**Senior Project Website**

**Version 5**

**Final Document**

CIS 4911 Senior Project

Section U01

Professor Masoud Sadjadi

**Prepared by**

Jacek Kopczynski

Yamel Peraza

July 28th, 2014

# 

Copyright

*All title, including but not limited to copyrights, in and to the SENIOR PROJECT WEBSITE are owned by School of Computing and Information Sciences Faculty of Florida International University. All title and intellectual property rights in and to the content which may be accessed through use of the SENIOR PROJECT WEBSITE is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content. All rights not expressly granted are reserved by School of Computing and Information Sciences Faculty of Florida International University.*



Executive summary

*The mission of this semester’s project is to tune the existing functionalities of the system and implement the request of virtual machine to the head professor through the use of the Senior Project website application.*

*The purpose of this document is to finalize the development of the Senior Project Website for version 5. In essence this document serves to combine all aspects from every prior document. For a less intense look at SPWS, this document serves as a good start.*

Table of Contents

*Senior Project Website V5*



[Copyright 2](#_Toc394049271)

[Executive summary 3](#_Toc394049272)

[Table of Contents 4](#_Toc394049273)

[1. Introduction 6](#_Toc394049274)

[1.1 Problem Definition 6](#_Toc394049275)

[1.2 Scope of System 7](#_Toc394049276)

[1.3 Overall Development Methodology 7](#_Toc394049277)

[1.4 Terminology – Definitions, acronyms, and abbreviations 9](#_Toc394049278)

[1.5 Overview of Document 10](#_Toc394049288)

[2 Feasibility Study 11](#_Toc394049289)

[2.1 Description of Current System 11](#_Toc394049290)

[11](#_Toc394049291)

[2.2 Alternative Solutions 11](#_Toc394049292)

[2.2.1 Description of Alternatives 12](#_Toc394049293)

[2.2.2 Selection Criteria 13](#_Toc394049294)

[2.2.3 Analysis of Alternatives 13](#_Toc394049295)

[2.3 Recommendations 14](#_Toc394049296)

[3 Project Plan 16](#_Toc394049297)

[3.1 Project Organization 16](#_Toc394049298)

[3.1.1 Project Personnel Organization 16](#_Toc394049299)

[3.1.2 Hardware and Software Resources 18](#_Toc394049300)

[3.2 Identification of Tasks, Milestones and Deliverables 19](#_Toc394049301)

[3.3 Cost Estimate Here are the cost estimate for the development in SPWv.4. 20](#_Toc394049302)

[4 Proposed System Requirements 21](#_Toc394049303)

[4.1 Functional Requirements 21](#_Toc394049304)

[4.2 Analysis of System Requirements 26](#_Toc394049305)

[4.2.1 Scenarios 26](#_Toc394049306)

[4.2.2 Use Case Model 28](#_Toc394049307)

[4.2.3 Static Model 28](#_Toc394049309)

[4.2.4 Dynamic Model 29](#_Toc394049311)

[5 System Design 31](#_Toc394049319)

[**5.1** **Overview** 31](#_Toc394049320)

[**5.2** **Subsystem Decomposition** 31](#_Toc394049321)

[**5.3** **Hardware and Software Mapping** 35](#_Toc394049322)

[**5.4** **Persistent Data Management** 35](#_Toc394049323)

[**5.5** **Security/Privacy** 41](#_Toc394049324)

[6 Detailed Design 43](#_Toc394049325)

[6.1. Overview 43](#_Toc394049326)

[**6.2. Static Model** . 43](#_Toc394049327)

[6.3. Dynamic Model 44](#_Toc394049331)

[**6.4. Code Specification** 46](#_Toc394049334)

[7. System Validation 48](#_Toc394049336)

[**7.1**  **Subsystem Tests** Associated stubs included in Appendix G 48](#_Toc394049337)

[**7.2** **System Tests** Note corresponds to used case id plus a ‘T’#testID 49](#_Toc394049338)

[**7.3** **Evaluation of Tests** 53](#_Toc394049339)

[8. Glossary 54](#_Toc394049340)

[9. Appendix 56](#_Toc394049350)

[**9.1** **Appendix A - Gant Chart** 56](#_Toc394049351)

[9.2 Appendix B - Use Cases 57](#_Toc394049352)

[9.3 Appendix C – User Interface Designs 97](#_Toc394049353)

[9.4 Appendix D – Static Models 106](#_Toc394049354)

[9.5 Appendix E – Dynamic Models 118](#_Toc394049366)

[9.6 Appendix G – Documented Code for Test Driven and Stubs 156](#_Toc394049422)

[10. References 158](#_Toc394049423)

1. Introduction

*Senior Project Website V5*



This section serves to introduce the primary focus and details of this version of SPWS.

* 1. Problem Definition

For Senior Project class, students are required to use virtual machines to develop and deploy their applications. The process of requesting virtual environments currently is done manually, given that Senior Project Website does not have the functionality of processing or managing virtual machine requests for its users. The impractical approach of manually finding each student’s email requesting virtual machines is time-consuming, lacks efficiency and is error prone.

The head professor would like to extend the usability of the system by allowing students to request and customize their virtual machine settings, while letting him manage the students’ requests.

We also plan to extend the functionality of the site by adding smaller features like: forgot password, adding students with incomplete grades, changing passwords through act as user, and fixing problems with functionality of the old site.

* 1. Scope of System

The objective of SPWv.5 is to make the Senior Project website more useful and responsive by tuning existing functionalities in the system and implementing a new feature that permits students to request their virtual machine needs to the head professor for completion of their project. Also, this feature will allow the head professor to have a much more organized web-tool to manage Senior Project class.

The scope of matchmaking will be very greatly expanded for the head professor. He will be doing two-phased matchmaking one more intensively and another more relaxed. The more intensive one they will have control over forcing the best team possible or compromising with the best team possible given students to want the particular project. The head professor will have some customizability to the more relaxed other projects. Overall he will know what his actions will do before committing to the match with varying degrees of ability to manipulate such a match.

* 1. Overall Development Methodology

The overall development methodology was agile. This is quite a departure from the waterfall method, with a greater focus on weekly incremental changes rather than much analysis of requirements followed by implementation at the end. Documentation of course is still constructed due to the nature of accreditation and for aide of future version developers.

The agile methodology allows of the ability to develop week by week taking client feedback to tweak or refine concepts in development, allowing more flexibility during the development process rather than the inflexible “setting in stone” of the waterfall methods of certain features. So with these benefits in mind agile methodology was the choice of software development this semester resulting in more development.

* 1. Terminology – Definitions, acronyms, and abbreviations

EULA: End-User License Agreement

FIU: Florida International University

Google Docs: Free web-based office suite offered by Google within Google Drive service.

PHP: Open source server-side scripting language designed for web development to produce dynamic web pages.

SPW: Senior Project Website

SPWv.1: Senior Project Website Version 1

SPWv.2: Senior Project Website Version 2

SPWv.3: Senior Project Website Version 3

SPWv.4: Senior Project Website Version 4

SPWv.5: Senior Project Website Version 5

**NRMP**: A form of matchmaking devised by the national residency matchmaking program (NRMP). The details of it are that hospitals have residency spots to fill and a ranked list of applicants they want filling them. Applicant ranked residency programs themselves. The matchmaking process works by having applicants apply to residencies in their list’s order. If another applicant tries to displace a current tentative applicant they displace the tentative applicant if the challenging applicant is more desired by the program else they try the next program, displaced applicants try their next program too. Matching ends when all applicants are match, or all applicants are match minus ones who went through their entire list.

**Heuristics**: Colloquially means “common sense approach”, in computer science the meaning is adapted to mean a technique applied to solve a problem. I.e. to shorten average job time in a computer do short jobs first.

**VIP**: Very important project, a project ranked by the head professor to be of a score between 2 and 100 this project will undergo intense matchmaking using heuristics to find a perfect team quickly (higher scored means higher priority)

**OP**: Other project, a project ranked by the head professor to have a score of 1, such projects will undergo NRMP matchmaking to give the ability to compromised project proposer and student constraints. These are more hands-off.

**LinkedIn:** Social media website for business professions to network and advertise themselves

* 1. Overview of Document

**Section 1** Introduces the final document with version specific details and ideals.

**Section 2** Discusses the feasibility of the current system and what the proposed system plans to do about it, also analyzes alternatives and passes judgment on the alternatives to the current.

**Section 3** Discusses project specific details with regards to cost and organization

**Section 4** Discusses the systems requirements, function and nonfunctional and performs analysis.

**Section 5** Discusses the design of the overall system, as well as decomposition into subsystems.

**Section 6** Detailed design discusses the matchmaking subsystem in specific as that’s the primary focus of this version

**Section 7** Validation tests to the subsystem and system as well as their evaluation occurs here

**Section 8** Glossary of various terms can be found here

**Section 9** Various diagrams, designs, and utilities are found here at the appendix.

**Section 10** References relating to this document can be found here

1. Feasibility Study

*Senior Project Website V5*



The feasibility study analyzes alternative solutions to the various problems. For each alternative solution, benefits and limitations, will be examined. After a detailed analysis, we will be able to determine the best possible solution. We will also describe the current system and the purpose of the proposed solution. Finally, the study will conclude with justifiable recommendations.

* 1. Description of Current System

Current version of Senior Project Website does not offer the functionality of requesting virtual machines to the head professor. A brief description of how this task is currently performed is discussed in this section.

Students enrolled in Senior Project class are automatically signed up into Senior Project Website and algorithmically assigned into a team-project. After the projects have been finally assigned to the student, the head professor sends an email to all active students in the class requesting their needs of virtual machines environments for the completion of their project. Then, each student that needs virtual environments replies to the head professor’s email with all their needs regarding virtual machine. Finally, the head professor tracks each individual’s email, gathers all the information, and sends it to the school system admin, who will create the virtual machines for the students.

There is a lot of manual work done to collect and send the request of virtual machines for each individual team. In general, previous semesters of Senior Project class have not developed a functionality into Senior Project Website that addresses this issue.

* 1. Alternative Solutions

This section will describe alternatives solutions to the problem. A selection criterion will be analyzed based on the feasibility of such alternatives. Finally all the alternatives will be compared concluding on the best option to pursue.

* + 1. Description of Alternatives

Alternative I: Current System

Keep the current practice of requesting the virtual machine environments manually. Although it is not an efficient way of requesting the student’s needs, it is proven to work. The main downside of this practice is that there is not a stable way of tracking student and virtual machine association. For example, emails requests can be lost or forgotten, implying the student has to submit a new request via email, etc.

Alternative II: Proposed System

Implementation of the proposed functionality for Senior Project Version 5. This proposed solution will efficiently allow users of Senior Project Website to request the creation of virtual machine for the completion of their project and will allow the head professor to have a much more organized web-tool to manage Senior Project class.

* + 1. Selection Criteria

The previous alternatives will be analyzed based on their operational, technical, schedule, and economic feasibility. Following criteria utilized from SPWv.4 Feasibility study.

Operational Feasibility

The operational feasibility deals with how functional an alternative is in dealing with the system requirements and how well it is received. An ideal solution would be user-friendly and well received. Is it easy to use? Does it solve the problem? Does it provide the users with accurate and desired information? Does it generate positive feedback by its users?

Technical Feasibility

The technical feasibility focuses on understanding the available technical resources and how they can be used to implement the system. Is the builder team familiar with the technologies? How well do they need to master the technologies? Are the technologies supported at the moment or/and in the future?

Schedule Feasibility

The schedule feasibility deals with the amount of time needed to implement an alternative. An ideal alternative will be quick to produce. How long will this solution take to design and implement?

Economic Feasibility

The economic feasibility analyzes the cost and benefits of an alternative. An ideal alternative will be inexpensive to produce. How much will it cost to implement? Will there be any gains? Will it have additional costs in the long run?

* + 1. Analysis of Alternatives

Alternative I, the current system is rather inefficient, making it hard to track all the students’ emails that are requesting virtual machines. Also it is time consuming when the head professor has to group all the virtual machine request made by students and send them via email to the school’s system admin.

Alternative II, the proposed system will potentially increase the functionality of the system and its efficiency without incurring in any economic expense. This proposed updated system will increase functionality and therefore be convenient for both students and professor. The cost effectiveness of this solution allows it to be done in a period of a semester to ensure timely accessibility by students.

* 1. Recommendations

The SPWv.5 recommends Alternative II as the proposed system because this system will guarantee that head professor and students can manage their virtual machine requests in an efficient way. Additionally, Alternative II does not incur in any economic expense and is feasible for students to complete in the span of a semester. Alternatively, Alternative I (even though it works) is mainly manually done and in some instances it can introduce problems regarding the management of the virtual requests. As a result, Alternative II is a more efficient choice.

1. Project Plan

*Senior Project Website V5*



This chapter will cover the project plan for the SPWv.5, personnel organization, hardware and software requirements, milestones of the project and the costs involved for its development.

* 1. Project Organization

This section details how the SPWv.5 will be organized throughout the life of the project. It also contains the technical needs of the project at this time; this includes the hardware and software needs of the developers.

* + 1. Project Personnel Organization

Even though two members of the team will work under the same project name, this project will be a one person-project because each individual will have different responsibilities and tasks at hand. The duration of the developmental part of the project is spread over 14 weeks and the agile management methodology tool used is Mingle.

A two weeks sprint length is planned for a total of five sprints. Every sprint except the last one will focus in developing specific user stories. Daily meetings will be conducted with the team’s members to talk about what was done the previous day, what needs to be done in the present day and any challenges we are facing to complete the tasks for the current sprint. Also, meetings with the product owner at the beginning and end of each sprint will be scheduled to discuss the progress of the project. After sprint number 5 the focus will be placed on the deliverables seen under tentative schedule in CIS moodle.

|  |  |
| --- | --- |
| Name | Role |
| Jacek Kopczynski  Yamel Peraza | Responsible for ensuring the project progresses based on specifications from client and advice from mentor. Leads discussion ideas and adapts to client needs. |

**Table 3.1** – Roles

|  |  |
| --- | --- |
|  | Jacek Kopczynski Yamel Peraza |
| Project Plan | ✓ |
| Deliverables | ✓ |
| User Interface/Views | ✓ |
| Database | ✓ |
| Controllers/Models | ✓ |
| User Authentication | ✓ |
| Testing | ✓ |
| Final Presentation | ✓ |
| Demo | ✓ |

**Table 3.2** – Project preparation and development roles (Note: “✓” denotes participation in role)

* + 1. Hardware and Software Resources

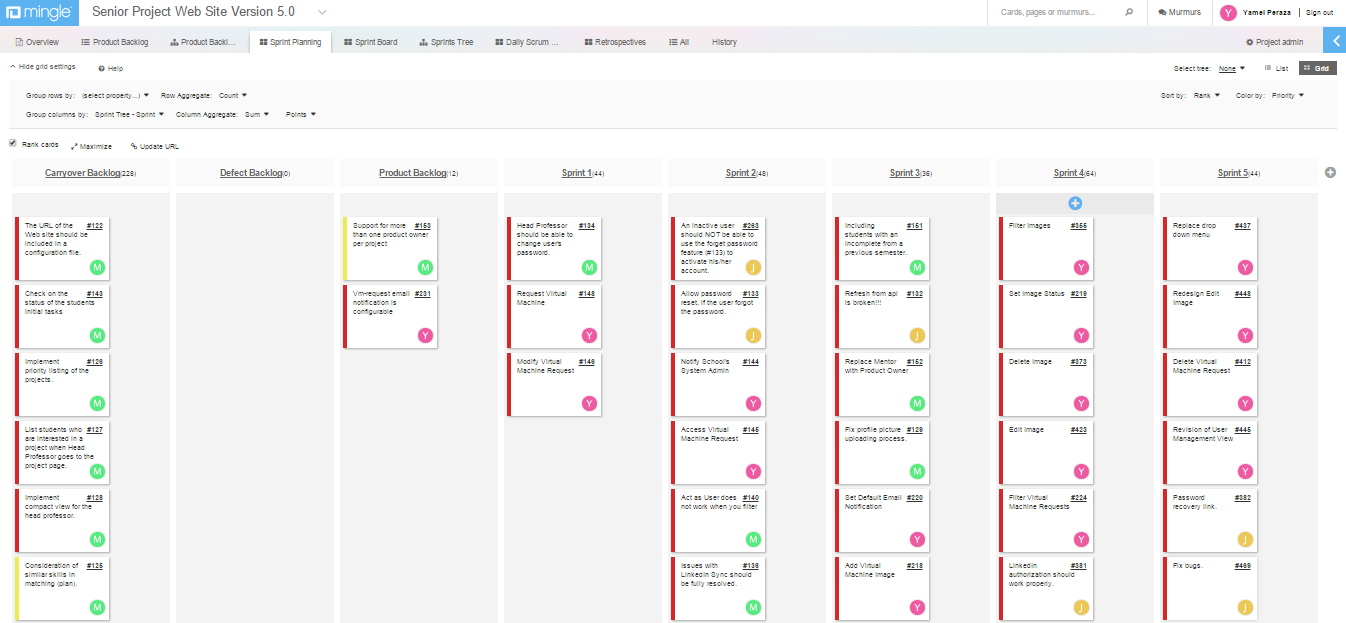
This section lists the software and hardware needed to develop and deploy the proposed project. Hardware and software will not change from version 1 to version 5.

|  |  |
| --- | --- |
| Software | Hardware |
| Sublime Text 2 | Processor: x86 / x64  2.0 GHz or faster |
| PHP 5.1 | Memory:  2 GB DDR3 or higher |
| MySQL 5.0 | Disk Space:  20 GB |
| Microsoft  Office Package 2010 | Display:  1024 X 768 with DirectX 9 |
| Mac OS X |  |
| Microsoft Windows 7  Home/Pro/Ultimate |  |
| Apache 2 |  |
| Firefox Mozilla  ver. 16.0 or higher |  |
| Internet Explorer  ver. 7.0 or higher |  |
| Safari  ver. 5.0 or higher |  |
| Google Chrome  ver. 19.0 or higher |  |
| Skype / Team Viewer |  |
| GitHub repository |  |

Note: This table is from the SPWv.1 “Feasibility Study & Project Plan” (Fernandez, Sanchez, Moya 1).

* 1. Identification of Tasks, Milestones and Deliverables

The tasks and milestones for this semester have essentially occurred on a weekly basis. The end result being the virtual machine system outlined in this proposal. All deliverables are to be turned in at the end of the semester. Also, deliverable-relevant content is to be created during the different sprint periods.

Project management meetings and tasks are generally defined by Mingle, a tool to aid in this agile methodology approach new this semester. Refer to picture below.

.

* 1. Cost Estimate  
     Here are the cost estimate for the development in SPWv.5.

|  |  |  |
| --- | --- | --- |
| Project Component | Description – Cost would not be an issue because students are improving on an existing solution to the Senior Project course student, professor, and mentor interaction | % of Total Project Cost |
| Project Management | 0 (Students are working all aspects of the project) | 0 |
| Hardware | 0 (already in place) | 0 |
| Software | 0 (open source) | 0 |
| Testing | 0 (testing tools open source/temporary free license products) | 0 |
| Training | 0 (Students are learning on their own) | 0 |
| Risk Management | 0 (No risk management budget is allocated) | 0 |
| Total | 0 | 0 |

1. Proposed System Requirements

*Senior Project Website V5*



System requirements are defined in this section. The first subsection features the functional requirements for SPWv.5, followed by, SPWv.4, SPWv.1, SPWv.2, and SPWv.3.

* 1. Functional Requirements

**The following use cases pertain to version SPWv.5**

* The system shall allow students to create a virtual machine request (SPW5\_108).
* Reliability: Students will be notified of bad input.
* Performance: UI allows student to create multiples requests on the same page.
* The system shall allow the head professor to delete a virtual machine request (SPW5\_119).
* Reliability: Head professor will be notified to confirm his action.
* The system shall allow the head professor access the virtual machine requests page from his email (SPW5\_111).
* The system shall allow the head professor to modify the settings of a virtual machine request (SPW5\_109).
* Reliability: Head professor will be notified of bad input.
* Performance: UI allows head professor to modify several requests at the same time.
* The system shall allow the head professor to filter the virtual machine requests (SPW5\_118).
* The system shall allow the head professor to notify the school’s system admin about virtual machine creation (SPW5\_110).
* Reliability: Head professor will be notified of bad input.
* The system shall allow the head professor to add an image’s name (SPW5\_113).
* Reliability: Head professor will be notified of bad input.
* The system shall allow the head professor to edit an image (SPW5\_117).
* Reliability: Head professor will be notified of bad input.
* The system shall allow the head professor to delete an image (SPW5\_116).
* Reliability: Head professor will be notified to confirm his action.
* The system shall allow the head professor to filter existing images (SPW5\_114).
* The system shall allow the head professor to set the default email for notification (SPW5\_112).
* Reliability: Head professor will be notified of bad input.
* The system shall allow the head professor to change the image status (SPW5\_115).

**These following use cases pertain to SPWv.4**

* The system shall allow the head professor to run a matching algorithm to match students to projects with some customization by him (SPW4\_100)
  + Usability: During the process the UI will be explained and to an extent intuitive and descriptive.
  + Performance: The algorithm will function better than the prior on average. And functions on “good” particular inputs quickly.
* The system shall allow the head professor to rank projects from 0 to 100. (SPW4\_101)
  + Reliability: Head professor will be notified of their bad input
* The system shall allow students to rank projects from -1 to 100. (SPW4\_102)
  + Reliability: Student will be notified of their bad input

Taken from SPWv.3 requirements document 4.1.

1. The following uses cases were defined by SPWv.1

* The system shall allow non-registered users to register as clients, professors or students using proprietary authentication system (SPW\_101)
* The system shall allow registered users to login as clients, professors or students using Google accounts (SPW\_102)
* The system shall allow registered users to login as clients, professors or students using Facebook accounts (SPW\_103)
* The system shall allow registered users to login as clients, professors or students using LinkedIn accounts (SPW\_104)
* The system shall allow registered users to login as clients, professors or students using proprietary authentication system (SPW\_105)
* The system shall allow filling students, professor and clients’ profile from LinkedIn (SPW\_201)
* The system shall allow students, professors and clients to edit their profile (SPW\_202)
* The system shall allow students, professors and clients to browse users’ profiles (SPW\_203)
* The system shall allow students, professors and clients to browse projects (SPW\_204)
* The system shall allow students, professors and clients to browse projects and users by skill (SPW\_205)
* The system shall allow students, professors and clients to browse past projects (SPW\_206)
* The system shall provide an ordered list with suggested projects for students, professors and clients (SPW\_301)
* The system shall provide an ordered list with suitable candidates for possible mentors (SPW\_302)
* The system shall provide an ordered list with suitable candidates for possible team members (SPW\_303)
* The system shall allow mentors and team members to edit their projects (SPW\_304)
* The system shall allow students, professors and clients to propose a project (SPW\_401)
* The system shall allow students, clients and professors to join a project during the first period of the semester (SPW\_402)
* The system shall allow students and mentors to leave a project during the first period of the semester (SPW\_403)
* The system shall allow team members and mentors to invite other users to their project during the first period of the semester (SPW\_404)
* The system shall allow head professor to accept/reject projects proposals (SPW\_405)
* The system shall allow administrator to do all previous actions (SPW\_501)
* The system shall allow head professor to activate/delete users (SPW\_502)

1. Next, the uses cases implemented for the Senior Project Website Version 2. Some were mentioned in the version 1 documentation but were not implemented; others were already developed in the previous version and will only suffer little modification while others were implemented from scratch. In addition we will consider security use cases.

* The system shall allow students enrolled on Senior Project Class to log in using FIU panther email software (SPW2\_101)
* The system shall allow non-registered users to login as guests. (SPW2\_102)
* The system shall allow professors to login using SPW authentication system. (SPW2\_103)
* The system shall allow students and professors to logout from the system (SPW2\_104)
* The API shall validate if a student is enrolled on Senior Project class. (SPW2\_105)
* The API shall provide a method to refresh SPW system with the latest info from the file. (SPW2\_106)
* The API shall create new students profiles.(SPW2\_107)
* The API shall delete students profiles for inactive (not registered on Senior Project class) students. (SPW2\_108)
* The API shall delete projects created for inactive students and that have status pending professor approval. (SPW2\_109)
* The API shall provide a list with information from all students enrolled on Senior Project class and the title of their project. (SPW2\_110)
* The system shall allow students and professors to edit their profile (SPW2\_201)
* The system shall allow students and professor to sync their profile with LinkedIn (SPW2\_202)
* The system shall allow students and professors to browse users’ profiles (SPW2\_301)
* The system shall allow students and professors to browse projects (SPW2\_302)
* The system shall allow students and professors to browse past projects (SPW2\_303)
* The system shall allow students and professors to create a project (SPW2\_401)
* The system shall allow students to join a project before the deadline period (SPW2\_402)
* The system shall allow students to leave a project before the deadline period (SPW2\_403)
* The system shall allow head professor to assign students to any project (SPW2\_404)
* The system shall allow professors to assign students to a project they created (SPW2\_405)
* The system shall allow head professor to remove students from any project (SPW2\_406)
* The system shall allow professors to remove students to a project they created (SPW2\_407)
* The system shall allow a student and a Professor to delete a project they created (SPW2\_408)
* The system shall allow head professor to delete any project (SPW2\_409)
* The system shall allow head professor to change a project status (SPW2\_410)
* The system shall allow head professor to assign a professor as a Mentor to a project (SPW2\_411)
* The system shall allow a registered user to edit a project (SPW2\_412)
* The system shall allow head professor to activate/deactivate users (SPW2\_501)
* The system shall allow head professor to add other professors to the system (SPW2\_502)
* The system shall allow head professor to set the join/leave/propose project time period (internal deadline). (SPW2\_503)
* The system shall allow registered users to upload a profile picture from local storage. (SPW2\_601)

1. Next, the uses cases implemented for the Senior Project Website Version 3.

* The system shall allow registered users to upload a new file to the project repository. (SPW3\_710)
* The system shall allow registered users to download a file from their repository. (SPW3\_720)
* The system shall allow registered users to delete file(s) from their repository. (SPW3\_730)
* The system shall allow the head professor to add a new milestone to the repository structure. (SPW3\_901)
* The system shall allow the head professor to edit the milestones used in the academic semester. (SPW3\_902)
* The system shall allow the head professor to delete milestones from the repository structure. (SPW3\_903)
* Users may download a zip file containing all documentation in a given project. (SPW3\_911)
* The system shall allow the head professor to manually add users to the database of users. ( SPW3\_912)
* The system shall allow the head professor to delete users from the database of users (SPW\_913)
* The system shall allow the head professor to modify users that are a part of the SPW (SPW\_914)
* The system shall allow the head professor to impersonate any other user and act on his behalf (SPW\_915)
* The system shall allow the head professor to bypass an activation link sent to new users (SPW\_916)
* The system shall allow the head professor to filter users based on their attributes to find a group of users more easily (SPW\_917)
* The system shall allow users to register for access to the SPW (SPW\_918)
* The system shall allow users to activate their accounts via an email activation link (SPW\_919)
* The system shall allow the head professor and students to rank projects for the matching algorithm.(SPW3\_205)
* The system shall allow the head professor to use the matching algorithm. (SPW3\_206)
* The system shall allow the head professor to change the settings of the algorithm per his criteria. (SPW3\_210)
  1. Analysis of System Requirements

Note: Details discussed here expanded upon in the appendix. See appendix A and B for Used Case discussion, appendix C for the static model, and appendix D for the dynamic model.

* + 1. Scenarios

This section pertains to scenarios for SPWv5.

SPW5\_108:

Lisa, a student, wants to create a virtual machine request. She tries to enter negative or non-numeric values into the input fields for “RAM”, “Storage” and “Number of VM”. The system tells her she cannot do it (SPW5\_108 Reliability). Finally, she enters numeric values greater than zero into the virtual machine settings. She then selects from the “Image Name” drop down menu the image of preference and submits her request.

SPW5\_109:

The head professor wants to modify a virtual machine request and by mistake enters a non-numeric value into one of the input fields of the VM settings. The system tells him that he cannot do it (SPW5\_109 Reliability). He realizes his error and enters a numeric value greater than zero and clicks the “Submit” button.

SPW5\_110:

The head professor wants to notify the school’s system admin about some virtual machine request for creation. By mistake he enters in the input field a non-valid email address. System tells him that he cannot do it (SPW5\_110 Reliability). After realizing his error, he enters a valid email address, and changes the status of the request to “APPROVED” and clicks on “Submit” button.

SPW5\_112:

The head professor wants to change/set the default email and name for virtual machine creation, he enters in the email input field a non-valid email address. System tells him that he cannot (SPW5\_112 Reliability). Then, he inputs a valid email address and clicks on the button “Set Default Email”.

SPW5\_113:

The head professor wants to add a new image’s name into the system, but he did not realized that the image already existed. System tells him that he cannot do it (SPW5\_113 Reliability). Finally, he realizes the mistake, enters a new image name, and clicks on the button “Add Image Name”.

SPW5\_114:

The head professor wants to filter existing images in the system. He looks for all images that are Windows related, then he enters on the image name filter option the word “Window” and hits the enter key. System prompts him with all existing images that have window in their names.

SPW5\_115:

The head professor want to change the status of the image Windows Server 2000 from “ACTIVE” to “INACTIVE”. He looks for the image name in the system and clicks on the green light icon. System successfully changes the image status to “INACTIVE”.

SPW5\_116:

The head professor wants to delete an image from the system Windows Server 2000. He looks for the image and clicks on the red X icon. System prompts a message asking him to confirm if he wants to delete the corresponding image from the system (SPW5\_116 Reliability). He then clicks on the “OK” button to confirm the action.

SPW5\_117:

The head professor want to edit Windows Server 2000 image from the system. He looks for the image name in the system, but by mistake he leaves the image name field empty. System tells him that he cannot proceed (SPW5\_117 Reliability). Finally, he realizes his error and enters a new name on the input field and clicks on the “Submit” button.

SPW5\_118:

The head professor wants to filter an existing virtual machine requests in the system. He looks for all images that are Windows related. He then enters on the image name filter option the word “Window” and hits the enter key. System prompts him with all existing virtual machine requests that have window as image of choice.

SPW5\_119:

The head professor wants to delete from the system a virtual machine requests that have Windows Server 2000 as image of choice. He looks for the request and clicks on the red X icon. System prompts a message asking him to confirm if he wants to delete the corresponding virtual machine request from the system (SPW5\_119 Reliability). He then clicks on the “OK” button to confirm action.

**This section pertains to scenarios for SPWv.4. Told through a narrative of how it is used.**

The head professor, Masoud, wants to match students to projects. Before class he sets the deadline for students to rank projects and the minimum they must rank by logging on to the SPW and going to admin. There he set the minimum to 3 and the deadline to the spring of 2014(note these actions have not changed from SPWv3).

