**Senior Project Website**

**Version 5**

**Design Document**

CIS 4911 Senior Project

Section U01

Professor Masoud Sadjadi

**Prepared by**

Jacek Kopczynski

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

*This document covers the design of the solution to the requirements analyzed in the requirement analysis page. The design of the solution includes an introduction to the overall document and project. Overall system design such as hardware/software mapping. Detailed design of particular subsystems. And an appendix of related information.*

Table of Contents

*Senior Project Website V4*



[Copyright 2](#_Toc393972296)

[Abstract 3](#_Toc393972297)

[Table of Contents 4](#_Toc393972298)

[**1.** **Introduction** 5](#_Toc393972299)

[**1.1.** **Problem Definition** 5](#_Toc393972300)

[**1.2.** **Design Methodology** 6](#_Toc393972301)

[**1.3.** **Terminology – Definitions, acronyms, and abbreviations** 7](#_Toc393972302)

[**1.4.** **Overview of Document** 9](#_Toc393972312)

[2. System Design 10](#_Toc393972313)

[**2.1.** **Overview** 10](#_Toc393972314)

[**2.2.** **Subsystem Decomposition** 10](#_Toc393972315)

[**2.3.** **Hardware and Software Mapping** 14](#_Toc393972316)

[**2.4.** **Persistent Data Management** 14](#_Toc393972317)

[**2.5.** **Security/Privacy** 19](#_Toc393972318)

[3. Detailed Design 22](#_Toc393972319)

[4. Glossary 71](#_Toc393972343)

[5. Appendix 73](#_Toc393972353)

[**5.1 Appendix A Used Case Diagrams (SPWv.4)** 73](#_Toc393972354)

[**5.2. Appendix A Used Cases (SPWv.4)** 73](#_Toc393972356)

[**5.3 Appendix C Class Interfaces (SPWv.4)** 76](#_Toc393972357)

[**5.4 Appendix D Diary of Meetings and Tasks** 79](#_Toc393972406)

[6. References 80](#_Toc393972407)

1. **Introduction**

*Senior Project Website V4*



This section introduce the SPWv.4 product overall. Starting off by define the problem, then the design methodology used in implementing the product’s design, followed by various terms used throughout the document, and finally a brief overview over all chapters of this document.

* 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”
3. The *student*, the prospective position filler, wants to be assigned a particular project, or if they want, they can declare they do not care which project they are assigned and would like to be in any project. Hence one can say they would like to be assigned a project best fitting their skills in that case.

**Speed Optimization of Matchmaking**

The generation of a team when brute-forced can be seen as trying every combination of a team made from every student for every project. Such a phenomenon has been observed as a “combinatorial explosion” which one can surmise would lead to an incredible large number of computations in search of a “perfect team” for a project fulfilling constraints outlined in the previous part. Mitigation of such phenomena will be a big point of interest in this version of Senior Project Website.

**Head Professor Matchmaking Feedback**

As the coordinator of the course and website the head professor should have final executive decision in regards to what assignment of students to project will end up being. Giving better user interface, better flexibility to the matchmaking process, and some amount of control regarding how matchmaking progresses all would benefit how effectively they can carry out their duties.

* 1. **Design Methodology**

The SPWv.4 has followed a new software development model this semester different from prior ones. The development model this semester was based on agile methodologies, wherein small incremental development happened on a week by week basis to result in the product released in the end.

In the development process weekly meetings resulted in discussion of what was done since the prior week, discussion of what is the plan for the following week, and scaling of expectations for the final developmental increments. The benefit of this system lies in its freedom to deploy development on a regular basis for feedback and tweaks, rather than in the waterfall method where much analysis happened with a rush of development in one period of time, rather than spreading out the development process allowing feedback during the implementation. The drawback is documentation related to the process has less time to be compiled, and changes are not fully analyzed to extreme depths as in the waterfall method. Nevertheless documentation is compiled for the benefit of future developers to highlight changes, and to greatly elaborate on the project’s status.

* 1. **Terminology – Definitions, acronyms, and abbreviations**

EULA: End-User License Agreement

FIU: Florida International University

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

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

SPW: Senior Project Website

SPWv.1: Senior Project Website Version 1

SPWv.2: Senior Project Website Version 2

SPWv.3: Senior Project Website Version 3

SPWv.4: Senior Project Website Version 4

**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 project as a whole as well as specific document introduction.

**Section 2:** Covers overall system design including subsystems, hardware/software making, and how data and security/privacy are addressed.

**Section 3:** The detailed design goes into more details regarding affected subsystems utilizing static and dynamic diagrams as well as code specifications.

**Section 4:** Contains a glossary of terms.

**Section 5:** The appendix contains more detailed information for prior sections as well as from prior documentation.

**Section 6:** Contains references utilized in the document.

1. System Design

*Senior Project Website V4*



This section will cover the variety of details regarding the systems overall design. Particularly covering a more detailed overview, the system and subsystem decomposition, how hardware and software is mapped, the manner that persistent data is defined, and how security and privacy is utilized.

* 1. **Overview**

The SPW’s system is based upon the Two-Tier and MVC architectures. It is further decomposed to 6 subsystems: matchmaking, login, project, user management, repository, and Rest API. For SPWv.5 matchmaking underwent a great overhaul and Rest API was fixed by note Jonathan Santiago of the Collaborative Platform team of Summer 2014.

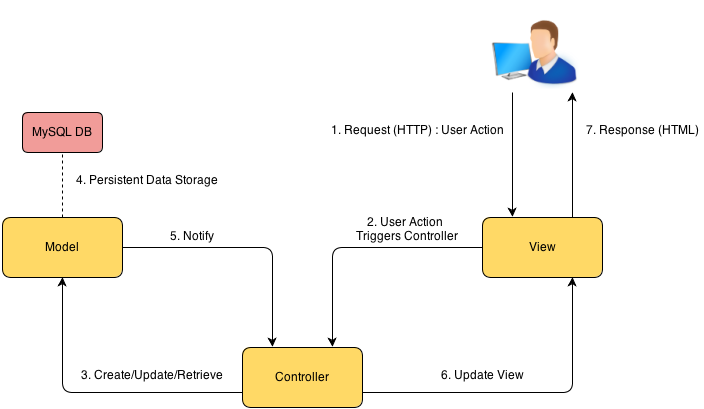
For deployment in hardware and software, the architecture is based on client and server, wherein a browser (client) accesses the SPW application (on a server). In addition the API works as an API for other senior projects requiring data on registered users.

Persistent data remains the same from SPWv.3 containing tables for users, language, projects, experience, skills, milestones, rankings, etc.

Security and privacy remains the same from SPWv.3. Having one way password encryption, session encryption, etc.

* 1. **Subsystem Decomposition**

The current system (SPWv.3) continues the system architecture of Three-Tier and MVC as the means to define how the system is structured. Likewise no change from that has occurred in SPWv.4. The style fits perfectly the function of the SPW, controllers maintain business logic for interactions, views allow users a means to observe relevant data, and models hold the means to access data from the database. For a user of a website that flow of logic is functional, they see from the view, their interactions on the views prompts the controller to respond, and the controller requests database data from models when required. From a three-tier perspective, the user’s browser and what they see (the view) is the presentation tier, the controller’s reaction to the user’s interaction is the logic tier, and the controllers request for data from the model is the data tier. Pictured below is a graphic describing the nature of the system from SPWv.3 design document.



The SPW is further decomposed to 6 subsystems, with their own MVCs, of them the primary focus for SPWv.4 is the matching algorithm subsystem which will be discussed first. For the purposes of documentation the remaining 5 subsystems will be described adapted from what was said in SPWv.3 design document.

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

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

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

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

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

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

* 1. **Hardware and Software Mapping**

This aspect for SPWv.4 is largely unchanged from the description of SPWv.3 in their design document. The hardware and software is essentially based on two architectural styles. The commonplace client-server architecture wherein the client runs a web browser to access the server running the SPW application running under an Apache Tomcat server interacting with a MySQL server for database needs.

