbcTemplate(dataSource);
371         String query = "INSERT INTO notifications (accountID, projectID, type)
372             VALUES (?, ?, ?)";
373         jdbcTemplateObject.update(query, new Object[]{ notif.getAccountID(),
374             notif.getProjectID(), notif.getType() });
375         return;
376     }
377
378     public List<Sprint> getAllOpenSprint(){
379         jdbcTemplateObject = new JdbcTemplate(dataSource);
380         String query = "SELECT * FROM sprints WHERE status = ?";
381         List<Sprint> sprint = jdbcTemplateObject.query(query, new Object[]{"
382             open"}, new SprintRowMapper());
383         return sprint;
384     }
385
386     public List<Meeting> getScrumMeeting(int sprintID){
387         jdbcTemplateObject = new JdbcTemplate(dataSource);
388         String query = "SELECT * FROM scrummeeting WHERE sprintID = ? GROUP BY
389             date ORDER BY meetingID DESC";
390         List<Meeting> meeting = jdbcTemplateObject.query(query, new Object[]{
391             sprintID }, new MeetingRowMapper());
392         return meeting;
393     }

```

```

388     public List<Meeting> getMeetingDetail(int sprintID, String date){
389         jdbcTemplateObject = new JdbcTemplate(dataSource);
390         String query = "SELECT * FROM scrummeeting WHERE sprintID = ? AND date
= ? ORDER BY meetingID DESC";
391         List<Meeting> meeting = jdbcTemplateObject.query(query, new Object[] {
sprintID, date}, new MeetingRowMapper());
392         return meeting;
393     }
394
395     public boolean isScrumMeeting(int sprintID){
396         jdbcTemplateObject = new JdbcTemplate(dataSource);
397         String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?";
398         int row = jdbcTemplateObject.queryForInt(query, new Object[] {sprintID}
);
399         if(row == 0){
400             return false;
401         }
402         return true;
403     }
404
405     public boolean isCurrentScrumMeeting(int sprintID){
406         jdbcTemplateObject = new JdbcTemplate(dataSource);
407         String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?
AND date = ?";
408         int row = jdbcTemplateObject.queryForInt(query, new Object[] {sprintID,
DateHandler.getCurrentDate()});
409         if(row == 0)
410             return false;
411         return true;
412     }
413
414     public boolean isScrumMeetingAnswered(int sprintID, int accountID){
415         jdbcTemplateObject = new JdbcTemplate(dataSource);
416         String query = "SELECT COUNT(*) FROM scrummeeting WHERE sprintID = ?
AND date = ? AND accountID = ?";
417         int row = jdbcTemplateObject.queryForInt(query, new Object[] {sprintID,
DateHandler.getCurrentDate(), accountID});
418         if(row == 0)
419             return false;
420         return true;
421     }
422 }
423
424     public void insertScrumMeeting(Meeting meeting){
425         jdbcTemplateObject = new JdbcTemplate(dataSource);
426         String query = "INSERT INTO scrummeeting (date, accountID,
accomplishment, plan, impediment, sprintID) VALUES (?, ?, ?, ?, ?, ?)";
427         jdbcTemplateObject.update(query, new Object[] {meeting.getDate(), meeting
.getAccountID(), meeting.getAccomplishment(), meeting.getPlan(),
meeting.getImpediment(), meeting.getSprintID()});
428         return ;
429     }
430
431     public boolean isReviewExisting(int sprintID){
432         jdbcTemplateObject = new JdbcTemplate(dataSource);
433         String query = "SELECT COUNT(*) from sprintreview WHERE sprintID = ?";
434         int row = jdbcTemplateObject.queryForInt(query, new Object[] {sprintID}
);
435         if(row == 0)
436             return false;
437         return true;
438     }
439 }
440
441     public List<Review> getSprintReview(int sprintID){
442         jdbcTemplateObject = new JdbcTemplate(dataSource);
443         String query = "SELECT * from sprintreview WHERE sprintID = ?";
444         List<Review> reviews = jdbcTemplateObject.query(query, new Object[] {
sprintID}, new ReviewRowMapper());
445         return reviews;
446     }
447
448     public void insertSprintReview(Review review){
449         jdbcTemplateObject = new JdbcTemplate(dataSource);
450         String query = "INSERT INTO sprintreview (sprintID, storyID, review)
VALUES (?, ?, ?)";
451         jdbcTemplateObject.update(query, new Object[] {review.getSprintID(),
review.getStoryID(), review.getReview()});
452         return;
453     }
454
455     public void updateStoryStatus(String status, int storyID){
456         jdbcTemplateObject = new JdbcTemplate(dataSource);
457         String query = "UPDATE userstories SET status = ? WHERE storyID = ?";
458         jdbcTemplateObject.update(query, new Object[] {status, storyID});
459         return;
460     }
461
462     public void closeSprint(int sprintID){
463         jdbcTemplateObject = new JdbcTemplate(dataSource);
464         String query = "UPDATE sprints SET status = ? WHERE sprintID = ?";
465         jdbcTemplateObject.update(query, new Object[] {StatusUtility.CLOSED,
sprintID});
466         return;
467     }
468
469     public void closeProject(int projectID){
470         jdbcTemplateObject = new JdbcTemplate(dataSource);
471         String query = "UPDATE projects SET status = ? WHERE projectID = ?";

```

```

472         jdbcTemplateObject.update(query, new Object[]{ StatusUtility.CLOSED,
473             projectID});
474     }
475     *
476     * Row mapper for projects table
477     */
478 }

```

Listing 14: ProjectDetailsRowMapper.java

```

1  /**
2  * ProjectDetailsRowMapper.java
3  *
4  * Row mapper for projects table
5  */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14
15 public class ProjectDetailsRowMapper implements RowMapper {
16
17     @Override
18     public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
19         // TODO Auto-generated method stub
20         Project project = new Project();
21         project.setProjectID(rs.getInt("projectID"));
22         project.setProjectName(rs.getString("name"));
23         project.setProjectDescription(rs.getString("description"));
24         project.setStatus(rs.getString("status"));
25         return project;
26     }
27 }
28 }

```

Listing 15: ProjectRowMapper.java

```

1  /**
2  * ProjectRowMapper.java
3  *
4  * Row mapper for projects table
5  *
6  */
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14
15 public class ProjectRowMapper implements RowMapper {
16
17     @Override
18     public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
19         Member member = new Member();
20         member.setMemberID(rs.getInt("memberID"));
21         member.setProjectID(rs.getInt("projectID"));
22         member.setRole(rs.getString("role"));
23         member.setAccountID(rs.getInt("accountID"));
24         // TODO Auto-generated method stub
25         return member;
26     }
27 }

```

Listing 16: Review.java

```

1  /**
2  * Review.java
3  *
4  * Model for object Review
5  */
6
7  package com.scrumprimer.model;
8
9  public class Review {
10     int reviewID;
11     int sprintID;
12     int storyID;
13     String review;
14     String status;
15     String storyName;
16     public int getReviewID() {
17         return reviewID;
18     }
19     public void setReviewID(int reviewID) {
20         this.reviewID = reviewID;
21     }
22     public int getSprintID() {

```



```

23         return sprintID;
24     }
25     public void setSprintID(int sprintID) {
26         this.sprintID = sprintID;
27     }
28     public int getStoryID() {
29         return storyID;
30     }
31     public void setStoryID(int storyID) {
32         this.storyID = storyID;
33     }
34     public String getReview() {
35         return review;
36     }
37     public void setReview(String review) {
38         this.review = review;
39     }
40     public String getStatus() {
41         return status;
42     }
43     public void setStatus(String status) {
44         this.status = status;
45     }
46     public String getStoryName() {
47         return storyName;
48     }
49     public void setStoryName(String storyName) {
50         this.storyName = storyName;
51     }
52 }
53 }

```

Listing 17: ReviewRowMapper.java

```

1  /**
2   * ReviewRowMapper.java
3   *
4   * Row mapper for reviews table
5   */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class ReviewRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Review review = new Review();
20         review.setReview(rs.getString("review"));
21         review.setReviewID(rs.getInt("reviewID"));
22         review.setSprintID(rs.getInt("sprintID"));
23         review.setStoryID(rs.getInt("storyID"));
24         return review;
25     }
26
27 }
28 }

```

Listing 18: Sprint.java

```

1  /**
2   * Sprint.java
3   *
4   * Model for object Sprint
5   */
6  package com.scrumprimer.model;
7
8  import java.util.Date;
9
10 public class Sprint {
11     int sprintID;
12     int projectID;
13     int sprintNumber;
14     String status;
15     String sprintReview;
16     String dateStarted;
17     public String getDateStarted() {
18         return dateStarted;
19     }
20     public void setDateStarted(String dateStarted) {
21         this.dateStarted = dateStarted;
22     }
23     public String getSprintReview() {
24         return sprintReview;
25     }
26     public void setSprintReview(String sprintReview) {
27         this.sprintReview = sprintReview;
28     }
29     public int getSprintID() {
30         return sprintID;
31     }

```

```

32     public void setSprintID(int sprintID) {
33         this.sprintID = sprintID;
34     }
35     public int getProjectID() {
36         return projectID;
37     }
38     public void setProjectID(int projectID) {
39         this.projectID = projectID;
40     }
41     public int getSprintNumber() {
42         return sprintNumber;
43     }
44     public void setSprintNumber(int sprintNumber) {
45         this.sprintNumber = sprintNumber;
46     }
47     public String getStatus() {
48         return status;
49     }
50     public void setStatus(String status) {
51         this.status = status;
52     }
53 }
54 }

```

Listing 19: SprintRowMapper.java

```

1  /**
2   * SprintRowMapper.java
3   *
4   * Row mapper for sprints table
5   */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class SprintRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Sprint sprint = new Sprint();
20         sprint.setProjectID(rs.getInt("projectID"));
21         sprint.setSprintID(rs.getInt("sprintID"));
22         sprint.setSprintNumber(rs.getInt("sprintNumber"));
23         sprint.setStatus(rs.getString("status"));
24         sprint.setSprintReview(rs.getString("sprintReview"));
25         sprint.setDateStarted(rs.getString("dateStarted"));
26         return sprint;
27     }
28 }
29 }

```

Listing 20: SprintStory.java

```

1  /**
2   * SprintStory.java
3   *
4   * Model for object SprintStory
5   */
6
7  package com.scrumprimer.model;
8
9  public class SprintStory {
10     int sprintID;
11     int storyID;
12     public int getSprintID() {
13         return sprintID;
14     }
15     public void setSprintID(int sprintID) {
16         this.sprintID = sprintID;
17     }
18     public int getStoryID() {
19         return storyID;
20     }
21     public void setStoryID(int storyID) {
22         this.storyID = storyID;
23     }
24 }

```

Listing 21: SprintStoryRowMapper.java

```

1  /**
2   * SprintStoryRowMapper.java
3   *
4   * Row mapper for sprintstory table
5   */
6
7  package com.scrumprimer.model;
8
9  import java.sql.ResultSet;

```

```

10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class SprintStoryRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         SprintStory sprint = new SprintStory();
20         sprint.setSprintID(rs.getInt("sprintID"));
21         sprint.setStoryID(rs.getInt("storyID"));
22         return sprint;
23     }
24
25 }

```

Listing 22: TaskForm.java

```

1 /**
2  * TaskForm.java
3  *
4  * Model for multiple tasks object
5  */
6
7 package com.scrumprimer.model;
8
9 import org.springframework.util.AutoPopulatingList;
10
11 public class TaskForm {
12     private AutoPopulatingList<Task> tasks;
13
14     public AutoPopulatingList<Task> getTasks() {
15         return tasks;
16     }
17
18     public void setTasks(AutoPopulatingList<Task> tasks) {
19         this.tasks = tasks;
20     }
21 }

```

Listing 23: TaskRowMapper.java

```

1 /**
2  * TaskRowMapper.java
3  *
4  * Row mapper for tasks table
5  */
6
7 package com.scrumprimer.model;
8
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import org.springframework.jdbc.core.RowMapper;
13
14 public class TaskRowMapper implements RowMapper {
15
16     @Override
17     public Object mapRow(ResultSet rs, int row) throws SQLException {
18         // TODO Auto-generated method stub
19         Task task = new Task();
20         task.setTaskName(rs.getString("taskName"));
21         task.setEstimate(rs.getInt("estimate"));
22         task.setSprintID(rs.getInt("sprintID"));
23         task.setTaskID(rs.getInt("taskID"));
24         task.setStatus(rs.getString("status"));
25         task.setStoryID(rs.getInt("storyID"));
26         task.setAccountID(rs.getInt("accountID"));
27
28         return task;
29     }
30
31 }

```

Listing 24: User.java

```

1 /**
2  * User.java
3  *
4  * Model for object User
5  */
6
7 package com.scrumprimer.model;
8
9 public class User {
10     private String firstName;
11     private String lastName;
12     private String email;
13     private String password;
14     private Integer userID;
15     private String username;
16     private String newPassword;
17     private String newPassword2;

```

```

18     private String role;
19     private String userType;
20
21     public String getUserType() {
22         return userType;
23     }
24     public void setUserType(String userType) {
25         this.userType = userType;
26     }
27     public String getRole() {
28         return role;
29     }
30     public void setRole(String role) {
31         this.role = role;
32     }
33     public String getNewPassword() {
34         return newPassword;
35     }
36     public void setNewPassword(String newPassword) {
37         this.newPassword = newPassword;
38     }
39     public String getNewPassword2() {
40         return newPassword2;
41     }
42     public void setNewPassword2(String newPassword2) {
43         this.newPassword2 = newPassword2;
44     }
45     public String getFirstName() {
46         return firstName;
47     }
48     public void setFirstName(String firstName) {
49         this.firstName = firstName;
50     }
51     public String getUsername() {
52         return username;
53     }
54     public void setUsername(String username) {
55         this.username = username;
56     }
57     public String getLastName() {
58         return lastName;
59     }
60     public void setLastName(String lastName) {
61         this.lastName = lastName;
62     }
63     public String getEmail() {
64         return email;
65     }
66     public void setEmail(String email) {
67         this.email = email;
68     }
69     public String getPassword() {
70         return password;
71     }
72     public void setPassword(String password) {
73         this.password = password;
74     }
75     public Integer getUserID() {
76         return userID;
77     }
78     public void setUserID(Integer userID) {
79         this.userID = userID;
80     }
81 }
82
83 }

```

Listing 25: UserDAO.java

```

1  /**
2   * UserDAO.java
3   */
4  package com.scrumprimer.model;
5
6  import java.util.ArrayList;
7  import java.util.List;
8
9  import javax.servlet.http.HttpSession;
10 import javax.sql.DataSource;
11
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.jdbc.core.JdbcTemplate;
14 import org.springframework.jdbc.support.rowset.SqlRowSet;
15 import org.springframework.security.authentication.encoding.PasswordEncoder;
16
17 import com.mysql.jdbc.ResultSet;
18 import com.scrumprimer.model.User;
19 import com.scrumprimer.model.UserRowMapper;
20 import com.scrumprimer.utility.SessionUtility;
21
22 public class UserDAO {
23
24     @Autowired
25     private DataSource dataSource;
26     private JdbcTemplate jdbcTemplateObject;
27     @Autowired
28     private PasswordEncoder passwordEncoder;
29     @Autowired

```

```

30     private ProjectDAO projectDAO;
31
32     public boolean insertUser(User newUser){
33         jdbcTemplateObject = new JdbcTemplate(dataSource);
34         String password = passwordEncoder.encodePassword(newUser.getPassword(),
35             null);
36
37         if(isUsernameExist(newUser.getUsername())){
38             return false;
39         }
40         String SQL = "insert into accounts (username, password, firstName,
41             lastName, email, type) values (?, ?, ?, ?, ?,?)";
42         jdbcTemplateObject.update( SQL,new Object []{newUser.getUsername(),
43             password, newUser.getFirstName(), newUser.getLastName(), newUser.
44             getEmail(), newUser.getUserType()});
45         return true;
46     }
47     public void addInvite(ArrayList<User> users, HttpSession session){
48         jdbcTemplateObject = new JdbcTemplate(dataSource);
49
50         for(User user : users){
51             if(isEmailExist(user.getEmail())){
52                 String query = "SELECT * FROM accounts WHERE email = ?";
53                 User user2 = jdbcTemplateObject.queryForObject(query,
54                     new Object []{user.getEmail()}, new UserRowMapper()
55                     );
56                 Member member = new Member();
57                 member.setAccountID(user2.getUserID());
58                 member.setProjectID((Integer) (session.getAttribute(
59                     SessionUtility.SESSION.PROJECT)));
60                 member.setRole(user.getRole());
61
62                 projectDAO.addMember(member);
63             }
64             else{
65                 String query = "INSERT into invite (firstName, lastName,
66                     projectID, email, role) values (?, ?, ?, ?, ?)";
67                 jdbcTemplateObject.update(query, new Object []{user.
68                     getFirstName(), user.getLastName(), session.
69                     getAttribute(SessionUtility.SESSION.PROJECT), user.
70                     getEmail(), user.getRole()});
71             }
72         }
73     }
74     public User getUser(User loginUser){
75         jdbcTemplateObject = new JdbcTemplate(dataSource);
76         String password = passwordEncoder.encodePassword(loginUser.getPassword()
77             , null);
78
79         jdbcTemplateObject = new JdbcTemplate(dataSource);
80
81         String query = "Select * from accounts where username = ? and password =
82             ?";
83
84         User user = jdbcTemplateObject.queryForObject(query, new Object []{
85             loginUser.getUsername(), password}, new UserRowMapper());
86         return user;
87     }
88     public User getUser2(Integer accountID){
89         jdbcTemplateObject = new JdbcTemplate(dataSource);
90
91         String query = "Select * from accounts where accountID = ?";
92
93         User user = jdbcTemplateObject.queryForObject(query, new Object []{
94             accountID}, new UserRowMapper());
95         return user;
96     }
97     public int getUserIDViaEmail(String email){
98         jdbcTemplateObject = new JdbcTemplate(dataSource);
99         String query = "Select * from accounts where email = ?";
100
101         User user = jdbcTemplateObject.queryForObject(query, new Object []{email
102             }, new UserRowMapper());
103         return user.getUserID();
104     }
105     public boolean isAccountExist(String username, String password){
106         jdbcTemplateObject = new JdbcTemplate(dataSource);
107         String encodePassword = passwordEncoder.encodePassword(password, null);
108         System.out.println(encodePassword);
109
110         String query = "Select COUNT(*) from accounts where username = ? and
111             password = ?";
112         @SuppressWarnings("deprecation")
113         int row = jdbcTemplateObject.queryForInt(query, new Object []{username,
114             encodePassword});
115         System.out.println(row);

```

```

111         if(row != 1)
112             return false;
113         else
114             return true;
115     }
116
117     public boolean isUsernameExist(String username){
118         jdbcTemplateObject = new JdbcTemplate(dataSource);
119
120         String query = "Select COUNT(*) from accounts where username = ?";
121         @SuppressWarnings(" deprecation")
122         int row = jdbcTemplateObject.queryForInt(query, new Object [] { username});
123         System.out.println(row);
124         if(row != 1)
125             return false;
126         else
127             return true;
128     }
129
130     public boolean isEmailExist(String email){
131         jdbcTemplateObject = new JdbcTemplate(dataSource);
132
133         String query = "Select COUNT(*) from accounts where email = ?";
134         @SuppressWarnings(" deprecation")
135         int row = jdbcTemplateObject.queryForInt(query, new Object [] { email});
136         if(row != 1)
137             return false;
138         else
139             return true;
140     }
141
142     public boolean isEmailInvite(String email){
143         jdbcTemplateObject = new JdbcTemplate(dataSource);
144
145         String query = "Select COUNT(*) from invite where email = ?";
146         int row = jdbcTemplateObject.queryForInt(query, new Object [] { email});
147         if(row == 0)
148             return false;
149         else
150             return true;
151     }
152
153
154     public void updateUser(User user){
155         jdbcTemplateObject = new JdbcTemplate(dataSource);
156
157         String query = "UPDATE accounts SET username = ?, firstName = ?, email =
158         ?, lastName = ? WHERE accountID = ?";
159         jdbcTemplateObject.update(query, new Object [] { user.getUsername(), user.
160             getFirstName(), user.getEmail(), user.getLastName(), user.getUserID
161             ()});
162         return;
163     }
164
165     public void updatePassword(User user){
166         jdbcTemplateObject = new JdbcTemplate(dataSource);
167         String encodePassword = passwordEncoder.encodePassword(user.
168             getNewPassword(), null);
169         String query = "UPDATE accounts SET password = ? WHERE accountID = ?";
170         jdbcTemplateObject.update(query, new Object [] { encodePassword, user.
171             getUserID () });
172         return;
173     }
174
175     public List<User> getAllUser(){
176         jdbcTemplateObject = new JdbcTemplate(dataSource);
177         String query = "SELECT * FROM accounts";
178         List<User> users = jdbcTemplateObject.query(query, new UserRowMapper());
179         return users;
180     }
181
182     public void deleteUser(int accountID){
183         jdbcTemplateObject = new JdbcTemplate(dataSource);
184         String query = "DELETE FROM accounts where accountID = ?";
185         jdbcTemplateObject.update(query, accountID);
186         return;
187     }
188 }

```

Listing 26: UserForm.java

