**Final Deliverable**

CIS 4911 – Senior Project U01

**Virtual Job Fair v3.0**

**Members**

Ana L. Hernandez

Manuel Bouza

Enio Pena Navarro

Tomas Acosta

**Instructor**

Masoud Sadjadi

**Mentor**

Masoud Sadjadi

**Date**

July 25th 2014

Copyright © Florida International University 2014. All Rights Reserved. No part of the Virtual Job Fair Project or documentation may be reproduced or modified without the express consent of Florida International University. Redistribution or commercial use is strictly prohibited.

Table of Contents

[1. Introduction 6](#_Toc393998128)

[1.1 Problem definition 6](#_Toc393998129)

[1.2 Scope of System 6](#_Toc393998130)

[1.3 Overall development methodology 7](#_Toc393998131)

[1.4 Definitions, acronyms, and abbreviations 8](#_Toc393998132)

[1.5 Overview of document 9](#_Toc393998133)

[2. Feasibility Study 10](#_Toc393998134)

[2.1 Description of Current System 10](#_Toc393998135)

[2.2 Description of alternative solutions considered. 12](#_Toc393998136)

[2.3 Recommendation of Selected Solution 16](#_Toc393998137)

[3. Project Plan 21](#_Toc393998138)

[3.1 Project Organization 21](#_Toc393998139)

[3.1.1 Project Personnel 22](#_Toc393998140)

[3.1.2 Hardware and Software Resources 22](#_Toc393998141)

[3.2 Identification of Tasks, Milestones and Deliverables 23](#_Toc393998142)

[3.3 Cost of the Project 24](#_Toc393998143)

[4. System Requirements 26](#_Toc393998144)

4.1 Functional and Nonfunctional Requirements 26

[4.2 Requirements Analysis 42](#_Toc393998146)

[4.2.1 Use case model 42](#_Toc393998145)

[4.2.2 Static model 42](#_Toc393998146)

[4.2.3 Dynamic model 43](#_Toc393998147)

[5. System Design 44](#_Toc393998148)

[5.1 Overview 46](#_Toc393998149)

[5.2 Subsystem Decomposition 46](#_Toc393998150)

[5.3 Hardware and Software Mapping 51](#_Toc393998151)

[5.4 Persistent Data Management 52](#_Toc393998152)

[5.5 Security and Privacy 58](#_Toc393998153)

[6. Detailed Design 60](#_Toc393998154)

[6.1 Overview 60](#_Toc393998155)

[6.2 Static model 60](#_Toc393998155)

[6.3   Dynamic model 61](#_Toc393998156)

[6.4 Code Specification 64](#_Toc393998157)

[7 System Validation 67](#_Toc393998158)

[7.1 Subsystem Tests 67](#_Toc393998159)

[7.2 System Tests 72](#_Toc393998160)

[7.3 Evaluation of Tests 93](#_Toc393998160)

[8. Glossary 96](#_Toc393998161)

[9. Appendix 97](#_Toc393998162)

[9.1 Appendix A - Project Performance 97](#_Toc393998163)

[9.2 Appendix B - All Use Case with Nonfunctional Requirements 97](#_Toc393998164)

[9.3 Appendix C - User Interface Designs 13](#_Toc393998165)2

[9.4 Appendix D - Analysis Models 147](#_Toc393998165)

[9.5 Appendix E - Design Models 157](#_Toc393998165)

[9.6 Appendix F - Documented Code 157](#_Toc393998165)

[9.7 Appendix G - Diary of Meeting and Tasks 159](#_Toc393998166)

[10. References 166](#_Toc393998167)

**EXECUTIVE SUMMARY**

The project was initially developed by a group of FIU students in a Senior Project class during spring 2013. The Virtual Job Fair v3.0 is an application that allows students the opportunity to find their ideal job with easy access to multiple job sites with a user-friendly design. Our goal is to provide students, and administrators with interfaces that enables students to easily search for jobs and receive notifications matching their criteria and based on their profile notification settings; similarly, administrator will be able to do multiple tasks not limited to adding/disabling/removing user accounts, configure global notification, configure settings to synchronize with CareerPath website API and manage Virtual Job Fair API keys. In order to accomplish our goals new functionalities were added to the system, and other were modified.

This document merges all the phases of the project into one work product. The document covers the current system with limitations and problems, project plan, and system’s requirements. Also, we present the software architecture with an overall system design covering the implemented use cases. Moreover, we have the detailed design, which gives an insight to the detailed classes and object interaction. Last but not least, the testing process shows the tests performed at the subsystem and system level based on the blueprints provided by the design model.

# Introduction

The introductory chapter gives some background information about the Virtual Job Fair v3.0 system. Firstly, the chapter states the problem definition, and scope of the system. Next, the design methodology used is identified. This methodology includes the software process models and the types of models used. Moreover, definitions, acronyms, and abbreviations of terms that will be used in this deliverable are introduced and explained. Finally, it contains an overview of the whole project, which explains the information contained on each chapter.

## 1.1 Problem definition

Employers looking for talent are always interested in filling out positions with the best possible candidates. In order to accomplish this task, the most effective method to date is to tap local talent, whether it is at universities or job fairs. Due to the given the increasing globalization trend, and the fact that not all employers have the financial or the time capabilities to be registering on every website to seek for potential candidates in different locations. In addition, the current solution provided by universities and job sites is less than ideal.

Our solution to this problem, we will provide FIU Computer Science students with a user-friendly interface where students will be able to get job matches from different job search sites, and not just from the employers who register in the VJF site. Students will receive job match notifications based on their skills and search criteria that they will be able to save for later use. Therefore, this provides employers and potential employees an easy and friendly way to find their match.

## 1.2 Scope of System

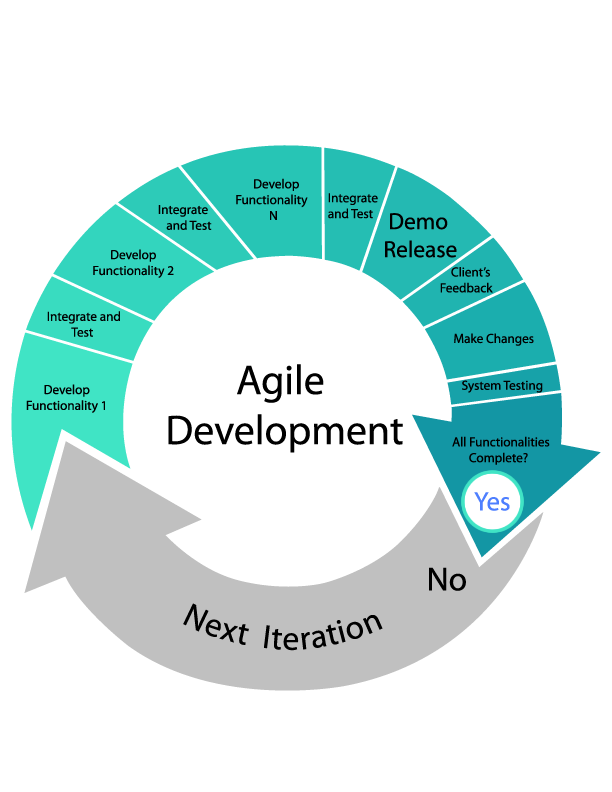
One of the main purposes of the Virtual Job Fair v3.0 is to expand the scope of the previous versions. While past versions provided students with the ability to search for jobs, it was restricted to employers who were registered with our system. In addition, the search engine was limited to specific criteria and complex searches were not possible. Moreover, the notification system lacked customization, and it was very primitive. The students did not have the option to merge and link accounts within the system in order to have a centralized login regardless of which third party they used to login.

Virtual Job Fair v3.0 expanded on existing functionalities of the system. However, the client decided to concentrate on the notifications, multiple logins, advanced search, and API to bring jobs from outside sources into the system. Thus, Virtual Job Fair v3.0 is currently not allowing Employers who register with our system to post jobs. Nevertheless, Employers can now receive customized notifications about students with low matching rate. Students can now search for jobs based on zip code, or distance. Students can now register/login with any other third party account other than Google, FIU, FIU Senior Project, and LinkedIn. Moreover, administrator can perform basic administrative tasks such as: managing users, posts, skills, API, and notifications system.

## 1.3 Overall development methodology

Agile Development Methodology was used in order to add the necessary functionality.  Since Agile is iterative and incremental; we were developing functionalities and testing it along the way. It is very important to follow the rule imposed by this methodology in order to deliver a quality product that conforms and exceeds all standards.

We have found the iterative process to be very simple since it allows us to make small but concise progress towards the completion of the project. We rely heavily in diagrams in order to visualize complicated concepts and devise solutions that are both efficient as well as robust. Below is the Agile Development Methodology diagram which was followed for the completion of this project.



**Figure 1.3-1 Agile Development Model**

We used AgileZen in order to track and document the project progress and requirements. AgileZen allowed us to break the development in pieces and follow the Agile Development Methodology with ease. Also, it provides with features that allowed us to organize the requirement based on priorities, and thus permitted us to focus on each task individually.

## 1.4 Definitions, acronyms, and abbreviations

**Definitions**

-          **Student**: an individual who is currently enrolled in the School of Computing & Information Sciences FIU

-          **Job:** an activity done in exchange for payment

-          **Full-time:** requiring 40 hours or more hours per week

-          **Part-time:** requiring less than 40 hours per week

-          **Paid internship:** an internship for which a student will receive compensation

-          **Unpaid internship:** an internship for which the student will not receive compensation

-          **Benefits:** non-salaried compensation for employees, such as insurance, tuition reimbursement, and retirement benefits

-          **Work authorization:** current legal work status of a student. Categories include U.S. Permanent Resident and U.S. Citizen

-          **Grade point average:** a number out of 4.0 which gives a representation of a student’s grades in his/her classes throughout his/her college career

**Acronyms**

-       **VJF**: Virtual Job Fair

-       **FIU:** Florida International University

-       **GPA:** Grade point average

-       **SCIS:** School of Computing & Information Sciences

-       **USDP:** Unified Software Development Document

**Abbreviations**

As of right now, there are no abbreviations for this project.

## 1.5 Overview of document

The next chapters of this document contain detailed information about the work product of the VJF development project. Chapter 2 goes over the current system and its limitations as well as the possible solutions and alternatives. Section 2.5 goes over the solution and a detailed explanation is given stating the reasons why this approach is significantly better than the other ones.

Chapter 3 presents the project plan that is the team organization; all resources needed such as the cost of the project, and the hardware and software requirements. Chapter 4 goes over the System requirements and includes diagrams that describe our current plan and implementation details. Chapter 5 gives a description of the overall system design, including subsystem decomposition, hardware and software mapping, and persistent data management.

