*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Vertically Integrated Projects 6.0

**Team Members:** Jose Ponce Diaz

Dafna Steinberg

**Product Owner(s)**: Masoud Sadjadi

Francisco Ortega

**Mentor(s)**: Leila Zahedi

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2017 Florida International University*

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***

*The Vertically Integrated Projects web application (VIP 6.0) improves upon its predecessors by polishing existing features of the site, as well as adding brand new functionalities. This document presents the information necessary to gain a good understanding of the changes made during the sixth iteration of the project (release 6.0).*

*After a brief overview of the current system, the document will present a detailed catalogue of user stories (ie, features) developed during this application cycle. Emphasis was put on fixing pre-existing bugs, polishing existing features, and enhancing application security.*

*The document also provides insight on the software development process by providing a brief summary on VIP’s system architecture, design, and deployment. Also included are sample test-cases which check the VIP web app against typical user-driven scenarios. Additional material, including screenshots, diagrams, and manuals are available in the appendix.*

**Table of Contents**

**Introduction** ………………………………………………………………………………………………………………………… **5**

Current System ……….…………………………………………………………………………………………………… 5

Purpose of New System …………………………………………………..…………………………………………….. 5

**User Stories** ………………………………………………………………………………………………………………………… **6**

Implemented User Stories ………………………………………….…………………………………………………. 6

Pending User Stories …………………………………………………………………...………………………………. 15

**Project Plan** ……………………………………………………………………………………………………………………….. **18**

Hardware and Software Resources……………………………………………………...……………………….… 18

Sprints Plan……………………………………………………………..………………………………………………… 19

*Sprint 1*  ………………………………………………………………………………………………………… 19

*Sprint 2*  ………………………………………………………………………………………………………… 20

*Sprint 3*  ………………………………………………………………………………………………………… 20

*Sprint 4*  ………………………………………………………………………………………………………… 21

**System Design** ……………………………………………………………………………………………………………………... **22**

Architectural Patterns …………………………………………………………………………………………….... 22

System and Subsystem Decomposition ……………………………………………………………………………… 24

Deployment Diagram……………………………………………………………………………………………………. 26

Design Patterns …………………………………………………………………………………………………………. 26

**System Validation**…………………………………………………………………………………………………………………. **27**

**Glossary**………………………………………………………………………………………………………………………………. **33**

**Appendix**  ……………………………………………………………………………………………………………………………  **34**

Appendix A - UML Diagrams ………………………………………………………………………………………….. 34

Appendix B - User Interface Design …………………………………………………………………………….…… 46

Appendix C - Sprint Review Report s…………………………………………………………………………...…… 51

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist …………… 53

**References** ………………………………………………………………………………………………………………………….. **70**

# Introduction

The VIP Web System facilitates the operation of the Vertically-Integrated Projects(VIP) Program at FIU. This system is designed to support most of the day to day operations of students and faculty involved with the VIP Program. Some of the main operations/features supported are project proposals by faculty and their approval or denial by a (PI/administrator), and student applications to participate in the previously mentioned projects which will also need to be approved or denied by either the staff in charge or the system administrator.

## Current System

The current system(as of the beginning of this semester/iteration of the project) consisted of a website with the following major features broken down by actor/user. Every user in the system is allowed to message other users by using the websites built-in messaging functionality. Students are allowed to login using their FIU credentials(google+ account) without signup/registrations necessary. They are also allowed to apply to join projects in the system and remove themselves from projects they have already joined. Faculty user’s main two functions as of this time are to propose projects and accept/reject users who attempt to join projects that they own. This is, after they have registered in the system and have been approved by the PI/administrator.

The PI/administrator is the user with the most features at his/her disposal. He/she manages both the projects and the users in the system by performing the following operations. The administrator may approve/reject the following operations by other users: create non-student account, faculty project proposal submission, student applications to join projects. Also, they can disable/enable projects that had been previously approved in the system. Finally, they can get a list of users and their information as well as changing some of it.

## Purpose of New System

The “new” system is really an extension of the old system. It mainly consists of the old system plus several bug fixes, user interface changes and additional security. The exact details of what was changed can be found under the User Stories -> Implemented user stories section, however, we will now list some of the main changes/improvements. The two main problems with the existing system were:

(1) It had little to no security in place to prevent malicious users from “attacking” the system. For example, the api was not secured. This means, it would have been extremely easy for an attacker to get a hold of most of the information in the system without even having to obtain credentials in the system, as well as destroying some of the information in the system. It’s also important to note, in the old system, passwords were being sent unencrypted(http and not https) in the login page which would make possible for an attacker to get a hold of a user’s credentials.

(2) The system had a somewhat large number of bugs that needed to be fixed before it could be released(note we are referring to Release 5.0 of the system, not Release 4.0 which was the one in production as of the beginning of this iteration).

It is obvious then that the purpose of the new system was to address the previously explained deficiencies/problems that were present in the old system.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the VIP website project. 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

**[vip/#671] Notification of Existing Projects**

* **Description:** As a faculty member or student, I would like to be notified about existing VIP projects.
* **Acceptance Criteria:**

1. Notification will take over emails.
2. Users should be notified monthly on existing projects.
3. User should have an option to unsubscribe from notifications.

**[vip/#1146] Make login permanently on the user's system after first login**

* **Description:** As a user, I would like to be permanently logged after first login
* **Acceptance Criteria:**

1. After user return to the VIP website is already logged in

**[vip/#1260] Allow Admin to login as different user types**

* **Description:** As an Admin/PI, I would like to be able to be able to impersonate other users in the VIP Website. By impersonating various users, the Admin will be better able to understand and test the features of the website.
* **Acceptance Criteria:**

1. Admin/PI can login to the site normally(as himself)
2. When Admin/PI goes to the admin panel, he will see an impersonate button next to each user in the users table.
3. When Admin/PI clicks the impersonate button next to some user, he gets logged in to the website as if he was that user, no credentials required.
4. The Admin/PI can later logout from the impersonated account and login again normally as himself.

**[vip/#1262] Notify users about new projects added since last notification**

* **Description:** As a user(any user) of the vip website, i would like to be notified daily of any new projects available in the website. I would also like to be able to opt-out of receiving this emails.
* **Acceptance Criteria:**

1. As a user, i should receive a daily email with a list of(and links to) all new projects created in the vip website since the previous day.
2. I will not receive an email if there are no new projects.
3. When i click on any of the links from the email it should take me to the project details page for that specific project.
4. I will see in every notification email an explanation of how to unsubscribe from this emails.
5. When i go to my profile page(after login) and uncheck the “allow notifications” checkbox and click update, i will no longer receive this project notification emails.

* **Use Case:**
  + **Name:** Daily project notifications.
  + **Actor:** any user(Student, Faculty-Staff, Pi-CoPi)
  + **Preconditions:** user has registered with the vip website
  + **Description:** An event is triggered by the system in the morning each day.

The system will send every user that hasn’t opted out of notifications an email

with a list of new projects created(approved) since the last time the notifications

were sent if there are new projects, else, no email is sent.

User receives email and clicks in one of the project links.

System presents the user with the detailed project description page.

**[vip/#1263] Allow admin to set a personalized email signature**

* **Description:** As an administrator, i would like to be able to change the email signature shown in all the email notifications from the vip website from the admin panel.
* **Acceptance Criteria:**

1. As an admin, when i go to the admin panel, i can see a textArea/textBox labeled email signature that has as a value the currently configured email signature.
2. As an admin, when i change the email signature from the admin panel and click the change button below it, the signature should be immediately saved and from that moment on, any email sent by the website should have the new email signature

**[vip/#1264] Allow admin to modify email notification settings**

* **Description:** As a PI, I would like the ability to change the default admin address used in notifications to another email address of my choice. I would like to be able to make these modifications via the admin panel.
* This story will allow the PI to easily make changes to the admin email without having to modify the source code.
* **Acceptance Criteria:**

1. The following “Admin Email” settings can be viewed, modified, and saved via the admin panel:
   1. The current / active e-mail address
   2. A list of alternate, user-defined e-mail addresses to select from
2. User must have EXACTLY one (1) active e-mail address to save settings.
3. Admin email settings are retrieved from and saved to a “settings” collection on the data base.
4. VIP website retrieves and uses the active email address whenever an automated message is sent to the admin.

* **Use Case:**
  + **Name:** Adding new, active email address to email settings.
  + **Actor:** Admin / Pi
  + **Preconditions:** User with Pi privileges has logged into VIP website
  + **Description:** User accesses admin panel page. Using the provided input form, the user adds a new entry to the “Admin Email” list. Active e-mail is toggled by selecting (clicking) the desired e-mail address on the list. User commits changes by pressing the “Change” button.

**[vip/#1265] Fix appearance of table headers in Admin Panel**

* **Description:** As an Admin / Pi, I would like the header on the admin panel table to have an orderly, easy to read appearance. A clean and consistent interface will improve user experience and increase productivity while using the admin table.
* **Acceptance Criteria:**

1. Table header is clean and uniform in appearance. Header text and input boxes are aligned.
2. Input boxes no longer shrink to an unreadable size when no matching data is found.

**[vip/#1269] Switch the website from http to https**

* **Description:** As a user, when i enter the website(especially in the login page) i would like the website to be using encryption(https instead of http) to send my information over the network.
* **Acceptance Criteria:**

1. As a user, when i go to any page in the vip website, the protocol used will be https as to make all information shared between me and the website encrypted
2. As a user, when i go to any page in the vip website using the http protocol instead of https, i will get redirected to/forced to use https to access the website.

**[vip/#1270] Secure the vip web rest api**

* **Description:** As a Pi or a developer, when a user makes a request to the vip web api, i would like that some authorization check is performed to make sure that the user making the request is logged in if necessary and if he/she should be allowed to perform that operation.
* **Acceptance Criteria:**

1. As a Pi/Developer, when a request is made to the vip web rest api, i would like for it to be authorized/checked to see if he(the user making the request) has permission to(should be allowed to) make this request.
2. As a Pi/Developer, when a request is made to the vip web rest api and the request should only be allowed for logged in users, i would like requests from users not logged into the system to be refused with a 401 Unauthorized error code.
3. As a Pi/Developer, when a request is made to the vip web rest api and the request should only be allowed for some specific users or type of users, i would like requests from any users not logged into the system or logged in but not the right type of user(or the right user) to be refused with a 403 FORBIDDEN error code.

**[vip/#1272] Enhance alert / to-do management system**

* **Description:**As an Admin/Faculty, I would like a more efficient way to address, dismiss or delete alerts (to-dos). Currently, to-dos are only marked as “done” if the task is accessed via the to-do panel. I would like a “smart” to-do system that automatically senses when an alert has been handled, and updates the to-do panel appropriately. This will help prevent hundreds of alerts from accumulating, and make new alerts more meaningful.
* **Acceptance Criteria:**

1. To-dos are retrieved from the database on a per-user basis. Website should only retrieve to-dos containing the owner ID of the current user.
   1. Pi/CoPi to-dos are not associated with any particular user ID and are viewable by all Pi/Copi users on the system.
2. (All) Users are able to clear to-dos from their inbox en-masse. Users are given the option to:
   1. delete all to-dos in their inbox
   2. delete to-dos aged one week (7 days) and older
   3. delete to-dos aged one month (30 days) and older
3. Pi/CoPi to-dos regarding project proposals and/or student applications are automatically detected and removed from the inbox whenever the proposal / application request is handled.

* **Use Case 1:** 
  + **Name:** Indirectly dismissing to-do messages via project review page
  + **Actor:** Admin / Pi
  + **Preconditions:** 
    - User with admin / Pi privileges is logged into the VIP website
    - There exists at least one (1) project pending approval visible on the “Review Project Proposals” page.
    - Pi/CoPi to-do message corresponding to above project proposal is visible on the to-do page, and has not yet been marked as read.
  + **Description:** Pi/Copi navigates to the “Review Projects Proposals” page. Pi/Copi can either accept or reject the desired project. Upon accept / reject, system will automatically detect the correct “to-do” and remove it from the PiCopi’s to-do inbox.
* **Use Case 2:** 
  + **Name:** Bulk deleting to-do messages
  + **Actor:** Any registered, non-guest user (Student, Faculty, Pi/Copi)
  + **Preconditions:** 
    - User is logged into the VIP website
  + There exists at least one (1) to-do available to the user in the to-do inbox.
  + **Description:** User navigates to the to-do inbox by clicking on the notification icon on the page header. The user proceeds to clear to-do messages by clicking the button with the trash icon. The to-dos should be correctly cleared based on the criteria selected.

