**Senior Project Website**

**Version 5**

**Requirement Analysis Document**

CIS 4911 Senior Project

Section U01

Professor Masoud Sadjadi

**Prepared by**

Jacek Kopczynski

Yamel Peraza

Date

# 

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.*



Abstract

*The purpose of this document is to analyze the requirements of the software in developments. Requirements and clearly described as actions one does to interact with the website. The interaction is greatly explored and elaborated on via UML models describing the actions and process of the actions in great details, as well as non-functional requirements tied to them. This analysis will lend itself to the design document wherein the analysis is further fleshed out to a design of a solution.*

*IMPORTANT: When applicable the work of this version, SPWv.5 will appear first in a given section followed by a tag saying “From SPWv.4:” and data from the SPWv.3 documents which was not superseded and changed by SPWv.5. This is done to aggregate data across prior version documentation for future reader. Feel free to skip such tags until the next section.*

**Table of Contents**

*Senior Project Website V4*



[Copyright 2](#_Toc393726747)

[Abstract 3](#_Toc393726748)

[1. Introduction 5](#_Toc393726749)

[1.1. Problem Definition 5](#_Toc393726750)

[1.2. Scope of System 7](#_Toc393726751)

[1.3. Definitions, Acronyms, and Abbreviations 7](#_Toc393726752)

[1.4. Overview of Document 9](#_Toc393726762)

[2. Current System 10](#_Toc393726764)

[3. Project Plan 11](#_Toc393726766)

[3.1. Project Organization 11](#_Toc393726767)

[3.2. Identification of Tasks, Milestones and Deliverables 12](#_Toc393726768)

[3.3. Cost Estimate 12](#_Toc393726769)

[4. Proposed System Requirements 13](#_Toc393726770)

[4.1. Functional Requirements 13](#_Toc393726771)

[4.2. Analysis of System Requirements 19](#_Toc393726772)

[4.2.1. Scenarios 19](#_Toc393726773)

[4.2.2. Use Case Model 21](#_Toc393726774)

[4.2.3. Static Model 21](#_Toc393726776)

[4.2.4. Dynamic Model 22](#_Toc393726778)

[5. Appendix 24](#_Toc393726786)

[5.1 Appendix A – Use Cases 24](#_Toc393726787)

[5.2 Appendix B – Use Case Diagrams 62](#_Toc393726788)

[5.3 Appendix C – Static UML Diagram 67](#_Toc393726789)

[5.4 Appendix D – Dynamic UML Diagrams 72](#_Toc393726790)

[5.5 Appendix E – User Interface Designs 101](#_Toc393726791)

[5.6 Appendix F – Diary of Meetings 110](#_Toc393726792)

[6. Glossary 111](#_Toc393726793)

[7. References 113](#_Toc393726803)

1. Introduction



This section introduces the “Senior Project Website: Version 5” pertinent document information. First a problem definition is given regarding the problem statement being addressed in this version. The system scope will be expounded upon with regards to how the scope this version will reach. Pertinent particular definitions, acronyms, and abbreviations used in this document will be defined here as well. Finally an overview of the entire document will be included for ease of navigability.

* 1. Problem Definition

For the Senior Project class students are required to join projects that are of interest to them making up a team for a project whose positions are of variable length. The process of creating such a team of course carries many constraints and fulfillment of those constraints while optimizing the speed of generating the team is the primary focus of this version of the Senior Project Website. Thus the primary focus of this version is improving the websites matchmaking capabilities.

**Constraint Fulfillment**

The generation of a team for a project carries many constraints requiring consideration. Those constraints are threefold:

1. The *head professor*, primary manager of the website and course, wants certain projects optimized with regards to having a team that fulfills all of a projects requirements, having students who have the most project-relevant skills for the project, and having students’ skills efficiently distributed whenever possible
2. The *project proposer*, the proposer of a project, can be the head professor, the person who proposed a certain project wants the same thing the head professor wants but only for their project. Can also be seen as “what the project wants”
   1. Scope of System

The objective of SPWv.5 is to make the matchmaking system more useful for the head professor overall. Its scope within the system therefore will be only to the matchmaking component the head professor interacts with. Minor changes to the ranking students do to projects as well with regards to instructions and verification.

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. 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

**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 SPWv.4 project goals.

**Section 2** Discusses the current system primarily as a base for comparison with what is discussed in this document.

**Section 3** Covers the project plan and organization particularly in relation to this document.

**Section 4** Is the main figurehead of the document wherein discussion of the requirements as well as analysis of them is greatly explore, furthermore explored via UML diagrams.

**Section 5** Contains the actual UML diagrams and requirement analysis detail discussed in section 4.

**Section 6** Has the glossary of various terms in this document

**Section 7** Has references great amount of details remain the same as from SPWv.3.

1. Current System

*Senior Project Website V4*



The current system requires a big update with regards to the matchmaking process of students to projects. The prior implementation is very slow falling into the trap of combinatorial explosion described in section 1.1 “Speed Optimization of Matchmaking”. It is slow to the point of never returning given a “fair” size of data, as in 8 projects and 24 students, instead just timing out.

The system also tries to fulfill the constraints of the matchmaking system by producing 10 different matches which considers the constraints outlined in section 1.1 as one score resulting in something not completely discrete in regards to function. More importantly matchmaking cannot run in an imperfect environment such a condition is rather unreasonable as it is not prudent for the head professor to discount a project because no students have one skill one project needs. A perfect matching after all is not always possible, nor reasonable to coerce.

The result page of the matchmaking process is also lacking in usefulness. The display lacks useful information and it automatically updates the database without the head professor doing anything other than ranking projects by their opinion of importance. Such disconnect between the head professor and his executive duty in matching students to projects makes the usefulness of the current system lacking.

1. Project Plan

*Senior Project Website V4*



This section covers project organization, task breakdown, and cost estimate relevant information regarding the project.

* 1. Project Organization

Roles of team are described in Table 3.1 in regards to the project role. The further specific roles are expanded on table 3.2. This pertains to this document specifically but no change has occurred since the feasibility document. The team is still one man developer based.

|  |  |
| --- | --- |
| Name | Role |
| Julio Perez | *Team Leader:* 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

|  |  |
| --- | --- |
|  | Julio Perez |
| 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. Identification of Tasks, Milestones and Deliverables

Task and milestones have essentially for this semester occurred on a week by week basis. The end result being the matchmaking system outlined in this document proposal. Task essentially boiled down to week by week tasks and debugging with a general milestone of achieving the overarching goal as well as defining the goal as weeks go on. Deliverables are to all be turned in at the end. Deliverable relevant content also is to be gained during the weeks. This style of software development is based on agile methodologies.

* 1. Cost Estimate

Cost estimate remains the same as from feasibility document appendix C.

|  |  |  |
| --- | --- | --- |
| 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 V4*



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

* 1. Functional Requirements

**These following use cases pertain to this version. 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.

SPW\_107:

A student is told that SPWv5. has a new feature that allows student to create and submit customizable vms and that feature is located in VM – Request page, but will not be available to students until choose a project deadline is over. Since the student has access to the website he wants to look for the page manually and adds to the url the words VM- Request, the system automatically checks if choose a project deadline is over and prompts the student with a warning message saying that he is not authorized to access the VM – Request page. Frustrated, the student waits for choose a project deadline to be over.

Finally, deadline to choose a project is over, and system shows Create VM-Request button in My Project page, allowing student to navigate to VM – Request page and create a vm request. Once the student is in VM – Request page they are allow to customize their vms settings by selecting the name of the image, RAM, Storage and the number of vms needed. If additional vms are needed with different settings, the student is able to add a new request by clicking on Add Another Request button and the system adds a new table row with new settings to select from. Once the student finished selecting the vms, he clicks on Submit button and the system processes the vm request.

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.

**SPW\_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.

**SPW\_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)

**SPW\_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 used case model describes the basic functionality to running a match. There are two actors the student and professor. Both of them can rank projects, the ranking which will take place during the matchmaking process. The professor can set the minimum number of projects students need to rank else they become “wildcard” meaning they are eligible for any projects. Head professor also sets the deadline for the ranking and also runs the match.

* + 1. Static Model

The static model remains largely unchanged since SPWv.3. The main difference is in the matchmaking controller, model, and views. The controller has many more methods added to it to account for the complex interaction occurring in matchmaking now. The model has one new method now to update the database to mark a student as a member of a project; this method was adapted from a very similar one in another model. The view now has many new views to account for the different pages the head professor now visits on the way for a final match result. The views there also have lines of association and from top to bottom declare an implicit order of visitation. I.E. match page goes to match start page or project priority page, match start page goes to match phase 1 auto or match phase 1 manual progress, etc.