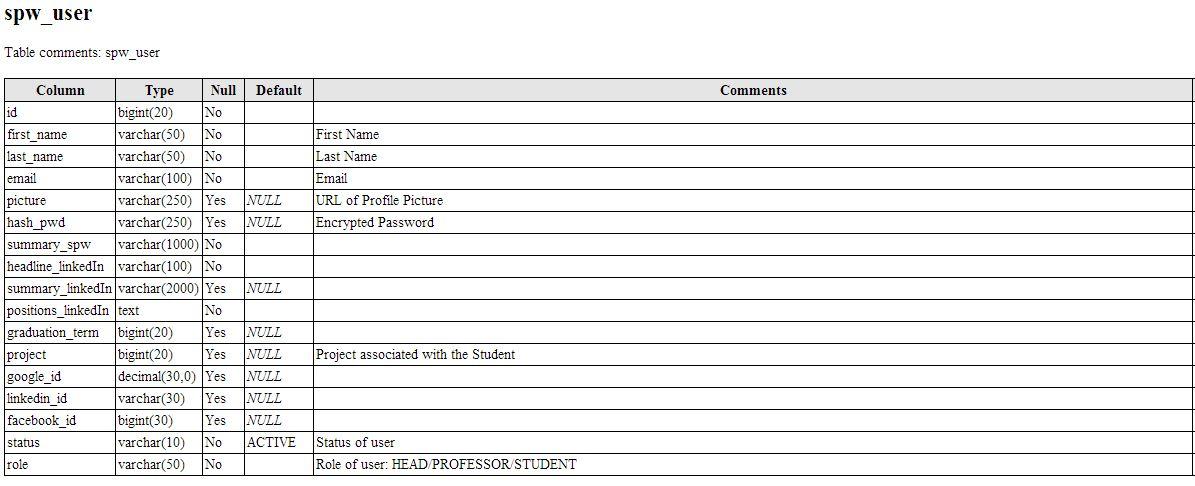
Additionally the server will run an API to verify members in the SPW amongst other projects in the senior project class. Below is a display of this from SPWv.3 design document.

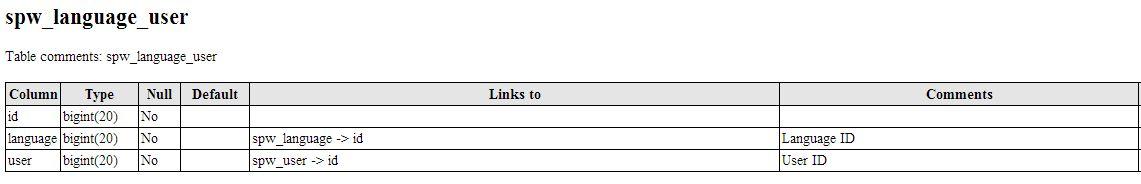


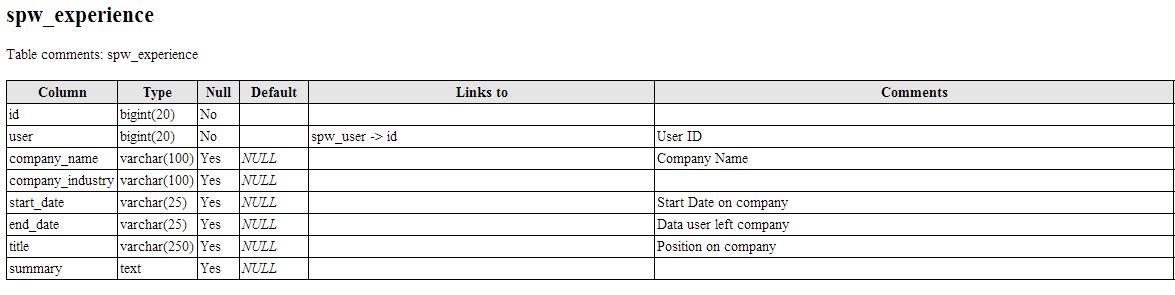
* 1. **Persistent Data Management**

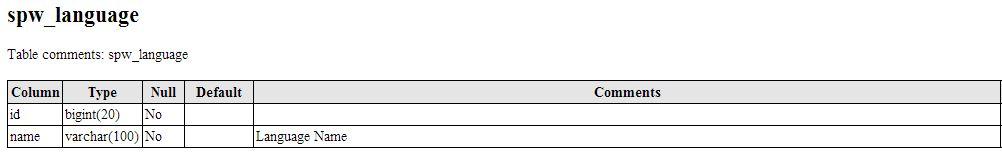
The persistent data did not change from SPWv.4 from SPWv.3. Particular tables and structures from SPWv.3 were of great utility for the matchmaking redesign particularly spw\_rank\_user. But no change at all has been made from SPWv.3.

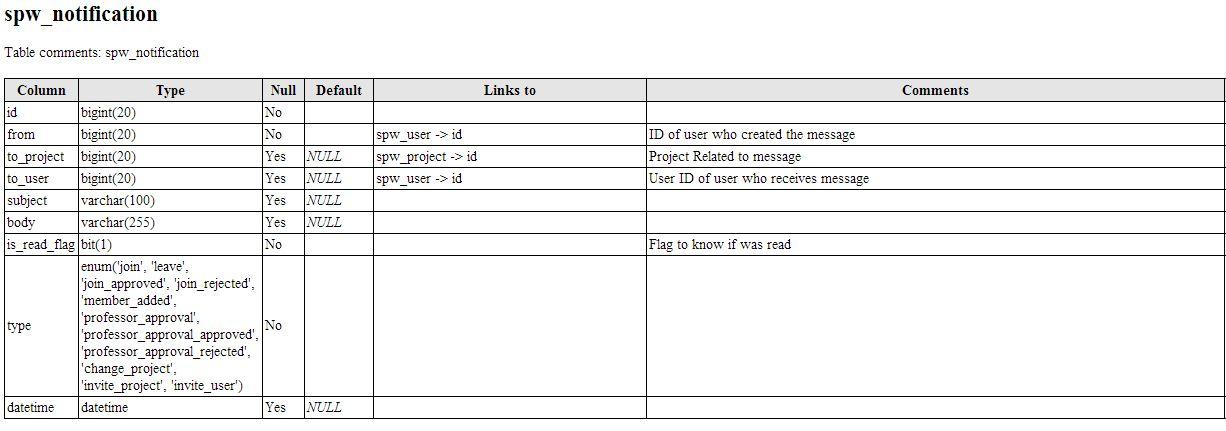
**Data Dictionary for SPW (from SPWv.3 design document):**