```

1  /**
2  * UserForm.java
3  *
4  * Model for multiple user object
5  */
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9
10 public class UserForm {
11
12     private ArrayList<User> users;
13
14     public ArrayList<User> getUsers() {
15         return users;
16     }

```

```

17         public void setUsers(ArrayList<User> users) {
18             this.users = users;
19         }
20     }
21 }
22 }

```

Listing 27: UserRowMapper.java

```

1  /**
2   * UserRowMapper.java
3   *
4   * Row mapper for users table
5   */
6  package com.scrumprimer.model;
7
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10
11 import org.springframework.jdbc.core.RowMapper;
12
13 public class UserRowMapper implements RowMapper {
14
15     @Override
16     public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
17         // TODO Auto-generated method stub
18         User user = new User();
19         user.setUserID(rs.getInt("accountID"));
20         user.setPassword(rs.getString("password"));
21         user.setUsername(rs.getString("username"));
22         user.setEmail(rs.getString("email"));
23         user.setFirstName(rs.getString("firstName"));
24         user.setLastName(rs.getString("lastName"));
25         user.setUserType(rs.getString("type"));
26         return user;
27     }
28 }
29 }

```

Listing 28: UserSprintForm.java

```

1  /**
2   * UserSprintForm.java
3   *
4   * Model for multiple sprintstory object
5   */
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9  import java.util.List;
10
11 public class UserSprintForm {
12
13     ArrayList<SprintStory> sprintStories;
14
15     public ArrayList<SprintStory> getSprintStories() {
16         return sprintStories;
17     }
18
19     public void setSprintStories(ArrayList<SprintStory> sprintStories) {
20         this.sprintStories = sprintStories;
21     }
22 }

```

Listing 29: UserStory.java

```

1  /**
2   * UserStory.java
3   *
4   * Model for object UserStory
5   */
6  package com.scrumprimer.model;
7
8  public class UserStory {
9     private int storyID;
10    private String detail;
11    private String storyName;
12    private String status;
13    private int projectID;
14    public int getStoryID() {
15        return storyID;
16    }
17    public void setStoryID(int storyID) {
18        this.storyID = storyID;
19    }
20    public String getDetail() {
21        return detail;
22    }
23    public void setDetail(String detail) {
24        this.detail = detail;
25    }
26    public String getStoryName() {
27        return storyName;
28    }

```

```

29     public void setStoryName(String storyName) {
30         this.storyName = storyName;
31     }
32     public String getStatus() {
33         return status;
34     }
35     public void setStatus(String status) {
36         this.status = status;
37     }
38     public int getProjectID() {
39         return projectID;
40     }
41     public void setProjectID(int projectID) {
42         this.projectID = projectID;
43     }
44 }
45 }
46 }
47 }

```

Listing 30: UserStoryForm.java

```

1  /**
2   * UserStoryForm.java
3   *
4   * Model for userstory object
5   */
6  package com.scrumprimer.model;
7
8  import java.util.ArrayList;
9
10 import org.springframework.util.AutoPopulatingList;
11
12 public class UserStoryForm {
13
14     private ArrayList<UserStory> stories;
15
16     public ArrayList<UserStory> getStories() {
17         return stories;
18     }
19
20     public void setStories(ArrayList<UserStory> stories) {
21         this.stories = stories;
22     }
23
24 }
25 }
26 }
27 }

```

Listing 31: UserStoryMapper.java

```

1  /**
2   * UserStoryMapper.java
3   *
4   * Row mapper for userstories table
5   */
6  package com.scrumprimer.model;
7
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10
11 import org.springframework.jdbc.core.RowMapper;
12
13 public class UserStoryMapper implements RowMapper {
14
15     @Override
16     public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
17         UserStory story = new UserStory();
18         story.setDetail(rs.getString("detail"));
19         story.setProjectID(rs.getInt("projectID"));
20         story.setStatus(rs.getString("status"));
21         story.setStoryID(rs.getInt("storyID"));
22         story.setStoryName(rs.getString("storyName"));
23         return story;
24     }
25 }
26 }
27 }

```

2. Controllers

Listing 32: AdminController.java

```

1  /**
2   * AdminController.java
3   *
4   * Controller for all Admin functionalities. Involve view accounts, delete accounts and
5     update account
6   */
7  package com.scrumprimer.controller;
8

```



```

9  import java.util.List;
10
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpSession;
13
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.context.ApplicationContext;
16 import org.springframework.context.support.ClassPathXmlApplicationContext;
17 import org.springframework.security.authentication.encoding.PasswordEncoder;
18 import org.springframework.stereotype.Controller;
19 import org.springframework.ui.Model;
20 import org.springframework.ui.ModelMap;
21 import org.springframework.validation.BindingResult;
22 import org.springframework.web.bind.annotation.ModelAttribute;
23 import org.springframework.web.bind.annotation.RequestMapping;
24 import org.springframework.web.bind.annotation.RequestMethod;
25 import org.springframework.web.bind.annotation.RequestParam;
26 import org.springframework.web.bind.annotation.SessionAttributes;
27 import org.springframework.web.servlet.ModelAndView;
28 import org.springframework.web.servlet.config.annotation.EnableWebMvc;
29 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
30
31 import com.scrumprimer.model.User;
32 import com.scrumprimer.model.UserDAO;
33 import com.scrumprimer.utility.SessionUtility;
34
35 @Controller
36 @SessionAttributes
37 @EnableWebMvc
38
39
40
41 public class AdminController {
42     @Autowired
43     UserDAO userDAO;
44     @Autowired
45     private PasswordEncoder passwordEncoder;
46
47     @RequestMapping("/Accounts")
48     public ModelAndView viewAccounts(HttpSession session){
49         ModelAndView view = new ModelAndView("accounts");
50         view.addObject("edituser", new User());
51         view.addObject("edituser2", new User());
52         List<User> users = userDAO.getAllUser();
53         view.addObject("users", users);
54         view.addObject("newUser", new User());
55         view.addObject("edit", new User());
56         view.addObject("activeaccounts", "active");
57         return view;
58     }
59
60     @RequestMapping("/deleteAccount")
61     public String deleteAccounts(HttpSession session, @RequestParam("aid")int
62         accountID, RedirectAttributes map){
63         userDAO.deleteUser(accountID);
64         map.addFlashAttribute("successfuldelete", "Account has been
65             successfully deleted");
66         return "redirect:Accounts";
67     }
68
69     @RequestMapping("/editAccount")
70     public String editAccount(HttpSession session, @RequestParam("aid")int
71         accountID, RedirectAttributes map){
72         User updateUser = userDAO.getUser2(accountID);
73         map.addFlashAttribute("updateuser", updateUser);
74         map.addFlashAttribute("update", "yes");
75         return "redirect:Accounts";
76     }
77
78     @RequestMapping("/updateAccount")
79     public String updateAccount(HttpSession session, @ModelAttribute("edit")User
80         user, RedirectAttributes map){
81         userDAO.updateUser(user);
82         map.addFlashAttribute("updateaccount", "Account has been successfully
83             updated");
84         return "redirect:Accounts";
85     }
86
87     @RequestMapping("/addAccount")
88     public String addAccount(HttpSession session, @ModelAttribute("newUser")User
89         user, RedirectAttributes map){
90         userDAO.insertUser(user);
91         map.addFlashAttribute("addaccount", "Account has been successfully
92             added");
93         return "redirect:Accounts";
94     }
95 }

```

Listing 33: LoginController.java

```

1  /**
2   * LoginController.java
3   *
4   * Controller for logging in and logout.
5   *

```

```

6  */
7  package com.scrumprimer.controller;
8
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpSession;
11
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.context.ApplicationContext;
14 import org.springframework.context.support.ClassPathXmlApplicationContext;
15 import org.springframework.security.authentication.encoding.PasswordEncoder;
16 import org.springframework.stereotype.Controller;
17 import org.springframework.ui.Model;
18 import org.springframework.ui.ModelMap;
19 import org.springframework.validation.BindingResult;
20 import org.springframework.web.bind.annotation.ModelAttribute;
21 import org.springframework.web.bind.annotation.RequestMapping;
22 import org.springframework.web.bind.annotation.RequestMethod;
23 import org.springframework.web.bind.annotation.SessionAttributes;
24 import org.springframework.web.servlet.ModelAndView;
25 import org.springframework.web.servlet.config.annotation.EnableWebMvc;
26 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
27
28 import com.scrumprimer.model.User;
29 import com.scrumprimer.model.UserDAO;
30 import com.scrumprimer.utility.SessionUtility;
31
32 @Controller
33 @SessionAttributes
34 @EnableWebMvc
35
36 public class LoginController {
37     @Autowired
38     UserDAO userDAO;
39     @Autowired
40     private PasswordEncoder passwordEncoder;
41
42     @RequestMapping(value = "/Login", method = RequestMethod.POST)
43     public String addUser(@ModelAttribute("login")
44                                     User login, BindingResult result
45                                     , HttpSession session,
46                                     RedirectAttributes map) {
47
48         //userDAO.getUser(login);
49         if (userDAO.isAccountExist(login.getUsername(), login.getPassword())){
50             User user = new User();
51             user = userDAO.getUser(login);
52
53             session.setAttribute(SessionUtility.USERNAME, user.getUsername()
54                                 );
55             session.setAttribute(SessionUtility.SESSION_ID, user.getUserID()
56                                 );
57             session.setAttribute(SessionUtility.SESSION_FIRSTNAME, user.
58                                 getFirstName());
59             session.setAttribute(SessionUtility.SESSION_LASTNAME, user.
60                                 getLastName());
61             session.setAttribute(SessionUtility.SESSION_EMAIL, user.getEmail
62                                 ());
63             session.setAttribute(SessionUtility.PASSWORD, login.getPassword
64                                 ());
65
66             if (user.getUserType().equalsIgnoreCase("admin") ){
67                 session.setAttribute(SessionUtility.SESSION_USERTYPE,
68                                     user.getUserType());
69                 return "redirect:Accounts";
70             }
71             return "redirect:Project";
72         }
73         else{
74             map.addFlashAttribute("errorlogin", "Invalid username and
75             password");
76             return "redirect:Home";
77         }
78     }
79
80     @RequestMapping("/Logout")
81     public String logout(HttpSession session){
82         session.invalidate();
83         return "redirect:Home";
84     }
85 }

```

Listing 34: ProjectController.java

```

1  /**
2   * ProjectController.java
3   *
4   * Controller for project's page.
5   *
6   */
7  package com.scrumprimer.controller;
8
9  import java.io.InputStream;
10 import java.io.UnsupportedEncodingException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import java.util.Properties;
14
15

```

```

16
17
18 import javax.servlet.http.HttpSession;
19
20 import org.springframework.beans.factory.annotation.Autowired;
21 import org.springframework.context.ApplicationContext;
22 import org.springframework.context.support.ClassPathXmlApplicationContext;
23 import org.springframework.stereotype.Controller;
24 import org.springframework.ui.ModelMap;
25 import org.springframework.validation.BindingResult;
26 import org.springframework.web.bind.annotation.ModelAttribute;
27 import org.springframework.web.bind.annotation.RequestMapping;
28 import org.springframework.web.bind.annotation.RequestMethod;
29 import org.springframework.web.bind.annotation.RequestParam;
30 import org.springframework.web.servlet.ModelAndView;
31 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
32
33
34 import com.scrumprimer.model.Member;
35 import com.scrumprimer.model.NotificationDAO;
36 import com.scrumprimer.model.Project;
37 import com.scrumprimer.model.ProjectDAO;
38 import com.scrumprimer.model.User;
39 import com.scrumprimer.model.UserDAO;
40 import com.scrumprimer.model.UserForm;
41 import com.scrumprimer.utility.EmailNotification;
42 import com.scrumprimer.utility.RoleUtility;
43 import com.scrumprimer.utility.SessionUtility;
44
45 @Controller
46 @SessionAttributes
47
48
49 public class ProjectController {
50     @Autowired
51     UserDAO userDAO;
52
53     @Autowired
54     ProjectDAO projectDAO;
55
56     @Autowired
57     NotificationDAO notifDAO;
58
59
60     @RequestMapping("/Project")
61     public ModelAndView viewProject(HttpSession session){
62         ModelAndView project = new ModelAndView("projects");
63         project.addObject("edituser", new User());
64         project.addObject("edituser2", new User());
65         List<Member> projects = new ArrayList<Member>();
66         List<Project> projectDetails = new ArrayList<Project>();
67
68         if(!projectDAO.isAccountHasProject((Integer)session.getAttribute(
69             SessionUtility.SESSION_ID)){
70             project.addObject("noexisting","No existing projects");
71         }
72         else{
73             projects = projectDAO.getProject((Integer)session.getAttribute(
74                 SessionUtility.SESSION_ID));
75             projectDetails = projectDAO.getProjectDetails(projects);
76         }
77         project.addObject("projects", projects);
78         project.addObject("projectDetails", projectDetails);
79         project.addObject("newProject", new Project());
80         project.addObject("activeproject", "active");
81         return project;
82     }
83
84     @RequestMapping("/editProject")
85     public String editProject(HttpSession session, @RequestParam("pid")int projectID
86         , RedirectAttributes map){
87         Project updateProject = projectDAO.getProjectByProjectID(projectID);
88         map.addFlashAttribute("updateProject", updateProject);
89         map.addFlashAttribute("update", "yes");
90         return "redirect:Project";
91     }
92
93     @RequestMapping("/updateProject")
94     public String updateProject(HttpSession session, @ModelAttribute("newProject")
95         Project newProject, BindingResult result, RedirectAttributes map){
96         projectDAO.updateProject(newProject);
97         map.addFlashAttribute("projectupdated", "Project has been updated");
98         return "redirect:Project";
99     }
100
101     @RequestMapping("/viewProject")
102     public String view_project(HttpSession session, @RequestParam("pid")int
103         projectID, RedirectAttributes map){
104         session.setAttribute(SessionUtility.SESSION_PROJECT, projectID);
105         String projectName = projectDAO.getProjectName(projectID);
106         session.setAttribute(SessionUtility.SESSION_PROJECT_NAME, projectName);
107         String role = projectDAO.getRole((Integer)session.getAttribute(
108             SessionUtility.SESSION_ID), projectID);
109         session.setAttribute(SessionUtility.SESSION_ROLE, role);
110         session.setAttribute(SessionUtility.SESSION_PROJECTSTATUS, projectDAO.
111             getProjectByProjectID((Integer)session.getAttribute(SessionUtility.
112                 SESSION_PROJECT)).getStatus());
113         if(projectDAO.isCurrentSprintExist((Integer)session.getAttribute(

```

```

107         SessionUtility.SESSION_PROJECT)))
108         map.addFlashAttribute("currentSprint", "true");
109         if (role.equalsIgnoreCase(RoleUtility.PO)){
110             return "redirect:viewProductBacklog";
111         }
112         else{
113             return "redirect:viewAllSprint";
114         }
115     }
116 }
117
118 @RequestMapping("/closeProject")
119 public String closeProject(HttpSession session){
120     projectDAO.closeProject((Integer)session.getAttribute(SessionUtility.
121         SESSION_PROJECT));
122     return "redirect:/Project";
123 }
124
125 @RequestMapping("/newProject")
126 public ModelAndView newProject(HttpSession session){
127     ModelAndView newProject = new ModelAndView("add_project");
128     newProject.addObject("newProject", new Project());
129     newProject.addObject("edituser", new User());
130     newProject.addObject("edituser2", new User());
131     return newProject;
132 }
133
134 @RequestMapping("/inviteMembers")
135 public ModelAndView addProject(@ModelAttribute("newProject")
136     Project newProject, BindingResult result, HttpSession session){
137     int projectid = projectDAO.addNewProject(newProject);
138     Member newMember = new Member();
139     newMember.setAccountID((Integer)session.getAttribute(SessionUtility.
140         SESSION_ID));
141     newMember.setProjectID(projectid);
142     newMember.setRole(RoleUtility.SM);
143     projectDAO.addMember(newMember);
144     ModelAndView members = new ModelAndView("new_member");
145     members.addObject("numberOfTM", newProject.getNumberOfTM());
146     members.addObject("numberOfPO", newProject.getNumberOfPO());
147     members.addObject("projectID", projectid);
148     members.addObject("newMember", new Member());
149     members.addObject("edituser", new User());
150     members.addObject("edituser2", new User());
151     ArrayList<User> team_members = new ArrayList<User>();
152     for(int i = 0; i < newProject.getNumberOfTM()+newProject.getNumberOfPO()
153         ; i++){
154         team_members.add(new User());
155     }
156     UserForm teamForm = new UserForm();
157     teamForm.setUsers(team_members);
158     members.addObject("teamForm", teamForm);
159
160     session.setAttribute(SessionUtility.SESSION_PROJECT, projectid);
161     session.setAttribute(SessionUtility.SESSION_PROJECT_NAME, newProject.
162         getProjectName());
163     return members;
164 }
165
166 @RequestMapping("/addMembers")
167 public String addMembers(@ModelAttribute("teamForm")UserForm user, BindingResult
168     result,HttpSession session, RedirectAttributes map){
169     ArrayList<User> team_members = user.getUsers();
170     for(User user2: team_members){
171         System.out.println(user2.getEmail());
172     }
173     EmailNotification invite = new EmailNotification();
174     if(invite.emailInvite(team_members))
175         map.addFlashAttribute("invite","Invitation sent! ");
176
177     userDAO.addInvite(team_members, session);
178
179     map.addFlashAttribute("addproject", "Project was successfully added.");
180     return "redirect:Project";
181 }

```

Listing 35: PublicController.java

```

1 /**
2  * PublicController.java
3  *
4  * Controller for public users
5  *
6  */
7 package com.scrumprimer.controller;
8
9 import org.springframework.context.ApplicationContext;
10 import org.springframework.context.support.ClassPathXmlApplicationContext;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.validation.BindingResult;
13 import org.springframework.web.bind.annotation.ModelAttribute;
14 import org.springframework.web.bind.annotation.RequestMapping;

```

```

15 import org.springframework.web.bind.annotation.RequestMethod;
16 import org.springframework.web.bind.annotation.SessionAttributes;
17 import org.springframework.web.servlet.ModelAndView;
18
19 import com.scrumprimer.model.User;
20
21 @Controller
22 @SessionAttributes
23
24 public class PublicController {
25     @RequestMapping("/Home")
26     public ModelAndView viewHomeRegular(){
27         ModelAndView homeView = new ModelAndView("home_regular");
28         homeView.addObject("login", new User());
29         homeView.addObject("user", new User());
30         return homeView;
31     }
32 }
33

```

Listing 36: RegisterController.java

```

1  /**
2   * RegisterController.java
3   *
4   * Controller for adding new users
5   *
6   */
7  package com.scrumprimer.controller;
8
9  import java.util.ArrayList;
10 import java.util.List;
11
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpSession;
14
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.context.ApplicationContext;
17 import org.springframework.context.support.ClassPathXmlApplicationContext;
18 import org.springframework.security.authentication.encoding.PasswordEncoder;
19 import org.springframework.stereotype.Controller;
20 import org.springframework.ui.Model;
21 import org.springframework.ui.ModelMap;
22 import org.springframework.validation.BindingResult;
23 import org.springframework.web.bind.annotation.ModelAttribute;
24 import org.springframework.web.bind.annotation.RequestMapping;
25 import org.springframework.web.bind.annotation.RequestMethod;
26 import org.springframework.web.bind.annotation.SessionAttributes;
27 import org.springframework.web.servlet.ModelAndView;
28
29
30
31 import org.springframework.web.servlet.config.annotation.EnableWebMvc;
32 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
33
34 import com.scrumprimer.model.Dummy;
35 import com.scrumprimer.model.Member;
36 import com.scrumprimer.model.Notification;
37 import com.scrumprimer.model.NotificationDAO;
38 import com.scrumprimer.model.ProjectDAO;
39 import com.scrumprimer.model.User;
40 import com.scrumprimer.model.UserDAO;
41 import com.scrumprimer.utility.NotificationUtility;
42 import com.scrumprimer.utility.RoleUtility;
43 import com.scrumprimer.utility.SessionUtility;
44
45
46 @Controller
47 @SessionAttributes
48 @EnableWebMvc
49
50 public class RegisterController {
51
52     @Autowired
53     UserDAO userDAO;
54
55     @Autowired
56     private PasswordEncoder passwordEncoder;
57
58     @Autowired
59     private ProjectDAO projectDAO;
60
61     @Autowired
62     private NotificationDAO notifDAO;
63
64
65     @RequestMapping(value = "/addUser", method = RequestMethod.POST)
66     public String addUser(@ModelAttribute("user")
67                           User user, BindingResult result,
68                           RedirectAttributes map) {
69
70         boolean email = userDAO.isEmailExist(user.getEmail());
71         user.setUserType("regular");
72         boolean success = userDAO.insertUser(user);
73
74         if(email){
75             map.addFlashAttribute("errorMessage2", "Email Address is already
76                                 registered");
77         }
78     }
79 }

```