* + 1. Dynamic Model

The dynamic model starts off by showing very complex interactions for how to “run a match”. But if carefully followed one can see the process. Essentially once the head professor has clicked run match after clicking the match button, they now choose to run VIP manually or automatically from the match start page.

The controller then takes a Boolean from the prior action (auto). It prepares the projects and students from the match model which holds the database data relevant to matchmaking. Now from auto the sequence splits into two paths which reconverges later.

On path one where auto is false, the manual VIP matching occurs (see paragraph 3 for if auto is true). The controller processes data, returns the match phase 1 manual progress view, and the head professor selects students from the ones that are forced versus the ones who want to be in the projects. The view takes the form data and the controller processes the next VIP, and repeats the process for each VIP. This ends when every VIP project is done see paragraph 4 for more information on this or it ends if the head professor chooses to “do the rest automatic” then the controller detects that and runs the remaining VIP automatically see paragraph 3 for this.

Now if auto is true or if during the manual progress the head professors choose to “run the rest automatic” then all the projects one by one are given optimal teams (on the two categories discussed in paragraph 2) based on head professor (HP) rating’s order. This is displayed as match phase 1 auto for the HP. Then the head professor chooses one of two options the forced students or the students who wanted the projects.

Now VIP matching is ended once the HP chooses one of the choices in automatic or if manual mode went through every project. This convergence now makes the sequence diagram very sequential from here on out. Once VIP matching ends the HP is show the match phase 1 final page for confirmation. HP clicks continue and the controller now process the OP and match phase 2 page displays two options one where students were matched via NRMP using student interest as the only metric or one where students choose their projects but are displaced if a better student for the project is displacing them. The head professor chooses one.

The controller now sends all the data for the match finalize page and the HP chooses confirm which finally goes back to the controller who communicates with the matchmaking model to change the database which propagates through the website. The HP is redirected to the match page found from the header and is given a message to indicate success in matchmaking.

The other two used cases are very straightforward. Head professor changes his ranks, and clicks saved and then it is saved. A similar process for the students occurs just on a different page. The only really difference is validation process to ensure the scoring meets the new SPWv.4 standard.

1. Appendix



* 1. Appendix A – Use Cases











|  |  |
| --- | --- |
| ***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 selecting from drop down menus: the image, the amount of RAM, the storage and the number of VMs. 4. System displays in drop down menus: the available image names in the system, the amount of RAM, the Storage amount and the number of virtual machines to choose from. 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. |

|  |  |
| --- | --- |
| ***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 – Requests page. |
| ***Flow of Events*** | 1. Head professor modifies virtual machine request by selecting from drop down menus: the image name, the amount of memory RAM, the storage amount, the number of VMs and the status of the request. 2. System displays in drop down menus 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. |

|  |  |
| --- | --- |
| ***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 in VM – Requests 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 – Request page in Senior Project Website and loads all virtual machine request from the corresponding project with status equal to “PENDING”. |
| ***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 clicks on “Search Filtered Images” button. 4. System processes request and prompts the images according to the selected filter option. |
| ***Alternative Flows*** | * {From step 1} Head professor selects from the filter options the search criteria, but does not click on “Search Filtered Images” button and navigates into a different page. System consider action invalid and it is destroyed. |
| ***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\_)** | 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 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 images. |

|  |  |
| --- | --- |
| ***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 |

**Following are SPWv3 related use cases (not superseded by this version):**

|  |  |
| --- | --- |
| ***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* (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) 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. |
|  |  |

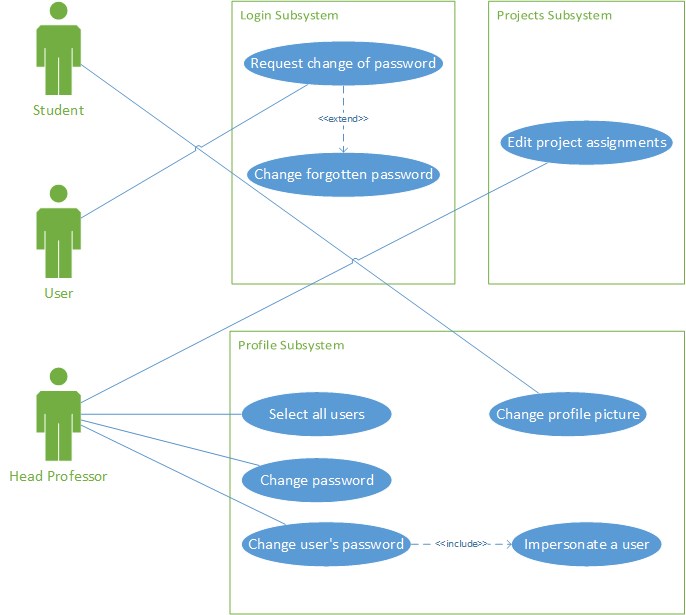
|  |  |
| --- | --- |
| Use Case Name | (SPW3) Filter Users |
| Related Use Cases | Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View All Users” 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. |
|  |  |

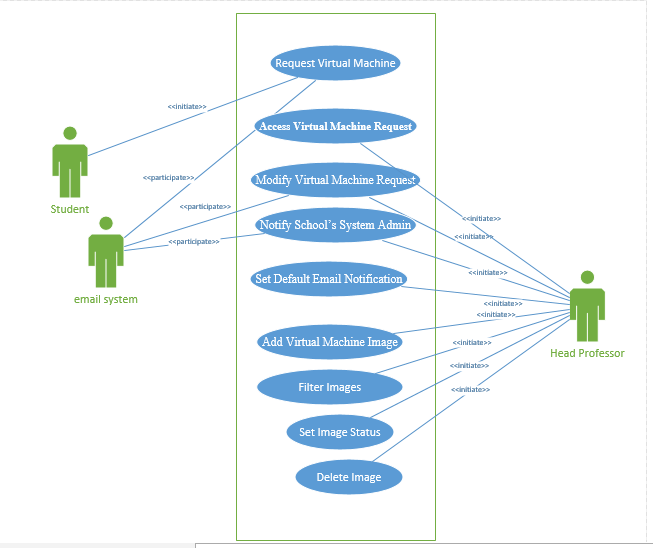
|  |  |
| --- | --- |
| Use Case Name | (SPW3) Bypass Activation |
| Related Use Cases | Filter Users |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “View 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. |
|  |  |

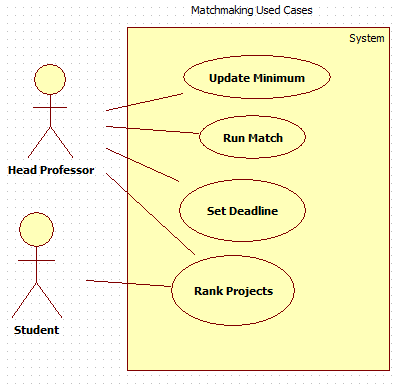
|  |  |
| --- | --- |
| Use Case Name | (SPW3) Impersonate User |
| Related Use Cases | Filter Users, Login |
| Participating Actor | Head Professor |
| Flow of Events | 1.      The Head Professor logs in to the SPW |
| 2.      The Head Professor clicks on the Admin tab on the menu bar and clicks the “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. |
|  |  |