Chapter 6 gives a full description of the system design showing the minimal class diagram for subsystems, detailed class design, and code specification. Chapter 7 discusses all the testing approaches that we used as well as more details about the tools we used and the test cases.

Chapter 8 is a glossary of the most important terms used in this document. Chapter 9 consists of a list of appendices for this document.

Appendix A contains the project schedule. Appendix B contains the current and new functional and non-functional requirements of the system, as well all the use cases. Appendix C shows the user interface designs. Appendix D shows the Object Diagrams Analysis. Appendix E gives detail about the subsystems implemented, class diagrams, and state machines. Appendix F contains documents code for new functionalities. Appendix G gives document test classes. And lastly, Appendix H gives a description of all meetings and task assigned for the entire project.

# Feasibility Study

The feasibility study chapter explores the idea of a virtual job fair from a practical point of view. First, it considers the limitations of the current system, which is Virtual Job Fair v2.0. Also, it explains the purpose of the Virtual Job Fair, explaining how the features of VJF will improve on current problems. Then, high-level user requirements are described. Moreover, alternatives to certain aspects of VJF are considered and analyzed, with quantitative data used to support the fact that these alternatives were not used.

## 2.1 Description of Current System

The current system was modified in the fall 2013 semester. As it is, it allows students to search for jobs, and get emails with jobs based on their skills. The current system also allows employers to post jobs for students to browse and apply. Moreover, it offers a messaging system, through which employers and students can communicate privately.

The current system also has standard functionality, allowing users to login and logout, to register in the system, to edit their profiles, to change their passwords and to retrieve forgotten passwords.

**Limitations and Constraints**

As implemented, the current system has the following limitations/constraints:

**- Aimed towards FIU Computer Science students**: the scope of the system is geared towards students who are both FIU students and Computer Science majors. However any Computer Science student can greatly benefit from this site.

**- Limited FIU Seniors:** even though the current system does not restrict any user from signing in, an easy way to register should be given to FIU Seniors, given that this project was born from an FIU SCIS Senior course and it is has been strongly suggested by the faculty to include this feature in the system.

- **Limited students registration:** even though the current system does contain a form that allow student to register, the current system lack a way for the students to register using third party accounts like (Google, LinkedIn, FIU Mail, Senior Project web).

- **Limited students’ login:** the current system lacks a way for the students’ user to login using a third party account like (Google, LinkedIn, FIU Mail, Senior Project web).

- **Limited students linking capabilities**: the system lack the ability to provide student a way to link all their third party accounts together into one account. Also the system lack a wizard interface where the student can choose between link conflicts. Example, the student should be able to choose between the name that he or she has on file, and the name that is coming from the third party accounts.

- **Student limitation:** the system lack a way to guide the student when he try to register with an e-mail that is already in the system. In addition; the student should be told that the e-mail is already in the system, and provide the student with some option of what to do next.

- **Inability for student to merge accounts:** the current system lacks a feature that allow student to merge two accounts into one.

**- Students search for job is restricted:** The current system only provides search by skills or search by company and job type.

**- Student job email notification based on skills:** student will only get notified of jobs matching their skills.

**- Student are not allow to save job search queries:** after students query the system for jobs, they are able to save query to future search.

**- Job posting limited to registered employers:** job posting are only available from registered employers only.

- **Limited student notification:** the current system does not allow student to enable/disable their notification. The student’s user is not able to say if he or she is looking for jobs. Since a student’s user may already be employ and want job notification to stop show in his or her e-mail.

- **Limited student job notification:** the current system do not provide the student’s user with job’s notifications base on their skillset.

- **Limited student notification based on saved custom job queries:** the current system does not allow student’s user to save their job search and receive notification base on it.

- **Limited admin setting notification:** the current system does not allow the admin to toggle between enable/disable notifications globally.

- **Limited employer notification:** the current system does not allow employers to receive notification of students matching job posting description.

- **Not API implement:** the current system lacks an API that allow external sources (CareerPath) to push job posting into the VJF system.

- **Limited admin API management:** the current system lacks to allow administrators to control all aspects of the implemented VJF API through an intuitive interface.

- **Limited admin interface:** the current system lacks an administrative interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API endpoint.

**- Limited interview functionality:** the system lacks for video interviews and live chat, limiting the interaction between students and employers to a very basic level.

**- Limited Document Collaboration functionality:** the current system has a very limited collaboration ability. It lacks the ability to have users of the system collaborate on documents in real-time which is a crucial part of an interview process.

**- Inability to share images between students and employers:** the current system lacks an image-sharing feature which allows students and employers to exchange pictures which can enhance the interview experience.

- **No drawing feature:** the current system does not have any type of drawing feature that may allow students and employers to brainstorm, exchange drawings or jot down ideas while interviewing.

**- Reminder system:** the current system lacks a way of reminding students or employers of upcoming interviews.

**- Users are not allowed to delete their accounts:** once registered, students and/or employees are not allowed to remove themselves from the system’s database.

**- Employers are not able to contact students through other means other than by email:** The current implementation makes it very hard to keep both students and employers connected outside of it.

**- Students and employers are not reminded of important deadlines:** The current system does not have any functionality that alerts users of new events.

## 2.2 Description of alternative solutions considered.

Below, alternative implementations for this project are mentioned and discussed:

* **Expand Job Search Sources**

**Alternative 1**

Make use of the Indeed.com web service in order to add more job posting to the job search results.

**Alternative 2**

Make use of the CareerBuilder.com web service in order to add more job posting to the job search results.

**Alternative 3**

Make use of the FIU SCIS CareerPath API in order to add more job posting to the job search results.

**Alternative 4**

Make use of the Monster.com API in order to add more job posting to the job search results.

* **Advanced Search**

**Alternative 1**

Expand the existing search to include company name, job type, skills, and position name on an exact match against the database.

**Alternative 2**

Make the advanced search to mimic Google Advanced Search in order to expand the search and make it more intuitive.

* **Navigation Bar Job Search**

**Alternative 1**

Expand job search from just skills to search by keyword using company name, job type, skills, and position name against the database.

**Alternative 2**

Expand search from just skills to search by keyword using company name, job type, skills, and position name against the database using full text, which provide sort by relevance.

* **Save Query**

**Alternative 1**

Allow students to save query from the advanced search form, using a “Save Query’ button. Then providing students the ability to name the query to their desired name, and allowing students to select/unselect which query to receive email notifications from; as well as the email preference of daily, weekly, or monthly.

* **Register using third party accounts**

**Alternative 1**

Make use of the Google API and fix what is not working from the previous version. (The same with the FIU Mail).

**Alternative 2**

Make use of the LinkedIn API, and fix what is not working from the previous version.

**Alternative 3**

Make use of the Senior Project Web Site API and fix what is not working from the previous version.

* **Login using third party accounts**

**Alternative 1**

Make use of the user ID number provide by the FIU Mail to keep track of the student, so when the student login with the Google login link. The system will look for the ID number of the student, and login the student to his or her account. (The same with the Google login link)

**Alternative 2**

Make use of the user ID number provide by the LinkedIn API to keep track of the student, and fix what is not working from the previous version

**Alternative 3**

Make use of the user ID number provide by the Senior Project Web Site API to keep track of the student, and fix what is not working from the previous version

* **Merging accounts**

**Alternative 1**

Provide the student with a view form to input the username and password of the other account. Then validate the information give, and if the information is verified. Merge the two accounts by comparing the student information that he or she has in the database. And give the user a choice to keep the information that he wants went a merge conflict happens.

* **Registration wizard**

**Alternative 1**

When a student try to register with an e-mail that is already in the database. The system will provide the student with a view form that let the student know. The email is in the database and gives him or her some choices of what to do next. Like getting his or her forgotten password, go to the login page, or go to the register page to register with a different email.

* **Linking accounts**

**Alternative 1**

Make use of the login links using third party accounts to let the student link all his or her third party accounts into one account.

**Alternative 2**

Make use of the profile page to give the student the option of link his or her third party accounts in two the current account the he or she is in. Also let the student know which of the third party accounts the student is already link, and which of the third party accounts the student is not link.

* **API for external sources**

**Alternative 1**

Make use of external sources like “CareerPath” by providing them with an API to VJF. The VJF API will allow this external sources to push job postings into the VJF system. In addition, the administrator will be able to control all aspects of the implemented VJF API through an intuitive interface. Last, provide the administrator with an interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API endpoint.

* **Administrator features and interface improvements**

**Alternative 1**

Make use of the current system administrator page, and provide features that improve the administrator interface. This task will requires to learn Yii framework and the Bootstrap version that is being used by the system. The extent of this task will depend on the time frame that will be given to it.

* **Student Notification**

**Alternative 1**

Make use of the current system notification and improve it. First task being to familiarize with the current system code a see what are the restriction that it has. Then research way to implement a way so that student's user can enable/disable the notification that they received. Also give the student’s user the choice of deciding if they want to keep receiving notification of job match or not. Since student may not want to keep receiving notification of new job if he is already employ or vice versa. In addition, the system should allow students to receive jobs notification based on saved custom job search queries to third party job websites.

* **Admin setting notification**

**Alternative 1**

Make use of the admin page and provide a setting that enable/disable the notification globally. In order to accomplish this task the developer needs to be familiarize with the Yii framework and the Bootstrap version that is in used. In addition, the developer need to be very constable with the notification code already implemented.

## 2.3 Recommendation of Selected Solution

**New System Feature Analysis**

· **Expand Job Search Sources**

The following section contains analysis of the Expanded Job Search Sources alternatives that were proposed on May 2014 and which have been revised at several points throughout the lifetime of the project.

**Alternative 1**

This alternative was selected since Indeed.com web service is free with registration, thus providing real time job search and results.

**Alternative 2**

This alternative was selected since CareerBuilder.com web service is free with registration, thus providing real time job search and results.

**Alternative 3**

After the development of the FIU SCIS CareerPath API, this alternative was selected, thus providing real time job search and results.

**Alternative 4**

This alternative was not selected because Monster.com does not provide a public API, however they do have an RSS feed which provides job search results but its capacities are very limited.

· **Advanced Search**

**Alternative 1**

The expanding of the existing search to include company name, job type, skills, and position name on an exact match against the database alternative was given extensive consideration, but it was decided against since it capacities were very limited and did not provide a broad enough search.

**Alternative 2**

The advanced search to mimic Google Advanced Search in order to expand the search and make it more intuitive alternative was selected since it prove to be the best possible solution because from the user perspective it is intuitive, easy to use, and provide the most thorough search.

· **Navigation Bar Job Search**

**Alternative 1**

The expand job search from just skills to search by keyword using company name, job type, skills, and position name against the database alternative was rejected due to the limited options it provided.

**Alternative 2**

The expand search from just skills to search by keyword using company name, job type, skills, and position name against the database using full text, which provide sort by relevance was selected since it provided a better solution and more accurate alternative.