**[vip/#1274] Required skills and knowledge fields should be more user friendly**

* **Description:** As a user, when i propose a project or apply to one, i would like the “required skills”/”skills and experience” textBox to offer the option to select from a list of options as well as being able to enter arbitrary text(skills) into a textbox.
* **Acceptance Criteria:**

1. As a non-student user, when i propose a project, i would like the “required skills and knowledge box” to be large enough(multiline textArea).
2. As a non-student user, when i propose a project, i would like the “required skills and knowledge box” to offer a dropdown with options/skills others users have selected in the past that is filtered based on what i have written so far and when i click on one, it should be added to the “required skills and knowledge box”.
3. As a student user, when apply for a project, i would like the “required skills and knowledge box” to be large enough(multiline textArea).
4. As a student user, when i apply for a project, i would like the “required skills and knowledge box” to offer a dropdown with options/skills others users have selected in the past that is filtered based on what i have written so far and when i click on one, it should be added to the “required skills and knowledge box”.
5. As a user, when i’m either applying for a project or proposing one, i would like the options from the dropdown in the “required skills and knowledge box” to contain all skills previously entered by any user before me.

* **Use Case**
  + **Name:** Project Proposal with Improved Required Skills and Knowledge.
  + **Actor:** Faculty-Staff, Pi-CoPi
  + **Preconditions:** user has registered with the vip website, has verified his email and was also verified by the Pi/CoPi
  + **Description:**
    - User logs in to the web site.
    - User is presented with the landing page by the system.
    - User clicks in the “Propose a Project” button in the landing page.
    - User is presented with the “Project Proposal” page/form.
    - User types the title of the project in the Project title textBox.
    - User selects the desired semester for the project in the semester dropdown.
    - User types in a description for the project.
    - User selects the “Required skills and Knowledge” textBox.
    - The system presents the user with a list of all Skills previously entered into the system by other users in that same textBox.
    - User clicks one of the options/skills shown by the system right below the “Required skills and Knowledge” textBox.
    - The system adds the selected skill to the “Required skills and Knowledge” textBox and puts a comma after it so the user can enter more skills.
    - User scrolls down and clicks submit.
    - The system saves the project and it also saves whatever skills the user entered that were not yet in the list of options.
    - The system sends emails to both the proposer and the Pi to respectively let them know that the system logged/created the project and that he/Pi needs to approve or reject the project.
    - The system displays a success message to the user right away so that the user knows his submission was successful.

**[vip/#1281] Change empty emails being sent to the Admin email account**

* **Description:** As an administrator, when i receive a notification email, it will always have enough information to be useful.Note the notification emails referred to here are:
  + Project cleared/user remove from project notification.
  + Project proposal approved notification.
  + Project proposal rejected notification.
  + Student project application approved notification.
  + Student project application rejected notification.
* **Acceptance Criteria:**

1. As an administrator, when i receive a notification email, it will never have an empty subject or an empty text/body.
2. As an administrator, when i receive a notification email, it will have enough information for me to act on it whenever an action on my part is required.

**[vip/#1283] Modify existing projects notifications to avoid spamming users**

* **Description:**As a user of the vip system, when the existing projects notifications are sent, i would like to only receive the notifications if i either am a new user(fall 2017 or newer) or i’m an older user that has explicitly opted in to receive them. I would also like to be able to opt out of receiving this notifications
* **Acceptance Criteria:**

1. As a user of the vip system who registered before fall 2017, when the existing projects notifications are sent, i would not like to receive them unless i explicitly opted in to receive notifications from my profile page.
2. As a user of the vip system who registered on or after fall 2017, when the existing projects notifications are sent, i would like to receive them unless i explicitly opted out either in my profile page or by using a link in the notifications themselves.
3. As a user of the vip system who has opted in to the system to receive notifications, when the existing project notifications are sent, i would like to have a link at the end of the email that i can click on and become unsubscribed from the notifications.
4. As a user of the vip system, when i’m logged into the system and go to my profile page, i would like to have a checkbox that i can use to indicate whether i want to receive this notifications or not.

## Non-Feature User Stories

**[vip/#1227] Test Release 4 User Stories**

* **Description:** Evaluate whether user stories from VIP Release 4.0 (live server) are functional and up to specifications described.
* **Acceptance Criteria:**

1. All user stories in release 4.0 have been tested either manually or via automation (Selenium).
2. All test results have been documented.

**[vip/#1228] Test release 5 user stories**

* **Description:**I must test all features from the stories from the previous release.
* **Acceptance Criteria:**

1. When i test each story in the previous release, the features developed match what was explained in the acceptance criteria in the corresponding stories.

**[vip/#1259] Combine bug fixes into dev website**

* **Description:** As a PI, I would like bug fixes from task #1245 to be merged and pushed onto the dev server. This will be done so that the project owners are able to test new features on one centralized server.
* **Acceptance Criteria:**

1. Dev server is updated with all stories from prior Sprints (1, 2).
2. Developers have drafted a set of testing instructions to be sent to the PO’s.
3. Project owners have received testing instructions.
4. Project owners have reviewed changes and notified PIs of any necessary fixes.

**[vip/#1267] Switch to npm3 to allow git checkout on windows**

* **Description:** As a developer, i would like to switch to from using npm 2.\* to npm 3 to allow for the vip web git repository to be checked out on windows.

* **Acceptance Criteria:**

1. The node\_modules folders are removed from the git repositories.
2. There are two build scripts one for windows and one for linux/mac that will pull the dependencies.
3. When npm version 3 is installed in the dev machine and the right build script is run, all the dependencies of the project should be installed and the project(the web server) should be runnable.

## Pending User Stories

**[vip/#1271] Reduce amount of data sent from the api where possible**

* **Description:**As a user of the system, when in a page that provides history/logs of any kind, i would like paging to be implemented so as to show only a portion of the history at a time. Also, i would like for filters to be added to the data sent from the api whenever possible so as to reduce the overall bandwidth used by the website(for example, the retrieve todos api).
* **Acceptance Criteria:**

1. As a user, when i visit the review student applications, review faculty registration, review project proposals or any other page that contains “transactions history” of some kind, i would like the history to be shown paged(a few, maybe 20-30, items at a time instead of all at the same time) to allow for faster page loads.
2. As a user, when i visit any page in the vip website that makes a request to the rest api that retrieves all of the data in a mongo collection, i would like the data retrieved from the api to be filtered at the api(or mongo) layer so as to reduce the bandwidth used by the application and improve performance(loading speed of pages mainly).

**[vip/#1273] Allow projects to span across multiple semesters**

* **Description**: As an Admin / Faculty, I would like the ability to select multiple semesters at a time while I am proposing or editing a project. This feature will replace the current single-semester system which requires that each project be transitioned from semester to semester manually. Once implemented, the new option will allow a project to stay active during the selected range of semesters. This feature will alleviate the need of manually editing each project at the end of the term.

* **Acceptance Criteria:**

1. During project proposal / edit, user is given the option to select more than one (1) terms for a project to belong to.
2. … (Needs to be furthere developed)

**[vip/#1275] Add a request changes option in the review project proposals page**

* **Description:** Add a request changes option in the review project proposals page

Request changes with comments instead of simply rejecting a project and making the proposer reenter all the information just for a few changes. This story needs to be better defined when we plan to implement it.

* **Acceptance Criteria:** (Undeveloped)

**[vip/#1276] At some point, the PI concept made sense, as there would be one PI for a project and then associated faculty. I don’t think it makes sense any longer. There should be: Students, Faculty/Staff, Admin**

* **Description:** At some point, the PI concept made sense, as there would be one PI for a project and then associated faculty. I don’t think it makes sense any longer. There should be: Students, Faculty/Staff, Admin. That’s all.
* People should not be able to apply for Admin.
* Admin should be created internally.
* This story needs to be better defined when we are planning to work on it.
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1277] Admin panel and User Profile icons should have a text label under them**

* **Description:**Admin panel and User Profile icons should have a text label under them saying what they link to.This story needs to be better defined once we plan to implement it.
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1278] How do I know which users are register for a project for a given semester**

* **Description:** How do I know which users are register for a project for a given semester This story needs to be better defined by the product owners before we can work on it.
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1285] Edit/Delete users from the admin panel**

* **Description:** As an administrator, when i go to the admin panel, i would like to have a way to modify or delete the information of users of the system. Note this story needs to be refined when more details are decided on(such as how the UI for this will look like, etc...)
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1286] Admin panel users table filters don't work correctly**

* **Description:** As an administrator, when i’m in the admin panel and i enter text in one or more of the filters in the users table, i would like to see the records/users filtered correctly as per my choices/filters. Note this story needs to be refined.
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1287] Export excel files does not work on mobile devices**

* **Description:** As an administrator, when i’m in a mobile device and i click the excel export icon in the admin panel, i would like for the excel file with all the desired information(users in the table) to be downloaded to my mobile device successfully.
* Acceptance criteria need to be added by the developer that picks up this story.
* **Acceptance Criteria:** (Undeveloped)

**[vip/#1288] Disable terms functionality in the admin panel does not work**

* **Description:**As an administrator, when i disable a term in the admin panel, i would like projects from those terms to be completely excluded from everywhere in the website.

Note this is part of a bigger effort to keep track of the terms each project(and possibly each users too) is related to.

* **Acceptance Criteria:** (Undeveloped)

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

**Hardware:**

-Development:

-2 x Dual core 2.5GHz CPU, 8GB RAM, 200GB RAM: approximate specs for development laptops used in the project.

- Quad core 2GHz CPU, 200GB HDD, 4GB RAM: for hosting both the web server and mongo server in the main test environment.

- 3 x 8-core AMD FX-8300 , 320GB HDD, 8GB RAM: for hosting other testing environments in Docker containers in a Kubernetes cluster.

-Production Server:

- Quad core 2GHz CPU, 200GB HDD, 4GB RAM: for hosting both the web server and mongo server in production.

-Client:

- Any device running the latest version of Google Chrome, Opera, Safari, Mozilla Firefox or Microsoft Edge with an internet connection with a minimum download speed of 1Mbps should be enough to use the website/system.

**Software:**

Development:

-NodeJS v4.3.2: NodeJS interpreter

-NPM v2.14.12: NodeJS package manager

-MongoDB v3.2.4: database server

-Kubernetes v1.6: container orchestration engine

-Docker v17.06.0-ce: container engine

-Teamcity v2017.1.1: build server

-IntelliJ Idea v2017.1.4: IDE

-Robomongo v1.0.0: Mongo client with a GUI

-Mingle(web): project management software

-Github(web): git repository posting

-Google Drive(web): file sharing

Production:

-Ubuntu 14.04.3 LTS: production server OS

-NodeJS v4.3.2: NodeJS interpreter

-NPM v2.14.12: NodeJS package manager

-MongoDB v3.2.4: database server

## Sprints Plan

### Sprint 1

**Attendees:** Jose Ponce, Masoud, Francisco, Dafna Steinberg

**Start time**: Monday, May 22 2017 at 8:00 am

**End time:** Monday, May 22 2017 at 9:00 am

After discussion, the velocity of the team were estimated to be 50 (25+25).

* The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.
  + User Story: Test VIP 4.0 release (#1227)
  + User Story: Test release 5 user stories(#1228):

As a user i would like to test user stories to best know how to use this website

* The team members indicated their willingness to work on the following user stories.
  + Jose C Ponce

User Story Test release 5 user stories(#1228)

* + Dafna Steinberg

User Story (#1227): Test VIP 4.0 release via the live server (vip.fiu.edu).