He informs students to make sure they are registered for the senior project website and to login thoroughly create their profile or sync with LinkedIn. And then rank 3 projects. He informs them that if they wish they can just not rank any project and they will be matched to projects that need them and their particular skills.

**SPW4\_102:**

Amy student who likes to be in particular projects after setting up her account tries to rank only one project. The system tells her she cannot. Frustrated she tries to rank projects with letters. She is then told letters are not allowed for ranking. She then decides to rank projects outside of the instructed rubric of -1 to 100 **but is told she cannot** (SPW4\_102 Reliability). Eventually she ranks 3 projects appropriately and saves her rankings. All other students rank projects or just leave it up to the system.

**SPW4\_101:**

Eventually the time for ranking projects is over and the head professor goes to the match tab after logging in. There he goes to project priority. There he ranks projects inappropriately accidently but the system informs him of his transgressions and **stops his rankings** (SPW4\_101 Reliability). **He ranks some projects 1’s to not worry about them intensely (OP) and ranks other projects he worries greatly about varying scores (2-100) based on his perceived importance of them.** (SPW4\_101 Usability)

**SPW4\_100:**

After ranking projects like he wanted he goes back to the match page via the match tab and clicks run match. He clicks run a match. **There he has the ability to run VIP manually or automatically.** (SPW4\_100 Usability) Feeling lucky he runs automatically. There he is given feedback regarding what the matching’s do, there are two categories **one for students and one for projects** (SPW4\_100 Usability). At a glance, after acclimating and reading up on the system, **he can tell** (SPW4\_100 Usability) that he likes some project matches in students but a few he rather dislikes.

So he goes back to match, run match, and tries running manually. There he can choose students between both categories. He tries picking more than the project maximum, by accident, **but is stopped** (SPW4\_100 Reliability). He also notices **he cannot select** (SPW4\_100 Reliability) the same student in both categories. He tries to continue running but the system **asks to confirm** (SPW\_100 Usability) if he is sure given that he accidently ranked 2 instead of 3 students. He did not notice so he goes back and puts a third can continues.

In the background he notices the **matchmaking is rather fast with a few stutter now and then but nothing too bad** (SPW4\_100 Performance). While going through the matchmaking he feels he selected his more important VIPs and clicks **“run the rest automatically”** (SPW4\_100 Usability). At the end he chooses **one of two modes** (SPW4\_100 Usability). He then sees a page showing his entire VIP match; there **he can see** (SPW4\_100 Usability) how the matchmaking fares along with many metadata.

He continues and sees the OP match and **chooses** the more project oriented option between “for student” or “for project” (SPW4\_100 Usability). He is then shown the final match page to be able to judge if the system’s match was adequate enough, he thinks so, so he clicks “finalize matchmaking”, he is **asked to confirm** database change (SPW\_100 Usability) which he does. Finally the matchmaking process is done and the database change is propagated throughout the system.

* + 1. Use Case Model

The use case model describes the basic functionality of this system to create, filter, delete and modify a virtual machine request. Also, the system reflects functionality on add, filter, delete and edit an image from the system. There are three actors in this model: the student, head professor, and email system. The student is allowed to create a virtual machine request, while the head professor can delete, filter, modify and send for creation a virtual machine request via email. In addition, the head professor is able to add, filter, edit and delete images in the system.

* + 1. Static Model

A total of three views were added to the Static Model to account for the request of virtual machine environments and image names in the system. In the model structure, a new model was added to support the data insertion and retrieval needs of this system. From the point of view of the controller, no new controller was added. The project controller and administrator controller were both expanded with a few more functions to support the new functionalities added to the system.

* + 1. Dynamic Model

The dynamic model shows the behavior of the system such as the messages sent and their interaction. This section contains functional specification of the sequences diagram in a straightforward way.

When creating a virtual machine request, the dynamic model starts when the actor “student” navigates to the corresponding page and selects from the “Image name” drop down menu one of the options available in the system. The student then inputs in the fields RAM, Storage, Number of VM the values of choice. Once the student clicks on the “Submit” button, the controller takes the data submitted by the student and passes it to the vm\_request model that inserts the request into the database.

When modifying a virtual machine request, the dynamic model starts when the actor “head professor” first navigates to the corresponding page and enters in the settings the values of choice for the virtual machine request. Then he clicks on the “Submit” button and the controller sends the information to be updated in the database through the vm\_request model.

When notifying the school’s system admin for virtual machine creation, the dynamic model starts when the actor “head professor” navigates to the corresponding page, then selects from the “STATUS” drop down menu the option “APPROVED” and clicks on the “Submit” button. The controller processes the data and collects all virtual machine with approved status to send an email to school’s system admin.

When setting default email notification, the dynamic model starts when the actor “head professor” inputs the new name and email address on the input fields for full name and email, then clicks on “Set Default Email” button. The controller then communicates with the vm\_request model to update the data into the database.

When adding a virtual machine image, the dynamic model starts when the actor “head professor” clicks on the button “VM Images”, the action then goes to the project controller that creates the vm\_images page. Then the head professor enters the image name in the input field and clicks on the “Add Image Name” button. The controller then calls the vm\_request model to insert the new image name into the database.

When setting the image status, the dynamic model starts when the actor “head professor” clicks on the button/link corresponding to the specific image. Then the controller changes the image status.

When filtering images, the dynamic model starts when the actor “head professor” inputs in the image name filter option (the full or partial name of an image) and/or selects from the drop down menu filter the status of choice. The controller then retrieves the filtered options.

When deleting an image, the dynamic model starts when the actor “head professor” clicks on the link/button corresponding to the specific image. Then the controller executes the action.

When filtering virtual machine requests, the dynamic model starts when the actor “head professor” inputs in the filter options the RAM, Storage, Number of VM and/or the image name, student’s full/partial name, and term name of choice. Then the controller processes requests/ prompts and filters virtual machines request.

When editing an image, the dynamic model starts when the actor “head professor” inputs in the field image name the new name and clicks on the “Submit” button. Then the new image name is created.

When deleting a virtual machine request, the dynamic model starts when the actor “Head Professor” clicks on the link/button corresponding to the specific virtual machine request. Then the controller deletes the virtual machine request.

1. System Design

*Senior Project Website V5*



This section will expand on the details of the system’s overall design. It will give the reader a more thorough overview of the system and subsystem decomposition, how hardware and software is mapped, the manner that persistent data is defined, and how security and privacy is utilized.

* 1. **Overview**

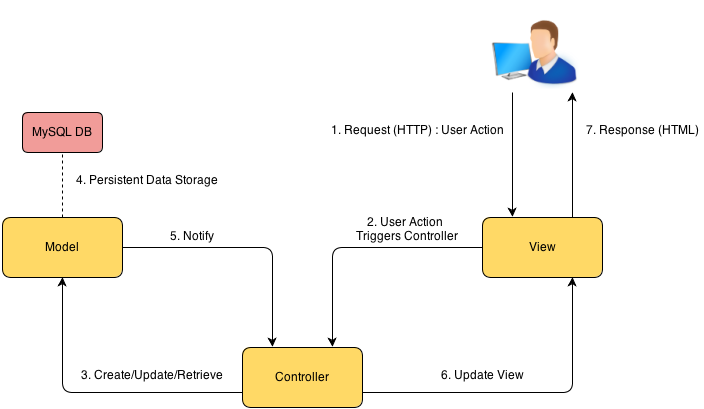
The SPW’s system is based upon the Three-Tier and MVC architectures. It is further decomposed into 7 subsystems: virtual machine, matchmaking, login, project, user management, repository, and Rest API. The main objective of SPWv.5 was tuning the existing functionality of the system plus add the new virtual machine subsystem.

For deployment in hardware and software, the architecture is based on client and server, wherein a browser (client) accesses the SPW application (on a server). In addition, the API works as an API for other senior projects requiring data on registered users.

The persistent data change with respect to the previous one (SPWv.4). In SPWv.5 a total of three new tables were added to handle the virtual machine requests, virtual machine images, and the default school’s system admin info. The rest of the tables remained the same, containing tables for users, language, projects, experience, skills, milestones, rankings, etc. Security and privacy features will remain the same as in SPWv.4. Having a one-way password encryption, session encryption, etc.

* 1. **Subsystem Decomposition**

The current system (SPWv.5) follows the system architecture of Three-Tier and MVC as a means to define how the system is structured. This structure continues being the same for SPWv.5. The style fits perfectly the function of the SPW, controllers maintain business logic for interactions, views allow users to observe relevant data, and models hold the means to access data from the database. For a user, that flow of logic is functional. Their interactions on the views prompts the controller to respond, and the controller requests database data from models when required. From a three-tier perspective, the user’s browser and what they see (the view) is the presentation tier, the controller’s reaction to the user’s interaction is the logic tier, and the controllers request for data from the model is the data tier. Refer to picture below for a graph describing the nature of the system from SPWv.4 design document.



The SPW is further decomposed to 7 subsystems. The virtual machine subsystem the most recently added therefore it will be discussed first. For the purposes of documentation, the remaining 6 subsystems will be described adapted from what was said in SPWv.4 design document.

**Virtual Machine Subsystem**: The subsystem will handle all the necessary logic for the request of virtual machines (creating virtual machine request, modifying virtual machine request, deleting virtual machine request, filtering virtual machine request, etc.). Allowing students to customize their virtual machines needs and giving to the head professor the means to manage students’ virtual requests.

**Matching Algorithm Subsystem:** This subsystem will facilitate the matching of students to projects by the head professor. It will function in a two-phase matchmaking system where the head professor’s very important projects are filled up first taking into account student interest and the head professor’s interest, with greater emphasis on the latter’s influence. The second phase emphasizes the student’s interest in projects the head professor has little need to put emphasis on. These will be match via NRMP with customization with regards to focus on student interest only or compromise with project’s need for optimization.

**Login Subsystem:** This subsystem will deal with all the logic involving the authentication of the users who use SPW.

**Projects Subsystem:** This subsystem will deal with all the logic that required in relation to projects. Such as proposing a project.

**User Management Subsystem:** This subsystem will deal with user management by the head professors (adding, deleting users, acting as users, etc). Also includes functionality such as synchronizing with LinkedIn, changing passwords, functionality involving the changing of a user’s state essentially.

**Repository Subsystem:** This subsystem will handle all the logic necessary to establish the project files repository functionality. Deals with file upload for projects and milestones (goal for project), allow team sharing, and allows the head professor to organize the structure of the repository.

**Rest API Subsystem:** The SPW will be accessing an external API that is going to be developed as a RESTful service. In essence, this API will be used to verify that a student who wishes to register with the SPW is indeed a student that is currently enrolled in the Senior Project course. The API was developed as part of the SPWv.2 project and only allows students that are currently enrolled in the senior project course to login as students. Without this API, any student with an FIU email account would be allowed access to the system. It is important to note that other services (developed by other senior project teams) will also access this API in order to conduct validation of their student users. Of note Jonathan Santiago of the Collaborative Platform team of Summer 2014 did work on this subsystem for SPWv.4 as it was broken and not functional.

* 1. **Hardware and Software Mapping**

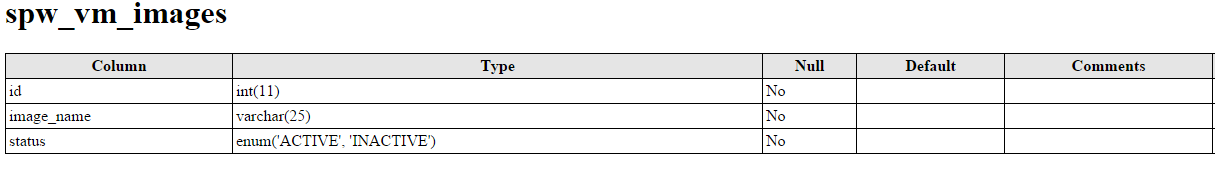
This aspect for SPWv.5 is largely unchanged from the description of SPWv.4 in their design document. The hardware and software are essentially based on three architectural patterns. In a commonplace client-server architecture, the client uses a web browser to access the server running the SPW application. This application interacts with a MySQL server for database needs and runs under an Apache Tomcat server.

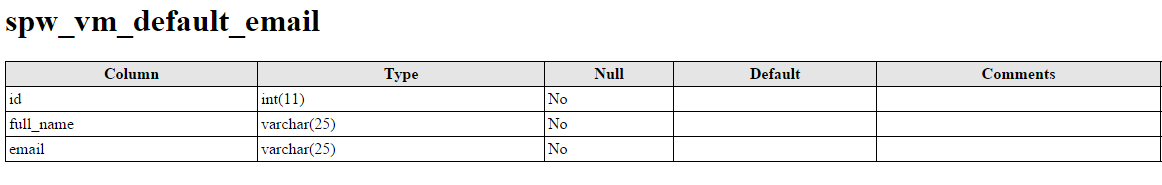
Additionally the server will run an API to verify members in the SPW amongst other projects in the senior project class. Below is a display of this from SPWv.3 design document.

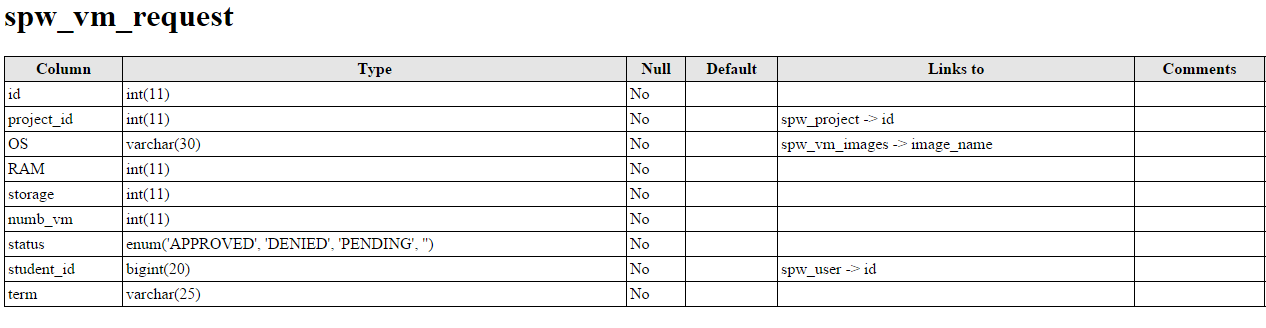


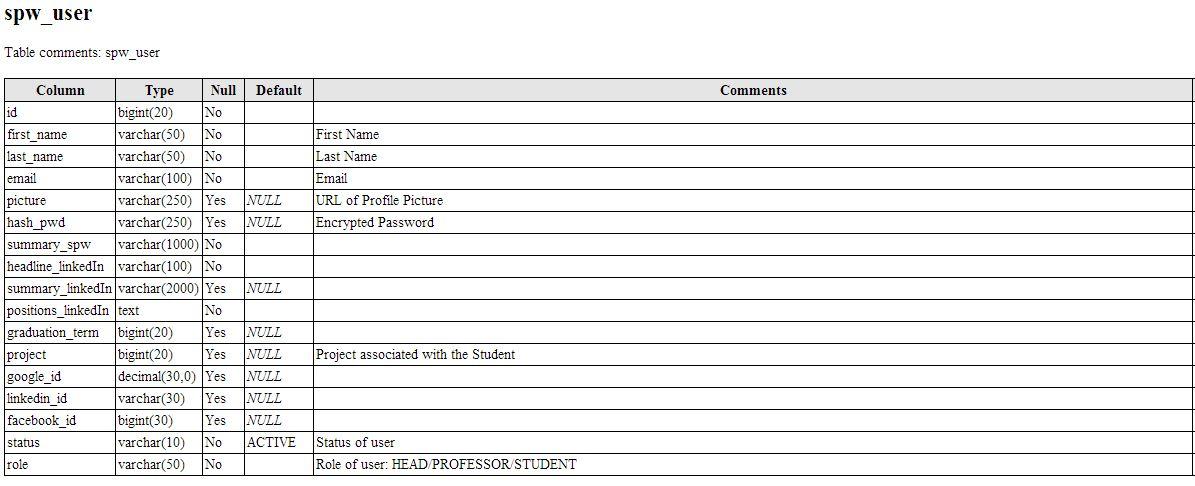
* 1. **Persistent Data Management**

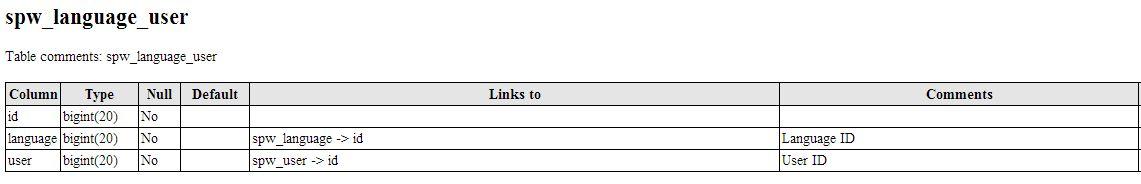
The persistent data did change from SPWv.4 to SPWv.5. Three new tables were added into the system, the rest of the tables were kept as from previous version SPWv.4.

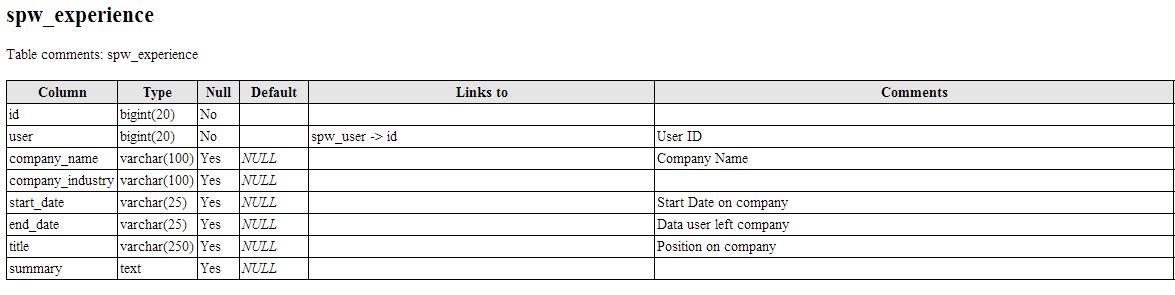
**Data Dictionary for SPW (from SPWv.4 design document):**

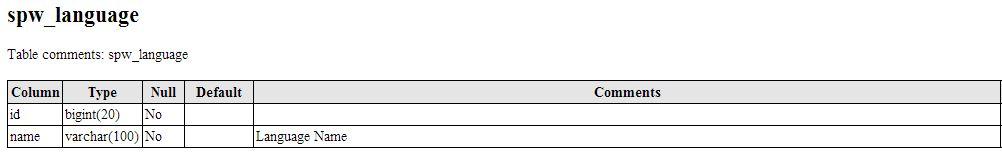


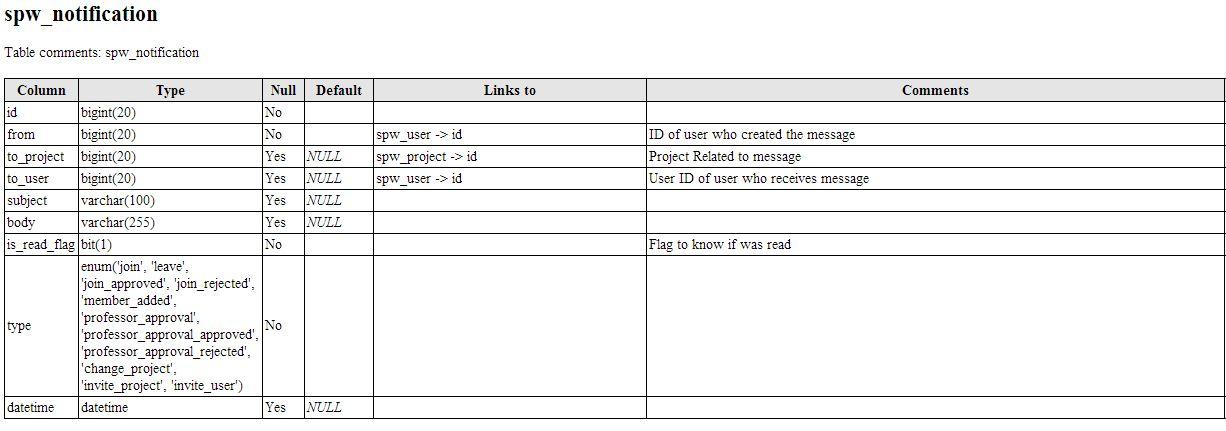


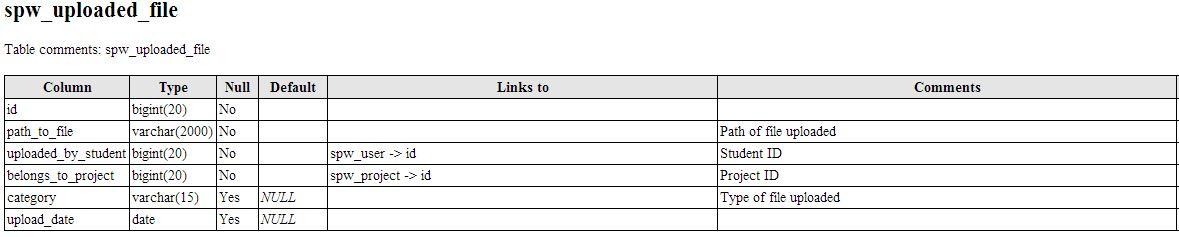
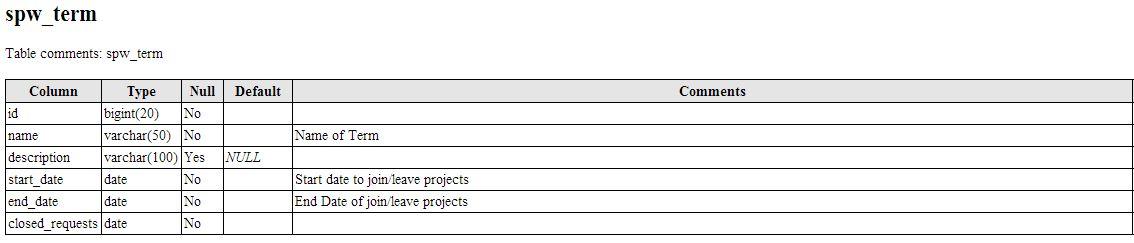
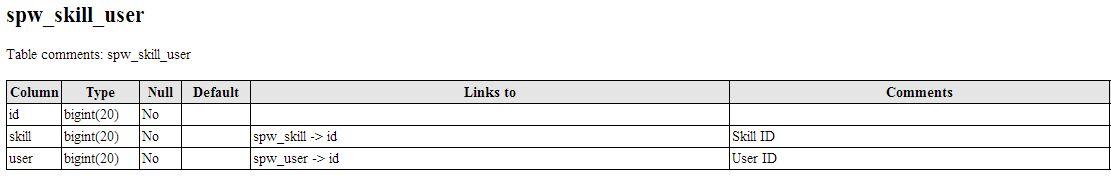
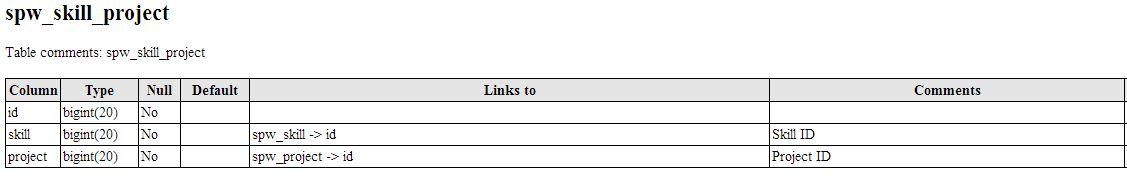
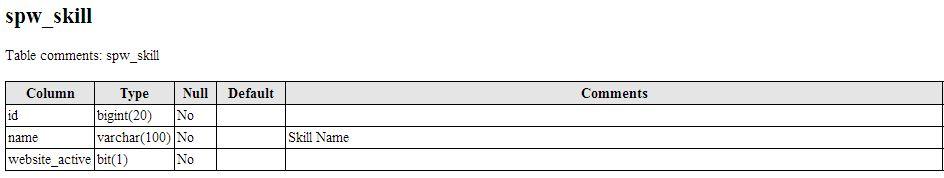
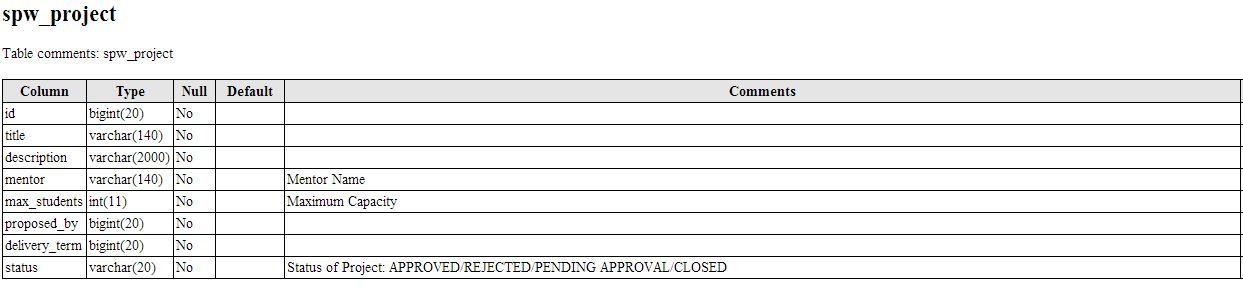


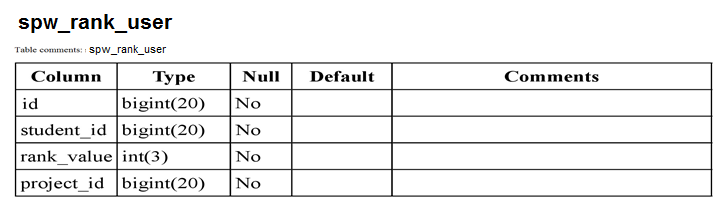


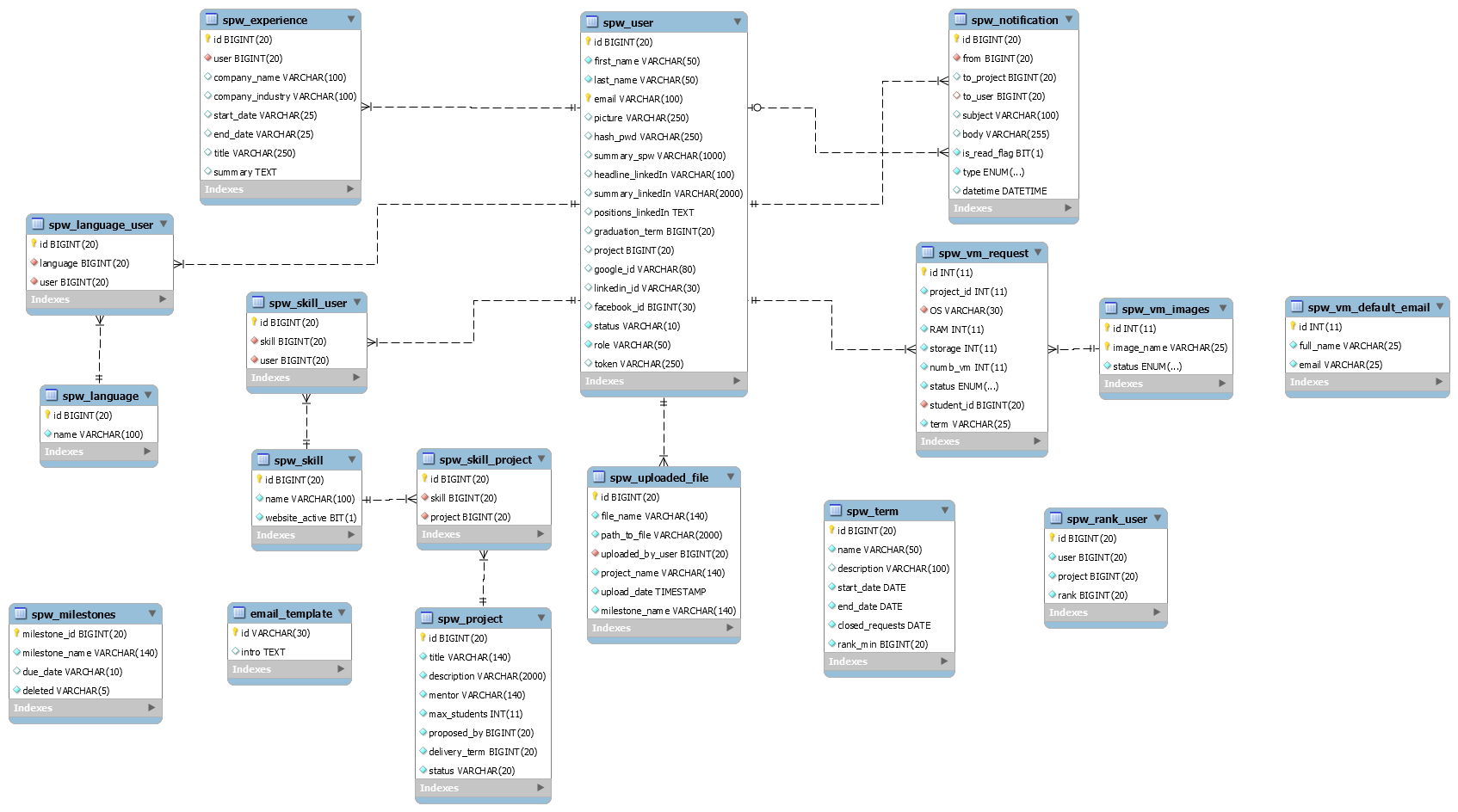










****

Entity-Relationship Diagram for Model Subsystem (From SPWv.3 Final Document)

* 1. **Security/Privacy**

No changes to any of the following occurs going from SPWv.4 to SPWv.5. The following information is entirely from SPWv.4 design document.

SQL Injection

System takes care of avoiding SQL injection attacks by using the Active Record class from the Code Igniter framework for all interaction with SPW Website, which produces safer queries since the values are escaped automatically by the system.

URI Security

Malicious data can be passed to the application via the URI strings. This is prevented in our system by the use of the Code Igniter framework which is fairly restrictive regarding which characters it allows in the URI strings.

Cross-site scripting (XSS)

Cross scripting is also prevented by the use of the Code Igniter Cross Site Scripting Hack prevention filter which has been configured to run automatically to filter all POST and COOKIE data that is encountered.

Cross-site request forgery (CSRF)

Cross site forgery has been also addressed by enabling the CSRF protection on Code Igniter, which turns on a complicated algorithm that allow SPW website to know if the data it receives actually comes from a form on this website, avoiding with it this type of attacks.

Session Encryption

SPW uses the Code Igniter Session class to manage user session information, which is stored as serialized and encrypted in a cookie.

One-Way password encryption

SHA-1 function is used to encrypt users’ passwords in database, which allow website to keep secured user’s password for the website and therefore all their profile information in the website.