**Save Query**

**Alternative 1**

This alternative was chosen since it proved to be efficient and sufficient for the purpose of this requirement.

· **Register using third party accounts**

**Alternative 1**

Make use of the Google API to get the student information. In order to create a new account for the student with the information provided by the Google API. (The same with the FIU Mail).

**Alternative 2**

Make use of the LinkedIn API. In order to create a new account for the student with the information provided by the LinkedIn API.

**Alternative 3**

Make use of the Senior Project Web Site API. In order to create a new account for the student with the information provided by the SPWS API.

· **Student login using third party accounts**

**Alternative 1**

Make use of the user ID number provide by the Google API to keep track of the student, so when the student login with the Google login link the system will look for the ID number of the student, and login the student to his or her account. (The same with the FIU Mail login link)

**Alternative 2**

Make use of the user ID number provide by the LinkedIn API to keep track of the student, so when the student login with the LinkedIn login link the system will look for the ID number of the student, and login the student to his or her account.

**Alternative 3**

Make use of the user ID number provide by the Senior Project Web Site API to keep track of the student, so when the student login with the SPWS login link the system will look for the ID number of the student, and login the student to his or her account.

· **Merging accounts**

**Alternative 1**

Provide the student with a view form to input the username and password of the other account. Then validate the information given, and if the information is verified, merge the two accounts by comparing the student information that he or she has in the database. And give the student user the choice to keep the information that he/she wants when a merge conflict happens.

· **Registration wizard**

**Alternative 1**

When a student try to register with an e-mail that is already in the database, the system will provide the student with a view form that let the student know the email is in the database and gives him or her some choices of what to do next. For instance, getting his or her forgotten password, go to the login page, or go to the register page to register with a different email.

· **Linking accounts**

**Alternative 1**

Make use of the login links using third party accounts to let the student link all his or her third party accounts into one account.

**Alternative 2**

Make use of the profile page to give the student the option of link his or her third party accounts into the current account the he or she is in. Also, let the student know which of the third party accounts the student have already linked, and which of the third party accounts the student is have not linked.

· **API for external sources**

**Alternative 1**

Make use of external sources like “CareerPath” by providing them with an API to VJF. The VJF API will allow this external sources to push job postings into the VJF system. In addition, the administrator will be able to control all aspects of the implemented VJF API through an intuitive interface. Last, provide the administrator with an interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API endpoint.

· **Administrator features and interface improvements**

**Alternative 1**

Make use of the current system administrator page, and provide features that improve the administrator interface. This task will requires to learn Yii framework and the Bootstrap version that is being used by the system. The extent of this task will depend on the time frame that will be given to it.

· **Student Notification**

**Alternative 1**

Make use of the current system notification and improve it. First task being to familiarize with the current system code a see what are the restriction that it has. Then research ways to implement a way so that student user can enable/disable the notification that they received. Also give the student user the choice of deciding if they want to keep receiving notification of job match or not. Since student may not want to keep receiving notification of new job if he is already employ or vice versa. In addition, the system should allow students to receive jobs notification based on saved custom job search queries to third party job websites.

· **Admin setting notification**

**Alternative 1**

Make use of the admin page and provide a setting that enable/disable the notification globally. In order to accomplish this task the developer needs to be familiarize with the Yii framework and the Bootstrap version that is in used. In addition, the developer need to be very constable with the notification code already implemented.

# Project Plan

The project plan chapter introduces VJF 3.0 from a project management perspective. Firstly, the project organization is described, with the roles for each member listed. After that, milestones, tasks, and deliverables will be listed. Finally, a cost estimate for whole project is presented in terms of a feasibility matrix.

## 3.1 Project Organization

For this project, each of the four members will be in charge of adding at least two pieces of functionality to the system.

Ana L. Hernandez will be responsible for creating a Google like advanced search which will allow student to search for jobs within our database (CareerPath jobs), and/or using external sources (Indeed & CareerBuilder). Also, she will expand the navigation bar search to search by keywords. In addition, she will be responsible for the integration for the Indeed.com and CareerBuilder.com API for the expansion of the job search for both the advanced search and the navigation bar search. Moreover, she will implement a save query feature which will allow students to save any query they just searched for, and then modify its preference to receive email notification with the saved queries.

Manuel Bouza will be responsible for the login and registration pages. He will find what it is wrong with the third party accounts links, and he will find a solution that will prevent this from happening again. In addition he will provide the student’s user with a wizard that will guide the student through the registration page when the student try to register with an email that is already in the database. Also Manuel will be responsible for finding a way so that the student’s user can link all their third party accounts into one account. Manuel will change the profile page so it lets the student’s user know to which third party account the student’s user is link, and to which the student’s user is not link. Finally Manuel will be responsible for providing in the toolbar of the student’s user a button to merge two accounts. Furthermore; Manuel has to provide the student’s user with a wizard that will allow student to choose which information the user want to keep when a merge conflict happens, or when linking accounts.

Enio Pena Navarro will be responsible for adding this next features. First, Enio will designing an API that allows external sources (SCIS CareerPath) to push job postings into the VJF system. Second, he will provide the administrators a way to control all aspects of the implemented VJF API through an intuitive interface, and provide an administrative interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API endpoint. Third, Enio will allow the administrators to maintain the system’s job skills database, and provide the administrators with an interface to manage job posting and users. In general, improve the administrator’s interface.

Tomas Acosta will be responsible of adding two main features to the system. First, Tomas will provide a user Interface settings in user’s profiles that enables the users to activate/deactivate notifications. Second, Tomas is responsible of Job Matching Notification Engine: This engine is in charge of sending students and employers notifications of job listings and students matches respectively.

### 3.1.1 Project Personnel

Below is a table which represents the roles of each of the members:

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Primary Task** | **General Task** |
| Ana L. Hernandez | Developer  Document Editor | Additional shared tasks |
| Tomas Acosta | Developer  Document Editor | Additional shared tasks |
| Manuel Bouza | Developer  Document Editor | Additional shared tasks |
| Enio Pena Navarro | Developer  Document Editor | Additional shared tasks |

### 3.1.2 Hardware and Software Resources

**Hardware**

In order to start developing the project, our team will need computers with at least the following specifications:

**- Processing Power:** Pentium IV 2.0 GHz processor or better

**- RAM Memory:** 1GB 133MHz SDRAM

**- Available space on hard drive:** 5GB

**Other Devices**

- **Input devices**

a) Standard wired/wireless K120 keyboard

b) Standard wired/wireless trackball/optical mouse

- **Output devices**

a) Standard VGA/DVI/HDMI monitor display

**Software**

**1) Google Chrome 30.8:** last, most updated version of the Chrome browser that will be used to test Virtual Job Fair.

**2) StarUML:** UML software platform that will be used to create diagrams for the document

**3) Yii Framework 1.1.14:** last, most updated version of Yii Framework, an MVC-based, PHP framework used for development. It will be used to develop the front-end and back-end of Virtual Job Fair.

**4) phpMyAdmin:** a DBMS (integrated into Yii) that will be used to manually manipulate the database when necessary.

**5) PhpStorm (IDE):** software platform for front-end design that will be used to create backbone of the user interface.

**6)** **VMWare:** virtual machine software that will be used for the deployment of the software.

## 3.2 Identification of Tasks, Milestones and Deliverables

Below is a list of all different tasks, milestones, and deliverables for the project:

|  |  |  |
| --- | --- | --- |
| Week | Date | Task, Milestones, Deliverables |
| 2 | 5/20/14 | Feasibility Study & Project Plan, &  Requirement Document, & Design Document  Design Milestone |
| 3 | 5/27/14 | Implementation: Unit Testing  Testing Milestone |
| 4 | 6/3/14 | Debugging & Code Refinement |
| 5 | 6/10/14 | Debugging & Code Refinement |
| 6 | 6/17/14 | Debugging & Code Refinement |
| 7 | 6/24/14 | Implementation: System Testing  Testing Milestone |
| 8 | 7/1/14 | Debugging & Code Refinement |
| 9 | 7/8/14 | Debugging & Code Refinement |
| 10 | 7/15/14 | Debugging & Code Refinement |
| 11 | 7/22/14 | Debugging & Code Refinement |
| 11 | 7/25/14 | Final Presentation, & Final Deliverable, &  Project Deployment  Documentation Milestone |

## 3.3 Cost of the Project

The following feasibility matrix represents an estimate of the items and labor required for the project. These estimated costs are accurate as of Friday, July 25th, 2014.

|  |  |
| --- | --- |
| **Item** | **Item Cost** |
| Yii Framework | $0.00 |
| Twitter Bootstrap | $0.00 |
| Easy RTC | $0.00 |
| Hardware | $1200.00 |
| Development | $0.00 |
| Testing | $0.00 |
| Contingencies | $240.00 (~20% of total cost) |
|  | **Total:**$1200.00 |

# 

# 4. System Requirements

The proposed system is Virtual Job Fair v3.0, which would expand upon the already-implemented Virtual Job Fair v2.0. The main idea behind Virtual Job Fair v3.0 is to add new dynamic functionality to the job search, login, notification, and administrator interface. This chapter will introduce the functional and non-functional requirements of the system and the requirements analysis phase of the system.

**4.1 Functional and Nonfunctional Requirements**

**Current System’s Requirements**

The system shall…

**Allow students and employers to register**

- **Usability**: The register form is simple and easy to follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 2 seconds.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students and employers to view respective profiles**

- **Usability**: Data displayed in profiles is easy to follow. Students are only able to see their own profile and the employer ones. Employers can see all student profiles.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students and employers to edit their basic profile information**

- **Usability**: The edit form is simple and easy to follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students and employers to take part in a video interview**

- **Usability**: Starting a video interview is simple and understandable.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 seconds when connecting.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to upload a resume and video resume**

- **Usability**: The upload form is simple and easy to follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1-5 seconds, depending on the file size.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students and employers to upload an image for their profile**

- **Usability**: The upload form is simple and easy to follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to associate skills to their profile**

- **Usability**: The ability to add skills to a profile is simple and understandable. It can be done by using LinkedIn connect or adding them manually.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within one 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to add and delete education information**

- **Usability**: The corresponding form is easy to complete and follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to add and delete experience information**

- **Usability**: The corresponding form is easy to complete and follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to apply to open job postings and provide a cover letter**

- **Usability**: Students are presented with a user-friendly interface that is easy to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to reply to an employer’s message**

- **Usability**: Students are presented with a clear and simple interface to send messages.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow students to search for jobs based on skills**

- **Usability**: The search form is easy to follow and complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to post jobs**

- **Usability**: The post job form is easy to understand and complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to close a job posting**

- **Usability**: The closing of a post is easy to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to associate skills to a job posting**

- **Usability**: the addition of skills to a post is simple to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second when adding each skill.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to search for students based on skills**

- **Usability**: The search form is simple to submit.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to view student profiles**

- **Usability**: The view of a student profile is easy to understand.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within one 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to send messages to students**

- **Usability**: Employers are presented with a clear and simple interface to send messages.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow employers to give students a “virtual handshake” to show interest in the student**

- **Usability**: The virtual handshake form is easy complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow an administrator to disable an account (security)**

- **Usability**: Disabling a user is simple to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow an administrator to close a job posting**

- **Usability**: Closing a job post is simple to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow an administrator to validate an employer registration (security)**

- **Usability**: The validation of an employer is done by one click and is simple to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Require a username and password to log into the system (security)**

- **Usability**: This is required for a user to log in. Form is simple and easy to follow.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Encrypt the user password before storing into the database (security)**

- **Usability**: Storing user password in a secure way without user intervention.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Require login before viewing user profiles (security)**

- **Usability**: Security measure for system. Interface is simple to complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easy to maintain and make appropriate changes.

