*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Project Title: VIP 2.0

**Team Members:**

Tiago Moore

Miguel Conde

Steven Rowe

Andres Villa

Victoriano Vega

Jorge Perez

Rodolfo Viant

**Product Owner(s)**:

Masoud Sadjadi

**Mentor(s)**:

Francisco Ortega

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *<2016> <FIU>*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of the VIP 2.0 project.VIP 2. is a web application that connects faculty members with students in order to facilitate research opportunities that also allow students to get credits for their discipline*

**Table of Contents**

**Introduction**

Current System

Purpose of New System

**User Stories**

Implemented User Stories

Pending User Stories

**Project Plan**

Hardware and Software Resources

Sprints Plan

**System Design**

Architectural Patterns

System and Subsystem Decomposition

Deployment Diagram

Design Patterns

**System Validation**

**Glossary**

**Appendix**

Appendix A - UML Diagrams

Appendix B - User Interface Design

Appendix C - Sprint Review Reports

Appendix D - Sprint Retrospective Reports

**References**

# Introduction

## Presented in the following section is the purpose of the system along with its scope, development methodology, and terms specific to the VIP system. The development methodology includes process models and UML models describing the system design.

## Current System

* The system will have to implement a way of saving member information, which will be done by creating accounts and using a log-in subsystem
* There are security constraints on the login subsystem, which will be accounted for to protect from false authorization of accounts
* There are further security constraints on the fact that member information is being stored by the system, which will also be accounted for in order to secure member information  
  System will need to implement username and password authentication, as well as Google login and integration
* Web forms and data submission will be handled by the browser and so browser compatibility will be a major constraint
* Members must be able to access the system whenever they desire
* Members must be able to make multiple searches in a manner of seconds in order to maintain a pleasurable user experience

## Purpose of New System

The purpose of The Vertically Integrated Projects (VIP) Program is to unite undergraduate education and faculty research in a team-based context. Undergraduate VIP students earn academic credits, while faculty and graduate students benefit from the design/discovery efforts of their teams.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the Mobile Judge and VIP applications. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

**User Story #547 - Implement Docker CI**

As a developer, I want to be able to push to github and trigger an automatic build in order to simulate production environment and run unit/functional tests if available  
   
Acceptance Criteria:  
 1. Build is triggered on every push  
 2. Monitors all branches on github

## Use Case

**Use Case ID:(MJ-547) Implement Docker CI**

Details:

Actor: Developer

Pre-conditions:

1. Developer has pushed code to github repo to a respective branch

Description:

1. Use case begins when github recognizes a push has occurred to the repository
2. Docker hub will recognize that a push has a occurred, and begin a build
3. Docker will pull the repository using the latest commit
4. Docker will begin building an image using the Dockerfile from the commit

Post-conditions:

1. The number of triggered builds in Docker goes up by 1
2. The build will complete, and report a failure or a success

Alternative Courses of Action:

1. After step D.4, the branch may be automatically merged if the build was a success

Exceptions:

N/A

Related Uses Case:

N/A

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into the social aspect integration features.

Risk: Medium. Implementing this use case employs standard web-based technology.

Constraints:

1. Usability:
   1. No training time  required.
   2. User guide should provide more in-depth information on Docker if necessary.
2. Reliability:
   1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   1. The build should be triggered within 20 seconds
4. Supportability:
   1. The docker build should support any branch on the repository

Modification History:

Owner: Andres Villa.

Initiation date: 01/28/2016

Date last modified: 01/28/2016

**User Story #542 - Add Check Box on Filters for People Selection**

As an admin, I want to have a check box to select and deselect all filter types on the people tab.

### Acceptance Criteria:

1. check box similar to the email wizard that will select all filters upon click

**Use Case**

**Use Case ID: MJ542 - Add Check Box on Filters for People Selection**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is using the People tab.

Description:

1. Use case begins when the admin clicks on the people tab as selects students or judge .
2. The system shall provide a checkbox to the left of the filters tab when selected students or judges.
3. The Actor shall click the checkbox to have all filters simultaneously selected.
4. The system shall select or unselect all filter boxes and return data accordingly.
5. Use case ends when all filters have been applied or unapplied and Actor exits the view.

Post-conditions:

1. The filters are all simultaneously selected.
2. All data has been returned accordingly.

Alternative Courses of Action:

1. Actor has option of using filters individually.

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into existing app function .

Risk: Medium. Implementing this use case requires Member to learn Sencha framework.

Constraints:

1. Usability:
   1. No training time  required.
   2. Button should be visible aligned to filter boxes.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. There should be no lag or delay between selection
   2. Check box should be able to be clicked at will.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Tiago Moore.

Initiation date: 01/26/2015

Date last modified: 02/01/2015

**User Story #523 - Add Search Feature to Email Wizard**

As an admin, I want to search for a name when using the email wizard tool so that I may easily find contacts on my contact list.

Acceptance Criteria:

1. Search bar will be same style as the one in People tab.
2. The search bar is available in Step 2 of the wizard.
3. The admin should be able to choose all students in Step 1, for example, and then in Step 2, he/she can filter some names, unselect them, and then go to the next step.
4. It should always keep track of the selections in Step 2, no matter what the filter currently shows.
5. If the user goes back to Step 1, the user should be warned that the changes made in Step 2, if any, will be lost. If there is no changes in Step 2, no need for such a warning.

## Use Case

**Use Case ID:**(MJ-523) - Admin filters results in email wizard

**Details:**

**Actor:** MobileJudge Admin

**Pre-conditions:**

1. Admin is logged into MobileJudge system
2. Admin is on Refine Search( 2nd tab) of email wizard

**Description:**

1. Use case begins when Admin types in search criteria in search bar and clicks on search icon.
2. The system shall reload the student and judge models that match search criteria based on name or email.
3. The system shall add clear icon on search bar.
4. The admin clicks on clear icon on search bar.
5. The system shall remove clear icon on search bar.
6. Use Case ends when model has been properly updated with any changes made by Admin.

**Relevant requirements:**

1.none

**Post-conditions:**

1. Model is properly updated with filter applied.
2. Model is properly updated with filter removed.

**Alternative Courses of Action:**

1. Admin clicks on Previous button
2. The system shall display message warning admin that all changes will be lost if he navigates back
3. Admin clicks yes
4. The system shall display previous view.

**Exceptions:**

none

**Related Use Cases:**

none

**Decision Support**

**Frequency:**  One average <5 searches are performed per navigation to page.

**Criticality:** Medium. Allows Admin to search for specific student/judge.

**Risk:** Low. Invoking this use case involves standard web based technology.

**Constraints:**

Usability:

* No previous training time required.
* On average, it should take the user less that 1 second to update model information.

Reliability:

* Mean time to failure - 1% for every attempt is acceptable.

Performance:

* Response time - system should react to a user input within 100ms - 500ms.
* Throughput - the system should accomplish the work within 50ms.
* Availability - always available unless the system is down for maintenance.

Supportability:

* Adaptability - should work on mobile devices of multiple screen sizes.
* Maintainability - changes to the application should work with these web browsers Chrome, Firefox, IE11.

**Modification History**

**Owner:** Miguel Conde

**Initiation date:** 01/22/2016

**Date last modified:** 01/28/201

**User Story #524 - Manage List Size**

As an admin, I want a setting to choose how many items to be included in the list on each page, so that I can easily scroll up or down with the amount of data I would like to view in one shot.

Acceptance Criteria:

1. All the pages with a list should have this option (e.g., Student’s page, Judge’s page, All Emails, etc.)
2. The admin should be able to enter any number.
3. If no number is entered, then all should be shown.
4. Default is 10.
5. If the admin changes the number, it should be saved for future use.
6. The saved number should be kept for each user and each page visited by the user.

**Use Case**

**Use Case ID: MJ524 - Manage List Sizes**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is on the Email or People tab.

Description:

1. The use case begins when the admin clicks the list size combo box on the Email or People view.
2. The admin then clicks on one of the list size options.
3. The size of the list then updates to only show the size that the admin selected.

Post-conditions:

1. List is now the size that the admin selected.

Exceptions:

1. Error updating the list size.

Related Uses Case:

Decision Support:

Frequency: Admins are expected to change list sizes at least 2 times per login.

Criticality: This use case is of medium importance. Helps the admin adjust list size depending on how many roles selected or filters applied.

Risk: Low risk involved. Feature should take about 16 hours of developer times.

Constraints:

1. Usability:
   1. Feature is implemented on many different platforms therefore, admin should not need training.
   2. If training is needed, admin should be trained and able to understand the functionality in 5 minutes.
2. Reliability:
   1. Mean time to Failure - 7 days
   2. Availability - 95% uptime
3. Performance:
   1. Admin should have list updated within 1 second of clicking the list size link or ‘OK’.
4. Supportability:
   1. Option to change list size should be available on Desktop and Mobile platforms.

Modification History:

Owner: Jorge Perez

Initiation date: 01/26/2016

Date last modified: 01/28/2016

**User Story #520 - Manage Profile**

As a user, I want to be able to click on my name and/or profile picture, which should be accessible from all pages, so that I can review and modify my profile, including my profile picture, password, name, etc.

Acceptance Criteria:

1. After a user logs in he will be able to see his/her name and/or profile picture on all pages.
2. User must be able to click on his name and/or profile picture to manage his/her profile.
3. All changes will be effected immediately when user successfully save the changes.

## Use Case

**Use Case ID: MJ520 - Manage Profile**

Details:

Actor: User

Pre-conditions:

1. Once user is logged in, the user profile information is loaded onto the screen

Description:

1. Use case begins when the user clicks his profile picture or name
2. The system will redirect the user to his profile page
3. Once redirected to the his/her profile page, all the profile information will be able to be edited.
4. The system will be able to save the new profile information changes.
5. Use case ends once the user clicks save.

Post-conditions:

1. n/a

Alternative Courses of Action:

1. n/a

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add/edit data into existing app function.

Risk: Medium. Implementing this use case requires Member to learn Sencha framework.

Constraints:

1. Usability:
   1. No training time required.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. User information will be saved and all relevant information will be updated with new information
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Marlon Rowe

Initiation date: 01/26/2015

Date last modified: 02/05/2015

**User Story #548 - Export Grades**

As an admin, I would like to be able to export final grades in a csv file, so that I can import them into the Moodle course.

Acceptance Criteria:

1. I should be able to filter the list of students for whom the grades will be exported.
2. The list must contain the student's email addresses to be used as the unique identifier when importing in Moodle.
3. Only the final grades, which are calculated using the accepted/approved grades will be included.
4. The list can be imported into Moodle.

***Use Case:***

**Use Case ID: MJ548 - Export Grades**

*Details:*

*Actor:* Administrator

*Preconditions:*

1. Administrator has successfully logged onto the system.

*Description*

1. Use case begins when the administrator clicks on People button in the left side panel.
2. Administrator clicks on students button on the left right panel of that page.
3. The grades are shown in the page.
4. The administrator clicks on the 'Export' button.
5. A small window comes up confirming the file to save, then click on the OK button. The file is saved in a specific location in csv format.
6. Use case ends when the administrator logins to Moodle and import the grades to the system.

*Post-Conditions*

1. The administrator is able to see the grades of the students in the Moodle system.

*Alternative Course of Action*

1. In step D.5 the administrator has the option to cancel the request.

**Constraints**

1. Usability

a. The administrator should be able to use this use case easily.

2. Performance

*a. Request for loading the sign up page should be loaded immediately after being requested.*

*b.* *The system should be able to handle as many requests as the administrator needs*

3. Supportability

a. The MobileJudge site will be correctly displayed in Firefox and Chrome.

**Modification History:**

*Owner:* Victoriano Vega

*Initiation Date:* 01/26/2016

**User Story #522 - Export Judge List**

As an admin, I would like to be able to export any filtered list of judges into a csv file.

Acceptance Criteria:

1. The exported list only includes the filtered judges.
2. Exported file must be a CSV file.
3. The destination directory and name can be chosen by the admin.

**Use Case**

**Use Case ID: MJ522 -Export Judge List**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is using the People tab.
3. Admin has clicked on the Judges tab.

Description:

1. Use case begins when the admin clicks on the people tab as selects students or judge .
2. The admin must click on the Judges tab and click the export button.
3. The system shall prompt the admin to save the csv file of judges.
4. The csv file with judge information shall be saved into the actors system.
5. Use case ends when the actor exits the view.

Post-conditions:

1. There are judges to export

Alternative Courses of Action:

1. Actor has option of using filters.

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into existing app function .

Risk: Medium. Implementing this use case requires Member to learn Sencha framework.

Constraints:

1. Usability:
   1. No training time  required.
   2. Button should be visible aligned to filter boxes.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. There should be no lag or delay between selection
   2. Check box should be able to be clicked at will.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Tiago Moore.

Initiation date: 01/26/2015

Date last modified: 02/015/2015

**User Story #623 - Visualize Account States**

As a user, I would like to see which accounts are currently linked so that I could link or unlink any accounts.

Acceptance Criteria:

1. There should be a checkbox field adjacent to the social media icon I would like to link or unlink.
2. The state should update as I link and unlink accounts.
3. The checkbox field should be immutable by the user.

**Use Case**

**Use Case ID: MJ623 - Visualize Linked accounts**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is using the Profile feature.
3. A checkbox will display a linked account.

Description:

1. Use case begins when the admin clicks his name or picture.
2. The system shall checkboxes which icons for which account the user would like to unlink or unlink .
3. The Actor shall click the icon for the account he would like to unlink.
4. The checkbox shall update accordingly if the account is linked or not.

Post-conditions:

1. If the account is linked, checkbox will be checked.
2. If account is unlinked, checkbox will be unchecked.

Alternate:

1. An admin will check the database to see if accounts are linked or not.

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into existing app function .

Risk: Medium. Unlink feature will require additional time.

Constraints:

1. Usability:
   1. No training time  required.
   2. Account buttons are visible on Profile tab.
   3. Visual will display an already linked account.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. Checkboxes should update immediately after linking and unlinking.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Tiago Moore.

Initiation date: 02/22/2015

Date last modified: 02/29/2015

**User Story #622 - Unlink Account**

As a user, I want to be able to unlink my social media accounts to my mobile judge account.

Acceptance Criteria:

1. I should be able to click and then will subsequently will not be allowed to login with that social media account.

**Use Case**

**Use Case ID: MJ622 - Unlink Accounts**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is using the Profile feature.
3. An account has to be previously unlinked.
4. A checkbox will display a linked account.

Description:

1. Use case begins when the admin clicks his name or picture.
2. The system shall boxes which icons for which account the user would like to unlink.
3. The Actor shall click the icon for the account he would like to unlink.
4. The system will prompt with a confirmation alert.

Post-conditions:

1. The account will be unlinked until it is relinked.

Alternate:

1. An admin will database access will manually remove linked accounts.

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into existing app function .

Risk: Medium. Unlink feature will require additional time.

Constraints:

1. Usability:
   1. No training time  required.
   2. Account buttons are visible on Profile tab.
   3. Visual will display an already linked account.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. There should be no lag or delay during button selections.
   2. Authentication should be done immediately after click.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Tiago Moore.

Initiation date: 02/22/2015

Date last modified: 02/29/2015

**User Story #525- Review Grades**

As an admin, I want to be able to Accept or Reject Grades entered by judges, so that I can finalize the grades for the students.

Acceptance Criteria:

1. The admin should be able to filter only the Rejected or Accepted grades and work on them.
2. The admin should be able to filter the grades using a search box similar to those in the Students page.
3. The admin should be able to select/deselect some/all of the grades in a filtered list and Reject/Accept the selected ones using one click.
4. This feature must be available on Mobile devices too.

## Use Case

Use Case ID:(MJ-525-1) - Admin Accepts Grades

**Details**:

**Actor**: Mobile Judge Admin Masoud Sadjadi

**Pre-conditions**:

1. Admin is logged into MJ system.

2. Admin is on the Grade Page

**Description**:

1. Use case begins when Admin clicks on the “Traffic Light” icon for one or

more students.

2. The system shall update the student’s grades in the Database.

3. The system shall then reload the Grade page with the updated data.

4. The system shall change the Traffic light to green for the selected student or all the students.

5. The use case ends when the model has been updated with any changes

made by the system Administrator.

**Post-conditions**: N/A

**Alternative Course of Action**:

1. Admin clicks on a different page.

2. The system shall display the new page.

**Exceptions**: N/A

**Related Use Case**: (MJ-525-2) - Admin Change Grades Status

**Decision Support**

**Frequency**: Used only at the end of semesters after Project Presentation.

**Criticality**: High. Allows Admin to remove unfair Grades or Remove too high Grades.

**Risk**: Low. Simple data fetch and update operation.

**Constraints**

**Usability**: No training required. It should take less than a minute to Accept/Reject a

grade.

**Reliability**: System should be able to query and update the required information in a

timely manner

**Performance**: System should be able to gather information and load the response in one

minute max.

**Supportability**

**Adaptability**: Should work on devices of multiple screen sizes.

**Maintainability**: Changes to the use case should be easy to perform and should be

reflected on all the devices supported.

**Modification History**:

*Owner*: Rodolfo Viant Rodriguez

*Initiation Date*: 01/28/2016.

*Termination Date*: 02/1/2016.

**User Story #619- Setup PM2 Server Monitoring**

As an admin, I would like to be able to monitor the production & development server, so that I may better see all the necessary processes needed to keep the application up and running.

Acceptance Criteria:

1. There should be clear user manual for the admin to use this feature.
2. It would be great if this feature can be used by the admin from within the web site, rather than command line; if too difficult, command line would be acceptable, only if the user manual is provided with sufficient details.

**User Story #543- Manage Judge**

As an admin, I would like to be able to add/delete/modify judges, so that I can better manage them.

Acceptance Criteria:

1. The screen should be very similar to the one under Settings → Questions with respect to adding new, delete, modify, refresh, etc.