The SHA-1 algorithm has two important properties that make of it the more secure algorithm website could use to secure user’s information. The first one is that is a one-way hashing algorithm, which makes impossible to revert back an encrypted output to the initial, plain-text input, and that any given input always maps to the same encrypted value. This ensures that the passwords stored on the server cannot be deciphered by anyone. This way, even if an attacker gains reading permission to the user table, it will do him no good.

It is important to add that no unnecessary information is stored for a user and email addresses are only displayed to registered users.

1. Detailed Design

*Senior Project Website V5*



These following sections cover on greater detail the subsystems described in section 5.

6.1. Overview

On this section the Virtual Machine subsystem will be featured with regards to its static and dynamic models of representation; the rest of the models remains the same. Prior document specifications will be added and mentioned as such.

**6.2. Static Model**  
A new model and views was added to SPWv.5 which supports the request of virtual machines and virtual machine images to the system. The project controller also had more methods added. What follows is adapted from the SPWv.4 design document and as before, version specific changes are at top of this section and noted as such.

The existing system was built using the Code Igniter framework. These class diagrams display the influence of Code Igniter upon the structure of the model and controller subsystems. Each model and controller defined to be part of the system inherits from CI\_Model and CI\_Controller, respectively. The data that is defined to be part of the each model subsystem is stored in a relational database. An entity relationship diagram shows how the different entities (representative of models) are related to each other in the database schema. The class diagram for the SPW-Register API will also be provided in this subsection. The view subsystem identifies the different views that are part of the system.

SPW makes use of the Front Controller design pattern as it is embedded within Code Igniter. It is used to initialize the base resources needed to run the website. If there are frequent users and they still have a cache file still existent, that cache file will be sent directly to the browser bypassing normal system execution and loading model, helpers, libraries and any other resources required to run the system. The behavior of this pattern is defined in the index.php from Code Igniter framework resources and will act as a main entrance to the webpage.

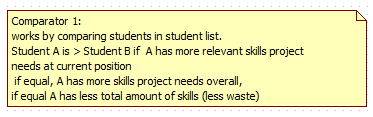
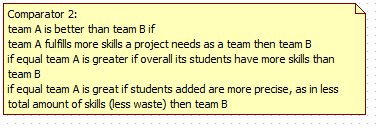
The *Singleton design pattern* is used to define the Term Deadline class. This class represents the available times for students to interact with the projects (propose, join, leave, etc) and it is modified solely by the Head Professor of the class. Only one instance of the class is necessary in the scope of SPW. The pattern ensures that no more than one instance is created at all times.

**6.3. Dynamic Model**

This section will feature the description of the core of the matchmaking two-phase algorithm showing both phases’ algorithm details. The accompanying dynamic diagrams are in the appendix. The two algorithms is one based on heuristics to cut down on search time for an optimal team. The other is the NRMP algorithm as a means to allow students to choose their projects with the possibility of compromise with regards to projects having good teams. The purpose of two phase is the underlying idea that the head professor really only cares about certain projects being optimized.

The specific algorithm details for the heuristic VIP matchmaking are detailed below

1. Given a project with X skill needs and Y positions
2. Go through the Y positions of the projects (going down until all position filled Y==0)
   1. For each student compose relevant skill array (skills project needs at the moment they have)
   2. Sort the student list by comparator 1 see below
   3. For each student
      1. Save relevant skill to has table
      2. If best team not initialized add student to current team and goto to 2. With Y-1 positions and X set difference (Students skills) skills
      3. If student’s relevant skills is less than ceil(count(X)/Y) exit 2.b.
      4. If student’s relevant skill in hash table go to 2.b. to next student
      5. If position Y is not 1, add student to current and goto to 2. With Y-1 positions and X set difference (Students skills) skills
      6. If position is 1, add student to current and compare current to best,
         1. if current better (See comparator 2) and fulfills all project skills exit 2. and best = current
         2. else go to 2.c.
   4. If exit go back to Y+1 2.b. removing latest addition to current
3. Best team is best team



The specific algorithm details for National Residency Matchmaking

Following is NRMP Matchmaking Algorithm Details

1. Select criteria to displace students as friendly (displace student who wants project least with one who wants it most) or compromise (displace student who contributes least to the project least with one who contributes more)
2. For each student unmatched
   1. Go through each project not yet traversed in the order they ranked them
      1. If position open student is matched and removed from unmatched, continue to 2.
      2. Else if student is better than the worst one based on criteria, add student to project, removed from unmatched and put worst student to unmatched, continue to 2.
   2. If student reached here and is unmatched remove from unmatched and put in unmatchable
3. End result students are matched or said to be unmatchable (i.e. their ranking ended with no match for them or too few projects)

**6.4. Code Specification**

Code specification details remain unchanged from SPWv.4. The following is almost entirely from SPWv.4 design document regarding that section.

The SPWv.1 and SPWv.2 were built using the Code Igniter framework, which is a powerful, lightweight PHP framework that enforces the MVC architecture. Therefore, the code of the SPWv.1 and SPWv.2 reflect the implementation of this pattern by having three main subsystems: model, view, and controller. The SPWv.3 will continue to use Code Igniter for its development, thus SPWv.3 will also maintain the MVC architecture. As a result, the code specification from SPWv.1 and SPWv.2 remain unchanged in SPWv.3.

For the Controller subsystem all classes inherit from the CI\_Controller, a class from the Code Igniter framework that handles all URI requests. For the Model subsystem all classes inherit from the CI\_Model, a class from the Code Igniter framework that uses a modified version of the Active Record Database Pattern. This pattern allows information to be retrieved, inserted, and updated in your database with minimal scripting. Therefore by inheriting from this class, all SPW model classes are able to use the Active Record Database Pattern capabilities implemented by the Code Igniter framework. For the View subsystem, all views and sub views are comprised of HTML and CSS code. All main views are inside the View package and use the header and footer views. Besides the main view, the SPW contains views related to each subsystem that will display to the user the appropriate information for that subsystem.

*Senior Project Website V5*



7. System Validation

In this chapter the testing process is described. It involved both static and dynamic testing approaches. Each defined test case has a test case id, purpose, test setup, input and expected output.

**7.1 Subsystem Tests**

* 1. **. System Tests**

Note corresponds to used case id plus a ‘T’#testID

|  |  |
| --- | --- |
| SPW5\_108T1 - Sunny | |
| Purpose | Test if the system allows students to create a virtual machine request |
| Test Setup | - Student is logged in  - Student went to My Project tab and clicked on Create VM-Request button |
| Input | - Numeric values greater than zero  - User clicks submit button |
| Expected Output | The new virtual machine request is added to the database with a status of pending. The head professor is notified via email about the new request. |

|  |  |
| --- | --- |
| SPW5\_108T2 - Rainy | |
| Purpose | Test if the system allows students to create a virtual machine request |
| Test Setup | - Student is logged in  - Student went to My Project tab and clicked on Create VM-Request button |
| Input | - Non-numeric values  - User clicks submit button |
| Expected Output | System prompts error message “value must be numeric and greater than zero” |

|  |  |
| --- | --- |
| SPW5\_109T3 - Sunny | |
| Purpose | Test if the system allows head professor to modify a virtual machine request |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - Numeric values greater than zero  - User clicks submit button |
| Expected Output | Virtual machine request is updated in the database |

|  |  |
| --- | --- |
| SPW5\_109T4 - Rainy | |
| Purpose | Test if the system allows head professor to modify a virtual machine request |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - Non-numeric values  - User clicks submit button |
| Expected Output | System prompts error message “value must be numeric and greater than zero” |

|  |  |
| --- | --- |
| SPW5\_119T5 - Sunny | |
| Purpose | Test if the system allows head professor to delete a virtual machine request |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User clicks delete icon |
| Expected Output | System deletes virtual machine request |

|  |  |
| --- | --- |
| SPW5\_119T6 - Rainy | |
| Purpose | Test if the system allows head professor to delete a virtual machine request |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User clicks delete icon |
| Expected Output | System prompts confirmation message “Are you sure you want to delete virtual machine from user’s name” |

|  |  |
| --- | --- |
| SPW5\_118T7 - Sunny | |
| Purpose | Test if the system allows head professor to filter virtual machine requests |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User enters a value in RAM filter  - User hits the enter key |
| Expected Output | System prompts the filtered virtual machine requests |

|  |  |
| --- | --- |
| SPW5\_118T8 - Rainy | |
| Purpose | Test if the system allows head professor to filter virtual machine requests |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User does not enters a value in RAM filter  - User hits the enter key |
| Expected Output | System prompts all values for RAM |

|  |  |
| --- | --- |
| SPW5\_110T9 - Sunny | |
| Purpose | Test if the system allows head professor to notify the school’s system admin about virtual machine requests |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User selects from the status drop down menu the APPROVED option  - User clicks on Submit button |
| Expected Output | System prompts with success message and sends an email to the school’s system admin |

|  |  |
| --- | --- |
| SPW5\_110T10 - Rainy | |
| Purpose | Test if the system allows head professor to notify the school’s system admin about virtual machine requests |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Requests button |
| Input | - User selects from the status drop down menu the APPROVED option  - User enters in the email input field an invalid email address  - User clicks on Submit button |
| Expected Output | System prompts invalid email address error message |

|  |  |
| --- | --- |
| SPW5\_113T11 - Sunny | |
| Purpose | Test if the system allows head professor to add an image name |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - Image name  - User clicks on Add Image Name button |
| Expected Output | The new image name is added to the database with a status of active |

|  |  |
| --- | --- |
| SPW5\_113T12 - Rainy | |
| Purpose | Test if the system allows head professor to add an image name |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - existing image name  - User clicks on Add Image Name button |
| Expected Output | System prompts error message “Image’s name already exists in the system” |

|  |  |
| --- | --- |
| SPW5\_117T13 - Sunny | |
| Purpose | Test if the system allows head professor to edit an image name |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - Image name  - User clicks on Submit button |
| Expected Output | The image name is updated in the database |

|  |  |
| --- | --- |
| SPW5\_117T14 - Rainy | |
| Purpose | Test if the system allows head professor to edit an image name |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - image name input field is empty  - User clicks on Submit button |
| Expected Output | System prompts error message “Image’s name cannot be empty” |

|  |  |
| --- | --- |
| SPW5\_116T15 - Sunny | |
| Purpose | Test if the system allows head professor to delete an image |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User clicks delete icon |
| Expected Output | System deletes image and virtual machine requests associated with the image |

|  |  |
| --- | --- |
| SPW5\_116T16 - Rainy | |
| Purpose | Test if the system allows head professor to delete an image |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User clicks delete icon |
| Expected Output | System prompts confirmation message “Are you sure you want to delete image’s name” |

|  |  |
| --- | --- |
| SPW5\_114T17 - Sunny | |
| Purpose | Test if the system allows head professor to filter images |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User enters partial image name in the filter option  - User hit enter key |
| Expected Output | System prompts all images that have within its name the filtered option |

|  |  |
| --- | --- |
| SPW5\_114T18 - Rainy | |
| Purpose | Test if the system allows head professor to filter images |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User does not enter a value in image name filter  - User hit the enter key |
| Expected Output | System prompts all image names |

|  |  |
| --- | --- |
| SPW5\_115T19 - Sunny | |
| Purpose | Test if the system allows head professor to change the image’s status |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User clicks on change image status icon |
| Expected Output | System changes the image status |

|  |  |
| --- | --- |
| SPW5\_115T20 - Rainy | |
| Purpose | Test if the system allows head professor to change the image’s status |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on VM-Images button |
| Input | - User clicks on change image status icon |
| Expected Output | System change image status |

|  |  |
| --- | --- |
| SPW5\_112T21 - Sunny | |
| Purpose | Test if the system allows head professor to set/modify the default email for vm creation |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and to Set Default Email for VM Creation form |
| Input | - Valid email address  - User clicks on Set Default Email button |
| Expected Output | System updates default email value into database |

|  |  |
| --- | --- |
| SPW5\_112T22 - Rainy | |
| Purpose | Test if the system allows head professor to set/modify the default email for vm creation |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and to Set Default Email for VM Creation form |
| Input | - invalid email address  - User clicks on Set Default Email button |
| Expected Output | System prompts invalid email address error message |

|  |  |
| --- | --- |
| SPW5\_T23 - Sunny | |
| Purpose | Test if the system allows head professor to filter users |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on All Users button |
| Input | - User enters a value in first name filter  - User hits the enter key |
| Expected Output | System prompts all users with filtered name |

|  |  |
| --- | --- |
| SPW5\_T24 - Rainy | |
| Purpose | Test if the system allows head professor to filter users |
| Test Setup | - Head professor is logged in  - Head Professor went to Admin tab and clicked on All Users button |
| Input | - User does not enters a value in first name filter  - User hits the enter key |
| Expected Output | System prompts all users’ name |

* 1. **Evaluation of Tests**

|  |  |  |
| --- | --- | --- |
| **Test** | **Type** | **Passed?** |
| SPW5\_108T1 | Sunny | Passed |
| SPW5\_108T2 | Rainy | Passed |
| SPW5\_109T3 | Sunny | Passed |
| SPW5\_109T4 | Rainy | Passed |
| SPW5\_119T5 | Sunny | Passed |
| SPW5\_119T6 | Rainy | Passed |
| SPW5\_118T7 | Sunny | Passed |
| SPW5\_118T8 | Rainy | Passed |
| SPW5\_110T9 | Sunny | Passed |
| SPW5\_109T10 | Rainy | Passed |
| SPW5\_113T11 | Sunny | Passed |
| SPW5\_113T12 | Rainy | Passed |
| SPW5\_117T13 | Sunny | Passed |
| SPW5\_117T14 | Rainy | Passed |
| SPW5\_116T15 | Sunny | Passed |
| SPW5\_116T16 | Rainy | Passed |
| SPW5\_114T17 | Sunny | Passed |
| SPW5\_114T18 | Rainy | Passed |
| SPW5\_115T19 | Sunny | Passed |
| SPW5\_115T20 | Rainy | Passed |
| SPW5\_112T21 | Sunny | Passed |
| SPW5\_112T22 | Rainy | Passed |

1. Glossary

*Senior Project Website V5*



EULA: End-User License Agreement

FIU: Florida International University

Google Docs: Free web-based office suite offered by Google within Google Drive service.

PHP: Open source server-side scripting language designed for web development to produce dynamic web pages.

SPW: Senior Project Website

SPWv.1: Senior Project Website Version 1

SPWv.2: Senior Project Website Version 2

SPWv.3: Senior Project Website Version 3

SPWv.4: Senior Project Website Version 4

SPWv.5: Senior Project Website Version 5

**NRMP**: A form of matchmaking devised by the national residency matchmaking program (NRMP). The details of it are that hospitals have residency spots to fill and a ranked list of applicants they want filling them. Applicant ranked residency programs themselves. The matchmaking process works by having applicants apply to residencies in their list’s order. If another applicant tries to displace a current tentative applicant they displace the tentative applicant if the challenging applicant is more desired by the program else they try the next program, displaced applicants try their next program too. Matching ends when all applicants are match, or all applicants are match minus ones who went through their entire list.

**Heuristics**: Colloquially means “common sense approach”, in computer science the meaning is adapted to mean a technique applied to solve a problem. I.e. to shorten average job time in a computer do short jobs first.

**VIP**: Very important project, a project ranked by the head professor to be of a score between 2 and 100 this project will undergo intense matchmaking using heuristics to find a perfect team quickly (higher scored means higher priority)

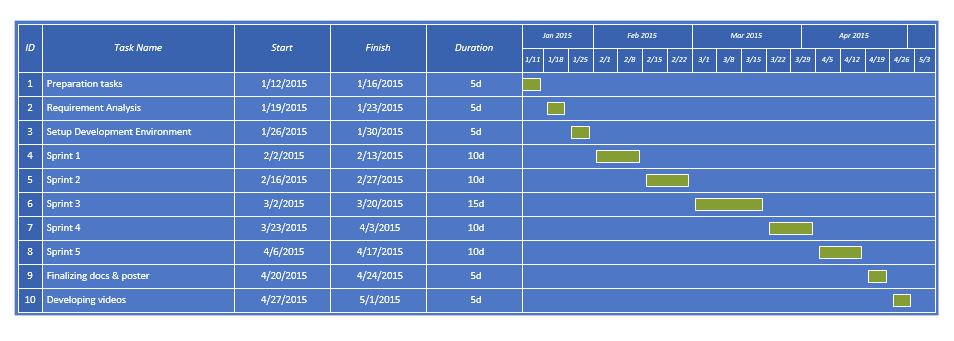
**OP**: Other project, a project ranked by the head professor to have a score of 1, such projects will undergo NRMP matchmaking to give the ability to compromised project proposer and student constraints. These are more hands-off.

**LinkedIn:** Social media website for business professions to network and advertise themselves

*Senior Project Website V5*



1. Appendix
   1. **Appendix A - Gant Chart**

Taken from the given syllabus and adapted to what actually occurred.

* 1. Appendix B - Use Cases

**Below are use cases specific to SPWv.5**

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_100)** | **Edit project assignments** |
| ***Participating Actor*** | Head Professor |
| ***Flow of Events*** | 1. Head Professor clicks on “Add Students to the Project” drop down menu. 2. System displays a list of active students. 3. Head Professor selects one student from the list and clicks on “Save Changes”. 4. Student is added to the project; system displays added student on the page. 5. At the bottom of the page system displays interest and qualification of selected students. |
| ***Alternative Flows*** | 1. Selected project has not enough room for adding another student, and “Add Students to the Project” drop down menu does not appear. 2. Head professor can sort students in the list by their skills and interest. |
| ***Entry Conditions*** | * Head Professor is logged into the site. * Head Professor is in edit project view (accessible by clicking on a project’s name in “Current Projects” tab). * Project has available space (capacity) for students to be assigned. |
| ***Exit Conditions*** | * Head Professor successfully assigned a student to a project. |
| ***Exceptions*** |  |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_101)** | **Change user’s password** |
| ***Participating Actor*** | Head Professor |
| ***Flow of Events*** | 1. Head Professor clicks on “View All Users” button. 2. System navigates to the user management page. 3. Head Professor selects one of the users and clicks on “Act As User” link. 4. System changes logged in user to the selected user, and navigates to the home page of that user. 5. Head Professor clicks on the user’s profile icon located on the menu bar. 6. System displays user’s profile dropdown menu. 7. Head professor clicks on “Head Professor” option located on the dropdown menu. 8. System navigates to user’s profile page. 9. Head Professor clicks “Click to change password” link. 10. System navigates to change password page. 11. Head Professor inserts new password in the top form, then inserts the same password in the bottom form; finally clicks on “Save Changes”. 12. System redirects Head Professor back to user’s profile page, and displays a message about successful change of password. |
| ***Alternative Flows*** | 1. Head Professor does not enter the same password for the two entries provided on the Reset Password page. In this case, System warns Head Professor and allows Head Professor to retry until the same password is entered for both entries. |
| ***Entry Conditions*** | * Head Professor is logged in. * Head Professor has navigated to admin dashboard. |
| ***Exit Conditions*** | * Head Professor successfully resets his/her account password. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_102)** | **Impersonate a user** |
| ***Participating Actor*** | Head Professor |
| ***Flow of Events*** | 1. Head Professor clicks on the Admin tab on the menu bar, and clicks the “View All Users” button on the dashboard. 2. The system displays a list of all the users in the database. 3. Head professor clicks on “Act As User” located under User’s picture. 4. The system changes logged in user to the selected user, and allows the Head Professor to act on behalf of that user. |
| ***Alternative Flows*** | 1. Head Professor selects filtering options, and then submits a query. This results in the system returning only users that match Head Professor’s search criteria. |
| ***Entry Conditions*** | * Head Professor in logged into the site. |
| ***Exit Conditions*** | * Head Professor successfully poses as selected user. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_103)** | **Change password** |
| ***Participating Actor*** | Head Professor |
| ***Flow of Events*** | 1. Head Professor clicks on the user’s profile icon located on the menu bar. 2. System displays user’s profile dropdown menu. 3. Head professor clicks on “Head Professor” option located on the dropdown menu. 4. System navigates Head Professor to user’s profile page. 5. Head Professor clicks on “Click to change your password”. 6. System navigates to change password page. 7. Head professor inserts current password and new password, and then clicks “Save Changes”. 8. System navigates back to user’s profile page, and displays a message about successfully changed password. |
| ***Alternative Flows*** | 1. Head Professor does not enter correct current password, which causes the system to prompt Head Professor for correct current password. |
| ***Entry Conditions*** | * Head Professor in logged into the site. |
| ***Exit Conditions*** | * Head Professor successfully changed profile password. |

|  |  |
| --- | --- |
| ***Entry Conditions*** | * Head Professor is logged in. * Head Professor has navigated to admin dashboard. |
| ***Exit Conditions*** | * Head Professor successfully changed profile password. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_104)** | **Request change of password** |
| ***Participating Actor*** | User |
| ***Flow of Events*** | 1. User clicks on the “Forgot Password?” link. 2. System displays a dialog prompt, containing a form asking for user’s email address. 3. User enters email address associated with his/her account, and clicks on “Send” button. 4. System verifies that the inserted email address exists in the system, and the user is active. If so, the system sends a message to the given email address with instructions for changing password. |
| ***Alternative Flows*** | 1. User does not enter correct email, which causes the System to display a message informing about incorrect email address. 2. User enters correct email, but the account associated with that email is inactive; this causes the System to display a message about inactive account. |
| ***Entry Conditions*** | * User has navigated to login page. * User has a manually created account. |
| ***Exit Conditions*** | * User successfully requested change of profile password. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_105)** | **Change forgotten password** |
| ***Participating Actor*** | User |
| ***Flow of Events*** | 1. User clicks on the link provided in the message from the System. 2. System navigates to the Reset Password page. 3. User inserts new password in the top form, then inserts the same password in the bottom form; finally clicks on “Save Changes”. 4. System checks that both passwords entered are the same. If so, System changes User’s password and redirects to login page; then displays a message informing about successful change of password. |
| ***Alternative Flows*** | 1. User does not enter the same password for the two entries provided on the Reset Password page. In this case, System warns User and allows User to retry until the same password is entered for both entries. |
| ***Entry Conditions*** | * User successfully used forgot password feature to request password change. * User has received the message from the system on his/her email account. |
| ***Exit Conditions*** | * User successfully changed profile password. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_106)** | **Change profile picture** |
| ***Participating Actor*** | Student |
| ***Flow of Events*** | 1. Student clicks on the user’s profile icon located on the menu bar. 2. The system displays user’s profile dropdown menu. 3. Student clicks on his/her name located on the dropdown menu. 4. The system navigates to user’s profile page. 5. Student clicks on the “Choose File” button. 6. System opens file manager window for the user to choose a file to upload. 7. Student selects image file, and then clicks on “Open” in file manager window. 8. System displays dialog prompt asking if Student wants to replace current picture. 9. Student clicks on “Yes”. 10. System displays a message about successful upload. |
| ***Alternative Flows*** | 1. Student clicks on “No” on the dialog prompt, which causes the system to abort the upload. |
| ***Entry Conditions*** | * Student is logged into the system. * Student already has a picture on his/her profile. |
| ***Exit Conditions*** | * Student successfully changed profile picture. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_107)** | **Select all users** |
| ***Participating Actor*** | Head Professor |
| ***Flow of Events*** | 1. Head Professor clicks on “View All Users” button. 2. System navigates to the user management page. 3. Head Professor clicks on “Select All” checkbox. 4. System checks all checkboxes associated with users. |
| ***Alternative Flows*** |  |
| ***Entry Conditions*** | * Head Professor is logged in. * Head Professor has navigated to admin dashboard. |
| ***Exit Conditions*** | * Head Professor successfully selects all users. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_108)** | **Request Virtual Machine** |
| * ***User Story* #148** | * As a student, I want to request a number of virtual machine resources to the head professor, so I have the opportunity to customize my virtual environment with the exact number of virtual machines, the image of preference, the amount of RAM and the storage amount needed for my project. |
| ***Participating Actor*** | Student, email system |
| ***Entry Conditions*** | * Student has logged into Senior Project Website. * Current date is after deadline to choose a project. * Student is in project\_details2 page. |
| ***Flow of Events*** | 1. Student clicks on Create VM-Request button in project\_details2 page. 2. System navigates to VM – Request page. 3. Student creates a VM request by entering in input fields: the image, the amount of RAM, the storage and the number of VMs. 4. System displays in input fields: the input image names, the amount of RAM, the Storage amount and the number of virtual machines. 5. Student clicks on “Submit” button. 6. System stores virtual machine request in database, and notifies Head Professor via email about request. |
| ***Alternative Flows*** | * {From step 3} Student selects virtual machine settings but does not click on “Submit” button and navigates away to a different page. Then, system considers virtual machine request invalid and destroyed. |
| ***Exit Conditions*** | * Student successfully submitted virtual machine request. |
| ***Exceptions*** | * If current day is under deadline and student attempts to access/hack into VM – Request page, system denies student’s access to VM-Request page and prompts warning message. * If student inputs in fields amount of RAM, storage amount and number of virtual machines a non-numerical value. System prompts error message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_109)** | **Modify Virtual Machine Request** |
| ***Participating Actor*** | Head professor, email system |
| * ***User Story* #146** | As a head professor, I want to check the virtual machine settings requested by students, so I can change the image name, amount of RAM, storage amount and the number of virtual machines if needed. |
| ***Entry Conditions*** | * Head professor is logged into the system. * Head professor is in vm\_requests2 page. |
| ***Flow of Events*** | 1. Head professor modifies a virtual machine request by selecting from drop down menus: the image name, the status of the request and entering in input fields the amount of memory RAM, the storage amount and the number of VMs. 2. System displays the selected image name, selected amount of RAM, selected storage amount, selected number of VMs and the status. 3. Head professor clicks on “Submit” button. 4. System updates virtual machine settings into database and notifies school’s system admin if the status of the request is approved. |
| ***Alternative Flows*** | * {From step 3} Head professor modifies virtual machine request’s settings but does not click on submit button and navigates into a different page. System considers request invalid and it is destroyed. |
| ***Exit Conditions*** | * Head professor successfully modified virtual machine request. |
| ***Exceptions*** | * If email notification input field is empty, system alerts with an error message. * If an invalid email address is used, system alerts with error message. * If head professor inputs in fields amount of RAM, storage amount and number of virtual machines a non-numerical value. System prompts error message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_110)** | **Notify School’s System Admin** |
| ***Participating Actor*** | Head professor, email system |
| * ***User Story* #144** | As a head professor, I want to be able to send email to our school’s system admins from the Senior Project Web Site based on the students’ requests for the vms that I have approved, so I don’t need to go into my email account and do it manually. |
| * ***Entry Conditions*** | * Head professor is logged into the system * Head professor is vm\_requests2 page * Head professor has modified virtual machine setting. |
| ***Flow of Events*** | 1. Head professor clicks on drop down menu “Status”. 2. System displays on the drop down menu the available status for the virtual machine request. 3. Head professor selects status equal to “APPROVED”. 4. System displays on the drop down menu the selected status. 5. Head professor enters on the email filed the school’s system admin email address and clicks on “Submit” button. 6. System updates virtual machine status in the database and notifies school’s system admin. |
| ***Alternative Flows*** | * {From step 3} Head professor selects from the drop down menu the “APPROVED” status but does not click on submit button and navigates into a different page. System considers request invalid and it is destroyed. |
| ***Exit Conditions*** | * Head professor successfully notified school’s system admin. |
| ***Exceptions*** | * If email notification input field is empty, system alerts with an error message. * If an invalid email address is used, system alerts with error message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_111)** | **Access Virtual Machine Request** |
| ***Participating Actor*** | Head professor |
| * ***User Story* #145** | As a head professor, I want to receive an email notification when a student requests for one or more vms, so I am able to easily find the request, using a link included in the email notification. |
| * ***Entry Conditions*** | * Student successfully submitted a virtual machine request and notified head professor via email about it. * Head professor is in his fiu.edu email account and in the student’s email notification. * Head professor is logged into the system. |
| ***Flow of Events*** | 1. The use case begins when the head professor clicks on the existing link on the email’s body page. 2. System navigates to VM – Requests page in Senior Project Website and loads all virtual machine request. |
| ***Alternative Flows*** | * {From step 2} Head professor clicks on the link in the email’s body but he is not logged in the system. Then system prompts him with the login page. |
| ***Exit Conditions*** | * Head professor successfully accessed virtual machine request. |
| ***Exceptions*** | * If a user other than head professor attempts to access/hack into VM – Requests page, system denies user’s access to VM-Requests page and prompts warning message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_112)** | Set Default Email Notification |
| ***Participating Actor*** | Head professor |
| * ***User Story* #145** | As a head professor I want to be able to choose a default person’s email to notify him about vm creation, so I don’t have to memorize his email, neither input it every time manually. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in admin\_dashboard page. |
| ***Flow of Events*** | 1. Head professors enters school’s system admin name in the input field named “Full Name”. 2. System shows inputted name. 3. Head professor enters school’s system admin name in the input filed named email. 4. System shows inputted email address. 5. Head professor clicks on “Set Default Email” button. 6. System validates email address and process request. 7. Head professor sees default name and email address. |
| ***Alternative Flows*** | * {From step 3} Head professor enters name and default email address of school’s system admin, but does not click on “Set Default Email” button and navigates into a different page. System considers request invalid and it is destroyed. |
| ***Exit Conditions*** | * Head professor successfully set/changed default name and email address. |
| ***Exceptions*** | * If head professor leaves “Full Name” filed empty and clicks on “Set Default Email” button, system shows error message. * If head professor leaves “Email” field empty and clicks on “Set Default Email” empty, system shows error message. * If head professor enters an invalid email address, system shows error message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_113)** | Add Virtual Machine Image |
| ***Participating Actor*** | Head professor |
| * ***User Story* #218** | As a head professor I want to be able to add new available virtual machine images in the system, so that students can choose the newly added images in their virtual machines requests. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in admin\_dashboard page. |
| ***Flow of Events*** | 1. Head professor clicks on “VM Images” button. 2. System navigates to vm\_images page. 3. Head professor enters the new image in the input field named “Image Name”. 4. System shows inputted image name. 5. Head professor clicks on “Add Image Name” button. 6. System processes request and prompts success message. |
| ***Alternative Flows*** | * {From step 3} Head professor enters image name, but does not click on “Add Image Name” button and navigates into a different page. System consider action invalid and it is destroyed. |
| ***Exit Conditions*** | * Head professor successfully added a new virtual machine image. |
| ***Exceptions*** | * If head professor leaves “Image Name” filed empty and clicks on “Add Image Name” button, system displays an error message. * If head professor enters an image name that already exists on the system, system displays an error message. * If a user other than head professor attempts to access/hack into image name page, system denies user’s access to such a page and prompts warning message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_114)** | Filter Images |
| ***Participating Actor*** | Head professor |
| * ***User Story* #355** | As a head professor I want to be able to filter the available image’s name in the system, so I have the option to easily and more efficiently search through the existing images in the system. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_images page. |
| ***Flow of Events*** | 1. This use case starts when the head professor selects from a drop down menu one of the available options “All Images”, “ACTIVE”, “INACTIVE” and/or input the full or partial name of an image. 2. System displays the selected filter option. 3. Head professor hits enter key. 4. System processes request and prompts the images according to the selected filter option. |
| ***Alternative Flows*** | * {From step 1} Head professor enters on the input field the full or partial name of an image and hits the enter key. System processes request and prompts all images that matched the search criteria. |
| ***Exit Conditions*** | * Head professor successfully filtered images. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_115)** | Set Image Status |
| ***Participating Actor*** | Head professor |
| * ***User Story* #355** | As a head professor I want to be able to set the status of the images in the system, so just active images are offered to students. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_images page. |
| ***Flow of Events*** | 1. This use case starts when the head professor clicks on an existing semaphore link/button in vm\_images page. 2. System collects image’s information, processes change status request and prompts success message. |
| ***Exit Conditions*** | * Head professor successfully changed images status. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_116)** | Delete Image |
| ***Participating Actor*** | Head professor |
| * ***User Story* #355** | As a head professor I want to be able to delete an image from the system, so images that are no longer in used can be removed. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_images page. |
| ***Flow of Events*** | 1. This use case starts when the head professor clicks on an existing “delete” link/button in vm\_images page. 2. System prompts confirmation message for deletion. 3. Head professor clicks on “okay” button. 4. System collects image’s information, processes delete request and prompts success message. |
| ***Alternative Flows*** | * {From step 3} Head professor clicks on “Cancel” button. System considers request invalid and cancel it. |
| ***Exit Conditions*** | * Head professor successfully deleted an image. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_117)** | Edit Image |
| ***Participating Actor*** | Head professor |
| * ***User Story* #355** | As a head professor I want to be able to edit an image from the system, so I can upgrade existing images from the system or correct the ones that already exist if need be. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_images page. |
| ***Flow of Events*** | 1. This use case starts when the head professor enters in the input field a new image name. 2. System displays the information on the input field. 3. Head professor clicks on “Submit” button. 4. System collects image’s information, processes request and prompts success message. |
| ***Alternative Flows*** | * {From step 3} Head professor enters image name, but does not click on “submit” button and navigates into a different page. System consider action invalid and it is destroyed. |
| ***Exit Conditions*** | * Head professor successfully edited an image. |
| ***Exceptions*** | * If head professor clicks on “Submit” button and input filed is empty. System alerts with an error message. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_118)** | Filter Virtual Machine Requests |
| ***Participating Actor*** | Head professor |
| * ***User Story* #355** | As a head professor I want to be able to filter all virtual machine requests on the system, so I can easily access the information that is of interest to me. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_requests2 page. |
| ***Flow of Events*** | 1. This use case starts when the head professor selects one or more of the available options to filter virtual machine requests, by status, amount of memory RAM, storage, number of VMs and/or inputs the full or partial name of a student and a term. 2. System displays the selected filter option. 3. Head professor hits the enter key. 4. System processes request and prompts virtual machine requests according to the selected filter options. |
| ***Alternative Flows*** | * {From step 1} Head professor enters on the input fields the full or partial name of an image, student name, term and hits the enter key. System processes request and prompts all images that matched the search criteria. |
| ***Exit Conditions*** | * Head professor successfully filtered virtual machine requests. |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW5\_119)** | Delete Virtual Machine Request |
| ***Participating Actor*** | Head professor |
| * ***User Story* #412** | As a head professor I want to be able to delete a virtual machine request from the system, so virtual machine requests that are no longer needed can be removed. |
| * ***Entry Conditions*** | * Head professor is logged in the system. * Head professor is in vm\_requests2 page. |
| ***Flow of Events*** | 1. This use case starts when the head professor clicks on an existing “delete” link/button in vm\_requests2 page. 2. System prompts confirmation message for deletion. 3. Head professor clicks on “okay” button. 4. System collects virtual machine request’s information, processes delete request and prompts success message. |
| ***Alternative Flows*** | * {From step 3} Head professor clicks on “Cancel” button. System considers request invalid and cancel it. |
| ***Exit Conditions*** | * Head professor successfully deleted a virtual machine request. |