```

75         map.addAttribute("email-taken");
76         map.addFlashAttribute("successmessage", null);
77         map.addFlashAttribute("errormessage", null);
78     }
79     }
80     else if (!success){
81         map.addFlashAttribute("errormessage", "Username is already taken
82         ");
83         map.addAttribute("username-taken");
84         map.addFlashAttribute("successmessage", null);
85         map.addFlashAttribute("errormessage2", null);
86     }
87     }
88     else {
89         map.addFlashAttribute("successmessage", "Account already made");
90         if (userDAO.isEmailInvite(user.getEmail())) {
91             List<Dummy> projectPending = projectDAO.
92                 getPendingProject(user.getEmail());
93             int accountID = userDAO.getUserIDViaEmail(user.getEmail
94                 ());
95             for (Dummy dummy : projectPending){
96                 Member member = new Member();
97                 member.setAccountID(accountID);
98                 member.setProjectID(dummy.getProjectID());
99                 member.setRole(dummy.getRole());
100                projectDAO.addMember(member);
101                Notification notif = new Notification();
102                notif.setAccountID(accountID);
103                notif.setProjectID(dummy.getProjectID());
104                notif.setType(NotificationUtility.NEW_MEMBER);
105                for (Member member2 : projectDAO.projectMembers(
106                    dummy.getProjectID()))
107                    if (member2.getRole().equalsIgnoreCase(
108                        RoleUtility.SM)){
109                        notifDAO.insertNotification(notif,
110                            member2.getAccountID());
111                    }
112                }
113            }
114            map.addFlashAttribute("errormessage2", null);
115            map.addFlashAttribute("errormessage", null);
116        }
117        return "redirect:Home";
118    }
119
120    @RequestMapping(value = "/editUser", method = RequestMethod.POST)
121    public String editUser(@ModelAttribute("edituser")
122        User editUser, BindingResult result, RedirectAttributes map, HttpSession session
123    ) {
124        editUser.setUserID((Integer) session.getAttribute(SessionUtility.
125            SESSION_ID));
126        userDAO.updateUser(editUser);
127
128        User user = userDAO.getUser2(editUser.getUserID());
129
130        session.setAttribute(SessionUtility.USERNAME, user.getUsername());
131        session.setAttribute(SessionUtility.SESSION_ID, user.getUserID());
132        session.setAttribute(SessionUtility.SESSION_FIRSTNAME, user.getFirstName
133            ());
134        session.setAttribute(SessionUtility.SESSION_LASTNAME, user.getLastName()
135            );
136        session.setAttribute(SessionUtility.SESSION_EMAIL, user.getEmail());
137
138        return "redirect:Project";
139    }
140
141    @RequestMapping(value = "/editPassword", method = RequestMethod.POST)
142    public String editPassword(@ModelAttribute("edituser2")
143        User editUser, BindingResult result, RedirectAttributes map, HttpSession session
144    ) {
145        editUser.setUserID((Integer) session.getAttribute(SessionUtility.
146            SESSION_ID));
147        System.out.println(editUser.getNewPassword());
148        userDAO.updatePassword(editUser);
149        session.setAttribute(SessionUtility.PASSWORD, editUser.getNewPassword())
150            ;
151        return "redirect:Project";
152    }
153
154    }
155
156 }

```

Listing 37: ScrumController.java

```

1  /**
2   * ScrumController.java
3   *
4   * Controller that involves the scrum functionalities.
5   */
6  package com.scrumprimer.controller;
7
8  import java.io.InputStream;
9  import java.io.UnsupportedEncodingException;
10 import java.util.ArrayList;

```

```

11 import java.util.List;
12 import java.util.Properties;
13
14 import javax.mail.Authenticator;
15 import javax.mail.Message;
16 import javax.mail.MessagingException;
17 import javax.mail.Multipart;
18 import javax.mail.PasswordAuthentication;
19 import javax.mail.Session;
20 import javax.mail.Transport;
21 import javax.mail.internet.InternetAddress;
22 import javax.mail.internet.MimeBodyPart;
23 import javax.mail.internet.MimeMessage;
24 import javax.mail.internet.MimeMultipart;
25 import javax.servlet.http.HttpSession;
26
27 import org.springframework.beans.factory.annotation.Autowired;
28 import org.springframework.context.ApplicationContext;
29 import org.springframework.context.support.ClassPathXmlApplicationContext;
30 import org.springframework.stereotype.Controller;
31 import org.springframework.ui.ModelMap;
32 import org.springframework.util.AutoPopulatingList;
33 import org.springframework.validation.BindingResult;
34 import org.springframework.web.bind.annotation.ModelAttribute;
35 import org.springframework.web.bind.annotation.RequestMapping;
36 import org.springframework.web.bind.annotation.RequestMethod;
37 import org.springframework.web.bind.annotation.RequestParam;
38 import org.springframework.web.bind.annotation.SessionAttributes;
39 import org.springframework.web.servlet.ModelAndView;
40 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
41
42 import com.scrumprimer.model.Activity;
43 import com.scrumprimer.model.BCModel;
44 import com.scrumprimer.model.Meeting;
45 import com.scrumprimer.model.MeetingForm;
46 import com.scrumprimer.model.Member;
47 import com.scrumprimer.model.Notification;
48 import com.scrumprimer.model.NotificationDAO;
49 import com.scrumprimer.model.Project;
50 import com.scrumprimer.model.ProjectDAO;
51 import com.scrumprimer.model.Review;
52 import com.scrumprimer.model.Sprint;
53 import com.scrumprimer.model.SprintStory;
54 import com.scrumprimer.model.Task;
55 import com.scrumprimer.model.TaskForm;
56 import com.scrumprimer.model.User;
57 import com.scrumprimer.model.UserDAO;
58 import com.scrumprimer.model.UserForm;
59 import com.scrumprimer.model.UserSprintForm;
60 import com.scrumprimer.model.UserStory;
61 import com.scrumprimer.model.UserStoryForm;
62 import com.scrumprimer.utility.DateHandler;
63 import com.scrumprimer.utility.NotificationUtility;
64 import com.scrumprimer.utility.RoleUtility;
65 import com.scrumprimer.utility.SessionUtility;
66 import com.scrumprimer.utility.StatusUtility;
67
68 @Controller
69 @SessionAttributes
70
71 public class ScrumController {
72     @Autowired
73     UserDAO userDAO;
74
75     @Autowired
76     ProjectDAO projectDAO;
77
78     @Autowired
79     NotificationDAO notifDAO;
80
81
82
83     @RequestMapping("/viewAllSprint")
84     public ModelAndView viewAllSprint(HttpSession session){
85         ModelAndView view = new ModelAndView("all_sprint");
86         view.addObject("sprint", "active");
87         view.addObject("edituser", new User());
88         view.addObject("edituser2", new User());
89         int sprintNumber = projectDAO.getLatestSprintNumber(((Integer)session.
90             getAttribute(SessionUtility.SESSION_PROJECT)) + 1);
91         view.addObject("sprintNumber", sprintNumber);
92         view.addObject("newSprint", new Sprint());
93         int notifs = notifDAO.getNumOfNewNotifs(((Integer)session.getAttribute(
94             SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
95             SessionUtility.SESSION_ID));
96         view.addObject("notifs", notifs);
97         view.addObject("project", projectDAO.getProjectByProjectID(((Integer)
98             session.getAttribute(SessionUtility.SESSION_PROJECT)));
99         view.addObject("members", projectDAO.projectMembers(((Integer)session.
100             getAttribute(SessionUtility.SESSION_PROJECT)));
101         if(projectDAO.isCurrentSprintExist(((Integer)session.getAttribute(
102             SessionUtility.SESSION_PROJECT))){
103             view.addObject("currentSprint", "yes");
104         }
105         if(!projectDAO.isUserStoryExisting(((Integer)session.getAttribute(
106             SessionUtility.SESSION_PROJECT))){
107             view.addObject("nostories", "No User Stories");
108         }
109         if(projectDAO.isSprintExist(((Integer)session.getAttribute(SessionUtility

```

```

103         .SESSION_PROJECT))){
104             List<Sprint> allSprints = projectDAO.getAllSprint((Integer)
105                 session.getAttribute(SessionUtility.SESSION_PROJECT));
106             view.addObject("allSprints", allSprints );
107         }
108         else{
109             view.addObject("nosprints", "No Sprints");
110         }
111         return view;
112     }
113
114
115     @RequestMapping("/viewProductBacklog")
116     public ModelAndView viewProductBacklog(HttpSession session){
117         ModelAndView view = new ModelAndView("product_backlog");
118         view.addObject("productbacklog", "active");
119         view.addObject("edituser", new User());
120         view.addObject("edituser2", new User());
121         int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
122             SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
123             SessionUtility.SESSION_ID));
124         view.addObject("notifs", notifs);
125         view.addObject("project", projectDAO.getProjectByProjectID((Integer)
126             session.getAttribute(SessionUtility.SESSION_PROJECT)));
127         view.addObject("members", projectDAO.getProjectMembers((Integer)session.
128             getAttribute(SessionUtility.SESSION_PROJECT)));
129         if(projectDAO.isUserStoryExisting((Integer)session.getAttribute(
130             SessionUtility.SESSION_PROJECT)){
131             List<UserStory> stories = projectDAO.getUserStory((Integer)
132                 session.getAttribute(SessionUtility.SESSION_PROJECT));
133             view.addObject("stories", stories);
134         }
135         else{
136             view.addObject("nostory", "Empty Product Backlog");
137         }
138         return view;
139     }
140
141     @RequestMapping("/editProductBacklog")
142     public ModelAndView editProductBacklog(HttpSession session){
143         ModelAndView view = new ModelAndView("edit_product_backlog");
144         view.addObject("productbacklog", "active");
145         view.addObject("edituser", new User());
146         view.addObject("edituser2", new User());
147         ArrayList<UserStory> stories = new ArrayList<UserStory>();
148         UserStoryForm storyForm = new UserStoryForm();
149         int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
150             SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
151             SessionUtility.SESSION_ID));
152         view.addObject("notifs", notifs);
153         view.addObject("project", projectDAO.getProjectByProjectID((Integer)
154             session.getAttribute(SessionUtility.SESSION_PROJECT)));
155         view.addObject("members", projectDAO.getProjectMembers((Integer)session.
156             getAttribute(SessionUtility.SESSION_PROJECT)));
157         if(projectDAO.isUserStoryExisting((Integer)session.getAttribute(
158             SessionUtility.SESSION_PROJECT)){
159             stories = (ArrayList<UserStory>) projectDAO.getUserStory((
160                 Integer)session.getAttribute(SessionUtility.SESSION_PROJECT
161                 ));
162         }
163         storyForm.setStories(stories);
164         view.addObject("storyForm", storyForm);
165         return view;
166     }
167
168     @RequestMapping("/saveProductBacklog")
169     public String saveProductBacklog(HttpSession session, @ModelAttribute("storyForm
170 ") UserStoryForm storyForm, BindingResult result){
171         ArrayList<UserStory> stories = storyForm.getStories();
172         for(UserStory story : stories){
173             System.out.println(story.getStoryID());
174         }
175         projectDAO.updateUserStories(stories);
176         return "redirect:viewProductBacklog";
177     }
178
179     @RequestMapping("/insertUserStories")
180     public String insertUserStories(HttpSession session, @ModelAttribute("newStory")
181 UserStoryForm newStory, BindingResult result){
182         ArrayList<UserStory> stories = newStory.getStories();
183         projectDAO.insertUserStories(stories);
184         return "redirect:viewProductBacklog";
185     }
186
187     @RequestMapping("/addUserStories")
188     public ModelAndView addUserStories(HttpSession session, @RequestParam("
189 numberOfStories")int story){
190         ModelAndView view = new ModelAndView("add_product_backlog");
191         view.addObject("productbacklog", "active");
192         view.addObject("edituser", new User());
193         view.addObject("edituser2", new User());
194         view.addObject("numberOfStory", story);
195         ArrayList<UserStory> newStories = new ArrayList<UserStory>();
196         for(int i = 0; i < story; i++){

```



```

184         newStories.add(new UserStory());
185     }
186     UserStoryForm newUserStories = new UserStoryForm();
187     newUserStories.setStories(newStories);
188     view.addObject("newStory", newUserStories);
189     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
190     view.addObject("notifs", notifs);
191     view.addObject("project", projectDAO.getProjectByProjectID((Integer)
        session.getAttribute(SessionUtility.SESSION_PROJECT)));
192     view.addObject("members", projectDAO.projectMembers((Integer)session.
        getAttribute(SessionUtility.SESSION_PROJECT)));
193     return view;
194 }
195
196 @RequestMapping("/viewSprintReview")
197 public ModelAndView viewSprintReview(HttpSession session, @RequestParam("sid")
    int sprintID){
198     ModelAndView view = new ModelAndView("sprint_review_view");
199     view.addObject("edituser", new User());
200     view.addObject("edituser2", new User());
201     view.addObject("sprintID", sprintID);
202     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
203     view.addObject("notifs", notifs);
204     view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
        getSprintNumber());
205     view.addObject("project", projectDAO.getProjectByProjectID((Integer)
        session.getAttribute(SessionUtility.SESSION_PROJECT)));
206     view.addObject("members", projectDAO.projectMembers((Integer)session.
        getAttribute(SessionUtility.SESSION_PROJECT)));
207     List<Review> reviews = projectDAO.getSprintReview(sprintID);
208     for(Review review : reviews){
209         review.setStoryName(projectDAO.getUserStoryByID(review.
            getStoryID()).getStoryName());
210         review.setStatus(projectDAO.getUserStoryByID(review.getStoryID()
            ).getStatus());
211     }
212     view.addObject("reviews", reviews);
213     return view;
214 }
215
216 @RequestMapping("/sprintReview")
217 public ModelAndView sprintReview(HttpSession session, @RequestParam("sid")int
    sprintID){
218     ModelAndView view = new ModelAndView("sprint_review");
219     view.addObject("edituser", new User());
220     view.addObject("edituser2", new User());
221     view.addObject("sprintID", sprintID);
222     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
223     view.addObject("notifs", notifs);
224     view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
        getSprintNumber());
225     MeetingForm reviews = new MeetingForm();
226     List<UserStory> stories = projectDAO.getSprintStories(sprintID);
227     ArrayList<Review> review = new ArrayList<Review>();
228     for(int i = 0; i < stories.size(); i++){
229         Review review2 = new Review();
230         review2.setStoryID(stories.get(i).getStoryID());
231         review2.setStoryName(stories.get(i).getStoryName());
232         review.add(review2);
233     }
234     reviews.setReviews(review);
235     view.addObject("reviewForm", reviews);
236     view.addObject("stories", stories);
237
238     return view;
239 }
240
241 @RequestMapping("/addReview")
242 public String addReview(@ModelAttribute("reviewForm")MeetingForm reviewForm,
    BindingResult result, HttpSession session, @RequestParam("sid")int sprintID
    ){
243     List<Review> reviews = reviewForm.getReviews();
244     for(Review review : reviews){
245         projectDAO.updateStoryStatus(review.getStatus(), review.
            getStoryID());
246         projectDAO.insertSprintReview(review);
247     }
248     projectDAO.closeSprint(sprintID);
249     return "redirect:/viewAllSprint";
250 }
251
252 @RequestMapping("/viewSprintBacklog")
253 public ModelAndView viewSprintBacklog(HttpSession session, @RequestParam("sid")
    int sprintID){
254     ModelAndView view = new ModelAndView("sprint_backlog");
255     notifSprintReview();
256     view.addObject("edituser", new User());
257     view.addObject("edituser2", new User());
258     view.addObject("sprintID", sprintID);
259     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
260     view.addObject("notifs", notifs);

```

```

261         view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
262             getSprintNumber());
263         view.addObject("sprintstatus", projectDAO.getSprintBySprintID(sprintID).
264             getStatus());
265         view.addObject("project", projectDAO.getProjectByProjectID((Integer)
266             session.getAttribute(SessionUtility.SESSION_PROJECT)));
267         view.addObject("members", projectDAO.getProjectMembers((Integer)session.
268             getAttribute(SessionUtility.SESSION_PROJECT)));
269         view.addObject("remainingdays", DateHandler.getDaysDifference(
270             DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(
271             sprintID).getSprintReview()));
272         view.addObject("newTask", new Task());
273         List<UserStory> sprintStories = projectDAO.getSprintStories(sprintID);
274         view.addObject("sprintstories", sprintStories);
275         if (DateHandler.getDaysDifference(DateHandler.getCurrentDate(),
276             projectDAO.getSprintBySprintID(sprintID).getSprintReview()) == 0){
277             view.addObject("sprintreview", "yes");
278         }
279         if (!projectDAO.isTaskExisting(sprintID)){
280             view.addObject("pokerplanning", "yes");
281         }
282         else{
283             List<Task> tasks = projectDAO.getTasks(sprintID);
284             for (Task task : tasks){
285                 //System.out.println(task.getStoryID());
286                 task.setStoryName(projectDAO.getUserStoryByID(task.
287                     getStoryID()).getStoryName());
288                 task.setVolunteer(userDAO.getUser2(task.getAccountID()).
289                     getFirstName());
290                 List<Activity> activities = projectDAO.getAllActivities(
291                     task.getTaskID());
292                 int sum = 0;
293                 for (Activity act : activities){
294                     sum += act.getTime();
295                 }
296                 task.setActual(sum);
297             }
298             view.addObject("tasks", tasks);
299         }
300         return view;
301     }
302
303     @RequestMapping("/addTask")
304     public String addTask(@ModelAttribute("newTask") Task task, BindingResult result,
305         HttpSession session, RedirectAttributes map){
306         task.setStatus(StatusUtility.NOT_STARTED);
307         List<Task> tasks = new ArrayList<Task>();
308         tasks.add(task);
309         projectDAO.insertTasks(tasks);
310         map.addFlashAttribute("taskadded", "New tasks has been added");
311         return "redirect:viewSprintBacklog?sid="+task.getSprintID();
312     }
313
314     @RequestMapping("/viewScrumMeeting")
315     public ModelAndView viewScrumMeeting(HttpSession session, @RequestParam("sid")
316         int sprintID){
317         ModelAndView view = new ModelAndView("scrummeeting");
318         view.addObject("edituser", new User());
319         view.addObject("edituser2", new User());
320         view.addObject("sprintID", sprintID);
321         int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
322             SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
323             SessionUtility.SESSION_ID));
324         view.addObject("notifs", notifs);
325         view.addObject("sprintNumber", projectDAO.getSprintBySprintID(sprintID).
326             getSprintNumber());
327         view.addObject("meeting", new Meeting());
328         view.addObject("sprintstatus", projectDAO.getSprintBySprintID(sprintID).
329             getStatus());
330         view.addObject("project", projectDAO.getProjectByProjectID((Integer)
331             session.getAttribute(SessionUtility.SESSION_PROJECT)));
332         view.addObject("members", projectDAO.getProjectMembers((Integer)session.
333             getAttribute(SessionUtility.SESSION_PROJECT)));
334         view.addObject("remainingdays", DateHandler.getDaysDifference(
335             DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(
336             sprintID).getSprintReview()));
337
338         if (projectDAO.isScrumMeeting(sprintID)){
339             List<Meeting> meetings = projectDAO.getScrumMeeting(sprintID);
340             view.addObject("meetings", meetings);
341         }
342         if (!projectDAO.isCurrentScrumMeeting(sprintID)){
343             view.addObject("currentmeeting", "no");
344         }
345         if (!projectDAO.isScrumMeetingAnswered(sprintID, (Integer)session.
346             getAttribute(SessionUtility.SESSION_ID))){
347             view.addObject("fillmeeting", "no");
348         }
349         return view;
350     }
351
352     @RequestMapping("/viewScrumMeetingDetail")
353     public ModelAndView viewScrumMeetingDetail(HttpSession session, @RequestParam("
354         date")String date, @RequestParam("sid")int sprintID){
355         ModelAndView view = new ModelAndView("scrummeeting_detail");
356         view.addObject("edituser", new User());
357         view.addObject("edituser2", new User());

```