## Use Case

**Use Case ID: MJ543 - Manage Judge**

Details:

Actor: Admin

Pre-conditions:

1. User must be logged in as admin

Description:

1. Use case begins when the admin double clicks on a row in the Peoples -> Judges tab.
2. The fields that are able to be modified become editable.
3. The admin can edit firstname, lastname, email, password, title, affiliation, and state
4. Once changes have been made, clicking on the ‘Update’ button, the store autoSyncs the data to the Judges API Route. Then saves the data to the backend.
5. Use case ends once fields return to normal, the data has been saved, and there are no errors.

Post-conditions:

1. n/a

Alternative Courses of Action:

1. n/a

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Admin to edit data into existing app function.

Risk: High. Implementing this use case requires Member to learn Sencha framework, Nodejs, and SQL database.

Constraints:

1. Usability:
   1. No training time required.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. Updated information will be saved to the database
4. Supportability:
   1. Feature must appear on desktop devices only.

Modification History:

Owner: Marlon Rowe

Initiation date: 02/14/2016

Date last modified: 02/26/2016

**User Story #544- Manage Student**

As an admin, I would like to be able to add/delete/modify students, so that I can better manage them.

Acceptance Criteria:

1. The screen should be very similar to the one under Settings → Questions with respect to adding new, delete, modify, refresh, etc.
2. After using the Sync button on the Students page, the Sync should still work exactly as before with one extra feature. The prompt should give the option not to delete extra users, just add and modify.

## Use Case

**Use Case ID: MJ544 - Manage Student**

Details:

Actor: Admin

Pre-conditions:

1. User must be logged in as admin

Description:

1. Use case begins when the admin double clicks on a row in the Peoples -> Student tab.
2. The fields that are able to be modified become editable.
3. The admin can edit Location by typing freely and State from a dropdown list.
4. Once changes have been made, clicking on the ‘Update’ button, the store gets notified and autoSyncs the data to the Student API route, then saves the data to the database.
5. Use case ends once the data has been saved and no errors have occured.

Post-conditions:

1. n/a

Alternative Courses of Action:

1. n/a

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Admin to edit data into existing app function.

Risk: High. Implementing this use case requires Member to learn Sencha framework, Nodejs, and SQL database.

Constraints:

1. Usability:
   1. No training time required.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. Updated information will be saved to the database
4. Supportability:
   1. Feature must appear on desktop devices only.

Modification History:

Owner: Marlon Rowe

Initiation date: 02/14/2016

Date last modified: 02/21/2016

**User Story #519- Link Login Accounts**

As a user, I would like to be able to login using my accounts on other supported networks, such as FIU, Google, LinkedIn, Facebook, and Twitter.

Acceptance Criteria:

1. After authentication the user will be able to link all his other supported accounts so that they could subsequently log in with any of them.
2. Under profile there will be a Manage Accounts option.
3. The manage accounts will allow you to link and unlink accounts to your username.
4. The user will subsequently be able to log in with any of those accounts.

**Use Case**

**Use Case ID: MJ519 : Link Login Accounts**

Details:

Actor: Admin

Pre-conditions:

1. Admin has successfully logged onto the system.
2. Admin is using the Profile feature.

Description:

1. Use case begins when the admin clicks his name or picture.
2. The system shall boxes which icons for which account the user would like to link.
3. The Actor shall click the icon for the account he would like to link.
4. The system will prompt the user to enter his credentials .
5. The system will link the accounts for any subsequent logins.

Post-conditions:

1. The account selected will be linked.
2. Subsequent login attempts will log user in appropriately.

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into existing app function .

Risk: Medium. Unlink feature will require additional time.

Constraints:

1. Usability:
   1. No training time  required.
   2. Account buttons are visible on Profile tab.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. There should be no lag or delay during button selections.
   2. Authentication should be done immediately after click.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Tiago Moore.

Initiation date: 02/15/2015

Date last modified: 02/22/2015

**User Story #856- Configure VIP server for development**

As a developer, I would like to install the software architecture stack onto the development server, so that developers have a server to add accepted user stories to.

Acceptance Criteria:

1. MEAN stack
2. System Diagram
3. Installation Guide

## 

## 

## 

## Pending User Stories

**User Story #927- Change Forgotten Password**

As a user, I would like to be able to change my password through my registered email, so that I may be able to access my account if I forgot my current password.

Acceptance Criteria:

1. When the registered user forgets its password, the system will ask for a valid email address
2. If the user enters a valid and also registered email address, the system will send a link to that email to start the process of changing the password. This message will be shown: “Please check further instructions in your email”
3. If the user enters an email address that is valid but not registered in the database, the system will say: “This email is not registered in the system”.
4. If the email address is invalid, the system will say: “Please enter a valid email”
5. The link sent to the email to start the process of changing the password will be valid for one hour if it was not touched.
6. If the user changes its password successfully, the link will not be valid anymore.
7. The password will have the same constraints as the ones in the “Register User” #717 user story
   1. Only Strong Passwords are allowed.
      * 1. Password should have at least 8 characters.
      * 2. Password should have
   2. i) at least one upper case letter (A – Z).
   3. ii) at least one lower case letter(a-z).
   4. iii) At least one digit (0 – 9) .
   5. iv) at least one special Characters of !@#$%&\*()

## Use Case

**Use Case ID:(VIP-927) Change Forgotten Password**

Details:

Actor: User

Pre-conditions:

2. User have successfully navigated to <http://vip-dev.cis.fiu.edu/#/>
4. User clicks on the blue icon in order to login

Description:

2. Use case begins when user clicks on “Forgotten your Password?” link, which is located in the bottom left part of the login window form.
4. Clicking on that link will redirect you to another page in order to reset your password.
6. The user will enter the registered email in order to receive the instructions to reset his/her password.
8. When the user clicks on the “Continue” button, the email will be sent to start the process of changing the password.
10. The user open his/her email and clicks on the link that was sent to his/her.
12. When the user clicks on the link, it will be redirect to the VIP page again in order to change the password.
14. User enters new password and retype again the new password to confirm that those match perfectly.
16. The new password will be stored in the database.

Post-conditions:

2. The password will be updated with the new information.

Alternative Course of Action

2. In step D.3, the user has to enter a valid and registered email
4. In step D.6, the user has to enter a strong password and also meet the requirements of entering the new password.

Constraints:

2. Usability:
3. 2. No training time required.
   4. The user should be able to use this case easily.
5. Reliability:
6. 2. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   4. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
8. Performance:
9. 2. Request for changing the password should be done immediately after being requested.
   4. The system should be able to handle as many forgotten password requests as the user needs.
11. Supportability:
12. 2. The link for reset the password will be corrected display in Chrome, IE, Firefox.

Modification History:

Owner: Victoriano Vega.

Initiation date: 04/21/2016

Date last modified: 04/22/2016

**User Story #915- Check Access Matrix**

As a developer, I would like to have an access matrix so that I know what user roles have access to different pages of the website.

Acceptance Criteria:

1. <https://docs.google.com/a/fiu.edu/spreadsheets/d/1RaRq3gyIgpkD_z0gslkeYlh3vXLB7yYfSmQGLWMFc0Q/edit?usp=sharing>

**User Story #912-Associate Hidden Tags with Left Sidebar in VIP My Teams page**

As a user, I would like to click on a tag in the VIP teams page so that I may see all the projects relating to that tag.

Acceptance Criteria:

1. Show only tags with listed projects
2. My Projects Tag – for logged in user
3. Should behave like People-Students/Judges Filter in Mobile Judge
4. Shows only active semester information

**User Story #911-Create Detail View page for the projects**

As a user i would like to view detailed information about a project when i click on it.

Acceptance Criteria:

1. The page should display the clicked project information.
2. The page should only display the activities associated with the clicked project.
3. User matrix access restrictions to features should be implemented.

**User Story #911-Create Detail View page for the projects**

As a user i would like to view detailed information about a project when i click on it.

Acceptance Criteria:

1. The page should display the clicked project information.
2. The page should only display the activities associated with the clicked project.
3. User matrix access restrictions to features should be implemented.

**User Story #848-View My VIP Project Page**

As a staff, I would like to view my VIP Teams/Projects page so that I can see all of the projects that have been accepted into the VIP platform.

Acceptance Criteria:

1. Main content should have a list of all accepted projects in VIP.
2. <https://docs.google.com/document/d/1MMHHyYt8vWC0fTwbb7rZ_KzZgyPuhmygsiEa117sG4Y/edit>
3. Each project should have a Name, Picture, and one sentence description associated with it. (Note, the plus sign is to be removed and the default should be Name, Picture, and Sentence).
4. Each project also has a hidden tag associated with it that relates to the tags on the left sidebar.
5. This should have the Projects that the student have applied to or worked on.

## Use Case

Use Case ID:(VIP-848-1) - Guest access Teams page

**Details**:

**Actor**: User John Smith

**Pre-conditions**:

1. User is logged in.
2. User is registered to at least one team.
3. User is in website.

**Description**:

1. Use case begins when user clicks on the TEAMs link in the nav bar.
2. The system shall then redirect the user to the TEAMs page.
3. The system shall verify the user and request all the teams the user is registered to.
4. The system shall present the user with a list of all the teams the user is registered to.
5. The use case ends when the user is page has been loaded with the teams the user is registered to.

**Post-conditions**: N/A

**Alternative Course of Action**:

1. The user clicks on a different link
2. The system shall then display the new page.

**Exceptions**: N/A

**Related Use Case**: N/A

**Decision Support**

**Frequency**: Used seldom to check the teams the user is registered to.

**Criticality**: Low allows users to see the teams they registered to.

**Risk**: Low. It uses simple authentication and data fetch operations.

**Constraints**

**Usability**:

1. No training required. It should take less than a minute to access the page.

**Reliability**:

1. System should be able to authenticate the user and request the data in a timely manner

**Performance**:

1. System should be able to gather information and load the response in one minute max.

**Supportability**

**Adaptability**:

1. Should work on devices of multiple screen sizes.

**Maintainability**:

1. Changes to the use case should be easy to perform and should be reflected on all the devices supported.

**Modification History**:

*Owner*: Rodolfo Viant Rodriguez

*Initiation Date*: 01/28/2016.

*Termination Date*: 02/1/2016.

**User Story #846-Apply For Existing Projects**

As a student i would like to be able to apply to a VIP Project.

Acceptance Criteria:

1. If student not logged in -> log in.
2. Once logged in the system will show the content of the student status to confirm its updated.
3. Notification is sent via Email to the VIP PI and Coordinator, add link to email to redirect to application.
4. Once they log into the system the application will appear in the VIP PI and Coordinator.

## Use Case

**Use Case ID:(VIP-846 ) Apply for Existing Project**

Details:

Actor: Student

Pre-conditions:

1. User has to be on application Page

Description:

1. Use case begins when user clicks on Apply button on Apply tab
2. VIP will fetch user information and all projects in database
3. VIP will display form with current user information and projects
4. Student will select project and update any of his information if needed
5. User will click Save&Submit button
6. VIP will update user model
7. VIP will send email notification about application

Post-conditions:

1. User will be notified that his application has been submitted and navigated to home page

Alternative Courses of Action:

1. User clicks on apply to project on project page
2. VIP will populate project dropdown input field with project selected by Student

Exceptions:

N/A

Related Uses Case:

1. US #848 - View my VIP project page
2. US #745 - View login Page

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: High. Allows the student to apply to VIP project.

Risk: Low. Implementing this use case employs standard web-based technology.

Constraints:

1. Usability:
   1. No training time  required.
   2. User guide should provide more in-depth information on Docker if necessary.
2. Reliability:
   1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   1. Web Page should be loaded on a broadband connection (>5Mbps) in 1 second or less
4. Supportability:
   1. Should work with current versions of most popular browsers.

Modification History:

Owner: Miguel Conde.

Initiation date: 03/28/2016

Date last modified: 04/01/2016

**User Story #746-Manage Profile**

As a user, I would like to have a profile page which I can access/edit information of my account, so that other students/faculty can view my information.

Acceptance Criteria:

1. Be able to edit account information and store when done.
2. Student is able to view their own profile and other students/faculty can view students profile as well.
3. Profile will retrieve information from registration form and registration form.
4. Look and Feel of Profile Page and Registration Page need to be the same.
5. Add “Major” Field. For students only. Faculty should not see Major field.
6. Students and Faculty cannot change their user type.
7. Students cannot change the email address.

## Use Case

**Use Case ID: VIP746 - View Website Header**

Details:

Actor: User

Pre-conditions:

1. User has logged in

Description:

1. Use case begins when the user enters his/her profile page
2. The user is able to change his/her information
3. System will be able to save the updated changes
4. Use case ends when user clicks save

Post-conditions:

1. n/a

Alternative Courses of Action:

1. n/a

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add/edit data into existing app function.

Risk: Low. Implementing this use case requires Member to know Angular, HTML, CSS

Constraints:

1. Usability:
   1. No training time required.
2. Reliability:
   1. Mean time to Failure – 1% failure is acceptable.
   2. Availability – Always available.
3. Performance:
   1. Response time - system should react to a user input within 100ms - 500ms.
   2. Throughput - the system should accomplish the work within 50ms.
   3. Availability - always available unless the system is down for maintenance.
4. Supportability:
   1. Feature must appear on mobile and non-mobile devices.

Modification History:

Owner: Marlon Rowe & Tiago moore & Rodolfo Viant

Initiation date: 03/10/2015

Date last modified: 04/04/2015

**User Story #745-Login**

As a user, I would like to log in to the VIP system.

Acceptance Criteria:

1. Student with FIU email and password.
2. Google OAuth2 API should handle password.
3. If successful user sent to their profile page/ homepage.
4. If unsuccessful an alert is given to the guest that information is wrong.
5. Only users with “Approved” account can login to the system.
6. Faculty/ Staff. PI/CoPI will register and use the system.

**User Story #717-Register User**

As a Guest User I would like to be able to register so that I can use the systems features.

Acceptance Criteria:

* Create a User Registration Page for new users to be added to our user database.
* Roles include: Student, Faculty/Staff- PI and Co/PI will be added by an admin.
* A Student does not register, he will interact with the system using his Gmail account.
* The information must be checked for authenticity (@fiu.edu email account etc). Every form should field a validation requirement.
* PI/CoPI does not require an FIU email. Email should still be validated for a valid email addresss. i.e. containing @.
* As soon as student clicks on Student user type - should be alerted that he cannot register.
* For panther ID, the input field should not allow user to input letters- only numbers are accepted.
* The user should receive an email indicating the successful creation of the account and a link to confirm the email address: “Please verify your email and wait for your account to be verified by the PI.”
* After the user has confirmed the email, an Email should be send to anyone with PI/CoPi as user type and PI/CoPi/Coordinator with a link indicating “account needs to be approved” to their TODO list.
* If the user tries to login at this point, the user would receive a message that the account needs to be confirmed by the PI: “ Your account needs to be confirmed by the PI”
* If the Email has not being confirmed the account can not be confirmed.
* User has the following stats: Guest User, Pending User, Pending user with Confirmed Email, Approved User (a.k.a, registered user or active user), Rejected User (See the State Chart Diagram).
* Only Strong Passwords are allowed.
  + 1. Password should have at least 8 characters.
  + Password should have

i) at least one upper case letter (A – Z).

ii) at least one lower case letter(a-z).

iii) At least one digit (0 – 9) .

iv) at least one special Characters of !@#$%&\*()

* Pending User with Confirmed Email should be in the todo list of all users with PI/CoPI/Coordinator rank until the user is approved or rejected. After that, it should be removed from the TODO list.
* Only the Approved User can proceed to the next feature of our application – the login to the system.

## Use Case

**Use Case ID:(VIP-717) Register User**

Details:

Actor: User

Pre-conditions:

1. User has navigated to VIP website

Description:

1. Use case begins when user clicks Login button.
2. Login page will have create account button.
3. User will click on button and register for an account.

Post-conditions:

1. User is on registration page.
2. User does not have a previous account .

Alternative Courses of Action:

N/A

Exceptions:

N/A

Related Uses Case:

N/A

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Medium. Allows the Member to add data into the social aspect integration features.

Risk: Low. Implementing this

Constraints:

1. Usability:
   1. No training time  required.
   2. User guide should provide more in-depth information on Docker if necessary.
2. Reliability:
   1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   1. The build should be triggered within 20 seconds
4. Supportability:
   1. The docker build should support any branch on the repository

Modification History:

Owner: Tiago Moore.

Initiation date: 03/20/2016

Date last modified: 4/11/2016

**User Story #693-Propose a new Project**

As a staff or faculty member I would like to be able to submit a new project proposal so that it can be reviewed, accepted, and rejected.

Acceptance Criteria:

1. The proposal should be a form

* The PI/Coordinator should receive an email with a link to the application review page and it should be included in the TO-Do list.
* Form Fields Include title, short description, disciplines (projects have to be multi-disciplinary), number of students for the first semester, max number of students per semester.

-Staff/Faculty can check the status of their pending proposal(s) on the website

1. Edit Proposal should be enabled.
2. Staff/Faculty can receive notifications about changes to approval status
   * Notifications are delivered through email, and/or website upon login.

UPDATED CRITERIA:

1. Make project description an html editor
2. All users with user type PI/Co-PI and Rank of PI, Co-PI, and Coordinator will get a notification for a project
3. When notification is sent, it will contain link to project, and admins will have a to-do list containing the pending project
4. Project must be in proposed, rejected or accepted state
5. When project is edited by PI or Co-pi and will send a notification when editing occurs
6. Disciplines/Majors need to be Collapsing Buttons

## Use Case

**Use Case ID:(VIP-693) Implement Project Proposal**

Details:

Actor: User

Pre-conditions:

1. User has navigated to VIP website

Description:

1. Use case begins when user clicks faculty/propose project
2. User will be returned a form containing required and optional fields for a project
3. Docker will pull the repository using the latest commit

Post-conditions:

1. Project proposal is created

Alternative Courses of Action:

1. User clicks edit on a project detailed page
2. Project proposal form is returned with data pre-filled from selected project
3. Certain fields cannot be changed when editing a project(Title)

Exceptions:

N/A

Related Uses Case:

N/A

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Medium. Allows the Member to add data into the social aspect integration features.