Evaluate whether user stories from VIP Release 4.0 (live server) are functional and up to specifications described. This will be achieved either via manual or automated testing. Test results should be documented.

### Sprint 2

**Attendees:** Jose Ponce, Masoud, Francisco, Dafna Steinberg

**Start time:** Tue, May 30th 2017 at 8:53 am

**End time**: Tue, May 30th 2017 at 9:05 am

After discussion, the velocity of the team were estimated to be 100 (50+50) hours.

* The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.
  + Finish creating tests for R5.(Jose)(#1243)
  + Finish creating tests for VIP R 4.0 (Dafna Steinberg) (#1244)
  + Fix bugs found during creation of Selenium Tests(on dev)(Jose)(#1245)
  + Fix bugs found in VIP 4.0 that have carried over to V5 (dev). (#1245) (Dafna)
* The team members indicated their willingness to work on the following user stories.
  + Jose Ponce: #1243, #1245
  + Dafna Steinberg #1244, #1245

### Sprint 3

**Attendees**: Dafna, Jose

**Start time**: Mon Jun 12, 2017 at 5:00 PM

**End time:** Mon Jun 12, 2017 at 5:10 PM

After discussion, the velocity of the team were estimated to be 100 (50 + 50 hours)

* The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.
  + [vip/#1259] Combine bug fixes into dev website
  + [vip/#1261] Create temporary emailing lists / address to be used during development
  + [vip/#1260] Allow Admin to login as different user types
* (The rest in no particular order:)
  + [vip/#1262] Notify users about new projects added since last notification
  + [vip/#1266] Fix formatting on email notifications
  + [vip/#1263] Allow admin to set a personalized email signature
  + [vip/#1265] Fix appearance of table headers in Admin Panel
  + [vip/#1264] Allow admin to modify email notification settings
  + [vip/#1267] Switch to npm3 to allow git checkout on windows
* The team members indicated their willingness to work on the following user stories.
  + Dafna Steinberg
    - [vip/#1259] Combine bug fixes into dev website (6 hrs)
    - [vip/#1261] Create temporary emailing lists / address to be used during development (4 hrs)
    - [vip/#1265] Fix appearance of table headers in Admin Panel (4 hrs)
    - [vip/#1264] Allow admin to modify email notification settings (30 hrs)
  + Jose
    - [vip/#1260] Allow Admin to login as different user types (25 hours)
    - [vip/#1262] Notify users about new projects added since last notification (8 hours)
    - [vip/#1266] Fix formatting on email notifications(5 hours)
    - [vip/#1263] Allow admin to set a personalized email signature(2 hour if #1264 is implemented before it 7 hours otherwise)
    - [vip/#1267] Switch to npm3 to allow git checkout on windows(5 hours)

### Sprint 4

**Attendees:** Dafna Steinberg , Jose Ponce

**Start time:** Mon, Jun 26 3:00 PM

**End time**: Wed Jun 28 9:00 AM

After discussion, the velocity of the team were estimated to be 100 (50 + 50)

* The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.
  + [vip/#1269] Switch the website from http to https
  + [vip/#1272] Enhance alert / to-do management system
  + [vip/#1273] Allow projects to span across multiple semesters
  + [vip/#1270] Secure the web rest api
  + [vip/#1281] Change empty emails being sent to the Admin email account
  + [vip/#1274] Required skills and knowledge field in the Propose project page should be more user friendly
* The team members indicated their willingness to work on the following user stories.
  + Dafna Steinberg
    - [vip/#1272] Enhance alert / to-do management system
    - [vip/#1273] Allow projects to span across multiple semesters
    - (BUG FIX) [vip/#1160] Add text box whispers to Social Icon Links
  + Jose Ponce
    - [vip/#1269] Switch the website from http to https
    - [vip/#1270] Secure the web rest api
    - [vip/#1281] Change empty emails being sent to the Admin email account
    - [vip/#1274] Required skills and knowledge field in the Propose project page should be more user friendly

# System Design

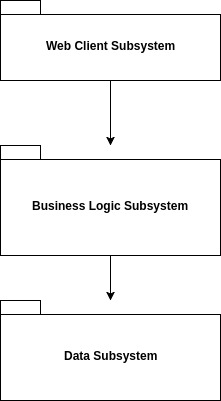
This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

The main architectural patterns chosen for this website were chosen simply because they have been traditionally used to solve exactly this type of problem with great success.

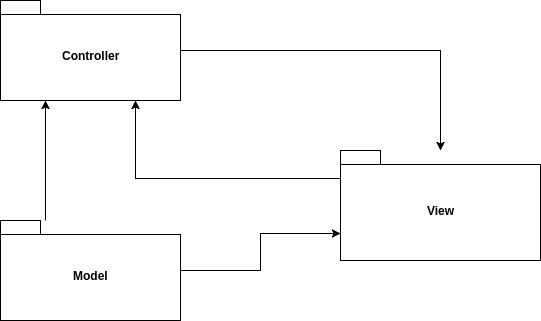
**3-tier architecture:**

This was the main pattern used and it describes the architecture of the system as a whole. This decision was fueled mainly by the fact that this pattern has been used in client-server software for a long time and is a proven way of architecting the same kind of system we were building. Furthermore, it prescribes a careful separation of the business logic/layer and the presentation layer. This is a significant advantage because it leaves open the possibility of adding other presentation layer applications/technologies in the future. Because of this, it would be possible, just to present an example, to create(in the future) a mobile application without it affecting any of the other subsystems(for the most part at least).



**MVC:**

We chose to use the MVC pattern in the web client subsystem simply because it is the right way of doing web development with angular, or at least most people seem to think it is. I would like to note that in the case of angular, many experts choose to use the term MVVM instead but given the striking similarities among the two, it is my opinion that we can for the most part treat them as one and the same. This pattern encourages a clear separation of concerns between the view, controller and model layers that makes the application simpler for new developers to understand and also easier to maintain. I believe this are important characteristics given the fact that new students will pick up this project every semester that may not be familiar with the software stack(MEAN) but they are almost always familiar with the MVC pattern since it’s one of the most commonly used in web development.



## System and Subsystem Decomposition

As explained above the VIP system is mainly decomposed into three subsystems: Web Client Subsystem, Business Logic Subsystem, Data Subsystem. Each one of these is explained in turn below:

**Web Client Subsystem:**

This subsystem corresponds to the Presentation Tier in a 3-tier architecture. As explained above, in the future, an extra subsystem could be created at this same tier(Presentation) to offer other interfaces to our customers, for example, an Android or Iphone application. It is the only interface for our customers/clients into the system. This subsystem was also implemented/designed by using the MVC architectural pattern, as explained above. It runs completely in the client’s browser and it interacts only with the Business Logic Subsystem to accomplish any operation that would somehow affect(or query) the state of the system. It is designed to contain all of the presentation layer logic leaving the remaining subsystems blissfully unaware of the details of the user interface of our system.

**Business Logic Subsystem:**

This subsystem corresponds to the Logic Tier in a 3-tier architecture. This is the core of the system. It contains, as it should if you follow a traditional 3-tier system approach, all of the business logic of the system. As such, it’s the only part of the system that would be nearly impossible to drastically change, that is, unless there are drastic changes to the problem domain itself(which is unlikely). This subsystems lives completely on a backend server running our NodeJS-Express web server whose main concern is to host a web api. In general this api is concerned with the fulfillment of requests from the presentation tier, which currently consists only of the Web Client Subsystem, that will alter(or query) the state of the system. Most importantly, this is the only subsystem allowed to decide when the state of the system changes and how. Finally, this is the only subsystem that is allowed to interact with the Data Subsystem.

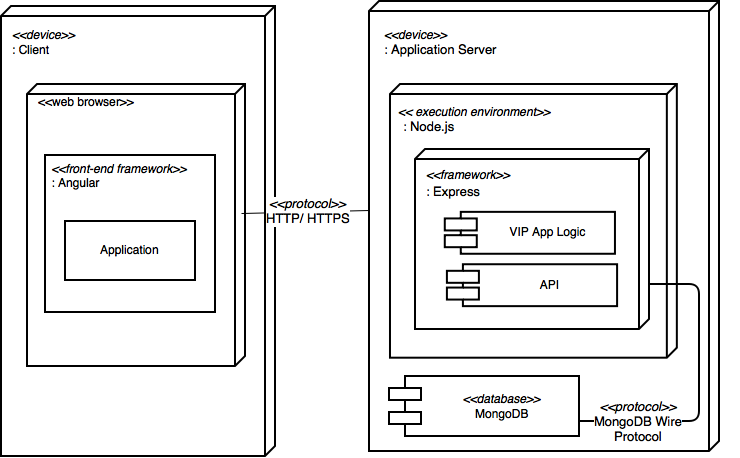
**Data Subsystem:**

This subsystem corresponds to the Data Tier in a 3-tier architecture. This is a backend “service” and should only be accessible by the Business Logic Subsystem for security/data integrity reasons. Important to note that includes the database access logic as well as the database server itself. As such, part of this subsystem has to be deployed with the Business Logic Subsystem(the database access logic) whereas the other part(the actual database server) can be located elsewhere. Note that as of right now both the Business Logic Subsystem and the Data Subsystem are deployed to the same server but there is no reason why that has to be the case, so long as the database is properly secured from access by anyone other than the Business Logic Subsystem. In our case the database management system is MongoDb and it’s obvious that our Mongo server(and the data it contains) make up the most important part of this subsystem.

## 

## 

## Deployment Diagram



## 

## Design Patterns

**Facade:**

This pattern's main characteristic/goal is to simplify interaction with a complex system. In our case, it is used to simplify the interaction of the Web Client Subsystem with the Business Logic Subsystem. The facade design pattern is implemented throughout the Web Client Subsystem in any place where interaction with the web api in the Business Logic Subsystem is required. Services are created to abstract the interaction with the api as well as to abstract all details of the http protocol itself from the controllers in our frontend. This not only simplifies the system but it avoids unnecessary code duplication and increases development speed since all boilerplate code to interact with the api is already contained in facade services.

**Repository:**

The main characteristic/goal of this pattern is to prevent data access logic from being spread out in a disorganized manner throughout your code, while at the same time avoiding unnecessary duplication. The repository pattern was implemented in the Data Subsystem, and is in the part of this subsystem that needs to be deployed with the Business Logic Subsystem. It’s implementation allowed the Business Logic Subsystem a simple and unified interface into the Data Subsystem and similar to the application of the facade pattern in the Web Client Subsystem it simplified the interaction between subsystems while at the same time avoiding code duplication.

# System Validation

## Unit Testing

* **Test case ID:** VIP-1262-U1 (Sunny day scenario)
* **Description/Summary of Test:** There is one new project(approved yesterday) and a user who has not opted out of notifications and then the sendNotifications method of the notificationService is triggered and this will cause an email to be sent to the user with a link to the project specified above.
* **Pre-condition:**
* There is a project approved with date of yesterday in the database.
* There is a user who hasn’t opted out of notifications in the database.
* The notificationService’s sendNotifications method is triggered.
* **Expected Results:**
* An email will be sent to the user specified above.
* The email sent to the user will contain a link to the project details page of the project specified above.
* **Actual Result:**
* An email will be sent to the user specified above.
* The email sent to the user will contain a link to the project details page of the project specified above.
* **Status:** PASS
* **Test case ID:** VIP-1264-U1
* **Description/Summary of Test:** User enters the input email@email.com into the text

area. User clicks the [Add New] button to append email to list, and the [Change] button

to commit changes.