**Allow users to delete a document.**

- **Usability:** The document deletion interface should be intuitive, easy to understand and navigate.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be supported on Google Chrome version 29+ the system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow users to import a document.**

- **Usability:** The import document interface should be intuitive, easy to understand and navigate.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 3 seconds.

- **Supportability:** The system should be supported on Google Chrome version 29+ the system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow users to open a document.**

- **Usability:** The document open interface should be intuitive, easy to understand and navigate.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 3 seconds.

- **Supportability:** The system should be supported on Google Chrome version 29+ the system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow employers to send SMS to students**

- **Usability**: Sending an SMS should not take more than 15 seconds for a novice user.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The SMS should be sent in under 2 seconds after user presses send

- **Supportability**: Sending SMS page should be supported by IE, Firefox, Chrome and Safari.

**Allow users to receive automatic email and SMS reminders**

- **Usability**: Not applicable

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: Users should receive a reminder 30 minutes before scheduled interview

- **Supportability**: Not applicable

**Allow users to confirm their phone numbers**

- **Usability**: Confirming a phone number should take less than 30 seconds for inexperienced users

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: Sending authentication code and validating it should take less than 2 seconds respectively.

- **Supportability**: Confirm phone number page should be supported by IE, Firefox, Chrome and Safari

**Allow students to build their profile (basic information, skills, education, and experience) from LinkedIn.**

- **Usability**: The connection with LinkedIn should be easy to follow. Users will enter their LinkedIn credentials and get appropriate data that the user allowed.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow FIU Computer Science Seniors to login using their FIU SCIS credentials**

- **Usability:** The system should provide an easy and integrated login process for FIU SCIS Seniors using the school UNIX account.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow FIU student to login using their FIU credentials**

- **Usability:** The system should provide an easy and integrated login process for FIU students using the school UNIX account.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow student to login using their Google credentials**

- **Usability:** The system should provide an easy and integrated login process for Google accounts.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow student to login using their LinkedIn credentials**

- **Usability:** The system should provide an easy and integrated login process for LinkedIn accounts.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**New System’s Requirements**

**Allow students to search for jobs based on skills, company name, job type, and position on navigation bar**

- **Usability**: The search form is easy to follow and complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 2 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to search for jobs based on skills, company name, job type, position, and location on the advanced search**

- **Usability**: The search form is easy to follow and complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to search for jobs based on skills, company name, job type, position, and location on the advanced search**

- **Usability**: The search form is easy to follow and complete.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow job search results from FIU CareerPath**

- **Usability**: The search button should be press.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow job search results from Indeed.com**

- **Usability**: The search button should be press.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow job search results from CareerBuilder.com**

- **Usability**: The search button should be press.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 3 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to receive email notification with expanded job search results**

- **Usability**: The profile preference are easy to setup.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should send email within 1, 7 or 30 days.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Prompt student to enter name for query to be save**

- **Usability**: The student should be able to enter name with ease.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow students to save queries to their profiles**

- **Usability**: The system should provide a save button.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to active / deactivate saved queries**

- **Usability**: The system should provide a checkbox.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to delete saved queries**

- **Usability**: The system should provide a delete icon next to query.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to link their third party accounts into one**

- **Usability:** The system should provide an easy and integrated linking process for third parties accounts.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow students to choose between their current information and the information coming in from their third party accounts when linking accounts.**

- **Usability:** The system should provide an easy and integrated interface for linking accounts.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Guide the students to recover his or her password; when trying to register with an e-mail that is already in the system.**

- **Usability:** The system should provide an easy and integrated interface for recovering password.

-  **Reliability:** The system should perform correctly 99% of the time.

- **Performance:** The system should respond in less than 1 second.

- **Supportability:** The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allows external sources (SCIS CareerPath) to push job postings into the VJF system.**

- **Usability**: The form should be easy to use, Administrators can specify from what day to what day, and submit.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow administrators to control all aspects of the implemented VJF API through an intuitive interface.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Provide an administrative interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API end-point.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow administrators to maintain the system’s job skills database.**

- **Usability**: The form should be easy to use, Administrators can specify from what day to what day, and submit.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes.

**Allow an interface for administrators to manage users.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow an interface for administrators to manage job postings.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow the administrator to manage the notification system.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow the administrator to manage the notification system.**

- **Usability**: The interface should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to enable/disable if he/she is looking for jobs**

- **Usability**: The button to enable/disable should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow admin users to enable/disable notifications globally**

- **Usability**: The button to enable/disable should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to receive jobs notification based on saved custom job search queries to third party job websites**

- **Usability**: The emails notifications should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow students to receive jobs notification based on their skillset**

- **Usability**: The emails notifications should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**Allow employers to receive notification of students matching job posting description**

- **Usability**: The emails notifications should be easy to use.

- **Reliability**: The system should perform correctly 99% of the time.

- **Performance**: The system should respond within 1 second.

- **Supportability**: The system should be easily maintainable, allowing for improvements, corrections and adaptation in environment, requirement and/or specification changes

**4.2 Requirements Analysis**

Analysis models – contains the complete functional specification and is mainly for the designers and programmers. This section describes the diagrams in the Appendices B - E and validates the models against the use cases.

### 4.2.1 Use case model

**Use Case Diagram Analysis**

The use case diagram in Appendix D describes the overall view and functionality that we will add to the system. The proposed functionality deals with the two type of users displayed in the diagram. Students, and Administrators are the targets of these functional requirements.

### 4.2.2 Static model

**Class Diagram Analysis**

The Class Diagram in Appendix E depicts the additional classes that will be added to the original system. It portrays the proposed changes to the scheme, where we build on the implementation of the previous team. The new proposed classes are colored green in order to emphasize the difference between the new and old implementations. In addition, the Diagram follows UML notation in order to facilitate the developers as well as adhere to industry standards.

### 4.2.3 Dynamic model

**Sequence Diagram Analysis**

In our sequence diagrams, actors can be either students, or admins. In all of our use cases, the actors communicate with an object from the view component. Also, some sequence diagrams require communication with our database. Please refer to Appendix D for diagrams.

# 5. System Design

This chapter gives an insight into the architectural pattern(s) used to build the system. Virtual Job Fair was subdivided into subsystems, each one with a specific functionality that adds richness to the interview process. In this chapter, an overview of the system design is introduced. Then, the decomposition of the system into subsystems is explained. Moreover, hardware and software mapping and persistent data management aspects of the project are discussed. Finally, the security and privacy issues of the system are introduced.

The new system shall…

Allow users to create a new document.

Allow users to import a document.

Allows students to receive SMS notifications

Allow student to confirm their phone numbers through SMS validation

**Allow students to search for jobs based on skills, company name, job type, and position on navigation bar**

**Allow students to search for jobs based on skills, company name, job type, position, and location on the advanced search**

**Allow students to search for jobs based on skills, company name, job type, position, and location on the advanced search**

**Allow job search results from FIU CareerPath**

**Allow job search results from Indeed.com**

**Allow job search results from CareerBuilder.com**

**Allow students to receive email notification with expanded job search results**

**Prompt student to enter name for query to be save**

**Allow students to save queries to their profiles**

**Allow students to active / deactivate saved queries**

**Allow students to delete saved queries**

**Allow FIU Computer Science Seniors to login using their FIU SCIS credentials**

**Allow FIU student to login using their FIU credentials**

**Allow student to login using their Google credentials**

**Allow student to login using their LinkedIn credentials**

**Allow students to link their third party accounts into one**

**Allow students to choose between their current information and the information coming in from their third party accounts when linking accounts.**

**Guide the students to recover his or her password; when trying to register with a e-mail that is already in the system.**

**Allow students to build their profile (basic information, skills, education, experience) from LinkedIn.**

**Allows external sources (SCIS CareerPath) to push job postings into the VJF system.**

**Allow administrators to control all aspects of the implemented VJF API through an intuitive interface.**

**Provide an administrative interface to bulk import jobs from the FIU SCIS CareerPath system, using a provided CareerPath API end-point.**

**Allow administrators to maintain the system’s job skills database.**

**Allow an interface for administrators to manage users.**

**Allow an interface for administrators to manage job postings.**

**Allow the administrator to manage the notification system.**

**Allow students to enable/disable if he/she is looking for jobs**

**Allow admin users to enable/disable notifications globally**

**Allow students to receive jobs notification based on saved custom job search queries to third party job websites**

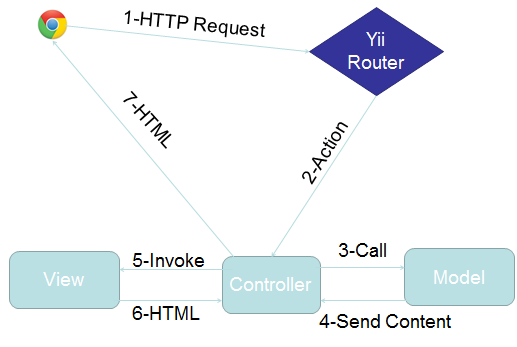
**Allow students to receive jobs notification based on their skillsets**

**Allow employers to receive notification of students matching job posting description**

## 5.1 Overview

The Virtual Job Fair was implemented using the Model-View-Controller architecture. Our team is adding functionality to the system, meaning that the Model-View-Controller architecture will be maintained, with models, views and controllers added to account for the new functionality. Below is their description of the architecture of the system:

 “The Architectural pattern used to build the system was the popular Model-View-Controller. This is the architecture implemented by the Yii Framework which was used by the team of developers. The following diagram depicts the architecture:”

  
**Figure 5.1.1 MVC using Yii Framework**

## 5.2 Subsystem Decomposition

Below is the subsystems decomposition found in the VJF system.