```

338         view.addObject(" sprintID", sprintID);
339         view.addObject(" date", date);
340         int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
            SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
            SessionUtility.SESSION_ID));
341         view.addObject(" notifs", notifs);
342         List<Meeting> details = projectDAO.getMeetingDetail(sprintID, date);
343         for(Meeting det : details){
344             det.setMember(userDAO.getUser2(det.getAccountID()).getFirstName
                ());
345         }
346         view.addObject(" details", details);
347         view.addObject(" project", projectDAO.getProjectByProjectID((Integer)
            session.getAttribute(SessionUtility.SESSION_PROJECT)));
348         view.addObject(" members", projectDAO.projectMembers((Integer)session.
            getAttribute(SessionUtility.SESSION_PROJECT)));
349         return view;
350     }
351
352     @RequestMapping("/addScrumMeeting")
353     public String addScrumMeeting(HttpSession session, @ModelAttribute(" meeting")
        Meeting meeting, BindingResult result){
354         meeting.setDate(DateHandler.getCurrentDate());
355         meeting.setAccountID((Integer)session.getAttribute(SessionUtility.
            SESSION_ID));
356         projectDAO.insertScrumMeeting(meeting);
357         Notification notif = new Notification();
358         notif.setProjectID((Integer)session.getAttribute(SessionUtility.
            SESSION_PROJECT));
359         notif.setType(NotificationUtility.SCRUM_MEETING);
360         notif.setSprintID(meeting.getSprintID());
361         if(session.getAttribute(SessionUtility.SESSION_ROLE).equals(RoleUtility.
            SM)){
362             for(Member member : projectDAO.projectMembers((Integer)session.
                getAttribute(SessionUtility.SESSION_PROJECT))){
363                 if(member.getRole().equalsIgnoreCase(RoleUtility.TM))
364                     notifDAO.insertNotification(notif, member.getAccountID()
                        ());
365             }
366         }
367         return "redirect:/viewScrumMeeting?sid="+meeting.getSprintID();
368     }
369
370     @RequestMapping("/viewScrumBoard")
371     public ModelAndView viewSprint(HttpSession session, @RequestParam(" sid")int
        sprintID){
372         ModelAndView view = new ModelAndView(" sprint");
373         view.addObject(" edituser", new User());
374         view.addObject(" edituser2", new User());
375         view.addObject(" sprintID", sprintID);
376         int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
            SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
            SessionUtility.SESSION_ID));
377         view.addObject(" notifs", notifs);
378         view.addObject(" sprintNumber", projectDAO.getSprintBySprintID(sprintID).
            getSprintNumber());
379         Sprint sprint = projectDAO.getSprintBySprintID(sprintID);
380         String date2 = sprint.getDateStarted();
381         sprint.setDateStarted(date2.substring(0,10));
382         view.addObject(" sprint", sprint);
383         view.addObject(" project", projectDAO.getProjectByProjectID((Integer)
            session.getAttribute(SessionUtility.SESSION_PROJECT)));
384         view.addObject(" members", projectDAO.projectMembers((Integer)session.
            getAttribute(SessionUtility.SESSION_PROJECT)));
385         view.addObject(" remainingdays", DateHandler.getDaysDifference(
            DateHandler.getCurrentDate(), projectDAO.getSprintBySprintID(
            sprintID).getSprintReview()));
386         if(!projectDAO.isTaskExisting(sprintID)){
387             view.addObject(" pokerplanning", "yes");
388         }
389         else{
390             List<Task> tasks = projectDAO.getTasks(sprintID);
391             List<BCModel> burnvalues = new ArrayList<BCModel>();
392
393             for(Task task : tasks){
394                 //System.out.println(task.getStoryID());
395                 task.setStoryName(projectDAO.getUserStoryByID(task.
                    getStoryID()).getStoryName());
396                 task.setVolunteer(userDAO.getUser2(task.getAccountID()).
                    getFirstName());
397             }
398             int total = projectDAO.getTotalEstimate(sprintID);
399             String date;
400             if(DateHandler.getDaysDifference(DateHandler.getCurrentDate(),
                sprint.getSprintReview()) == 0){
401                 date = sprint.getSprintReview();
402             }
403             else{
404                 date = DateHandler.getCurrentDate();
405             }
406             for(int i = 0; i <= DateHandler.getDaysDifference(sprint.
                getDateStarted(), date); i++){
407                 BCModel burn = new BCModel();
408                 burn.setDay(i);
409                 int sum = 0;
410                 for(Task task : tasks){
411                     sum += projectDAO.getTotalActivityPerDay(task.
                        getTaskID(), DateHandler.getDayAdded(sprint.
                            getDateStarted(), i));

```

```

412     }
413     total = total - sum;
414     burn.setTotal(total);
415     burnvalues.add(burn);
416 }
417 String burns = "["+burnvalues.get(0).getDay()+","+burnvalues.get
418 (0).getTotal()+"]";
419 if(burnvalues.size() > 1){
420 for(int i = 1; i < burnvalues.size(); i++){
421     String concat = "["+burnvalues.get(i).getDay()+","+
422     burnvalues.get(i).getTotal()+"]";
423     burns = burns + concat;
424 }
425 }
426 view.addObject("tasks",tasks);
427 view.addObject("burnvalues", burns);
428 int totalDays = DateHandler.getDaysDifference(sprint.
429     getDateStarted(), sprint.getSprintReview());
430 view.addObject("totalDays", totalDays);
431 view.addObject("totalEstimate", projectDAO.getTotalEstimate(
432     sprintID));
433 }
434 return view;
435 }
436
437 @ModelAttribute("taskForm")
438 public TaskForm getTaskForm(){
439     TaskForm taskForm = new TaskForm();
440     taskForm.setTasks(new AutoPopulatingList<Task>(Task.class));
441     return taskForm;
442 }
443
444 @RequestMapping("/pokerPlanning")
445 public ModelAndView pokerPlanning(HttpSession session, @RequestParam("sid")int
446     sprintID, @RequestParam("rmng") int remaining){
447     ModelAndView view = new ModelAndView("poker_planning");
448     view.addObject("edituser", new User());
449     view.addObject("edituser2", new User());
450     view.addObject("sprintID",sprintID);
451     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
452     SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
453     SessionUtility.SESSION_ID));
454     view.addObject("notifs",notifs);
455     TaskForm taskForm = new TaskForm();
456     taskForm.setTasks(new AutoPopulatingList<Task>(Task.class));
457     view.addObject("taskForm", taskForm);
458     List<UserStory> stories = projectDAO.getSprintStories(sprintID);
459     List<Member> members = projectDAO.projectMembers((Integer) session.
460     getAttribute(SessionUtility.SESSION_PROJECT));
461     for(Member member : members){
462         member.setName(userDAO.getUser2(member.getAccountID()).
463             getFirstName());
464     }
465     view.addObject("members", members);
466     view.addObject("story", stories.get(remaining));
467     view.addObject("ts", stories.size());
468     view.addObject("remaining", remaining+1);
469     view.addObject("project",projectDAO.getProjectByProjectID((Integer)
470     session.getAttribute(SessionUtility.SESSION_PROJECT)));
471     view.addObject("members", projectDAO.projectMembers((Integer)session.
472     getAttribute(SessionUtility.SESSION_PROJECT)));
473
474     return view;
475 }
476
477 @RequestMapping("/addTasks")
478 public String addTasks(HttpSession session, @ModelAttribute("taskForm") TaskForm
479     taskForm, BindingResult result, RedirectAttributes map, @RequestParam("
480     rmng") int remainingStory, @RequestParam("ts") int totalStory,
481     @RequestParam("sid")int sprintID ){
482     List<Task> tasks = taskForm.getTasks();
483     List<UserStory> stories = projectDAO.getSprintStories(sprintID);
484     for(Task task : tasks){
485         task.setSprintID(sprintID);
486         task.setStatus(StatusUtility.NOT_STARTED);
487         task.setStoryID(stories.get(remainingStory-1).getStoryID());
488     }
489     projectDAO.insertTasks(tasks);
490     if(remainingStory < totalStory){
491         return "redirect:/pokerPlanning?sid="+sprintID+"&rmng="+
492             remainingStory;
493     }
494     else{
495         return "redirect:/viewSprintBacklog?sid="+sprintID;
496     }
497 }
498
499 @RequestMapping(method = RequestMethod.GET, value="/appendStoryView")
500 protected String appendStoryField(@RequestParam Integer fieldId, ModelMap model,
501     HttpSession session)
502 {
503     model.addAttribute("storyNumber", fieldId);
504     List<Member> members = projectDAO.projectMembers((Integer) session.

```

```

        getAttribute(SessionUtility.SESSION_PROJECT));
495     for (Member member : members){
496         member.setName(userDAO.getUser2(member.getAccountID()).
            getFirstName());
497         System.out.println(member.getName());
498     }
499     model.addAttribute("members", members);
500     int sprintID = projectDAO.getLatestSprintNumber((Integer)session.
        getAttribute(SessionUtility.SESSION_PROJECT));
501     model.addAttribute("sprintID", sprintID);
502
503
504     return "addTasksRow";
505 }
506
507
508
509
510 @RequestMapping("/addSprint")
511 public ModelAndView addSprint(HttpSession session, @ModelAttribute("newSprint")
    Sprint newSprint, BindingResult result){
512     projectDAO.insertSprint(newSprint);
513     System.out.println(newSprint.getSprintReview());
514     Sprint sprint = projectDAO.getCurrentSprint((Integer) session.
        getAttribute(SessionUtility.SESSION_PROJECT));
515     ModelAndView view = new ModelAndView("add_sprint");
516     view.addObject("edituser", new User());
517     view.addObject("edituser2", new User());
518     List<UserStory> userStories = projectDAO.getOngoingUserStory((Integer)
        session.getAttribute(SessionUtility.SESSION_PROJECT));
519     view.addObject("userStories", userStories);
520     view.addObject("sprintID", sprint.getSprintID());
521     view.addObject("sprintNumber", newSprint.getSprintNumber());
522     UserSprintForm storyForm = new UserSprintForm();
523     ArrayList<SprintStory> sprintStories = new ArrayList<SprintStory>();
524     for (int i = 0; i < userStories.size(); i++){
525         SprintStory sprintStory = new SprintStory();
526         sprintStories.add(sprintStory);
527     }
528     storyForm.setSprintStories(sprintStories);
529     view.addObject("sprintForm", storyForm);
530     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
531     view.addObject("notifs", notifs);
532     view.addObject("project", projectDAO.getProjectByProjectID((Integer)
        session.getAttribute(SessionUtility.SESSION_PROJECT)));
533     view.addObject("members", projectDAO.getProjectMembers((Integer)session.
        getAttribute(SessionUtility.SESSION_PROJECT)));
534
535     return view;
536 }
537
538
539 @RequestMapping("/insertSprintStories")
540 public String insertSprintStories(HttpSession session, @ModelAttribute("
    sprintForm")UserSprintForm sprintForm, BindingResult result,
    RedirectAttributes map){
541     projectDAO.insertSprintStories(sprintForm.getSprintStories());
542     map.addFlashAttribute("addsprintstory", "New Sprint Added!");
543     return "redirect:viewAllSprint";
544 }
545
546 @RequestMapping("/viewActivity")
547 public ModelAndView viewActivity(HttpSession session, @RequestParam("tid")int
    taskID){
548     ModelAndView view = new ModelAndView("activity");
549     Task task = projectDAO.getTaskViaTaskID(taskID);
550     view.addObject("edituser", new User());
551     view.addObject("edituser2", new User());
552     task.setVolunteer(userDAO.getUser2(task.getAccountID()).getFirstName());
553     task.setStoryName(projectDAO.getUserStoryByID(task.getStoryID()).
        getStoryName());
554     view.addObject("sprintstatus", projectDAO.getSprintBySprintID(task.
        getSprintID()).getStatus());
555     view.addObject("task", task);
556     view.addObject("datetoday", DateHandler.getCurrentDate());
557     String datestarted = projectDAO.getSprintBySprintID(task.getSprintID()).
        getDateStarted().substring(0, 10);
558     view.addObject("datestarted", datestarted);
559     if (!projectDAO.isActivityExisting(taskID)){
560         view.addObject("noactivity", "yes");
561     }
562     else {
563         List<Activity> activities = projectDAO.getAllActivities(taskID);
564         for (Activity act : activities){
565             act.setTaskStatus(task.getStatus());
566             act.setDoer(userDAO.getUser2(act.getAccountID()).
                getFirstName());
567         }
568         view.addObject("activities", activities);
569     }
570     view.addObject("newActivity", new Activity());
571     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
        SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
        SessionUtility.SESSION_ID));
572     view.addObject("notifs", notifs);
573     view.addObject("project", projectDAO.getProjectByProjectID((Integer)
        session.getAttribute(SessionUtility.SESSION_PROJECT)));

```

```

574         view.addObject("members", projectDAO.projectMembers((Integer)session .
575             getAttribute(SessionUtility.SESSION_PROJECT)));
576     }
577     return view;
578 }
579
580 @RequestMapping("/addActivity")
581 public String addActivity(@ModelAttribute("newActivity") Activity activity ,
582     BindingResult result , HttpSession session){
583     projectDAO.insertActivity(activity);
584     projectDAO.updateTaskStatus(activity.getTaskStatus(), activity.getTaskID
585         ());
586     return "redirect:viewActivity?tid="+activity.getTaskID();
587 }
588
589 @RequestMapping("/viewNotification")
590 public ModelAndView viewNotification(HttpSession session){
591     ModelAndView view = new ModelAndView("notification");
592     List<Notification> notification = notifDAO.getNotification((Integer)
593         session.getAttribute(SessionUtility.SESSION_PROJECT), (Integer)
594         session.getAttribute(SessionUtility.SESSION_ID));
595     for(Notification not : notification){
596         if(not.getType().equals(NotificationUtility.NEW_MEMBER))
597             not.setNewMember(userDAO.getUser2(not.getAccountID()).
598                 getFirstName());
599     }
600     view.addObject("edituser", new User());
601     view.addObject("edituser2", new User());
602     view.addObject("notification", "active");
603     view.addObject("notifications", notification);
604     int notifs = notifDAO.getNumOfNewNotifs((Integer)session.getAttribute(
605         SessionUtility.SESSION_PROJECT), (Integer)session.getAttribute(
606         SessionUtility.SESSION_ID));
607     view.addObject("notifs", notifs);
608     view.addObject("project", projectDAO.getProjectByProjectID((Integer)
609         session.getAttribute(SessionUtility.SESSION_PROJECT)));
610     view.addObject("members", projectDAO.projectMembers((Integer)session .
611         getAttribute(SessionUtility.SESSION_PROJECT)));
612     return view;
613 }
614
615 @RequestMapping("/notifSprintReview")
616 public void notifSprintReview(){
617     List<Sprint> openSprint = projectDAO.getAllOpenSprint();
618     for(Sprint sprint : openSprint){
619         if(DateHandler.getDaysDifference(DateHandler.getCurrentDate(),
620             sprint.getSprintReview()) <= 1){
621             Notification notif = new Notification();
622             notif.setProjectID(sprint.getProjectID());
623             notif.setType(NotificationUtility.SPRINT_REVIEW);
624             notif.setSprintID(sprint.getSprintID());
625             for(Member member : projectDAO.projectMembers(sprint .
626                 getProjectID())){
627                 if(!notifDAO.isSprintReview(sprint.getSprintID()))
628                     notifDAO.insertNotification(notif, member .
629                         getAccountID());
630             }
631         }
632     }
633 }
634 }
635 }
636 }

```

3. Utilities

Listing 38: DateHandler.java

```

1  /**
2   * DateHandler.java
3   */
4  package com.scrumprimer.utility;
5
6  import java.text.DateFormat;
7  import java.text.ParseException;
8  import java.text.SimpleDateFormat;
9  import java.util.Calendar;
10 import java.util.Date;
11 import java.util.List;
12
13
14 public class DateHandler
15 {
16     /**
17     * Date today in Calendar format
18     */
19     private static Calendar TODAY = Calendar.getInstance();
20
21
22     /**
23     * Format for parsing date Strings
24     */
25     private final static SimpleDateFormat DATE_FORMAT = new SimpleDateFormat("yyyy-MM-
    dd");

```

```

26
27
28 /**
29  * Subtracts earlier date from today
30  *
31  * @param earlierDateStr Previous date in String format
32  * @return Working days passed from earlierDateStr until today
33  */
34
35 public static String getCurrentDate()
36 {
37     String curDate = DATEFORMAT.format( TODAY.getTime() );
38     return curDate;
39 }
40
41 public static int getDaysDifference( String earlierDateStr )
42 {
43
44     int daysDifference = 0;
45
46     if( !earlierDateStr.isEmpty() )
47     {
48         Calendar earlierDate = Calendar.getInstance();
49         try {
50             earlierDate.setTime( DATEFORMAT.parse( earlierDateStr )
51                                 );
52             } catch (ParseException e) {
53             }
54
55         //-1 to take account the excess milliseconds passed from today, check in
56         calculateWorkingDaysDifference() while loop
57         daysDifference = calculateWorkingDaysDifference( earlierDate, TODAY ) - 1;
58     }
59     if(daysDifference < 0)
60     {
61         daysDifference = 0;
62     }
63
64     return daysDifference;
65 }
66
67 /**
68  * @param d1 string of date to be compared
69  * @param d2 string of date to be compared
70  * @return true if String d1 is earlier than d2
71  */
72 public static boolean isEarlier(String d1, String d2)
73 {
74     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
75     Date date1 = null;
76     try {
77         date1 = sdf.parse(d1);
78     } catch (ParseException e) {
79     }
80
81     Date date2 = null;
82     try {
83         date2 = sdf.parse(d2);
84     } catch (ParseException e) {
85     }
86
87
88     if( date1.compareTo( date2 ) >= 0 )
89     {
90         return false;
91     }
92     else return true;
93 }
94
95 public static String getDayAdded(String dateStr, int num){
96     if(!dateStr.isEmpty()){
97         Calendar date = Calendar.getInstance();
98         try {
99             date.setTime(DATEFORMAT.parse(dateStr));
100
101             } catch (ParseException e) {
102             // TODO Auto-generated catch block
103             e.printStackTrace();
104             }
105         date.add(Calendar.DAY_OF_MONTH, num);
106         return DATEFORMAT.format(date.getTime());
107     }
108     else{
109         return null;
110     }
111 }
112
113
114 /**
115  * overloaded, subtracts earlier date from later date
116  *
117  * @param earlierDateStr Previous date in String format
118  * @param laterDateStr Later date in String format
119  * @return Working days passed from earlierDateStr until laterDateStr
120  */
121
122 public static int getDaysDifference( String earlierDateStr, String laterDateStr )

```

```

123     {
124         int daysDifference = 0;
125
126         if( !earlierDateStr.isEmpty() && !laterDateStr.isEmpty() )
127         {
128             Calendar earlierDate = Calendar.getInstance();
129             Calendar laterDate = Calendar.getInstance();
130             try {
131                 earlierDate.setTime( DATEFORMAT.parse( earlierDateStr )
132                                     );
133                 laterDate.setTime( DATEFORMAT.parse( laterDateStr ) );
134             } catch (ParseException e) {
135             }
136
137
138             daysDifference = calculateWorkingDaysDifference( earlierDate, laterDate );
139         }
140     }
141     return daysDifference;
142 }
143
144 /**
145  * @param earlierDate Previous date in String format
146  * @param laterDate Later date in String format
147  * @return Total working days between dates
148  */
149 private static int calculateWorkingDaysDifference( Calendar earlierDate, Calendar
150     laterDate )
151 {
152     int workDays = 0;
153
154     SimpleDateFormat date = new SimpleDateFormat( "MM-dd" );
155
156     while( earlierDate.getTimeInMillis() < laterDate.getTimeInMillis() )
157     {
158         //System.out.println( DATEFORMAT.format( earlierDate.getTime() ) );
159         workDays++;
160
161         earlierDate.add( Calendar.DAY_OF_MONTH, 1 );
162     }
163     return workDays;
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }

```

Listing 39: EmailNotification.java