* **Pre-condition:**
* User with Pi privileges has logged into the VIP website.
* Web browser is open to the admin panel page
* **Expected Results:**
* New entry []email@email.com persists after refreshing the admin panel or navigating off the page.
* **Actual Result:**
* New entry []email@email.com persists after refreshing the admin panel or navigating off the page.
* **Status:** PASS
* **Test case ID:** VIP-1264-U2
* **Description/Summary of Test:** User changes active email-address to ‘email@email.com’ by selecting the entry []email@email.com from the list. The user than clicks the [Change] button to commit changes.
* P**re-condition:**
* User with Pi privileges has logged into the VIP website.
* []email@email.com entry has been previously added to the email list
* Email list contains at least one other entry. Exactly one (1) of these email

addresses should be selected as the “active” e-mail [x]any@any.com

* Web browser is open to the admin panel page
* **Expected Results:**
* The entry [x]email@email.com remains marked as active after refreshing the admin panel or navigating off the page.
* **Actual Result:**
* The entry [x]email@email.com remains marked as active after refreshing the admin panel or navigating off the page.
* **Status:** PASS
* **Test case ID:** VIP-1272-U1
* **Description/Summary of Test:** Deleting to-do notifications older than one week of age.
* **Pre-condition:**
* User is logged into VIP website
* There exists at least one (1) to-do belonging to the user and visible in the to-do inbox.
* **Expected Results:**
* All to-dos aged seven (7) days and older are removed from the to-do inbox. Deletion persists when the to-do inbox is refreshed.
* **Actual Result:**
* All to-dos aged seven (7) days and older are removed from the to-do inbox. Deletion persists when the to-do inbox is refreshed.
* **Status:** PASS
* **Test case ID:** VIP-1274-U1 (Sunny day scenario)
* **Description/Summary of Test:** There is a registered confirmed(email and Pi) staff/faculty user already in the system(call him Tom). Tom proposes a project and in the project proposal page, he selects a skill from the list shown below the skills textArea and then submits his project.
* **Pre-condition:**
* Tom is logged into the system.
* Tom is in the project proposal page.
* Tom fills out the form as follows:
  + Title: “nada”
  + Semester: “spring 2017”
  + Description: “nada tampoco”
* For skills, he selects java from the list presented.
* Tom then hits submit.
* **Expected Results:**
* An email will be sent to the user specified above saying that his project proposal was received by the system.
* An email will be sent to the admin to approve/reject the project.
* A success dialog is shown to the user within seconds.
* **Actual Results:**
* An email will be sent to the user specified above saying that his project proposal was received by the system.
* An email will be sent to the admin to approve/reject the project.
* A success dialog is shown to the user within seconds.
* **Status:** PASS

## Integration Testing

* **Test case ID:** VIP-1262-G1 (No email is sent when there are no new projects)
* **Description/Summary of Test:** There are no new project(approved yesterday) and there is at least one user who has not opted out of notifications and then the sendNotifications method of the notificationService is triggered and no emails should be sent since there are no projects to notify users about.
* **Pre-condition:**
* There are no projects approved with date of yesterday in the database.
* There is at least one user who hasn’t opted out of notifications in the database.
* The notificationService’s sendNotifications method is triggered.
* **Expected Results:**
* No emails will be sent in response to the sendNotifications method being called.
* **Actual Result:**
* No emails were sent.
* **Status:** PASS
* **Test case ID**: VIP-1264-G1 (Admin email settings, sending emails)
* **Description/Summary of Test:** Tester modifies admin email settings to use an e-mail address of their choosing. (We will refer to this address as any@any.com ). User submits project edit form, which toggles an automated message sent to thee-mail address any@any.com.
* **Pre-condition:**
* User with Pi privileges has logged into the VIP website.
* User has ownership of at least one active project.
* Tester has access to inbox of email address used during testing.
* **Expected Results:**
* Tester receives email regarding a project edit proposal at the address any@any.com
* **Actual Result:**
* Tester receives email regarding a project edit proposal at the address any@any.com
* **Status:** PASS
* **Test case ID:** VIP-1272-G1 - Matching and removal of todos
* **Description/Summary of Test:** User accepts or rejects a project on the project proposal interface. System matches selected project to a “to-do” by using the project title, proposer’s name and a timestamp. Corresponding to-do task is then automatically removed from the user’s inbox.
* **Pre-condition:**
* User with admin / Pi privileges is logged into the VIP website
* There exists at least one (1) project pending approval visible on the “Review Project Proposals” page.
* **Expected Results:**
* The correct to-do is removed from the user’s inbox. All other to-dos should remain intact and unaltered.
* **Actual Result:**
* The correct to-do is removed from the user’s inbox. All other to-dos should remain intact and unaltered.
* **Status:** PASS
* **Test case ID:** VIP-1272-G2 - Matching and removal of todos related to identically-named projects
* **Description/Summary of Test:** User accepts (or rejects) one of two projects with identical project names. System matches the selected project to a “todo” by using the project status, the proposer’s name, and a timestamp.
* **Pre-condition:**
* User with admin / Pi privileges is logged into the VIP website
* There exists at least two (2) projects pending approval visible on the “Review Project Proposals” page. These two projects share an identical project name.
* **Expected Results:**
* The correct to-do is removed from the Pi’s to-do inbox. The other to-do correlated to the similarly named project should remain intact. All other unrelated to-dos should also remain unaltered.
* **Actual Result:**
* The correct to-do is removed from the Pi’s to-do inbox. The other to-do correlated to the similarly named project should remain intact. All other unrelated to-dos should also remain unaltered.
* **Status:** PASS
* **Test case ID:** VIP-1274-G1 - New skills are saved in the database
* **Description/Summary of Test:** There is a registered confirmed(email and Pi) staff/faculty user already in the system(call him Tom). Tom proposes a project and in the project proposal page, he types in a skill not in the list shown below the skills textArea and then submits his project.
* **Pre-condition:**
* Tom is logged into the system.
* Tom is in the project proposal page.
* Tom fills out the form as follows:
  + Title: “nada”
  + Semester: “spring 2017”
  + Description: “nada tampoco”
* For skills, Tom types a skill not present in the list of skills shown to him..
* Tom then hits submit.
* **Expected Results:**
* An email will be sent to the user specified above saying that his project proposal was received by the system.
* An email will be sent to the admin to approve/reject the project.
* The new skill entered by Tom is added to the database and will be shown in the future to users who propose projects.
* A success dialog is shown to the user within seconds.
* **Actual Result:**
* An email was sent to the user specified above saying that his project proposal was received by the system.
* An email was sent to the admin to approve/reject the project.
* A success dialog was shown to the user within seconds.
* New skills was successfully saved to the database.
* **Status:** PASS

# Glossary

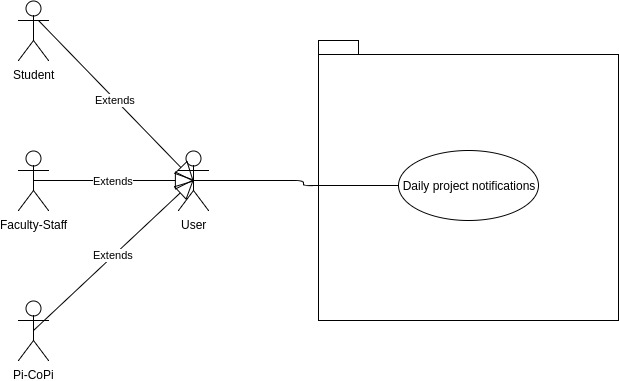
* Guest - Lowest level user on the site can only view the homepage, and projects
* Student - Entry level user, can join projects
* Faculty/Staff - Mid Level user, can create projects, and accept students into projects
* Pi/CoPi - Top level user, full access to all of sites features, serves as administrator to the site. Projects - Created by Faculty/Staff and Pi/CoPi team undertakings that are proposed on the site Terms - Semesters that are accessible on the site, used to archive projects
* Skill item - necessary knowledge set on the system that can be stated in a project details to match students knowledge base