* 1. Appendix B – Use Case Diagrams



**Below are Use Cases from SWPv.4 and SWPv.3 unchanged in this version:**

**  
Below are Use Cases from SWPv.3 unchanged in this version:**

Use cases related with Login, Logout, Guest Access and Security

Use cases related to the interaction between the Student users and the Projects



Use cases related to Projects



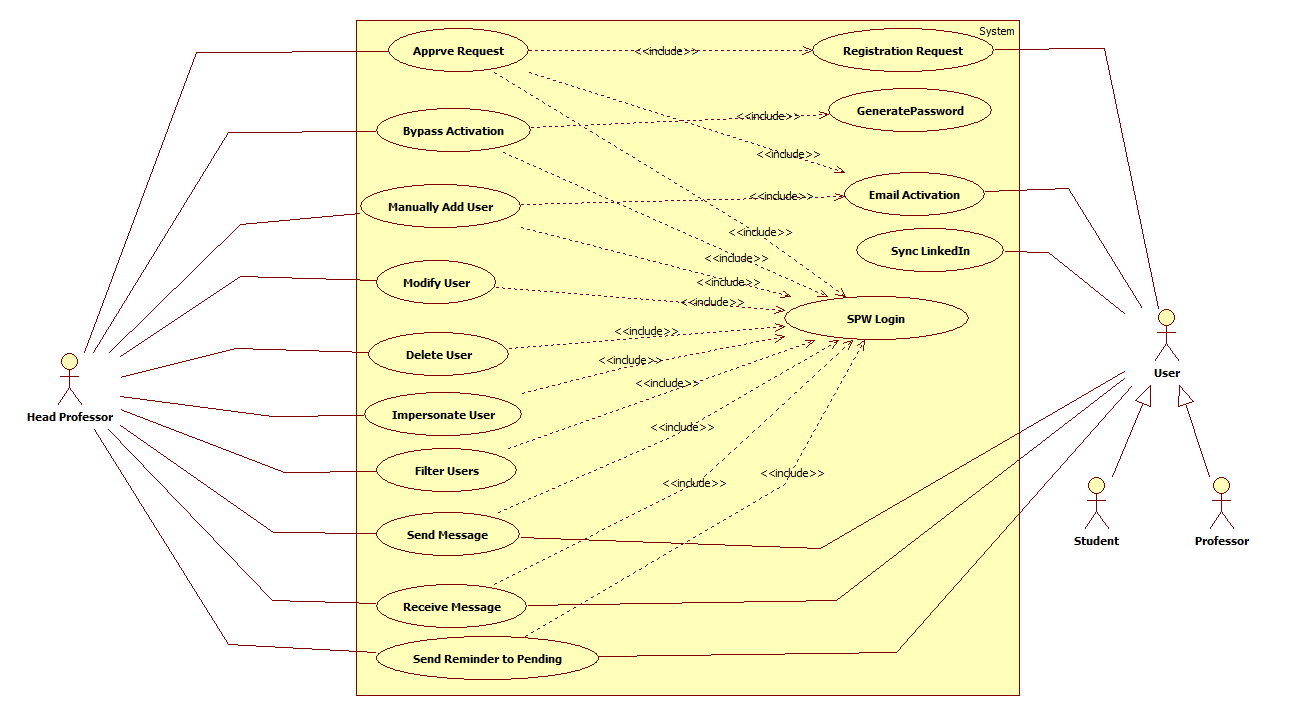
Use cases related to the Users, their Profiles and the Term Deadline



Use cases related to the Rest API



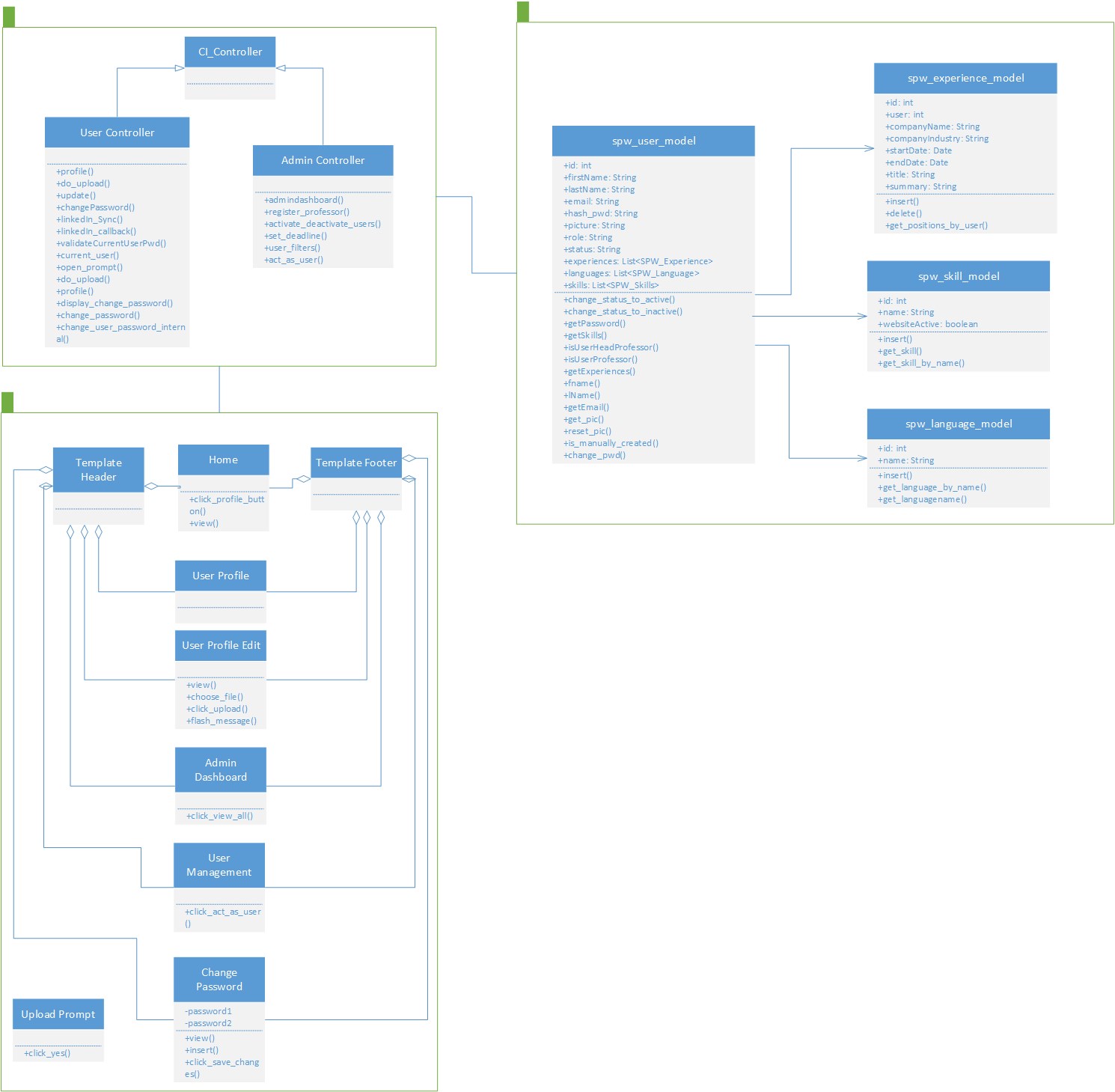
Use cases related to the Files Repository Subsystem