```

1  /**
2   * EmailNotification.java
3   */
4  package com.scrumprimer.utility;
5
6  import java.io.InputStream;
7  import java.util.ArrayList;
8  import java.util.Properties;
9
10 import javax.mail.Authenticator;
11 import javax.mail.PasswordAuthentication;
12 import javax.mail.Session;
13 import javax.mail.Authenticator;
14 import javax.mail.Message;
15 import javax.mail.MessagingException;
16 import javax.mail.Multipart;
17 import javax.mail.PasswordAuthentication;
18 import javax.mail.Session;
19 import javax.mail.Transport;
20 import javax.mail.internet.InternetAddress;
21 import javax.mail.internet.MimeBodyPart;
22 import javax.mail.internet.MimeMessage;
23 import javax.mail.internet.MimeMultipart;
24
25 import com.scrumprimer.controller.ProjectController;
26 import com.scrumprimer.model.User;
27
28 public class EmailNotification {
29
30     private static Properties properties = null;
31
32     static {
33
34         try
35         {
36             properties = new Properties();
37             InputStream inputStream = ProjectController.class.getClassLoader().
38                 getResourceAsStream( "mail.properties" );
39             properties.load(inputStream);
40         }
41         catch( Exception e )

```



```

41     {
42         e.printStackTrace();
43     }
44 }
45
46 public static boolean emailInvite( ArrayList<User> team_members){
47     try{
48         String subject = "Scrum Primer Project Invitation";
49         Session session1 = Session.getInstance( properties, new
50             SocialAuth() );
51         Message msg = new MimeMessage( session1 );
52         InternetAddress from = new InternetAddress( ( String ) properties.get( "
53             mail.login.username" );
54             ( String ) properties.get( "mail.login.fromname" ) );
55         msg.setFrom( from );
56         InternetAddress[] toAddresses = new InternetAddress[team_members.size()
57             ];
58         int i = 0;
59         for(User user2: team_members){
60             toAddresses[i] = new InternetAddress(user2.getEmail());
61             i++;
62         }
63         msg.setRecipients( Message.RecipientType.TO, toAddresses );
64         msg.setSubject( subject );
65
66         MimeBodyPart htmlPart = new MimeBodyPart();
67         String body = "<center>Register or login <a href = " + "http://localhost
68             :8080/ScrumPrimer/" + ">here to be part of the team </a>";
69         htmlPart.setContent( body, "text/html" );
70         Multipart multipart = new MimeMultipart();
71         multipart.addBodyPart( htmlPart );
72         msg.setContent( multipart );
73         Transport.send( msg );
74         return true;
75     } catch( Exception e ){
76         return false;
77     }
78 }
79
80 static class SocialAuth extends Authenticator
81 {
82     /**
83     * @return
84     * @see javax.mail.Authenticator#getPasswordAuthentication()
85     */
86     @SuppressWarnings( "javadoc" )
87     @Override
88     protected PasswordAuthentication getPasswordAuthentication()
89     {
90         return new PasswordAuthentication( ( String ) properties.get( "mail.login.
91             username" ),
92             ( String ) properties.get( "mail.login.password" ) );
93     }
94 }
95 }
96 }

```

Listing 40: NotificationUtility.java

```

1  /**
2   * NotificationUtility.java
3   */
4  package com.scrumprimer.utility;
5
6  public class NotificationUtility {
7      public static final String NEW_MEMBER = "new_member";
8      public static final String SPRINT_REVIEW = "sprint_review";
9      public static final String SCRUM_MEETING = "scrum_meeting";
10 }
11 }

```

Listing 41: RoleUtility.java

```

1  /**
2   * RoleUtility.java
3   */
4  package com.scrumprimer.utility;
5
6  public class RoleUtility {
7      public static final String SM = "scrum_master";
8      public static final String TM = "team_member";
9      public static final String PO = "product_owner";
10 }
11 }

```

Listing 42: SessionUtility.java

```

1  /**
2   * SessionUtility.java

```

```

3  */
4  package com.scrumprimer.utility;
5
6  public class SessionUtility {
7
8      public static final String SESSION_FIRSTNAME = "sessionName";
9
10     public static final String SESSION_LASTNAME = "sessionLast";
11
12     public static final String SESSION_ID = "accountID";
13
14     public static final String SESSION_ROLE = "role";
15
16     public static final String SESSION_PROJECT = "projectID";
17
18     public static final String SESSION_PROJECT_NAME = "projectName";
19
20     public static final String USERNAME = "username";
21
22     public static final String PASSWORD = "password";
23
24     public static final String SESSION_EMAIL = "sessionEmail";
25
26     public static final String SPRINT = "sprintNumber";
27
28     public static final String SESSION_PROJECTSTATUS = "status";
29
30     public static final String SESSION_USERTYPE = "userType";
31
32 }
33

```

Listing 43: StatusUtility.java

```

1  /**
2   * StatusUtility.java
3   */
4  package com.scrumprimer.utility;
5
6  public class StatusUtility {
7      public static final String ONGOING = "ongoing";
8      public static final String CLOSED = "closed";
9      public static final String NOT_STARTED = "not yet started";
10     public static final String DONE = "done";
11     public static final String SPRINT_CLOSE = "closed";
12     public static final String SPRINT_OPEN = "open";
13 }

```

4. Views

```

1  <!-- import.jsp -->
2  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3  <title>SPP</title>
4  <link href="<c:url value="/resources/css/bootstrap.css" />" rel="stylesheet" media="
  screen">
5  <link href="<c:url value="/resources/css/additional.css" />" rel="stylesheet" media="
  screen">
6
7
8  <script type="text/javascript" src="<c:url value="/resources/js/jquery.js" />"></script>
9  <script type="text/javascript" src="<c:url value="/resources/js/jquery-1.91.js" />"></
  script>
10 <script src="<c:url value="/resources/js/jquery-ui.js" />"></script>
11 <script type="text/javascript" src="<c:url value="/resources/js/bootstrap.js" />"></
  script>
12 <script type="text/javascript" src="<c:url value="/resources/js/highcharts.js" />"></
  script>
13
14
15 <!-- navbar_regular.jsp -->
16 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
17     pageEncoding="ISO-8859-1"%>
18 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
19 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
20 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
  html4/loose.dtd">
21 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
22 <!-- navbar -->
23     <div class = "navbar navbar-default" role = "navigation">
24         <div class = "navbar-header">
25             <center><label class = "navbar-brand"> Scrum Primer </
26                 label></center>
27             <button class="navbar-toggle" data-target="#navbar-
28                 collapse" data-toggle="collapse" type="button">
29                 <span class="sr-only">
30                     Toggle navigation
31                 </span>
32                 <span class="icon-bar"></span>
33                 <span class="icon-bar"></span>
34                 <span class="icon-bar"></span>
35             </button>
36         </div>
37     </div>

```

```

1 <!-- navbar_scrum.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5   <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
   html4/loose.dtd">
7   <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
8   <!-- navbar -->
9       <div class = "navbar navbar-default" role = "navigation">
10          <div class = "navbar-header">
11              <center><label class = "navbar-brand"> Scrum Primer </
   label></center>
12              <button class="navbar-toggle" data-target="#navbar-
   collapse" data-toggle="collapse" type="button">
13                  <span class="sr-only">
14                      Toggle navigation
15                  </span>
16                  <span class="icon-bar"></span>
17                  <span class="icon-bar"></span>
18                  <span class="icon-bar"></span>
19              </button>
20          </div>
21          <div class="navbar-collapse collapse" id = "navbar-collapse">
22              <ul class="nav navbar-nav">
23                  <li class = "${sprint }>
24                      <a href = "${contextPath
25                        }/viewAllSprint"
26                          data-toggle = "
27                            tooltip" data-
28                              placement = "bottom
29                                " title = "Sprint -
30                                  the incremental
31                                    durations" >Sprint
32                      </a>
33                  </li>
34                  <li class = "${productbacklog }">
35                      <a href = "${contextPath }/
36                        viewProductBacklog" data-toggle = "
37                          tooltip" data-placement = "bottom"
38                            title = "Product Backlog - contains
39                              the list of overall requirements
40                                written as user stories by the
41                                  Product Owner">Product Backlog</a>
42                  </li>
43                  <li class = "${notification }">
44                      <a href = "${contextPath }/
45                        viewNotification">Notification <
46                          span class = "badge">${notifs }</
47                          span></a>
48                  </li>
49              </ul>
50              <ul class = "nav navbar-nav navbar-right" style = "margin-right:
51                5px;">
52                  <li class="dropdown dropdown-inverse"><a class="
53                    dropdown-toggle" data-toggle="dropdown"
54                      href="#"><u><c:out value = "${sessionScope.
55                        projectName } (${sessionScope.role })/></
56                      u><b class="caret" data-toggle = "tooltip"
57                        data-placement = "bottom" title = "-Project
58                          Profile , -Back to Projects"></b></a>
59                      <ul class = "dropdown-menu">
60                          <li><a href = "#
61                            projectprofilemodal" data-
62                              toggle = "modal">Project
63                                Profile</a></li>
64                          <li class = "divider"></li>
65                          <li><a href = "${contextPath }/
66                            Project">All Projects</a></
67                          li>
68                      </ul>
69                  </li>
70                  <li class="dropdown dropdown-inverse"><a class="
71                    dropdown-toggle" data-
72                      toggle="dropdown" href="#"><span
73                        class = "glyphicon glyphicon-user
74                          "></span>&nbsp;<c:out value = "${
75                        sessionScope.username }"/><b class
76                          ="caret"></b></a>
77                      <ul class="dropdown-menu">
78                          <li><a href="#"
79                            profilemodal" data-
80                              toggle = "modal">
81                                Edit profile</a></
82                          li>
83                          <li><a href = "#
84                            passwordmodal" data-
85                              toggle = "modal">

```

```

56         Change Password</a>
57         <li class = "divider"></li>
58         <li><a href="{contextPath }/Logout">Logout</a></li>
59     </ul>
60
61 </div>
62 </div>
63
64 <!-- project profile modal -->
65 <div class="modal fade" id = "projectprofilemodal" tabindex = "-1" >
66     <div class="modal-dialog" style = "max-width: 70%;">
67         <div class="modal-content">
68             <div class="modal-header">
69                 <button type="button" class="close" data-dismiss="modal" aria-
70                     hidden="true">&times;</button>
71                 <h4 class="modal-title">Project Profile</h4>
72             </div>
73             <div class="modal-body">
74                 <table align = "center" cellpadding = "3px" width = "80%">
75                     <tr>
76                         <th>Project Name
77                         <td>${project.projectName}
78                     </tr>
79                     <tr>
80                         <th>Project Description
81                         <td>${project.projectDescription}
82                     </tr>
83                     <tr>
84                         <th>Status
85                         <td>${project.status }
86                     </tr>
87                 </table>
88                 <br>
89                 <br>
90                 <table align = "center" width = "80%" cellpadding = "3px">
91                     <tr>
92                         <th>Member
93                         <th> Role
94                     </tr>
95                     <c:forEach items = "${members }" var = "member">
96                         <tr>
97                             <td>${member.name }
98                             <td>${member.role }
99                         </tr>
100                     </c:forEach>
101                 </table>
102             </div>
103         </div>
104     </div>
105 </div>
106 </div>
107
108 <!-- edit profile modal -->
109 <div class="modal fade" id = "profilemodal" tabindex = "-1" >
110     <div class="modal-dialog" style = "max-width: 70%;">
111         <div class="modal-content">
112             <div class="modal-header">
113                 <button type="button" class="close" data-dismiss="modal" aria-
114                     hidden="true">&times;</button>
115                 <h4 class="modal-title">Edit Profile</h4>
116             </div>
117             <div class="modal-body">
118                 <form form method="post" action="editUser" modelAttribute = "
119                     edituser" onSubmit = "alertChange();" >
120                     <table align = "center" cellpadding = "3px" width =
121                         "80%">
122                         <tr>
123                             <td align = "center"><form:label path =
124                                 "firstName" class="col-sm-2 control
125                                     -label">First Name</form:label>
126                             <td align = "center"><form:input name =
127                                 "firstname" path="firstName" value
128                                 = "${sessionScope.sessionName }"
129                                 class = "form-control" id = "
130                                     firstname" required = "required" />
131                         </tr>
132                         <tr>
133                             <td align = "center"><form:label path =
134                                 "lastName" class="col-sm-2
135                                     control-label">Last Name</form:
136                                 label>
137                             <td align = "center"><form:input name =
138                                 "lastname" path="lastName" value =
139                                 "${sessionScope.sessionLast }"
140                                 class = "form-control" id = "
141                                     lastname" required = "required" />
142                         </tr>
143                         <tr>
144                             <td align = "center"><form:label path =
145                                 "username" class="col-sm-2
146                                     control-label">Username</form:
147                                 label>

```

```

130         <td align = "center"><form:input path="
username" value = "${sessionScope.
username }" class = "form-control"
id = "username" required = "
required"/>
131     </tr>
132 <tr>
133     <td align = "center"><form:label path =
"email" class="col-sm-2 control-
label">Email Address</form:label>
134
135     <td align = "center"><form:input type =
"email" path="email" value = "${
sessionScope.sessionEmail }" class
= "form-control" id = "email"
required = "required" />
136
137     </tr>
138 <tr>
139
140
141     <td align = "center"><button type="submit" class
="btn btn-success">Save Changes</button>
142
143     </tr>
144 </table>
145 </form:form>
146
147
148 </div> <!-- modal-body -->
149 </div><!-- /.modal-content -->
150 </div><!-- /.modal-dialog -->
151 </div><!-- /.modal -->
152
153 <!-- change password modal -->
154 <div class="modal fade" id = "passwordmodal" tabindex = "-1" >
155     <div class="modal-dialog" style = "max-width: 70%;">
156     <div class="modal-content">
157     <div class="modal-header">
158         <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
159         <h4 class="modal-title">Change Password</h4>
160     </div>
161     <div class="modal-body">
162         <form:form method="post" action="editPassword" modelAttribute =
"edituser2" onsubmit="return validate('${sessionScope.
password }');">
163             <table align = "center" cellpadding = "3px" width =
"80%">
164                 <tr>
165                     <td align = "center"><form:label path =
"password" class="control-label">
Current Password</form:label>
166                     <td align = "center"><form:input name =
"password" path="password" class =
"form-control" required = "required"
" type ="password" id = "pass" />
167                 </tr>
168                 <tr>
169                     <td align = "center"><form:label path =
"newPassword" class="control-
label">New Password</form:label>
170                     <td align = "center"><form:input name =
"newPassword" path="newPassword"
class = "form-control" id = "first"
required = "required" type = "
password" />
171                 </tr>
172                 <tr>
173                     <td align = "center"><form:label path =
"newPassword2" class="control-
label">Type new password again</
form:label>
174                     <td align = "center"><form:input path="
newPassword2" class = "form-control"
id = "second" required = "
required" type = "password"/>
175                 </tr>
176                 <tr>
177                     <td align = "center"><button type="submit" class
="btn btn-success">Save Changes</button>
178                 </tr>
179                 </tr>
180             </table>
181         </form:form>
182     </div>
183 </div> <!-- modal-body -->
184 </div><!-- /.modal-content -->
185 </div><!-- /.modal-dialog -->
186 </div><!-- /.modal -->
187
188 </script>
189
190 function alertChange(){
191
192
193
194
195

```

```

196         alert("Changes has been saved");
197     }
198 }
199 function validate(password){
200     var firstInput = document.getElementById("first").value;
201     var secondInput = document.getElementById("second").value;
202     var input = document.getElementById("pass").value;
203
204
205     if(firstInput != secondInput)
206     {
207         alert("New passwords did not match");
208         return false;
209     }
210 }
211 if(input != password){
212     alert("Invalid password");
213     return false;
214 }
215
216 alert("Password has been changed");
217 return true;
218 }
219 }
220 }
221 </script>

1 <!-- navbar_afterlogin.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3     pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
7     html4/loose.dtd">
8 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
9 <c:set var="currentURL" value="${pageContext.request.requestURI}"/>
10 <c:set var="lol" value="lol"/>
11 <!-- navbar -->
12     <div class="navbar navbar-default" role="navigation">
13         <div class="navbar-header">
14             <label class="navbar-brand"> Scrum Primer </label>
15             <button class="navbar-toggle" data-target="#navbar-
16                 collapse" data-toggle="collapse" type="button">
17                 <span class="sr-only">
18                     Toggle navigation
19                 </span>
20                 <span class="icon-bar"></span>
21                 <span class="icon-bar"></span>
22                 <span class="icon-bar"></span>
23             </button>
24
25
26
27         </div>
28         <div class="navbar-collapse collapse" id="navbar-collapse">
29             <ul class="nav navbar-nav">
30                 <li class="${activeproject}">
31                     <a href="${contextPath}/Project" data-toggle="
32                         tooltip" data-placement="bottom" title =
33                         "list of all projects"> Project </a>
34                 </li>
35                 <c:if test="${sessionScope.userType eq 'admin'}">
36                     <li class="${
37                         activeaccounts }"><a
38                         href="${
39                             contextPath }/
40                             Accounts">Accounts
41                         </a> </li>
42                 </c:if>
43             </ul>
44
45             <ul class="nav navbar-nav navbar-right" style="margin-right:
46                 5px;">
47                 <li class="dropdown dropdown-inverse"><a
48                     class="dropdown-toggle" data-
49                     toggle="dropdown" href="#"><span
50                     class="glyphicon glyphicon-user
51                     "></span>&nbsp;<c:out value="${
52                         sessionScope.username }"/><b class
53                         ="caret"></b></a>
54                     <ul class="dropdown-menu">
55                         <li><a href="#"
56                             profilemodal" data-
57                             toggle="modal">
58                             Edit profile </a></
59                             li>
60                         <li><a href="#"
61                             passwordmodal" data-
62                             toggle="modal">
63                             Change Password </a>
64                         <li class="divider"></
65                             li>
66                         <li><a href="${
67                             contextPath }/
68                             Logout">Logout </a>
69                         </li>

```

```

46                                     </ul>
47                                     </ul>
48                                 </div>
49                             </div>
50                         </div>
51
52
53                     <!-- edit profile modal -->
54                     <div class="modal fade" id = "profilemodal" tabindex = "-1" >
55                         <div class="modal-dialog" style = "max-width: 70%;" >
56                             <div class="modal-content">
57                                 <div class="modal-header">
58                                     <button type="button" class="close" data-dismiss="modal" aria-
59                                         hidden="true">&times;</button>
60                                     <h4 class="modal-title">Edit Profile</h4>
61                                 </div>
62                                 <div class="modal-body">
63                                     <form:form method="post" action="editUser" modelAttribute = "
64                                         edituser" onSubmit = "alertChange();" >
65                                         <table align = "center" cellpadding = "3px" width =
66                                             "80%" >
67                                             <tr>
68                                                 <td align = "center"><form:label path =
69                                                     "firstName" class="col-sm-2 control-
70                                                         label">First Name</form:label>
71                                                 <td align = "center"><form:input name =
72                                                     "firstName" path="firstName" value =
73                                                         "${sessionScope.sessionName }"
74                                                         class = "form-control" id = "
75                                                         firstName" required = "required" />
76                                             </tr>
77                                             <tr>
78                                                 <td align = "center"><form:label path =
79                                                     "lastName" class="col-sm-2
80                                                         control-label">Last Name</form:
81                                                         label>
82                                                 <td align = "center"><form:input name =
83                                                     "lastName" path="lastName" value =
84                                                         "${sessionScope.sessionLast }"
85                                                         class = "form-control" id = "
86                                                         lastName" required = "required" />
87                                             </tr>
88                                             <tr>
89                                                 <td align = "center"><form:label path =
90                                                     "username" class="col-sm-2
91                                                         control-label">Username</form:
92                                                         label>
93                                                 <td align = "center"><form:input path="
94                                                     username" value = "${sessionScope.
95                                                         username }" class = "form-control"
96                                                         id = "username" required = "
97                                                         required"/>
98                                             </tr>
99                                             <tr>
100                                                 <td align = "center"><form:label path =
101                                                     "email" class="col-sm-2 control-
102                                                         label">Email Address</form:label>
103                                                 <td align = "center"><form:input type="
104                                                     "email" path="email" value = "${
105                                                         sessionScope.sessionEmail }" class
106                                                         = "form-control" id = "email"
107                                                         required = "required" />
108                                             </tr>
109                                             <tr>
110                                                 <td align = "center"><button type="submit" class
111                                                     = "btn btn-success">Save Changes</button>

```