**E-mail Message Subsystem**

The E-mail messaging subsystem allows users to keep in contact and communicate with each other right on the system. It is very similar to a traditional inbox, only that it is internal to the system, similar to LinkedIn’s messaging. The messaging subsystem uses database tables to store and retrieve messages sent between users. It allow employers to message students and in doing so open up a line of communication with them; i.e., a student is able to message an employer only after the employer has initiated the communication with that particular student. Storing and retrieving messages efficiently is imperative to a successful messaging system. The messaging subsystem uses AJAX to rapidly access and store data; allowing users to interact faster with the system without having to wait for server calls.

The use cases related to this subsystem are:

·         VJF-0022 Reply to Message

·         VJF-0023 Send Message

**Notification Subsystem**

The subsystem is what allows users to stay up to date with the latest system interactions. The notification subsystem alerts users of any interaction by other users that might implicate them, such as a video interview been schedule for a user, or a new job post that matches a user’s skills.

The notification subsystem relies on the database structure to be able to efficiently keep the users inform. Due to the relationships between tables in the database the notification subsystem can easily detect what notification belongs to what user. Also it is important to sort each notification by category and level of importance. The notification subsystem achieves this by storing different types of notifications in the database and mapping them to their respective categories.

**Automated Notification**

This subsystem make use of the “job matching notification engine”. Who is in charge of sending notification emails with the job listings to the students based on jobs matching their skills or based on customized saved queries preference. Students can choose the email frequency of daily, weekly, or monthly. They job listing receive in the email will contain jobs from outside sources like Indeed.com, and CareerBuilder.com, as well as the job posting from the database which will contain jobs from FIU CareerBuilder.

The uses cases related to this subsystem are:

·         VJF-0021 Accept Interview

·         VJF-001 Registration

·         VJF-0034 Read notification

VJF-059 Set Notifications On

VJF-064 Admin Enable Notification

**Student Profile subsystem**

Being able to create a good profile fast and efficiently is very important. The profile creation subsystem takes care of this by allowing students to import profile information from third party websites such as LinkedIn. This ensures integrity of the data in students’ profiles, and makes it very easy for students to create their profiles.

This is possible by using API calls to LinkedIn and retrieving the data from LinkedIn users. As it is to expect, the user must grant permission to do this by providing his/her login credentials which are handled by the LinkedIn API.

**Merge Account**

This subsystem provides students with a view form to input the username and password of the other account. Then validate the information given, and if the information is verified, merge the two accounts by comparing the student information that he or she has in the database. And give the student user the choice to keep the information that he/she wants when a merge conflict happens.

**Linking Account**

This subsystem provides students with a way to link all third party account into one account. Also, it lets the student know if they are linked to a third party account or not, and which one. When linking, students can choose which information to keep if there is a conflict among information.

The use cases related to this subsystem are:

·         VJF-001 Registration

·         VJF-0019 Integrate LinkedIn

·         VJF-003 Edit Basic Info

·         VJF-004 Verify Email

·         VJF-008 Edit Picture

·         VJF-009 Upload Resume

·         VJF-0011 Add Education

·         VJF-0012 Delete Education

·         VJF-0013 Add Experience

·         VJF-0014 Delete Experience

VJF-063 Linking Account Google

VJF-067 Merge Account

**Student job match subsystem**

Making the right connection is what this web application is all about. Therefore, an efficient algorithm to match students to job openings is very important. The student job match subsystem takes care of matching students with the required skills to job post, making the job of the recruiters easier, as it shrinks the search to only the most qualified individuals for the job.

The student job match subsystem relies on the relationships between the data in the job table. By matching job skills to students skills listed on their profile the algorithm can effectively narrow down the search to only those individuals who possess those skills.

The uses cases related to this subsystem are:

·         VJF-0016 Add Skill

·         VJF-0017 Delete skill

·         VJF-0018 Change skills Order

**SMS message subsystem**

This subsystem gives the ability to send messages to students as well as allow the users to receive reminders about upcoming interviews directly in their SMS enabled phones. The SMS messaging subsystem is designed to allow this kind of functionality which can keep the users connected and active with the system.

All of this is possible by using the Twilio cloud communications service. Twilio is a company that provides services such as SMS sending and receiving, speech and text recognition, conference calling etc.  The most compelling feature is that an extensive and well documented API is provided to developers, this will be necessary in order to create a system that allows for sending text messages to users, as well as validating their identities.

* VJF-045 Send SMS to student
* VJF-046 Send interview reminder
* VJF-047 Confirm phone number

**Search Subsystem**

The search subsystem is compose of the key features of the job search, which allows students to performed advanced search, navigation bar search, as well as save queries to later receive email notifications with job results. This subsystem make the use of job search possible, giving the student the chance to find their ideal job.

The uses cases related to this subsystem are:

* VJF-057 Advance search
* VJF-060 Navigation bar Search
* VJF-061 Reset Advance Search Inputs
* VJF-062 Set Job Search Status On
* VJF-066 Save Query

**API Subsystem**

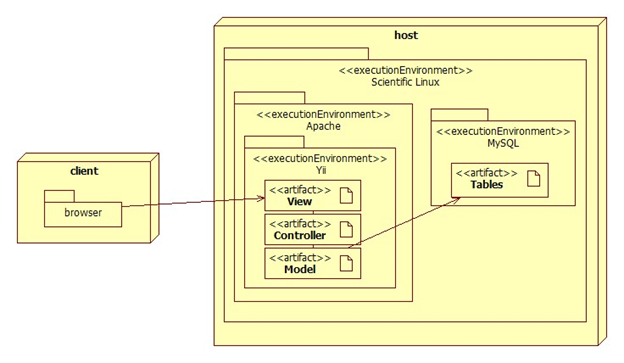
This subsystem allows admin to manage API keys from which other services can push jobs to out system. Also, it enables the CareerPath job import, synchronization, and lets admin turn on the ability of others to push into our system.

The uses cases related to this subsystem are:

* VJF-058 Disable API Querying
* VJF-065 Import Jobs

## 5.3 Hardware and Software Mapping

Below is the deployment diagram which indicates the mapping of hardware and software. Virtual Job Fair v3.0, is using the Model-View-Controller architectural pattern. Moreover, our team requested a virtual machine running on Scientific Linux with an Apache server and the Yii framework on the FIU SCIS network, identical to the set up used by the previous team. For that reason, the same deployment diagram as before will be used.

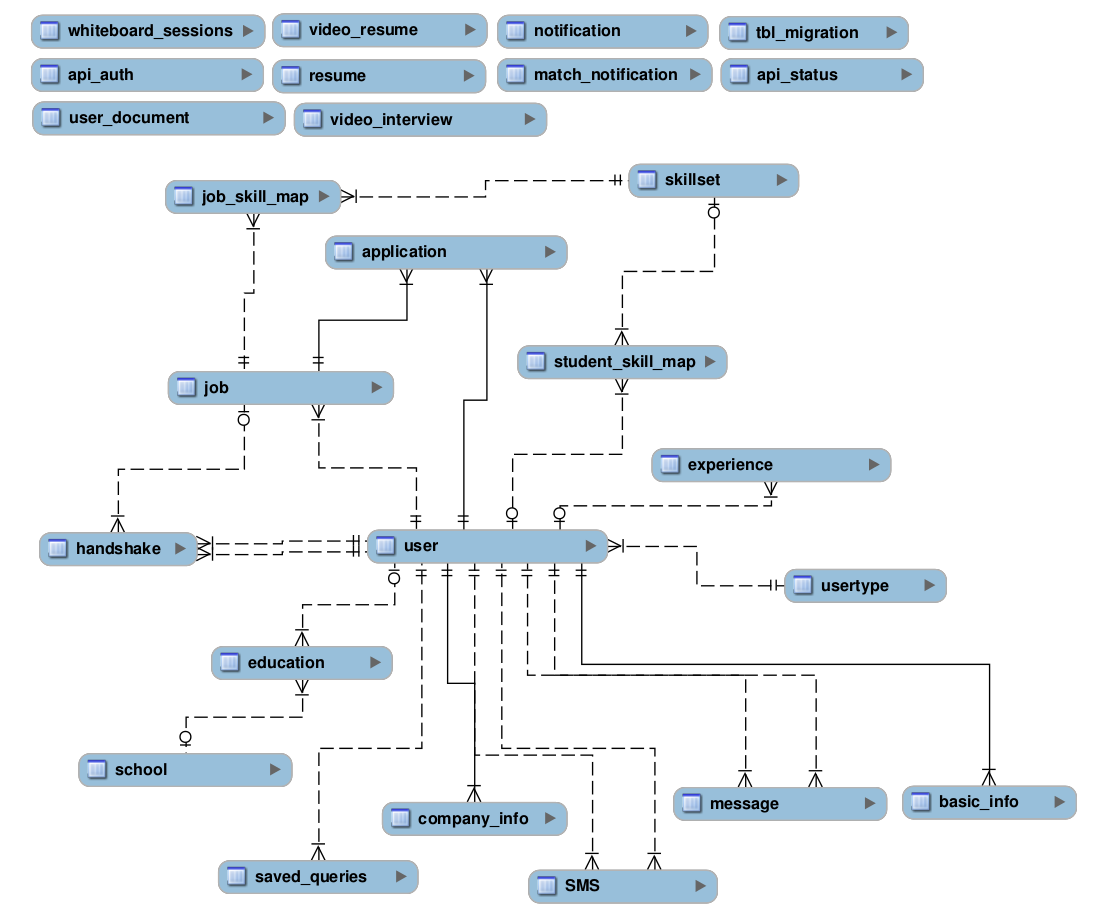


**Figure 5.3.1 Hardware and Software**

The deployment diagram shown above represents the hardware and software mapping in the Virtual Job Fair system.  The main components of the system are the Apache and MySQL environment hosted on a Linux operating system.  The Yii framework environment is using apache to execute, and contains our various artifacts used in development (Model, View, and Controller).  The models are mapped to tables in the MySQL environment set up on the same machine.  The browser on the client’s machine communicates with the server using HTTP.

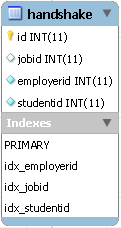
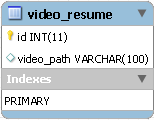
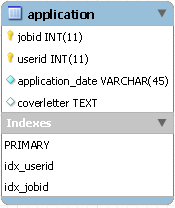
## 5.4 Persistent Data Management

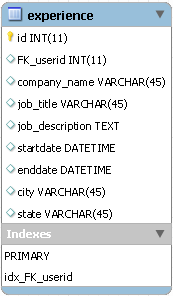
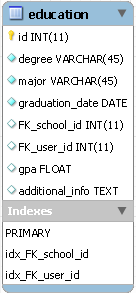
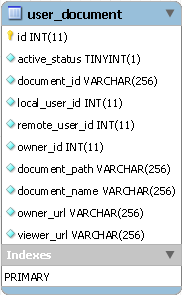
Below is the simplified ER diagram. Each specific table can be seen in the Data Dictionary section.