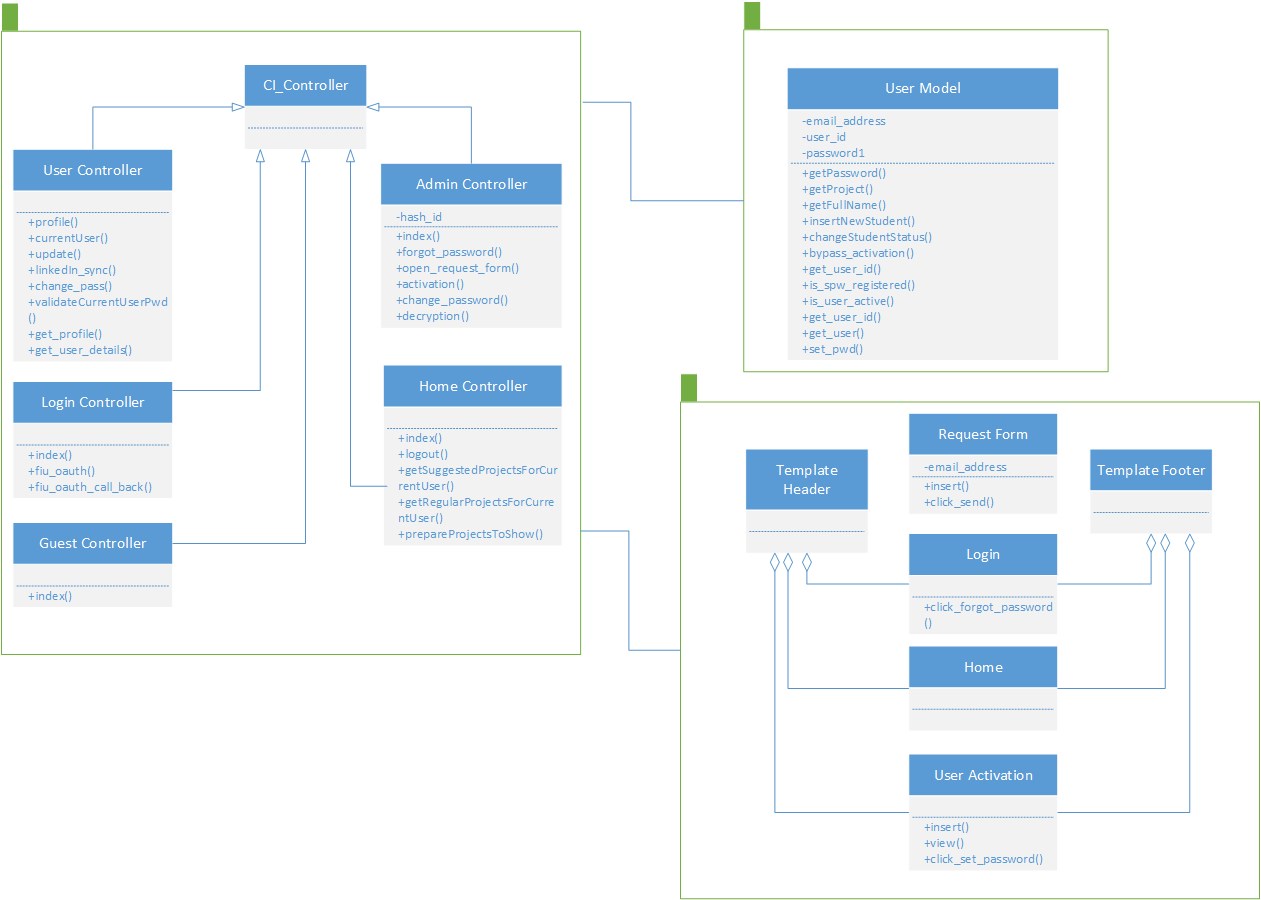
Use Cases for User Management

* 1. Appendix C – Static UML Diagram

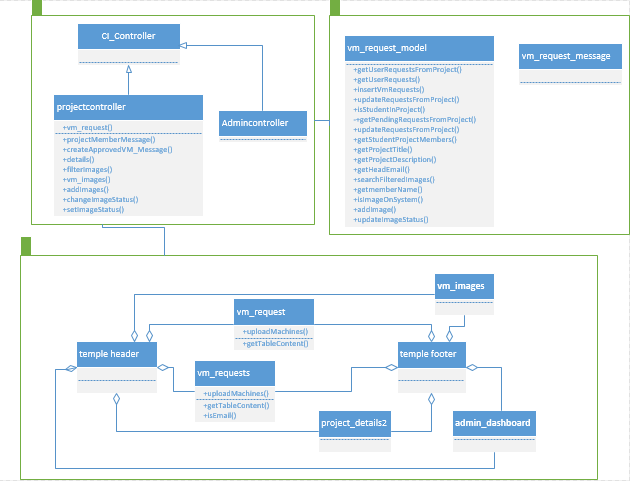
Profile Subsystem



Login Subsystem



Virtual Machine Subsystem



**Below are static diagrams from SPWv.4 documentation:**

Largely unchanged from SPWv.3. Any pertinent change noted below. Mainly just added views for matchmaking. Changed/added controller and model functions.







**Below are static diagrams from SPWv.3 documentation:**

Note more views for matchmakings were added and controller/model for matchmaking has more/different methods.