**Below are use cases specific to SPWv.4**

|  |  |
| --- | --- |
| ***Use Case Name* (SPW4\_100)** | **Run Match** |
| ***Participating Actor*** | Head Professor user |
| ***Flow of Events*** | 1. Head professor clicks to run VIPs automatically or manually. If automatically chosen goes to alternative flow 1 2. If manual one by one head professor chooses up to “project\_max” students per project between students who want the project and students forced into project. 3. After choosing head professor clicks next for manual or when they click “run rest automatically” goes to Alternative flow 1 with remaining VIPs 4. Page showing how VIP matches will be at the end, head professor clicks button to proceed to OP matchmaking 5. Head professor choose to accept “friendly” student interest decides how matchmaking occurs totally for his not-as-important projects or chooses “compromise” where students chooses projects they want to be displaced only by better skilled student 6. Professor clicks one or other and chooses next. 7. Head professor sees all matching metadata for VIP and OP and chooses to finalize, he is asked to confirm and does so. 8. Match data is prorogated to database and students are matched |
| ***Alternative Flows*** | Alternative flow 1: If automatically chosen or “run rest automatically” chosen (From 1 or 3 respectively). Then head professor chooses between a matching of students who wants all the remaining VIPs and a matching where some students may be forced into VIPs. Two choices in other words with how the rest (or all) of the VIPs are matched. After choosing one or other goes to 4 |
| ***Entry Conditions*** | * Head professor is logged in * Head professor went to match then clicked run match |
| ***Exit Conditions*** | * Head professor successfully saved match result * Head professor left at any point for w/e reason nothing changes |
| ***Exceptions*** | * If head professor tries to pass not “project\_max” number of students in 2. He will be asked to confirm |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW4\_101)** | **Save Head Professor Ranks** |
| ***Participating Actor*** | Head professor |
| ***Flow of Events*** | 1. Head professor see list of project and text field beside them 2. He ranks projects either 0, 1 or 2-100 based on if they’ll be not considered, OP, or VIP 3. He saves his ranks and is redirect to match main page |
| ***Entry Conditions*** | * Head professor is logged in went to match then clicked “Go to project priority” |
| ***Exit Conditions*** | * He successfully saves his rankings or is told of problems |
| ***Exceptions*** | * Head professor puts number out or range, not integer and is told error message |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW4\_102)** | **Save Student Ranks** |
| ***Participating Actor*** | Student |
| ***Flow of Events*** | 1. Student sees projects and ranks them between -1 and 100, if less than 1 they don’t want it. By default all is -1. 2. Student clicks saves ranks |
| ***Entry Conditions*** | * Student is logged in and in the main page |
| ***Exit Conditions*** | * He successfully saves his rankings or is told of problems; if rank minimum not save those ranks are saved but until rank minimum is ranked then he will still be considered for all projects |
| ***Exceptions*** | * Student puts number out or range, not integer , or has not ranked a minimum number of projects and is told error message |

**Below are used cases from other versions**

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_101)** | **FIU Panther EMail Login** |
| ***Related Use Cases*** | * API Validation (SPW2\_105) |
| ***Participating Actor*** | Student User |
| ***Flow of Events*** | 1. User clicks “ FIU Panther Email” button 2. Server redirects to FIU login page 3. User enter its FIU credentials 4. Submit information and Consent 5. Google Authorization Server redirects user back to our website with a Google Code 6. Server send this Google Code to the Google OAuth2 Authorization Server in order to retrieve a Google Token 7. Google OAuth2 Authorization Server responds with a Token(Access Token) 8. Server call Google API with the Google Token requesting Google ID for this user, first name, last name and email 9. Server validate the student against API 10. API responds to validation 11. Server logs and redirects student to his Homepage |
| ***Entry Conditions*** | * User is in Login Page |
| ***Exit Conditions*** | * User login successfully * User is automatically logged in |
| ***Exceptions*** | * FIU panther email does not allow login * User is not enrolled in Senior Project class |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_102)** | **SPW2 Guest Access** |
| ***Participating Actor*** | Guest user |
| ***Flow of Events*** | 1. Guest clicks in Login as a Guest 2. Server redirects user to the Homepage |
| ***Entry Conditions*** | * User is in Login page |
| ***Exit Conditions*** | * User access successfully to the website as a guest |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_103)** | **SPW2 Professor Login** |
| ***Related Use Cases*** | * Password encryption/decryption (SPW2\_110) |
| ***Participating Actor*** | Professor user |
| ***Flow of Events*** | 1. User enters his credentials and submit 2. Client/Server validation of provided information 3. Server queries the Database to validate credentials 4. Database responds to the query 5. Server redirects user to the Home Page |
| ***Entry Conditions*** | * User is in Login page |
| ***Exit Conditions*** | * User access successfully to the website using his previous created profile |
| ***Exceptions*** | * Missing username or password * Invalid credentials |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_104)** | **SPW2 Log Out** |
| ***Participating Actor*** | Student user , Professor user |
| ***Flow of Events*** | 1. User clicks in Logout button |
| ***Entry Conditions*** | * User is logged into the system |
| ***Exit Conditions*** | * Server redirects user to the login page |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_105)** | **API Validation** |
| ***Participating Actor*** | SPW , Virtual Job Fair external systems |
| ***Flow of Events*** | 1. The external systems makes a call to the API passing and email of a student and a token  2. The API validates the token  3. API validates if the student is on the file.  4. API returns a JSON string with the user info and a the property “valid” = true . |
| ***Alternate Flow of Events*** | If at step 3 the student is not on the file then the API returns a JSON object with the property “valid” = false |
| ***Entry Conditions*** | * The API is running * The external systems have a valid token |
| ***Exit Conditions*** | * JSON object is returned |
| ***Exceptions*** | Token provided is not valid |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_106)** | **API Refresh Data** |
| ***Related use case*** | **API Create new students (SPW2\_107)**  **API Remove inactive students(SPW2\_108)**  **API Delete pending projects for inactive students (SPW2\_109)** |
| ***Participating Actors*** | SPW external system |
| ***Flow of Events*** | 1. Head Professors clicks on the “refresh from API” button on the admin dashboard 2. SPW sends a request to the API to refresh the data 3. API Creates new students 4. API Removes inactive students 5. API Deletes pending projects for inactive students |
| ***Entry Conditions*** | * Head professor is on admin dashboard * API is running on Tomcat server |
| ***Exit Conditions*** | * SPW database has the latest info updated. |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_107)** | **API Create new students** |
| ***Participating Actors*** | SPW external system |
| ***Flow of Events*** | 1. API verifies the file with the students list. 2. API queries database for students that are not on the database 3. Database responds 4. API creates new students on database 5. Database responds |
| ***Entry Conditions*** | * Request to refresh data from API has been made by an external system. |
| ***Exit Conditions*** | * SPW database has the latest info updated. |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_108)** | **API Remove inactive students** |
| ***Participating Actors*** | SPW external system |
| ***Flow of Events*** | 1.API queries database for inactive students  2.Database responds.  3.API deletes students on database.  4.Database responds |
| ***Entry Conditions*** | * Request to refresh data from API has been made by an external system. |
| ***Exit Conditions*** | * SPW database has the latest info updated. |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_109)** | **API Delete pending projects for inactive students** |
| ***Participating Actors*** | SPW external system |
| ***Flow of Events*** | 1.API queries database for projects that have a status PENDING and the student who created it has been deleted because it is inactive  2.Database responds.  3.API deletes the projects.  4.Database responds |
| ***Entry Conditions*** | * Request to refresh data from API has been made by an external system. |
| ***Exit Conditions*** | * SPW database has the latest info updated. |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_110)** | **API Get all students' project Title** |
| ***Participating Actors*** | Mobile Judge external system |
| ***Flow of Events*** | 1. External system requests the API the list of students on the class and their project title.  2.API reads the file for the list of students.  3.API queries SPW database to get the project title of each student on the file.  4.Database responds |
| ***Entry Conditions*** | * API is running on the server. |
| ***Exit Conditions*** | * API returns a JSON object with the list of all students and the project title. |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_201)** | **Edit User Profile** |
| ***Participating Actors*** | Registered user |
| ***Flow of Events*** | 1. User clicks the “My Profile” link on the navigation bar on top 2. Server requests the user information to the database 3. Database responds 4. Server displays the user information page 5. User edits the displayed information 6. User clicks the “Save Changes” button on the bottom 7. Server requests the user change to the database 8. Database responds |
| ***Entry Conditions*** | * Registered user is logged into the system * Any page with the navigation bar is show |
| ***Exit Conditions*** | * User profile updated message shown |
| ***Exceptions*** | * No information was changed |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_202)** | **Update Profile With LinkedIn** |
| ***Participating Actor*** | Registered User |
| ***Flow of Events*** | 1. User clicks “Sync with LinkedIn” button 2. Server redirects to LinkedIn Log In page 3. User enter its LinkedIn credentials 4. Submit information and Consent 5. LinkedIn Authorization Server redirects user back to our website with a LinkedIn 6. Server send this LinkedIn Code to the LinkedIn OAuth Authorization Server in order to retrieve a LinkedIn Token 7. LinkedIn OAuth Authorization Server responds with a Token(Access Token) 8. Server calls LinkedIn API with the LinkedIn Token requesting for his user: LinkedIn ID, First name, Last name, e-mail, Biography, Skills, Languages and Experience. 9. Server request database to update skills for user 10. Server request database to update languages for user 11. Server request database to update experience for user 12. Server request database to update Biography for user 13. Server request database to update picture url for user 14. Server request database to update first name and last name for user 15. Server returns the updated profile to the user |
| ***Alternative flows*** | * In step 13 if user profile has picture url then:  1. Do not update it  * In step 14 if user profile has first name and last name then:  1. Do not update it |
| ***Entry Conditions*** | * User is in his Profile page * LinkedIn accessible |
| ***Exit Conditions*** | * User profile updated with the information pulled from LinkedIn |
| ***Exceptions*** | * Invalid LinkedIn credentials * Cancel Sign in LinkedIn |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_301)** | **Browse User’s profiles** |
| ***Participating Actors*** | User |
| ***Flow of Events*** | 1. User clicks in on a user name link 2. Server requests the user information to the database 3. Database responds 4. Server displays the past projects summary information |
| ***Entry Conditions*** | * Registered user is logged into the system. |
| ***Exit Conditions*** | * User summary information displayed |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_302)** | **Browse Open Projects** |
| ***Participating Actors*** | User |
| ***Flow of Events*** | 1. User clicks in “Open Projects” link located in top menu of the page 2. Server requests the past projects information to the database 3. Database responds 4. Server displays the past |
| ***Entry Conditions*** | * Registered user is logged into the system. * Any page with the navigation bar is show |
| ***Exit Conditions*** | * Project summary information displayed |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_303)** | **Browse Past Projects** |
| ***Participating Actors*** | User |
| ***Flow of Events*** | 1. User clicks in “Past Projects” link located in top menu of the page 2. Server requests the past projects information to the database 3. Database responds 4. Server displays the past projects summary information |
| ***Entry Conditions*** | * Registered user is logged into the system. * Any page with the navigation bar is show |
| ***Exit Conditions*** | * Past projects summary information displayed |
| ***Exceptions*** | * No matching found |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_304)** | **Browse Users and Projects by keyword** |
| ***Participating Actors*** | User |
| ***Flow of Events*** | 1. User enters a keyword in the search box to look for a specific User or Project 2. Search for keywords in the database 3. Database responds 4. Server displays the matching information |
| ***Entry Conditions*** | * Registered user is logged into the system. * Any page with the navigation bar is show |
| ***Exit Conditions*** | * Matching information is displayed |
| ***Exceptions*** | * No matching found |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_401)** | **Create Project** |
| ***Participating Actor*** | Registered User |
| ***Flow of events*** | 1. User clicks “Create New Project” button on “My Projects” page 2. User enter all the information required to create a new project 3. Server send a request to database to insert a new project 4. Database acknowledges this new data 5. If user is student a notification for approval of the project is sent to the professor in charge of the Senior Project class 6. Server returns the My Projects page with all the information for the new created project. |
| ***Entry Conditions*** | * User is in his My Projects page * The term deadline has not passed |
| ***Exit Conditions*** | * Project created * The projects table on the database in increased by 1 * If user is student the project created is pending approval by the professor, otherwise it is approved * If user is Professor he is the Mentor of the Project |
| ***Exceptions*** | * User leaves a text field empty * The Database was not able to store the document |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_402)** | **Join Project** |
| ***Participating Actor*** | Student user |
| ***Flow of Events*** | 1. Student clicks the “Join” button showed on the project details summary from the project he wants to join 2. Server send a request to database to insert a new team member to the project 3. Database acknowledges this new data 4. The System sends a notification to all the team members 5. Student is added as a team member to the project he requested to join |
| ***Entry Conditions*** | * User is in his homepage where projects are listed * List of projects from as a result of user search are shown * The term deadline has not passed |
| ***Exit Conditions*** | * User is added to the project * Project’s team members is increased by 1 |
| ***Exceptions*** | * User is already a team member of a project * Project student’s maximum capacity is full |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_403)** | **Leave Project** |
| ***Participating Actors*** | Student user |
| ***Flow of Events*** | 1. Student clicks the “Leave” button showed on the project details summary from the project he wants to leave 2. Server send a request to database to remove the user from the project 3. Database acknowledges the request 4. The System sends a notification to all the team members 5. System displays a confirmation flash message |
| ***Entry Conditions*** | * Student user is logged into the system * User is a team member of the project |
| ***Exit Conditions*** | * Student is no longer team member of the project * Team members of the project is decreased by one |
| ***Exceptions*** | * The proposer of a project cannot leave the project * The term deadline has passed |

|  |  |
| --- | --- |
| ***Use case Name* (SPW\_404)** | **Assign student to any project** |
| ***Participating Actor*** | Head Professor user |
| ***Flow of Events*** | 1. Head Professor opens the “Add Students to the Project” dropdown menu button in one of the projects 2. Head Professor chooses from a list of students without an assigned project 3. Head Professor clicks on the “Save Changes” button 4. Server send a request to the database to insert a new team member to the project 5. Database acknowledges this new data 6. The System sends a notification to the user added 7. Server displays updated notification message |
| ***Entry Conditions*** | * Head Professor is logged into the system * A project page is displayed * The System previously filled the drop down menu with a list of students without a project |
| ***Exit Conditions*** | * User is added to the project * Project’s team members is increased by 1 |
| ***Exceptions*** | * The Database was not able to store the settings |

|  |  |
| --- | --- |
| ***Use case Name* (SPW\_405)** | **Assign student to a project** |
| ***Participating Actor*** | Professor user |
| ***Flow of Events*** | 1. Head Professor opens the “Add Students to the Project” dropdown menu button in one of the projects 2. Head Professor chooses from a list of students without an assigned project 3. Head Professor clicks on the “Save Changes” button 4. Server send a request to the database to insert a new team member to the project 5. Database acknowledges this new data 6. The System sends a notification to the user added 7. Server displays updated notification message |
| ***Entry Conditions*** | * Professor is logged into the system * A project page is displayed * Professor created the project * The System previously filled the drop down menu with a list of students without a project |
| ***Exit Conditions*** | * User is added to the project * Project’s team members is increased by 1 |
| ***Exceptions*** | * The Database was not able to store the settings |

|  |  |
| --- | --- |
| ***Use case Name* (SPW\_406)** | **Remove student from any project** |
| ***Participating Actor*** | Head Professor user |
| ***Flow of Events*** | 1. Head Professor clicks the “Remove” button below the user he wished to remove 2. The Systems displays a confirmation message 3. The User clicks Yes 4. Server send a request to database to remove the user from the project 5. Database acknowledges the request 6. The System sends a notification to the user removed 7. Server displays flash notification message |
| ***Entry Conditions*** | * Professor is logged into the system * A project page is displayed |
| ***Exit Conditions*** | * User is removed from the project * Project’s team members is decreased by 1 |
| ***Exceptions*** | * The Database was not able to delete the student |

|  |  |
| --- | --- |
| ***Use case Name* (SPW\_407)** | **Remove student from a project** |
| ***Participating Actor*** | Professor user |
| ***Flow of Events*** | 1. Professor clicks the “Remove” button below the user he wished to remove 2. The Systems displays a confirmation message 3. The User clicks Yes 4. Server send a request to database to remove the user from the project 5. Database acknowledges the request 6. The System sends a notification to the user removed 7. Server displays flash notification message |
| ***Entry Conditions*** | * Professor is logged into the system * A project page is displayed * Professor created the project |
| ***Exit Conditions*** | * User is removed from the project * Project’s team members is decreased by 1 |
| ***Exceptions*** | * The Database was not able to delete the student |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_408)** | **Delete Proposed Project** |
| ***Participating Actors*** | Student user, Professor user |
| ***Flow of Events*** | 1. User clicks the “Delete Project” button 2. The Systems displays a confirmation message 3. The User clicks Yes 4. Server requests the user changes to the database 5. Database responds 6. The System notifies team members 7. The system removes all the team members from the project |
| ***Entry Conditions*** | * User is logged into the system * A project created by the user is displayed * If user is Student the project is still pending for approval or rejected |
| ***Exit Conditions*** | * Project was successfully deleted * The projects table on the database is decreased by 1 * User is redirected to his Homepage |
| ***Exceptions*** | * The database was not able to delete the project * The database was not able to remove the team members from the project |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_409)** | **Delete Any Project** |
| ***Participating Actors*** | Head Professor user |
| ***Flow of Events*** | 1. User clicks the “Delete Project” button 2. The Systems displays a confirmation message 3. The User clicks Yes 4. Server requests the user changes to the database 5. Database responds 6. The System notifies team members 7. The system removes all the team members from the project |
| ***Entry Conditions*** | * Head Professor user is logged into the system * A project is displayed |
| ***Exit Conditions*** | * Project was successfully deleted * The projects table on the database is decreased by 1 * User is redirected to his Homepage |
| ***Exceptions*** | * The database was not able to delete the project * The database was not able to remove the team members from the project |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_410)** | **Change Project Status** |
| ***Participating Actor*** | Head Professor user |
| ***Flow of Events*** | 1. Head Professor opens the status drop down menu 2. Head Professor chooses a new status 3. Head Professor clicks on the “Save Changes” button 4. Server sends updated status to the database 5. Database successfully responds 6. Server displays updated project page |
| ***Entry Conditions*** | * Head Professor is logged into the system * Project page is displayed |
| ***Exit Conditions*** | * Project’s new status is store in the database |
| ***Exceptions*** | * The database was not able to store the new Status |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_411)** | **Add a Mentor to a Project** |
| ***Participating Actor*** | Head Professor user |
| ***Flow of Events*** | 1. Head Professor starts typing a professor’s name into the autocomplete “Add a Mentor” form 2. The System sends a request to the database to retrieve User’s names similarities. 3. The database successfully responds. 4. The System displays a list of possible matches. 5. The Head Professor clicks on a name from the list. 6. The Head Professor clicks on the “Save Changes” button. 7. The System sends a request to the database to store the new settings. 8. The database responds successfully. 9. The System sends a notification to the new Member 10. Server displays updated project page |
| ***Entry Conditions*** | * Head Professor is logged into the system * Project page is displayed |
| ***Exit Conditions*** | * The Professor User is added as a Mentor to the Project. |
| ***Exceptions*** | * The database was not able to store the new Mentor |

|  |  |
| --- | --- |
| ***Use Case Name* (SPW2\_412)** | **Edit Project** |
| ***Participating Actor*** | Registered User |
| ***Flow of Events*** | 1. User types the new information 2. User clicks on the “Save Changes” button 3. The System sends a request to the database to update the project. 4. Database returns a successful response 5. The System displays the updated project. |
| ***Entry Conditions*** | * User is logged into the system * Project page is displayed |
| ***Exit Conditions*** | * Project is updated in the database |
| ***Exceptions*** | * The database was not able to update the Project |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Use Case Name* (SPW2\_501)** | | **Activate/Deactivate User** | |
| ***Participating Actor*** | | Head Professor user | |
| ***Flow of Events*** | | 1. In the section titled: "Activate/Deactivate Users", the Head Professor ticks the checkboxes next to all the accounts he/she wants to activate or deactivate. 2. The Head Professor selects option to apply as: “Activate the selected user(s)” or “Deactivate the selected user(s)”. Note: user can only choose one since these are radio buttons. 3. User clicks the “Execute Changes” button. | |
| ***Alternative Flows*** | | N/A | |
| ***Entry Conditions*** | | * Head Professor user is logged into the system. * The admin dashboard page is ready (this is accessible from the navigation bar). | |
| ***Exit Conditions*** | | * Head Professor is redirected to the admin dashboard. * The system displays a message notifying the user that the selected accounts have been activated or deactivated depending on the action selected. Additionally, the status of those accounts should now be changed to display the current statuses. Active users should be highlighted in green. Inactive users should be highlighted in red. | |
| ***Exceptions*** | | * No user accounts were selected before the form was submitted. A notification should be displayed informing the user of this event. | |
| ***Use case Name* (SPW2\_502)** | | **Create Professor User** | |
| ***Participating Actors*** | | Head Professor user | |
| ***Flow of Events*** | | 1. In the section titled: "Create a New Professor User", the Head Professor clicks the input field with the placeholder: “email@example.com” and enters the email of the new professor user. 2. The Head Professor clicks on the input field with the placeholder: “Password” and enters a temporary password for the new professor account. 3. The Head Professor clicks on the input field with the placeholder: “Confirm Password” and re-enters the temporary password. 4. The Head Professor clicks the “Create Professor” button. | |
| ***Entry Conditions*** | | * Head Professor user is logged into the system * The admin dashboard page is ready (this is accessible from the navigation bar). | |
| ***Exit Conditions*** | | * Head Professor is redirected to the admin dashboard. * The system displays a message notifying the user that the new professor account was created. | |
| ***Exceptions*** | | * The email entered is already in use by some other user in the system. * The temporary passwords do not match. * The temporary passwords are too short (min is 6 characters). | |
|  | |  | |
| ***Use case Name* (SPW2\_503)** | **Set join/leave/propose project deadline** | |
| ***Participating Actors*** | Head Professor user | |
| ***Flow of Events*** | 1. In the section titled: "Set Deadline for Students to Choose a Project", the Head Professor makes selections using the Calendar pickers that correspond to both the "Start Date" and "End Date" form fields. 2. The Head Professor submits the form by clicking the "Set Deadline" button. 3. The system displays a message notifying the Head Professor that the deadline has been successfully updated. | |
| ***Entry Conditions*** | * Head Professor user is logged into the system * The admin dashboard page is ready (this is accessible from the navigation bar). | |
| ***Exit Conditions*** | * Head Professor is redirected to the admin dashboard. * The new deadline has been set and stored in the database. | |
| ***Exceptions*** | * Errors in the data entry:   + A field cannot be parsed as a Date according to the format: *mm/dd/yyyy.*   + One or both of the required fields is empty.   + The start date is greater than or equal to the end date (not a valid time window to denote the deadline). | |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Manually Add User |
| Related Use Cases | Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor enters a new user’s name, email address, and role |
| 2.      The Head Professor uses the “Create New User” button |
| 3.      The system sends an activation email to the new user’s email address |
| 4. The system saves the new user in the DB as a “PENDING” user |
| 5. The system alerts the Head Professor that the new user was created |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click the "Manually Add User" accordion menu. |
|  |  |
| Exit Conditions | 1. The system displays confirmation of newly added user 2. An activation email is sent to the newly added user |
| Exceptions | 1. A user was already created and the system alerts the Head Professor that this user already exists |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Modify User |
| Related Use Cases | Filter Users, Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” button on his dashboard. |
| 3.      The system displays a list of all the users in the database |
| 4. The Head Professor selects the filters that he wants to use to search for a user and submits his query |
| 5. The system returns only the users that match the Head Professor’s search results |
| 6. The Head Professor finds the name of the user he wishes to modify |
| 7. The Head Professor has the ability to change the user’s status or edit the user’s information |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click View All Users button at the top right |
|  |  |
| Exit Conditions | 1.      The system displays the updated user information |
| Exceptions |  |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Delete User |
| Related Use Cases | Filter Users, Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” button on his dashboard. |
| 3.      The system displays a list of all the users in the database |
| 4. The Head Professor selects the filters that he wants to use to search for a user and submits his query |
| 5. The system returns only the users that match the Head Professor’s search results |
| 6. The Head Professor finds the name of the user he wishes to modify |
| 7. The Head Professor clicks on the “Delete User” button and confirms that he wants to delete the user from the database |
| 8.      The system deletes the user from the database |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click View All Users button at the top right |
|  |  |
| Exit Conditions | 1.      The system displays a confirmation of the deleted user |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Filter Users |
| Related Use Cases | Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” button on his dashboard. |
| 3.      The system displays a list of all the users in the database |
| 4. The Head Professor selects the filters that he wants to use to search for a user and submits his query |
| 5. The system returns only the users that match the Head Professor’s search results |
| 6. The Head Professor can repeat this process or choose to execute one of the user operations available on each user’s row |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click View All Users button at the top right |
|  |  |
| Exit Conditions | 1.      The system displays a list of users that match the filters set by the Head Professor |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Bypass Activation |
| Related Use Cases | Filter Users |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” button on his dashboard. |
| 3.      The system displays a list of all the users in the database |
| 4. The Head Professor selects the filters that he wants to use to search for a user and submits his query |
| 5. The system returns only the users that match the Head Professor’s search results |
| 6. The Head Professor finds the name of the user he wishes to activate |
| 7. The Head Professor clicks on the “Bypass Activation” button and confirms that he wants to activate new user |
| 8.      The system generates a password for the new user and sends the new user a welcome message with his new password |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click View All Users button at the top right |
|  |  |
| Exit Conditions | 1.      The system generates a secure password for the new user. |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Generate Password |
| Related Use Cases | Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The use case begins when the user clicks the “Browse...” button in the web application. |
| 2.      The system shall provide a new window explorer to select the file to be uploaded. |
| 3.      The user shall click the "upload" button. |
| 4. The system sends a requests to initiate an upload. |
| 5. The system then selects the project that the user is currently in. |
| 6. The system retrieves the file from the user's computer. |
| 7. The system now uploads the file into FIU's server by adding the file path, the user ID, the project ID, and the category of the file. |
| 8. Once the system has uploaded the file into the server, then it will display the file name. |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select User Management from the dashboard. |
|  | 3. Filter through all active users to find those which are pending.  4. Bypass the activation email for a pending user |
|  |  |
| Exit Conditions | 1.      The system displays a confirmation that the password was generated. |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Impersonate User |
| Related Use Cases | Filter Users, Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” button on his dashboard. |
| 3.      The system displays a list of all the users in the database |
| 4. The Head Professor selects the filters that he wants to use to search for a user and submits his query |
| 5. The system returns only the users that match the Head Professor’s search results |
| 6. The Head Professor finds the name of the user he wishes to activate |
| 7. The Head Professor clicks on the “Act As User” link located beneath the picture of any user |
| 8.      The system changes logged in user to the selected user and allows the Head Professor to act on behalf of that user |
| Entry Conditions | 1.      Login as Head Professor. |
|  | 2. Select Admin tab from menu bar. |
|  | 3. Click View All Users button at the top right |
|  |  |
| Exit Conditions | 1. The system records all given input as input from the selected user, not the Head Professor. 2. The Head Professor selects to return to his own role |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| Use Case Name | (SPW3) Email Activation |
| Related Use Cases |  |
| Participating Actor | Guest User |
| Flow of Events | 1.      User logs into his or her email client |
| 2.      The user clicks on the link in the email that he or she was sent |
| 3.      The user is asked to enter some additional information including a password. |
| 4. The system updates the user’s information in the database by setting his or her status to ACTIVE and recording the user’s new password. |
| 5. The user is redirected to the login page to try and use his or her new dance moves. |
| Entry Conditions | 1.      User checks his or her email |
|  | 2. User clicks on the provided link to provide additional information. |
|  |  |
|  |  |
| Exit Conditions | 1.      The system updates the new user’s information in the database. |
| Exceptions | 1. |
|  |  |