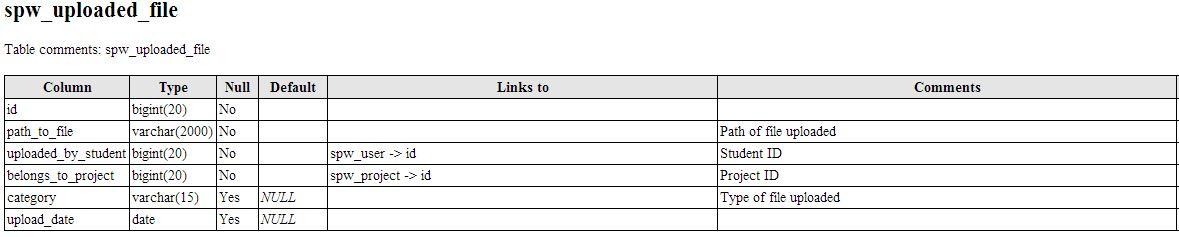
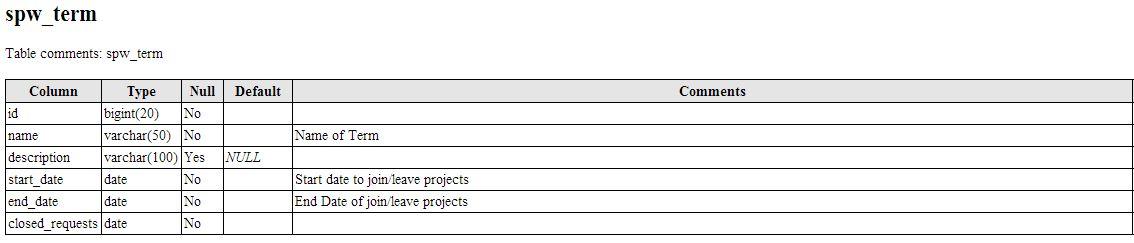
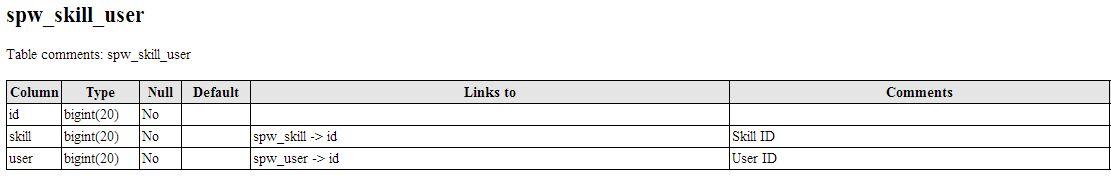
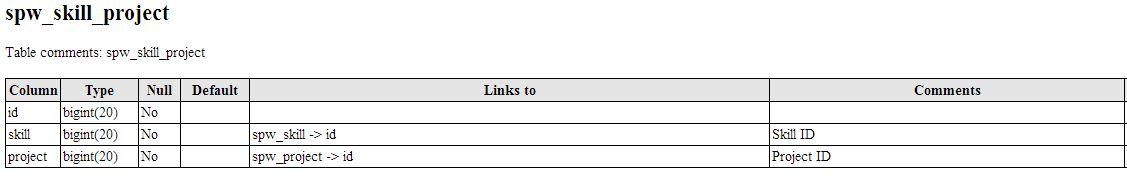
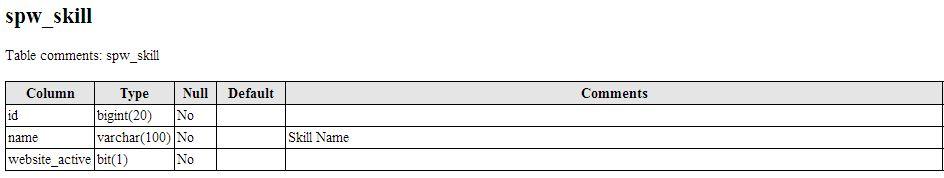
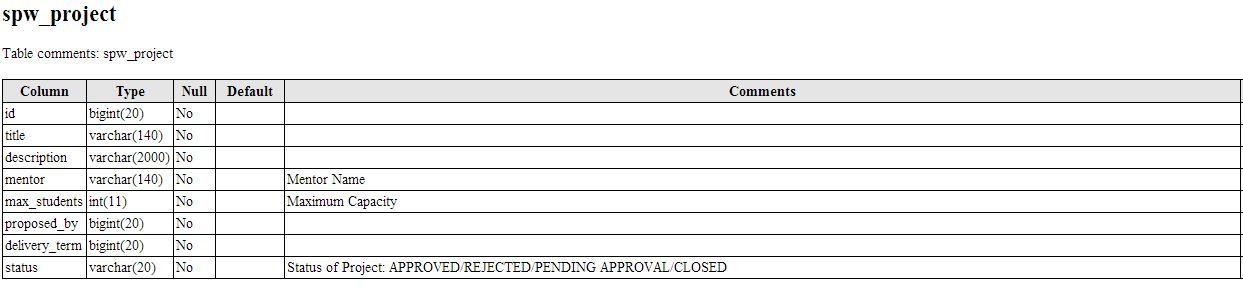


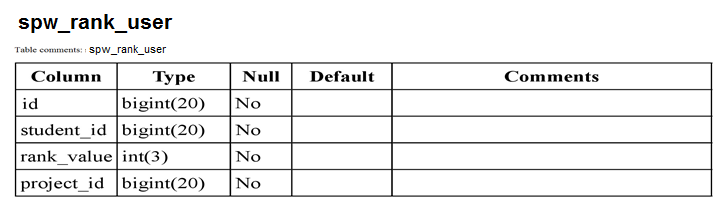














Entity-Relationship Diagram for Model Subsystem

* 1. **Security/Privacy**

No changes to any of the following occurs going from SPWv.3 to SPWv.4. The following information is entirely from SPWv.3 design document.

SQL Injection

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

URI Security

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

Cross-site scripting (XSS)

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

Cross-site request forgery (CSRF)

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

Session Encryption

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

One-Way password encryption

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

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

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

1. Detailed Design

*Senior Project Website V4*



This sections covers greater details with regards to subsystems described ins section 2.

* 1. **Overview**

Matchmaking will be heavily featured with regards to its static and dynamic models of representation. Prior document specifications will be added and mentioned as such.

* 1. **Static Model**

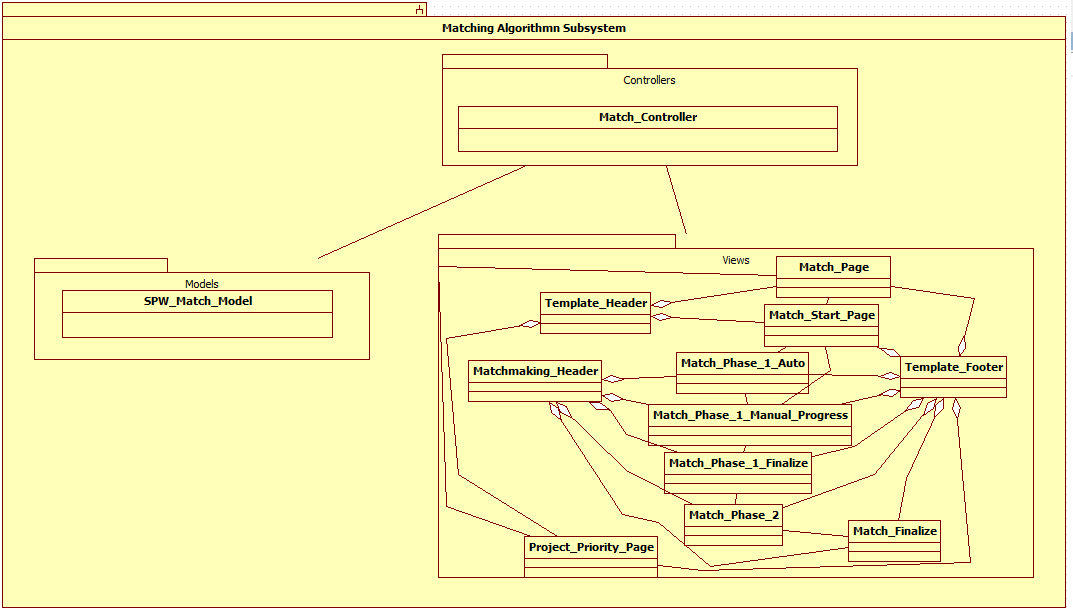
Going to SPWv.4 the static model’s principles from SPWv.3’s description remains unchanged. The biggest change is the matchmaking controller and models have had more methods added, and the views include more matchmaking specific views. What follows is adapted from the SPWv.3 design document and as before version specific changes are at top of this section and noted as such.