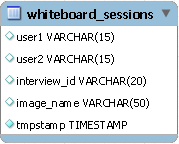
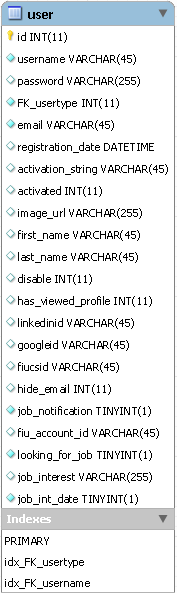
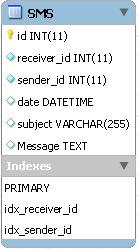


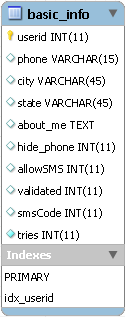
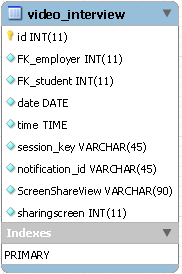
**Figure 5.4-1 ER Diagram**

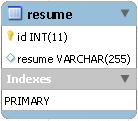
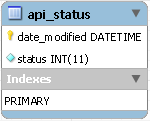
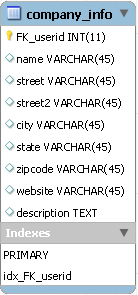
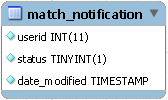
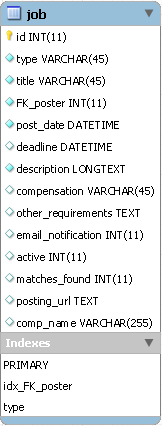
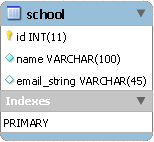
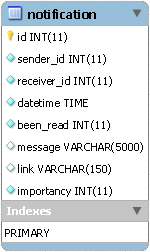
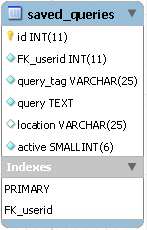
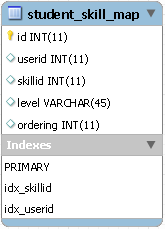
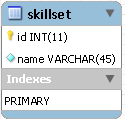
**Data dictionary:**











## 

## 5.5 Security and Privacy

The previous team defined security and privacy features for login, registration, and access to the Yii framework. Moreover, security and privacy features were added for each of the functionality that was implemented. Both of these are described below:

**Security Features**

·         User password will be hashed in the database.

Upon registration into the system, passwords entered will be hashed right away and will not be saved anywhere on the system.  Upon login, the password entered again will be hashed and the hashed data will be used to query the database.

**·         Administrator will be able to disable users and delete jobs**

An administrative console will be provided to a person to allow basic duties that may be needed in the future.  Due to abuse of the system, it may be necessary to delete jobs or disable users.

· **Yii access control rules**

The Yii framework provides access control with respect to any controller being used.  This access control will reject a subset of users (not logged, students, employers, etc…) from performing certain actions.  For example, users that are not logged in will not have access to profile pages.

**·         Cross-site Scripting Prevention**

The Yii framework takes measures against common web exploitations such as cross-site scripting or MySQL injection.  Using Yii, we can be rest assured that such things should not occur.

**·         Secure registration process**

The registration process is not as simple as most sites, especially for employers.  Administrators will have to verify employers after they register to ensure they are actual employers to ensure the integrity of the system.  Only then will they be able to post jobs and interact with students.

**Privacy**

Students and Employers are distinct user types and therefore have distinct permissions.  It may be necessary to allow employers to do actions that students cannot.  For example, students should not be able to post a job or schedule an interview, which clearly employers should be able to.  Likewise, students will only be able to view an employer’s profile and will not be able to view other students’ profiles, since it may contain information which should not be shared, such as phone number or email.

Below are these security and privacy features for the new functionality of the system:

# Detailed Design

In this chapter a deeper and more complete description of the system will be discussed. The classes use and the interactions between them will be described. The design patterns applied to our system design will be covered more in detail during this part of the document. The refined sequence diagrams will be outlined. The main control class interfaces will described by its functions and constraints.

**6.1 Overview**

The main subsystems in our system are the automated notification subsystem, merge and link subsystem, search, and API subsystems. Below is a brief description of each of them.

The Automated subsystem make use of the “job matching notification engine”. Who is in charge of sending notification emails with the job listings to the students based on jobs matching their skills or based on customized saved queries preference.

Merge Account is part of the profile subsystem, it provides students with a view form to input the username and password of the other account, and merge them. The Linking Account subsystem is also part of the profile subsystem, it provides students with a way to link all third party account into one account. Also, it lets the student know if they are linked to a third party account or not, and which one. When linking, students can choose which information to keep if there is a conflict among information. The student job match subsystem relies on the relationships between the data in the job table. By matching job skills to students skills listed on their profile the algorithm can effectively narrow

The search subsystem is compose of the key features of the job search, which allows students to performed advanced search, navigation bar search, as well as save queries to later receive email notifications with job results. This subsystem make the use of job search possible, giving the student the chance to find their ideal job.

The API subsystem allows admin to manage API keys from which other services can push jobs to our system. Also, it enables the CareerPath job import, synchronization, and lets admin turn on the ability of others to push into our system.

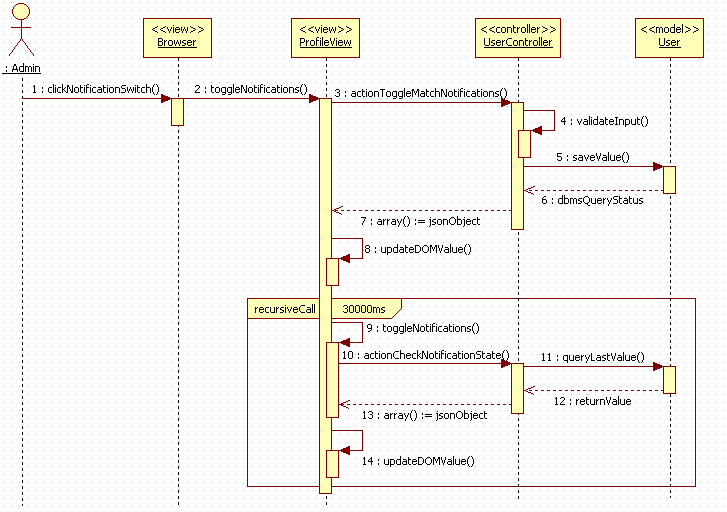
## 6.2 Static model

All components of the system are shown through the class diagram in Appendix E. The design pattern that was used was the structural design pattern because it helps guarantee that when one of the parts changes, the entire structure does not need to change. The module pattern was used since it can be used to further emulate the concept of classes so that we are able to include both public and private methods and variables inside a single object, thus shielding particular parts from the global scope. This was helpful due to its ability to make the public parts of our code able to touch the private parts, however the outside world is unable to see the class's private parts

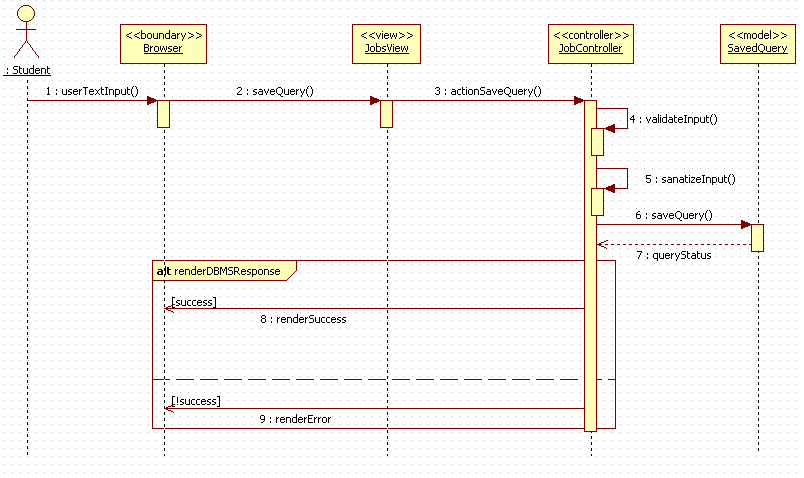
## 6.3   Dynamic model

In the following section, are the main sequence diagrams for the above subsystem. The rest of the sequence diagram can be found in Appendix D.

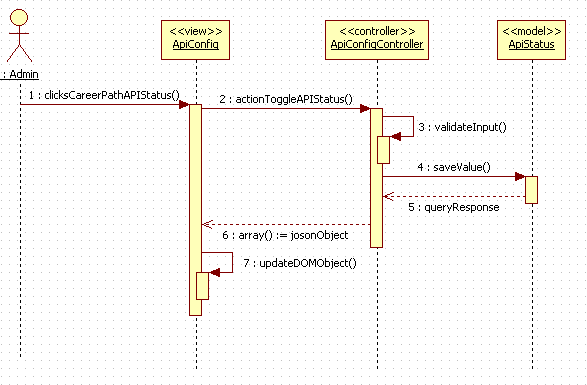
**Automated Notification - Admin Enable Notification**