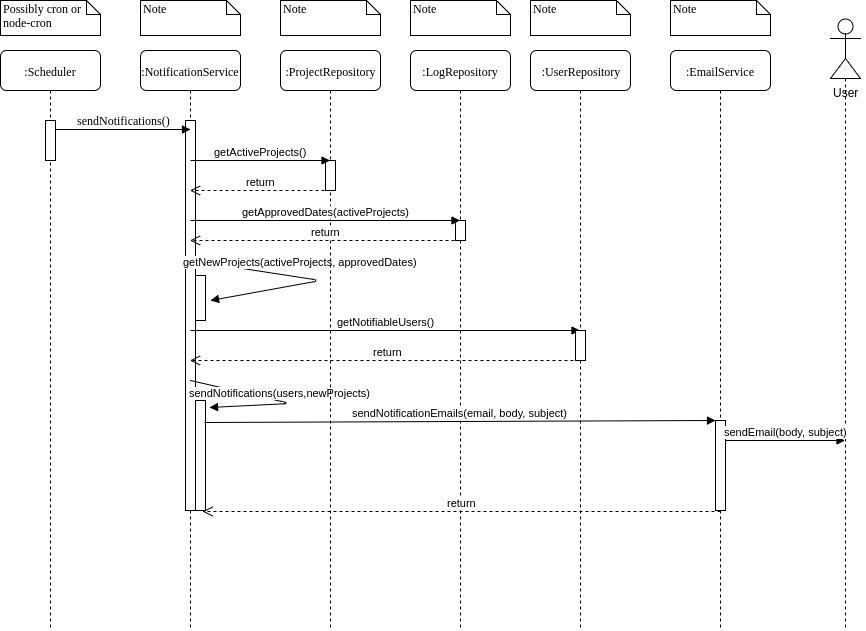
# 

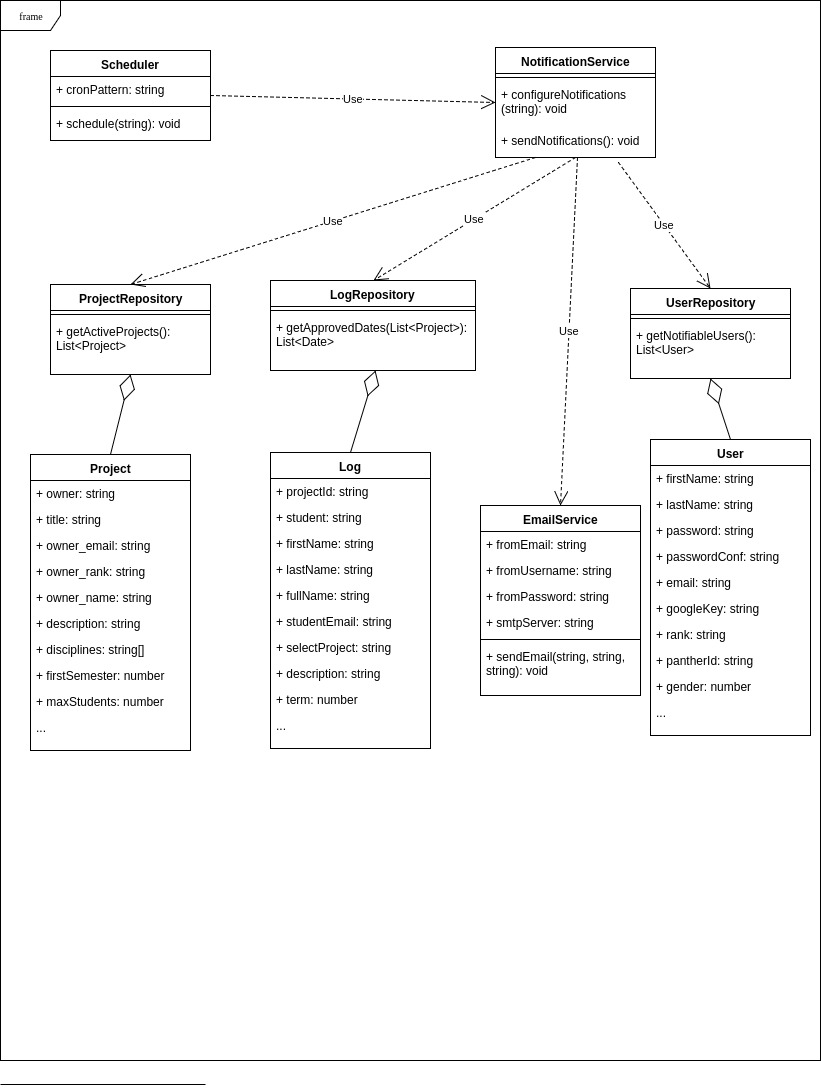
# 

# Appendix

## Appendix A - UML Diagrams

**[vip#1262]-Fig.1:** *“Notify users about new projects added since last notification”, use-case diagram*

**[vip-#1262]-Fig.2:** *“Notify users about new projects added since last notification”, sequence diagram*

**[vip#1262]-Fig.3:** *“Notify users about new projects added since last notification”, class diagram*

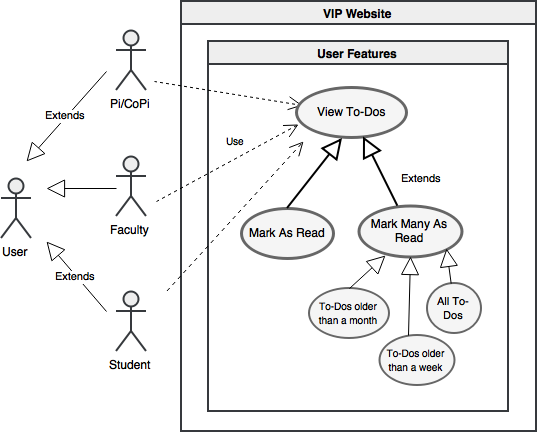
## usecase_1.png

**[vip#1264]-Fig.1*:*** *“Allow admin to modify email notification settings”, use-case diagram*

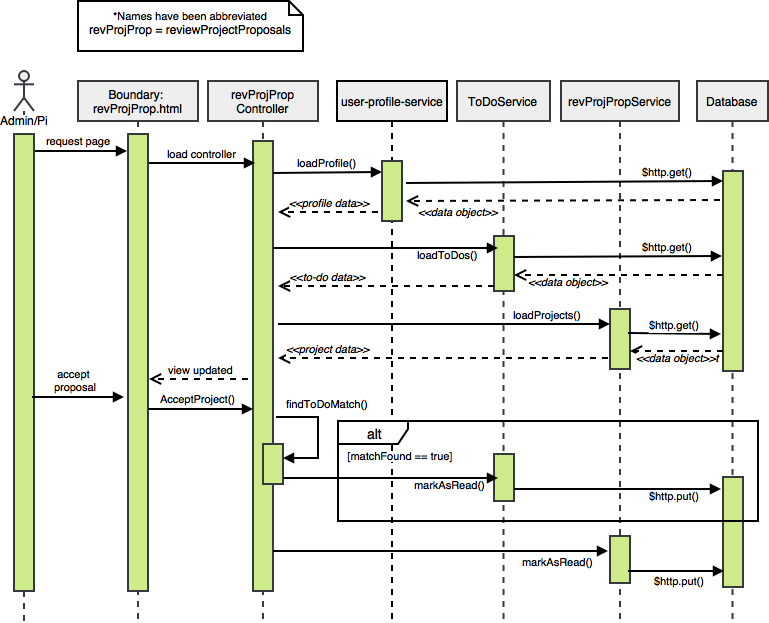
## sequence_1.png

**[vip#1264]-Fig.2*:*** *“Allow admin to modify email notification settings”, sequence diagram*

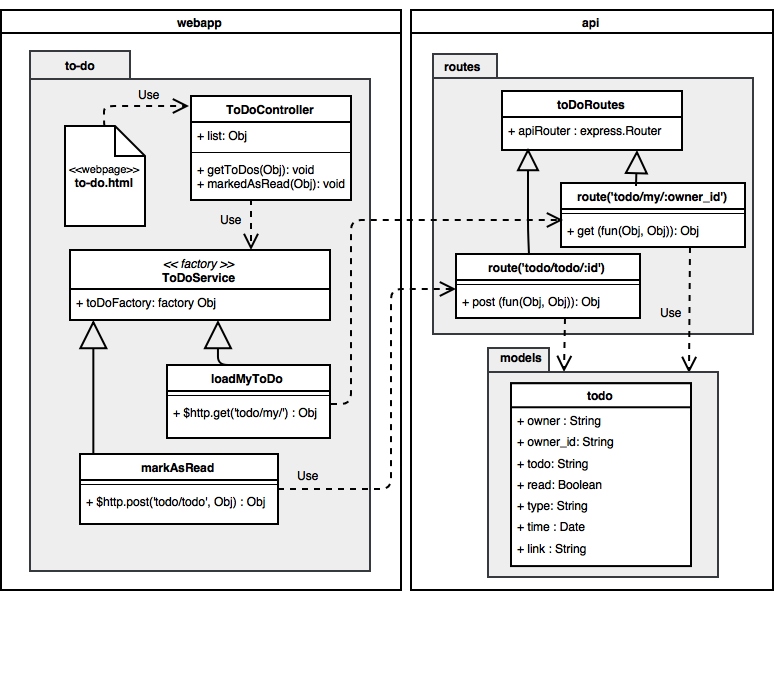
## classDgrm_redo.png[vip#1264]-Fig.3: *“Allow admin to modify email notification settings”, class diagram*



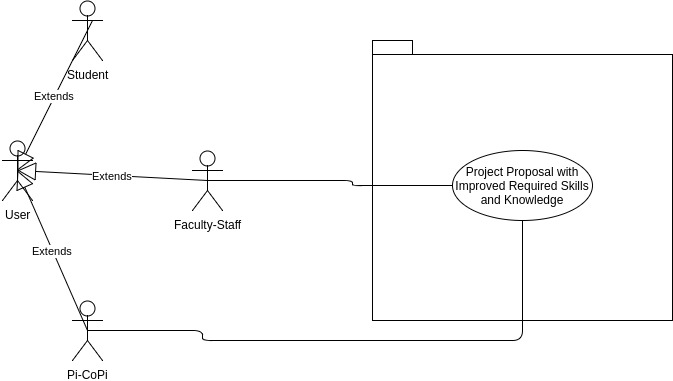
**[vip#1272]-Fig.1:** *“Enhance alert / to-do management system”, use-case diagram*



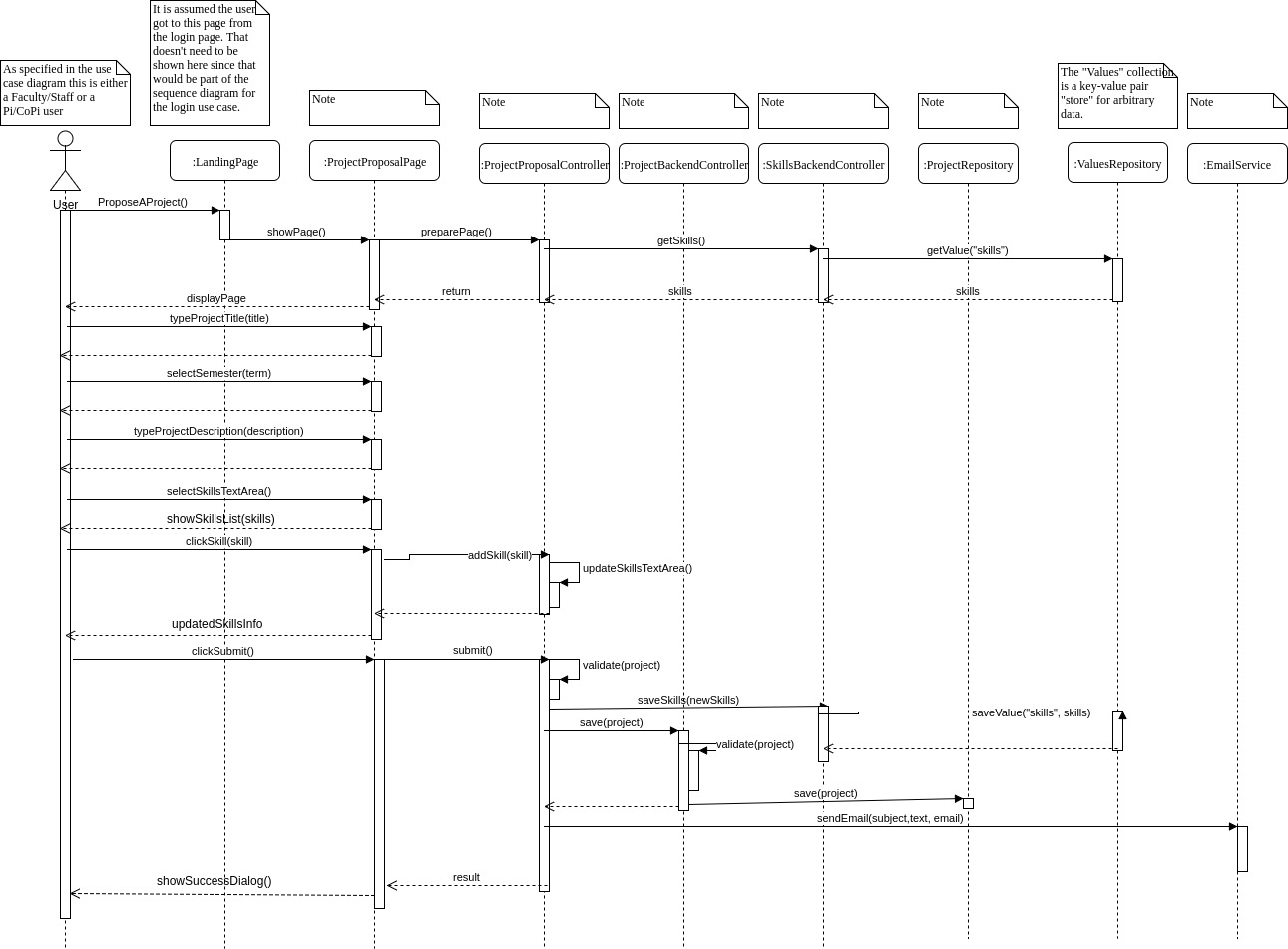
**[vip#1272]-Fig.2:** *“Enhance alert / to-do management system”, sequence diagram*



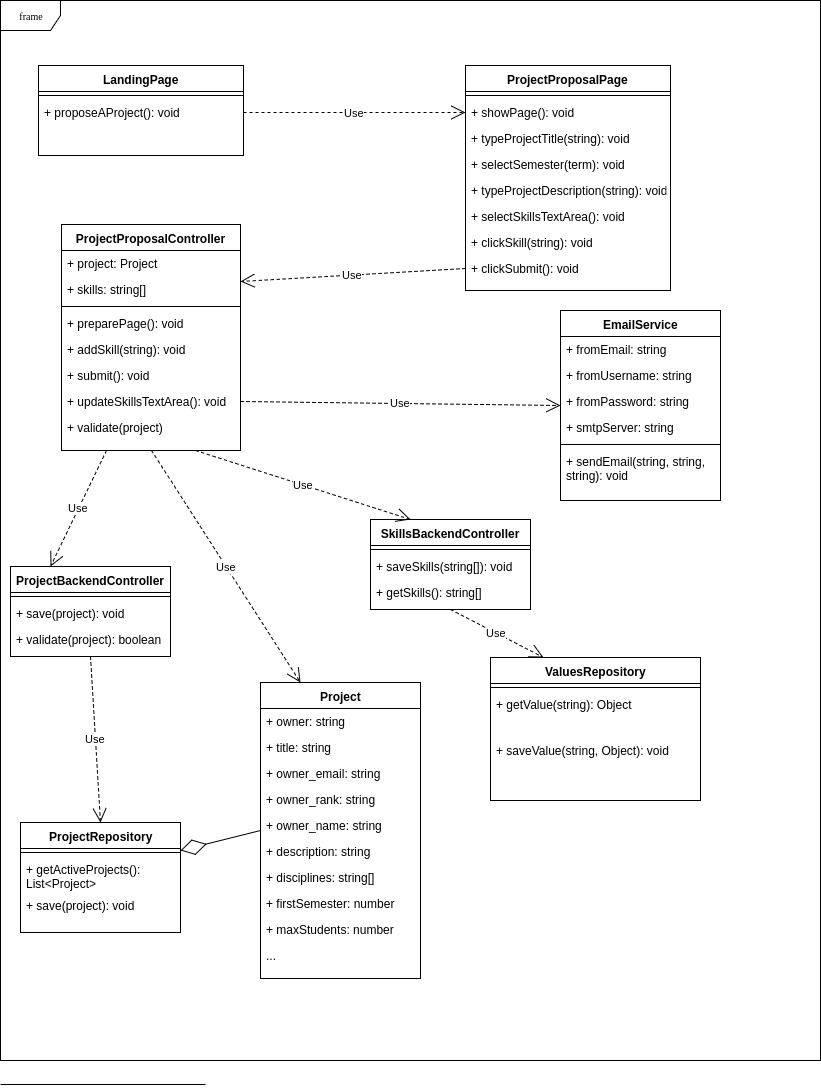
**[vip#1272]-Fig.3:** *“Enhance alert / to-do management system”, class diagram*