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

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

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

**Specific to version 4:**



Minimal Class Diagram for Matchmaking

****

Controller for matchmaking



Model for matchmaking



Views for Matchmaking

**Specific to version 3:**

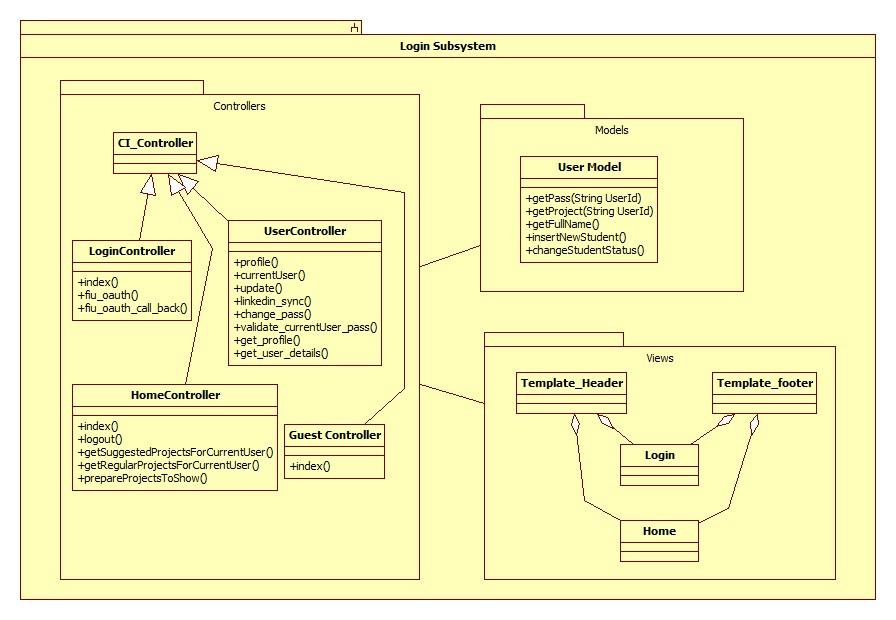
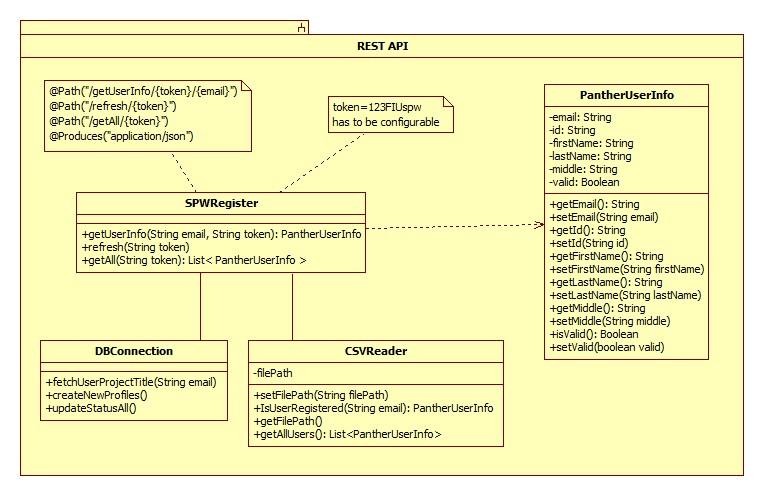
UML Detailed Class Diagrams for the SPWv.3