```

111         "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" />
112     </tr>
113     <tr>
114         <td align = "center"><form:label path =
            "newPassword" class="control-
            label">New Password</form:label>
115     <td align = "center"><form:input name =
            "newPassword" path="newPassword"
            class = "form-control" id = "first"
            required = "required" type = "
            password" />
116     </tr>
117     <tr>
118         <td align = "center"><form:label path =
            "newPassword2" class="control-
            label">Type new password again</
            form:label>
119     <td align = "center"><form:input path="
            newPassword2" class = "form-control"
            " id = "second" required = "
            required" type = "password"/>
120     </tr>
121     <tr>
122     <td align = "center"><button type="submit" class
            ="btn btn-success">Save Changes</button>
123     </td>
124     </tr>
125     </tr>
126     </table>
127     </form:form>
128
129
130
131
132
133
134     </div> <!-- modal-body -->
135     </div><!-- /.modal-content -->
136     </div><!-- /.modal-dialog -->
137     </div><!-- /.modal -->
138 <script>
139     function alertChange(){
140         alert("Changes has been saved");
141     }
142
143     function validate(password){
144     var firstInput = document.getElementById("first").value;
145     var secondInput = document.getElementById("second").value;
146     var input = document.getElementById("pass").value;
147
148
149     if(firstInput != secondInput)
150     {
151         alert("New passwords did not match");
152         return false;
153     }
154
155     if(input != password){
156         alert("Invalid password");
157         return false;
158     }
159
160     alert("Password has been changed");
161     return true;
162 }
163
164 }
165 </script>
166
1
2 <!-- accounts.jsp -->
3 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
4     pageEncoding="ISO-8859-1"%>
5 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
6 <%@ taglib prefix="fn" uri="http://www.springframework.org/tags/form" prefix="form"%>
7 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
9     html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <jsp:include page="import.jsp" />
14 </head>
15 <body>
16 <jsp:include page = "navbar_afterlogin.jsp"/>
17 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
18 <c:if test="${update eq 'yes'}">
19     <script type="text/javascript">
20         $(window).load(function(){
21             $('#UserEditorModal').modal('show');
22         });
23     </script>
24 </c:if>

```



```

24         <c:if test="${successfuldelete != null }">
25             <div class="alert alert-success alertMessage">
26                 <c:out value="${successfuldelete }"/>
27                 <button type="button" class="close" data-dismiss="alert">x</
                button>
28             </div>
29 </c:if>
30 <c:if test="${updateaccount != null }">
31     <div class="alert alert-success alertMessage">
32         <c:out value="${updateaccount }"/>
33         <button type="button" class="close" data-dismiss="alert">x</
            button>
34     </div>
35 </c:if>
36 <div class = "container-project" style = "background-color: transparent;">
37     <a href="#AddUserModal" data-toggle="modal"><button class = "btn btn-info">&nbsp;Add User</button></a>
38     <br>
39     <table class="table" cellpadding = "5px" width = "930px" >
40         <thead>
41             <tr bgcolor = "lightsteelblue" style = "
                display: block; width: 930px;">
42                 <th style = "width: 230px;">&nbsp;
                </th>
43                 <th style = "width: 230px;">Name
                </th>
44                 <th style = "width: 230px;">
                Username</th>
45                 <th style = "width: 230px;">
                Email</th>
46             </tr>
47         </thead>
48         <tbody style = "display: block; width: 930px;">
49             <c:forEach items = "${users }" var = "
                user">
50                 <tr>
51                     <td style = "width: 230px;"><a
                    href = "${contextPath }/
                    deleteAccount?aid=${user.
                    userID}" onclick = "return
                    confirm('Are you sure you
                    want to delete?');"><button
                    class = "btn btn-danger"><
                    span class = "glyphicon
                    glyphicon-trash"></span></
                    button></a>
52                     &nbsp;<a href = "${contextPath
                    }/editAccount?aid=${user.
                    userID}"><button class = "
                    btn btn-info">Edit</button
                    ></a>
53                     <td style="width: 230px;"><c:out
                    value = "${user.firstName
                    }"/> <c:out value = "${user
                    .lastName }"/>
54                     <td style = "width: 230px;"><c:
                    out value = "${user.
                    username }"/>
55                     <td style = "width: 230px;"><c:
                    out value = "${user.email
                    }"/>
56                 </tr>
57             </c:forEach>
58         </tbody>
59     </table>
60 </div>
61 <!-- edit profile modal -->
62 <div class="modal fade" id = "UserEditorModal" tabindex = "-1" >
63     <div class="modal-dialog" style = "max-width: 70%;">
64     <div class="modal-content">
65     <div class="modal-header">
66         <button type="button" class="close" data-dismiss="modal" aria-
        hidden="true">&times;</button>
67         <h4 class="modal-title">Edit User</h4>
68     </div>
69     <div class="modal-body">
70         <form:form method="post" action="updateAccount" modelAttribute =
        "edit">
71             <table align = "center" cellpadding = "3px" width =
        "80%">
72                 <tr>
73                     <td align = "center"><form:label path =
        "firstName" class="col-sm-2 control
        -label">First Name</form:label>
74                     <td align = "center"><form:input name =
        "firstName" path="firstName" value
        = "${updateuser.firstName }" class
        = "form-control" id = "firstName"
        required = "required" />
75                 </tr>
76                 <tr>
77                     <td align = "center"><form:label path =
        "lastName" class="col-sm-2
        control-label">Last Name</form:
        label>
78                     <td align = "center"><form:input name =
        "lastName" path="lastName" value =
79                 </tr>
80

```

```

    "${updateuser.lastName}" class = "
    form-control" id = "lastname"
    required = "required" />
81 </tr>
82 <tr>
83 <td align = "center"><form:label path =
84 <form:label path =
    "username" class="col-sm-2
    control-label">Username</form:
    label>
85 <td align = "center"><form:input path="
    username" value = "${updateuser.
    username}" class = "form-control"
    id = "username" required = "
    required"/>
86 </tr>
87 <tr>
88 <td align = "center"><form:label path =
    "email" class="col-sm-2 control-
    label">Email Address</form:label>
89 <td align = "center"><form:input type =
90 "email" path="email" value = "${
    updateuser.email}" class = "form-
    control" id = "email" required = "
    required" />
91 <form:input type="hidden" path = "
    userID" value = "${updateuser.
    userID}" />
92 </tr>
93 <tr>
94 <td align = "center"><button type="submit" class
95 ="btn btn-success">Save Changes</button>
96 </tr>
97 </table>
98 </form:form>
99 </div> <!-- modal-body -->
100 </div><!-- /.modal-content -->
101 </div><!-- /.modal-dialog -->
102 </div><!-- /.modal -->
103 <!-- edit profile modal -->
104 <div class="modal fade" id = "AddUserModal" tabindex = "-1" >
105 <div class="modal-dialog" style = "max-width: 70%;" >
106 <div class="modal-content">
107 <div class="modal-header">
108 <button type="button" class="close" data-dismiss="modal" aria-
109 hidden="true">&times;</button>
110 <h4 class="modal-title">Edit User</h4>
111 </div>
112 <div class="modal-body">
113 <form:form method="post" action="addAccount" modelAttribute = "
114 newUser">
115 <table align = "center" cellpadding = "3px" width =
116 "80%">
117 <tr>
118 <td align = "center"><form:label path =
119 "firstName" class="col-sm-2 control
120 -label">First Name</form:label>
121 <td align = "center"><form:input name =
    "firstname" path="firstName" class
    = "form-control" id = "firstname"
    required = "required" />
122 </tr>
123 <tr>
124 <td align = "center"><form:label path =
    "lastName" class="col-sm-2
    control-label">Last Name</form:
    label>
125 <td align = "center"><form:input name =
    "lastname" path="lastName" class =
    "form-control" id = "lastname"
    required = "required" />
126 </tr>
127 <tr>
128 <td align = "center"><form:label path =
129 "username" class="col-sm-2
    control-label">Username</form:
    label>
130 <td align = "center"><form:input path="
    username" class = "form-control"
    id = "username" required = "
    required"/>
131 </tr>
132 <tr>
133 <td align = "center"><form:label path =
    "password" class="col-sm-2 control-
    label">Password</form:label>
134 <td align = "center"><form:input type =
135 "password" path="password" class =
    "form-control" id = "password"
    required = "required" />

```

```

136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

52         </thead>
53
54         <tbody style = "display: block; overflow-y: auto; max-height: 200
55             px; width: 920px;">
56             <c:forEach items = "${activities}" var = "act">
57                 <tr>
58                     <td style = "width: 186px;"><c:out value = "${
59                         act.detail }" />
60                     <td style = "width: 186px;"><c:out value = "${
61                         act.time }" />
62                     <td style = "width: 186px;"><c:out value = "${
63                         act.date }" />
64                     <td style = "width: 186px;"><c:out value = "${
65                         act.taskStatus }" />
66                     <td style = "width: 186px;"><c:out value = "${
67                         act.doer }" />
68                 </tr>
69             </c:forEach>
70         </tbody>
71     </table>
72     <c:if test = "${sessionScope.role eq 'team_member' || sessionScope.role
73         eq 'scrum_master' && sprintstatus == 'open'}">
74     <center><a href = "#newactivitymodal" data-toggle = "modal"><button
75         class = "btn btn-primary">Add Activity</button></a></center>
76     </c:if>
77     </c:if>
78 </div>
79 <!-- newactivitymodal -->
80 <div class="modal fade" id = "newactivitymodal" tabindex = "-1" >
81     <div class="modal-dialog" style = "max-width: 70%;">
82     <div class="modal-header">
83         <button type="button" class="close" data-dismiss="modal" aria-
84             hidden="true">&times;</button>
85         <h4 class="modal-title"><c:out value = "New Activity for ${task.
86             taskName}"></c:out></h4>
87     </div>
88     <div class="modal-body">
89     <form:form action = "addActivity" modelAttribute = "newActivity" >
90     <table align = "center" cellpadding = "5px">
91         <tr>
92             <td><form:label path = "detail" class = "control-label">
93                 Detail</form:label>
94             <td><form:textarea class = "form-control" path
95                 = "detail" required = "required"></form:
96                 textarea>
97             <td><form:input type = "hidden" path = "taskID"
98                 value = "${task.taskID }"/>
99             <td><form:input type = "hidden" path = "accountID"
100                 value = "${sessionScope.accountID }"/>
101         </tr>
102         <tr>
103             <td><form:label path = "time" class = "control-label">
104                 Actual (no. of hours)</form:label>
105             <td><form:input type = "number" path = "time" class = "
106                 form-control" required = "required" min = "1"/>
107         </tr>
108         <tr>
109             <td><form:label path = "time" class = "control-label">
110                 Date</form:label>
111             <td><form:input type = "date" path = "date" class = "
112                 form-control" required = "required" min = "${
113                 datestarted }" max = "${datetoday }"/>
114         </tr>
115         <tr>
116             <td><form:label path = "taskStatus" class = "control-
117                 label">Task Status</form:label>
118             <td><form:select path = "taskStatus" class = "form-
119                 control" required = "required">
120                 <option value = "ongoing">Ongoing
121                 <option value = "done">Done
122             </form:select>
123         </tr>
124         <tr>
125             <td align = "center"><button type="submit" class
126                 ="btn btn-success">Add Activity</button>
127         </tr>
128     </table>
129     </form:form>
130 </div> <!-- modal-body -->
131 </div><!-- /.modal-content -->
132 </div><!-- /.modal-dialog -->
133 </div><!-- /.modal -->
134 </body>
135 </html>
136
137 1 <!-- add_product_backlog.jsp -->
138 2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
139 3     pageEncoding="ISO-8859-1"%>

```

```

4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
  html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <jsp:include page="import.jsp" />
13 </head>
14 <body>
15 <jsp:include page = "navbar_scrum.jsp"/>
16 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
17 <div class="alert alert-info alertMessage">
18     User story is a project requirement that can be a functional
19     requirement, non-functional requirement, or issue.
20     <button type="button" class="close" data-dismiss="alert">x</
      button>
21 </div>
22 <div class = "container-project">
23     <div class="table-scroll">
24         <form:form action = "insertUserStories" method = "POST" modelAttribute =
      "newStory">
25             <table class = "table" cellpadding = "5px" height = "330px" width = "930
      px">
26                 <thead>
27                     <tr bgcolor = "white" style = "display: block; width:
      930px;">
28                         <th style = "width: 310px;">User
      Story</th>
29                         <th style = "width: 310px;">
      Details</th>
30                         <th style = "width: 310px;">
      Status</th>
31                     </tr>
32                 </thead>
33                 <tbody style = "display: block; overflow-y: auto; max-height: 300
      px; width: 930px;">
34                     <c:forEach items = "${ newStory.stories}" varStatus = "
      index" >
35                         <tr>
36                             <td style = "width: 310px;"><input name = "
      stories[${index.index}].storyName" class =
      "form-control" required = "required" >
37                             <td style = "width: 310px;"><textarea name = "
      stories[${index.index}].detail" class = "
      form-control" required = "required" ></
      textarea>
38                             <td style = "width: 310px;"><select name = "
      stories[${index.index}].status" class = "
      form-control">
39                                 <option value = "not yet started
      ">not yet started</option>
40                                 <option value = "ongoing">ongoing</
      option>
41                                 <option value = "done">done</option>
42                             </select>
43                             <input type = "hidden" name = "stories[
      {index.index}].projectId" value =
      "${sessionScope.projectID}" />
44                         </tr>
45                     </c:forEach>
46                 </tbody>
47             </table>
48             <center><button class = "btn btn-primary" type = "submit">Add</button></
      center>
49         </form:form>
50     </div>
51 </div>
52 </body>
53 </html>

```

```

1 <!-- add_project.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
  html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <jsp:include page="import.jsp" />
13 </head>
14 <body>
15     <jsp:include page = "navbar_afterlogin.jsp" />
16 <div class = "container-project">
17     <center><strong><h3>New Project</h3></strong></center>
18     <div style = "margin-top: 30px;">
19     <form:form action = "inviteMembers" modelAttribute = "newProject">
20         <table align = "center" cellpadding = "3px" width = "600px">
21             <tr>

```

```

22         <td><form:label path = "projectName" class = "control-label">Project
23             Name</form:label>
24         <td><form:input path = "projectName" class = "form-control" required = "
25             required"></form:input>
26     </tr>
27     <tr>
28         <td><form:label path = "projectDescription" class = "control-label">
29             Project Description</form:label>
30         <td><form:textarea path = "projectDescription" class = "form-control"
31             required = "required"></form:textarea>
32     </tr>
33     <tr>
34         <td><form:label path = "numberOfTM" class = "control-label">Number of
35             Team Members</form:label>
36         <td><form:input type = "number" path = "numberOfTM" class = "form-
37             control" required = "required" ></form:input>
38             <form:input type = "hidden" path = "numberOfPO" value = "1" />
39     </tr>
40     <tr>
41         <td align = "center"><button type="submit" class="btn btn-success">
42             Submit</button>
43         <td align = "center"><button type="reset" class="btn btn-default">Clear</button>
44     </tr>
45 </table>
46 </form:form>
47 </div>
48 <br>
49 <br>
50 <div class="alert alert-warning alertMessage">
51     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>

```

```

1 <!-- add_sprint.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3     pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
9     html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <jsp:include page="import.jsp" />
14 </head>
15 <body>
16     <jsp:include page = "navbar_scrum.jsp" />
17     <div class = "container-project">
18     <center><strong><h3>User Stories for Sprint ${sprintNumber } </h3></strong></center>
19     <div style = "margin-top: 30px; margin-left: auto; margin-right: auto; width: 400px;">
20     <form:form action = "insertSprintStories" modelAttribute = "sprintForm">
21     <table class="table" cellpadding = "5px" height = "240px" width = "400px" align
22         = "center">
23     <thead style = "display:block; width: 400px; background-color: white">
24     <tr>
25     <td align = "center" width = "400px">Unfinished User Stories
26     </tr>
27     </thead>
28     <tbody style = "display:block; overflow-y:auto; height: 240px; width:400px;">
29     <c:forEach items = "${userStories }" var = "story" varStatus = "index">
30     <tr>
31     <td width = "400px"><input type = "checkbox" name = "sprintStories[${
32         index.index }].storyID" value = "${story.storyID }" ><c:out value =
33         "${story.storyName }"/>
34     <input type = "hidden" name = "sprintStories[${index.index }].sprintID"
35         value = "${sprintID }" />
36     </tr>
37     </c:forEach>
38     </tbody>
39     </table>
40     <table width = "100%">
41     <tr>
42     <td align = "center"><button type="submit" class="btn btn-success">Submit</
43         button>
44     <td align = "center"><button type="reset" class="btn btn-default">Clear</button>
45     </tr>
46     </table>
47     </form:form>
48 </div>
49 </div>
50 </body>
51 </html>

```

```

1 <!-- edit_product_backlog.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
   html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <jsp:include page="import.jsp" />
13 </head>
14 <body>
15 <jsp:include page = "navbar_scrum.jsp"/>
16 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
17 <div class = "container-project">
18   <div class="table-scroll">
19     <form:form action = "saveProductBacklog" method = "POST" modelAttribute
20       = "storyForm">
21       <table class = "table" cellpadding = "5px" height = "330px" width = "930
22         px">
23         <thead>
24           <tr bgcolor = "#ffffcc" style = "display: block; width:
25             930px;">
26             <th style = "width: 310px;">User
27             <th style = "width: 310px;">
28               Story</th>
29             <th style = "width: 310px;">
30               Details</th>
31             <th style = "width: 310px;">
32               Status</th>
33           </tr>
34         </thead>
35         <tbody style = "display: block; overflow-y:auto; max-height: 300
36           px; width:930px;">
37           <c:forEach items = "${ storyForm.stories}" var = "story"
38             varStatus = "index">
39             <tr>
40               <td style = "width: 310px;"><input name = "
41                 stories[${index.index}].storyName" class =
42                 "form-control" required = "required" value
43                 = "${story.storyName }">
44               <td style = "width: 310px;"><textarea name = "
45                 stories[${index.index}].detail" class = "
46                 form-control" required = "required" ><c:out
47                 value = "${story.detail }" /></textarea>
48               <td style = "width: 310px;"><select name = "
49                 stories[${index.index}].status" class = "
50                 form-control">
51                 <option value = "${story.status }"><c:
52                   out value="${story.status}"/></
53                   option>
54                 <optgroup label = "-----">
55                 <option value = "not yet started">not
56                   yet started</option>
57                 <option value = "ongoing">ongoing</
58                   option>
59                 <option value = "done">done</option>
60                 </optgroup>
61                 </select>
62               <input type = "hidden" name = " stories[${index.
63                 index }].storyID" value = "${story.storyID
64                 }" />
65             </tr>
66           </c:forEach>
67         </tbody>
68       </table>
69       <center><button class = "btn btn-primary" type = "submit">Save Changes</
70         button></center>
71     </form:form>
72   </div>
73 </div>
74 </body>
75 </html>

```

```

1 <!-- home_regular.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
   html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <jsp:include page="import.jsp" />
13 </head>
14 <body>
15   <jsp:include page = "navbar_regular.jsp" />
16   <c:if test="${successmessage != null }">
17     <div class="alert alert-success alertMessage">
18       <c:out value="${successmessage }"/>

```