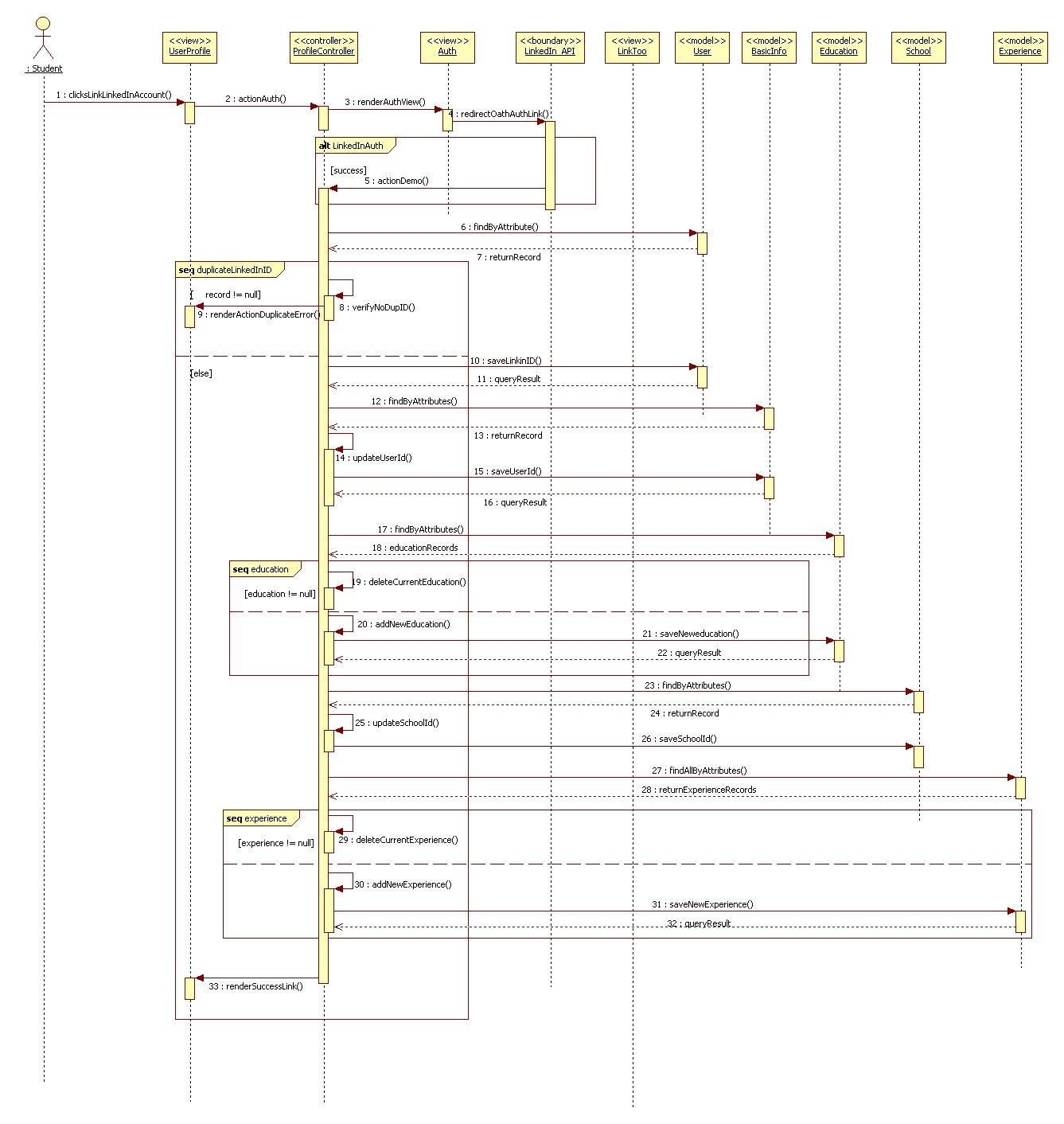
**Search Subsystem - Save Query**



**API subsystem - Import Jobs**



**Link Account Subsystem - Linking Account**



## 6.4 Code Specification

In this section we present a brief description of some of the methods that realize the main control object for each component that make up the VJF system. The code for each can be found in Appendix F.

**Automated Notification Subsystem**

The following are some of the main methods that realize the Automated Notification service itself.

**buildTable()**

This method can be found in the JobMatchedCommand class. This method builds all the tables for the email notifications for the student, and employer. This method creates the job match search results for the student notification based on skills, or customized saved queries. Also, it creates the table for the employer notification whenever there are students matching any job posting.

**customJobSearch()**

This method can be found in the JobMatch class. This method returns an array containing job postings from Indeed.com, CareerBuilder.com, and CareerPath matching the customized saved queries each student has. The customJobSearch() method uses the Job model to obtain the jobs from the database. It also calls the indeed and careerBuilder methods.

**Merge Subsystem**

The following is the the main method that realize the Merge service itself.

**actionMergeAccounts()**

This method can be found in the UserController class. This method renders a view to input username and password for the account to be merged. Also validate the username and password. This method uses the User, and Education, Experience models, also the LinkTooForm and LoginForm. actionMergeAccounts calls actionLink to render a view to the user with any merge conflicts.

**Link Subsystem**

The following are the the main methods that realize the Linking service itself.

**actionFiuAuth()**

This method can be found in the ProfileController class. This method prompts the user with a view to input username and password which is provided by the Google API; after validation, this methods calls actionLinkToo which will render a view to the user with any link and merge conflicts.

**actionGoogleAuth()**

This method can be found in the ProfileController class. This method prompts the user with a view to input username and password which is provided by the Google API; after validation, this methods calls actionLinkToo which will render a view to the user with any link and merge conflicts.

**actionFiuCsSeniorAuth()**

This method can be found in the ProfileController class. This method prompts the user with a view to input username and password which is provided by the Senior Project website API; after validation, this methods calls actionLinkToo which will render a view to the user with any link and merge conflicts.

**actionDemo()**

This method can be found in the ProfileController class. This method prompts the user with a view to input username and password which is provided by the LinkedIn API; after validation, this methods calls actionLinkToo which will render a view to the user with any link and merge conflicts.

**Search Subsystem**

The following are the the main methods that realize the Advance Search and Save Query service.

**actionHome()**

This method can be found in the JobController class. This method takes care of getting the jobs from the database (pushed CareerPath jobs). It validates the input search query when users perform an advanced search. It calls helper methods careerBuilder, indeed, and xmlToArray in order to obtain jobs matching the search query. The return are arrays contains the results for each the job search source (database, Indeed, CareerPath).

**actionSaveQuery()**

This method can be found in the JobController class. This method validates the search query to be saved to ensure proper the query is valid. It then uses the User model to save the customized query into the saved\_query table under the user id, with the user provided name.

**API Subsystem**

The following are the the main methods that realize the Sync/Post of jobs to database from CareerPath service.

**actionPost()**

This method can be found in the APIController class. It takes care of creating jobs (pushed through the exposed API end-point) in the database. It verifies the API key provided to make sure it is a valid key before proceeding to create the job. The return value notifies the calling service whether the job creation succeeded.

**careerPathSync()**

This method can be found in the ApiConfigController class. This method handles the importing of jobs from the CareerPath API. It takes specified date range and creates a request for a list of jobs from the CareerPath system; it then maps the received json object to the current database job table columns and creates a user to which the job belong. All fields are validated before being persisted to the database.

# 7 System Validation

The system validation chapter introduces testing to make sure that the project works as was intended. For this specific phase of the project, we performed system testing and subsystem testing, which involves testing using the GUI and then going deep into code to go ahead and test it. We tested the system using Integration Testing to ensure that the back-end of the application was working as expected with the front-end. For this, used the Bing Bang approach due the interdependence among our modules. Also, we did Manually Exploratory Testing by impersonating the role of the student and tested all of the features of the application to ensure proper behavior. In the following sections we have include the drivers and script for both system and subsystem for VJF v2.0.

## 7.1 Subsystem Tests

Due to problems implementing the original real-time whiteboard, and due to having been forced to select an alternate, simpler approach to the whiteboard with tight time constraints, the whiteboard subsystem was not tested. It was, however, tested with system testing.

|  |  |
| --- | --- |
| 1. Identifier | VJF\_SubSystem\_SMSconfirm01 |
| Purpose | This test verifies that the system stores a digit between 1000 and 999 as code |
| Setup | Set user as hello5  Create a fixture with basic info information |
| Input |  |
| Expected Output | Basic->info->code = 0; |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSautocomplete01 |
| Purpose | This test verifies that the system correctly autocompletes the given input |
| Setup | Create a user with username “test” |
| Input | $\_GET ['term'] = 'te'; |
| Expected Output | “test” |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSautocomplete02 |
| Purpose | This test verifies that the system correctly autocompletes the given input |
| Setup | Create a user with username “test” |
| Input | $\_GET ['term'] = 'xxxxxxxx'; |
| Expected Output | “” |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSverify01 |
| Purpose | This test verifies that the system correctly checks for valid screenName |
| Setup | No user in database with username “John” |
| Input | $\_POST ['username'] = 'john'; $\_POST ['SMS'] ['Message'] = "hello"; |
| Expected Output | "User does not exist. <br />" |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSaverify02 |
| Purpose | This test verifies that the system correctly validates message values |
| Setup | Create a user with username “Peter”  Set user as not validated |
| Input | $\_POST ['username'] = 'Peter' ;$\_POST ['SMS'] ['Message'] = ""; |
| Expected Output | "Please enter a message to be sent. No blank messages allowed. <br />User has not set up the SMS functionality. <br />" |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSverify03 |
| Purpose | This test verifies that the system does not send SMS longer than allowed |
| Setup | Create user in database with username “Mike”  Set user as not validated |
| Input | $\_POST ['username'] = 'mike';  $\_POST ['SMS'] ['Message'] = " In a relational database, a Weak Entity is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key.,"; |
| Expected Output | ""The message is too big. Please create a message with less than 160 characters <br />User does not have a phone number in record. <br />User has not set up the SMS functionality. <br />" |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSvalidate01 |
| Purpose | This test verifies that the system correctly handles validation |
| Setup | Create user in database with username “Hello5”  Set Validation Code as 5543 |
| Input | $\_POST ['BasicInfo'] ['smsCode'] = null; |
| Expected Output | "Please enter a code" |
|  | Validated = 0; |
|  |  |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSvalidate02 |
| Purpose | This test verifies that the system correctly handles validation |
| Setup | Create user in database with username “Hello5”  Set Validation Code as 5543  Set tries = 0; |
| Input | Create user in database with username Create $\_POST ['BasicInfo'] ['smsCode'] = '23';  $\_POST ['BasicInfo'] ['smsCode'] = '23';  $\_POST ['BasicInfo'] ['smsCode'] = '23';  $\_POST ['BasicInfo'] ['smsCode'] = '23'; |
| Expected Output | "Invalid codeInvalid codeInvalid codeInvalid code"  Tries = 4;  Validated = 0; |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSvalidate03 |
| Purpose | This test verifies that the system correctly handles validation |
| Setup | Create user in database with username “Hello5”  Set Validation Code as 5543  Set tries = 0; |
| Input | $\_POST ['BasicInfo'] ['smsCode'] = 2; |
| Expected Output | $this->expectOutputString ( "Too many wrong inputs, please contact Hola@aol.com\n" );  Tries = 4;  Validated = 0; |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSvalidate04 |
| Purpose | This test verifies that the system correctly handles validation |
| Setup | Create user in database with username “Hello5”  Set Validation Code as 9555  Set tries = 0; |
| Input | $\_POST ['BasicInfo'] ['smsCode'] = 9555; |
| Expected Output | Tries = 0;  Validated = 1; |

|  |  |
| --- | --- |
| Identifier | VJF\_SubSystem\_SMSchangePref\_01 |
| Purpose | This test verifies that the system correctly changes the sms preferences |
| Setup | Create user in database with username “Hello5”  Set allowSMS = 1 |
| Input | $\_POST ['BasicInfo'] ['allowSMS'] = 0; |
| Expected Output | allowSms = 0; |

|  |  |
| --- | --- |
| Identifier | VJF-055-ss1 test login |
| Purpose | This test verifies that users can successfully login using valid FIU SCIS credentials |
| Setup | set valid fiu SCIS user  Create a fixture with basic info information |
| Input |  |
| Expected Output | User->fiucsid > 1000 |

|  |  |
| --- | --- |
| Identifier | VJF-055-ss2 test user is not duplicate |
| Purpose | This test verifies that FIU SCIS users trying to log in do not have a previous account in the system |
| Setup | set valid fiu SCIS user with an already linked non-FIU SCIS account |
| Input | $\_POST ['panthermail'] = 'ncapo006';  $\_POST ['pantherid'] = "999999"; |
| Expected Output | User email is already linked with another account. |

|  |  |
| --- | --- |
| Identifier | VJF-055-ss3 test user is activated |
| Purpose | This test verifies that FIU SCIS users have their accounts activated immediately upon account creation |
| Setup | set valid fiu SCIS user with no previous account |
| Input |  |
| Expected Output | User->activated >= 1 |