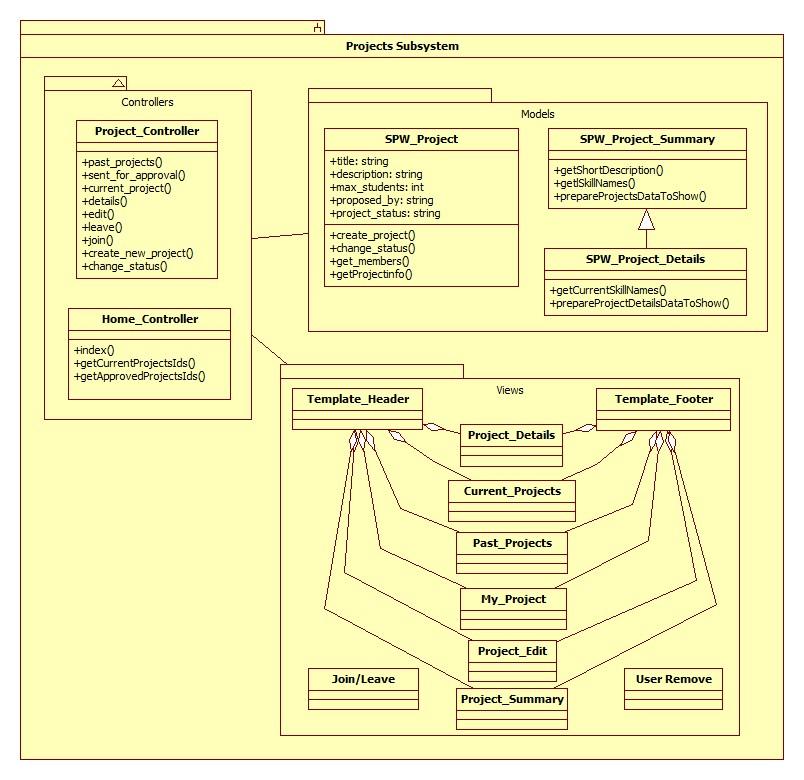




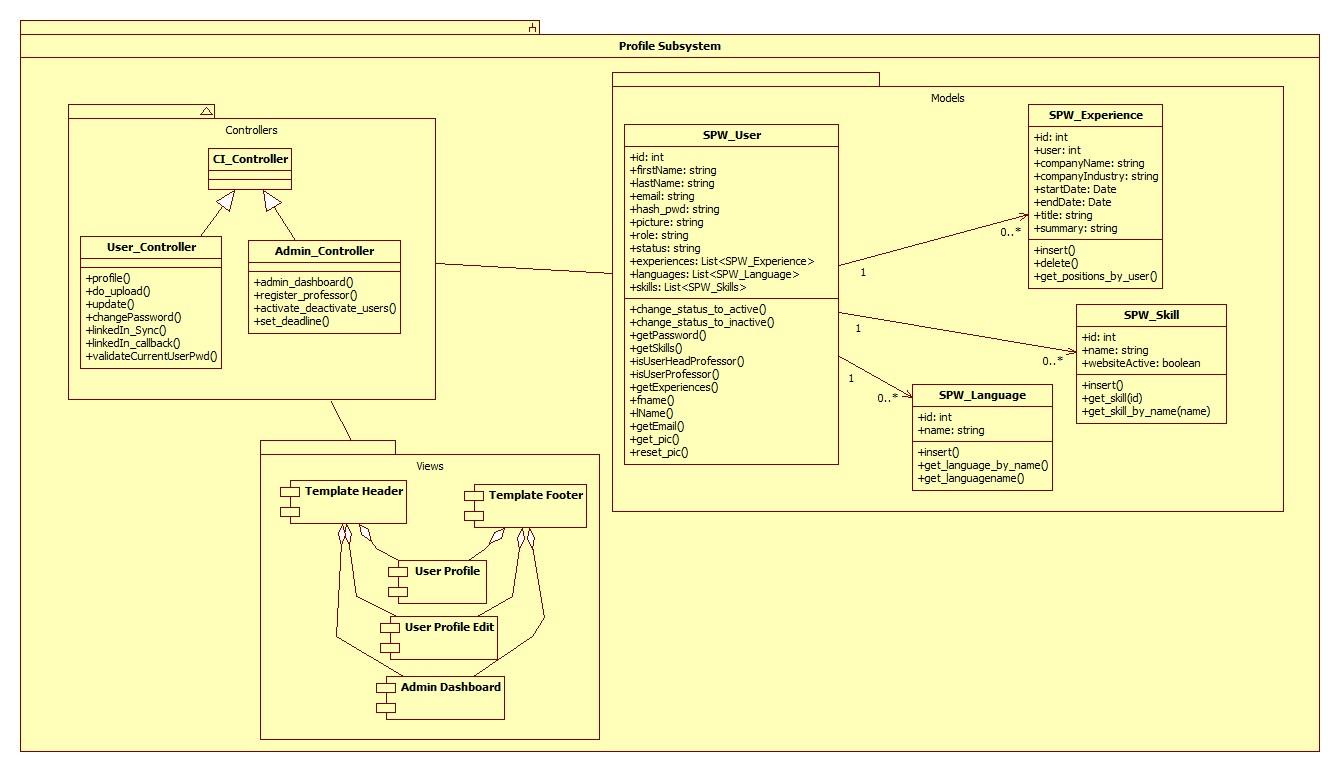
**Specific to version 2:**

 Class Diagram Login Subsystem

Class Diagram for SPW Register API



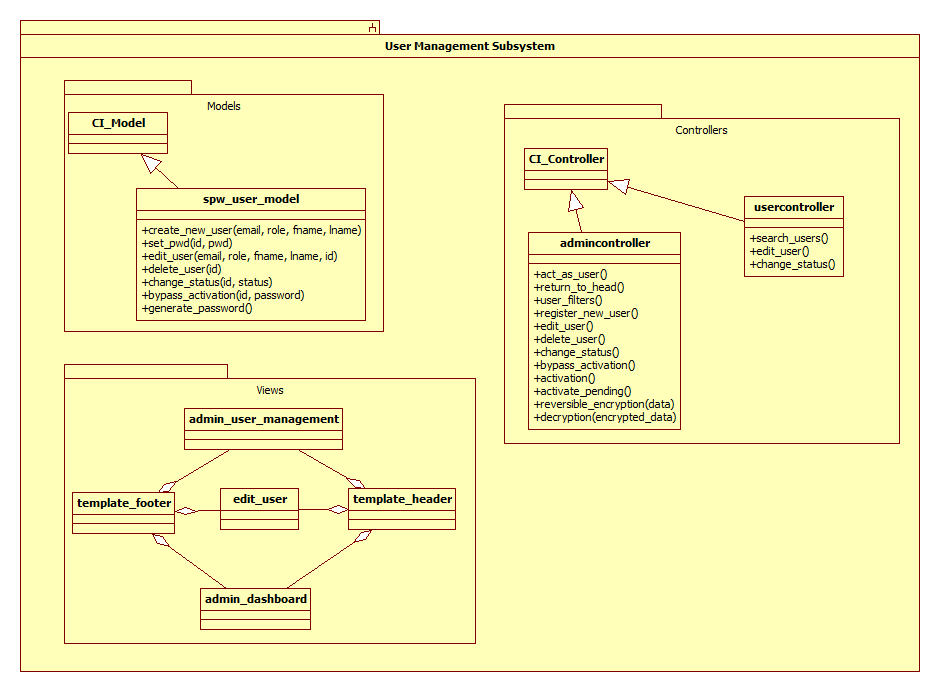
Class Diagram for the Projects Subsystem



Class Diagram for the Profile Subsystem



Class Diagram for the Repository Subsystem

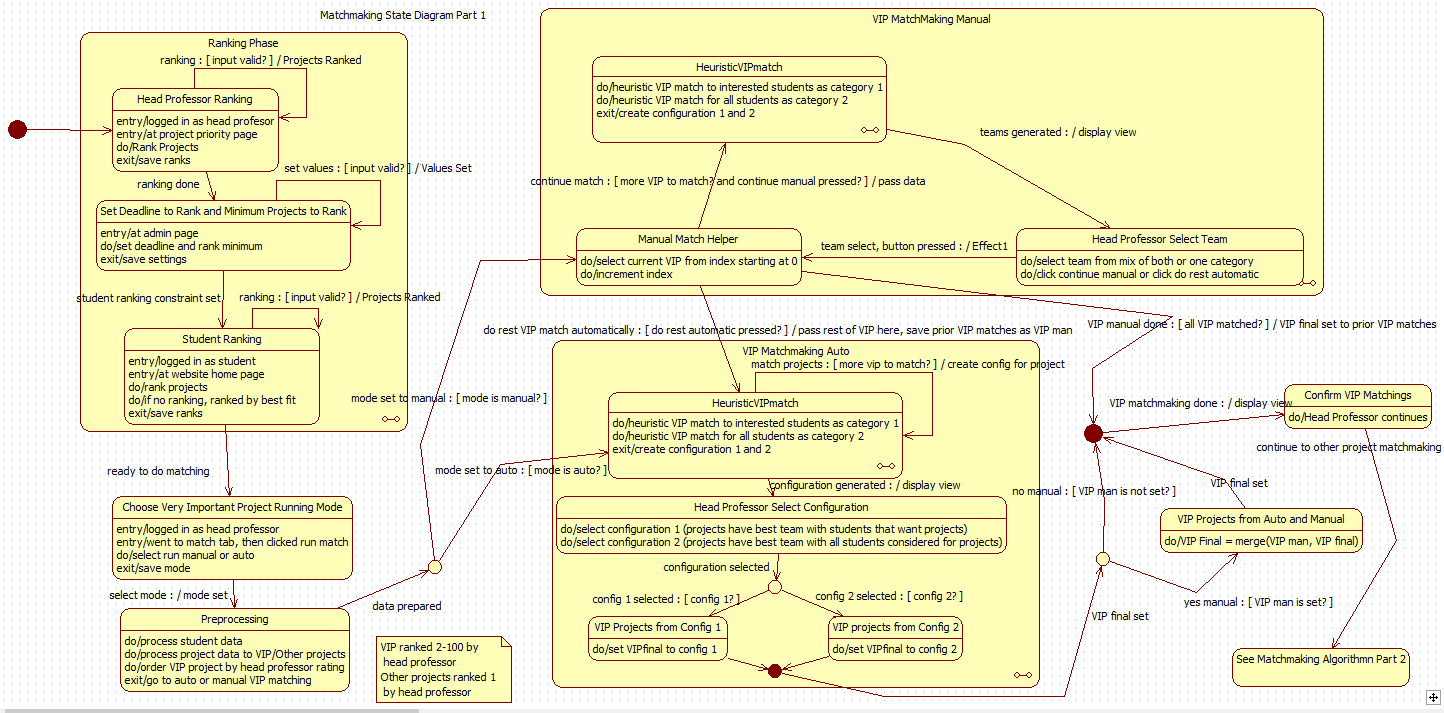


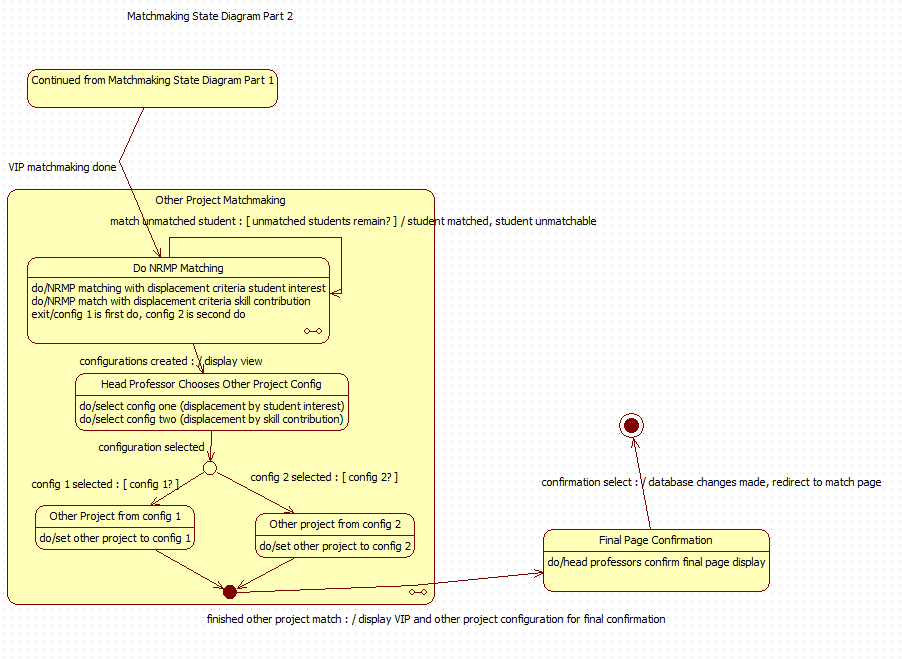
UML Class Diagram for the User Management subsystem