Risk: Low. Implementing this use case employs standard web-based technology.

Constraints:

1. Usability:
   1. No training time  required.
   2. User guide should provide more in-depth information on Docker if necessary.
2. Reliability:
   1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   1. The build should be triggered within 20 seconds
4. Supportability:
   1. The docker build should support any branch on the repository

Modification History:

Owner: Andres Villa.

Initiation date: 03/09/2016

Date last modified: 03/09/2016

**User Story #928-Review User Registration**

As a PI/Coordinator I should be able to review a users registration so that I can accept or reject a registration request.

Acceptance Criteria:

1. All PI/CoPI should receive an email confirmation per application with a link a registration.
2. If the Faculty reviews the registration the user will get a notification informing the application been accepted and a welcome message should be sent via email.
3. The PI/CoPI should be linked to the users profile page and there should be an Accept/Reject option at the bottom.

**User Story #908-Review Project proposal**

As a PI/Co-PI/Coordinator I should be able to review a new project proposal, so that I can review, approve or reject a new project proposal.

Acceptance Criteria:

1. All users with User Type of PI/Co-PI and Rank of PI/Co-PI/Coordinator (note that External Member rank is not included) must receive an email notification regarding the submission of a new project proposal with a link to project proposal.
2. If the status of the new project proposal is Pending, this new project proposal must be included in the TO-DO list of all the users with User Type of PI/Co-PI and Rank of PI/Co-PI/Coordinator.
3. If the status of the new project proposal is no longer Pending (meaning that one of the users with rank of PI/Co-PI/Coordinator has approved or rejected the proposal), then the new project proposal must NOT appear in an of the corresponding users’ TO-DO list.
4. The PI/Co-PI/Coordinator who reviews the project proposal should be able to enter a text regarding the reasons for the decision that was made and the steps to address the issues and/or to guide the applicant of the next steps of the process.
5. When the status of the new project proposal changes, the Faculty/Staff who submitted the proposal will get an email notification (including the project proposal, the decision, the explanation, and the link to the proposal) informing the Faculty/Staff user of the status of the application.
6. If the application is approved/rejected by a PI/Co-PI/Coordinator then the Lead Project Faculty/Staff member who proposed the project will receive an email notification with the above mentioned information that includes a link to the proposal and should be added to the TO-DO list of the Faculty/Staff member. After the project proposal is visited by the Faculty/Staff member, it should no longer be in the TO-DO list.

**User Story #907-Review Student Application**

As a PI/Co-PI/Coordinator, I would like to be informed when a student applies to a project, so that I can review, approve or reject the application. Also, as a Faculty/Staff member who have proposed this project, I would like to be informed when a student application is approved by a PI/Co-PI/Coordinator, so that I can review, approve, reject the application.

Acceptance Criteria:

1. All users with User Type of PI/Co-PI and Rank of PI/Co-PI/Coordinator (note that External Member rank is not included) must receive an email notification regarding the submission of a new student application to join an approved project with a link to the student application.
2. If the status of the student application is Pending, this student application must be included in the TO-DO list of all the users with User Type of PI/Co-PI and Rank of PI/Co-PI/Coordinator.
3. If the status of the student application is no longer Pending (meaning that one of the users with rank of PI/Co-PI/Coordinator has approved or rejected the application), then the student application must NOT appear in an of the corresponding users’ TO-DO list.
4. The PI/Co-PI/Coordinator who reviews the application should be able to enter a text regarding the reasons for the decision that was made and the steps to address the issues and/or to guide the applicant of the next steps of the process.
5. When the status of the student application is approved by a PI/Co-PI/Coordinator, the lead Faculty/Staff of the requested project will get an email notification (including the application, the decision, the explanation, and a link to the application) informing the Faculty/Staff user that a new student application is awaiting their review. Also, the student must receive an email notification with all the above information too Both the Faculty/Staff member and student should see this in their TO-DO list until they review or visit the application, respectively.
6. The Faculty/Staff member should be able to enter a text regarding the reasons for the decision that was made and the steps to address the issues and/or to guide the applicant of the next steps of the process.
7. If the application is approved/rejected by the Lead Project Faculty/Staff member, it should no longer be included in his/her TO-DO list, and the student will receive an email notification that includes the application, decision, the explanation, and the link to the application and should be added to the TO-DO list of the student too. After the student application is visited by the student, it should no longer be in the TO-DO list.
8. If the student application is approved by the Lead Faculty/Staff member, in the student notification When Faculty approves the student should receive an email with instructions on how to register to the VIP Course.

# Project Plan

The following Project Plan provides the organization and requirements for the production, and development of the VIP2 application. This section specifies the roles of the members of the team and hardware and software requirements for developing the software. The Project Plan provides a general framework and establishes specific strategies and milestones for the development and production of the VIP2.

## Hardware and Software Resources

The development stack that was chosen for this product was the MEAN (Mongo, Express, Angular, Node) stack. This uses lightweight frameworks such as Angular and Node and is currently one of the leading stacks used in the industry for website development. It relies almost exclusively one open-source contributions from the community, which provides the team and abundance of resources to use while developing this product at no extra cost. Due to it’s popularity, it is also one of the most widely discussed stacks on open forums, which enables us as developers to seek help from the community when we are blocked on a specific problem.

Angular also provides us with the tools needed to make this application dynamic. With frameworks such as bootstrap and other angular libraries, that transition from desktop, mobile and tablet is extremely easy to implement, which removes having to do double the work (write code for mobile and desktop separately).

Hardware Requirements

|  |  |
| --- | --- |
| **System** | **Minimum Requirement** |
| Operating system | Ubuntu Server |
| Processor |  |
| Speed | 2.8 GHz or faster |
| RAM | 1 GB |

Client

- CPU - 1.5 GHz, Dual-Core

- RAM - 512 MB

- Hard Drive Space - 500 MB

- Mozilla Firefox

- Google Chrome

- Safari

**Sprint 1 Planning**

**Date: 01/22/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Get familiar with Sencha development framework by implementing a feature into the Mobile Judge application.

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story #547 Implement Docker CI
* User Story #519 Link Login Accounts
* User Story #520 Manage Profile
* User Story #524 Manage List Show Size
* User Story #542 Add Check Box on Filters for People Selection
* User Story #525 Review Grades
* User Story #526 SEnd Mass Emails from Mobile Device
* User Story #548 Export Grades
* User Story #544 Manage Student
* User Story #543 Manage Judge
* User Story #522 Export Judge List
* User Story #521 Export Student List
* User Story #529 Add Device Compatibility
* User Story #35 Delete Term

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore

-User Story: #542 Add Check Box on Filters for People Selection

Tasks :

* + - Use Case(1)
    - Sequence Diagram(2)
    - Class Diagram(2)
    - Implementation(30)
    - Testing(3)
    - Integration(2)
* Student 2: Miguel Conde

- User Story - #523 - Add Search Feature to Email Wizard

* + - Tasks
    - Sequence Diagram
    - Class Diagram
* Student 3: Jorge Perez
  + - User Story - #524 - User List Show Sizes
    - Tasks
    - Sequence Diagram
    - Class Diagram
* Student 4: Marlon Rowe
  + -User Story - # 520 - Manage Profile
    - Tasks
    - Sequence Diagram
    - Class Diagram
    - Implementation
* Student 5: Victoriano Vega
  + - User Story - # - Export Grades
* Student 6: Rodolfo Viant
  + - User Story - #525 - Review Grades
    - Tasks
    - Sequence Diagram
    - Class Diagram
    - Use Case Diagram
* Student 7: Andres Villa

-User Story: #547 Implement Docker CI

Tasks :

* + - Use Case(5)
    - Sequence Diagram(5)
    - Class Diagram(3)
    - Implementation(16)
    - Testing(3)
    - Integration(2)

----------------------------------------------------------------------------------------------------------------------------

**Sprint 2 Planning**

**Date: 2/1/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Finish implementing the user stories requested by the product owner in Sprint 1. Finalize testing and integration for product release.

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #547 Implement Docker CI
* User Story #519 Link Login Accounts
* User Story #520 Manage Profile
* User Story #524 Manage List Show Size (Carry over)
* User Story #542 Add Check Box on Filters for People Selection
* User Story #525 Review Grades
* User Story #526 SEnd Mass Emails from Mobile Device
* User Story #548 Export Grades
* User Story #544 Manage Student
* User Story #543 Manage Judge
* User Story #522 Export Judge List
* User Story #521 Export Student List
* User Story #529 Add Device Compatibility
* User Story #35 Delete Term

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story: #542 Add Check Box on Filters for People Selection

Tasks :

* + - Generate Test Cases
    - Create technology depth to refactor directory structures
    - Framework Feasibility Study
    - Add screenshots to user guide
    - Standardize group diagrams
* Student 2: Miguel Conde
  + User Story - #523 - Add Search Feature to Email Wizard
* Student 3: Jorge Perez
  + User Story - #524 - User List Show Sizes
    - Continue Trying to implement.
    - Generate Test Cases
    - Add Screenshots to User Guide
    - Update Docs to match Team’s Documentation Theme
* Student 4: Marlon Rowe
  + User Story - # 520 - Manage Profile
* Student 5: Victoriano Vega
  + User Story - # - Export Grades
* Student 6: Rodolfo Viant
  + User Story - #525 - Review Grades
* Student 7: Andres Villa
  + User Story: #547 Implement Docker CI

Tasks :

----------------------------------------------------------------------------------------------------------------------------

**Sprint 3 Planning**

**Date: 2/15/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Enable user registration, login,homepage ,and form submission.

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #623 Visualize Account State
* User Story #622 Unlink Accounts
* User Story #525 Review Grades
* User Story #602 Testing MJ 8.0
* User Story #619 Setup PM2 Server Monitoring
* User Story #543 Manage Judge
* User Story #544 Manage Students
* User Story #519 Link Login
* User Story #618 Change Judge Password

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story #623 Visualize Accounts
  + User Story #519 Link Login
* Student 2: Victoriano Vega
  + User Story #602 Testing MJ 8.0
* Student 3: Marlon Rowe
  + User Story #543 Manage Judge
  + User Story #544 Manage Student
* Student 4: Jorge Perez
  + User Story #619 Setup PM2 Server Monitoring
* Student 5: Andres Villa
  + User Story #622 Unlink Accounts
* Student 6: Miguel Conde
  + User Story #523 Add Search Filter to Email Wizard (Bug Fix)
* Student 7: Rodolfo Viant
  + User Story #525 Review Grades

----------------------------------------------------------------------------------------------------------------------------

**Sprint 4 Planning**

**Date: 2/29/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Setup Development Environment for MEAN stack application, Put Static pages to play with new environment

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #840 View Website Header
* User Story #844 View Page Footer
* User Story #842 View Contact Page
* User Story #843 View Main Page
* User Story #847 View How VIP Credits Count
* User Story #854 View VIP Competitions
* User Story #856 Configure VIP Server for Development
* User Story #855 Write Database Schema
* User Story #841 View About VIP Page
* User Story #848 View My VIP Projects Page

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story #844 View Page Footer
* Student 2: Victoriano Vega
  + User Story #841 View VIP Page
* Student 3: Marlon Rowe
  + User Story #843 View Main Page
* Student 4: Jorge Perez
  + User Story #847 View How VIP Credits Count
  + User Story #855 Write Database Schema
* Student 5: Andres Villa
  + User Story #842 View Contact Page
  + User Story #854 View VIP Competitions
* Student 6: Miguel Conde
  + User Story #856 Configure VIP Server for Development
* Student 7: Rodolfo Viant
  + User Story #848 View My VIP Projects Page

----------------------------------------------------------------------------------------------------------------------------

**Sprint 5 Planning**

**Date: 3/21/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Get the rest of the Static Pages done, Start working on Dynamic pages

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #840 View Website Header
* User Story #841 View About VIP Page
* User Story #906 View VIP Organization Page
* User Story #853 View VIP Syllabi Information
* User Story #894 View Class Days, Time, and Location
* User Story #899 View Presentation & Publications
* User Story #851 View Peer Evaluation Page
* User Story #893 View Permit Request Form
* User Story #895 View Apply Undergraduate Page
* User Story #895 View Apply Graduate Page
* User Story #902 View Classroom Schedules Page
* User Story #848 View My VIP Projects Page
* User Story #912 Associate Hidden Tags with Left Sidebar in My VIP Teams Page
* User Story #911 Create Detail View Page for Projects
* User Story #915 Check Access Matrix
* User Story #717 Register User
* User Story #745 Login

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story #717 Register User
* Student 2: Victoriano Vega
  + User Story #841 View About VIP Page
  + User Story #853 View VIP Syllabi Information
* Student 3: Marlon Rowe
  + User Story #840 View Website Header
  + User Story #899 View Presentations & Publications
* Student 4: Jorge Perez
  + User Story #745 Login
  + User Story #906 View VIP Organization Page
  + User Story #894 View Class Days, Time, and Location
  + User Story #915 Access Matrix
* Student 5: Andres Villa
  + User Story #895 View Apply Undergraduate Page
  + User Story #896 View VIP Apply Graduates Page
* Student 6: Miguel Conde
  + User Story #851 View Peer Evaluation Page
  + User Story #893 View Permit Request Form
  + User Story #902 View Classroom Schedule Page
* Student 7: Rodolfo Viant
  + User Story #848 View My VIP Projects Page
  + User Story #912 Associate Hidden Tags with Left Sidebar in My VIP Teams Page
  + User Story #911 Create Detail View Page for Projects

----------------------------------------------------------------------------------------------------------------------------

**Sprint 6 Planning**

**Date: 4/4/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Work on the dynamic aspects of the website

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #913 Redesign Components of the Website
* User Story #672 Receive Project Proposal
* User Story #693 Propose a new Project
* User Story #717 Register User
* User Story #745 Login
* User Story #846 Apply for Existing Projects
* User Story #746 Manage Profile
* User Story #907 Review Student Applications
* User Story #914 View ToDo List
* User Story #908 Review Project Proposal
* User Story #915 Check Access Matrix
* User Story #912 Associate Hidden Tags With Left Sidebar in VIP My Teams Page
* User Story #911 Create Detail view page for projects
* User Story #848 View My VIP Projects Page
* User Story #927 Change Forgotten Password
* User Story #928 Review User Registration
* User Story #929 Create Term CRUD operations

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story #717 Register User
  + User Story #746 Manage Profile
  + User Story #928 Review User Registration
* Student 2: Victoriano Vega
  + User Story #672 Receive Project Proposal
  + User Story #927 Change Forgotten Password
* Student 3: Marlon Rowe
  + User Story #913 Redesign Components of the Website
  + User Story #914 View ToDo List
* Student 4: Jorge Perez
  + User Story #745 Login
  + User Story #915 Check Access Matrix
* Student 5: Andres Villa
  + User Story #693 Propose a new Project
* Student 6: Miguel Conde
  + User Story #846 Apply for Existing Projects
* Student 7: Rodolfo Viant
  + User Story #912 Associate Hidden Tags With Left Sidebar in VIP My Teams Page
  + User Story #911 Create Detail view page for projects
  + User Story #848 View My VIP Projects Page

----------------------------------------------------------------------------------------------------------------------------

**Sprint 7 Planning**

**Date: 4/18/2016**

**Attendees:Tiago,Rodolfo,Andres,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

Sprint goal: Finish implementations and solve bugs, Work on documentations

After discussion, the velocity of the team were estimated to be \_\_\_\_.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority:

* User Story #693 Propose a new Project
* User Story #717 Register User
* User Story #745 Login
* User Story #846 Apply for Existing Projects
* User Story #746 Manage Profile
* User Story #907 Review Student Applications
* User Story #914 View ToDo List
* User Story #908 Review Project Proposal
* User Story #915 Check Access Matrix
* User Story #912 Associate Hidden Tags With Left Sidebar in VIP My Teams Page
* User Story #911 Create Detail view page for projects
* User Story #848 View My VIP Projects Page
* User Story #927 Change Forgotten Password
* User Story #928 Review User Registration
* User Story #929 Create Term CRUD operations

The team members indicated their willingness to work on the following user stories.

* Student 1: Tiago Moore
  + User Story #717 Register User
  + User Story #746 Manage Profile
  + User Story #928 Review User Registration
* Student 2: Victoriano Vega
  + User Story #927 Change Forgotten Password
* Student 3: Marlon Rowe
  + User Story #914 View ToDo List
* Student 4: Jorge Perez
  + User Story #745 Login
  + User Story #915 Check Access Matrix
* Student 5: Andres Villa
  + User Story #693 Propose a new Project
* Student 6: Miguel Conde
  + User Story #846 Apply for Existing Projects
* Student 7: Rodolfo Viant
  + User Story #912 Associate Hidden Tags With Left Sidebar in VIP My Teams Page
  + User Story #911 Create Detail view page for projects
  + User Story #848 View My VIP Projects Page

# 

# 

# 

# System Design

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

The Three-Tier pattern was chosen because the software system is a web application, so that the web pages and views presented to each user naturally decompose into the Presentation layer, the functions executed on the server to carry out the user's’ commands are easily encapsulated within the Application Logic layer, and the persistent data stored about the users and the map clearly corresponds to the Data Management layer.

The Client/Server Pattern was chosen as a secondary pattern to reflect the fact that many users will connect to one central server which will handle requests by the users to modify or display data stored on the server’s central data repository. Users (clients) will interact with the system (server) by requesting searches and queries on the central data, and the system will serve each client’s request by executing the application logic and interacting with the database.