|  |  |
| --- | --- |
| ***Use case Name* (SPW2\_601)** | **Upload profile picture from local storage** |
| ***Participating Actors*** | Registered user |
| ***Flow of Events*** | 1. The user selects on the option to browse/choose a file (This option is located directly below the user's current profile picture). 2. The user chooses a desired file. Supported types are: PNG, JPG, JPEG, and GIF. 3. The user clicks the "Upload Picture" button. 4. The system displays a message notifying the user that their profile picture upload was successful. |
| ***Entry Conditions*** | * User is in his/her Profile page |
| ***Exit Conditions*** | * The user's profile display is refreshed and the uploaded image is displayed as the user's current profile picture. |
| ***Exceptions*** | * The user clicks the "Upload Picture" button without selecting a file. * The user chooses to upload a file that is not one of the supported file types.   Note: If any of the above occur, the user will be presented with an appropriate message informing them of the error. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_710) Upload a File |
| Related Use Cases | Delete a File, Download a File |
| Participating Actor | Student, Professor and Head Professor |
| Flow of Events | 1.      The use case begins when the user clicks the “Browse...” button in the web application. |
| 2.      The system shall provide a new window explorer to select the file to be uploaded. |
| 3.      The user shall click the "upload" button. |
| 4. The system sends a requests to initiate an upload. |
| 5. The system then selects the project that the user is currently in. |
| 6. The system retrieves the file from the user's computer. |
| 7. The system now uploads the file into FIU's server by adding the file path, the user ID, the project ID, and the category of the file. |
| 8. Once the system has uploaded the file into the server, then it will display the file name. |
| Entry Conditions | 1.      Login as an active user. |
|  | 2. Select Repository tab from navigation bar. |
|  | 3. Click the "Expand All" button. |
|  | 4.      Except for the Head Professor, must be a member of the selected project. |
| Exit Conditions | 1.      The system displays the file name in the repository. |
| Exceptions | 1. The system does not display the file name |
|  | 2. The system displays the file name but it is not store in the database. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_720) Download a File |
| Related Use Cases | Delete a File, Upload a File |
| Participating Actor | Student, Professor and Head Professor |
| Flow of Events | 1.      The use case begins when the user clicks the “Download” button next to the file. |
| 2.      The system sends a request to initiate a download. |
| 3.      The system requests the path of the selected file to be downloaded. |
| 4. The file path is then retrieve from the database. |
| 5. The system then downloads the physical file from FIU's server. |
| 6. The system now displays the file in the user's computer. |
| Entry Conditions | 1.      Login as an active user. |
|  | 2. Select Repository tab from navigation bar. |
|  | 3. Click the "Expand All" button. |
| Exit Conditions | 1.      The file is downloaded into the user's computer. |
| Exceptions | 1. The file was not found in the database. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_730) Delete a File |
| Related Use Cases | Download a File, Upload a File |
| Participating Actor | Student, Professor and Head Professor |
| Flow of Events | 1.      The use case begins when the user marks the checkbox next the file name. |
| 2.      The user clicks the "Delete" button. |
| 3.      The system requests the path of the selected file to be downloaded. |
| 4. The file path is then retrieve from the database. |
| 5. The system then deletes the physical file from FIU's server. |
| 6. The system now removes the file name from the repository. |
| Entry Conditions | 1.      Login as an active user. |
|  | 2. Select Repository tab from navigation bar. |
|  | 3. Click the "Expand All" button. |
|  | 4.      Except for the Head Professor, must be the owner of the file to be deleted. |
| Exit Conditions | 1.      The file is deleted from FIU's server. |
| Exceptions | 1. The file was not found in the database. |
|  | 2. The file was not removed from the server. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_901) Add New Milestone |
| Related Use Cases | Edit Milestone, Delete Milestone |
| Participating Actor | Head Professor |
| Flow of Events | 1. The use case begins when the head professor clicks the “Manage Milestones” button. |
| 2. The head professor shall then click the "Add New" button. |
| 3. The system displays a new row with a text field for the name, a checkbox for deletion, and a calendar icon and text field to set the due date. |
| 4. The head professor shall type the name of the new milestone on the first field text provided. |
| 5. The head professor shall click the calendar icon and select a due date. |
| 6. The head professor shall click the "Save" button. |
| 7. The system allocates the new milestone in the repository |
| Entry Conditions | 1. Head Professor must be logged in. |
|  | 2. Navigate to the Files repository page. |
| Exit Conditions | The system displays the new milestone's name and due date. |
| Exceptions | The system does not display the new milestone's name and due date. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_902) Edit Milestone |
| Related Use Cases | Delete Milestone, Add New Milestone |
| Participating Actor | Head Professor |
| Flow of Events | 1. The use case begins when the head professor clicks the “Manage Milestones” button. |
| 2. The head professor shall edit the name or due date of an existing milestone. |
| 3. The system displays the edited name and due date. |
| 7. The head professor shall click the "Save" button. |
| 8. The system updates the milestone's name and due date in the database. |
| Entry Conditions | 1. Head Professor must be logged in. |
|  | 2. Navigated to the Files repository page. |
| Exit Conditions | 1. The system does not display the correct due date. |
|  | 2. The system does not display the correct name. |
| Exceptions | The system displays no change. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_903) Delete Milestone |
| Related Use Cases | Edit Milestone, Add New Milestone |
| Participating Actor | Head Professor |
| Flow of Events | 1. The use case begins when the head professor clicks the “Manage Milestones” button. |
| 2. The head professor shall click the checkbox next to the name of the milestone he wants to delete. |
| 3. The head professor shall click the "Delete" button. |
| 4. The system then displays a warning window |
| 5. The head professor shall click "Ok" on the warning window. |
| 6. The system removes the milestone's row from the list |
| 7. The head professor shall click the "Save" button. |
| 8. The system deletes the milestones from the database. |
| Entry Conditions | 1. Head Professor must be logged in. |
|  | 2. Had navigated to the Files repository page. |
| Exit Conditions | The system does not display the milestone's row. |
| Exceptions | The system displays the deleted milestone's row. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_205) Rank Projects |
| Related Use Cases | Match Projects |
| Participating Actor | Head Professor and Student |
| Flow of Events | 1.      The use case begins when the user scrolls through the projects and gives a rank number (-1 to n) to each project. |
| 2.      The user then scrolls to the bottom of the page and clicks on the "Save" button. |
| Entry Conditions | 1.      Login as an active user. |
|  | 2. Select Current Projects tab from navigation bar. |
| Exit Conditions | 1.     The ranking scheme is saved in the system. |
| Exceptions | 1. A textbox was left blank, without a rank number. |

|  |  |
| --- | --- |
| Use Case Name | (SPW3\_206) Run Match |
| Related Use Cases | Rank Projects |
| Participating Actor | Head Professor |
| Flow of Events | 1. The use case begins when the head professor clicks the “Auto-Match” button at the bottom of the page. |
| 2. The head professor will be redirected to the "Matching" page. |
| 3. The system will be running the matching algorithm to find the best student-project match up. |
|  | 4. The head professor will then be redirected to the "Results" page that displays the projects and the students for that project. |
| Entry Conditions | 1. Head Professor must be logged in. |
|  | 2. Navigated to the Current Projects page. |
| Exit Conditions | 1. The system displays all students matched to projects. |
| Exceptions | The system displays no matches. |

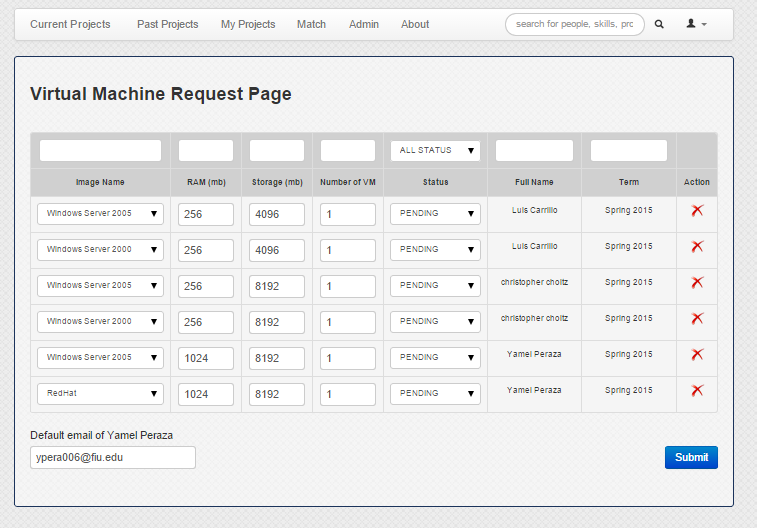
|  |  |
| --- | --- |
| Use Case Name | (SPW3\_210) Update Minimum |
| Related Use Cases | Rank Projects |
| Participating Actor | Head Professor |
| Flow of Events | 1. The use case begins when the head professor clicks the "Settings” button at the bottom of the "Current Projects" page. |
|  | 2. The head professor will then be redirected to a page that provides settings for the matching algorithm. |
|  | 3. The head professor will then set the required minimum and other settings at this page. |
|  | 4. The head professor will then scroll to the bottom of the page and click the "Save" button. |
|  | 5. The head professor will then be notified that the settings have been saved. |
| Entry Conditions | 1. Head Professor must be logged in. |
| Exit Conditions | 1. The system saves the settings to the algorithm provided by the head professor. |
| Exceptions |  |

* 1. Appendix C – User Interface Designs

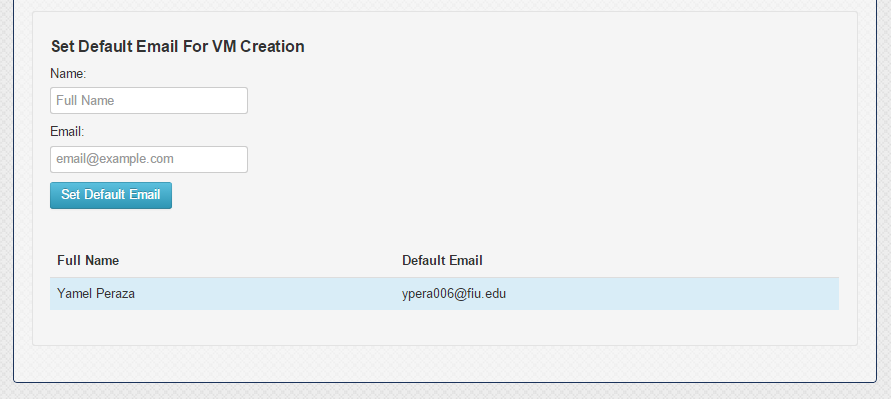
**UI from SPWv5**

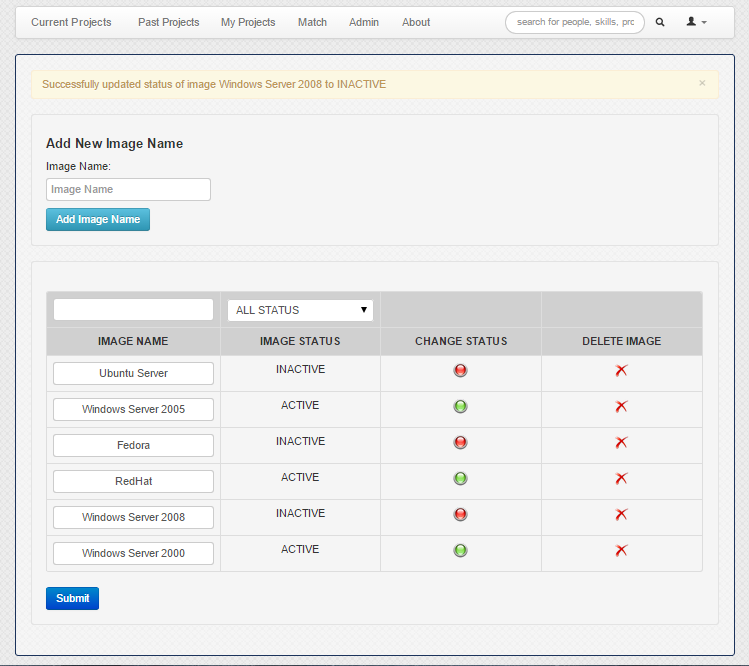
Student Request Virtual Machine page

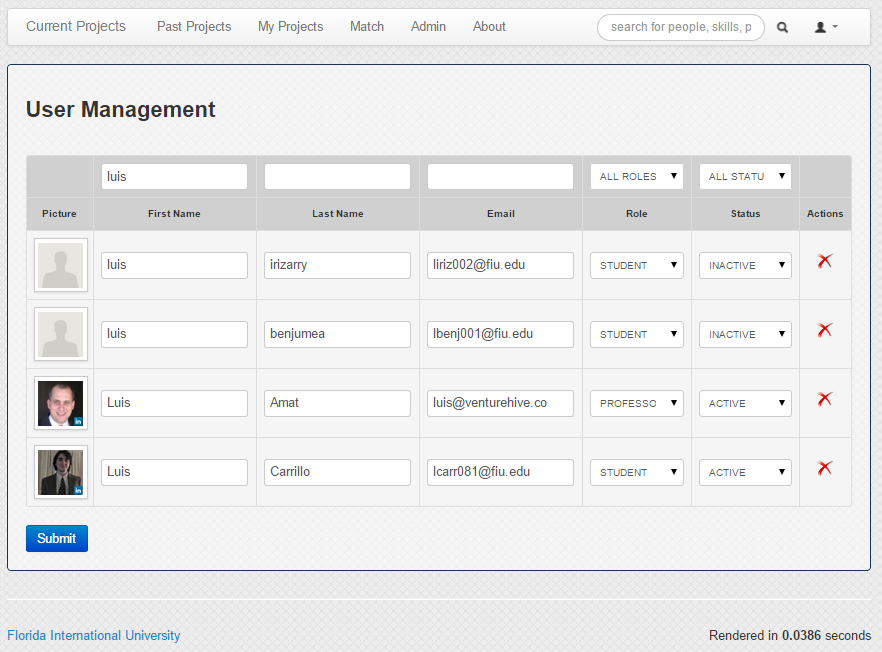


Head Professor Virtual Machine Request page

Set Default Email for Virtual Machine Creation/Notification to school’s system admin



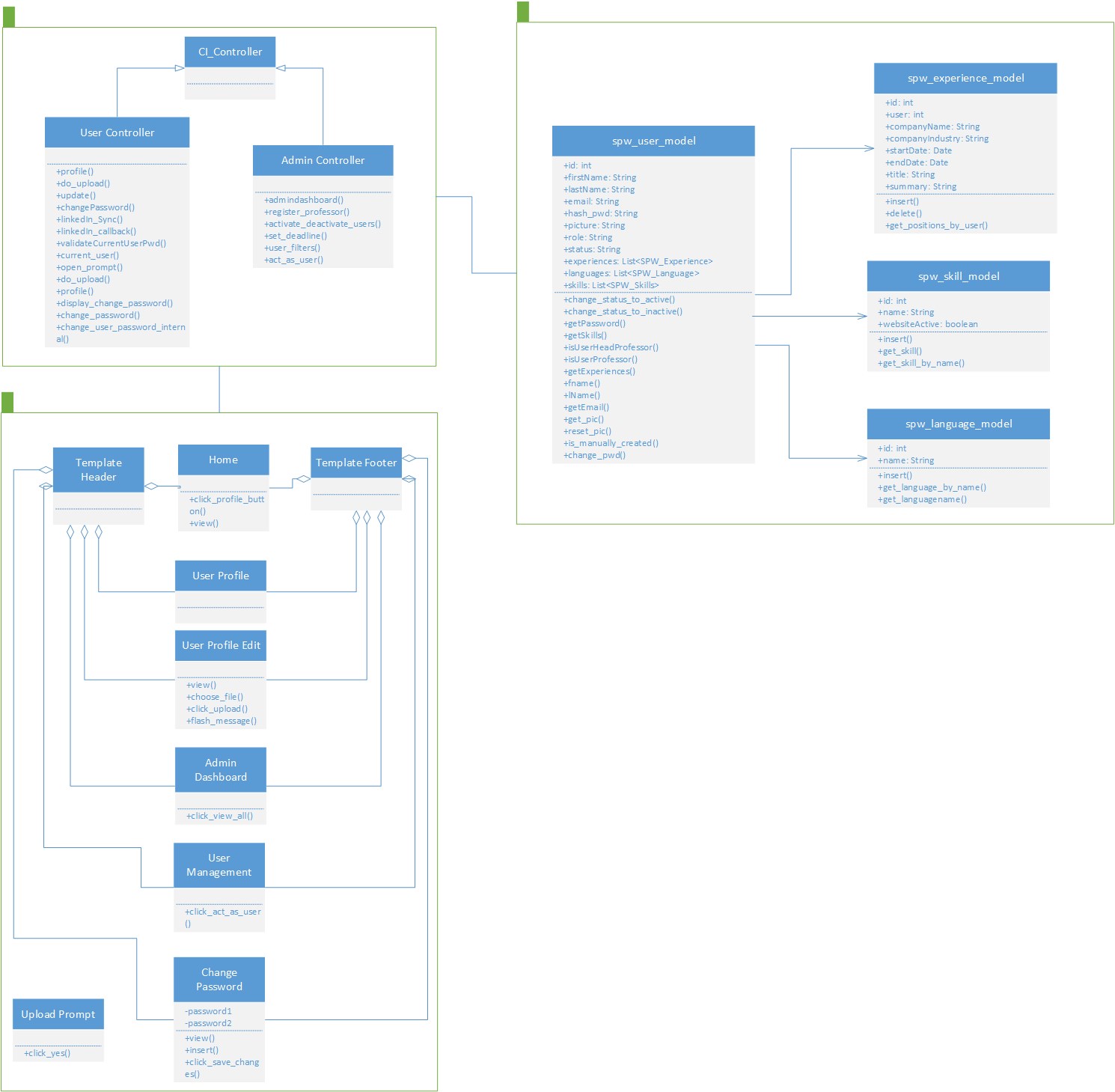
Image management page

New User Management page

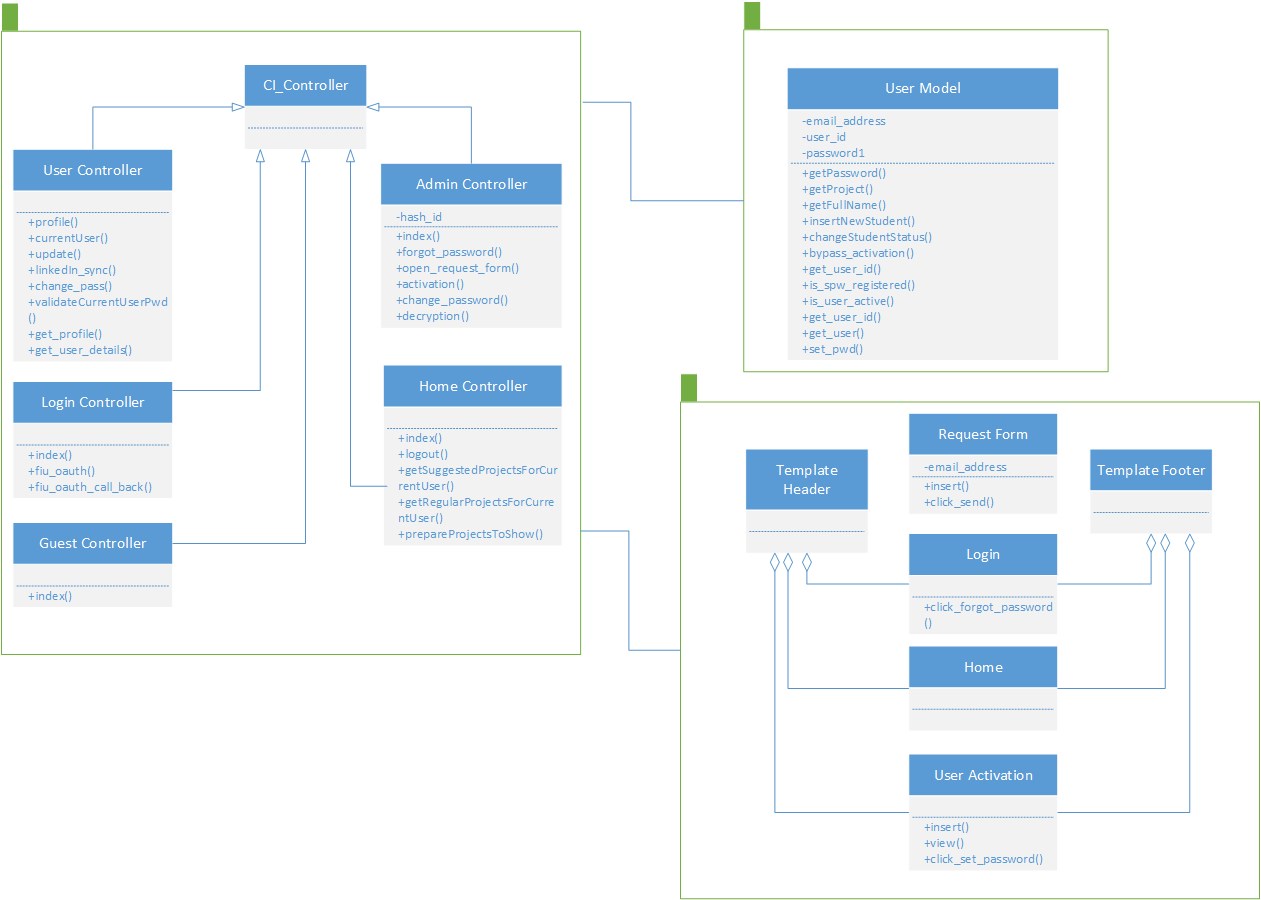
* 1. Appendix D – Static Models

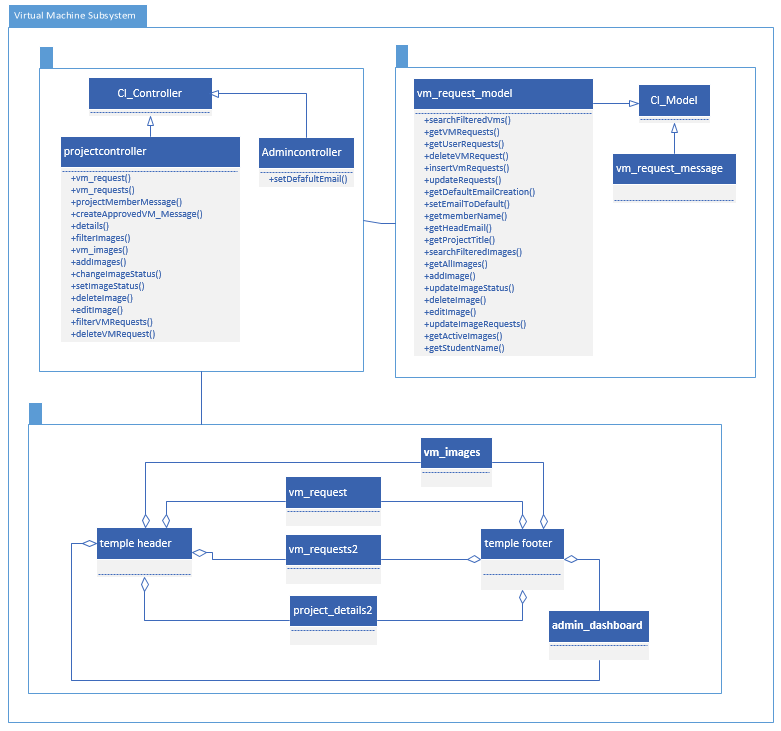
**Specific to Version 5:**

Profile subsystem

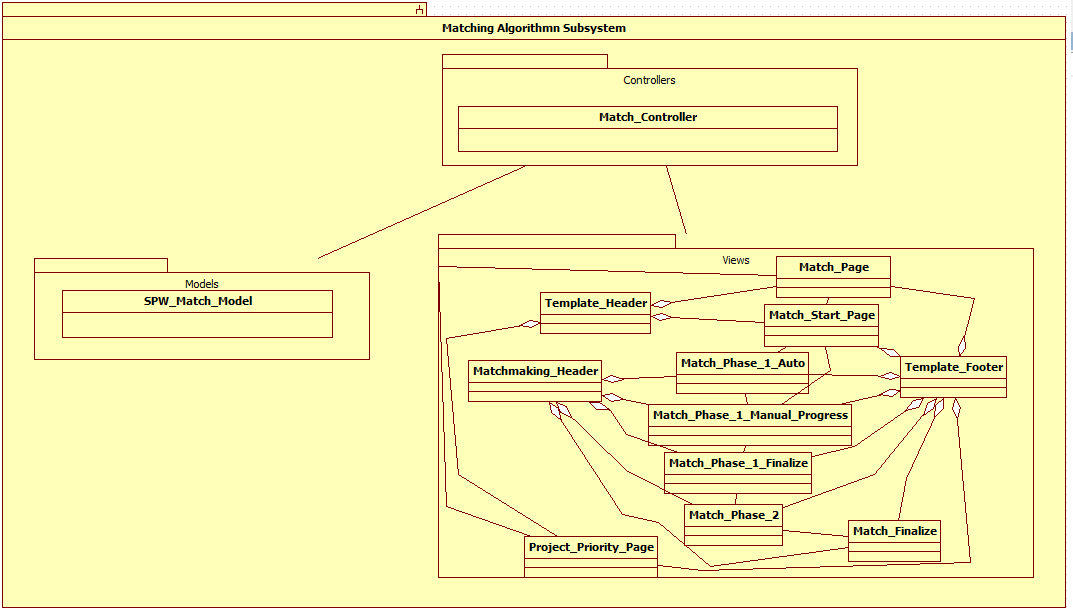


Logging subsystem

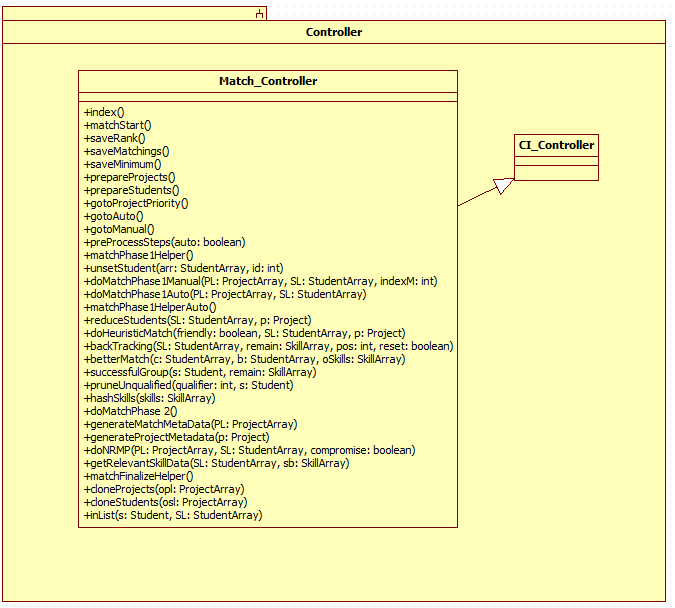




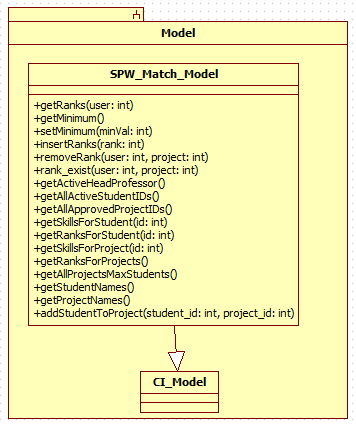
**Specific to version 4:**



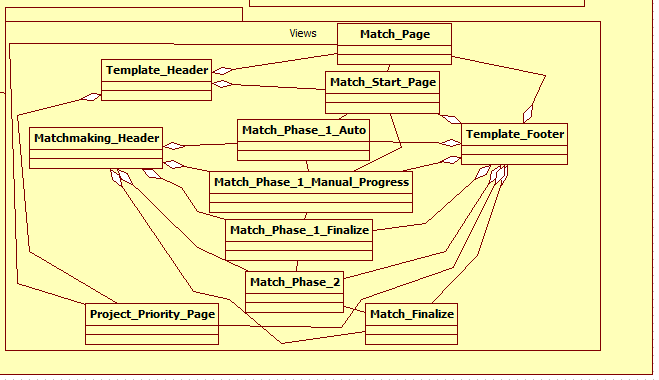
Minimal Class Diagram for Matchmaking

****

Controller for matchmaking

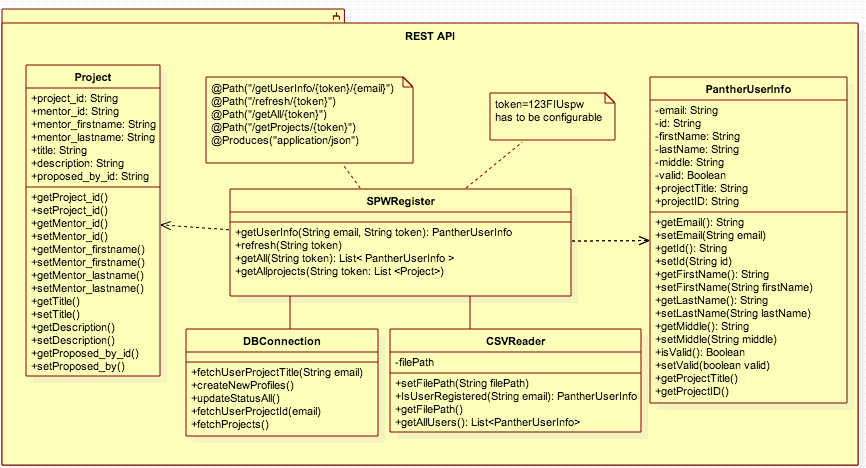


Model for matchmaking



Views for Matchmaking

**By Jonathan Santiago from Collaborative Platform v2 Team**

****

**Specific to version 3:**

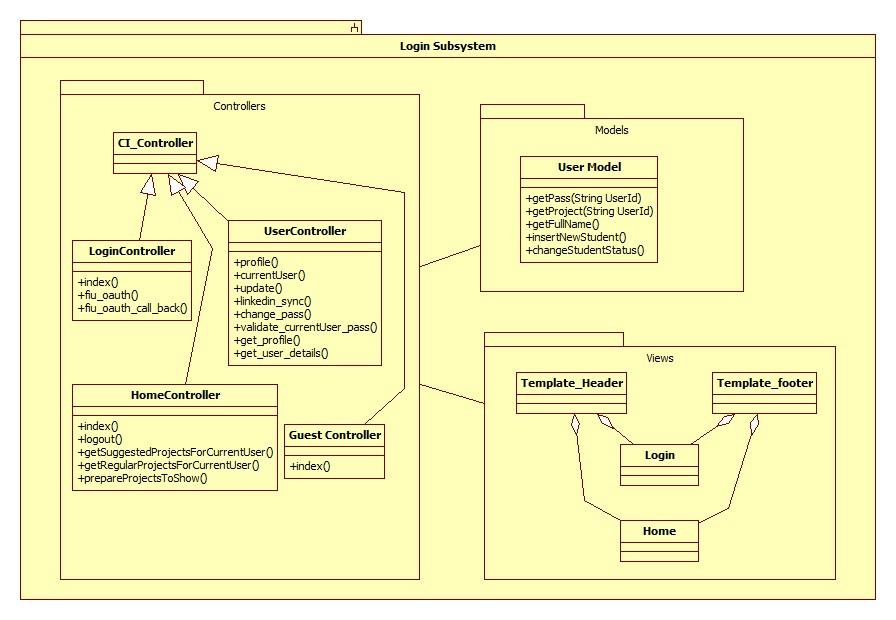
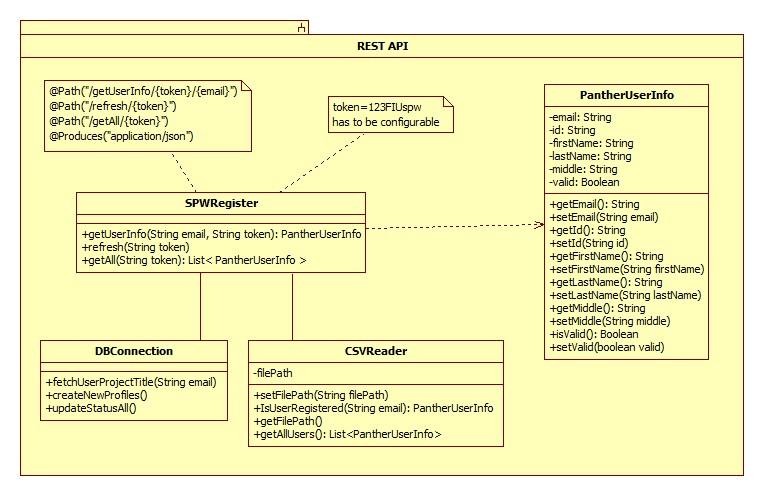
UML Detailed Class Diagrams for the SPWv.3