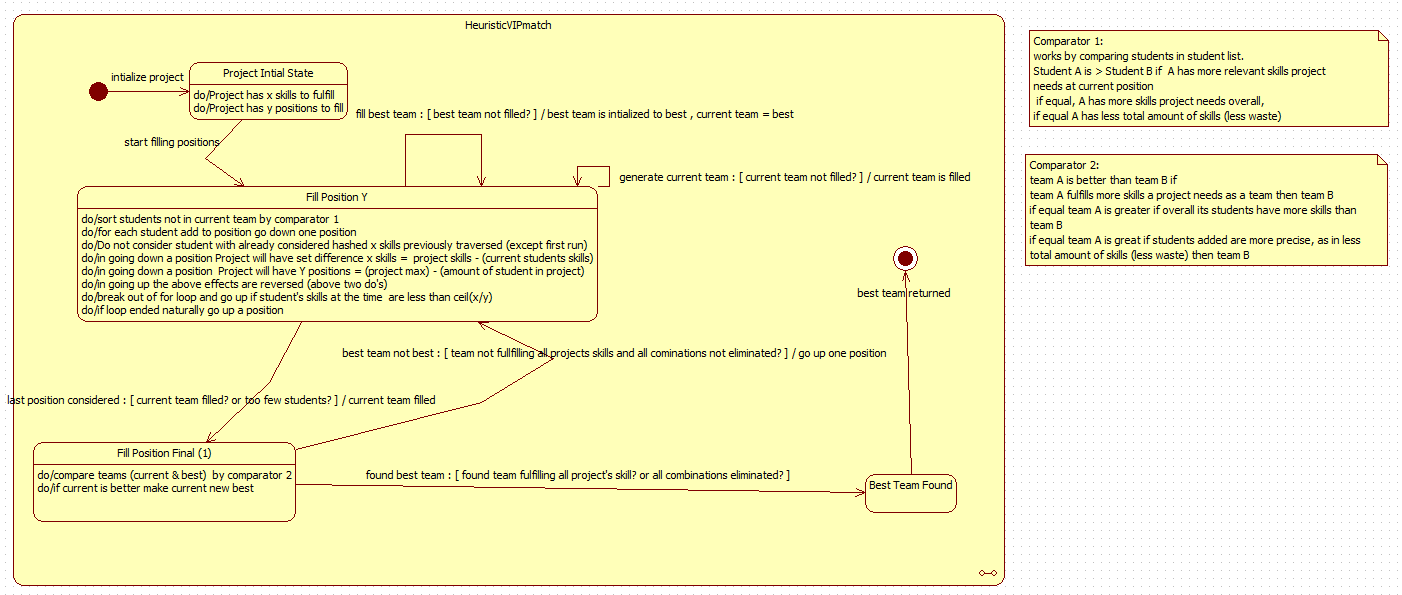
* 1. **Dynamic Model**

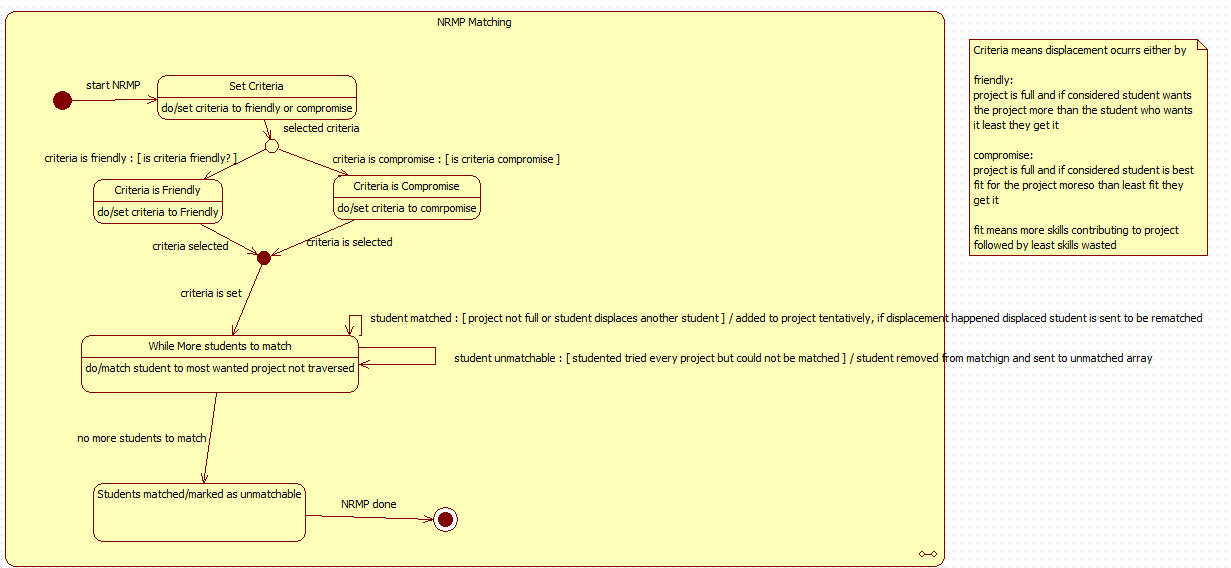
The first part of this section will cover dynamic models specific from SPWv.4, followed by all dynamic models from SPWv.3 minus certain things that were overwritten by SPWv.4 (models related to matchmaking).

**Specific to SPWv.4:**

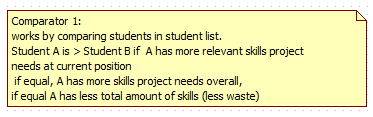
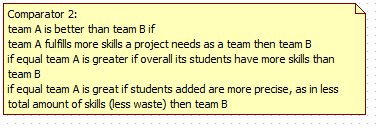
Following two diagrams depict matchmaking state machine.



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

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

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



The specific algorithm details for National Residency Matchmaking

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

Following is sequence diagram for running a match



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

Student Save Rank (SPW3\_205)



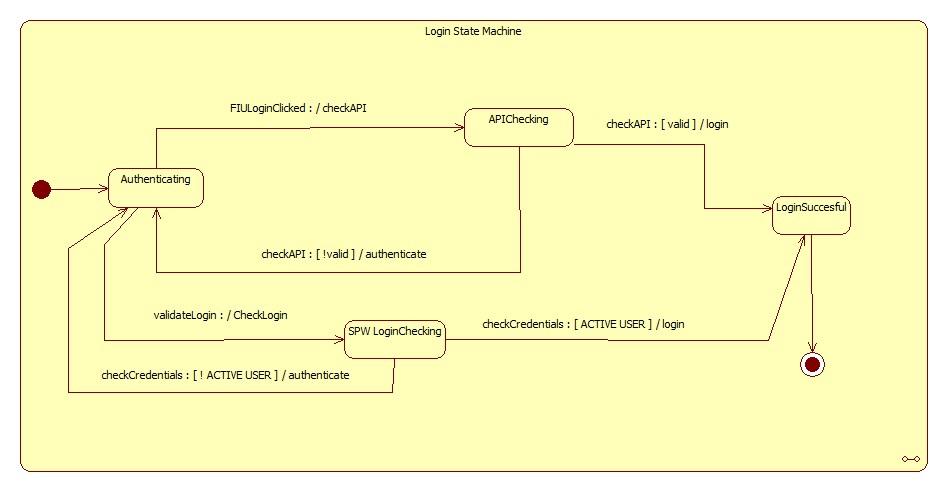
Professor Save Rank (SPW3\_205)