**[vip#1274]-Fig.1:** *“Required skills and knowledge fields should be more user friendly”, user diagram*

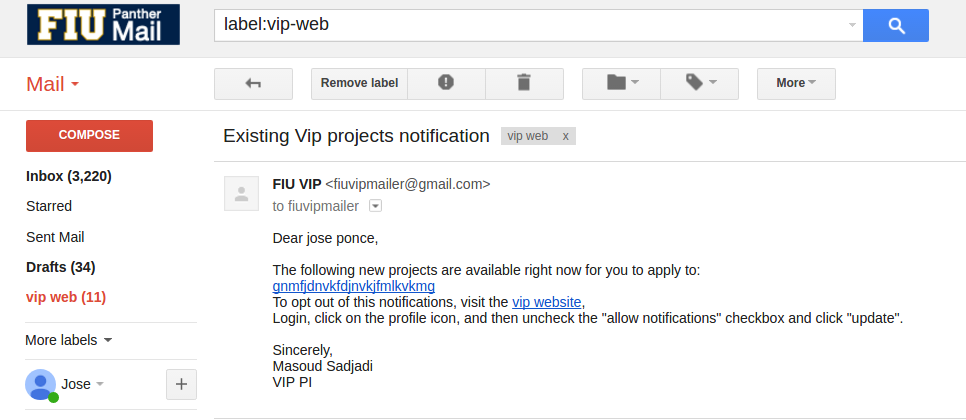


**[vip#1274]-Fig.2:** *“Required skills and knowledge fields should be more user friendly”, sequence diagram*

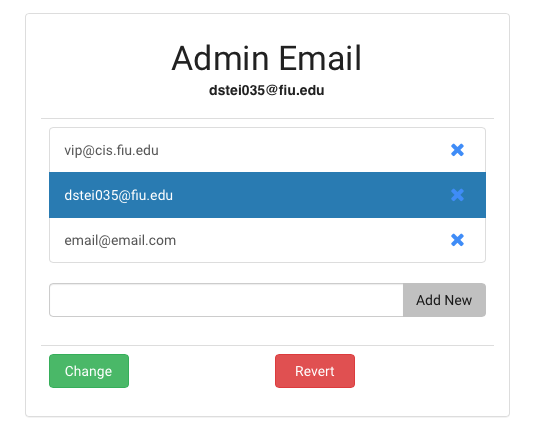


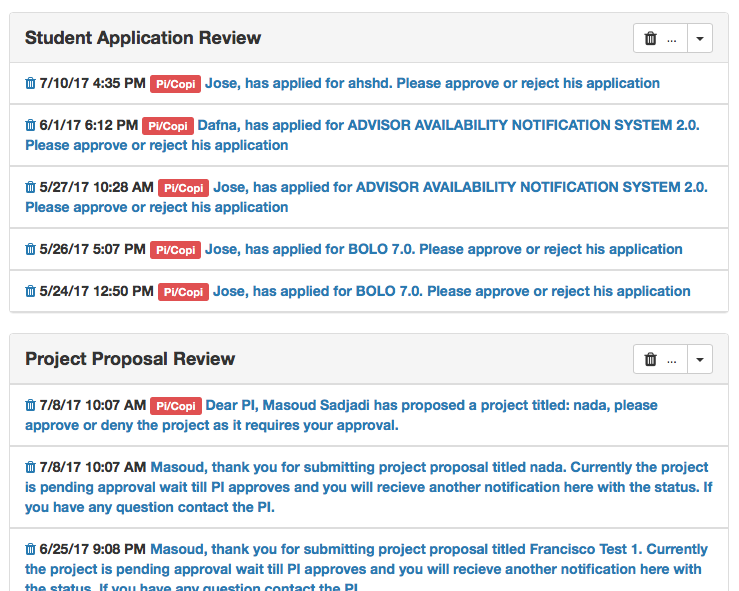
**[vip#1274]-Fig.3:** *“Required skills and knowledge fields should be more user friendly”, class diagram*

## Appendix B - User Interface Design

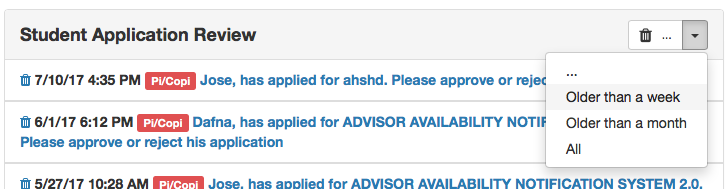


**[vip#1262]-Fig.4***: “Notify users about new projects added since last notification”, screenshot of mail notification in user’s inbox*

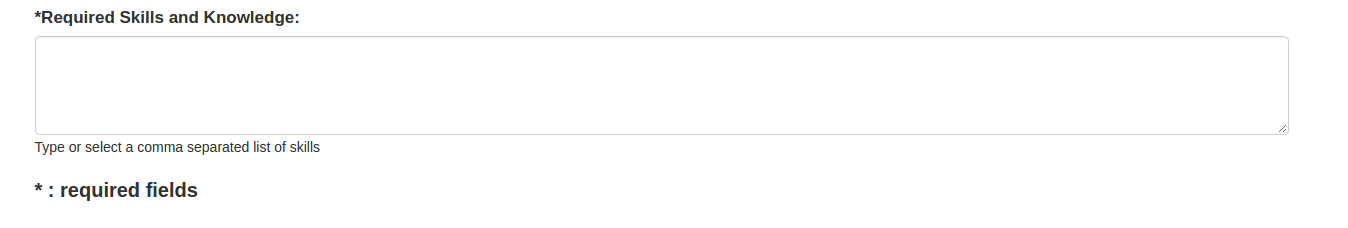
**[vip#1264]-Fig.4:** *“Allow admin to modify email notification settings”, Admin e-mail section of admin panel page.*



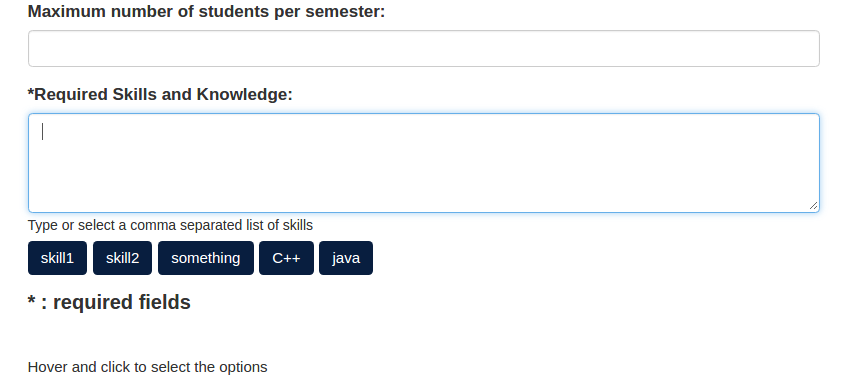
**[vip#1272]-Fig.4:** *“Enhance alert / to-do management system”, Red tags have been added on to-dos that require Pi/CoPi activity. New “trash” button added on header of every to-do section and provides a way for to-dos to be cleared in bulk.*



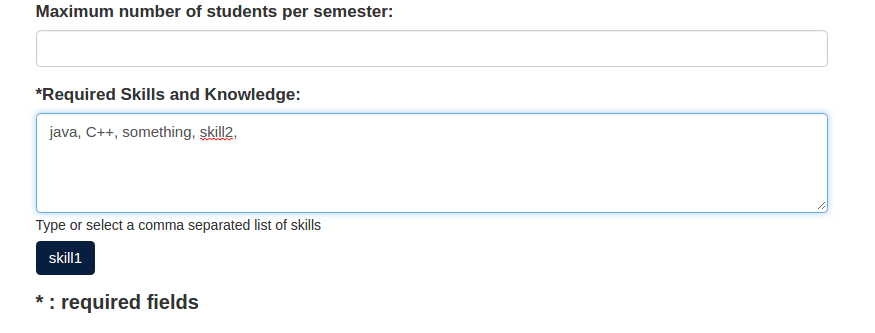
**[vip#1272]-Fig.5:** *“Enhance alert / to-do management system”, Drop-down showing options for to-do deletion criteria.*



**[vip#1274]-Fig.4:** *“Required skills and knowledge fields should be more user friendly”, unselected skills text area*



**[vip#1274]-Fig.5:** *“Required skills and knowledge fields should be more user friendly”, selected skills text area*



**[vip#1274]-Fig.6:** *“Required skills and knowledge fields should be more user friendly”, skills text area after skills have been selected.*

## 

## 

## Appendix C - Sprint Review Reports

### Sprint 1

**Attendees:** Jose Ponce, Masoud, Francisco, Dafna Steinberg

**Start time**: Tuesday, May 30 2017 at 8:16 am

**End time:** Tuesday, May 30 2017 at 8:46 am

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

* VIP 4.0 Selenium Testing - Ongoing; current test scripts seem to be acceptable, will continue and complete remaining stories. (Dafna Steinberg)
* VIP 5.0 Selenium Testing: Done with the exception of one pending story.(Jose)

### Sprint 2

**Attendees**: Dafna, Jose Ponce, Masoud, Francisco

**Start time:** 2:00 PM

**End time:** 3:00 PM

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

* Jose:
  + #671: Notification of existing projects.
  + #1146: Make login permanent.
  + #1245: Fix bugs found during sprint 1 testing.
* Dafna Steinberg :
  + [vip/#1244] Test remaining VIP 4.0 User Stories
  + [vip/#1245] Fix bugs found during testing in Sprint 1 Release 6 (Task 1251)

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.

NONE

***Sprint 3***

**Attendees**: Dafna Steinberg, Jose Ponce

**Start time:** Mon Jun 26, 2:00 PM

**End time**: Mon Jun 26, 2:30 PM

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

* Dafna Steinberg:
  + [vip/#1265] Fix appearance of table headers in Admin Panel
  + [vip/#1264] Allow admin to modify email notification settings
  + [vip/#1259] Combine bug fixes into dev website
* Jose Ponce:
  + [vip/#1260] Allow Admin to login as different user types
  + [vip/#1262] Notify users about new projects added since last notification
  + [vip/#1266] Fix formatting on email notifications
  + [vip/#1263] Allow admin to set a personalized email signature
  + [vip/#1267] Switch to npm3 to allow git checkout on windows

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.

* [vip/#1269] Switch the website from http to https. This story was incomplete and will continue to be worked in sprint 4.
* [vip/#1270] Secure the web rest api. This story was incomplete and will continue to be worked in sprint 4.

### Sprint 4

**Attendees**: Dafna Steinberg, Jose Ponce

**Start time:** Mon Jul 10, 2:00 PM

**End time:** Mon Jun 10, 2:30 PM

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

* Dafna Steinberg:
  + [vip/#1272] Enhance alert / to-do management system
  + (Bug fix) [vip/#1160] Add text box whispers to Social Icon Links
* Jose Ponce:
  + [vip/#1269] Switch the website from http to https
  + [vip/#1270] Secure the vip web REST api
  + [vip/#1281] Change empty emails being sent to the Admin email account
  + [vip/#1274] Required skills and knowledge fields should be more user friendly
  + [vip/#1283] Modify existing projects notifications to avoid spamming users.

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.