## Deployment Diagram

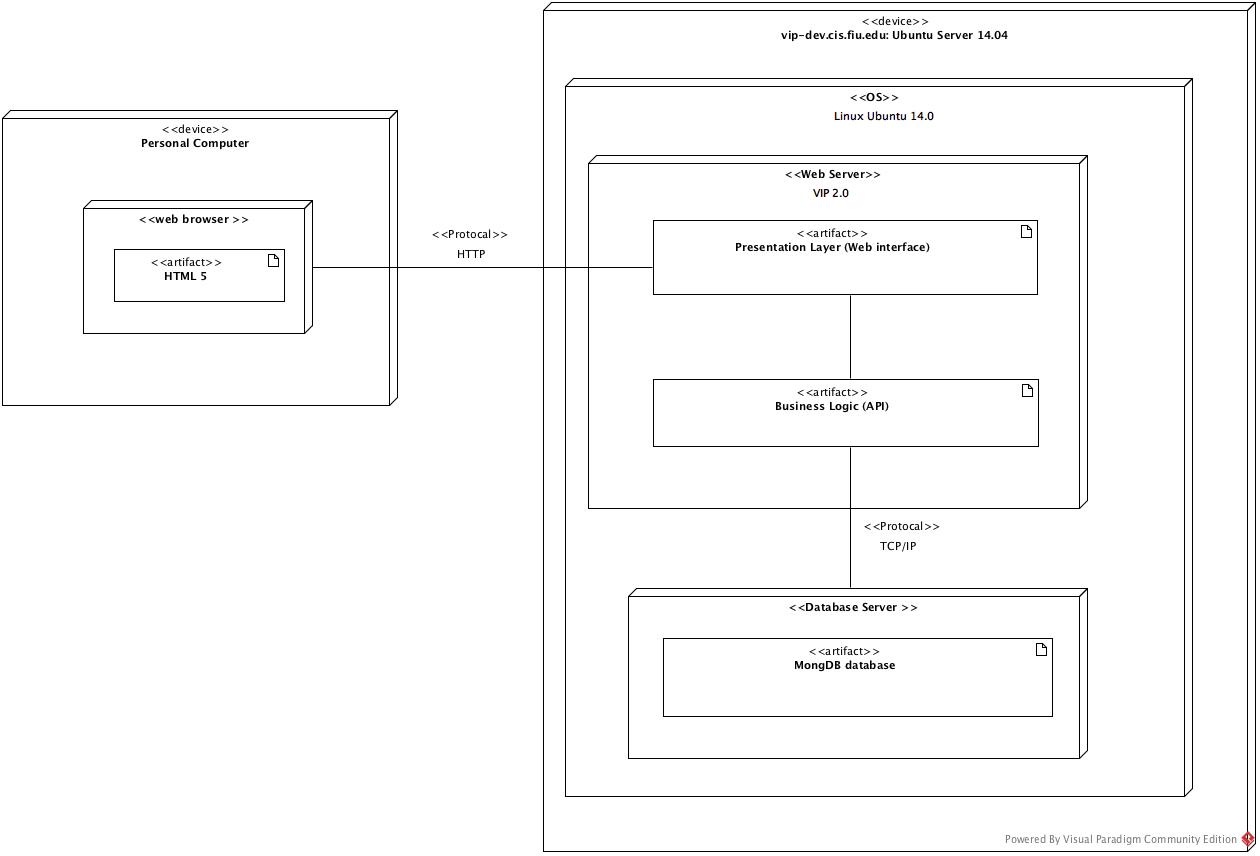


Figure 2 - Deployment Diagram

## Design Patterns

The Main design pattern used in VIP 2.0 was the MVC pattern. It allowed for separation of the data on the server from the data that was being presented to the user. It also ensures the data provided by the user can be checked and sanitized before a request is made.

The main design pattern used in Mobile Judge was the MVVM pattern. MVVM bypasses the use of a controller in MVC. MVVM facilitates the use of data binding. Much of the data in Mobile Judge due in part to Sencha binds data from the view to the viewmodel, and calls event handlers built into the framework. So MVVM allowed the data to be passed to the view using the built in functionality of Sencha.

# System Validation

This section describes the tests used to verify and validate the functional and nonfunctional requirements of the proposed system. System Tests check the correct implementation for the use cases developed for the system. Subsystem Tests are provided for validating the outlined subsystem. The Evaluation of Tests provide results of the system and subsystem tests. Tools and Purpose explain which testing tools were used and the reasons for picking those tools.

**ID: MJ-SD-522-01**

**Purpose**

* Test to see if export button returns csv file

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on judged tab
* The database contains judge information
* User has clicked the export button

**Input**

* A click activated by the user on the button.

**Expected Result**

* All filters should be selected.
* A CSV file with all judge information.

**Actual Result**

* A CSV file with all judge information.

**ID: MJ-SD-542-02**

**Purpose**

* Test if checkbox returned selected list of judges with filter applied .

**Precondition**

* User has logged on mobile judge.
* User has clicked on the People tab.
* User has clicked on judged tab.
* The database contains student information.
* User has clicked the export button.

**Input**

* Admin clicks on the export judges button.

**Expected Result**

* A CSV file with selected judges should be returned

**Actual Result**

* All judges were returned FAILED

**ID: MJ-RD-522-01**

**Purpose**

* Test if CSV file with judges is returned when there is an empty database.

**Precondition**

* User has logged on mobile judge.
* User has clicked on the People tab.
* User has clicked on judged tab.
* The database is empty.
* User has clicked the export button.

**Input**

* Admin clicks on the export judges button.

**Expected Output**

* All filters should be selected
* Grid should be populated

**Actual Result**

* All filters were selected
* Grid was populated

**ID: MJ-SD-542-01**

**Purpose**

* Test if checkbox selects all student filters

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* The database contains student information
* Check box is in the unchecked state

**Input**

* User clicks on the checkbox with all the filters selected

**Expected Result**

* All filters should be selected
* The grid should be populated with all the students

**Actual Result**

* Grid was populated with all the students

**ID: MJ-SD-542-02**

**Purpose**

* Test if checkbox selects all student filters when one or more filters unchecked

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* The database contains student information
* One or more filters are unchecked
* Check box is in the unchecked state

**Input**

* User clicks on the checkbox with all the filters selected

**Expected Result**

* All filters should be selected
* The grid should be populated with all the students

**Actual Result**

* All filters were selected
* Grid was populated with all the students

**ID: MJ-SD-542-03**

**Purpose**

* Test if checkbox unselects all filters when or more filters unchecked

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* The database contains student information
* Check box is in the checked state

**Input**

* User clicks on the checkbox with or more filters unchecked

**Expected Result**

* All filters should be unselected
* The grid show be empty

**Actual Result**

* All filters were unselected
* Grid is empty

**ID: MJ-SD-542-04**

**Purpose**

* Test if checkbox unselects all filters when all filters are checked

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* The database contains student information
* Check box is in the checked state

**Input**

* User clicks on the checkbox with all the filters selected

**Expected Result**

* 404 Error

**Actual Result**

* 404 Error

**ID: MJ-RD-542-04**

**Purpose**

* Test if checkbox selects all filters when loss of internet connection

**Precondition**

* User has logged on mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* The database contains student information
* Check box is in the checked state
* No internet connection

**Input**

* User clicked on checkbox when selected

**Expected Output**

* All filters should be selected
* Grid should be populated

**Actual Result**

* All filters were selected
* Grid was populated

**ID: MJ-SD-520-01**

**Purpose**

* Test if user can save new information

**Precondition**

* User has logged into Mobile Judge
* User went to his/her profile page by clicking on his/her picture or name

**Expected Result**

* User was able to save all information on the form

**Actual Result**

* User was able to save all information on the form to the database

**ID: MJ-RD-520-01**

**Purpose**

* Test if user can save first name, last name or email with empty fields

**Precondition**

* User has logged into Mobile Judge
* User went to his/her profile page by clicking on his/her picture or name

**Expected Result**

* User was not able to save first name, last name, or email fields information to database

**Actual Result**

* User was able to save all information on the form to the database

**ID: MJ-SD-543-01**

**Purpose**

* Test if admin can modify judge information

**Precondition**

* User has logged into Mobile Judge as a admin
* User navigated to People->Judges page

**Expected Result**

* User was able to save changed information

**Actual Result**

* User was able to save the information he/she changed on the judge

**ID: MJ-SD-543-02**

**Purpose**

* Test if admin can update a judge's email that already exists in the database

**Precondition**

* User has logged into Mobile Judge as a admin
* User navigated to People->Judges page

**Expected Result**

* User was not able to change judges email due to it already existing in the database

**Actual Result**

* User was not able to change judges email due to it already existing in the database

**ID: MJ-RD-543-01**

**Purpose**

* Test if user can remove title and/or affiliation, leaving the field blank

**Precondition**

* User has logged into Mobile Judge as a admin
* User navigated to People->Judges page

**Expected Result**

* User was able to delete the value in the field, leaving the field blank in the database

**Actual Result**

* Judge still has previous value as before, title/affiliation did not get updated.

**ID: MJ-SD-548-01**

**Purpose**

* To test if administrator can click on the Export button and save as an csv file.

**Precondition**

* Administrator has successfully logged onto MobileJudge site.

**Input**

* Administrator clicks on the Export button and a small windows comes up.
* Administrator saves the file with the default name.
* Administrator clicks on the OK button.

**Expected Result**

* The file is saved successfully.

**Actual Result**

* The file is saved successfully.

**ID: MJ-RD-548-02**

**Purpose**

* To test if administrator can click on the Export button and save as an csv file.

**Precondition**

* Administrator has successfully logged onto MobileJudge site.

**Input**

* Administrator clicks on the Export button and a small window comes up.
* Administrator clicks on the 'Open with' button and the program select does not open csv files.
* Administrator clicks on the OK button.

**Expected Result**

* The file is saved successfully.

**Actual Result**

* File is not saved successfully because the program selected will not open the file.

**ID: MJ-SD-547-01**

**Purpose**

* Ensure that the Docker image contains the appropriate services

**Precondition**

* Docker build was completed successfully

**Input**

* docker run -P -d andresvilla/mobilejudge8:dockerBranch

**Expected Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80> loads mobile judge website

**Actual Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80> loads mobile judge website

**ID: MJ-SD-547-02**

**Purpose**

* Ensure that the Docker does not hold any true database information

**Precondition**

* Docker build was completed successfully

**Input**

* docker run -P -d andresvilla/mobilejudge8:dockerBranch

**Expected Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80>/phpmyadmin/ and logging in, shows that there is no database in phpmyadmin

**Actual Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80>/phpmyadmin/ and logging in, shows that there is no database in phpmyadmin

**ID: MJ-RD-547-01**

**Purpose**

* Ensure that Docker image does not update if a bad build is triggered

**Precondition**

* Code was successfully pushed to GitHub

**Input**

* Non functioning code has been pushed to GitHub repository

**Expected Result**

* Docker container will not update the image with the latest broken build

**Actual Result**

* Docker container will not update the image with the latest broken build

**ID: MJ-RD-547-02**

**Purpose**

* Ensure that Docker Image is independent of Docker container

**Precondition**

* Code was successfully pushed to GitHub

**Input**

* Docker container is started up
* Entire docker container is deleted (sudo rm -rf \*) from root directory

**Expected Result**

* Docker image will remain functional and identical to image prior to test

**Actual Result**

* Docker image will remain functional and identical to image prior to test

**ID: MJ-SD-623-01**

**Purpose**

* Test if checkbox is checked after linking account.

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* A user links his account.

**Input**

* User clicks social icon and enters authentication.

**Expected Result**

* A pop up window comes up for confirmation
* A user clicks yes to confirm.
* The user account social media account gets linked.
* The checkbox is checked.

**Actual Result**

* A pop up window comes up for confirmation
* A user clicks yes to confirm.
* The user account social media account gets linked.
* The checkbox is checked.

**ID: MJ-SD-623-02**

**Purpose**

* Test to see if checkbox is unchecked if user unlinks account.

**Precondition**

* User has previously linked accounts.

**Input**

* User clicks on social media icon and unlinks account.

**Expected Result**

* A pop up window comes up for confirmation .
* The social media account is unlinked.
* The previously checked box will be unchecked.

**Actual Result**

* A pop up window comes up for confirmation .
* The social media account is unlinked.
* The previously checked box will is unchecked.

**ID: MJ-SR-623-03**

**Purpose**

* Test to see if checkbox remains unchecked after incorrect user authentication.

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* User start link account process.

**Input**

* User entered incorrect login information.

**Expected Result**

* A pop up window comes for authentication.
* The checkbox is not checked.

**Actual Result**

* A pop up window comes for authentication.
* The checkbox is not checked.

**ID: MJ-SD-524-01**

**Purpose**

* Test if the ‘10’ list size button changes the list to 10 items when the default size list size is shown (25).

**Precondition**

* User is logged into the system
* User is in the People page
* User is in the Students Tab.

**Input**

* User clicks List Size Button ‘10’

**Expected Result**

* List size of students changes from 25 to 10.

**Actual Result**

* List size of students changes from 25 to 10.

**ID: MJ-SD-524-02**

**Purpose**

* Test if the ‘50’ list size button changes the list to 50 items when the default size list size is shown (25).

**Precondition**

* User is logged into the system
* User is in the People page
* User is in the Students Tab.

**Input**

* User clicks List Size Button ‘25’

**Expected Result**

* List size of students changes from 10 to 25.

**Actual Result**

* List size of students changes from 10 to 25.

**ID: MJ-SD-524-01**

**Purpose**

* Test if the list size changes when 25 is clicked and the list size is already 25.

**Precondition**

* User has logged onto mobile judge
* User has clicked on the People tab
* User has clicked on the student tab of the people page
* List size is already 25

**Input**

* User clicks the List Size Button ‘25’

**Expected Output**

* List size stays at 25

**Actual Result**

* List size stays at 25

**ID: MJ-SD-622-01**

**Purpose**

* Test if account removed works after confirmation .

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* User is linked to an account.

**Input**

* User clicks on social media icon .

**Expected Result**

* A pop up window comes up for confirmation
* A user clicks yes to confirm.
* The user account social media account gets unlinked.

**Actual Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**ID: MJ-SD-622-02**

**Purpose**

* Test if account is properly unlinked.

**Precondition**

* User has previously linked accounts.

**Input**

* User clicks on social media icon.

**Expected Result**

* A pop up window comes up for confirmation
* The social media account is unlinked.

**Actual Result**

* A pop up window comes up for confirmation
* The social media account is unlinked.

**ID: MJ-RD-622-01**

**Purpose**

* Account is not unlinked if confirmation fails

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* The user clicks no on unlink confirmation prompt

**Input**

* User clicks on social media icon
* User clicks no on confirmation prompt

**Expected Result**

* A pop up window comes for confirmation.
* The user account does not get unlinked to the social media account.

**Actual Result**

* A pop up window comes for confirmation.
* The user account does not get unlinked to the social media account.

**ID: MJ-RD-622-02**

**Purpose**

* Ensure feedback is provided if popups are disabled

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* The user clicks no on unlink confirmation prompt

**Input**

* User clicks on social media icon
* User has pop-ups blocked

**Expected Result**

* An alert comes up notifying the user that the Oauth popup could not come up
* The alert signals to the user the cause

**Actual Result**

* An alert comes up notifying the user that the Oauth popup could not come up
* The alert signals to the user the cause

**ID: MJ-SD-519-01**

**Purpose**

* Test if authentication works after icon click.

**Precondition**

* Admin has successfully logged onto the system.
* Admin is using the Profile feature.

**Input**

* User clicks on social media icon

**Expected Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**Actual Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**ID: MJ-SD-519-02**

**Purpose**

* Test if authentication works after icon click.

**Precondition**

* Admin has successfully logged onto the system.
* Admin is using the Profile feature.

**Input**

* User clicks on social media icon

**Expected Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**Actual Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**ID: MJ-SD-519-03**

**Purpose**

* Test if authentication works after icon click.

**Precondition**

* Admin has successfully logged onto the system.
* Admin is using the Profile feature.

**Input**

* User clicks on social media icon

**Expected Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**Actual Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account gets linked to the social media account.

**ID: MJ-SD-519-04**

**Purpose**

* Test if accounts are linked upon a failed authentication attempt.

**Precondition**

* User has successfully logged onto the system.
* User is using the Profile feature.
* User enters erroneous account information.

**Input**

* User clicks on social media icon
* User inputs wrong username and password

**Expected Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account does not gets linked to the social media account.

**Actual Result**

* A pop up window comes up for authentication
* A user enters his login information
* The user account does not gets linked to the social media account.

**ID: MJ-SD-525-01**

**Purpose**

* Test if change grade status for a student.

**Precondition**

* Admin has successfully logged onto the system.
* Judges have graded the students.

**Input**

* Admin clicks on green traffic light icon .

**Expected Result**

* Traffic Light changes color to red.
* Traffic Light icon tooltip changed to “Rejected”.