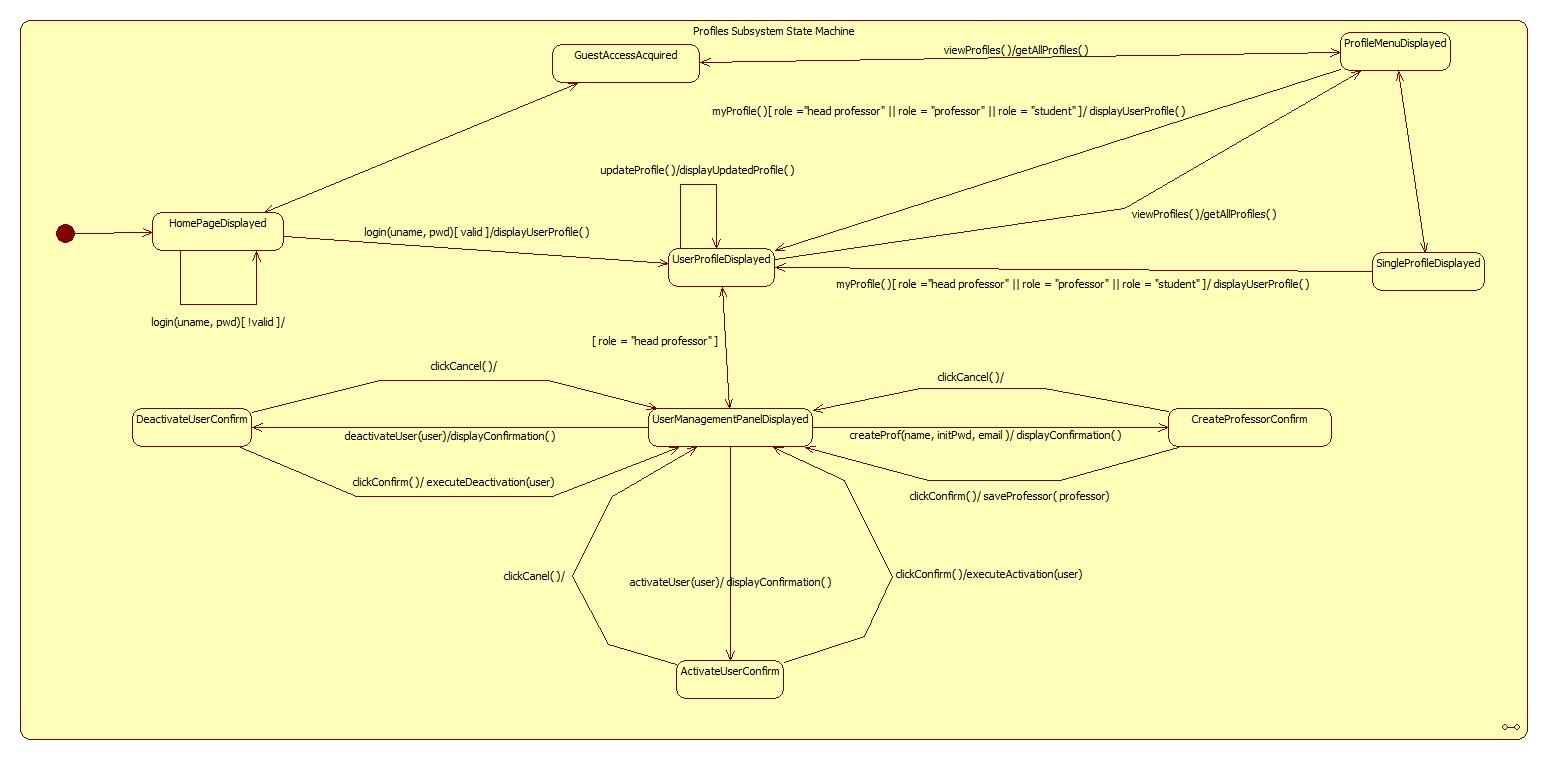
Project Priority Page not Match Page



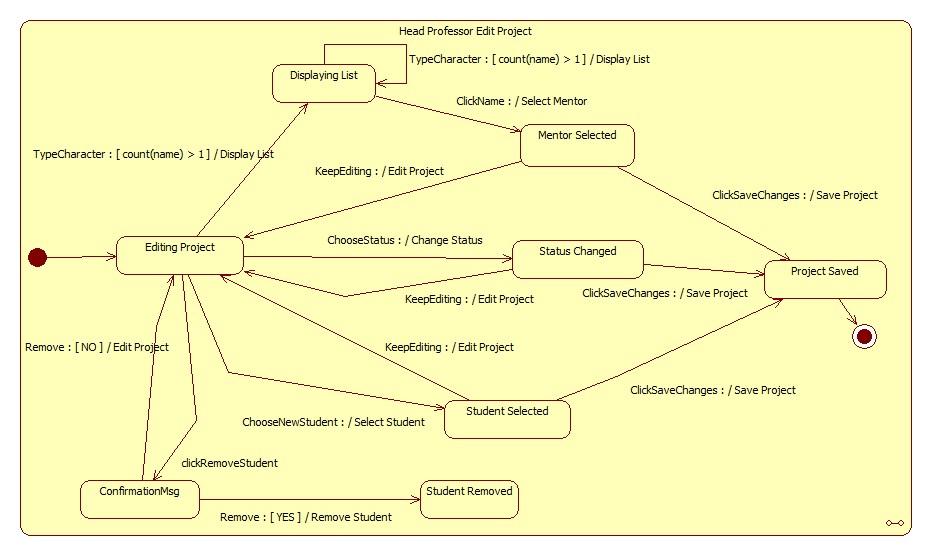
**The following dynamic models are from SPWv.3:**

**Login State Machine**



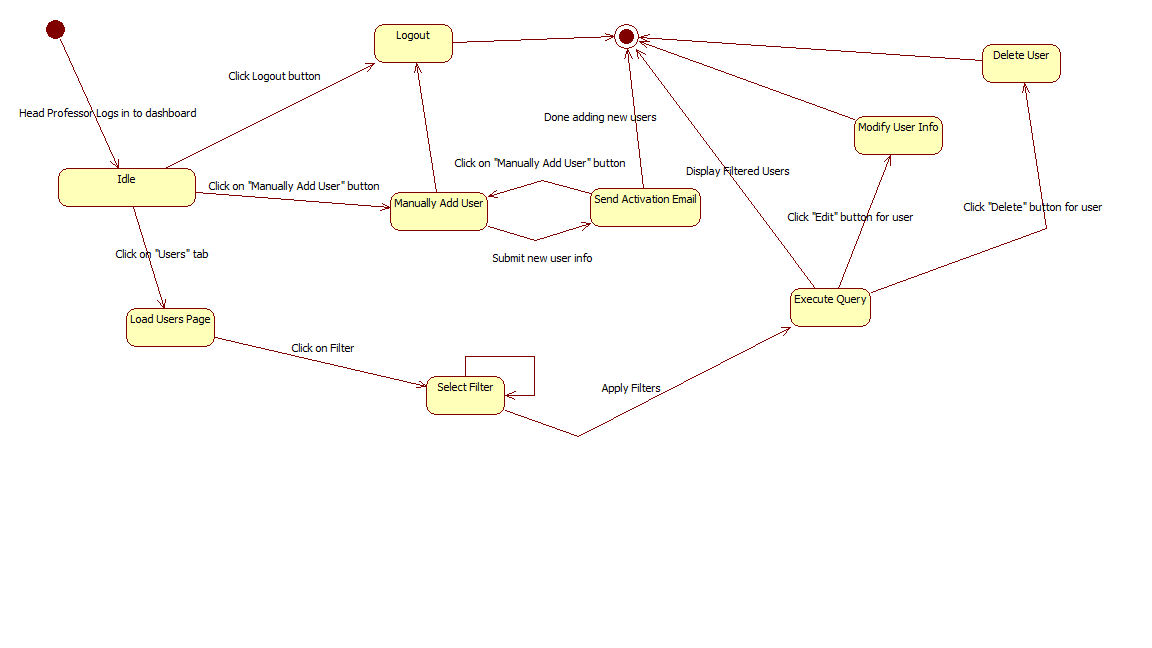
**Profile Subsystem State Machine**

**Head Professor Edit Project**



## Activity Diagram for Repository:

****

**Activity Diagram for User Management: **

Professor Login (SPW2\_103)



Guest Access (SPW2\_102)



Logout (SPW2\_104)





























Upload File (SPW3\_710)



Download File (SPW3\_720)

Delete File (SPW3\_730) 

Add New Milestone (SPW3\_901) 

Edit Milestone (SPW3\_902) 

Delete Milestone (SPW3\_903) 



Update Minimum (SPW3\_210)



* 1. **Code Specification**

Code specification details remain unchanged from SPWv.3. The following is almost entirely from SPWv.3 design document regarding that section, besides minor alteration.