```

19         <button type="button" class="close" data-dismiss="alert">x</
20         button>
21     </div>
22 </c:if>
23 <c:if test="{errorMessage != null}">
24     <div class="alert alert-warning alertMessage">
25         <c:out value="{errorMessage}" />
26         <button type="button" class="close" data-dismiss="alert">x</
27         button>
28     </div>
29 </c:if>
30 <c:if test="{errorMessage2 != null}">
31     <div class="alert alert-warning alertMessage">
32         <c:out value="{errorMessage2}" />
33         <button type="button" class="close" data-dismiss="alert">x</
34         button>
35     </div>
36 </c:if>
37 <c:if test="{errorlogin != null}">
38     <div class="alert alert-warning alertMessage">
39         <c:out value="{errorlogin}" />
40         <button type="button" class="close" data-dismiss="alert">x</
41         button>
42     </div>
43 </c:if>
44 <div class="span1">
45 <div class="tabbable">
46     <ul class="nav nav-tabs">
47         <li class="active"><a href="#features" data-toggle="
48         tab"><strong>Features</strong></a></li>
49         <li><a href="#about" data-toggle="tab"><strong>Scrum
50         </strong></a></li>
51     </ul>
52     <div class="tab-content">
53         <div id="features" class="tab-pane active">
54             <br>
55             <div id="carousel-example-generic" class="carousel slide
56             " data-ride="carousel">
57                 <!-- Indicators -->
58                 <ol class="carousel-indicators">
59                     <li data-target="#carousel-example-generic" data-slide-to="0"
60                     class="active"></li>
61                     <li data-target="#carousel-example-generic" data-slide-to="1"></
62                     li>
63                     <li data-target="#carousel-example-generic" data-slide-to="2"></
64                     li>
65                     <li data-target="#carousel-example-generic" data-slide-to="3"></
66                     li>
67                 </ol>
68                 <!-- Wrapper for slides -->
69                 <div class="carousel-inner">
70                     <div class="item active">
71                         <img src = "<c:url value = "/resources/scrum/scrumboard.JPG"/>">
72                         <div class="carousel-caption">
73                             Scrum Board
74                         </div>
75                     </div>
76                     <div class="item">
77                         <img src = "<c:url value = "/resources/scrum/burndownchart.JPG
78                         "/>">
79                         <div class="carousel-caption">
80                             Burndown Chart
81                         </div>
82                     </div>
83                     <div class="item">
84                         <img src = "<c:url value = "/resources/scrum/sprintbacklog.JPG
85                         "/>">
86                         <div class="carousel-caption">
87                             Sprint Backlog
88                         </div>
89                     </div>
90                     <div class="item">
91                         <img src = "<c:url value = "/resources/scrum/productbacklog.JPG
92                         "/>">
93                         <div class="carousel-caption">
94                             Product Backlog
95                         </div>
96                     </div>
97                 </div>
98                 <!-- Controls -->
99                 <a class="left carousel-control" href="#carousel-example-generic" data-slide="prev">
100                 <span class="glyphicon glyphicon-chevron-left"></span>
101                 </a>
102                 <a class="right carousel-control" href="#carousel-example-generic" data-slide="next">
103                 <span class="glyphicon glyphicon-chevron-right"></span>
104                 </a>
105             </div>
106         </div> <!-- features -->
107         <div id="about" class="tab-pane">

```



```

103 <p>Scrum mainly tackles the use of iterative development
      by implementing incremental durations called <i>
104 Sprints</i>.
      Sprints are timeboxed which means that they cannot be
      extended even if the work required is done or not.
105 At the beginning of
      each Sprint the Scrum Team is committed to complete
      these requirements, given and prioritized by the
106 Product Owner, each sprint.
      At the end of each Sprint, there will be a Sprint Review
      together with the Product Owner and some
107 stakeholders. The team will
      demonstrate what is built in the whole Sprint. The
      reviews in each Sprint Review will be incorporated
108 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>
109
110 <h4>FAQ</h4>
111 <ul>
112     <li><u>What is a Scrum Team?</u><br> <br>
113         <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>
114     <li><u>What is the Product Backlog</u><br><br>
115         <p>Product backlog contains the list of
            overall requirements written as
            user stories provided by the
            Product Owner.</p>
116     <li><u>What is a User Story?</u><br><br>
117         <p>User story is a project requirement
            that can be a functional
            requirement, non-functional
            requirement, or issue.
118     <li><u>What is the Sprint Backlog?</u><br><br>
119         <p>Sprint backlog contains the list of
            the tasks for the whole Sprint.</p>
120     <li><u>What is the Burndown Chart?</u><br><br>
121         <p>It is the chart which shows the trend
            of work to the remaining time in a
            Sprint.</p>
122     <li><u>What is a Scrum Board?</u><br><br>
123         <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".
124
125         </ul>
126     </div> <!-- about -->
127 </div> <!-- tabbable -->
128 </div>
129 <div class = "span2">
130 <div class = "container-login">
131     <h4>Sign in</h4>
132     <form:form method="post" action="Login" modelAttribute = "login">
133         <table align = "center" width = "380px" cellpadding = "3px">
134             <tr>
135                 <td align = "right"><form:label path="username" class="
136                     col-sm-2 control-label">Username</form:label>
137                 <td><form:input class="form-control" path= "username"
138                     required = "required" />
139             </tr>
140             <tr>
141                 <td align = "right"><form:label path = "password" class
142                     ="col-sm-2 control-label">Password</form:label>
143                 <td><form:input type="password" class="form-control"
144                     path="password" required = "required" />
145             </tr>
146         </table>
147         <table align = "center" width = "380px">
148             <tr>
149                 <td align = "center"><button type="submit" class="btn
150                     btn-success">Sign in</button>
151             </tr>
152         </table>
153     </div>
154 </div>
155 </form:form>
156
157 </div> <!-- container-login -->
158 <div class = "container-register">

```

```

159         New User? Sign up for new account.
160         <div style = "padding-top: 10px;">
161             <!-- registration form -->
162             <form:form method="post" action="addUser" modelAttribute = "user"
163                 id = "registration-form">
164                 <table align = "center" width = "380px" height = "180px"
165                     cellpadding = "2px">
166                     <tr>
167                         <td align = "center"><form:input name =
168                             "firstname" path="firstName"
169                             placeholder = "First Name" class =
170                             "form-control" id = "firstname"
171                             required = "required" pattern = "[a
172                                 -zA-Z0-9]+" />
173                         <td align = "center"><form:input name =
174                             "lastname" path="lastName"
175                             placeholder = "Last Name" class = "
176                             form-control" id = "lastname"
177                             required = "required" pattern = "[a
178                                 -zA-Z0-9]+" />
179                     </tr>
180                     <tr>
181                         <td align = "center"><form:input type =
182                             "email" path="email" placeholder =
183                             "Email Address" class = "form-
184                             control" id = "email" required = "
185                             required" />
186                     </tr>
187                     <tr>
188                         <td align = "center"><form:input path="
189                             username" placeholder="Username"
190                             class = "form-control" id = "
191                             username" required = "required"/>
192                         <td align = "center"><form:input path="
193                             password" type = "password"
194                             placeholder = "Password" class = "
195                             form-control" id = "password"
196                             required = "required"/>
197                     </tr>
198                     <tr>
199                         <td align = "center"><button type="submit" class
200                             ="btn btn-success">Submit</button>
201                         <td align = "center"><button type="reset" class
202                             ="btn btn-default">Clear</button>
203                     </tr>
204                 </table>
205             </form:form>
206         </div>
207     </div> <!-- container-register -->
208 </div>
209 </body>
210 </html>
211
212 1 <!-- new_member.jsp -->
213 2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
214 3     pageEncoding="ISO-8859-1"%>
215 4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
216 5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
217 6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
218 7
219 8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
220     html4/loose.dtd">
221 9 <html>
222 10 <head>
223 11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
224 12 <jsp:include page="import.jsp" />
225 13 </head>
226 14 <body>
227 15     <jsp:include page = "navbar_afterlogin.jsp" />
228 16     <div class = "container-members">
229 17         <form:form action = "addMembers" modelAttribute = "teamForm">
230 18             <h3>Invite team members</h3>
231 19             <table class = "table">
232 20                 <thead>
233 21                     <tr>
234 22                         <th>First Name
235 23                         <th>Last Name
236 24                         <th>Email Address
237 25                     </tr>
238 26                 </thead>
239 27                 <tbody>
240 28                     <c:forEach items = "${teamForm.users}" varStatus = "index"
241                         begin= "0" end = "${numberOfTM-1}">
242 29                     <tr>
243 30                         <td><input name = "users[${index.index}].firstName"
244                             class = "form-control" required = "required"
245                             pattern = "[a-zA-Z0-9]+">
246 31                         <td><input name = "users[${index.index}].lastName" class
247                             = "form-control" required = "required" pattern =
248                             "[a-zA-Z0-9]+">
249 32                         <td><input type = "email" name = "users[${index.index}].
250                             email" class = "form-control" required = "required

```

```

33         ">
34         <input type = "hidden" value = "team_member" name = "
35             users [ ${index.index} ].role" />
36     </tr>
37 </tbody>
38 </table>
39 <h3>Invite product owner</h3>
40 <table class = "table">
41 <thead>
42 <tr>
43     <th>First Name
44     <th>Last Name
45     <th>Email Address
46     <th>Outside Client?
47 </tr>
48 </thead>
49 <tbody>
50 <c:forEach items = "${teamForm.users}" varStatus = "index" begin
51     = "${numberOfTM}" end = "${numberOfTM + numberOfPO-1}">
52 <tr>
53     <td><input name = "users [ ${index.index} ].firstName"
54         class = "form-control" required = "required">
55     <td><input name = "users [ ${index.index} ].lastName" class
56         = "form-control" required = "required">
57     <td><input type = "email" name = "users [ ${index.index} ].
58         email" class = "form-control" required = "required
59         ">
60     <td><input type = "checkbox" value = "outsideclient">
61     <input type = "hidden" value = "product_owner" name = "
62         users [ ${index.index} ].role" />
63 </tr>
64 </c:forEach>
65 </tbody>
66 </table>
67 <br>
68 <td align = "center"><button type="submit" class="btn btn-success">
69     Submit</button>
70 <td align = "center"><button type="reset" class="btn btn-default">Clear</button>
71 </form:form>
72 </div>
73 </body>
74 </html>
75
76 1 <!-- notification.jsp -->
77 2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
78 3     pageEncoding="ISO-8859-1"%>
79 4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
80 5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
81 6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
82 7
83 8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
84     html4/loose.dtd">
85 9 <html>
86 10 <head>
87 11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
88 12 <jsp:include page="import.jsp" />
89 13 </head>
90 14 <body>
91 15 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
92 16 <jsp:include page = "navbar_scrum.jsp" />
93 17
94 18 <div class = "container-project">
95 19 <table class="table table-striped" cellpadding = "5px" height = "330px" width =
96     "930px" >
97 20 <thead style = "background-color: #ffffcc; width:930px;" >
98 21 <tr>
99 22 <th>Notifications
100 23 <th>&nbsp;
101 24 </tr>
102 25 </thead>
103 26 <tbody style = "display: block; overflow-y:auto; height: 300px; width:930
104     px;" >
105 27 <c:forEach items = "${notifications}" var = "notif">
106 28 <c:if test = "${notif.type == 'new.member'}">
107 29 <tr>
108 30 <td style = "width: 630px;"><c:out value = "${notif.newMember} was
109     added to the group"/>
110 31 <td style = "width: 300px;"><c:out value = "${notif.date}"/>
111 32 </tr>
112 33 </c:if>
113 34 <c:if test = "${notif.type eq 'scrum-meeting'}">
114 35 <tr>
115 36 <td style = "width: 630px;"><a href = "${contextPath }/viewScrumMeeting?
116     sid=${notif.sprintID}"><c:out value = "Please participate the daily
117     scrum meeting" /></a>
118 37 <td style = "width: 300px;"><c:out value = "${notif.date}"/>
119 38 </tr>
120 39 </c:if>
121 40 <c:if test = "${notif.type eq 'sprint-review'}">
122 41 <tr>
123 42 <td style = "width: 630px;"><c:out value = "Notice: Upcoming Sprint
124     Review!" />
125 43 <td style = "width: 300px;"><c:out value = "${notif.date}"/>
126 44 </tr>
127 45 </c:if>
128 46 </c:forEach>

```

```

47         </table>
48     </div>
49
50 </body>
51 </html>

1  <!-- poker_planning.jsp -->
2  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3     pageEncoding="ISO-8859-1"%>
4  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7  <%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>
8
9  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
10     html4/loose.dtd">
11 <html>
12 <head>
13 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
14 <jsp:include page="import.jsp" />
15 </head>
16 <body>
17 <jsp:include page = "navbar_scrum.jsp"/>
18 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
19 <div class="alert alert-info alertMessage">
20     Estimation of number of hours per task should be done using
21     Poker Planning - where values are fibonacci numbers.
22     <button type="button" class="close" data-dismiss="alert">x</
23     button>
24 </div>
25 <div class = "container-members">
26     <c:out value = "${story.storyName}"/>
27     <form:form method="post" name="classForm" id="classForm" modelAttribute="
28         taskForm" action = "${contextPath }/addTasks?sid=${sprintID}&rmng=${
29         remaining }&ts=${ts}">
30     <table class="table" cellpadding = "5px" width = "920px" >
31         <thead style = "background-color:#ffffcc;" width = "920px">
32             <tr>
33                 <td>Task Name
34                 <td>Volunteer
35                 <td>Estimate (in hrs)
36             </tr>
37         </thead>
38         <tbody>
39             <tr>
40                 <td><spring:bind path="taskForm.tasks[0].taskName">
41                     <form:input path="${status.expression}" class =
42                         "form-control"/>
43                     </spring:bind>
44                 <td><spring:bind path="taskForm.tasks[0].accountID">
45                     <select class = "form-control" name = "tasks[0].
46                         accountID" >
47                         <c:forEach items = "${members}" var = "member">
48                             <c:if test = "${member.role ne 'product.owner'
49                                 }">
50                                 <option value = "${member.accountID }"><c:out
51                                     value = "${member.name }"/></option>
52                             </c:if>
53                         </c:forEach>
54                     </select>
55                 <td><spring:bind path="taskForm.tasks[0].estimate">
56                     <select name = "tasks[0].estimate" class = "form
57                         -control">
58                         <option value = "1">1
59                         <option value = "2">2
60                         <option value = "3">3
61                         <option value = "5">5
62                         <option value = "8">8
63                         <option value = "13">13
64                     </select>
65                 <td><input type="button" id="addTaskButton" value="Add"
66                     />
67             </tr>
68         </tbody>
69     </table>
70     <button type="submit" value="Save Class" class = "btn btn-success" >Save</button
71     >
72 </form:form>
73 </div>
74 <script type="text/javascript">
75     $(document).ready(function() {
76         var storyPosition = 0;
77         $("#addTaskButton").click(function() {
78             storyPosition++;
79             $.get("<%=request.getContextPath()%>/appendStoryView", { fieldId:
80                 storyPosition },
81                 function(data){

```

```

81         $("#submitRow").before(data);
82     });
83 });
84 </script>
85 </body>
86 </html>
87
1 <!-- product_backlog.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
   html4/loose.dtd">
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <jsp:include page="import.jsp" />
13 </head>
14 <body>
15 <jsp:include page = "navbar_scrum.jsp"/>
16 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
17 <div class="alert alert-info alertMessage">
18     Product Backlog contains the list of overall requirements written as "
19     User Stories" by the Product Owner.
20     <button type="button" class="close" data-dismiss="alert">x</
21     button>
22 </div>
23 <div class = "container-project">
24     <div class="table-scroll">
25         <table class = "table" cellpadding = "5px" height = "300px" width = "930
26         px">
27             <thead>
28                 <tr bgcolor = "white" style = "display: block; width:
29                 930px;">
30                     <th style = "width: 310px;">User
31                     Story</th>
32                     <th style = "width: 310px;">
33                     Details</th>
34                     <th style = "width: 310px;">
35                     Status</th>
36                 </tr>
37             </thead>
38             <tbody style = "display: block; overflow-y:auto; max-height: 300
39             px; width:930px; background-color: #ffffcc">
40                 <c:forEach items = "${stories}" var = "story">
41                     <tr>
42                         <td style = "width: 310px;"><c:out value = "${
43                         story.storyName }" />
44                         <td style = "width: 310px;"><c:out value = "${
45                         story.detail }" />
46                         <td style = "width: 310px;"><c:out value = "${
47                         story.status }" />
48                     </tr>
49                 </c:forEach>
50             </tbody>
51         </table>
52     </div>
53 <c:if test = "${nostory != null}">
54     <center><strong><c:out value="${nostory }"/></
55     strong></center>
56 </c:if>
57 <c:if test = "${sessionScope.role eq 'product_owner' && sessionScope.
58     status == 'ongoing' }">
59     <table cellpadding = "10px" width = "930px">
60     <tr>
61     <td><form action = "addUserStories"><input type = "number" class = "
62     form_control" name = "numberOfStories" required = "required"
63     placeholder = "No. of new user stories" min = "1"> <button class = "
64     btn btn-primary" type = "submit" >Add User Story</button></form>
65
66     <td align = "right"><c:if test = "${stories != null}">
67     <center><a href = "${pageContext.request.contextPath}/
68     editProductBacklog"><button class = "btn btn-primary">Edit Product
69     Backlog</button></a></center>
70     </c:if>
71     </tr>
72     </table>
73 </c:if>
74 </div>
75 </body>
76 </html>
77
1 <!-- projects.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7

```

```

8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
9 html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <jsp:include page="import.jsp" />
14 </head>
15 <body>
16 <jsp:include page = "navbar_afterlogin.jsp"/>
17 <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
18 <c:if test="${update eq 'yes'}">
19     <script type="text/javascript">
20         $(window).load(function(){
21             $('#ProjectEditorModal').modal('show');
22         });
23     </script>
24 </c:if>
25 <c:if test="${addproject != null}">
26     <div class="alert alert-success alertMessage">
27         <c:out value="${addproject}"/>
28         <button type="button" class="close" data-dismiss="alert">x</
29         button>
30     </div>
31 </c:if>
32 <c:if test="${invite != null}">
33     <div class="alert alert-success alertMessage">
34         <c:out value="${invite}"/>
35         <button type="button" class="close" data-dismiss="alert">x</
36         button>
37     </div>
38 </c:if>
39 <div class = "container-project">
40     <div class="table-scroll">
41         <table class="table" cellpadding = "5px" height = "330px
42             " width = "930px" >
43             <thead>
44                 <tr bgcolor = "#94ff70" style = "display
45                     : block; width: 930px;">
46                     <th style = "width: 230px;">My
47                     Projects</th>
48                     <th style = "width: 340px;">
49                     Description</th>
50                     <th style = "width: 130px;">Role
51                     </th>
52                     <th style = "width: 130px;">
53                     Status</th>
54                     <th style = "width: 100px;">&
55                     nbsp;</th>
56                 </tr>
57             </thead>
58             <tbody style = "display: block; overflow-y: auto;
59                 max-height: 300px; width: 930px; background-
60                 color: #ffffcc">
61                 <c:forEach var = "proj" items = "${
62                     projects }">
63                     <c:forEach var = "projDetail" items = "$
64                         {projectDetails }">
65                         <c:if test = "${proj.projectID eq
66                             projDetail.projectID }">
67                             <tr>
68                                 <td style = "width: 230px;"><a
69                                     href = "${contextPath }/
70                                     viewProject?pid=${proj.
71                                     projectID}"><c:out value =
72                                     "${projDetail.projectName
73                                     }"/></a>
74                                 <td style = "width: 340px;"><c:
75                                     out value = "${projDetail.
76                                     projectDescription }" />
77                                 <td style = "width: 130px;"><c:
78                                     out value = "${proj.role }"
79                                     />
80                                 <td style = "width: 130px;"><c:
81                                     out value = "${projDetail.
82                                     status }" />
83                                 <c:if test = "${proj.role eq '
84                                     scrum-master }"><td style
85                                     = "width: 100px;"><a href =
86                                     "${contextPath }/
87                                     editProject?pid=${proj.
88                                     projectID}"><button class =
89                                     "btn btn-info">Edit</
90                                     button></a></c:if>
91                             </tr>
92                         </c:forEach>
93                     </c:forEach>
94                 </tbody>
95             </table>
96             <c:if test = "${noexisting != null}">
97                 <center><strong><c:out value="${noexisting
98                     }"/></strong></center>
99             </c:if>
100         </div>
101     </div>
102     <c:if test = "${sessionScope.username ne 'powner'}">
103         <center><a href = "${pageContext.request.contextPath }/
104             newProject"><button class = "btn btn-primary">New
105             Project</button></a></center>

```