**Actual Result**

* Traffic Light changed color to red
* Traffic Light tooltip changed to “Rejected”

**ID: MJ-SD-523-01**

**Purpose**

* Test if Student name is found on valid search

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard

**Input**

* User types in name in the search bar and clicks enter with students grid open

**Expected Result**

* Grid returns Student record that name matches search criteria
* The student record must be selectable
* Text “(must be cleared to proceed to next) should appear in red, bold color/font
* Next button is disabled

**Actual Result**

* Grid returns Student record that name matches search criteria
* The student record is selectable
* Text “(must be cleared to proceed to next) appears in red, bold color/font
* Next button is disabled

**ID: MJ-SD-523-02**

**Purpose**

* Test if Judge name is found on valid search

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard

**Input**

* User types in name in the search bar and clicks enter with judge grid open

**Expected Result**

* Grid returns Judge record that name matches search criteria
* The judge record must be selectable
* Text “(must be cleared to proceed to next) should appear in red, bold color/font
* Next button is disabled

**Actual Result**

* Grid returns judge record that name matches search criteria
* The judge record is selectable
* Text “(must be cleared to proceed to next) appears in red, bold color/font
* Next button is disabled

**ID: MJ-SD-523-03**

**Purpose**

* Test if warning box displays when user clicks previous button

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard

**Input**

* User clicks on the previous button

**Expected Result**

* Warning message is displayed

**Actual Result**

* Warning message is displayed

**ID: MJ-SD-523-04**

**Purpose**

* Test if clicking yes on warning message navigates user to previous tab

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard
* User has clicked previous button and warning message is displayed

**Input**

* User clicks on Yes button

**Expected Result**

* User is navigated to previous tab

**Actual Result**

* User is navigated to previous tab

**ID: MJ-SD-523-05**

**Purpose**

* Test if clicking ‘x’ on search filter text field displays full list and enables next

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard
* User has a search filter active

**Input**

* User clicks on ‘x’ button on search filter text field

**Expected Result**

* All grids return to their original state before search filter was applied.
* Records are selectable
* Text “(must be cleared to proceed to next) should disappear
* Next button is enabled

**Actual Result**

* All grids return to their original state before search filter was applied.
* Records are selectable
* Text “(must be cleared to proceed to next) should disappear
* Next button is enabled

**ID: MJ-SD-523-06**

**Purpose**

* Test if clicking no on warning message removes message box without navigating user

**Precondition**

* User has logged on mobile judge
* User is on the second tab of the email Wizard
* User has clicked previous button and warning message is displayed

**Input**

* User clicks on No button

**Expected Result**

* User is not navigated to previous tab and message box disappears

**Actual Result**

* User is not navigated to previous tab and message box disappears

**ID: VIP-SD-854-01**

**Purpose**

* Ensure that the competition page contains the appropriate data

**Precondition**

* User has reached the VIP webpage

**Input**

* User clicks Competition Information tab

**Expected Result**

* Competition Information is returned to the browser

**Actual Result**

Competition Information is returned to the browser

**ID: VIP-SD-840-01**

**Purpose**

* Test if any user can navigate the entire site using the navigation bar

**Precondition**

* User is already on VIP website

**Expected Result**

* User can navigate the entire site using the navigation bar

**Actual Result**

* User can navigate the entire site using the navigation bar

**ID: VIP-RD-843-01**

**Purpose**

* Test if navigation shows on every page

**Precondition**

* User is already on VIP site

**Expected Result**

* Navigation bar is displayed on every page

**Actual Result**

* Navigation bar does not display on every page

**ID: VIP-SD-854-01**

**Purpose**

* Ensure that the Peer Evaluation page contains the appropriate data

**Precondition**

* User has reached the VIP webpage

**Input**

* User clicks Peer Evaluation tab

**Expected Result**

* User is navigated to Peer evaluation page, which contains accurate and up to date information.

**Actual Result**

* User is navigated to Peer evaluation page, which contains accurate and up to date information.

**ID: VIP-SD-854-01**

**Purpose**

* Ensure that the competition page contains the appropriate data

**Precondition**

* User has reached the VIP webpage

**Input**

* User clicks Competition Information tab

**Expected Result**

* Competition Information is returned to the browser

**Actual Result**

* Competition Information is returned to the browser

**ID: VIP-SD-841-01**

**Purpose**

* To test if the user can see the correct “About VIP” information.

**Precondition**

* User has reached VIP web page

**Input**

* User clicks “ABOUT VIP” on header

**Expected Result**

* A page with all the information about the VIP program will be shown.

**Actual Result**

* A page with all the information about the VIP program will be shown.

**ID: VIP-RD-841-01**

**Purpose**

* To test if the user can see the correct “About VIP” information.

**Precondition**

* User has reached VIP web page

**Input**

* User clicks “ABOUT VIP” on header

**Expected Result**

* A page with all the information about the VIP program will be shown.

**Actual Result**

* “About VIP” page is returned with no info. The web server will not deliver the information.

**ID: VIP-SD-853-01**

**Purpose**

* To test if the user can see the correct “syllabus file” information.

**Precondition**

* User has reached VIP web page

**Input**

* User clicks “SYLLABUS – SPRING 2016” button

**Expected Result**

* A word document will be downloaded.

**Actual Result**

* A word document will be downloaded.

**ID: VIP-RD-853-01**

**Purpose**

* To test if the user can see the correct “syllabus file” information.

**Precondition**

* User has reached VIP web page

**Input**

* User clicks “SYLLABUS – SPRING 2016” button

**Expected Result**

* A word document will be downloaded.

**Actual Result**

* The file will be downloaded because the there was no connection with the database.

**ID: VIP-SD-843-01**

**Purpose**

* Test if any user can access the website

**Precondition**

* n/a

**Expected Result**

* User can see VIP main page

**Actual Result**

* User can see VIP main page

**ID: VIP-RD-843-01**

**Purpose**

* Test if any user can access the website

**Precondition**

* n/a

**Expected Result**

* User can see VIP main page

**Actual Result**

* User can’t see main page

**ID: VIP-SD-542-01**

**Purpose**

* To ensure the correct contact information is displayed when user clicks Contact Us

**Precondition**

* User has reached VIP webpage

**Input**

* User clicks Contact on header

**Expected Result**

* A page with all the directors and collaborators to be displayed

**Actual Result**

* A page with all the directors and collaborators to be displayed

**ID: VIP-RD-542-01**

**Purpose**

* Contact page is clicked with no users in database

**Precondition**

* User has reached VIP webpage
* Database has been initialized with no data

**Input**

* User clicks Contact on header

**Expected Result**

* Contact page is returned with no users. Sections are still present

**Actual Result**

* Contact page is returned with no users. Sections are still present

**ID: VIP-SD-672-01**

**Purpose**

* To test if the notification can be sent properly when the user creates a new project.

**Precondition**

* Faculty and staff have navigated to <http://vip-dev.cis.fiu.edu/#/project-proposal>
* Faculty and staff have filled out correctly the “Team Proposal Form”

**Input**

* User clicks on “Submit” button.

**Expected Result**

* An email notification is sent to the PI.

**Actual Result**

* An email notification is sent to the PI.

**ID: VIP-RD-672-01**

**Purpose**

* To test if the notification can be sent properly when the user creates a new project.

**Precondition**

* Faculty and staff have navigated to <http://vip-dev.cis.fiu.edu/#/project-proposal>
* Faculty and staff have filled out correctly the “Team Proposal Form”

**Input**

* User clicks on “Submit” button.

**Expected Result**

* An email notification is sent to the PI.

**Actual Result**

* The email notification will not be sent because the mail server did not recognize the email of the PI.

**ID: VIP-SD-914-01**

**Purpose**

* Test if user is able to see if all unread todos get populated

**Precondition**

* User is signed into VIP
* User is of faculty/PI/CoPI
* User is add todo page

**Expected Result**

* User sees all unread todos

**Actual Result**

* User sees all unread todos

**ID: VIP-RD-914-01**

**Purpose**

* Test if any user, including guest and students, can read todos

**Precondition**

* Guest or student is on the site

**Expected Result**

* Guest and student can not see todos

**Actual Result**

* Guest and student can see todos

**ID: VIP-SD-717-01**

**Purpose**

* Ensure Registration form creates a user.

**Precondition**

* User fills out entire form .

**Input**

* User Clicks on Register User .

**Expected Result**

* User Created message.

**Actual Result**

* User Created message.

**ID: VIP-SD-717-02**

**Purpose**

* Ensure Registration form creates a user when PI does not enter fiu email .

**Precondition**

* User fills out entire form .

**Input**

* User Clicks on Register User and excludes an fiu email.

**Expected Result**

* User Please Verify Email Message.

**Actual Result**

* User Please Verify Email Message.

**ID: VIP-SD-717-03**

**Purpose**

* Ensure Registration form creates a user when PI does not have panther ID.

**Precondition**

* User fills out entire form and excludes a panther ID.

**Input**

* User Clicks on Register User.
* Entire form is completed except panther ID.

**Expected Result**

* User Created message.

**Actual Result**

* User Created message .

**ID: VIP-RD-717-01**

**Purpose**

* Ensure Registration form creates a user when user inputs invalid password.

**Precondition**

* User fills out entire form and excludes a capital letter from the password.

**Input**

* User Clicks on Register User.
* Entire form is completed password or confirm password.

**Expected Result**

* Error message saying password must contain capital letter .

**Actual Result**

* Error message saying password must contain capital letter .

**ID: VIP-SD-672-01**

**Purpose**

* To test if the notification can be sent properly when the user creates a new project.

**Precondition**

* Faculty and staff have navigated to <http://vip-dev.cis.fiu.edu/#/project-proposal>
* Faculty and staff have filled out correctly the “Team Proposal Form”

**Input**

* User clicks on “Submit” button.

**Expected Result**

* An email notification is sent to the PI.

**Actual Result**

* An email notification is sent to the PI.

**ID: VIP-RD-672-01**

**Purpose**

* To test if the notification can be sent properly when the user creates a new project.

**Precondition**

* Faculty and staff have navigated to <http://vip-dev.cis.fiu.edu/#/project-proposal>
* Faculty and staff have filled out correctly the “Team Proposal Form”

**Input**

* User clicks on “Submit” button.

**Expected Result**

* An email notification is sent to the PI.

**Actual Result**

* The email notification will not be sent because the mail server did not recognize the email of the PI.

**ID: VIP-SD-746-01**

**Purpose**

* Test if user can save information

**Precondition**

* User is signed into VIP
* Is at the profile page

**Expected Result**

* User can update changed information

**Actual Result**

* User updated information

**ID: VIP-RD-746-01**

**Purpose**

* Test if user can leave required fields empty

**Precondition**

* User is signed into VIP
* Is at the profile page

**Expected Result**

* User is not able to save with empty fields

**Actual Result**

* User was able to save information with empty fields

**ID: VIP-SD-846-01**

**Purpose**

* Ensure Student can apply to VIP project

**Precondition**

* Student has reached the VIP site and reached project application form

**Input**

* Student inputs data into form
* Student clicks submit

**Expected Result**

* Student project application is created

**Actual Result**

* Student project application is created

**ID: VIP-SD-846-02**

**Purpose**

* Ensure when student chooses to apply to a specific project the project is automatically selected in the application form

**Precondition**

* User has reached the VIP
* User has reached a project page

**Input**

* Student clicks apply button

**Expected Result**

* Project is automatically selected in the application form on the Project drop down tab

**Actual Result**

* Project is automatically selected in the application form on the Project drop down tab

**ID: VIP-RD-846-03**

**Purpose**

* Ensure project application must contain valid data for all required fields

**Precondition**

* Student has reached the VIP site and reached project application form

**Input**

* Student leaves a required field blank or inputs invalid data

**Expected Result**

* Submit button becomes disabled

**Actual Result**

* Submit button becomes disabled

**ID: VIP-SD-693-01**

**Purpose**

* Ensure User can create a new project

**Precondition**

* User has reached the VIP and reached project form

**Input**

* User inputs data into form
* User clicks submit

**Expected Result**

* Project proposal is created

**Actual Result**

* Project proposal is created

**ID: VIP-SD-693-02**

**Purpose**

* Ensure User can edit a story

**Precondition**

* User has reached the VIP
* User has clicked edit project

**Input**

* User updates form
* User clicks submit

**Expected Result**

* Project proposal is created

**Actual Result**

* n/a (failing)

**ID: VIP-RD-693-03**

**Purpose**

* Ensure Project must contain data for all required fields

**Precondition**

* User has reached the VIP and reached project form

**Input**

* User leaves title field blank

**Expected Result**

* Form notifies user that title field is required

**Actual Result**

* Form notifies user that title field is required

**ID: VIP-SD-854-01**

**Purpose**

* Ensure Project cannot have the same name as another Project

**Precondition**

* User has reached the VIP and reached project form

**Input**

* User inputs data into form
* Title matches a name of another project

**Expected Result**

* Project proposal fails

**Actual Result**

* n/a(failing)

**ID: VIP-SD-848**

**Purpose**

* To ensure the correct contact information is displayed when user clicks Teams

**Precondition**

* User has VIP webpage loaded.

**Input**

* User clicks TEAMS on nav bar.

**Expected Result**

* A page with all the Teams should be displayed.

**Actual Result**

* A page with all the Teams should be displayed.

**ID: VIP-SD-848**

**Purpose**

* Teams page is clicked with no Teams in the Database

**Precondition**

* User has VIP webpage loaded.
* Database has no data stored for Teams

**Input**

* User clicks TEAMS on nav bar.

**Expected Result**

* A page with no Teams should be displayed.

**Actual Result**

* A page with no Teams should be displayed.

# 

# 

# Glossary

**PI**: (Principal Investigator)someone making a significant contribution to a project.

**Co-PI**: (CO-Principal Investigator) is an individual that the PI relies on to assume responsibilities above those of other personnel.

**Judge**: technical and non-technical people who will grade your skill level.

**Guest**: Basic access to the web application. Default type of access when entering for the first time

**Students**: Advance access the web application with the ability to have a profile

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

Minimal Class Diagram.png

Figure A1 - VIP Minimal Class Diagram



Figure A2 - Mobile Judge Minimal Class Diagram

…

### Dynamic UML Diagrams

UseCaseDiagramVIP (1).png

Figure A3 - VIP use case diagram

## projectProposalSequence (1).png

Figure A4 - Sequence Diagram for VIP

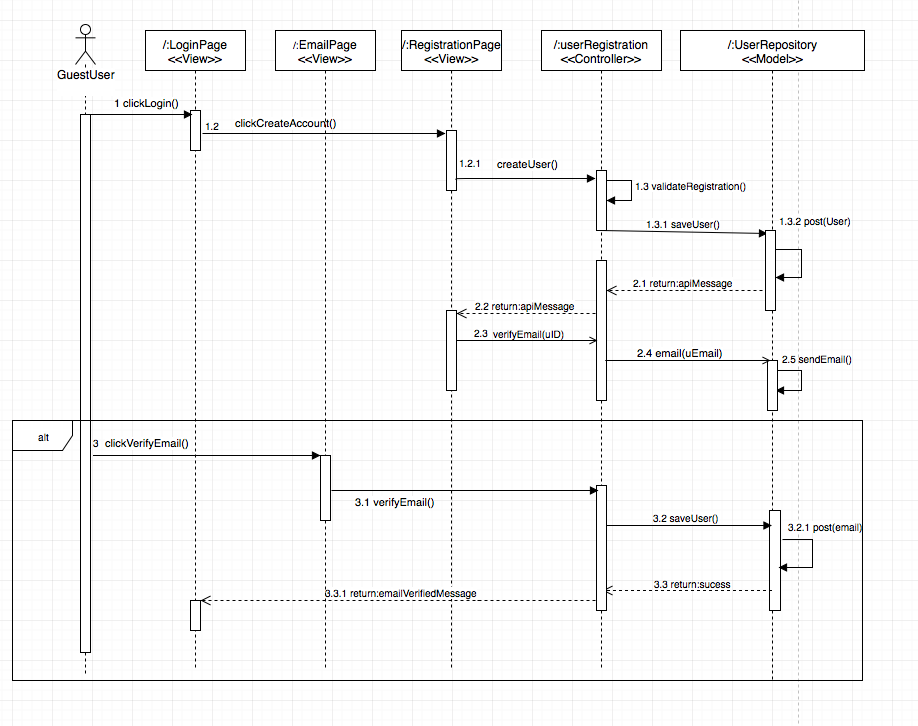
Figure A5 - Sequence Diagram for VIP



Figure A6 - Sequence Diagram for VIP

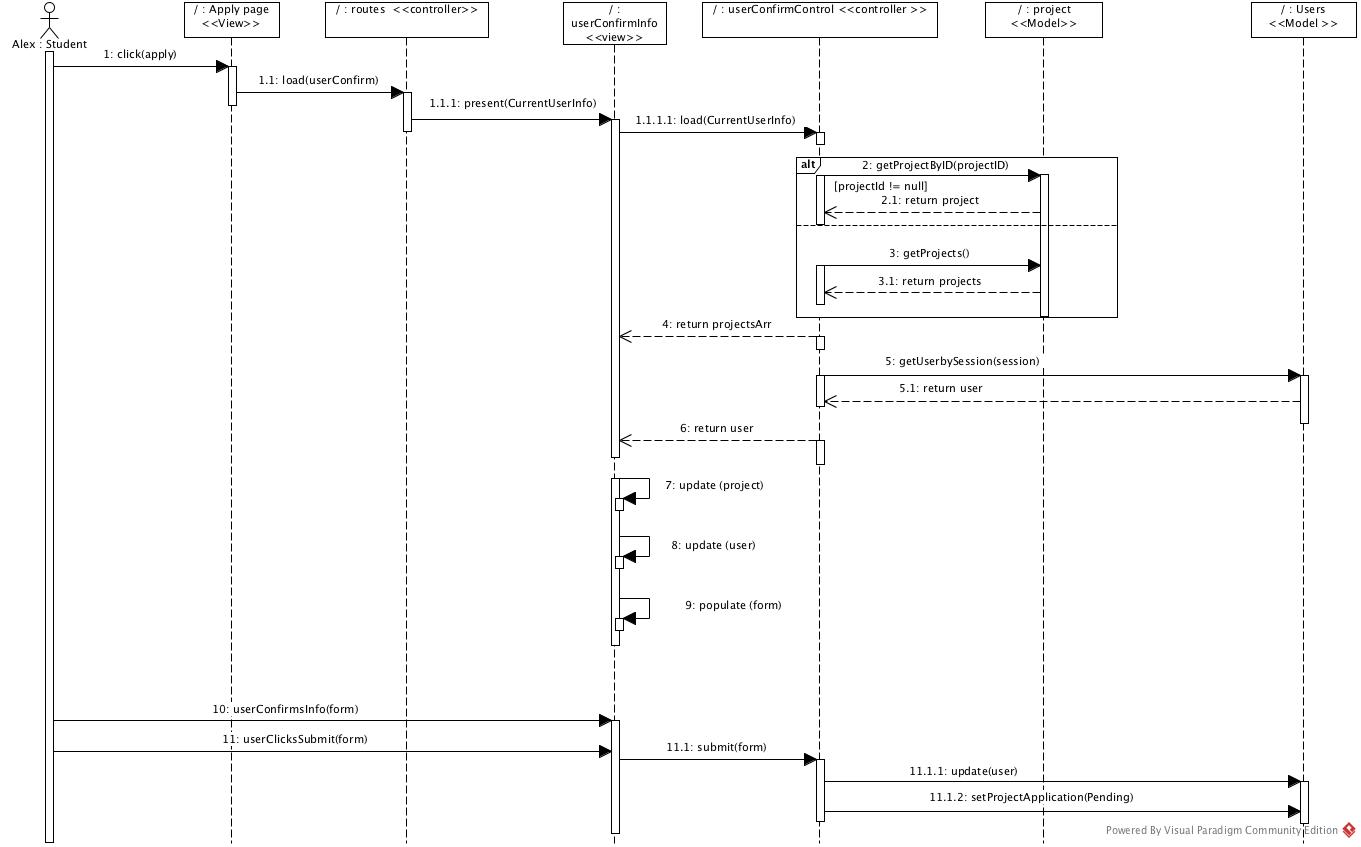


Figure A7 - Sequence Diagram for VIP

Sequence Diagram User Story 848.png

Figure A8 - Sequence Diagram for VIP

## competitionSequence (1).png

Figure A9 - Sequence Diagram for VIP

## competitionSequence (1).png

Figure A10 - Sequence Diagram for VIP



Figure A11 - Sequence Diagram for VIP

## Contactsequence.png

Figure A12 - Sequence Diagram for VIP #842

C:\Users\Vituchin\Documents\FIU\spr2016\senior\VIP\nodemailer\Untitled Diagram.png