* [vip/#1273] Allow projects to span across multiple semesters
  + Was moved to backlog, and will be addressed during the next release cycle of the VIP website

## Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

## User Manuals

1. **Student Manual:** 
   1. Logging In
      1. Click on the “Log In” button on the right top corner of this website.
      2. Students do not need to make an account and may login through google using their FIU email account. Click the red “G” button and follow the prompts.
      3. If the user has cookies enabled, he / she will stay logged into the site on subsequent visits.
   2. Editing Profile / Opt-out from Notifications
      1. Log in to the VIP website (see IA).
      2. Click on the user icon found on the page header. (Icon can be found directly left of the “Log Out” button).
      3. Use the form and follow instructions to update profile information.
      4. To disable e mail notifications, uncheck the “Allow Notifications” box found near the bottom of the form.
      5. Press the “Update” button to commit changes.
   3. Viewing Projects
      1. Click on the “Projects” link on the header navigation.
      2. To view more detail, click on the project’s name. This will allow you to see a project’s team members, as well as any associated videos and / or links.
   4. Applying to Project
      1. Log in to the VIP website (see IA).
      2. To access a project application, you may…
         1. Click on the “Apply For a Project” button
         2. Use the header navigation. An application link can be found under “Prospective Students” >> “Apply”
         3. Click the “Apply” button located on the right hand side while viewing a project detail page (IC).
      3. Fill out the application form with the instructions provided.
      4. Upon submitting the form, your application will be sent out for approval. Your application will be reviewed by staff. If accepted, you will be added to the project roster.
   5. Leaving Project
      1. Log in to the VIP website (see IA).
      2. Go to the project’s detail page (IC).
      3. Click the “Leave” button located on the right side of the projects detail page.
      4. To finish, dismiss the dialog message by selecting the “I’m sure” button.

1. **Faculty Manual:** 
   1. Account Creation / Logging In
      1. Click on the “Log In” button on the right top corner of this website.
      2. To create an account, click on the “Create an account” link under the “Non-Student Login” section. Note that all faculty accounts MUST be associated with an fiu e-mail address.
      3. On the “Log In” page, enter your email and password in the forms under the “Non-Student Login” section. Press the blue “Log in” button to submit.
      4. If the user has cookies enabled, he / she will stay logged into the site on subsequent visits.
   2. Editing Profile / Opt-out from Notifications
      1. Log in to the VIP website (see IIA).
      2. Click on the user icon found on the page header. (Icon can be found directly left of the “Log Out” button).
      3. Use the form and follow instructions to update profile information.
      4. To disable e mail notifications, uncheck the “Allow Notifications” box found near the bottom of the form.
      5. Press the “Update” button to commit changes.
   3. Proposing Project
      1. Log in to the VIP website (see IIA).
      2. To propose a project, you may…
         1. Click on the “Propose a Project” button
         2. Use the header navigation. A project proposal application link can be found under “Faculty” >> “Propose Project”
      3. Fill out the project proposal form with the instructions provided.
      4. Upon submitting the form, your project proposal will be sent out for approval. If accepted, the project will added to the website.
   4. Editing Project
      1. Log in to the VIP website (see IIA).
      2. Click “Projects” on the navigation bar. Go to the projects details page by clicking the name of the project that you wish to edit.
      3. On the projects detail page, click on the “Edit” button on the right hand side.
      4. Fill out the project edit form. This process is identical to proposing a project (see IIC).
      5. Upon submitting the form, your edits will be sent to admin for approval. The unedited project will still be available for viewing on the Projects page.

* 1. Applying to Project:
     1. Log in to the VIP website (see IIA).
     2. To join a project, you may…
        1. Click on the “Apply For a Project” button
        2. Click the “Join” button located on the right hand side while viewing a project detail page (IC).
     3. Fill out the application form with the instructions provided. Form in similar to the one that students fill out when applying to join a project.
     4. Upon submitting the form, your application will be sent out for approval. Your application will be reviewed by staff. If accepted, you will be added to the project roster.

* 1. Leaving Project
     1. Log in to the VIP website (see IIA).
     2. Go to the project’s detail page (IC).
     3. Click the “Leave” button located on the right side of the projects detail page.
     4. To finish, dismiss the dialog message by selecting the “I’m sure” button.

1. **Pi/CoPi Manual:** 
   1. Account Creation/ Logging in
      1. Click on the “Log In” button on the right top corner of this website.
      2. To create an account, click on the “Create an account” link under the “Non-Student Login” section.
      3. On the “Log In” page, enter your email and password in the forms under the “Non-Student Login” section. Press the blue “Log in” button to submit.
      4. If the user has cookies enabled, he / she will stay logged into the site on subsequent visits.
   2. Project Proposal Review
      1. Log in to the VIP website (see IIIA)
      2. Access the review projects proposal page by going to “Faculty” >> “Review Project Proposals” on the header navigation menu.
      3. Projects may be accepted by pushing the green “Yes” button, or rejected by pushing the red “No” button.
      4. If you wish to put an accepted / rejected project back into review, find the appropriate project under “Review Project Proposal History” and press “Undo”.
   3. Faculty Registration Review
      1. Log in to the VIP website (see IIIA)
      2. Access the user registration review page by going to “Prospective Students” >> “Review Faculty Registrations” on the header navigation menu.
      3. Faculty account registration requests can be accepted by pushing the green “Yes” button, or rejected by pushing the red “No” button.
      4. If you wish to move a faculty account back into review, locate it under “Review Faculty Application History” and press “Undo”. (\*Note, this only works for previously-accepted users. To review a rejected account, please ask that user to attempt signing up again.)
   4. Student Application Review
      1. Log in to the VIP website (see IIIA)
      2. Access the student application review page by going to “Prospective Students” >> “Review Student Applications” on the header navigation menu.
      3. Student application requests can be accepted by pushing the green “Yes” button, or rejected by pushing the red “No” button.
      4. If you wish to move a student application back into review, locate it under “Review Student Application History” and press “Undo”. (\*Note, this only works for previously-accepted students. To review a rejected application, please ask the user to submit a new application.)
   5. Admin Panel
      1. Log in to the VIP website (see IIIA)
      2. Access the admin panel by clicking the wrench icon on the top-right corner of the page header.

## Installation/Maintenance Document

Note that the most important parts of what follows are on the youtube videos from VIP-6.0

1. **Development Environment Installation:**
   1. Pre-requisites:
      1. NodeJS(and npm) installed: Any version after the one mentioned in the “Hardware and Software Resources” section would be fine. <https://nodejs.org/en/>
      2. MongoDB: You will need a MongoDB server to run this application, if you don’t have one, you can install and run it on your own development laptop/desktop, no special configuration is required. Any version after the one mentioned in the “Hardware and Software Resources” section would be fine. <https://www.mongodb.com/download-center#community>
      3. Git: <https://git-scm.com/downloads>
      4. An IDE is recommended: any IDE that can deal with NodeJS/HTML/CSS should do.
   2. Execute:
      1. Checkout the git repository for the project, if you don’t know where it’s located just ask your Product Manager. From here on i will refer to the folder where the code was checked out from git as {{gitCheckoutDir}}.
      2. “Install” the application by running “{{gitCheckoutDir}}/Code/install.sh” for linux/MAC or “{{gitCheckoutDir}}/Code/install.bat” for windows. Note this only does an npm install in the “{{gitCheckoutDir}}/Code/” and “{{gitCheckoutDir}}/Code/Deployment/” folders. If you don’t understand what that means, you may want to research it. If this script fails, then there is a problem with your npm install.
      3. Configure the application to run locally:
         1. Make sure the host and protocol are correct in the “{{gitCheckoutDir}}/Code/server.js” file. You normally want that file to have two lines like this near the beginning of the file:

app.set("host", "localhost");

app.set("protocol", "http");

* + - 1. Make sure the “{{gitCheckoutDir}}/Code/config/config file looks like this:

module.exports = {

'port':3000,

'externalPort':3000,

'database':'mongodb://localhost:27017/admin',

'secret':'whateveryouwant',

'secure':false

};

Adjust the mongo url if your mongoDB is not on your development machine.

* + 1. Run the application by executing “node {{gitCheckoutDir}}/Code/server.js” into your terminal. If this fails, then there is a problem with your nodejs installation.
    2. You can now access the website at “<http://localhost:3000>”
    3. You can create a PI account with sadjadi’s email for testing purposes on your local environment since that account requires no verification.

\*Note that google+ login(student login) will not work in your local environment.

1. **Running Selenium Tests:**
   1. The Selenium tests are in a java maven project. You may either install maven and build it yourselves but it’s considerably simpler to have your IDE do that for you.
   2. Once your ide can compile the project, which is under “{{gitCheckoutDir}}/Tests/SeleniumTests”, install the Firefox Selenium Driver as indicated in <https://developer.mozilla.org/en-US/docs/Mozilla/QA/Marionette/WebDriver>.
   3. After that, use your test runner of choice to run the junit tests under “{{gitCheckoutDir}}/Tests/SeleniumTests/src/test/java/edu/fiu/vip\_web/vip\_r5\_stories/tests”.

Note that most java IDEs can run junit tests without any work on your part so I recommend you use one.

1. **(IMPORTANT!) Adding javascript or css files to the website:**
2. **Background:**

If you look at the “{{gitCheckoutDir}}/Code/webapp/index.html” file you will see that almost the only javascript and css files referenced are “distrib/styles.css” and “distrib/scripts.js”. This is because all the js and css files are combined into one(one js and one css) so as to minimize the number of browser requests and therefore improve performance of the website. The files in the distrib folder are auto-generated whenever the web server starts(it’s triggered from server.js).

1. **Consequences:**
2. All js and css files added to the system will need to be added in the “{{gitCheckoutDir}}/Code/deployment/gulpfile.js”. See [this document](https://docs.google.com/document/d/1k8B7o3CNcZ6_--V0AZIpoperG9l19JnrivG545zHkhk/pub) for details on how to add them.
3. If changes are made to any of the js files(or css), the server has to be restarted for the changes to take effect.

Note that all of this means you will not add any link(for css) or script(for js) tags to the index.html file since adding the files to the gulpfile.js file mentioned above will take care of this files to be delivered to the browser/client.

1. **Production Installation(this also includes the vip-dev server):**

\*\* Note that it is very important that the deployment folder stays the same(/var/www/VIP-FINAL/) so that the certificate renewal process works automatically.

* 1. Stop the web server:
     1. “sudo forever stopall” should work, if it doesn’t then execute “sudo kill $(pidof nodejs)” to force close/kill the process.
  2. Copy the Code folder to the “/var/www/VIP-FINAL/” folder into the server(vip-dev or production):
     1. Note it has to be this folder and not elsewhere, nor can this folder be renamed because there is a cron job in the server that will make sure the certificates are put there whenever they are renewed and this process would be broken/useless if you deploy elsewhere/rename the folder.
     2. Note that if any changes to the certificateRenewal.bash file are to be really used, this file needs to be copied to the “/var/www/” folder and you have to execute “sudo chmod 711 /var/www/certificateRenewal.bash” afterwards to make the script executable.
  3. Https/ssl certificate setup (copy certificates to the deployment folder):
     1. Run

sudo cp /etc/letsencrypt/live/vip-dev.cis.fiu.edu/cert.pem /var/www/VIP-FINAL/Code/

* + 1. Run

sudo cp /etc/letsencrypt/live/vip-dev.cis.fiu.edu/chain.pem /var/www/VIP-FINAL/Code/

* + 1. Run

sudo cp /etc/letsencrypt/live/vip-dev.cis.fiu.edu/privkey.pem /var/ww w/VIP-FINAL/Code/

Note that in all of the above commands you need to change “vip-dev.cis.fiu.edu” by “vip.fiu.edu” when in the production server.

* 1. Configure /var/www/VIP-FINAL/Code/api/config/config:
     1. Make sure /var/www/VIP-FINAL/Code/api/config/config looks like this(use your editor of choice, example, nano or emacs):

module.exports = {

'port':3000,

'externalPort':443,

'database':'mongodb://localhost:27017/admin',

'secret':'whateveryouwant',

'secure':true

};

Note this is different from the configuration on your local environment.