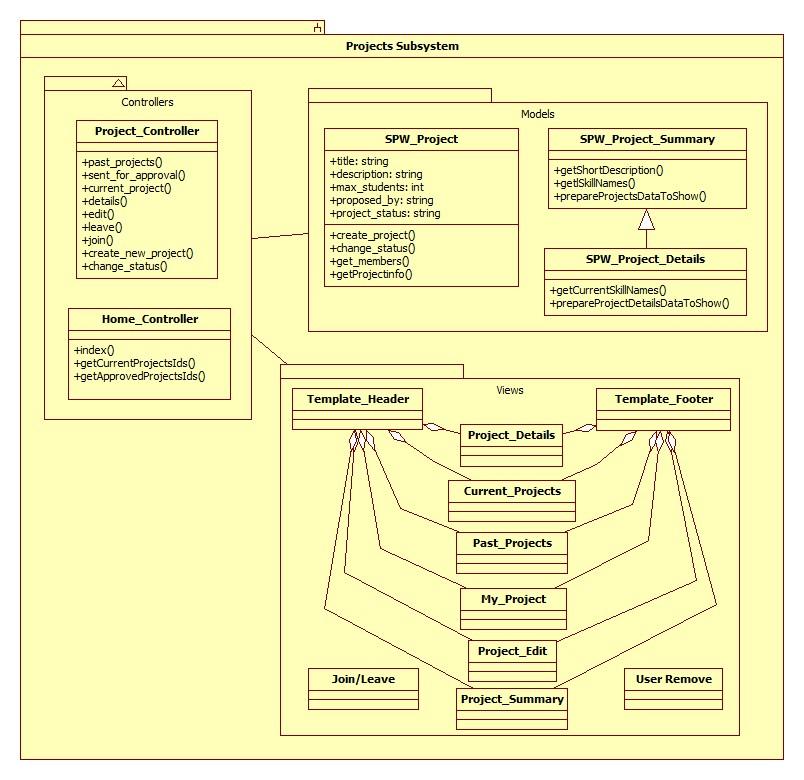




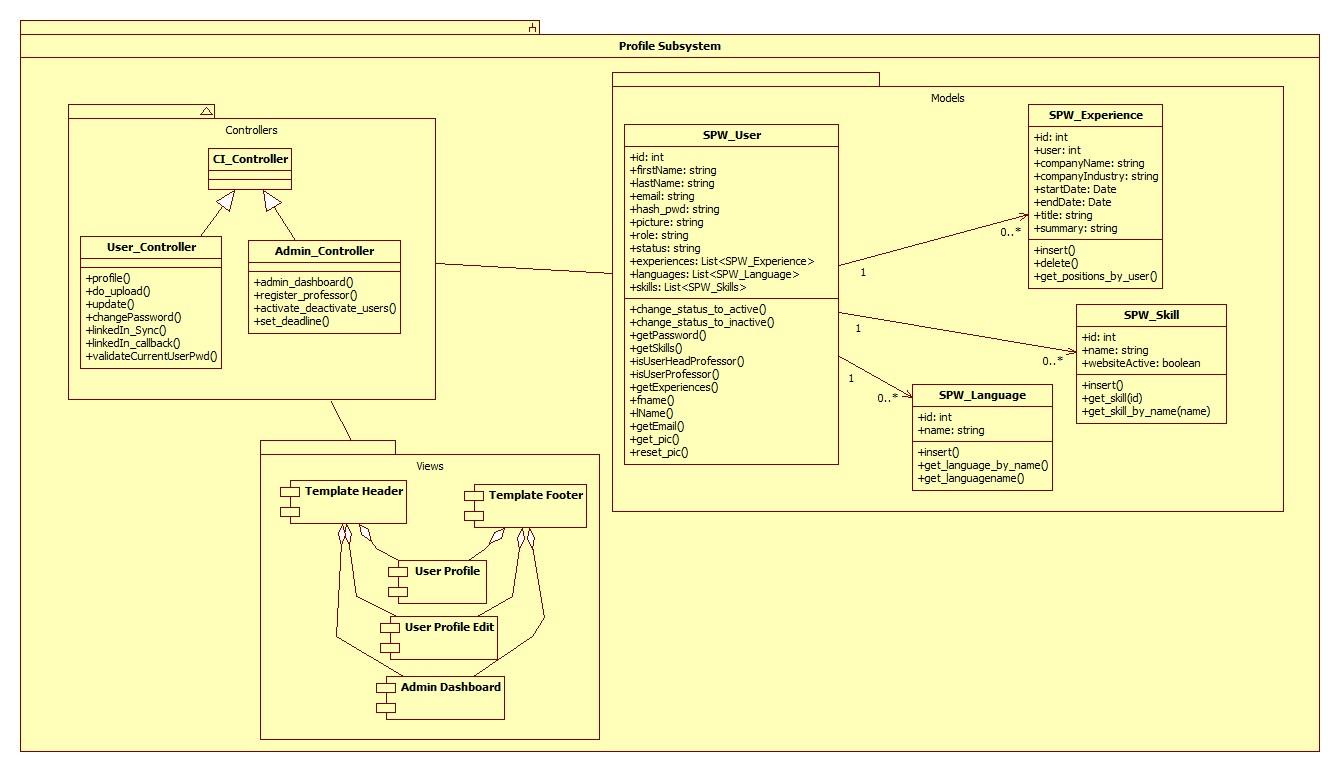
**Specific to version 2:**

 Class Diagram Login Subsystem

Class Diagram for SPW Register API



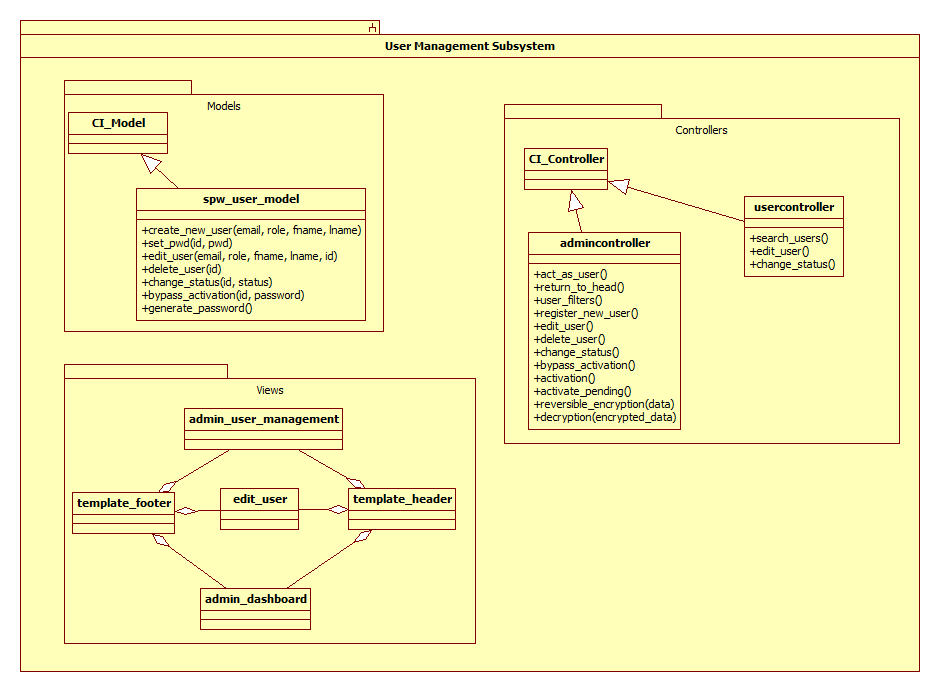
Class Diagram for the Projects Subsystem



Class Diagram for the Profile Subsystem



Class Diagram for the Repository Subsystem

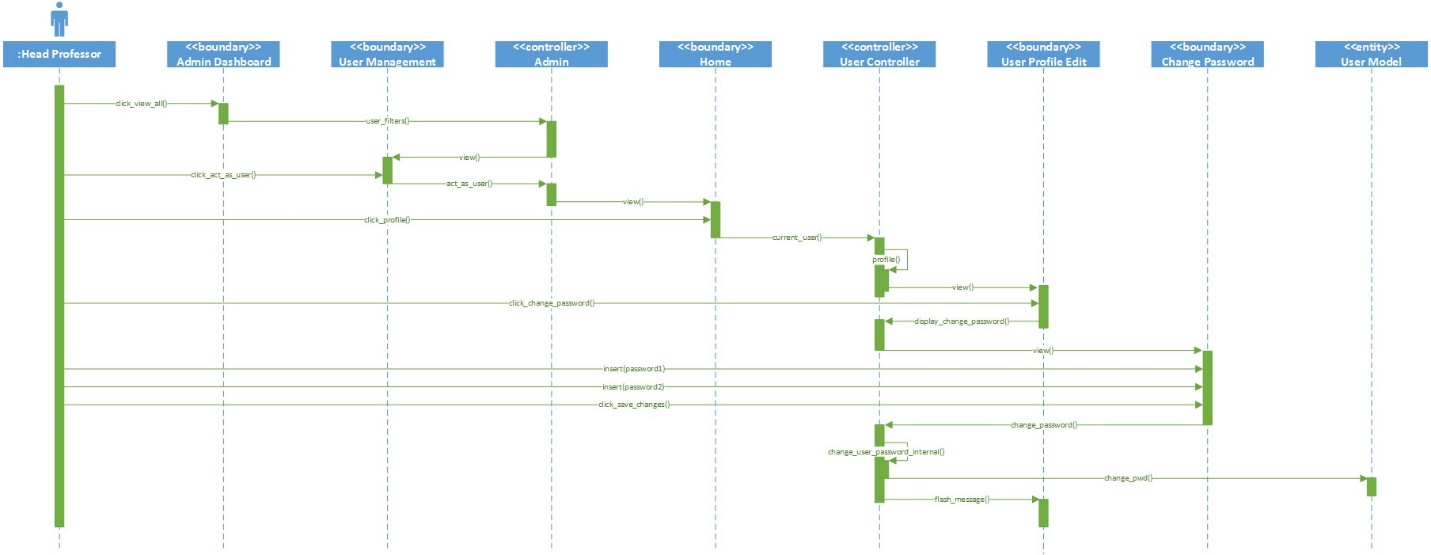


UML Class Diagram for the User Management subsystem

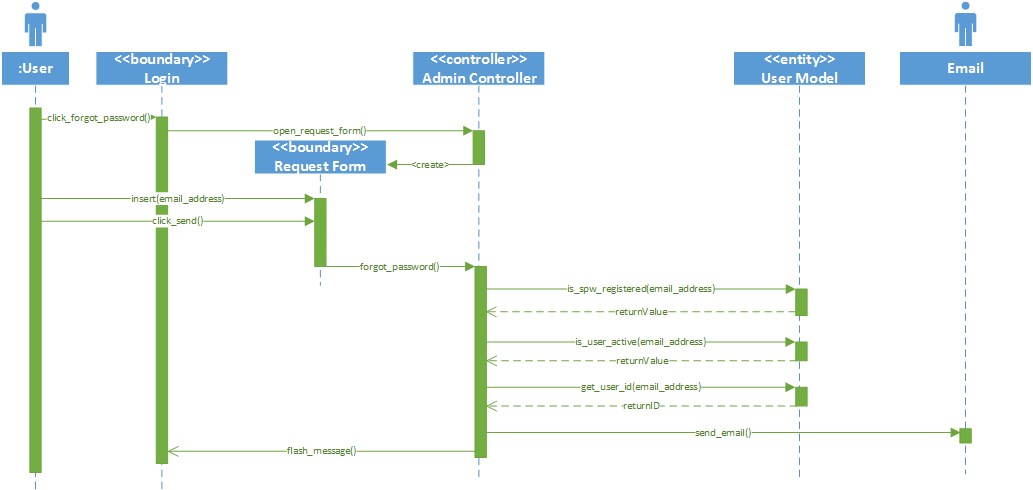
* 1. Appendix E – Dynamic Models

**Specific to SPWv.5**

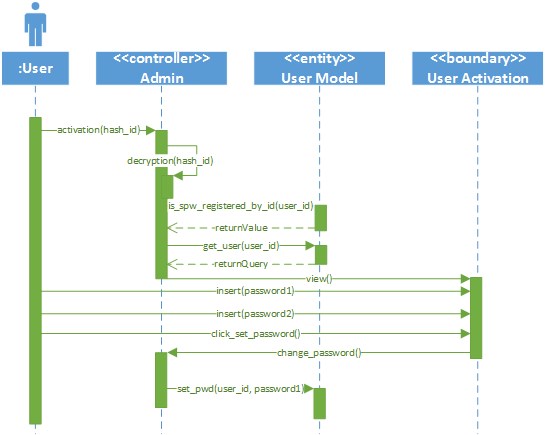
SPW5\_101 Change user’s password



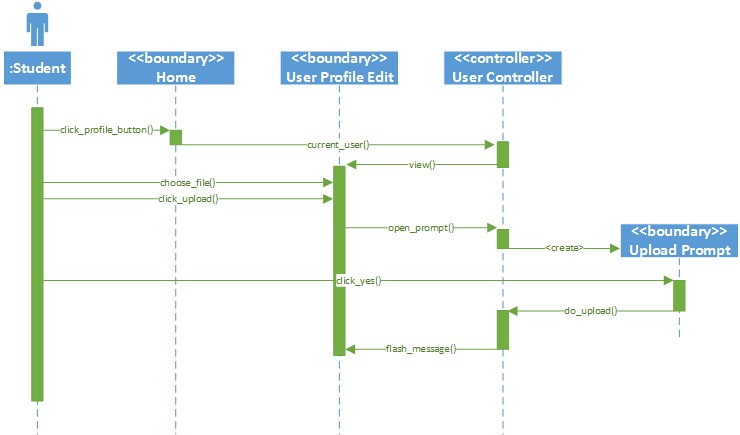
SPW5\_104 Request change of password

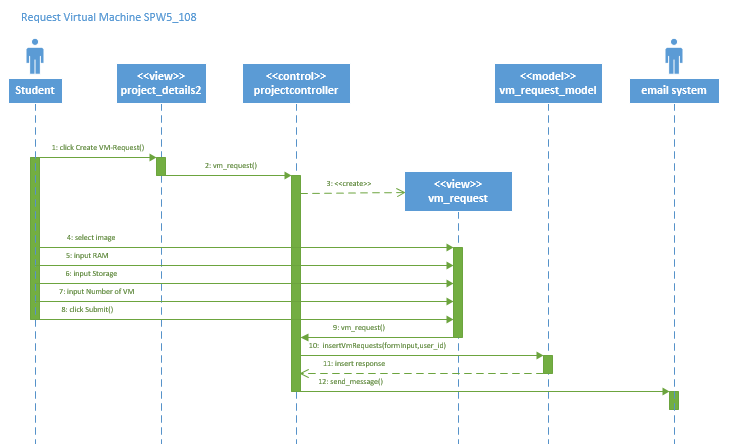


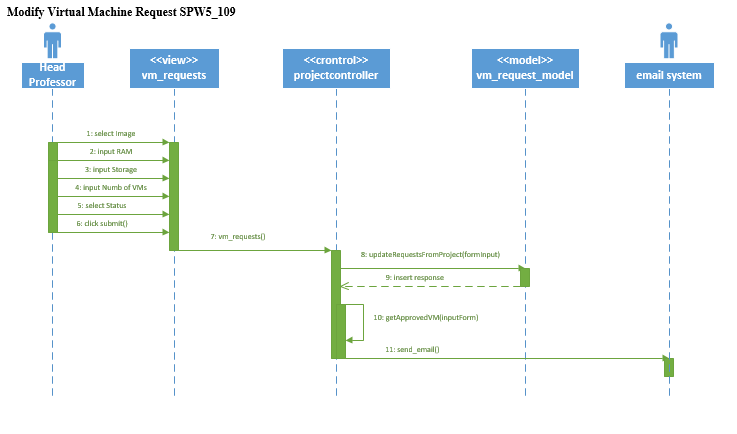
SPW5\_105 Change forgotten password

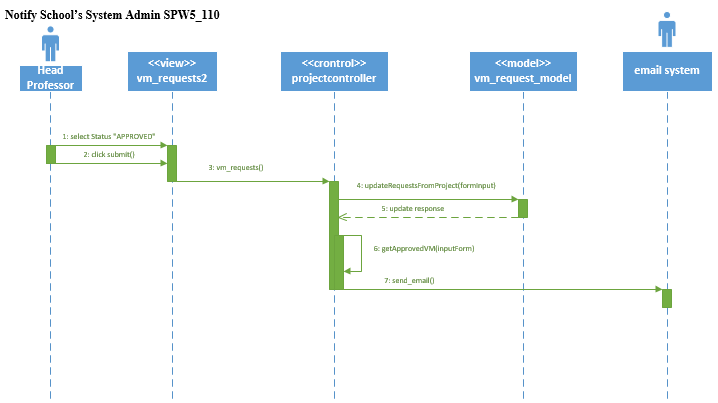


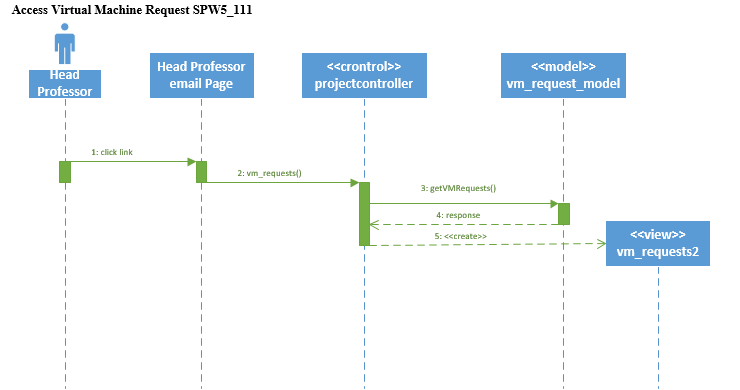
SPW5\_106 Change profile picture

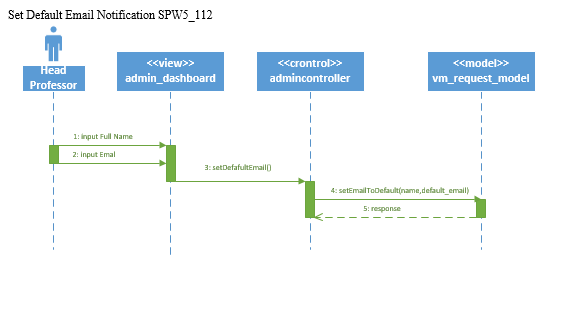


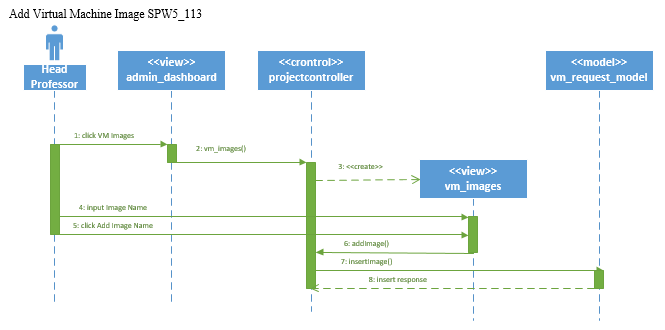


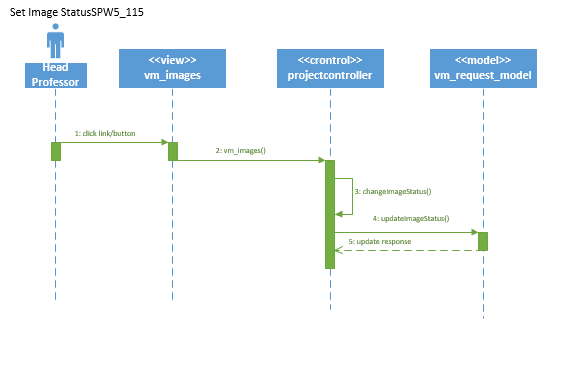


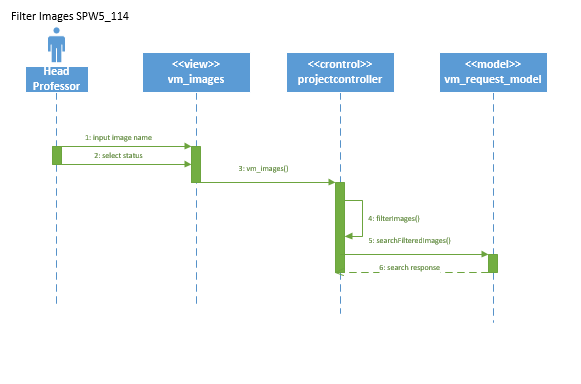


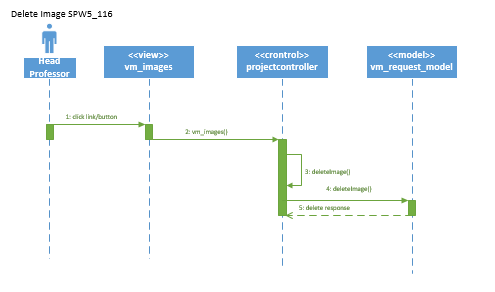


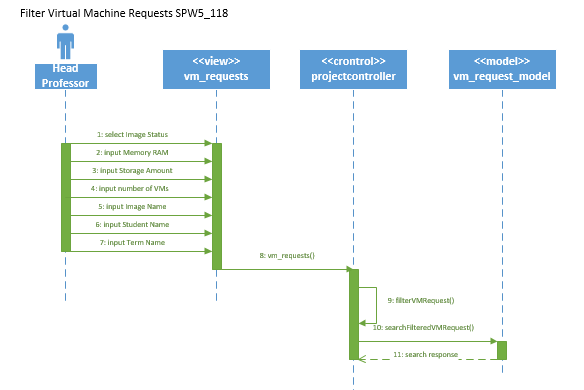


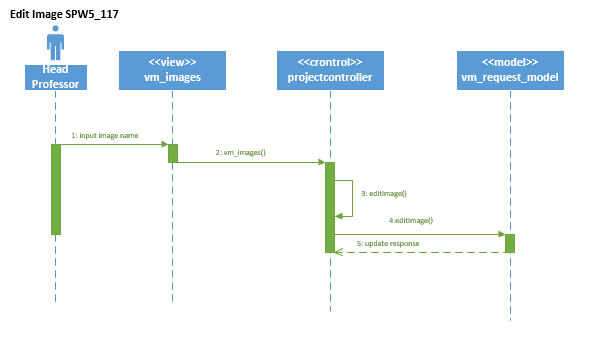


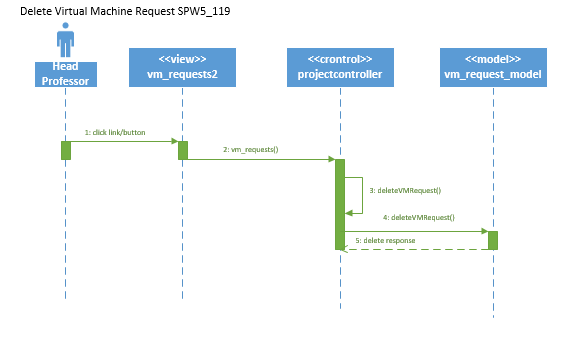




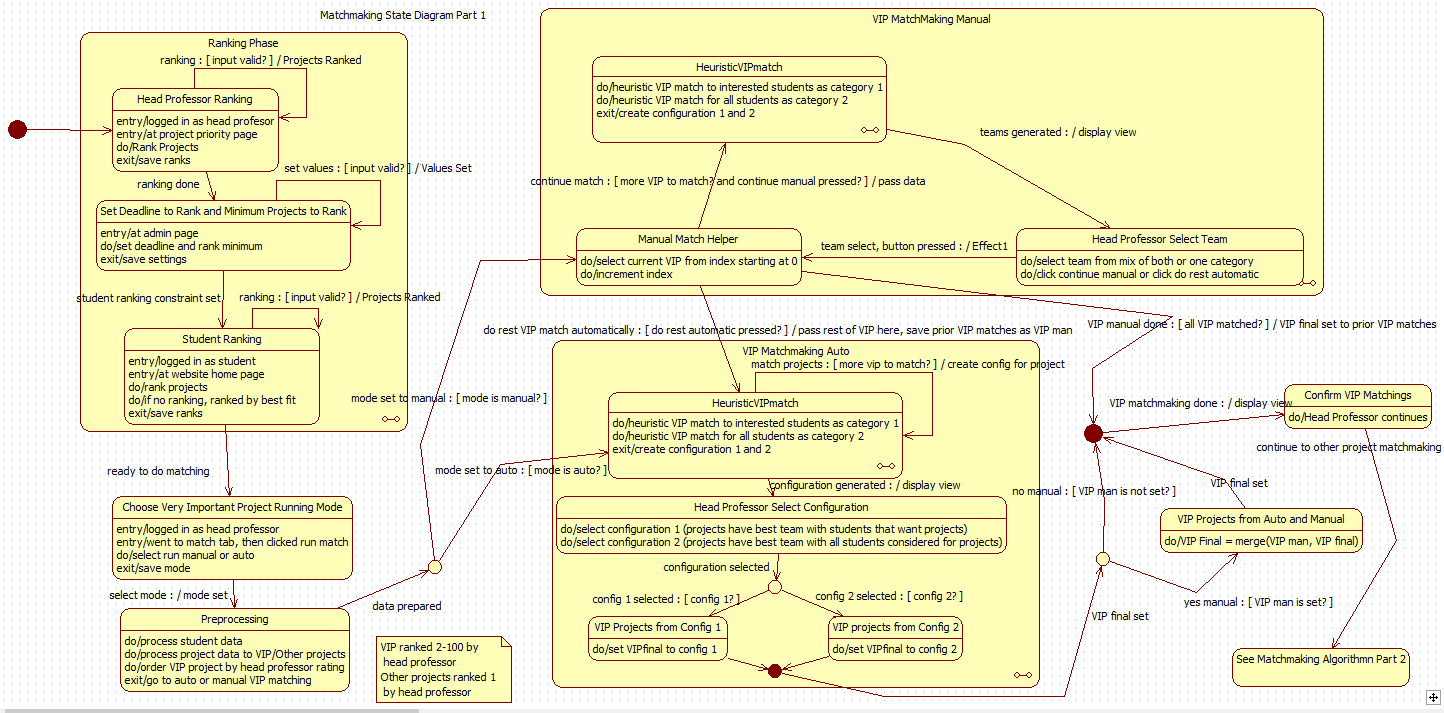




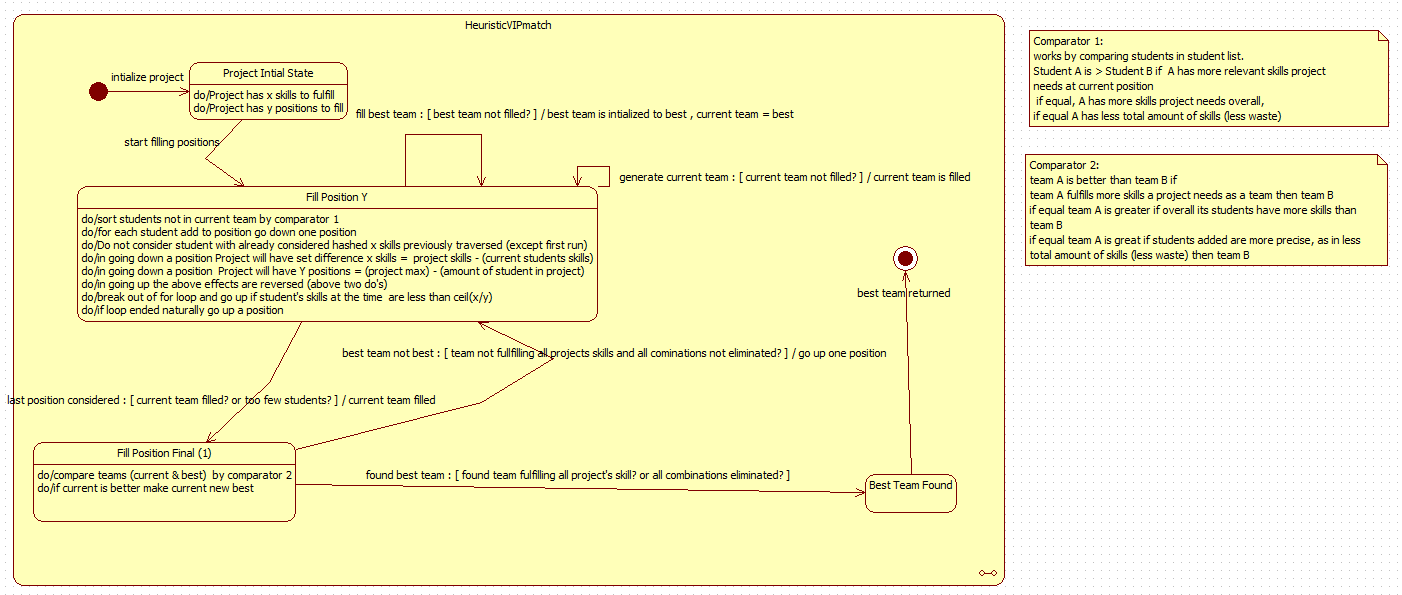


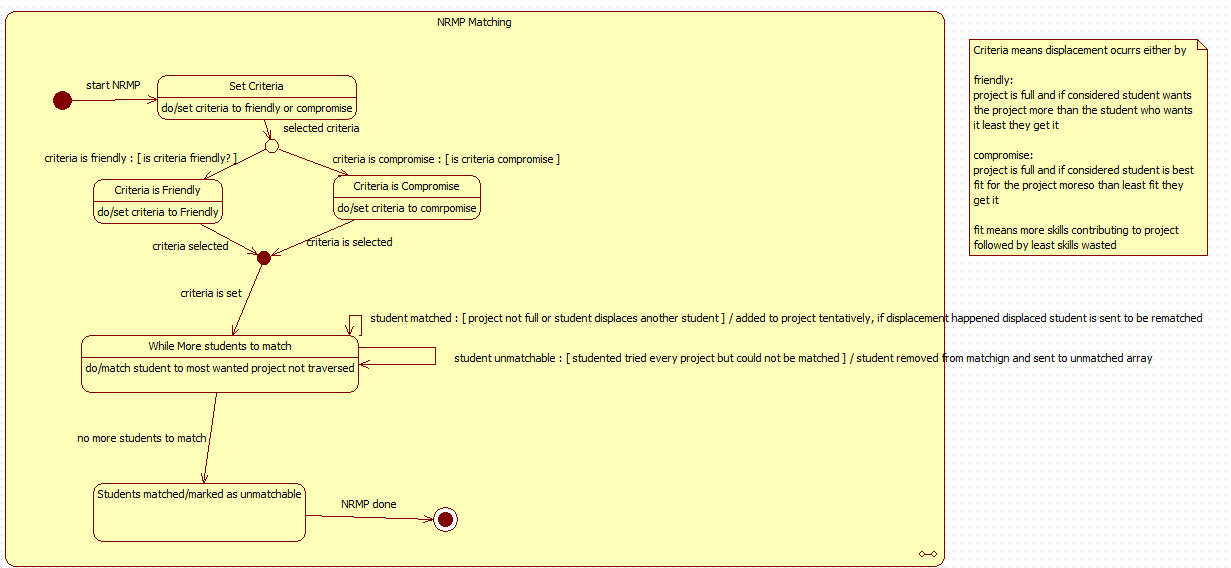


**Specific to SPWv.4:**

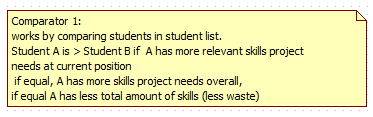
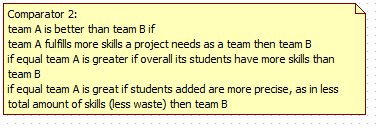
Following two diagrams depict matchmaking state machine.