Figure A13 - Sequence Diagram for VIP #672

## C:\Users\Vituchin\Documents\FIU\spr2016\senior\VIP\forget_password\forget_password\reset_password_last.png

Figure A14 - Sequence Diagram for VIP #927

## C:\Users\Vituchin\Documents\FIU\spr2016\senior\VIP\nodemailer\seq_diag_notification.png

Figure A15 - Sequence Diagram for VIP #853

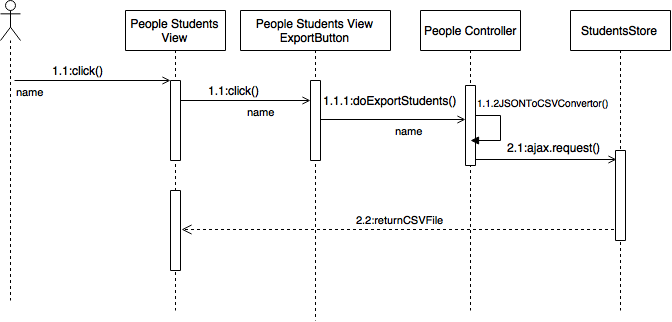


Figure A16 - Sequence Diagram for MobileJudge #548

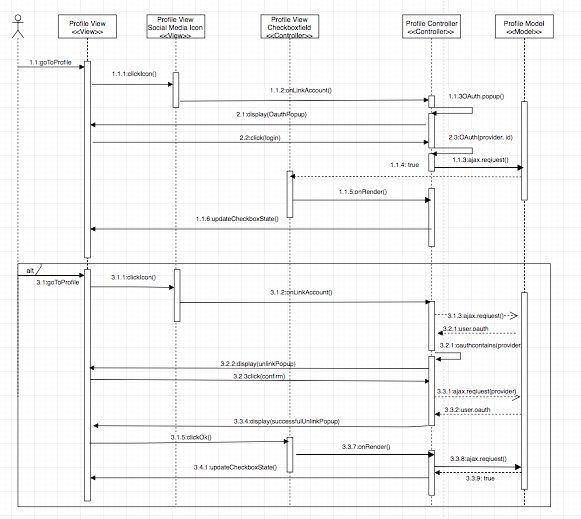


Figure A17 - Sequence Diagram for MobileJudge #519

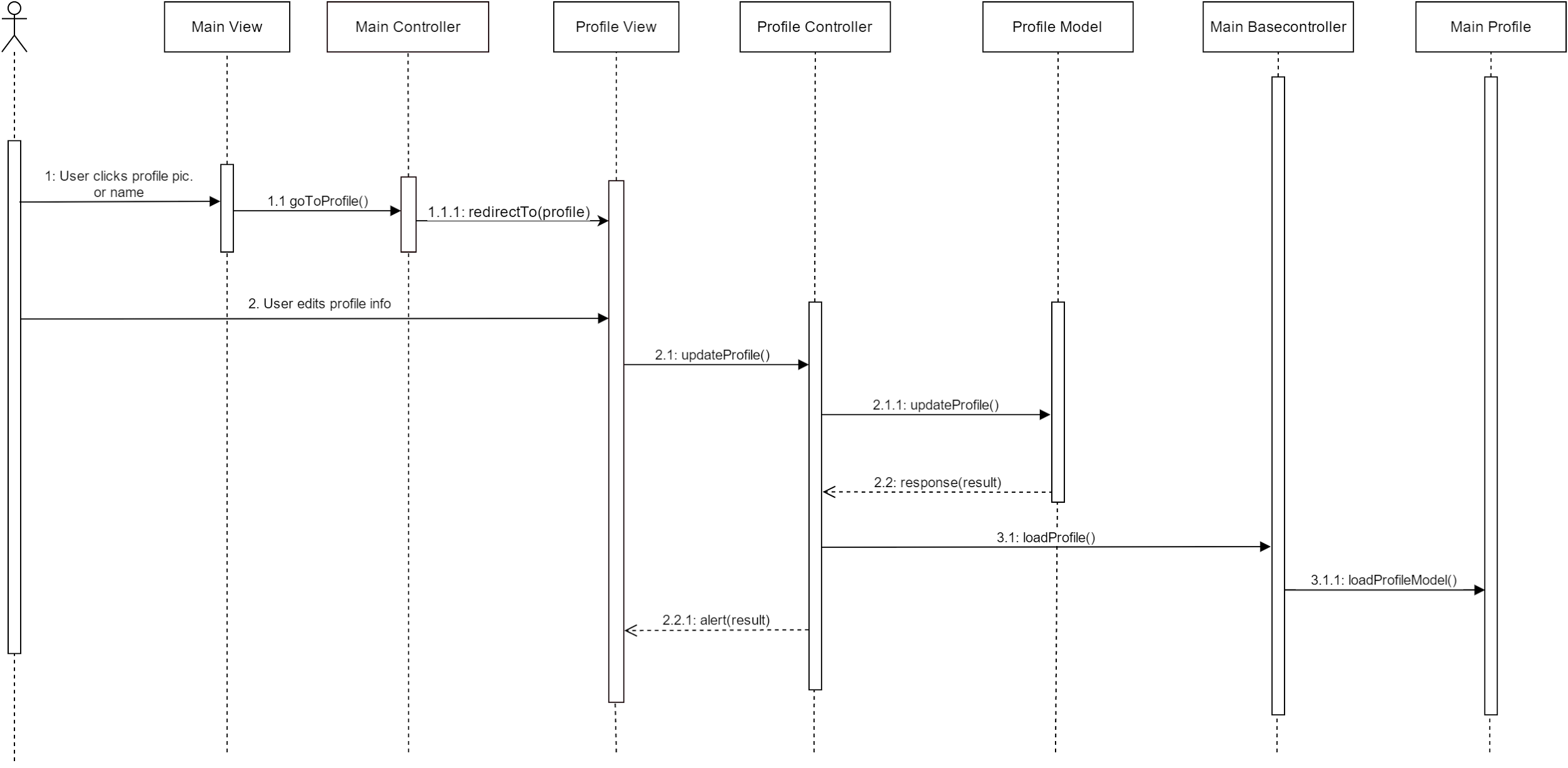


Figure A18 - Sequence Diagram for MobileJudge #520

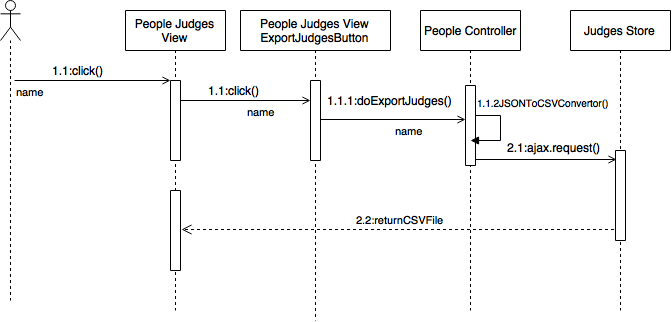


Figure A19 - Sequence Diagram for MobileJudge #522

## Figure A20 - Sequence Diagram for MobileJudge #523

524 - Sequence Diagram.png

Figure A21 - Sequence Diagram for MobileJudge #524

Updated User Story #525 Sequence Diagram first.png

Figure A22 - Sequence Diagram for MobileJudge #525

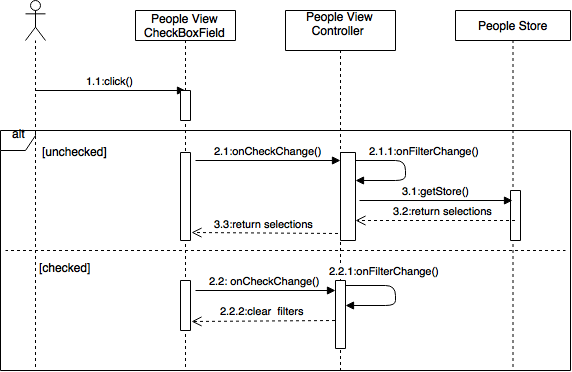


Figure A23 - Sequence Diagram for VIP #623

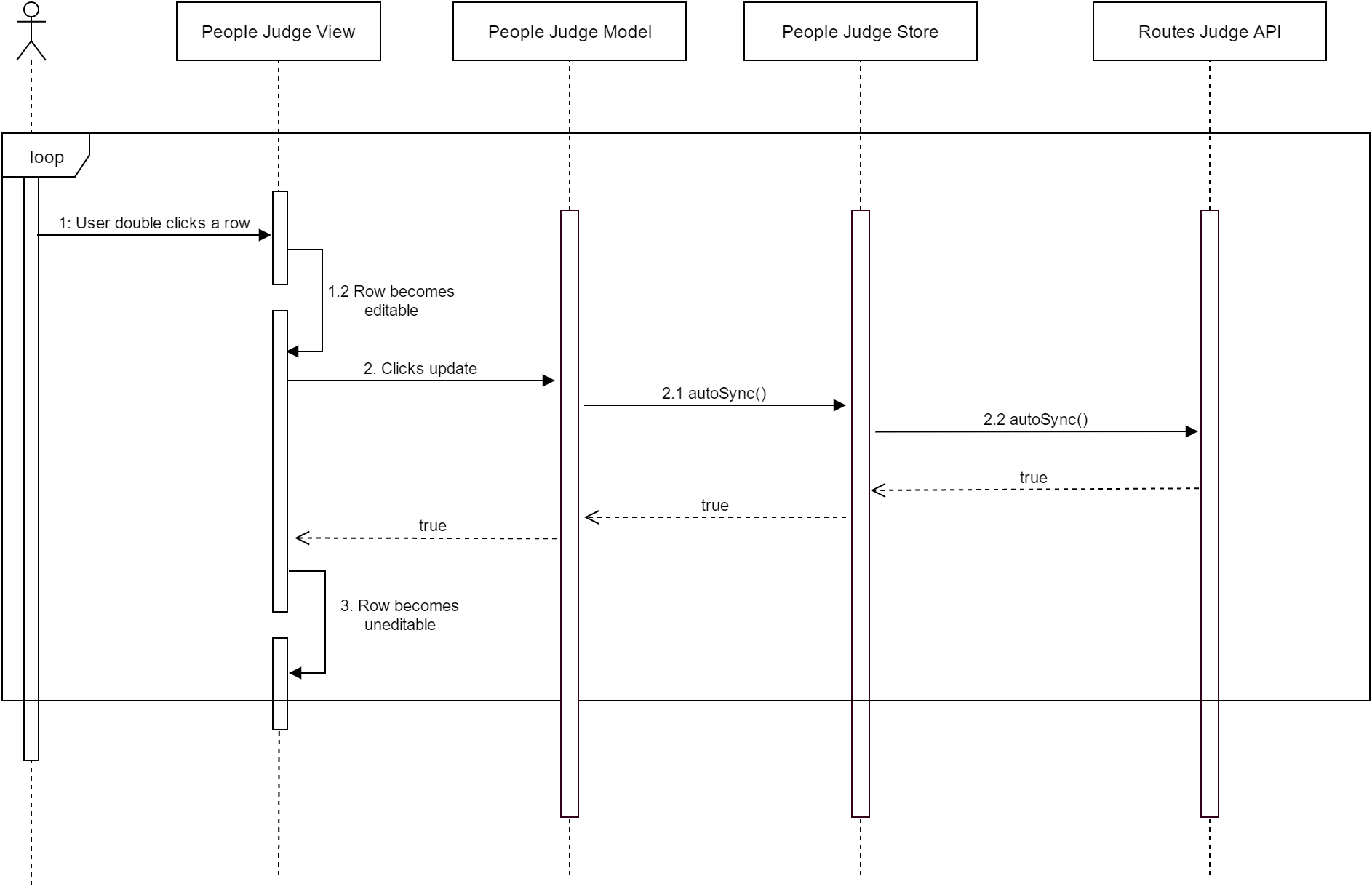


Figure A23 - Sequence Diagram for VIP #543

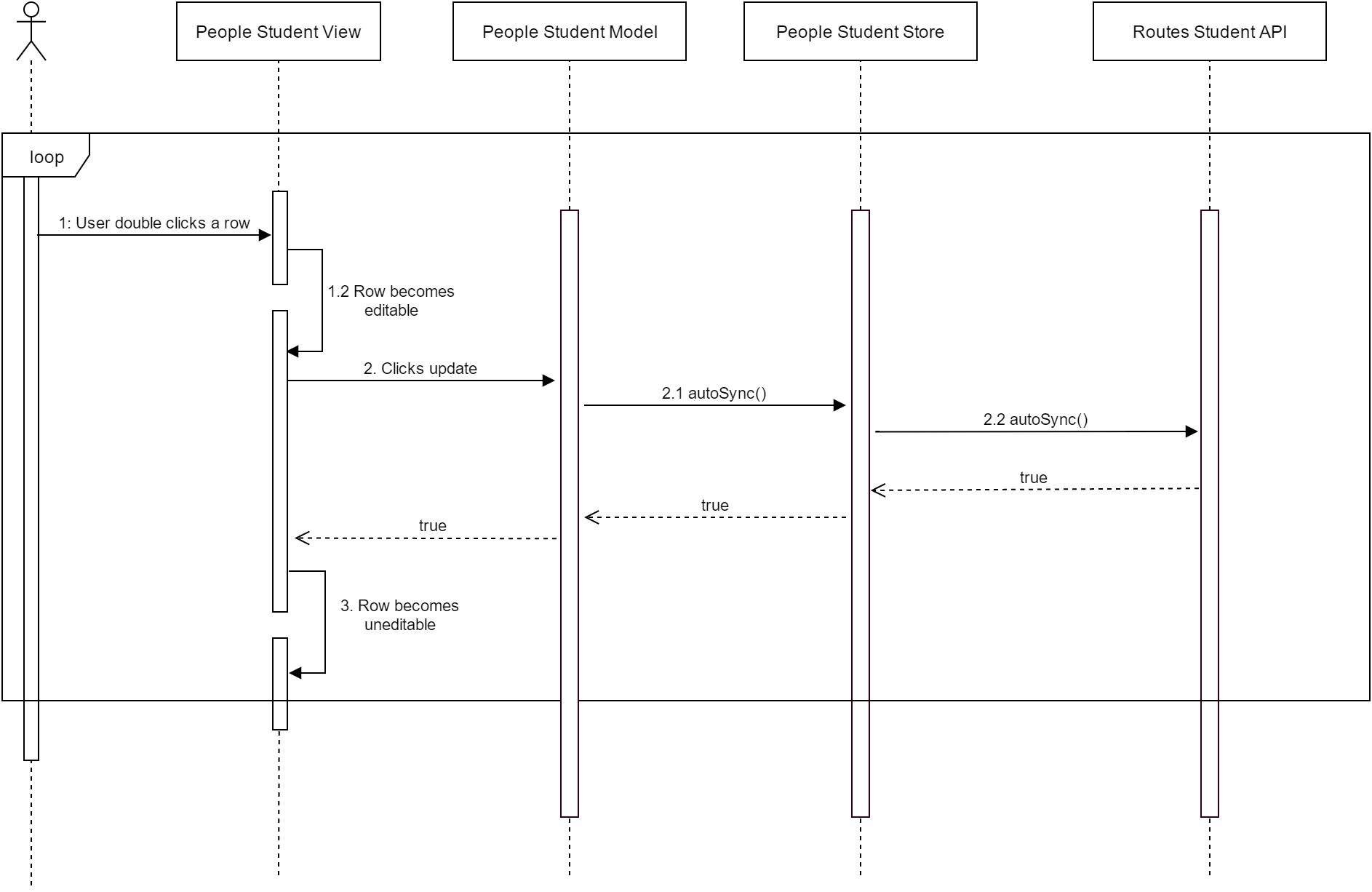


Figure A24 - Sequence Diagram for VIP #544

547-DockerSequenceDiagram.png

Figure A25 - Sequence Diagram for VIP #547

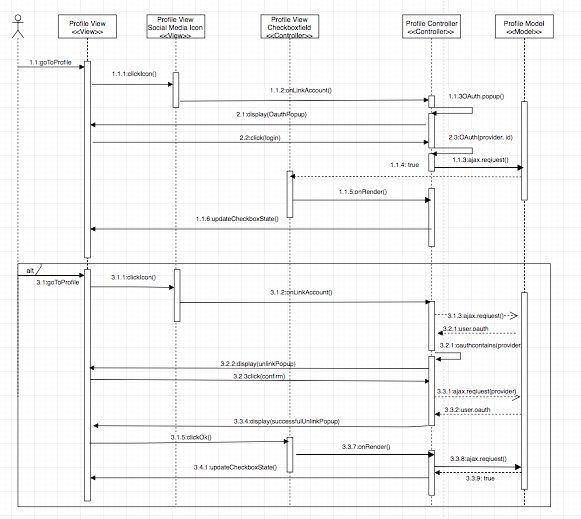


Figure A26 - Sequence Diagram for VIP #622

## 

Figure A27 - Sequence Diagram for VIP #623

## Appendix B - User Interface Design

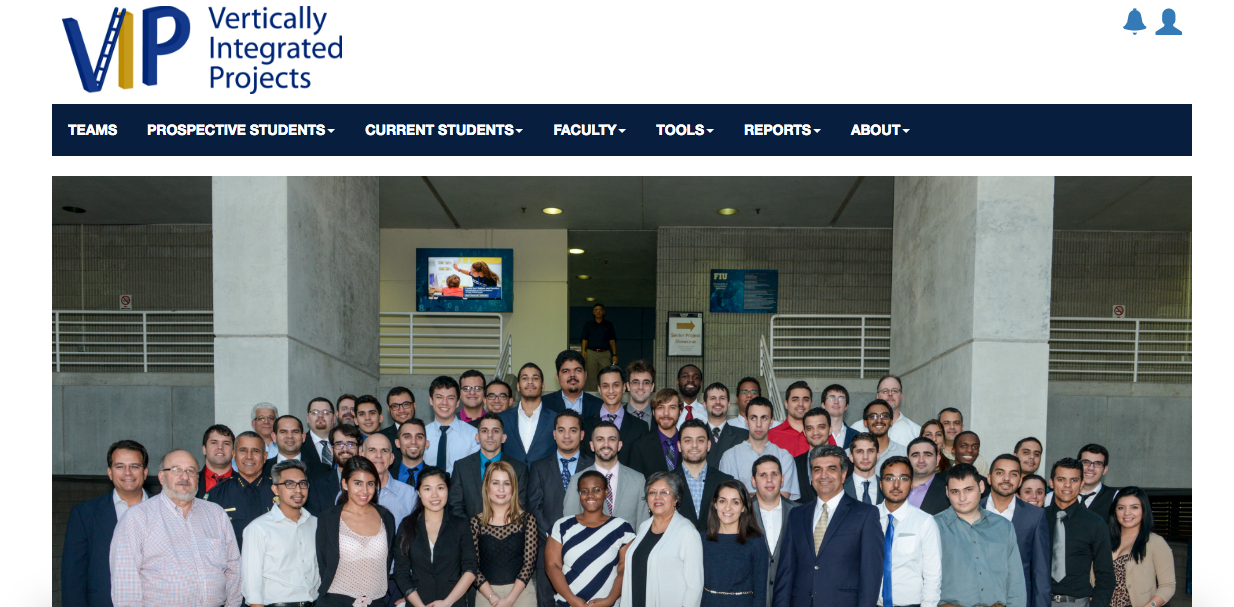


Figure B1 - Homepage for VIP

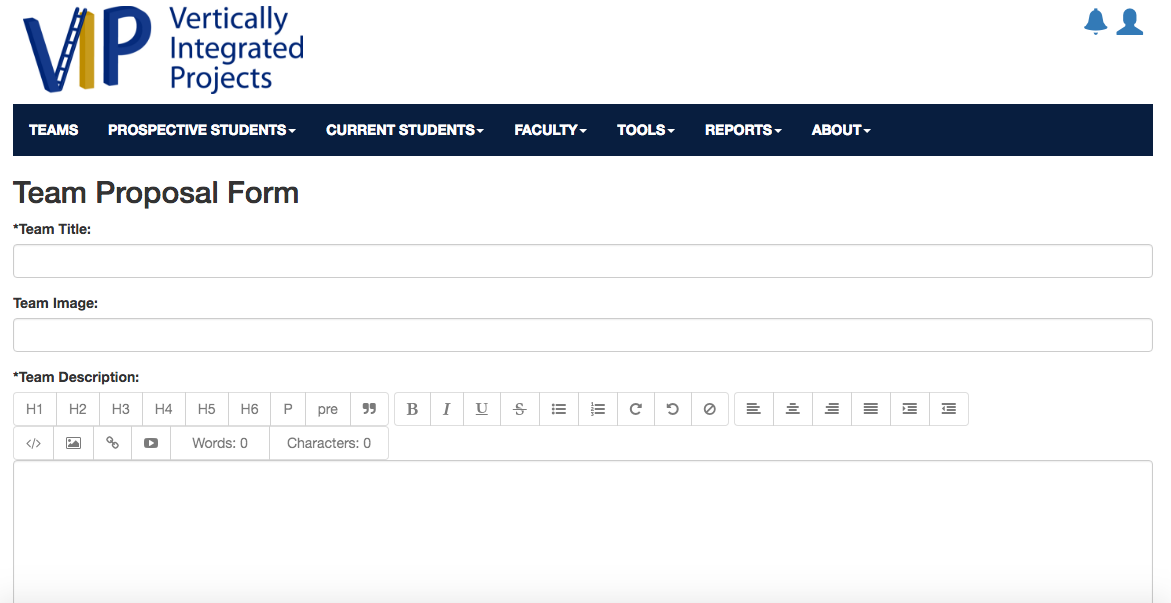


Figure B2 - Project Proposals form

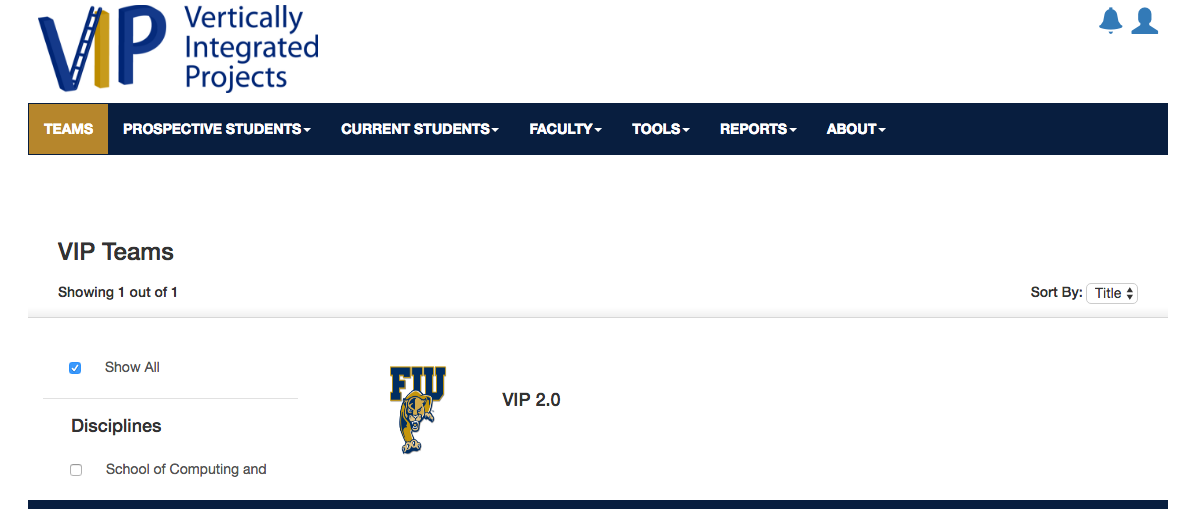


Figure B3 - Teams Page

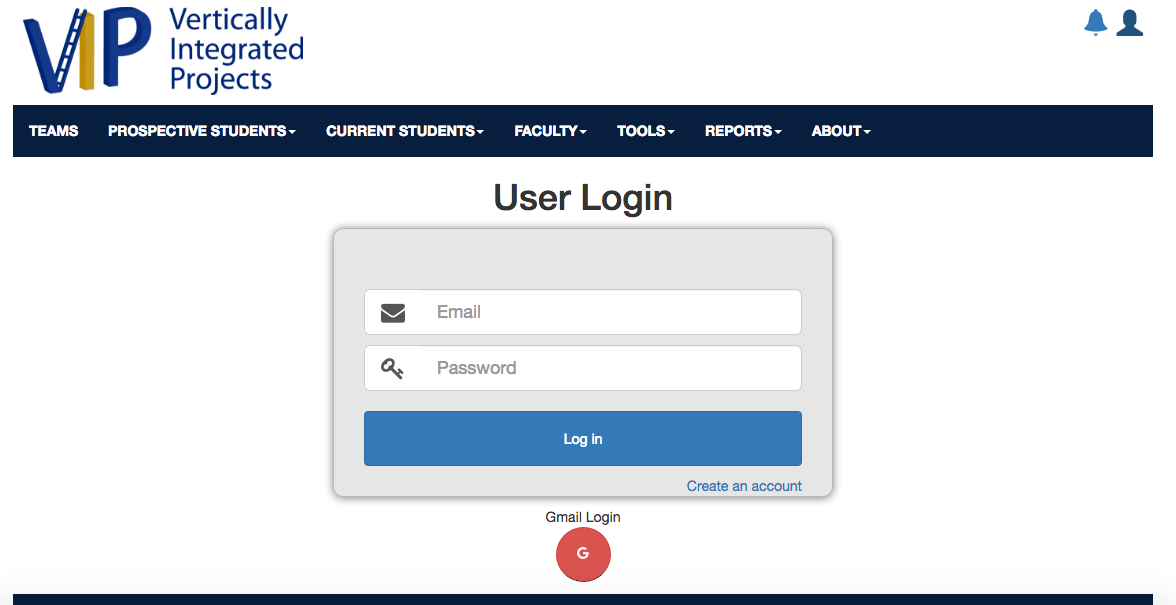


Figure B4 - Login Page

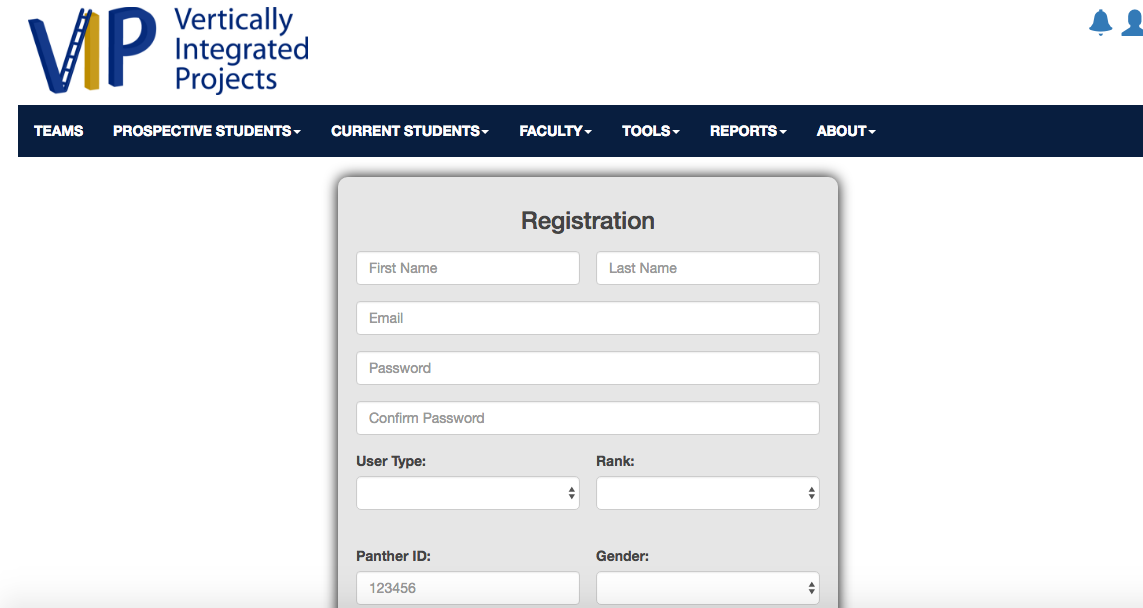


Figure B5 - Faculty Registration Form

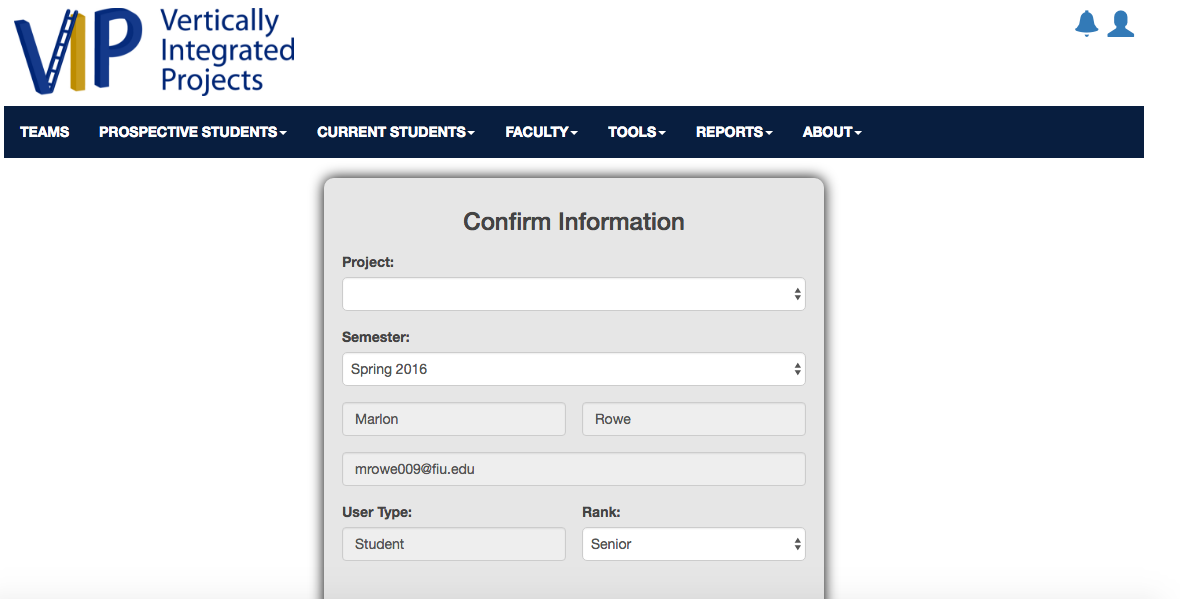


Figure B6 - Student Team Application Form

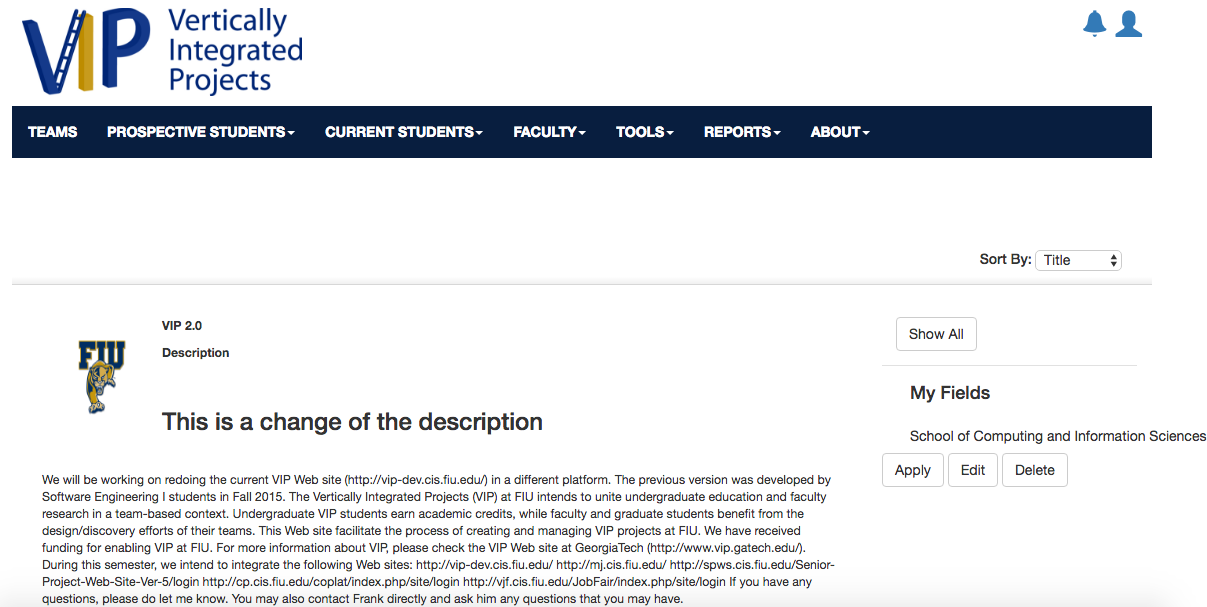


Figure B7 - Team Details Page

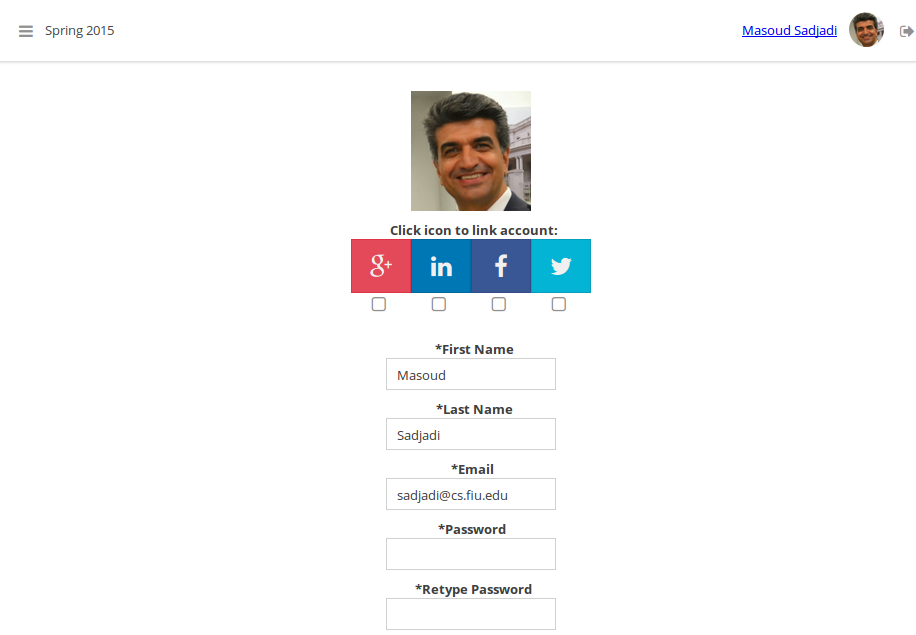
## 

## 

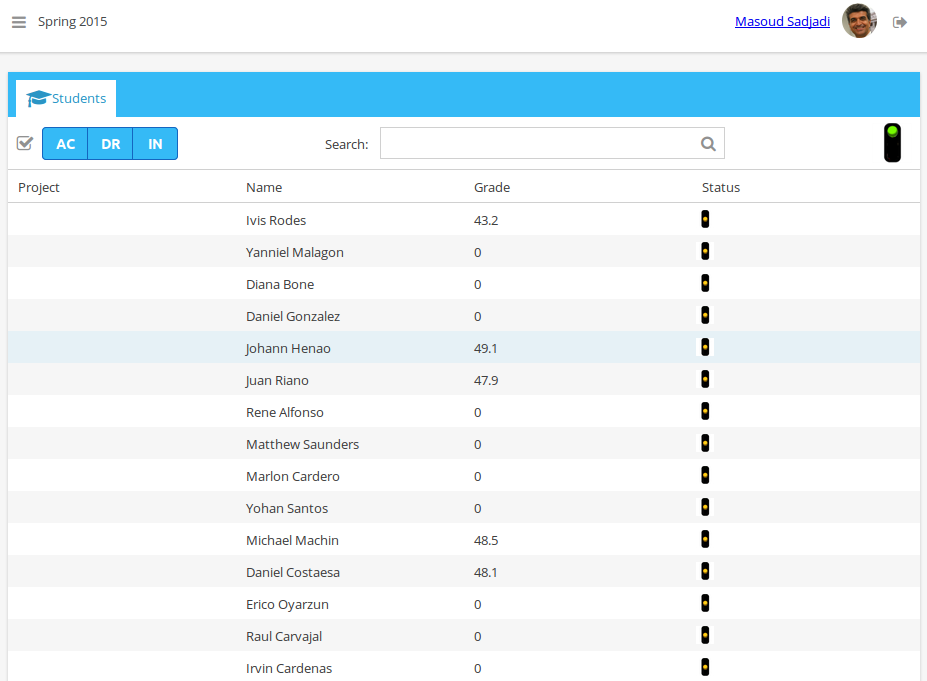
## 

## Screen Shot 2016-05-05 at 4.34.22 PM.png

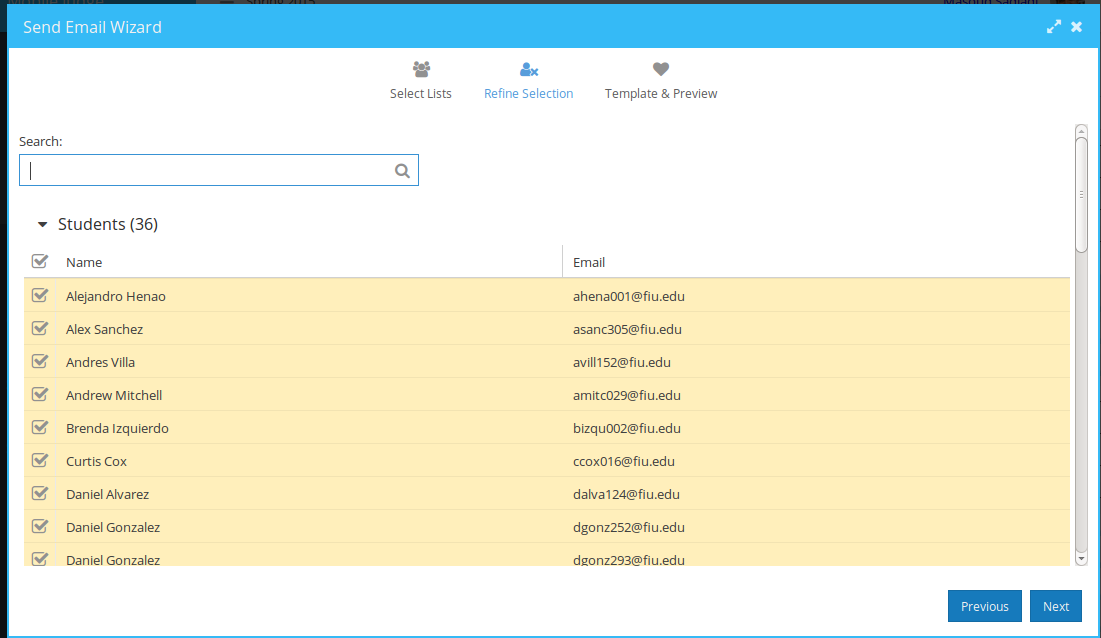
## Figure B8 - Mobile Judge Home screen



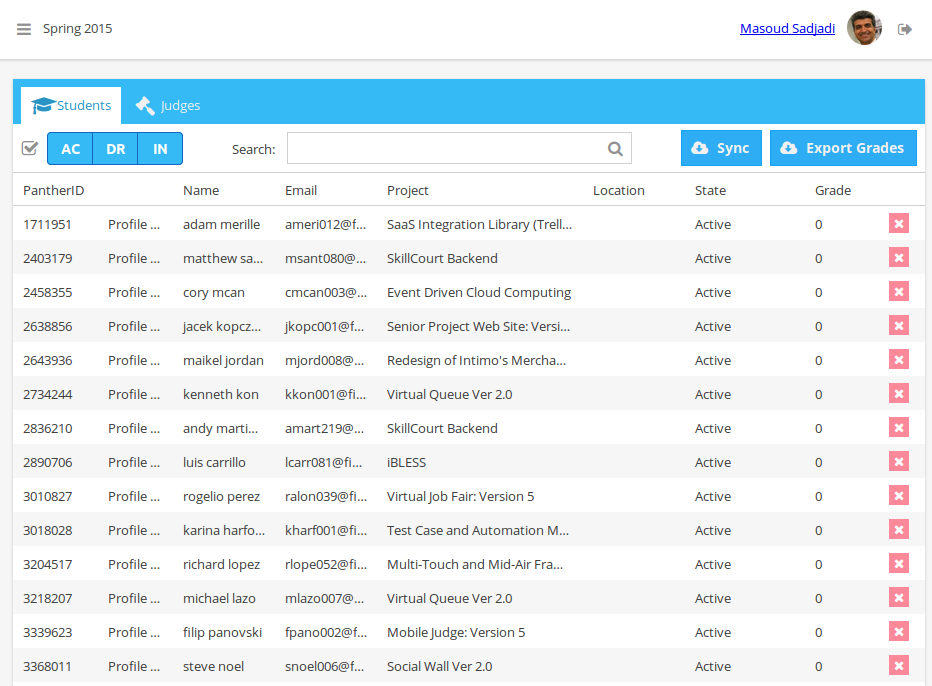
## Figure B9 - Mobile Judge Profile Page



## Figure B10 - Mobile Judge Accept/Reject grades



## Figure B11 - Mobile Judge email search filter



## Figure B12 - Mobile Judge Export grades

## 

## 

## Appendix C - Sprint Review Reports

**Sprint 1 Review Meeting Notes**

**Date: 2/1/2016**

**Attendees:Tiago,Rodolfo,Victoriano,Jorge,Andres,Miguel,Steven**

**Start time:2100**

**End time:2130**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #542 Add Check Box on Filters for People Selection - Upon small changes

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story: #542 Add Check Box on Filters for People Selection
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* User Story - #523 - Add Search Feature to Email Wizard
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Fix bugs in the implementation
* User Story: - User Story - #524 - User List Show Sizes
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* User Story:-User Story - # 520 - Manage Profile
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* - User Story - # - Export Grades
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Define tasks
    - Redraw Diagrams
* User Story: #547 Implement Docker CI
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add user tutorial

**Sprint 2 Review Meeting Notes**

**Date: 2/15/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time: 2100**

**End time: 2230**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #547 Implement Docker CI
* User Story: #523 Add Search Feature to Email Wizard
* User Story: #524 Manage List Size
* User Story: #520 Manage Profile
* User Story: #548 Export Grades
* User Story: #598 Feasibility Story

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story: #543 Manage Judges
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* User Story - #544 Manage Students
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional Tasks

**Sprint 3 Review Meeting Notes**

**Date: 2/28/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time:2100**

**End time: 2300**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #623 Visualize Account States
* User Story: #622 Unlink Accounts
* User Story: #525 Review Grades
* User Story: #543 Manage Judges
* User Story: #544 Manage Students
* User Story: #519 Link Login

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

**Sprint 4 Review Meeting Notes**

**Date: 3/21/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time: 2100**

**End time: 2300**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #844 View Page Footer
* User Story: #842 View Contact Page
* User Story: #843 View Main Page
* User Story: #847 View How VIP Credits Count
* User Story: #854 View VIP Competitions