UML Detailed Class Diagrams for the SPWv.3

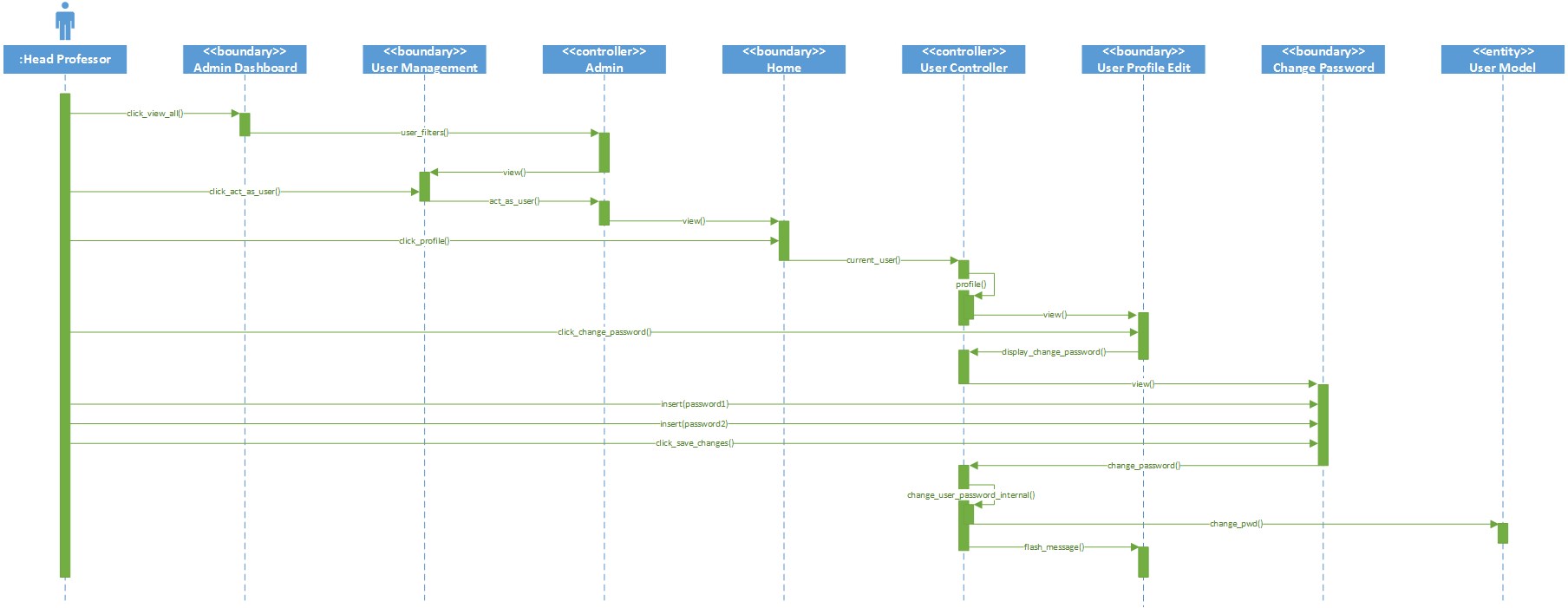




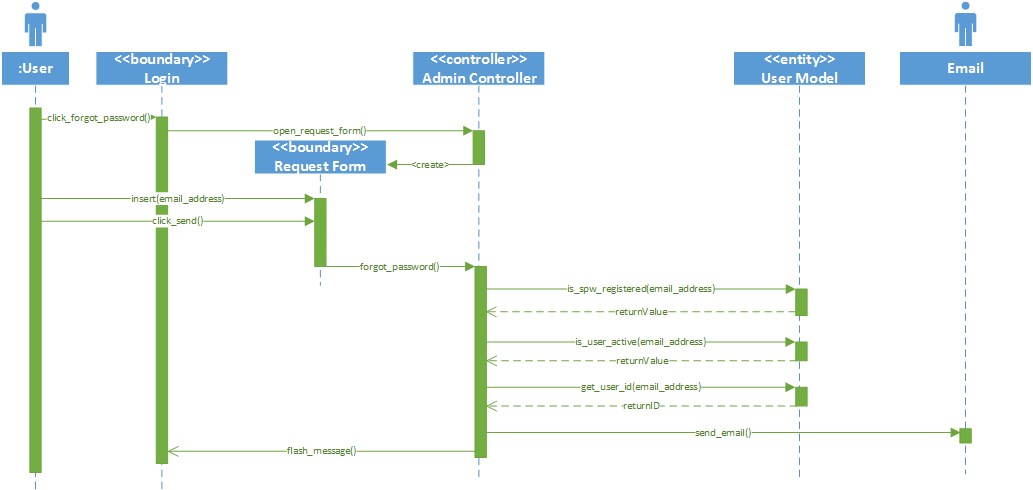


* 1. Appendix D – Dynamic UML Diagrams

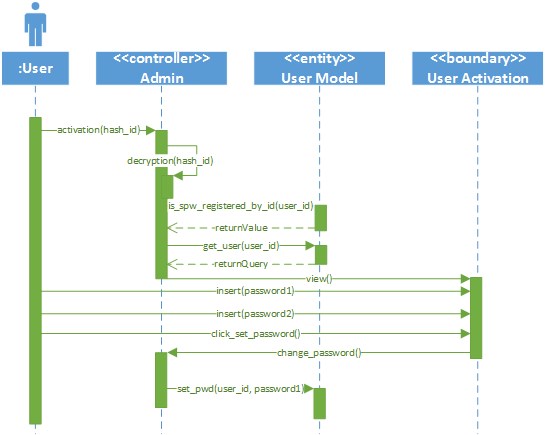
SPW5\_101 Change user’s password



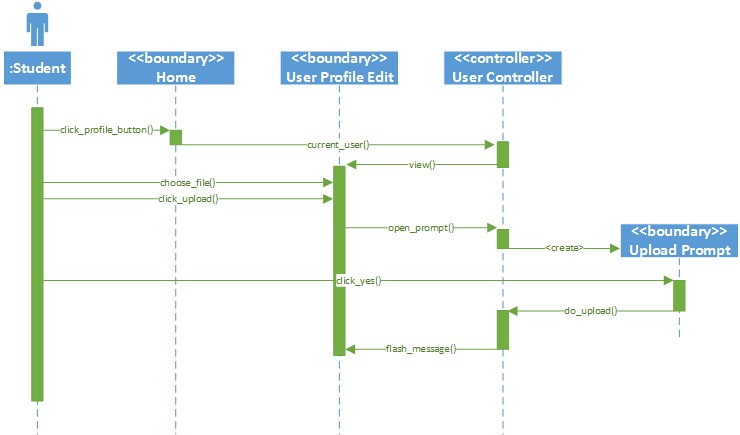
SPW5\_104 Request change of password

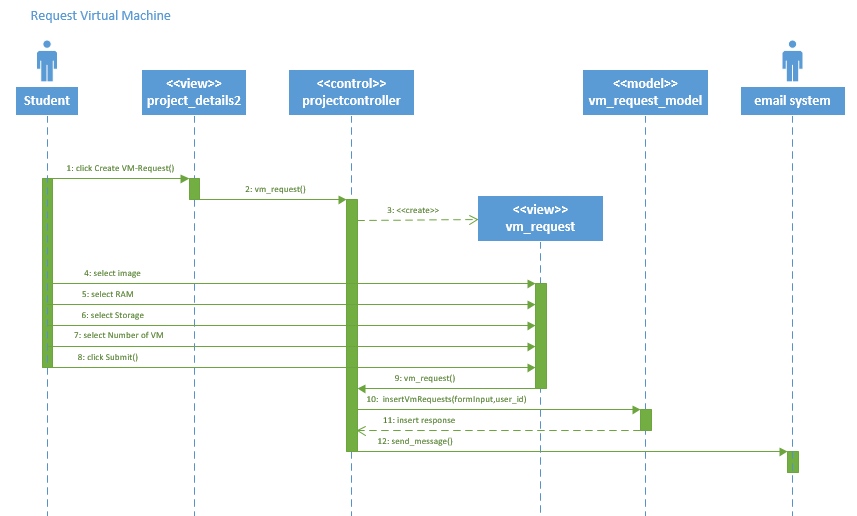


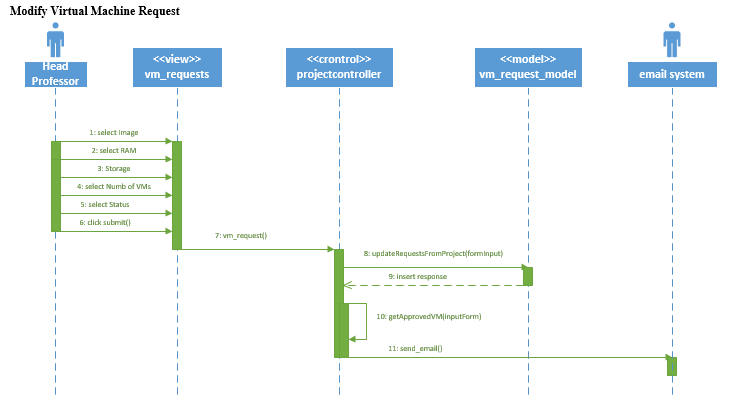
SPW5\_105 Change forgotten password

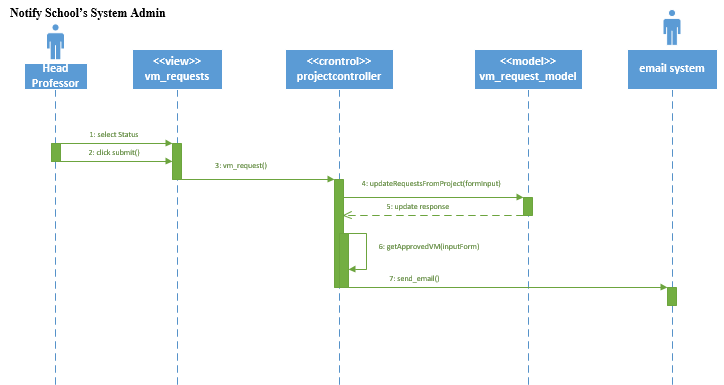


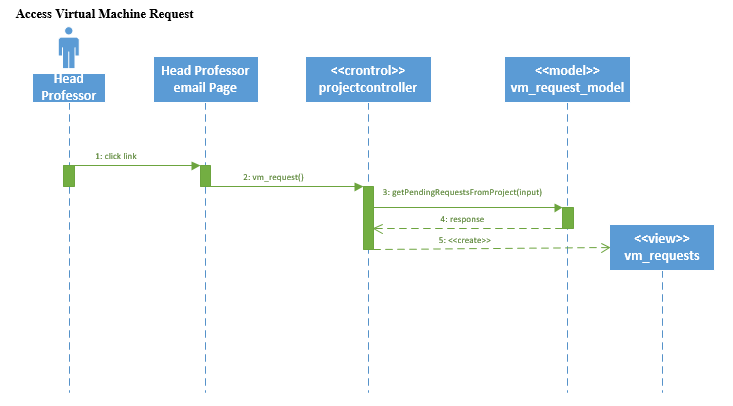
SPW5\_106 Change profile picture

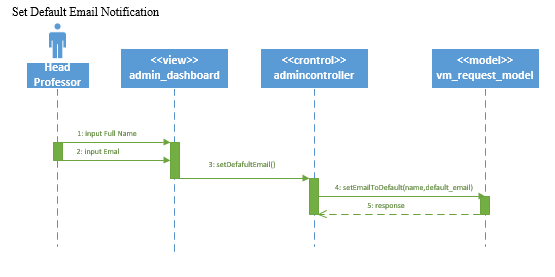