* 1. Configure /var/www/VIP-FINAL/Code/api/config/auth.js:
     1. Make sure /var/www/VIP-FINAL/Code/api/config/auth.js looks like this:

module.exports ={

‘googleAuth': {

'clientID' : '1056710173783-00k7c8he5utpi75h8jtn3183cns3suq1.apps.googleusercontent.com',

'clientSecret': 'gqwU3jy3K-anElJ6Vf3j7Py6',

'callbackURL': 'http://vip-dev.cis.fiu.edu/auth/google/callback'

}

};

Note that in production you want vip-dev.cis.fiu.edu to be replaced by

vip.fiu.edu in the callbackURL.

* 1. Configure /var/www/VIP-FINAL/Code/server.js:
     1. Make sure that the host and protocol are correctly set in the /var/www/VIP-FINAL/Code/server.js file. There should be two lines near the top of the file that start with app.set("host" and app.set("protocol" , make sure they say app.set("host", "vip-dev.cis.fiu.edu"); and app.set("protocol", "https"); respectively.

Note that when deploying to production(and not vip-dev) you want the host to be “vip.fiu.edu” instead.

* 1. Start the web server with forever:
     1. Run sudo forever start /var/www/VIP-FINAL/Code/server.js
  2. Verify server is running:
     1. Perform at least a student login through google+ and a regular login just to ensure the server is correctly configured and running.

1. **Understanding the certificate renewal cron job**
   1. **High level overview of the process:**

From a high level the automated certificate renewal process for the vip website is

quite simple. The first step it to obtain/install the certbot-auto command line utility which we will use to obtain and manage our certificates(note most of the subsequent steps will use this tool). Then, we will obtain a new certificate for the domain you will be serving requests on. Once we have a certificate, we need a process to ensure that it is automatically renewed by having a cron job that executes monthly(it could be more often if you want). This cron job will execute the certificateRenewal.bash which it expects to be under the /var/www/ folder in your server. This bash script will then be the one to call the certbot-auto command line tool, which it expects to be under the /var/www/ folder in your server, to attempt to renew the certificates. If it is too early to renew your certificates(they have more than 30 days left until expiration), then nothing else is done by the script. However, if the certificates were close enough to the expiration date, they will be renewed and the new certificates will be placed in the /etc/letsencrypt/live/{{domain}}/ folder, where {{domain}} should be replaced by your domain. Once that the certificates have been renewed, the script will copy the new certificate files to the website’s root folder, which it will assume to be located at /var/www/VIP-FINAL/Code. Finally, the script will restart the web server so that it can start using the new certificates.

**It is important that you note that as a consequence of the above process, the**

**deployment folder for your code in the dev and prod servers has to be /var/www/VIP-FINAL**. If you violate this, the automated certificate renewal process will “fail”(technically it will work, it just won’t be useful since it will try to copy the new certificates to the wrong folder). It is recommended that you don’t change the deployment folder in the dev and prod environments. However, if you do change it, you will be responsible for updating the cron job and the certificate renewal script accordingly on both servers.

B. **Detailed explanation of the important steps:**

Note all of this has to be done in the hosting(dev or prod) server and some of this operations may need the web server to be restarted to take effect. Also note, that you should not have to do any of the below operations since they have already been performed by the VIP-6.0 team in both dev and prod servers. This is mostly for informational purposes so that future developers are aware of how it all works.

**To obtain a new certificate(linux):**

1. Get certbot if you don’t yet have it in the server:
   1. cd /var/www/
   2. wget <https://dl.eff.org/certbot-auto>
   3. chmod a+x ./certbot-auto
2. Run certbot to get the certificate:
   1. sudo /var/www/certbot-auto certonly --webroot -w /var/www/VIP-FINAL/Code/webapp/ -d vip-dev.cis.fiu.edu
      1. Here -w is the location of your webapp folder and -d is the domain on which you are serving(the domain may be different from the one above)
      2. Note the web server needs to be running before this step, otherwise, you will be unable to “prove you own this domain”. See the documentation at <https://certbot.eff.org/docs> if you want to know what this means.
   2. Reply to any prompts you get
      * 1. Enter the vip admin email when prompted for an email address, if you don’t know it, ask your PM
3. Copy certificate files to the website root:
   1. sudo su
   2. cp /etc/letsencrypt/live/vip-dev.cis.fiu.edu/\* /var/www/VIP-FINAL/Code/
      1. Replace “vip-dev.cis.fiu.edu” by your domain.
4. Restart the vip website.

Note this requires the nodejs server to be running and serving on the domain you want to get the certificate for.

**To renew a certificate manually(linux, where the certificate was obtained through the process above):**

1. cd into the folder you previously downloaded certbot to.
2. ./certbot-auto renew
3. Restart the vip website.

Note this will only renew the certificate if it’s close to expiration(less than 30 days).

**Create a cron job to attempt certificate renewal weekly:**

1. Create cron job to run the script on a regular basis at a convenient time
   1. cd /var/www/
   2. sudo cp /var/www/VIP-FINAL/certificateRenewal.bash ./
   3. chmod 711 certificateRenewal.bash
   4. Open with your text editor of choice the certificateRenewal.bash file and make sure that this line VIP\_WEB\_DOMAIN="vip-dev.cis.fiu.edu" is correct, meaning that the domain is that of the server you are creating the cron job on.
   5. sudo crontab -e
   6. Scroll to the bottom of the file and type the following in a new line:
      1. CronPattern /var/www/certificateRenewal.bash
         1. Here replace CronPattern for an appropriate pattern, for example, ‘0 0 1 \* \*’ without the quotes will attempt renewing the certificates every month on the first day.
   7. Save and Close the file

We got this certificates through Certbot from EFF, you can find more documentation on Certbot at <https://certbot.eff.org/docs>.

This documentation would include how to use their CLI and what we included here is a very small subset of the information found in their documentation.

Note the environment variables referred to in that document(the link) are no longer necessary since they are now found in the certificateRenewal.bash script itself(hardcoded).

## Shortcomings/Wishlist Document

This section will list, and offer a high level explanation of some of the main shortcoming/wishlist items for the VIP System as of the end of release 6.0.

1. *Configuration spread out*

Currently, there is configuration in three different places that needs to be touched whenever the application is deployed. This is confusing and error prone since people are likely to forget one or more files at deployment time. For that reason, i believe the configuration found in the files “.../Code/api/config/config”, “.../Code/api/config/auth.js”, and some of the lines at the top of “.../Code/server.js”, host and protocol, should be consolidated into a single configuration file to reduce the room for error when configuring the application after a deployment.

1. *Term/semester management*

This is probably the main shortcoming of the VIP system today. To illustrate the problem, as of today(VIP-6.0) the Admin/Pi is unable to get a list of active users in the system. That is, a list of users that are active on this semester. The same problem applies to projects to some extent. See below a detailed list of the main problems related to this and some proposed solutions.

* 1. *Admin/Pi user interface to manage semesters:*

The VIP system currently lacks a way for the Admin/Pi to manage

semesters(obviously having to login into MongoDB and changing the data

directly is unacceptable). Therefore, a UI, possibly a page or a section in the

admin panel, should be provided so that the Admin can manage this data.

* + 1. *Active/Inactive:*

Currently, the system has a method of setting terms as disabled/enabled,

however it is ineffective/buggy for several reasons. The main one being

the way that the system is currently choosing what the currently active

semester is. The current logic has several problems. First, it’s only

executed at server startup time and doesn’t change during the application

lifetime. Second, all projects are being given the termId of the “active”

erm, regardless of the term chosen in the project proposal. Third, our

logic, currently ignores the disabled property set when “disabling” a

project through the admin panel, which means that a disabled project

could very well be chosen as the “active” project, which is obviously an

error. While there is often more than one active term in the database, the

current logic will only care about the first one. The PMs need to decide if

more than one term active at a time is a valid scenario or not.

* + 1. *Create new:*

The current method for adding a new term/semester consists of a

developer login into the development or production server and manually

inserting the records into MongoDB. There are two main problems with

this. First, it is error prone since a developer will have to type all this

information into the console. Second, and most important, the current

process doesn’t allow the Admin/Pi to add semesters to the system, which

is the way it should be.

* 1. *Hardcoded terms in some pages:*

Even though there currently is an api to retrieve/modify the list of terms in the

system, some pages still have hardcoded lists of semesters. This is error prone

because a developer will have to update those every semester manually in several

places. It is also, unnecessary since as we already explained before there is an api

for doing this and using it would simply remove this manual step and make the

semester information shown on those pages consistent with the rest of the system.

* 1. *Setting termId on new projects/users:*

As explained above, one of the main reasons for these changes to the way

semesters are treated in the system are so that the system can support some operations that are impossible as of this time. Mainly, listing users/projects per semester. To this end, it was recommended by the PMs that a termId field should be added to the users and projects collections in MongoDB. Note that there is already a termId field in the projects collection but it’s historically been wrong(see “i. Active/Inactive:” above). To this end, it is necessary to keep track of the “current” semester(correctly) so that we can tag every new user in the system as active in the current term. In the case of projects, it would be as simple as making sure the project is assigned the correct termId instead of always setting the “active” term, which is incorrect right now, as the termId of the project regardless of the term the project was really created for.

* 1. *Historical update of the termId field in the projects collection:*

As was already mentioned before, the termId field of the projects in the

MongoDB database has been historically wrong. For this reason, if it is decided by the PMs(and future devs) to really use the termId field in the projects collection, you may want to do a historical update to correct the termId of the existing projects in dev and prod. However, you may just chose to ignore this if you believe it will not cause any issues in the future.

* 1. *Enforcing projects/users for active semester(s):*

To make the changes to the terms/semesters system really useful, they will have to

be enforced. This part is where the developer needs to be really careful since the changes will be everywhere through the front-end as well as in the api. For example, you need to enforce that projects for inactive/disabled semesters don’t show in the projects page as well as in the “apply to project” page, my recommendation is that much of that filtering out of projects should happen in either the api or on the angular service wrappers around the api calls to avoid duplicated code and inconsistencies.

1. *User management by the Admin/Pi:*

Some basic functionality for the Admin/Pi to manage users was requested. This is

initially limited to allowing the Amin/Pi to be able to remove users and edit their information from the admin panel(or some new page, confirm with your PM). Most of the work here would consist of designing/implementing the UI to support this functionality. It is our opinion that some parts may be taken(used as a basis) from the user profile page since the functionality requested here is almost the same as the one offered in that page to all users. We would also like to not that methods to support this functionality are already present in the api so there is no work required at that layer(or the database layer for that matter).

1. *Reduce the amount of unnecessary data being retrieved from the api:*

In several methods in the REST api, there is a huge amount of information being sent as a response. For example, when retrieving the logs/history for student application reviews(or faculty registration reviews), all of the history is being sent back to the browser. This is obviously unnecessary and it slows down the website, it takes a considerable amount of time for the history to show in these pages in the production and development servers. Instead, some type of paging should be implemented, and the records should be filtered at the api instead of sending it all back for the angular code to sort through. This would speed up the response times and would offer a more user friendly view, since no one really wants to see 2.5+ years of history anyways. For example, history could be shown 20-30 records at a time with some paging implemented so that the user can move back and forth in the logs without having to retrieve a massive amount of data. Story #1271 on Mingle was created for this feature.

Note that the same issue occurs in many other api calls and not just in the ones dealing with logs. Whenever possible, the data sent back to the client/browser should be filter at the api layer to reduce the bandwidth used and reduce response times for the client. Also note that this same issue used to be present in the Todo api and was fixed/modified this release so you can use those changes as a reference.

For a more comprehensive list of pending work at the end of the VIP release 6 take a look at the User Stories -> Pending User Stories section.

# References

* “Agile Tutorial for the Senior Project Class School of Computing and Information Sciences Florida International University”

(Adapted from *Agile!: The Good, the Hype and the Ugly* by Bertrand Stabley (ISBN 978-3-319-05154-3).

* Driessen, Vincent “A Successful Git Branching Model” <<http://nvie.com/posts/a-successful-git-branching-model/>>
* Florida International University, WReSTT-CyLE <<http://wrestt.cis.fiu.edu/about-wrestt-com>>
* VIP-Website-6.0 Github Repo

<https://github.com/FIU-SCIS-Senior-Projects/VIP-Website-6.0>