* User Story: #856 Install Development Stack onto Development Server
* User Story: #855 Write Database Schema

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story: #840 View Website Header
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* User Story: #841 View About VIP Page
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks

**Sprint 5 Review Meeting Notes**

**Date: 4/4/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time: 2100**

**End time:2300**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #840 View Website Header
* User Story: #841 View About VIP Page
* User Story: #906 View VIP Organization Page
* User Story: #853 View VIP Syllabus Information
* User Story: #894 View Class Days, Time, and Location
* User Story: #899 View Presentations & Publications
* User Story: #851 View Peer Evaluations Page
* User Story: #893 View Permit Request Form
* User Story: #895 View Apply Undergraduate Page
* User Story: #896 View VIP Apply Graduate Page
* User Story: #902 View Classroom Schedule Page

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

**Sprint 6 Review Meeting Notes**

**Date: 4/16/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time:2100**

**End time:2300**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #913 Redesign Components of the Website
* User Story: #672 Receive Project Proposal

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story: #745 Login
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks & fix the bugs
* User Story: #717 Register User
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional tasks
* User Story #846 Apply for Existing Projects
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional Tasks
* User Story #746 Manage Profile
  + Reason for rejection:
    - Incomplete
  + How this should be reflected on the user story definition in Mingle:
    - Add additional Tasks

**Sprint 7 Review Meeting Notes**

**Date: 4/15/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time:2300**

**End time:2330**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story: #693 Propose a new Project
* User Story: #717 Register User
* User Story: #745 Login
* User Story: #846 Apply for Existing Projects
* User Story: #746 Manage Profile
* User Story: #914 View ToDo List
* User Story: #848 View My VIP Projects Page
* User Story: #927 Change Forgotten Password

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## 

## 

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective Meeting Notes**

**Date: 2/1/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time: 2100**

**End time: 2130**

What went wrong?

* Team didn’t use the same configuration for the diagrams. All the styles were different.

Did we do a good job estimating our team's velocity?

* No, because we underestimated the overhead in learning Sencha due to lack of Technical Mentor was not available in a consistent manner - few days with no contact existing product documentation.

Did we do a good job estimating the points (time required) for each user story?

* No for both documentation tasks and implementation due to Sencha learning curve.

Did each team member work as scheduled?

* Yes.

What went right?

* Scrum.
* Setting up development environment.
* Good teamwork.
* Daily communication and remote calls.

How to address the issues in the next sprint?

* Begin using the same tools and color scheme in order to write code and documentation (Google Docs, Draw.io, WebStorm IDE).

How to improve the process?

* Plan to meet with Frank to understand existing implementation.

How to improve the product?

Continue integrating features and completing User Stories.

**Sprint 2 Retrospective Meeting Notes**

**Date: 2/14/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time: 1000**

**End time: 1030**

What went wrong?

* Sencha is still giving problems. It’s very difficult to find useful documentation.

Did we do a good job estimating our team's velocity?

* No, because we underestimated the overhead in learning Sencha due to lack of Technical Mentor was not available in a consistent manner - few days with no contact existing product documentation. Still learning Sencha.

Did we do a good job estimating the points (time required) for each user story?

* No for both documentation tasks and implementation due to Sencha learning curve.

Did each team member work as scheduled?

* Yes.

What went right?

* Scrum.
* Good teamwork.
* Daily communication and remote calls.

How to address the issues in the next sprint?

* Continue trying to learn Sencha effectively in order to better estimate work.

How to improve the process?

* Read Sencha 6.0.2 docs online and try to find a Sencha forum.

How to improve the product?

* Continue integrating features and completing User Stories.

**Sprint 3 Retrospective Meeting Notes**

**Date: 2/28/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time:1830**

**End time:1930**

What went wrong?

* Problems with git practice.