```

71         </c:if>
72     </div>
73     <!-- edit profile modal -->
74     <div class="modal fade" id = "ProjectEditorModal" tabindex = "-1" >
75         <div class="modal-dialog" style = "max-width: 70%;" >
76             <div class="modal-content">
77                 <div class="modal-header">
78                     <button type="button" class="close" data-dismiss="modal" aria-
79                         hidden="true">&times;</button>
80                     <h4 class="modal-title">Edit Project</h4>
81                 </div>
82                 <div class="modal-body">
83                     <form:form method="post" action="updateProject" modelAttribute =
84                         "newProject">
85                         <table align = "center" cellpadding = "3px" width =
86                             "80%">
87                             <tr>
88                                 <td align = "center"><form:label path =
89                                     "projectName" class="col-sm-2
90                                     control-label">Project Name</form:
91                                     label>
92                                 <td align = "center"><form:input path="
93                                     projectName" value = "${
94                                     updateProject.projectName }" class
95                                     = "form-control" required = "
96                                     required" />
97                             </tr>
98                             <tr>
99                                 <td align = "center"><form:label path =
100                                    "projectDescription" class="col-sm
101                                    -2 control-label">Project
102                                    Description</form:label>
103                                 <td align = "center"><textarea name="
104                                    projectDescription" class = "form-
105                                    control" required = "required" ><c:
106                                    out value = "${updateProject.
107                                    projectDescription }"/></textarea>
108                                 <form:input type = "hidden" path = "
109                                    projectID" value = "${updateProject
110                                    .projectID }"/>
111                             </tr>
112                             <tr>
113                                 <td align = "center"><button type="submit" class
114                                     ="btn btn-success">Save Changes</button>
115                             </tr>
116                         </table>
117                     </form:form>
118                 </div> <!-- modal-body -->
119             </div><!-- /.modal-content -->
120         </div><!-- /.modal-dialog -->
121     </div><!-- /.modal -->
122 </body>
123 </html>
124
125 1 <!-- register.jsp -->
126 2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
127 3     pageEncoding="ISO-8859-1"%>
128 4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
129 5 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
130 6 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
131 7 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
132     html4/loose.dtd">
133 8 <html>
134 9 <head>
135 10 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
136 11 <jsp:include page="import.jsp" />
137 12 </head>
138 13 <body >
139 14 <jsp:include page = "navbar.regular.jsp" />
140 15
141 16 <div class = "container-register">
142 17     Sign up for new account.
143 18 <div style = "padding-top: 10px;">
144 19 <!-- registration form -->
145 20 <form:form method="post" action="addUser" modelAttribute = "user" id = "
146 21     registration-form">
147 22     <table align = "center" width = "380px" height = "180px" >
148 23         <tr>
149 24             <td align = "center"><form:input name = "firstname" path="
150                 firstName" placeholder = "First Name" class = "form-control
151                 " id = "firstname" required = "required" />
152 25             <td align = "center"><form:input name = "lastname" path="
153                 lastName" placeholder = "Last Name" class = "form-control"
154                 id = "lastname" required = "required" />
155 26         </tr>
156 27
157 28         <tr>
158 29             <td align = "center"><form:input type = "email" path="email"
159                 placeholder = "Email Address" class = "form-control" id = "
160                 email" required = "required" />
161 30
162 31         </tr>
163 32

```

```

33         <tr>
34             <td align = "center"><form:input path="username" placeholder="
                Username" class = "form-control" id = "username" required =
                "required"/>
35             <td align = "center"><form:input path="password" type = "
                password" placeholder = "Password" class = "form-control"
                id = "password" required = "required"/>
36         </tr>
37     </tr>
38     <tr>
39     <td align = "center"><button type="submit" class="btn btn-success">
                Submit</button>
40     <td align = "center"><button type="reset" class="btn btn-default">Clear
                </button>
41     </tr>
42 </tr>
43 </table>
44 </form:form>
45 </div>
46 </div>
47 </body>
48 </html>
49
50 <!-- scrummeeting.jsp -->
51 <%@ 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>
        </td>
        </tr>
        </c:forEach>
        </tbody>
        <c:if test = "${sprintstatus == 'open' }">
        <c:if test = "${sessionScope.role eq 'scrum.master' && currentmeeting eq
            'no' }">
        <tr>

```



```

58         <td style ="width: 500px;"><a href = "#newmeetingmodal"
          data-toggle = "modal"><button class = "btn btn-
          primary">Preside Daily Scrum Meeting</button></a>
59     </tr>
60 </c:if>
61 <c:if test = "${sessionScope.role eq 'team_member' && fillmeeting eq 'no
62     ' && currentmeeting eq null }">
63     <tr>
64         <td style ="width: 500px;"><a href = "#newmeetingmodal2"
          data-toggle = "modal"><button class = "btn btn-
          primary">Answer Daily Scrum Meeting</button></a>
65     </tr>
66 </c:if>
67 </table>
68 </div>
69
70 <!-- newcurrentmodal -->
71 <div class="modal fade" id = "newmeetingmodal" tabindex = "-1" >
72     <div class="modal-dialog" style = "max-width: 70%;">
73     <div class="modal-content">
74     <div class="modal-header">
75         <button type="button" class="close" data-dismiss="modal" aria-
          hidden="true">&times;</button>
76         <h4 class="modal-title">Daily Scrum Meeting</h4>
77     </div>
78     <div class="modal-body">
79     <form:form action = "addScrumMeeting" modelAttribute = "meeting" >
80     <table align = "center" cellpadding = "5px">
81         <tr>
82             <td><form:label path = "accomplishment" class = "control-
          -label">What have you accomplished in the last
          scrum meeting?</form:label>
83             <form:input type = "hidden" path = "sprintID"
          value = "${sprintID }"/>
84         </tr>
85         <tr>
86             <td><form:textarea class = "form-control" path
          = "accomplishment"></form:textarea>
87         </tr>
88         <tr>
89             <td><form:label path = "plan" class = "control-label">
90             What are you planning to do/finish for the next
          scrum meeting?</form:label>
91         </tr>
92         <tr>
93             <td><form:textarea class = "form-control" path
          = "plan"></form:textarea>
94         </tr>
95         <tr>
96             <td><form:label path = "impediment" class = "control-
          label">Is there any impediment?</form:label>
97         </tr>
98         <tr>
99             <td><form:textarea class = "form-control" path
          = "impediment"></form:textarea>
100        </tr>
101        <tr>
102            <td align = "center"><button type="submit" class
          ="btn btn-success">Submit</button>
103        </tr>
104    </table>
105    </form:form>
106    </div><!-- modal-body -->
107    </div><!-- /.modal-content -->
108    </div><!-- /.modal-dialog -->
109 </div><!-- /.modal -->
110
111 <!-- newcurrentmodal -->
112 <div class="modal fade" id = "newmeetingmodal2" tabindex = "-1" >
113     <div class="modal-dialog" style = "max-width: 70%;">
114     <div class="modal-content">
115     <div class="modal-header">
116         <button type="button" class="close" data-dismiss="modal" aria-
          hidden="true">&times;</button>
117         <h4 class="modal-title">Daily Scrum Meeting</h4>
118     </div>
119     <div class="modal-body">
120     <form:form action = "addScrumMeeting" modelAttribute = "meeting" >
121     <table align = "center" cellpadding = "5px">
122         <tr>
123             <td><form:label path = "accomplishment" class = "control-
          -label">What have you accomplished in the last
          scrum meeting?</form:label>
124             <form:input type = "hidden" path = "sprintID"
          value = "${sprintID }"/>
125         </tr>
126         <tr>
127             <td><form:textarea class = "form-control" path
          = "accomplishment"></form:textarea>
128         </tr>
129         <tr>
130             <td><form:label path = "plan" class = "control-label">
131             What are you planning to do/finish for the next
          scrum meeting?</form:label>
132             <form:input type = "hidden" path = "sprintID"
          value = "${sprintID }"/>
133         </tr>
134         <tr>
135             <td><form:textarea class = "form-control" path
          = "accomplishment"></form:textarea>

```

```

136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

```

</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>

```

```

46         </tbody>
47     </table>
48 </div>
49
50 </body>
51 </html>

1  <!-- sprint.jsp -->
2  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3     pageEncoding="ISO-8859-1"%>
4  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
9     html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <jsp:include page="import.jsp" />
14 </head>
15 <body>
16     <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
17     <jsp:include page="navbar-scrum.jsp" />
18     <div class="alert alert-info alertMessage">
19         Scrum Board is a visual task-tracking tool that contains the current tasks and is
20         grouped thru the table that labeled "Not yet started", "Ongoing" and "Completed"
21         <button type="button" class="close" data-dismiss="alert">x</
22         button>
23     </div>
24     <div class="span3">
25         <table width="100%" height="100%" cellpadding="20px">
26             <tr align="center">
27                 <td><h4>Sprint ${sprintNumber}</h4>
28             </tr>
29             <tr align="center">
30                 <td>Remaining days before Sprint Review: <b><c:out value="${
31                     remainingdays }"/></b>
32             </tr>
33             <tr align="center">
34                 <td><button class="btn btn-primary" href="${contextPath}/
35                     viewScrumBoard" disabled data-toggle="tooltip" data-placement="
36                     right" title="Scrum Board - a visual task-tracking tool that
37                     contain the current sprint tasks that is grouped thru the table
38                     that labeled 'Not yet Started', 'Ongoing' and 'Completed'">Scrum
39                     Board</button>
40             </tr>
41             <tr align="center">
42                 <td><a href="${contextPath}/viewScrumMeeting?sid=${sprintID}"><button
43                     class="btn btn-primary" data-toggle="tooltip" data-placement="
44                     right" title="Scrum Meeting - contains the accomplishments, the
45                     plans for the next meeting and the impediments">Scrum Meeting</
46                     button></a>
47             </tr>
48             <tr align="center">
49                 <td><a href="${contextPath}/viewSprintBacklog?sid=${sprintID}"><
50                     button class="btn btn-primary" data-toggle="tooltip" data-
51                     placement="right" title="Sprint Backlog - contains the list of
52                     the tasks for the whole sprint">Sprint Backlog</button></a>
53             </tr>
54         </table>
55     </div>
56     <div class="span4" style="background-color: inherit">
57         <table class="table" height="450px" width="770px">
58             <thead>
59                 <tr bgcolor="#94ff70" style="display
60                     :block; width: 770px;">
61                     <th width="256px">Not Yet
62                         Started</th>
63                     <th width="256px">Ongoing</th>
64                     <th width="256px">Completed</
65                     th>
66                 </tr>
67             </thead>
68             <tbody style="display: block; overflow-y: auto;
69                 height: 450px; width: 770px; background-
70                 color:#ccb299">
71                 <c:forEach items="${tasks}" var="task">
72                     <c:if test="${task.status eq 'not yet
73                         started' }">
74                         <tr>
75                             <td width="256px"><div style="
76                                 background-color: #ffb8ff; width:
77                                 210px; padding: 10px; box-shadow:
78                                 10px 10px 5px #888888;">User Story:
79                                 <c:out value="${task.storyName
80                                 }"/><br/>Task Name: <c:out value =
81                                 "${task.taskName }"/><br />
82                                 Estimate: <c:out value = "${task.
83                                 estimate }"/></div>
84                             <td width="256px">
85                             <td width="256px">
86                         </tr>
87                     </c:if>
88                 <c:if test="${task.status eq 'ongoing' }">
89                     <tr>
90

```



```

133     legend: {
134         layout: 'vertical',
135         align: 'right',
136         verticalAlign: 'middle',
137         borderWidth: 0
138     },
139     series: [{
140         name: 'Actual',
141         data: [{burnvalues}]
142     }, {
143         name: 'Ideal',
144         data: [ [0, ${totalEstimate}], [ ${totalDays}, 0 ] ]
145     } ]
146 });
147 });
148 </script>
149 </body>
150 </html>

1 <!-- sprintbacklog.jsp -->
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3     pageEncoding="ISO-8859-1"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
9     html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <jsp:include page="import.jsp" />
14 </head>
15 <body>
16     <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
17     <jsp:include page="navbar_scrum.jsp" />
18     <div class="alert alert-info alertMessage">
19         Sprint Backlog contains the list of tasks for the whole Sprint.
20         <button type="button" class="close" data-dismiss="alert">x</
21         button>
22 </div>
23 <div class="span3">
24     <table width="100%" height="100%" cellpadding="20px">
25     <tr align="center">
26         <td><h4><u>Sprint <c:out value="${sprintNumber}" /></u></h4>
27     </tr>
28     <tr align="center">
29         <td>Remaining days before Sprint Review: <b><c:out value="${
30             remainingdays}" /></b>
31     </tr>
32     <tr align="center">
33         <td><a href="${contextPath}/viewScrumBoard?sid=${sprintID}"><button
34             class="btn btn-primary" data-toggle="tooltip" data-placement="
35             right" title="Scrum Board - a visual task-tracking tool that
36             contain the current sprint tasks that is grouped thru the table
37             that labeled 'Not yet Started', 'Ongoing' and 'Completed'">Scrum
38             Board</button></a>
39     </tr>
40     <tr align="center">
41         <td><a href="${contextPath}/viewScrumMeeting?sid=${sprintID}"><button
42             class="btn btn-primary" data-toggle="tooltip" data-placement="
43             right" title="Scrum Meeting - contains the accomplishments, the
44             plans for the next meeting and the impediments">Scrum Meeting</
45             button></a>
46     </tr>
47     <tr align="center">
48         <td><button class="btn btn-primary" href="${contextPath}/
49             viewSprintBacklog" disabled data-toggle="tooltip" data-placement
50             = "right" title="Sprint Backlog - contains the list of the tasks
51             for the whole sprint">Sprint Backlog</button>
52     </tr>
53 </table>
54 </div>
55 <div class="span4">
56     <c:if test="${pokerplanning eq 'yes'}">
57         <div class="alert alert-success alertMessage">
58             <c:out value="No tasks for the sprint"/>
59             <button type="button" class="close" data-dismiss="alert">x</
60             button>
61         </div>
62         <c:if test="${sessionScope.role eq 'scrum_master'}">
63             <center><a href="${contextPath}/pokerPlanning?sid=${sprintID}
64                 &rmng=0"><button class="btn btn-primary">Add Task</
65                 button></a></center>
66         </c:if>
67     </c:if>
68     <c:if test="${pokerplanning eq null}">
69         <br>
70         <c:if test="${taskadded != null}">
71             <div class="alert alert-success alertMessage">
72                 <c:out value="${taskadded}"/>
73                 <button type="button" class="close" data-dismiss="alert">x</
74                 button>
75             </div>
76         </c:if>
77     </c:if>
78 <table class="table table-striped" cellpadding="5px" width="770px">
79     <thead align="center">

```

```

61         <tr bgcolor = "white" style = "display: block; width:
62             770px;" >
63             <th style = "width: 154px;" data-
64                 -toggle = "tooltip" data-
65                 placement = "top" title = "
66                 User Story - required
67                 features, improvement goals
68                 , and, possibly known
69                 defects provided by the
70                 Product Owner">User Story</
71                 th>
72             <th style = "width: 154px;" data-
73                 -toggle = "tooltip" data-
74                 placement = "top" title = "
75                 Task -- certain actions
76                 that are created to comple
77                 a certain user story">
78                 Sprint Task</th>
79             <th style = "width: 154px;" >
80                 Volunteer</th>
81             <th style = "width: 77px;" >
82                 Estimate</th>
83             <th style = "width: 77px;" >
84                 Actual </th>
85             <th style = "width: 154px;" >
86                 Status</th>
87         </tr>
88     </thead>
89     <tbody align = "center" style = "min-height:300px;" >
90         <c:forEach items = "${tasks}" var = "task">
91             <tr style = "display:block; width:770px; background-
92                 color: #ffffcc; ">
93                 <td style = "width: 154px;" ><c:out value = "${
94                     task.storyName }"></c:out></td>
95                 <td style = "width: 154px;" ><a href = "${
96                     contextPath }/viewActivity?tid=${task.
97                     taskID}"><c:out value = "${task.taskName
98                     }"></c:out></a></td>
99                 <td style = "width: 154px;" ><c:out value = "${
100                     task.volunteer }"></c:out></td>
101                 <td style = "width: 77px;" ><c:out value = "${
102                     task.estimate }"></c:out></td>
103                 <td style = "width: 77px;" ><c:out value = "${
104                     task.actual }"></c:out></td>
105                 <td style = "width: 154px;"><c:out value = "${
106                     task.status }"></c:out></td>
107             </tr>
108         </c:forEach>
109     </tbody>
110 </table>
111 <c:if test = "${sprintstatus == 'open' && sessionScope.role ne '
112     product_owner' }">
113     <center><a href = "#newtaskmodal" data-toggle = "modal"><button
114         class = "btn btn-primary">Add New Task</button></a></center>
115 </c:if>
116 <c:if test = "${(sessionScope.role eq 'product_owner') && (sprintreview
117     == 'yes') && (sprintstatus == 'open')}" >
118     <center><a href = "${contextPath }/sprintReview?sid=${sprintID
119         }"><button class = "btn btn-primary" data-toggle = "tooltip" data-
120         placement = "top" title = "Sprint Review - list of
121         the comments about the output of the current sprint">Sprint
122         Review</button></a></center>
123 </c:if>
124 <c:if test = "${ (sprintstatus == 'closed')}" >
125     <center><a href = "${contextPath }/viewSprintReview?sid=${sprintID}"><
126         button class = "btn btn-primary" data-toggle = "tooltip" data-
127         placement = "top" title = "Sprint Review - list of the comments
128         about the output of the current sprint">View Sprint Review</button
129         ></a></center>
130 </c:if>
131 </div>
132 <div class="modal fade" id = "newtaskmodal" tabindex = "-1" >
133     <div class="modal-dialog" style = "max-width: 70%;" >
134         <div class="modal-content">
135             <div class="modal-header">
136                 <button type="button" class="close" data-dismiss="modal" aria-
137                     hidden="true">&times;</button>
138                 <h4 class="modal-title">New Task</h4>
139             </div>
140             <div class="modal-body">
141                 <form:form action = "addTask" modelAttribute = "newTask" >
142                 <table align = "center" cellpadding = "5px">
143                     <tr>
144                         <td><form:label path = "storyID" class = "control-label">User
145                             Story</form:label>
146                         <td><form:select path = "storyID" class = "form-control"
147                             required = "required">
148                             <c:forEach items = "${sprintstories }" var = "story">
149                                 <option value = "${story.storyID }"><c:out value = "${
150                                     story.storyName }"/></option>
151                             </c:forEach>
152                     </tr>

```

```

116                                     </form:select >
117     </tr>
118     <tr>
119         <td><form:label path = "taskName" class = "control-label">Task
120             Name</form:label>
121             <td><form:input class = "form-control" path = "
122                 taskName" required = "required"></form:
123                 input>
124     </tr>
125     <tr>
126         <td><form:label path = "accountID" class = "control-label">
127             Volunteer</form:label>
128         <td><form:select path = "accountID" class = "form-control"
129             required = "required">
130             <c:forEach items = "${members}" var = "member">
131                 <c:if test = "${member.role ne '
132                     product_owner' }">
133                     <option value = "${member.accountID }"><
134                         c:out value = "${member.name }"/>
135                 </c:if>
136             </c:forEach>
137         </form:select >
138     </tr>
139     <tr>
140         <td><form:label path = "estimate" class = "control-label">
141             Estimate</form:label>
142         <td><form:select path = "estimate" class = "form-control"
143             required = "required">
144             <option value = "1">1
145             <option value = "2">2
146             <option value = "3">3
147             <option value = "5">5
148             <option value = "8">8
149             <option value = "13">13
150         </form:select >
151         <form:input type = "hidden" value = "${sprintID }" path
152             = "sprintID"/>
153     </tr>
154     <tr>
155         <td align = "center"><button type="submit" class
156             ="btn btn-success">Add Activity</button>
157     </tr>
158 </table>
159 </form:form>
160 </div>
161 </div>
162 </div>
163 </div>
164 </div>
165 </body>
166 </html>
167
168 1 <!-- sprint_review_view.jsp -->
169 2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
170 3     pageEncoding="ISO-8859-1"%>
171 4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
172 5 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
173 6 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
174 7
175 8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
176     html4/loose.dtd">
177 9 <html>
178 10 <head>
179 11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
180 12 <jsp:include page="import.jsp" />
181 13 </head>
182 14 <body>
183 15     <c:set var="contextPath" value="${pageContext.request.contextPath}"/>
184 16     <jsp:include page = "navbar.scrum.jsp" />
185 17
186 18     <div class = "container-project">
187 19         <ul class="breadcrumb">
188 20             <li>
189 21                 <a href="${contextPath }/viewSprintBacklog?sid=${sprintID}">Sprint Backlog</a> <
190                 span class="divider"></span>
191             </li>
192 22             <li class="active">Sprint Review</li>
193 23         </ul>
194 24
195 25         <br>
196 26         <table class = "table" cellpadding = "5px" height = "300px" width = "920
197             px">
198 27             <thead>
199 28                 <tr bgcolor = "white" style = "display: block; width:
200                 920px;" >
201 29                     <th style = "width: 200px;">User
202                     Story for this Sprint</th>
203 30                     <th style = "width: 500px;">
204                     Review</th>
205 31                     <th style = "width: 220px;">User
206                     Story Status</th>
207 32
208 33
209 34
210 35
211 36             </thead>

```

```
37 <tbody style = "display: block; overflow-y: auto; max-height: 280px; width
38 :920px; background-color: #ffffcc">
39 <c:forEach items = "${reviews }" varStatus = "index" var
40 = "review" >
41 <tr>
42 <td style = "width: 200px;"><c:out value = "${
43 review.storyName }"/>
44 <td style = "width: 500px;"><c:out value = "${
45 review.review }"/>
46 <td style = "width: 220px;"><c:out value = "${
47 review.status }"/>
48 </tr>
49 </c:forEach>
50 </tbody>
51 </table>
52 </div>
53 </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 sakín 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!