The following state diagrams depict the states of Heuristic matchmaking and NRMP matching with more details.

****The specific algorithm details for the heuristic VIP are detailed below

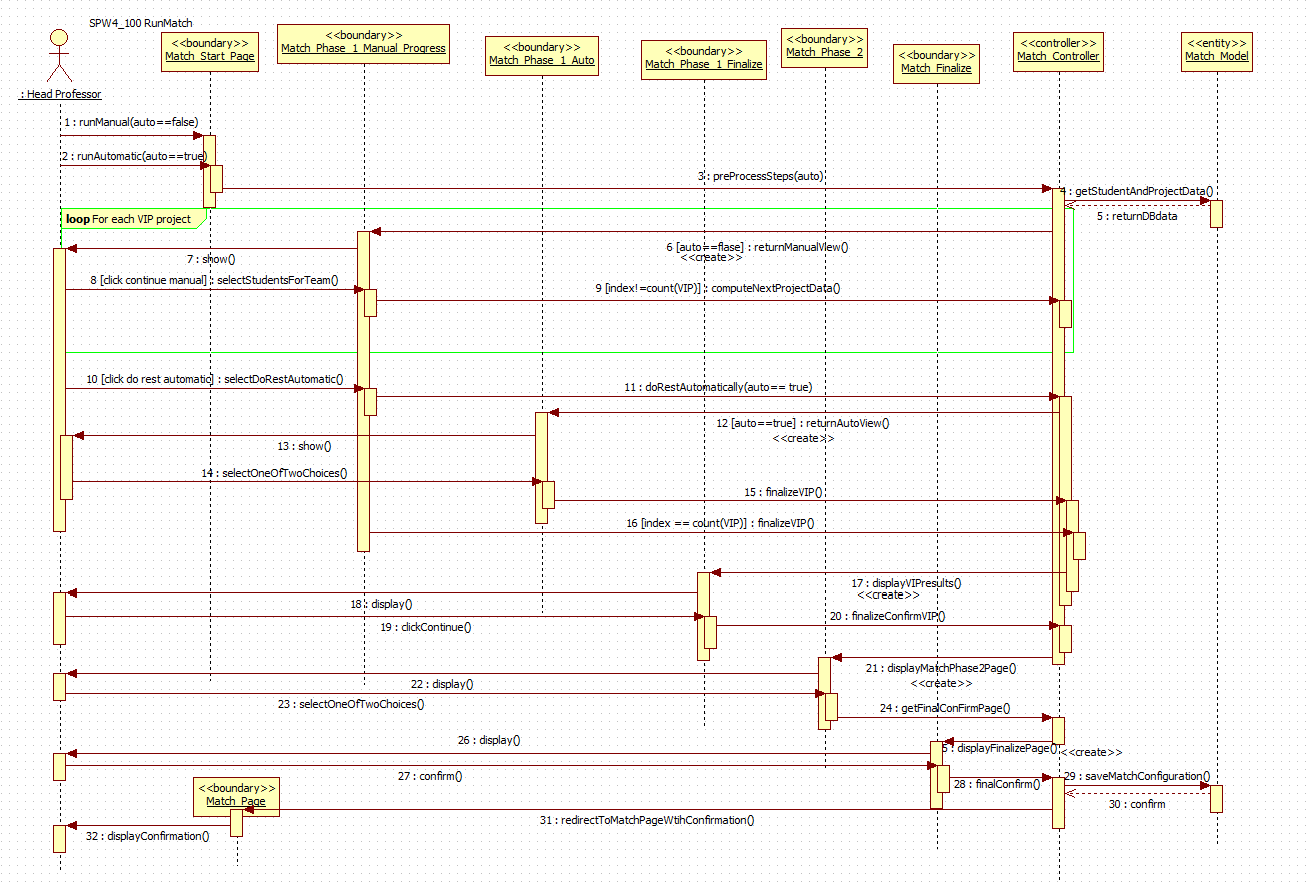
1. Given a project with X skill needs and Y positions
2. Go through the Y positions of the projects (going down until all position filled Y==0)
   1. For each student compose relevant skill array (skills project needs at the moment they have)
   2. Sort the student list by comparator 1 see below
   3. For each student
      1. Save relevant skill to has table
      2. If best team not initialized add student to current team and goto to 2. With Y-1 positions and X set difference (Students skills) skills
      3. If student’s relevant skills is less than ceil(count(X)/Y) exit 2.b.
      4. If student’s relevant skill in hash table go to 2.b. to next student
      5. If position Y is not 1, add student to current and goto to 2. With Y-1 positions and X set difference (Students skills) skills
      6. If position is 1, add student to current and compare current to best,
         1. if current better (See comparator 2) and fulfills all project skills exit 2. and best = current
         2. else go to 2.c.
   4. If exit go back to Y+1 2.b. removing latest addition to current
3. Best team is best team



The specific algorithm details for National Residency Matchmaking

1. Select criteria to displace students as friendly (displace student who wants project least with one who wants it most) or compromise (displace student who contributes least to the project least with one who contributes more)
2. For each student unmatched
   1. Go through each project not yet traversed in the order they ranked them
      1. If position open student is matched and removed from unmatched, continue to 2.
      2. Else if student is better than the worst one based on criteria, add student to project, removed from unmatched and put worst student to unmatched, continue to 2.
   2. If student reached here and is unmatched remove from unmatched and put in unmatchable
3. End result students are matched or said to be unmatchable (i.e. their ranking ended with no match for them or too few projects)

Following is sequence diagram for running a match



Sequence diagram for ranking projects for students and head professor. Validation has changed so remains same from SPWv.3 other than validation of results.

Student Save Rank (SPW3\_205)



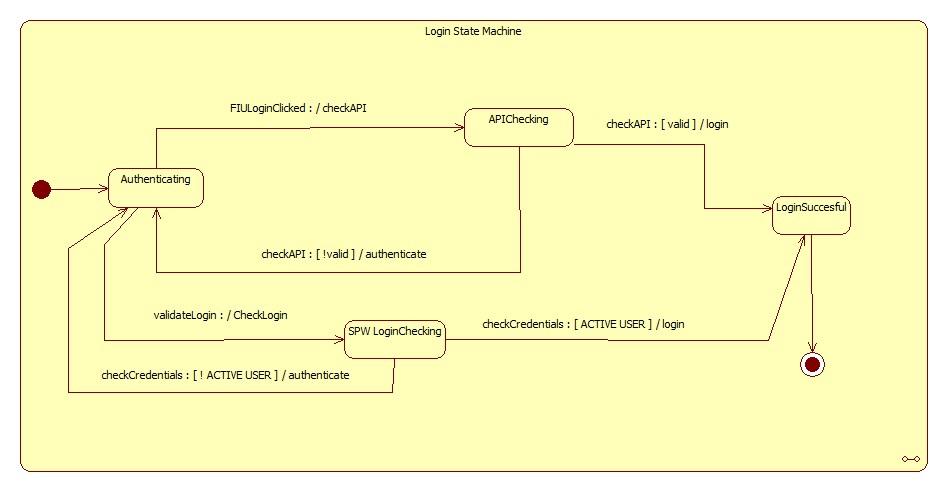
Professor Save Rank (SPW3\_205)

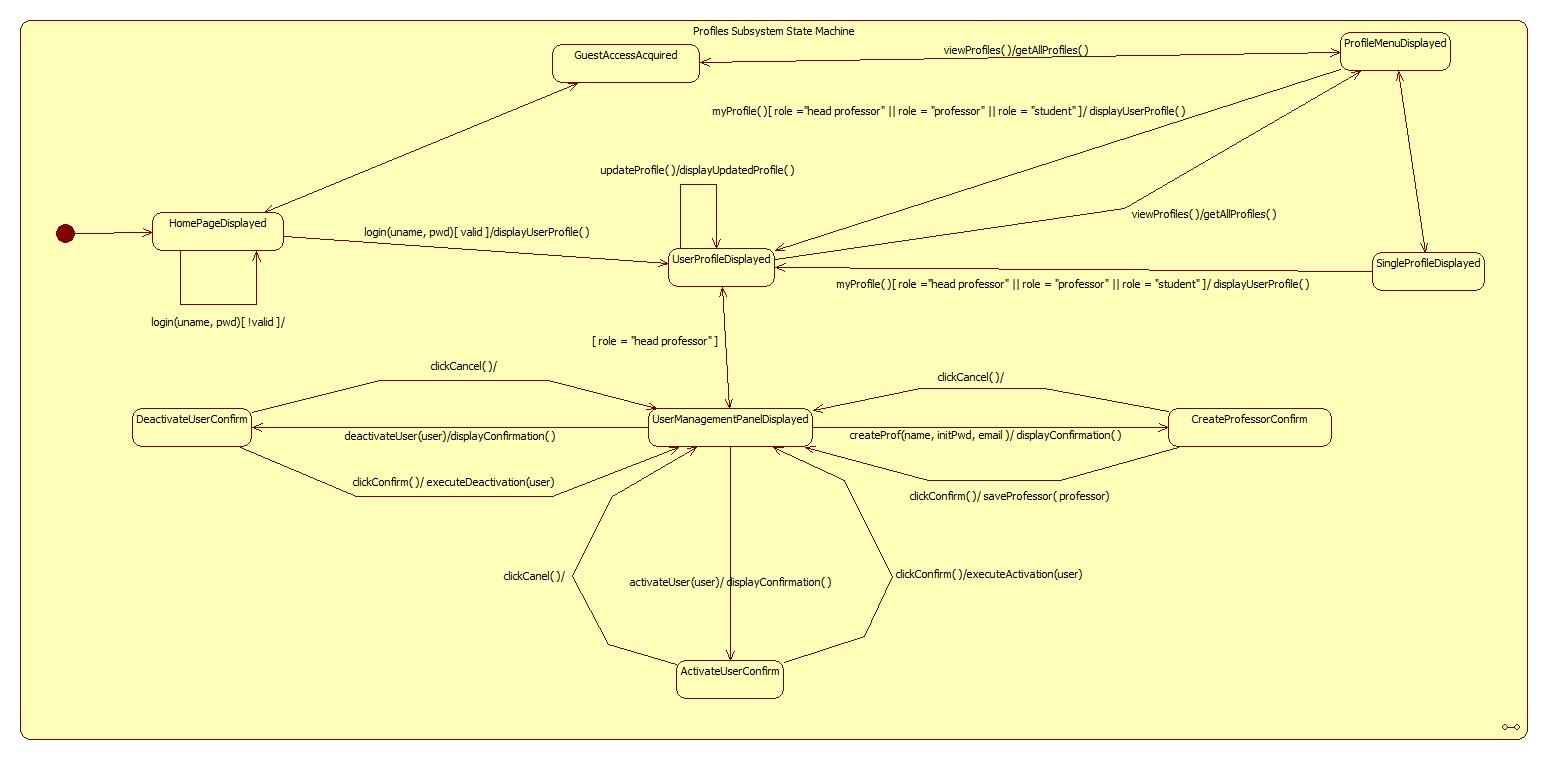
Project Priority Page not Match Page



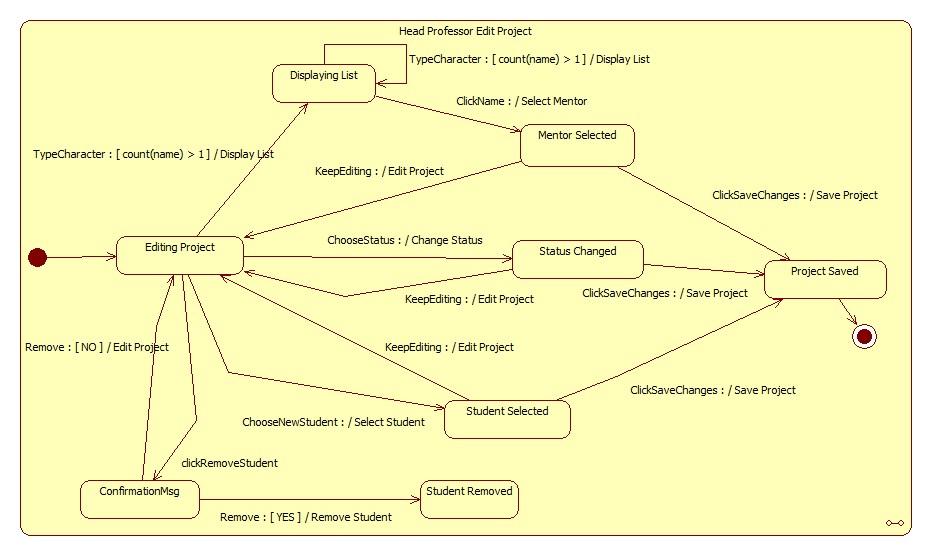
**The following dynamic models are from SPWv.3:**

**Login State Machine**



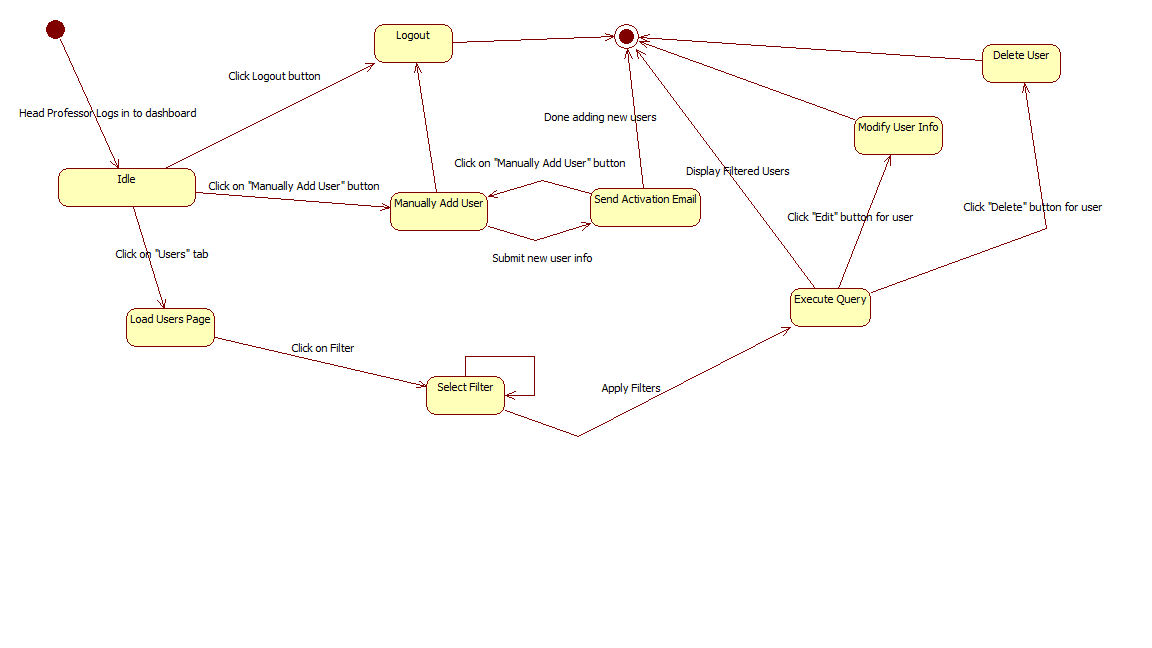
**Profile Subsystem State Machine**

**Head Professor Edit Project**



## Activity Diagram for Repository:

****

**Activity Diagram for User Management: **

Professor Login (SPW2\_103)

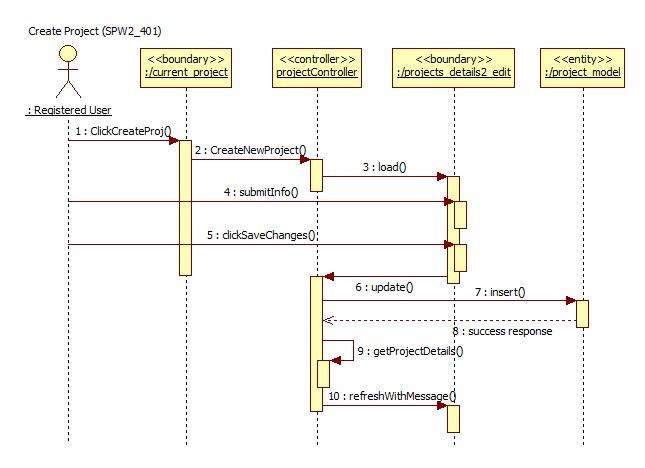


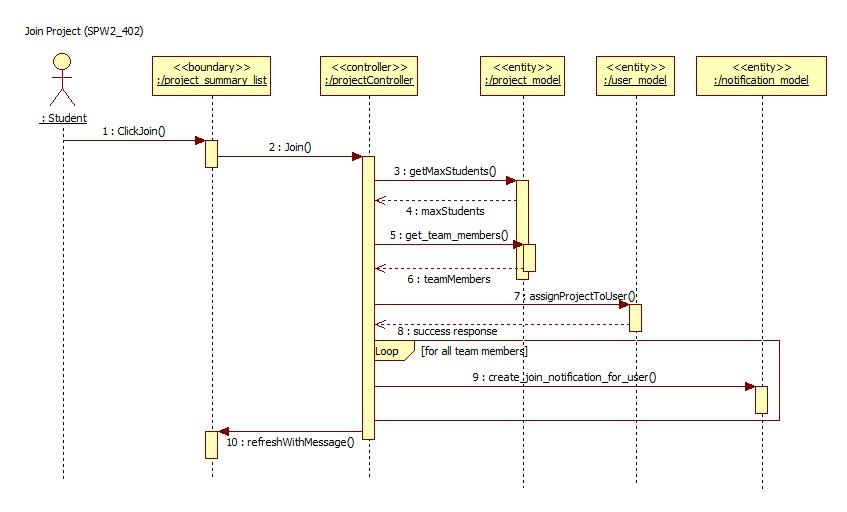
Guest Access (SPW2\_102)

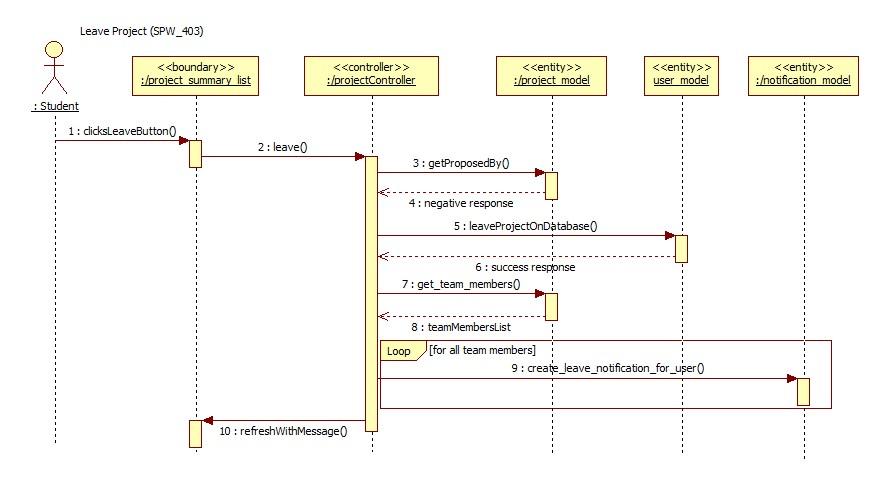


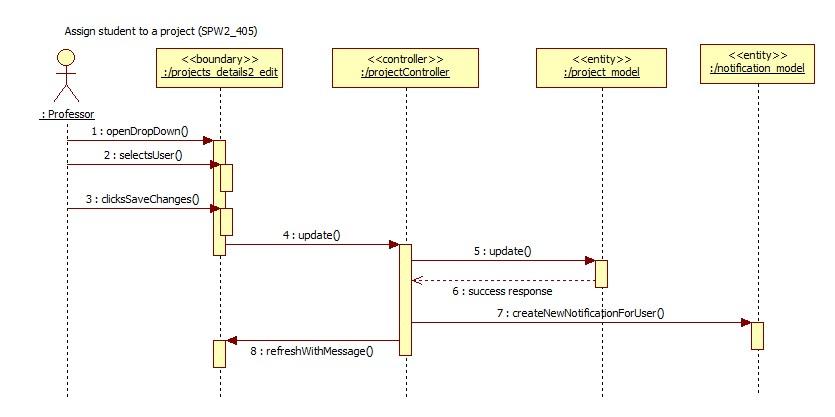
Logout (SPW2\_104)