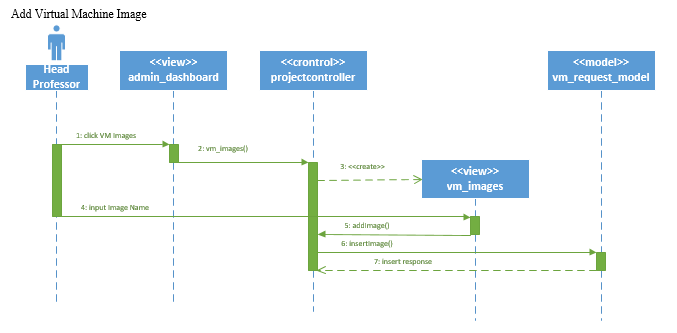


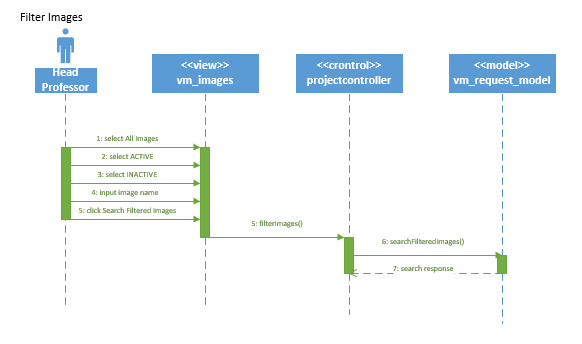


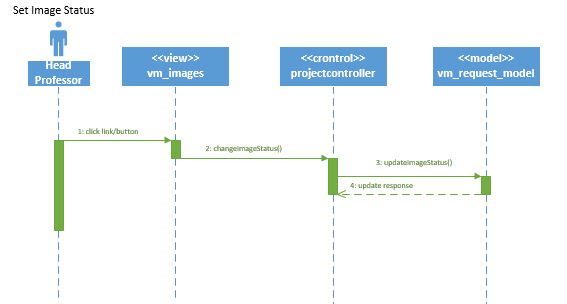


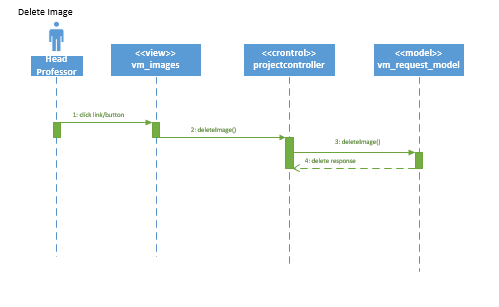






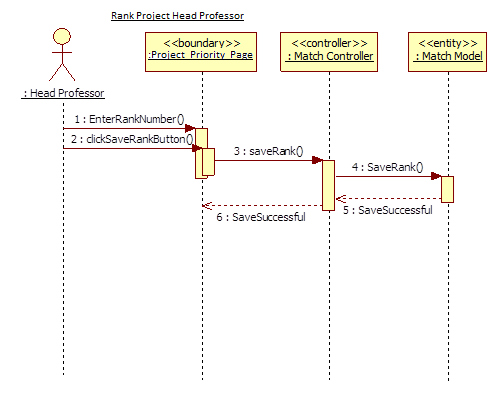








SPW4\_101 is the similar to SPW3\_205 from SPW3 document (changed only on boundary)



SPW4\_101 is the same as to SPW3\_205 (Student) from SPWv.3 document. Change is in validation



**Below are non-superseded sequence diagrams from SPWv.3 documentation:**

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) 

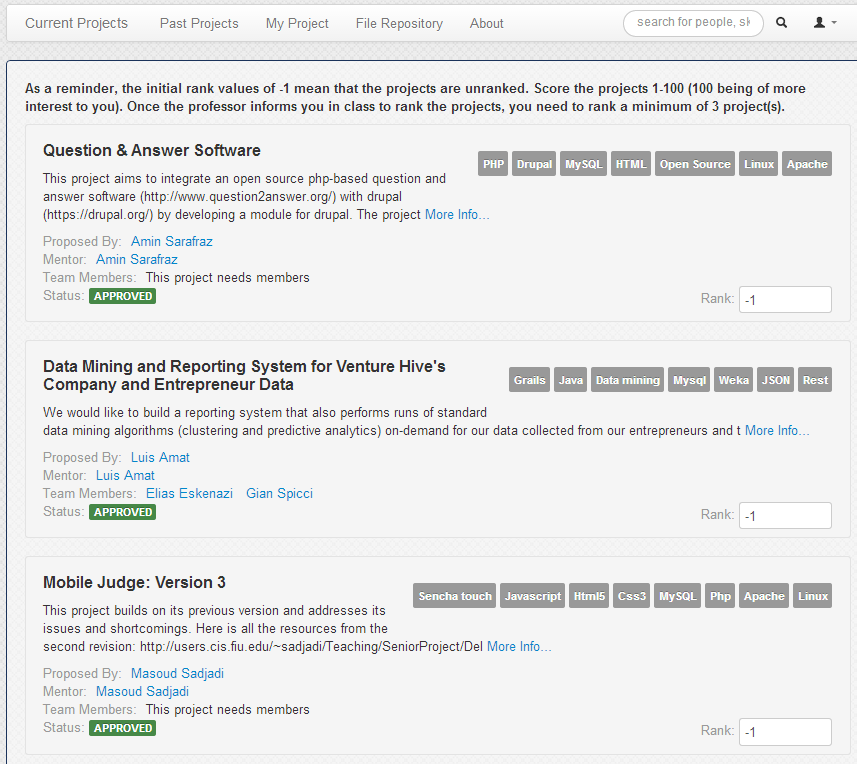


Update Minimum (SPW3\_210)