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

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

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.

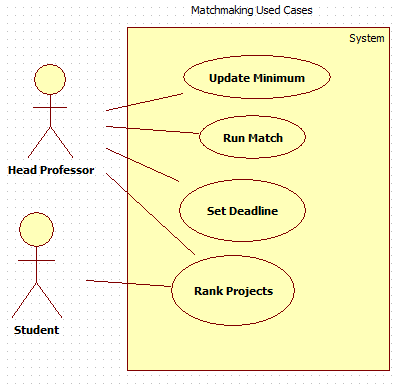
**LinkedIn:** Social media website for business professions to network and advertise themselves

1. Appendix

*Senior Project Website V4*



**5.1 Appendix A Used Case Diagrams (SPWv.4)**

****

**5.2. Appendix A Used Cases (SPWv.4)**

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

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

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

**5.3 Appendix C Class Interfaces (SPWv.4)**

**Matchmaking Controller**:

*Declaration*: class MatchController extends CI\_Controller

*Constructor*:

public function \_\_construct() {

parent::\_\_construct();

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

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

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

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

}

*Methods*:

public function backtracking($SL,$remain,$pos,$reset)

public function bettermatch($c,$b,$oSkills)

public function cloneProjects($opl)

public function cloneStudents($osl)

public function doHeuristicMatch($friendly,$SL,$p)

public function doMatchPhase1Auto($PL,$SL)

public function doMatchPhase1Manual($PL,$SL,$indexM)

public function doMatchPhase2()

public function doNRMP($PL,$SL,$min)

public function generateDemoData($PL,$SL,$min) NOTE THIS IS FOR DEMO PURPOSES

public functiongenerateMatchMetaData($PL)

public function generateProjectMetadata($p)

private function getRegularProjectsForCurrentUser($lSuggProjectIds)

public function getRelaventSkillData($SL,$sb)

public function gotoAuto()

public function gotoManual()

public function gotoProjectPriority()

public function hashSkills($skills)

public function inList($s,$SL)

public function index()

public function matchFinalizeHelper()

public function matchPhase1Helper()

public function matchPhase1HelperAuto()

public function matchStart()

public function preProcessSteps($auto)

public function prepareProjects()

private function prepareProjectsToShow($lProjectsIds)

public function prepareStudents()

public function pruneDupe($s,$checked)

public function pruneUnqualified($qualifier, $s)

public function reduceStudents($SL, $p)

public function saveMatchings()

public function saveMinimum()

public function saveRank()

public function sucessfulGroup($s,$remain)

public function unsetStudent($arr,$id)

**Matchmaking Model**

*Declaration*: class **SPW\_Match\_Model** extends CI\_Model

*Constructor:*

public function \_\_construct() {

parent::\_\_construct();

}

*Methods:*

public function getRanks($user)

public function getMinimum()

public function setMinimum($minVal)

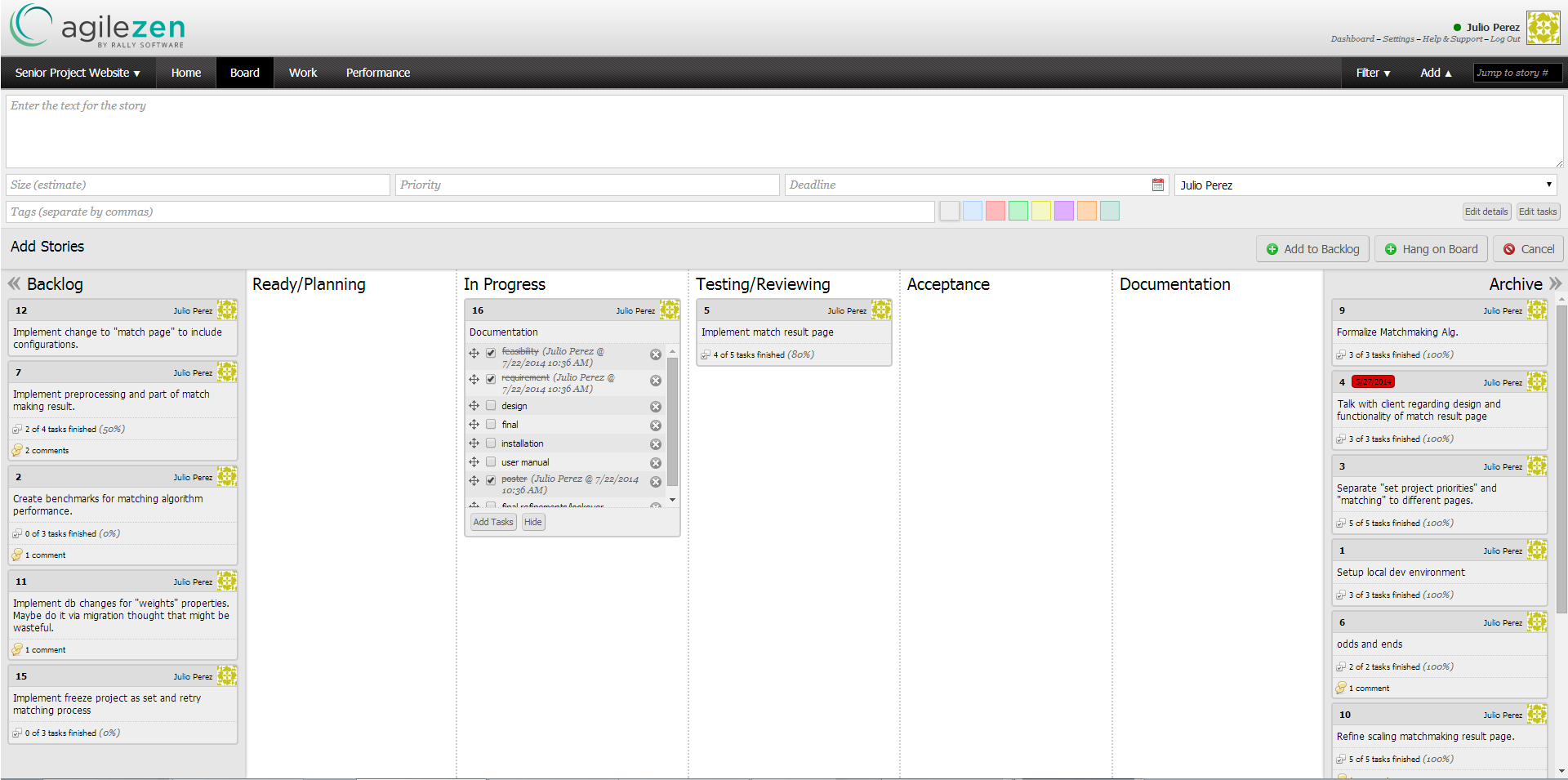
public function insertRanks($rank)

public function removeRank($user, $project)

public function rank\_exists($user, $project)

public function addStudentToProject( $student\_id,$project\_id)

**5.4 Appendix D Diary of Meetings and Tasks**

No official meetings have occurred since this group is a team of one. Weekly meetings have occurred with the client to ascertain status and upcoming tasks to develop and tackle. Such a manner of development is known as agile methodologies which were aided in part by Agile Zen pictured below.

1. References

*Senior Project Website V4*



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

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

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