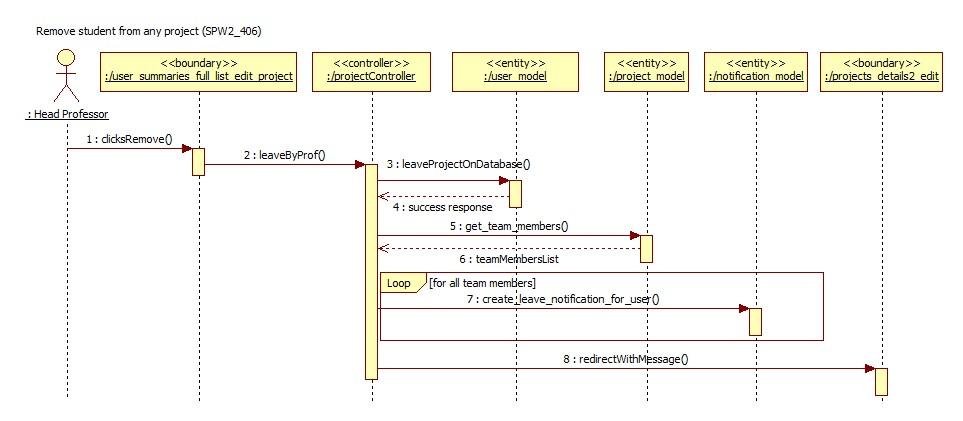


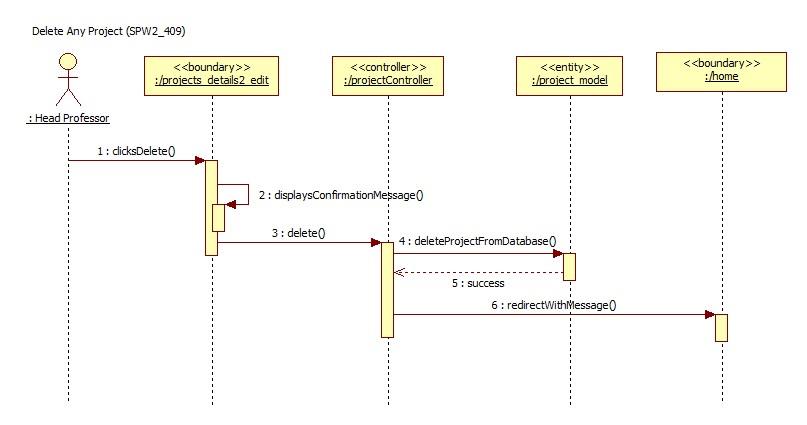


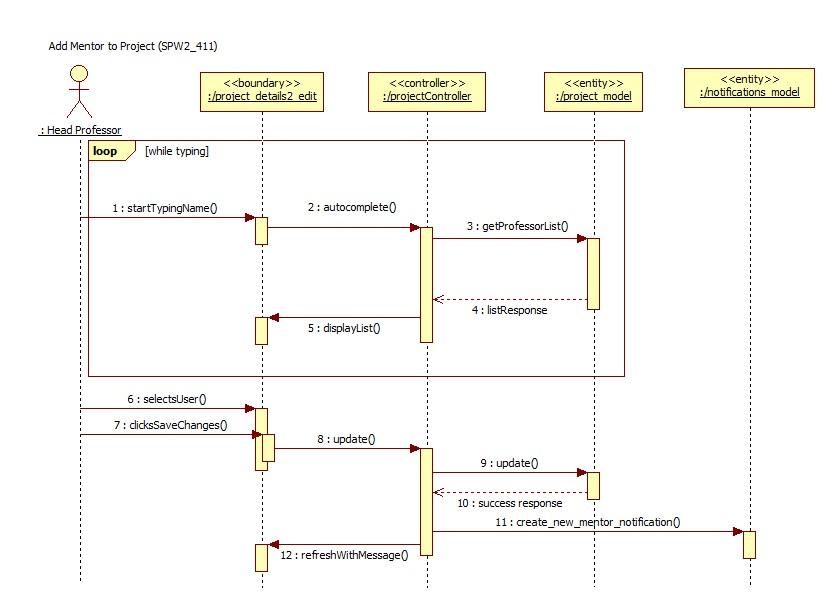


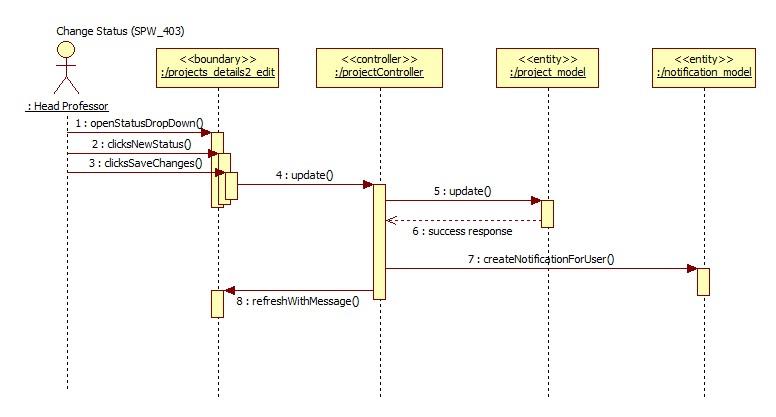


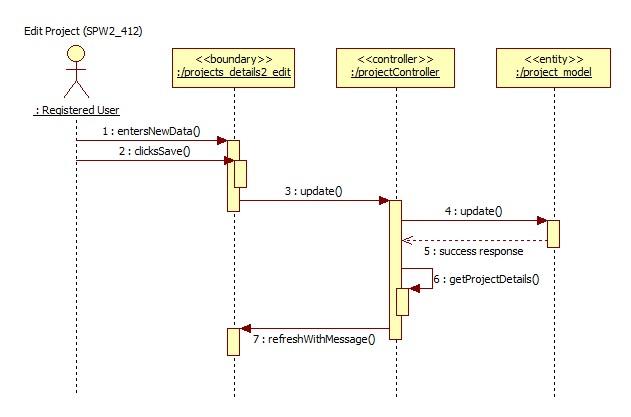


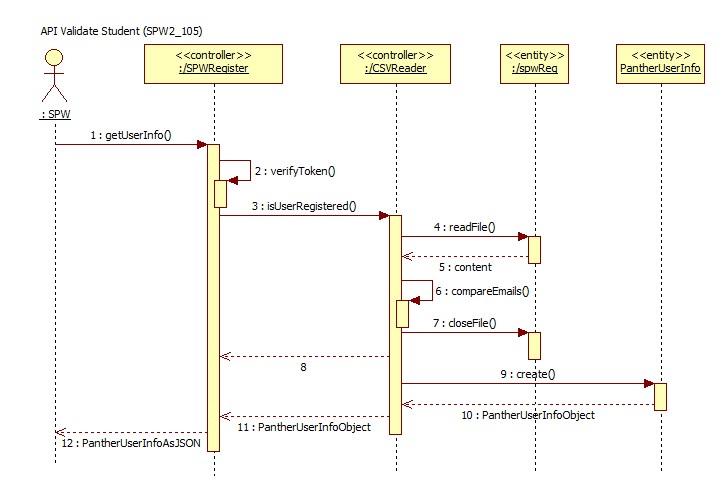


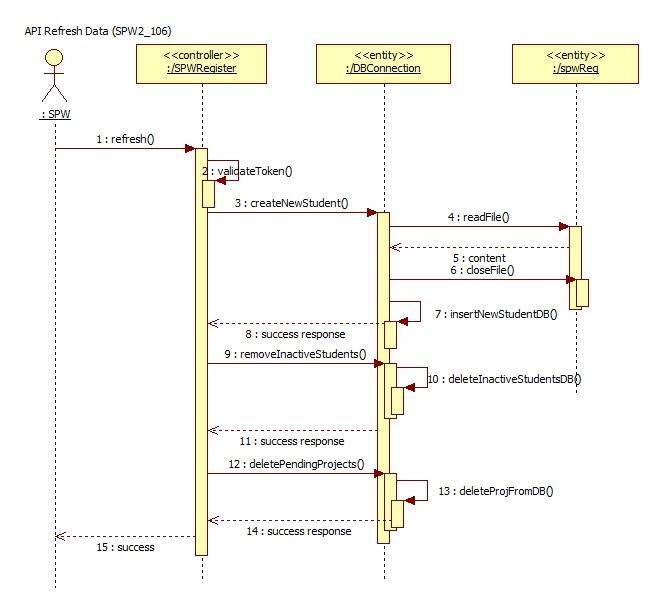


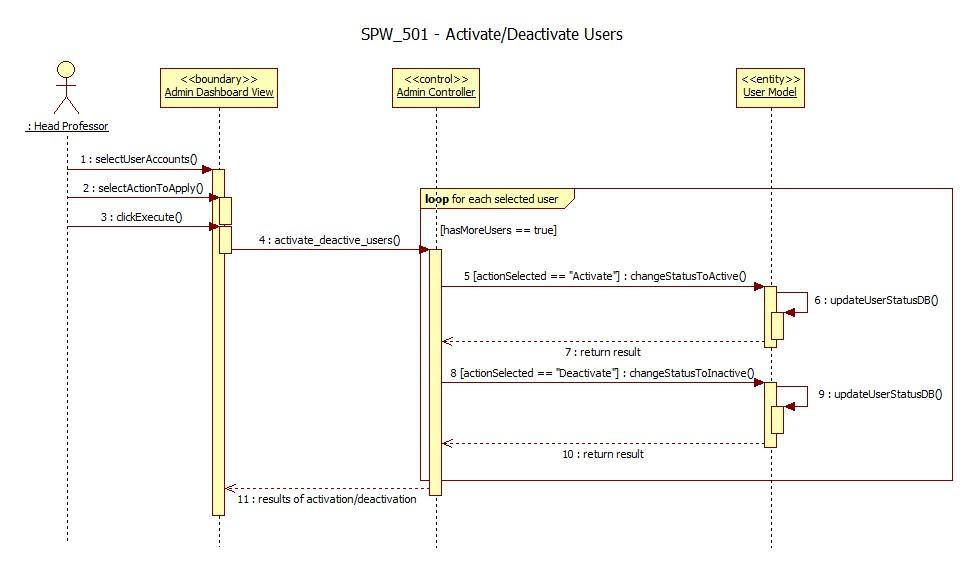


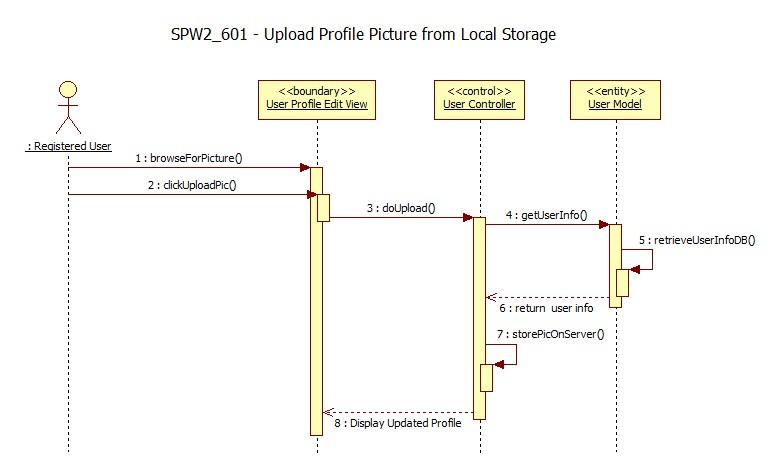
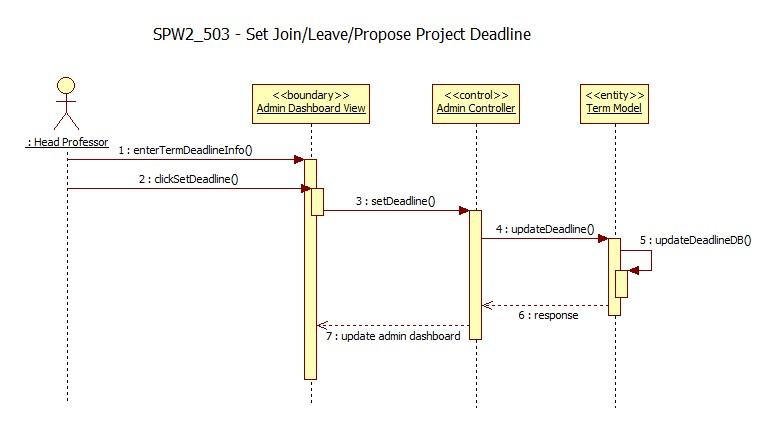
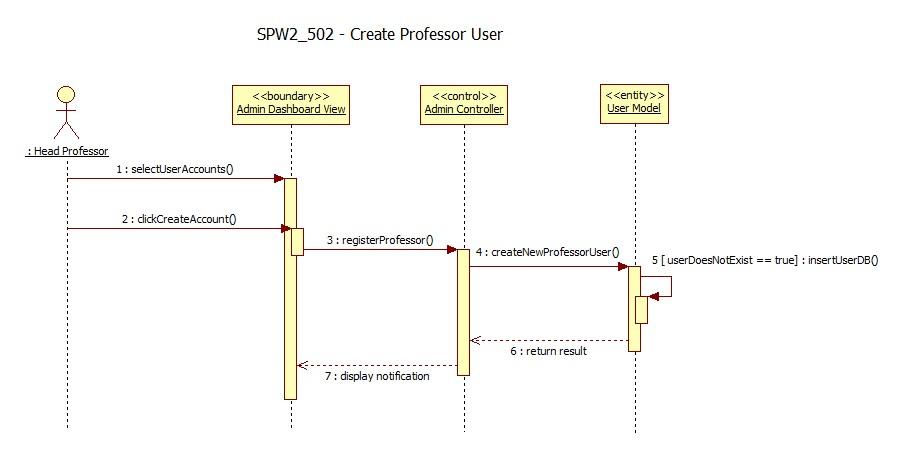












Upload File (SPW3\_710)



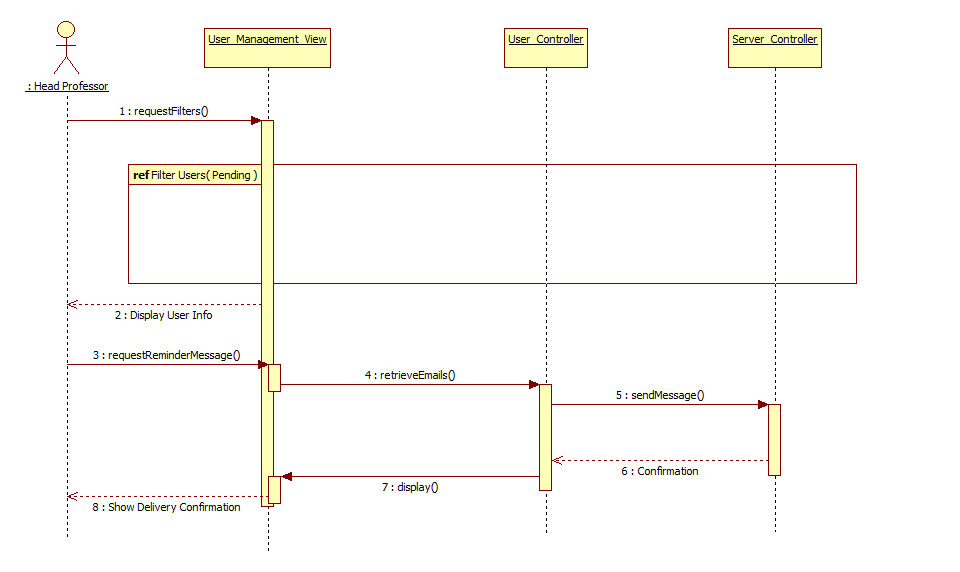
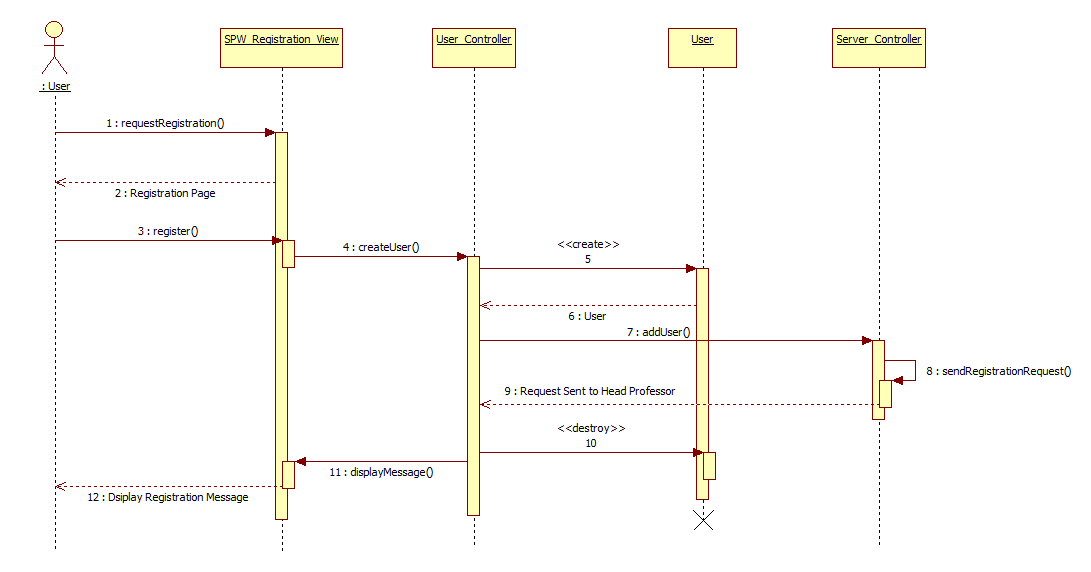
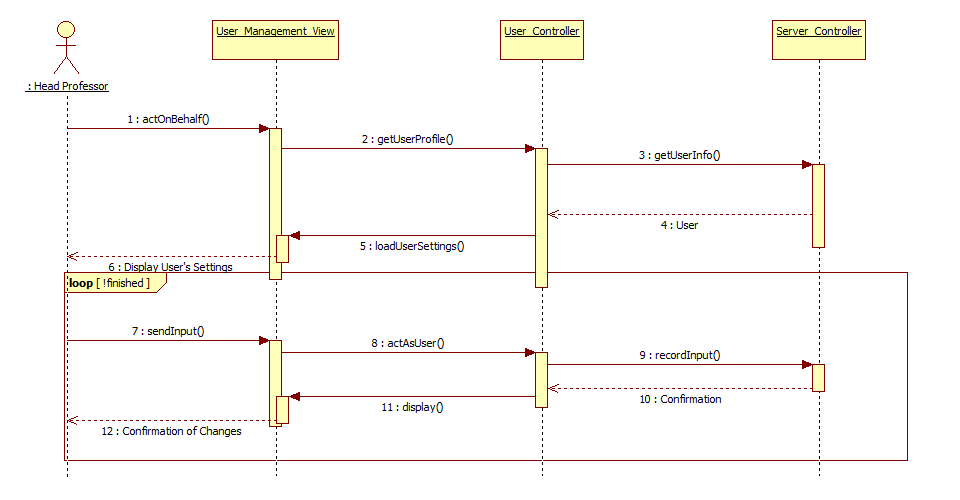
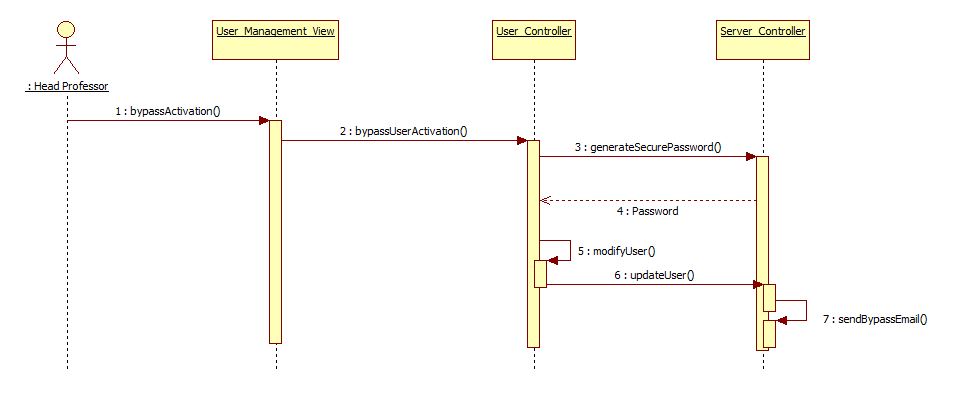
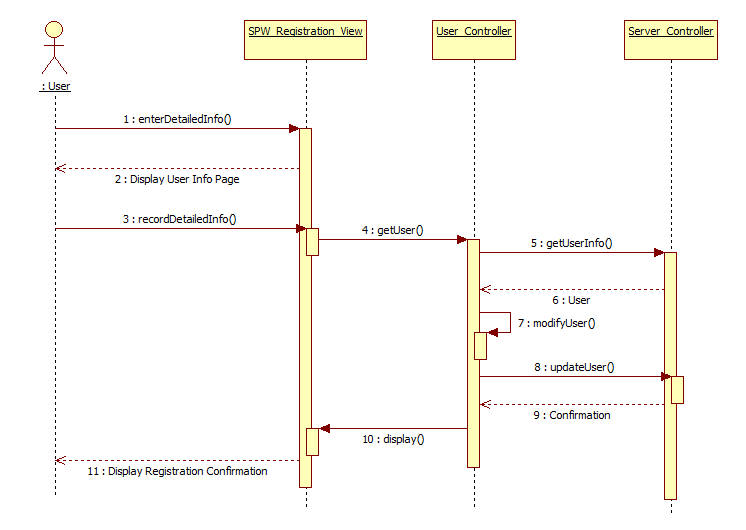
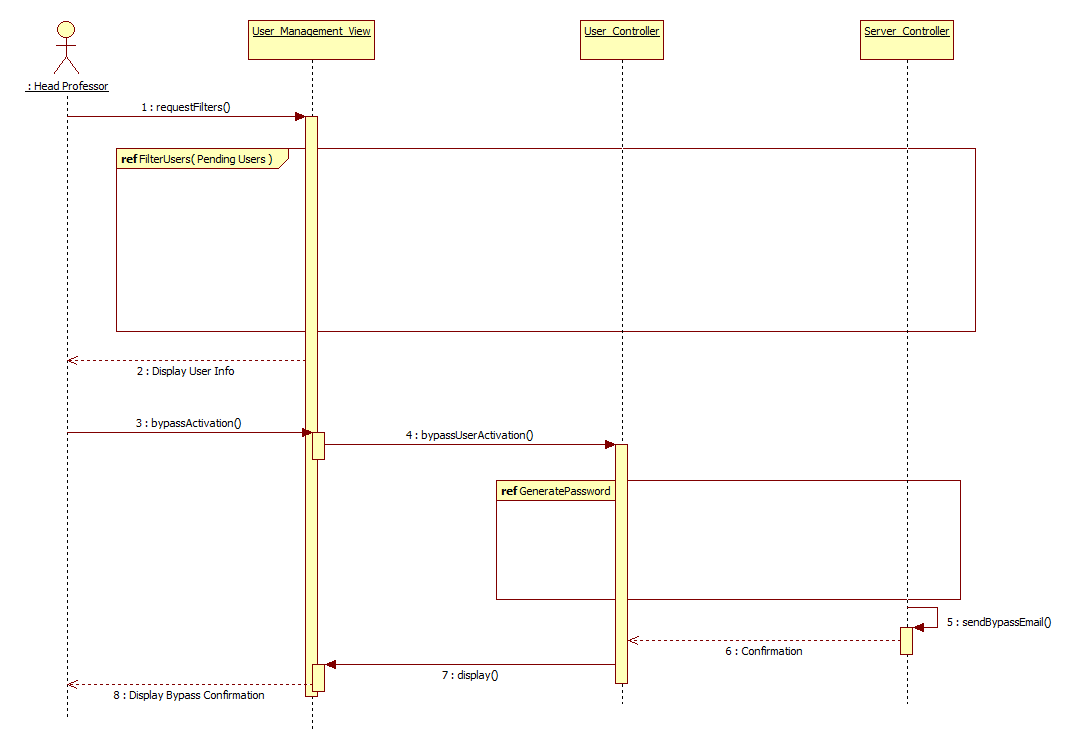
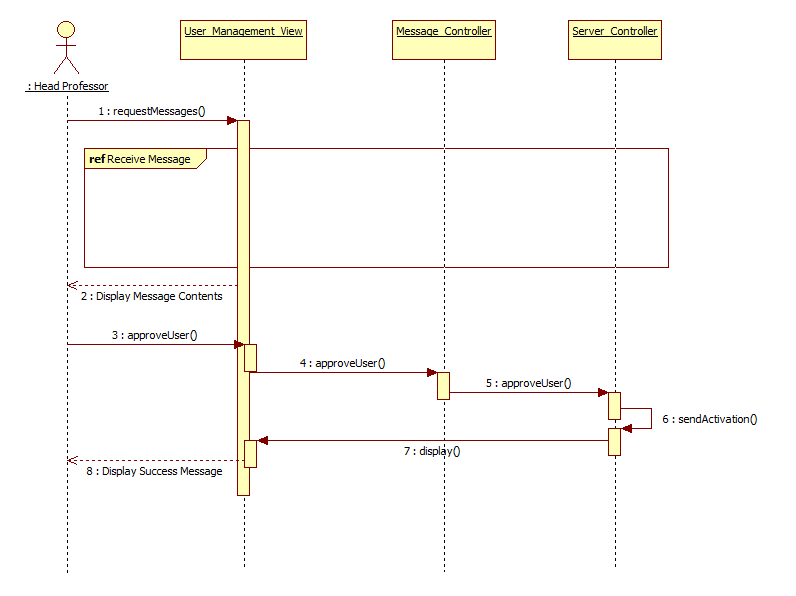
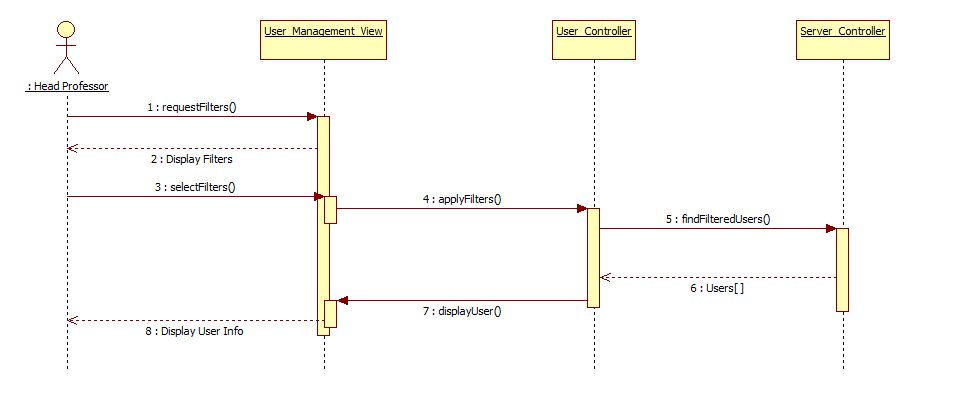
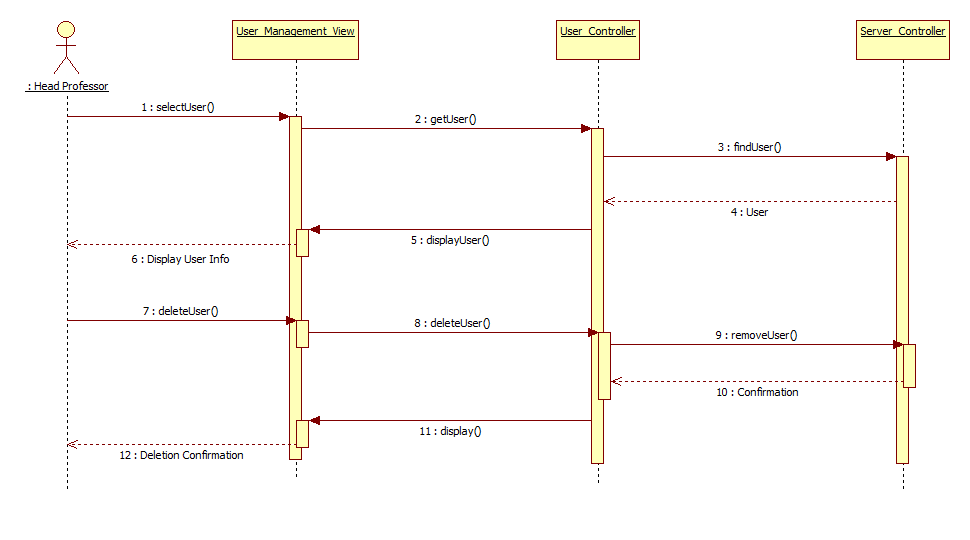
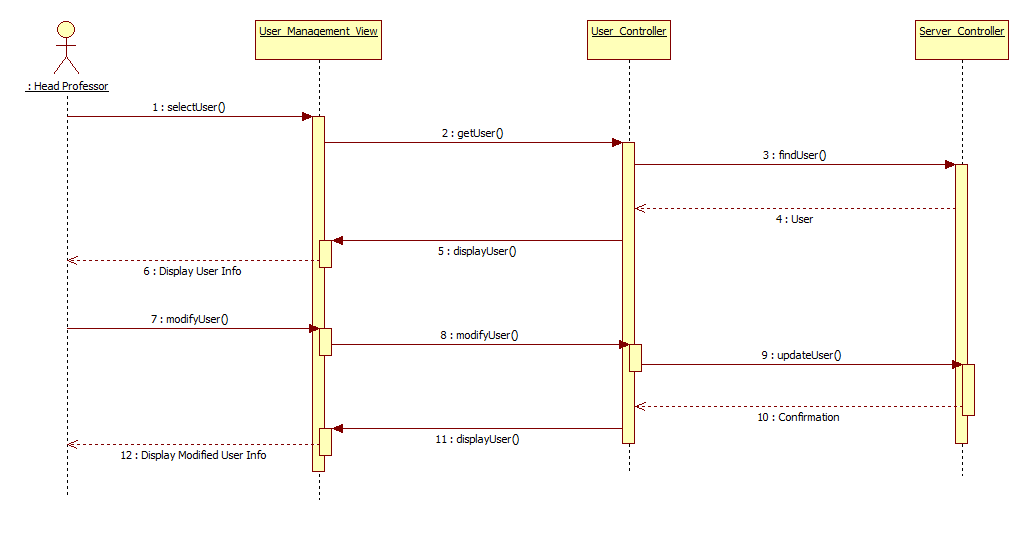
Download File (SPW3\_720)

Delete File (SPW3\_730) 

Add New Milestone (SPW3\_901) 

Edit Milestone (SPW3\_902) 

Delete Milestone (SPW3\_903) 



Update Minimum (SPW3\_210)



**9.6 Appendix F – Documented Class Interfaces**

**Project Controller**:

Declaration: class ProjectController extends CI\_Controller

public function \_\_construct()

{

parent::\_\_construct();

$this->load->helper('project\_summary\_view\_model');

$this->load->helper('request');

$this->load->helper('flash\_message');

load\_project\_summary\_models($this);

$this->load->model('SPW\_Project\_Details\_View\_Model');

$this->load->model('spw\_notification\_model');

$this->load->model('spw\_vm\_request\_model');

$this->load->model('spw\_user\_model');

}

Methods:

public function deleteVMRequest($request\_id,$image,$f\_ram,$storage,$f\_qty,$status,$name,$term)

public function filterVMRequests($image,$f\_ram,$storage,$f\_qty,$status,$name,$term)

public function vm\_requests()

public function loadEditImage()

public function editImage()

public function deleteImage($delete\_image\_name, $image, $status)

public function changeImageStatus($image\_name, $change\_status, $image, $status)

public function filterImages($image, $status)

public function vm\_images()

public function addImages()

public function vm\_request()

**Virtual Machine Model**

*Declaration*: class SPW\_vm\_request\_Model extends CI\_Model

*Constructor:*

public function \_\_construct()

{

parent::\_\_construct();

}

*Methods:*

public function searchFilteredVms($where)

public function getVMRequests()

public function getUserRequests($user\_id)

public function deleteVMRequest($request\_id)

public function insertVmRequests($requests,$user\_id)

public function updateRequestsFromProject($requests)

public function getHeadEmail()

public function getProjectTitle($project\_id)

public function searchFilteredImages($where)

public function getVMDefaultEmailCreation()

public function getAllImages()

public function addImage($image)

public function updateImageStatus($status,$image)

public function deleteImage($image)

public function editImage($old\_image, $new\_image)

public function updateImageRequests($requests)

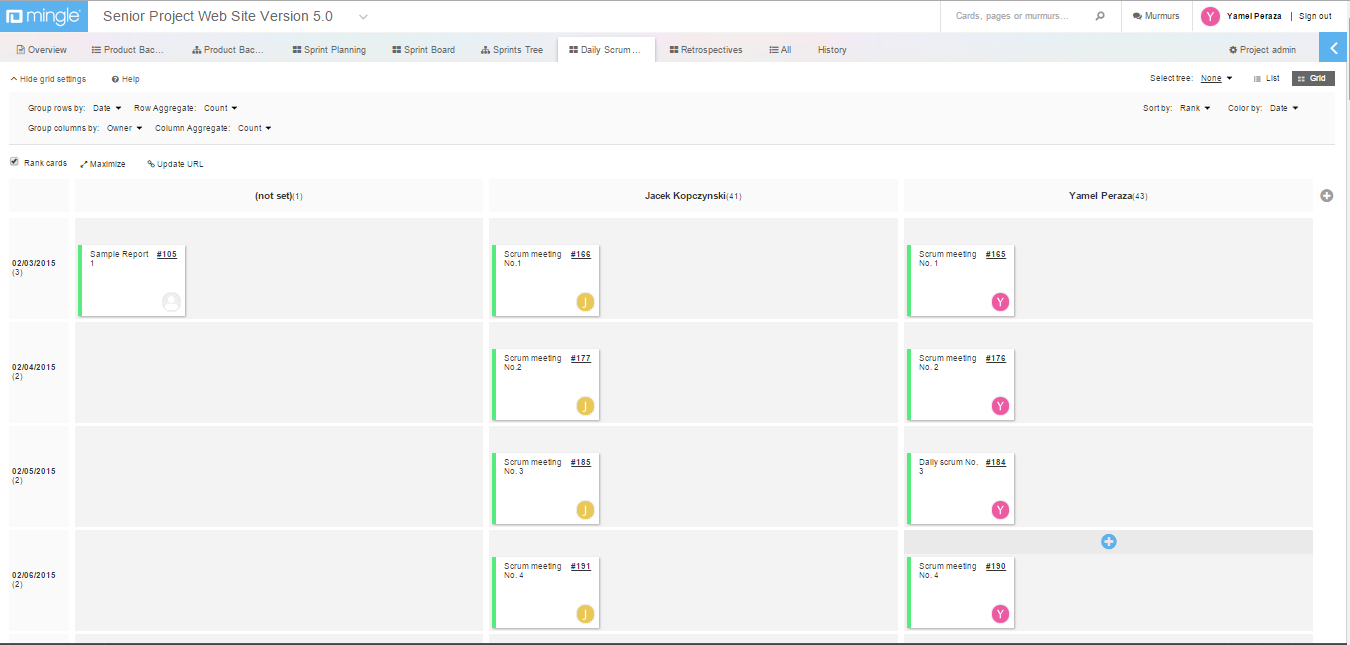
public function getActiveImages()

* 1. Appendix G – Documented Code for Test Driven and Stubs

The system was tested using Selenium, which is primarily used for testing purposes in automating web applications. The Selenium IDE was installed into the web browser Mozilla Firefox and test scripts were created to simulate user’s input.

**9.8 Appendix H – Diary of Meetings**

Developer members of the team met on a daily basis. Below is a screenshot of our daily meeting using Mingle. Meeting with product owner and lead developer occurred weekly resulting in the proposed implementation generally described in this document.



1. References

*Senior Project Website V5*



Julio Perez. "*Design Document for Senior Project Website: Version 4*" (Summer 2014). Florida International University.

Julio Perez. "*Requirements Documents for Senior Project Website: Version 4*" (Summer 2014). Florida International University.

Julio Perez. "*Feasibility Study & Project Plan for Senior Project Website: Version 3*" (Spring 2014). Florida International University.

Christopher Kerrutt, William Marquez, Cynthia Tope. "*Design Document for Senior Project Website: Version 3*" (Spring 2014). Florida International University.

National Resident Matching Program “The Match” Retrieved January 25th 2014, from http://www.nrmp.org/match-process/match-algorithm/

Arbelaez, A., Hamadi, Y., & Sebag, M. (n.d.). Online Heuristic Selection in Constraint Programming. Retrieved July 07, 2014, from research microsoft: http://research.microsoft.com/en-us/people/youssefh/search-socs.pdf

Christopher Kerrutt, William Marquez, Cynthia Tope. "*Requirements Documents for Senior Project Website: Version 3*" (Spring 2014). Florida International University.

Christopher Kerrutt, William Marquez, Cynthia Tope. "*Feasibility Study & Project Plan for Senior Project Website: Version 3*" (Spring 2014). Florida International University.

Fernandez, Yaneli, Camilo Sanchez, and Keiser Moya. "*Feasibility Study & Project Plan for Senior Project Website: Version 1.*" (Spring 2013). Florida International University.