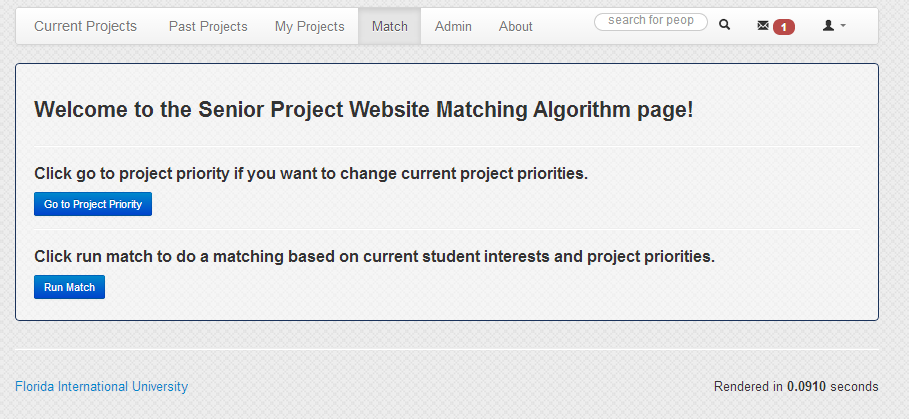


* 1. Appendix E – User Interface Designs

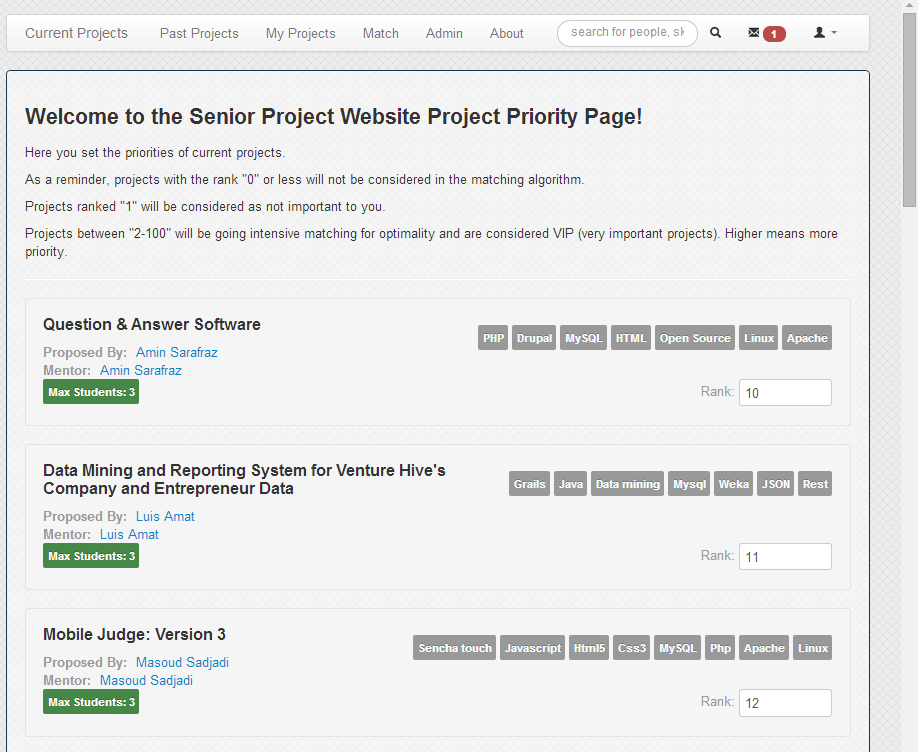
Student ranking view



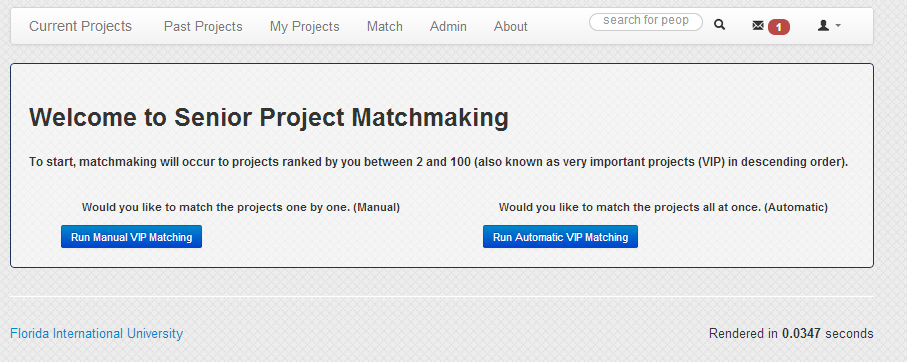
Match Page



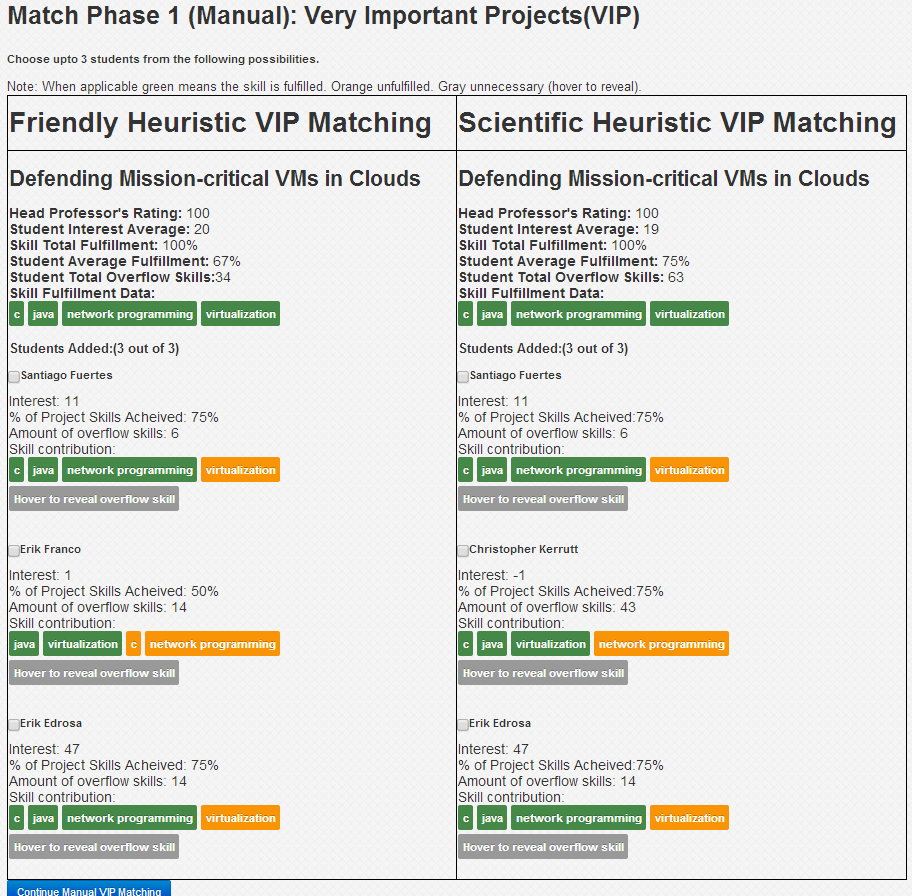
Professors Ranking Page



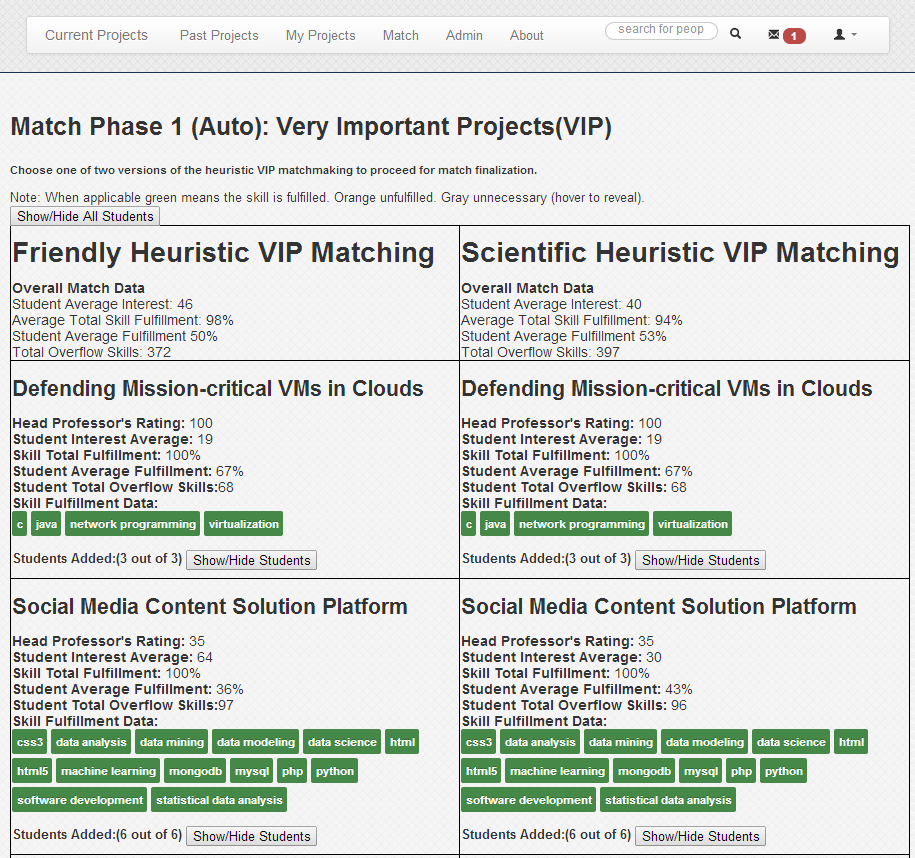
Run Match Page 1



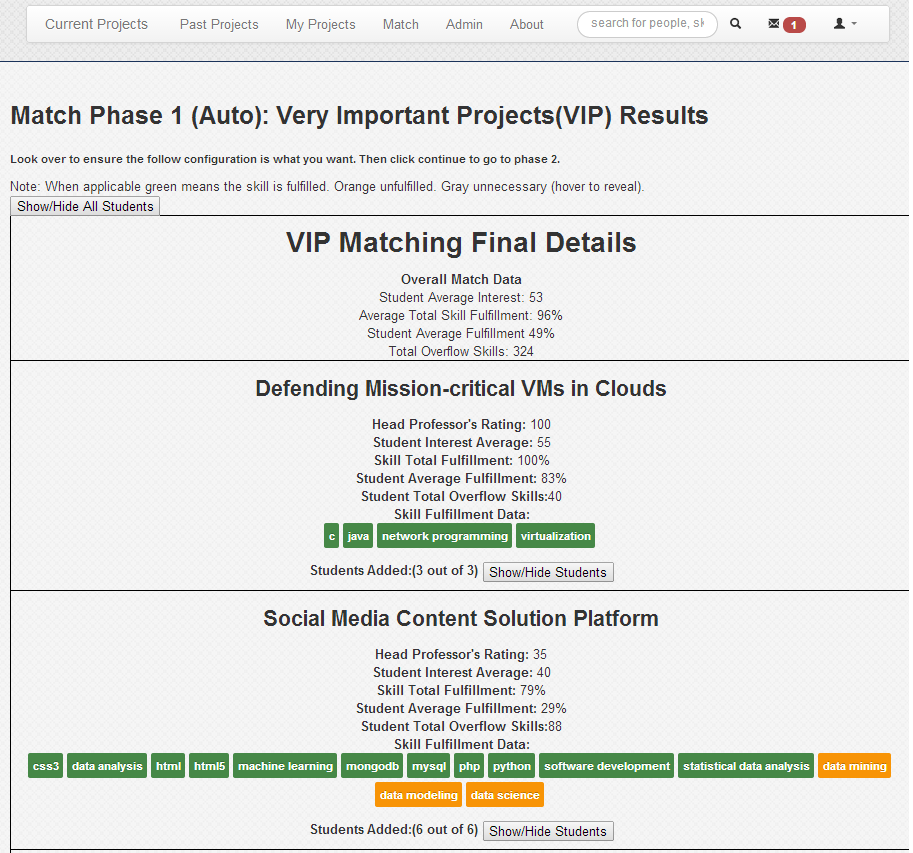
Run Match Page 2 Manual



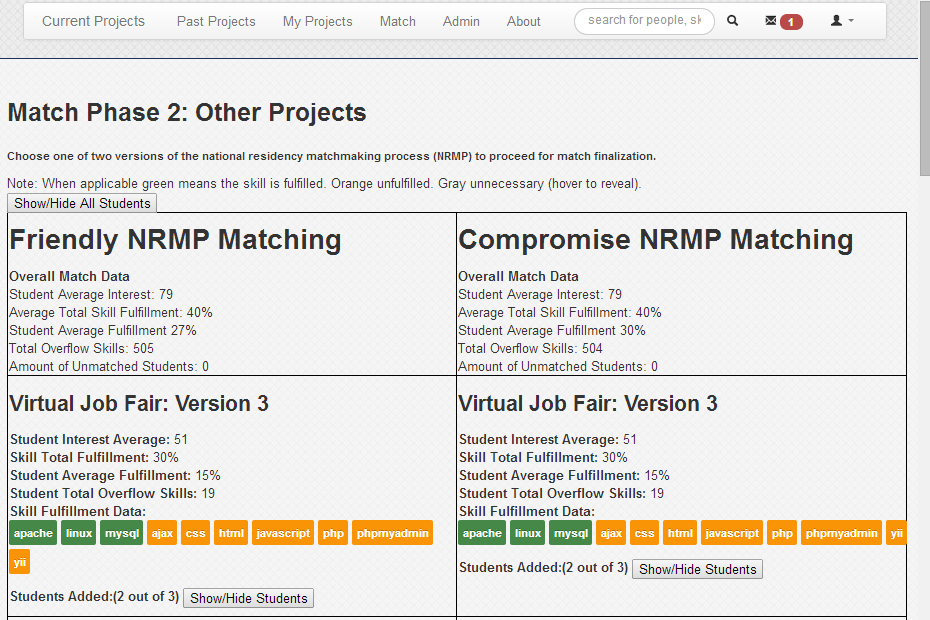
Run Match Page 2 Automatic



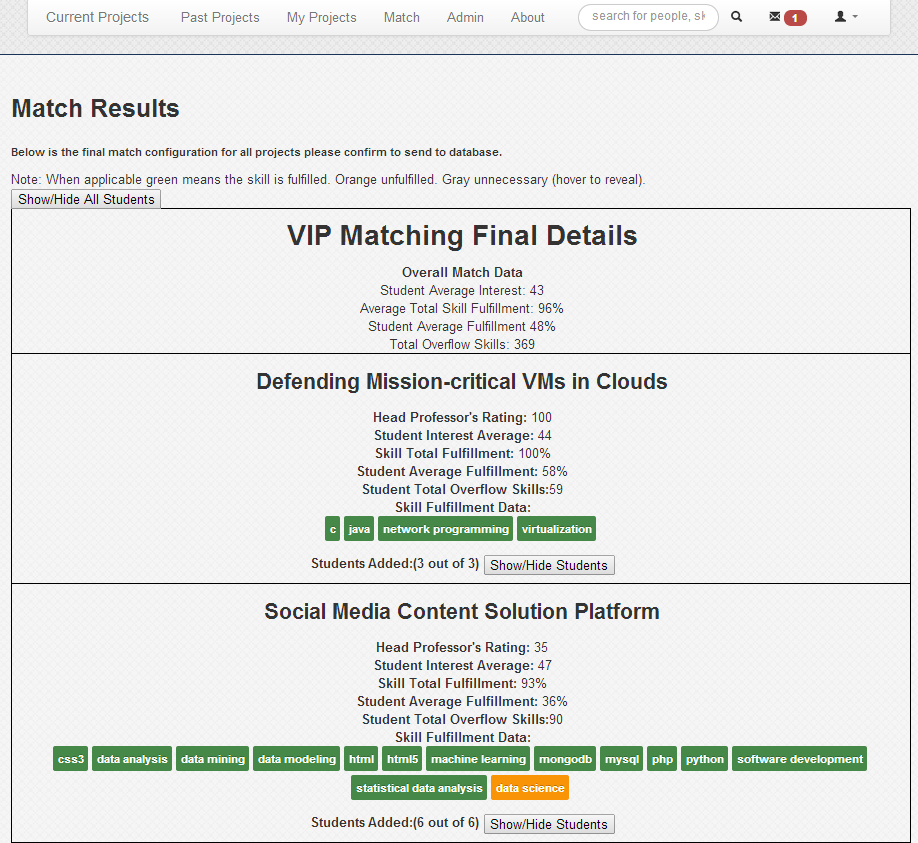
Run Match Page 3



Run Match Page 4

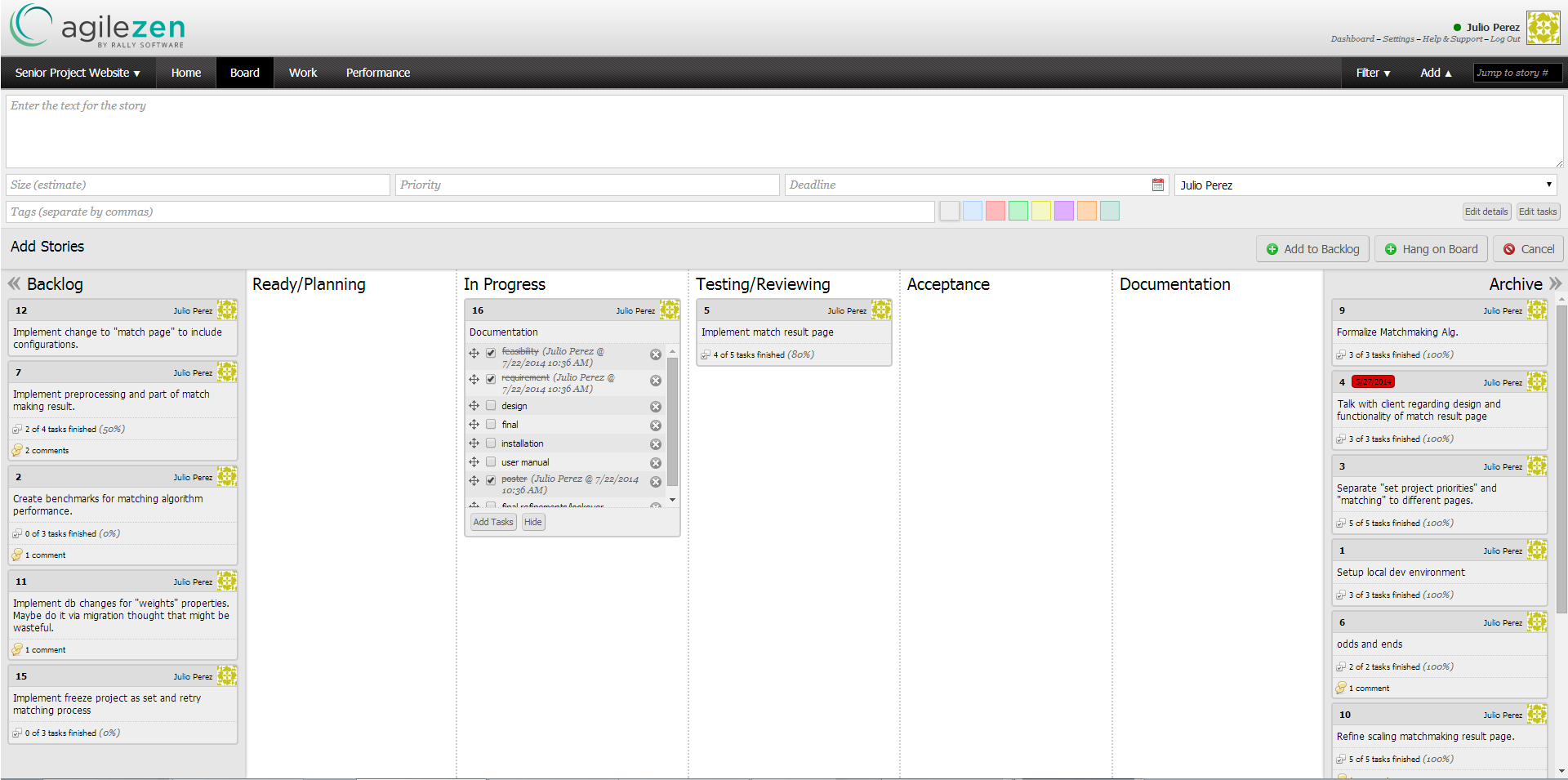


Run Match Page 5



* 1. Appendix F – Diary of Meetings

Given that the team consists of one person no official meeting occurred. Meeting with client and lead developer occurred weekly resulting in the proposed implementation generally described in this document. Tasks have also been on a week by week basis. This style is known as agile methodologies, Agile Zen a web application was used as a means to track these tasks on a regular basis, shown below.



1. Glossary

*Senior Project Website V4*



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

**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. References

*Senior Project Website V4*



Christopher Kerrutt, William Marquez, Cynthia Tope. "*Requirements Documents 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