* Sencha is still difficult to understand.
* Team still made a few assumptions instead of asking PO.
* Still having difficulties communicating with technical mentor.

Did we do a good job estimating the points (time required) for each user story?

* No. Still difficult to estimate work because team is still unfamiliar with Sencha. (Tiago and Andres did estimate well).

Did each team member work as scheduled?

* Yes.

What went right?

* Teamwork.
* Scrum Standups.
* Communication within team.
* Pair Programming.

How to address the issues in the next sprint?

* Jorge will provide git demonstration along with team reading git documentation.
* Team needs to communicate more with PO in order to better understand user story and not make any assumptions.

How to improve the process?

* Learn to better estimate what amount of work will be done. (Maybe need to add a research task to each user story in order to better estimate)

How to improve the product?

* Continue integrating features.
* Get ready for production release.

**Sprint 4 Retrospective Meeting Notes**

**Date: 3/21/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time:1830**

**End time:1930**

What went wrong?

* Still problems with git.
* Team still made a few assumptions instead of asking PO.

Did we do a good job estimating the points (time required) for each user story?

* Yes. Team did a much better job estimating their hours.

Did each team member work as scheduled?

* Yes.

What went right?

* Teamwork.
* Scrum Standups.
* Communication within team.
* Pair Programming.

How to address the issues in the next sprint?

* Team needs to practice git flow more.
* Team needs to communicate more with PO in order to better understand user story and not make any assumptions.

How to improve the process?

* Continue learning about MEAN Stack

How to improve the product?

* Continue integrating features.
* Begin working on Backend part of VIP

**Sprint 5 Retrospective Meeting Notes**

**Date: 4/3/2016**

**Attendees: Tiago, Victoriano, Steven, Jorge, Andres, Miguel, Rodolfo**

**Start time:1200**

**End time:1300**

What went wrong?

* Team had problems setting up development environment but went overall much better than Sencha.
* Getting the static pages on Github was difficult with so many people touching git at the same time.

Did we do a good job estimating the points (time required) for each user story?

* Yes. Team did a much better job estimating their hours.

Did each team member work as scheduled?

* Yes.

What went right?

* Teamwork.
* Scrum Standups.
* Communication within team.
* Communication with PO.
* Pair Programming.
* Working with NodeJS and setting up development environment.

How to address the issues in the next sprint?

* Team needs to practice git flow more.

How to improve the process?

* Continue learning about MEAN Stack
* Estimate hours early and then track time to see if estimation was accurate.

How to improve the product?

* Fix mistakes, implement the features correctly now that the crunch time is over.

**Sprint 6 Retrospective Meeting Notes**

**Date: 4/16/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time:2100**

**End time:2130**

What went wrong?

* Team is a little slow picking up MEAN but doing a good job overall.

Did we do a good job estimating our team's velocity?

* Yes.

Did we do a good job estimating the points (time required) for each user story?

* No. Still early for some of our team to estimate correctly with MEAN stack.

Did each team member work as scheduled?

* Yes.

What went right?

* Scrum.
* Good teamwork.
* Daily communication and remote calls.
* MEAN stack process is much easier to learn and implement.

How to address the issues in the next sprint?

* Read over Git tutorial that Jorge made.

How to improve the process?

* Pair Program a little more so that we can learn from those who know MEAN better.

How to improve the product?

Fix any mistakes made and bugs to get ready for show case.

**Sprint 7 Retrospective Meeting Notes**

**Date: 4/15/2016**

**Attendees: Tiago, Rodolfo, Andres, Victoriano, Jorge, Miguel, Steven**

**Start time:2100**

**End time:2130**

What went wrong?

* Time Management

Did we do a good job estimating our team's velocity?

* Yes. Worked mostly on Docs for the presentation.

Did we do a good job estimating the points (time required) for each user story?

* No new user stories. Only worked on fixing bugs.

Did each team member work as scheduled?

* Yes.

What went right?

* Scrum.
* Good teamwork.
* Daily communication and remote calls.
* Fixing Documents

How to address the issues in the next sprint?

* Git is still a problem.

How to improve the process?

* Watch Git tutorials, practice using git, practice pulling and pushing repo, switching branches, etc.

How to improve the product?

* Implement Wishlist and Shortcomings

# References

The work of the previous group, for Mobile Judge 7 was continued, and refined upon.