* 1. **System Tests**

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Send\_SMS\_TC01 |
| Purpose | This test verifies that the system does not allow empty messages to be sent |
| Setup | User is logged in as an employer  User has validated phone number |
| Input | Student username: johnm  Message: “” |
| Expected Output | "Please enter a message to be sent. No blank messages allowed” |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Send\_SMS\_TC02 |
| Purpose | This test verifies that a username does not exist and notifies the user |
| Setup | User is logged in as an employer  User has validated phone number  Student username: Markus25” is not in database |
| Input | Student username: “Markus25”  Message: “Hello sir, how are you? |
| Expected Output | “User does not exist” |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Send\_SMS\_TC03 |
| Purpose | This test verifies that usernames with special characters are properly handled |
| Setup | User is logged in as an employer  User has validated phone number  Student username: @#$#@%###” is not in database |
| Input | Student username: “Markus25”  Message: “Hello sir, how are you? |
| Expected Output | “User does not exist” |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Confirm\_Phone\_number\_TC01 |
| Purpose | This test verifies that correct validation codes are accepted. |
| Setup | User is logged in as an employer  User has validated phone number  User has code 5252 associated with his account |
| Input | Code: 5252 |
| Expected Output | User is redirected to SMS page |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Confirm\_Phone\_number\_TC02 |
| Purpose | This test verifies that incorrect validation codes are not accepted. |
| Setup | User is logged in as an employer  User has validated phone number  User has code 5252 associated with his account |
| Input | Code: 4444 |
| Expected Output | "Invalid code" |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Confirm\_Phone\_number\_TC03 |
| Purpose | This test verifies that null codes are not accepted and user is notified |
| Setup | User is logged in as an employer  User has validated phone number  User has code 5252 associated with his account |
| Input | Code: “” |
| Expected Output | "Please enter a code" |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Share\_Screen\_TC01 |
| Purpose | This test verifies that users are able to share their screens |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page |
| Input | User clicks on Screen Share  User clicks on share Screen |
| Expected Output | “Your screen is being Shared”  Java applet loads |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Share\_Screen\_TC02 |
| Purpose | This test verifies that users are able to share their screens after cancelling the java applet |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page |
| Input | User clicks on Screen Share  User clicks on share Screen  User cancels java applet popup  User clicks on Share Screen  User clicks ok on java applet |
| Expected Output | “Your screen is being Shared”  Java applet loads |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Share\_Screen\_TC03 |
| Purpose | This test verifies that users are not able to share more than two screens at a time |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page |
| Input | User clicks on Screen Share  User clicks on share Screen  User clicks ok on java applet  Screen Sharing app starts  User clicks on Share Screen |
| Expected Output | "There is an active screenShare at the moment, please wait until it has ended before attempting to share a screen" |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_Share\_Screen\_TC04 |
| Purpose | This test verifies that users are not able to share more than two screens at a time |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page |
| Input | User clicks on Screen Share  User clicks on share Screen  User clicks ok on java applet  Screen Sharing app starts  Other user involved in interview clicks on share |
| Expected Output | "There is an active screenShare at the moment, please wait until it has ended before attempting to share a screen" |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_View\_Screen\_Share\_TC01 |
| Purpose | This test verifies that users can correctly see each other’s screen |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  Student is sharing Screen |
| Input | User clicks on screen Share  User clicks on view screen |
| Expected Output | Other user’s screen is displayed. |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_View\_Screen\_Share\_TC02 |
| Purpose | This test verifies that users can correctly hide the screen Share |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  Student is sharing Screen |
| Input | User clicks on screen Share  User clicks on view screen  User clicks on hide screen |
| Expected Output | Screen is hidden. |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_View\_Screen\_Share\_TC03 |
| Purpose | This test verifies that the system correctly displays the stock screen |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  No one is sharing screen |
| Input | User clicks on screen Share  User clicks on view screen |
| Expected Output | Stock screen is displayed |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_End\_Screen\_Share\_TC01 |
| Purpose | This test verifies that the system correctly terminates the screen Share |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  User is Sharing screen |
| Input | User clicks on end Screen |
| Expected Output | Screen Share is terminated |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_End\_Screen\_Share\_TC02 |
| Purpose | This test verifies that the system correctly terminates the screen Share |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  User is Sharing screen |
| Input | User clicks on stop share  User clicks on end Screen |
| Expected Output | Screen Share is terminated |

|  |  |
| --- | --- |
| Identifier | VJF\_SYSTEM\_End\_Screen\_Share\_TC03 |
| Purpose | This test verifies that the system correctly terminates the screen Share |
| Setup | User is logged in as an employer  User has set up interview with student  User is in interview page  User is Sharing screen |
| Input | User presses refresh on browser window |
| Expected Output | Screen Share is terminated |

|  |  |
| --- | --- |
| Identifier | VJF-048 Create Document |
| Purpose | This test verifies that users can successfully create a new document |
| Setup | 1. click on Collaborative Editor 2. click on create new document button |
| Input |  |
| Expected Output | "New Document is created” |

|  |  |
| --- | --- |
| Identifier | VJF-048 Create Document |
| Purpose | This test verifies that only loggeid in users can successfully create a new document |
| Setup | 1. click on Collaborative Editor 2. click on create new document button |
| Input | NONE |
| Expected Output | "Unable to create new document, user has been logged out due to innactivity” |

|  |  |
| --- | --- |
| Identifier | VJF-049 Share Active Document |
| Purpose | This test verifies that users can successfully share a new document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents button 3. select active document 4. click on share document 5. Invitation is sent to other user 6. Other user receives notification 7. Other user joins the shared document session |
| Input | NONE |
| Expected Output | "New Document is shared and collaboration on document is initiated” |

|  |  |
| --- | --- |
| Identifier | VJF-049 Share Active Document |
| Purpose | This test verifies that users can not share a new document when a remote user has not logged into the interview |
| Setup | 1. click on Collaborative Editor 2. click on manage documents button 3. select active document 4. click on share document 5. Invitation is sent to other user |
| Input | NONE |
| Expected Output | "User is notified that remote user has not logged in” |

|  |  |
| --- | --- |
| Identifier | VJF-049 Share Active Document |
| Purpose | This test verifies that users can’t share a document unless they has selected one from his list of documents or an active document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents button 3. click on share document |
| Input | NONE |
| Expected Output | "User is notified that no active document has been selected to share” |

|  |  |
| --- | --- |
| Identifier | VJF-050 Delete Document |
| Purpose | This test verifies that users can successfully delete an existing document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. select document to delete 4. click delete document |
| Input | NONE |
| Expected Output | "Document is deleted” |

|  |  |
| --- | --- |
| Identifier | VJF-050 Delete Document |
| Purpose | This test verifies that users can’t delete a document unless they have selected one from his list of documents or an active document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click delete document |
| Input | NONE |
| Expected Output | " User is notified that no active document has been selected to delete” |

|  |  |
| --- | --- |
| Identifier | VJF-051 Import Document |
| Purpose | This test verifies that users can successfully import a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on document import 4. drag document over import area 5. document is imported |
| Input | NONE |
| Expected Output | “Document is imported” |

|  |  |
| --- | --- |
| Identifier | VJF-051 Import Document |
| Purpose | This test verifies that users only import a document by dragging it into the specified import area |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on document import 4. drag document outside of import area |
| Input | NONE |
| Expected Output | “Document import fails” |

|  |  |
| --- | --- |
| Identifier | VJF-051 Import Document |
| Purpose | This test verifies that users can only upload a document that of a valid document format |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on document import 4. drag document inside import area |
| Input | NONE |
| Expected Output | “Document import fails, document is invalid” |

|  |  |
| --- | --- |
| Identifier | VJF-052 Rename Document |
| Purpose | This test verifies that users can successfully rename a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on rename document 4. enter new name in input box 5. document is renamed |
| Input | NONE |
| Expected Output | “Document is renamed” |

|  |  |
| --- | --- |
| Identifier | VJF-052 Rename Document |
| Purpose | This test verifies that users can only rename a document following the utf8-en encoding character set |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on rename document 4. enter new name in input box 5. document is renamed |
| Input | NONE |
| Expected Output | “Document is not renamed, user entered an invalid character” |

|  |  |
| --- | --- |
| Identifier | VJF-053 Save Document |
| Purpose | This test verifies that users can successfully save a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click on save document 4. document is saved |
| Input | NONE |
| Expected Output | “Document is saved” |

|  |  |
| --- | --- |
| Identifier | VJF-053 Save Document |
| Purpose | This test verifies that users are warned of document data loss if user chooses to navigate away from active document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. user navigates away from interview portal without saving |
| Input | NONE |
| Expected Output | “User is warned that if they navigates away from editor, document data will be lost/not saved” |

|  |  |
| --- | --- |
| Identifier | VJF-054 Open Document |
| Purpose | This test verifies that users can successfully open a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. select document to open 4. click open document 5. document is opened |
| Input | NONE |
| Expected Output | “Document is opened in the editor” |

|  |  |
| --- | --- |
| Identifier | VJF-054 Open Document |
| Purpose | This test verifies that users can’t open a document unless they have selected one from his list of documents or an active document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click open document |
| Input | NONE |
| Expected Output | “User is notified that no active document has been selected to open” |

|  |  |
| --- | --- |
| Identifier | VJF-055 Export Document |
| Purpose | This test verifies that users can successfully export a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. select document to export 4. click export document 5. document is exported |
| Input | NONE |
| Expected Output | “Document is downloaded on the user’s browser” |

|  |  |
| --- | --- |
| Identifier | VJF-055 Export Document |
| Purpose | This test verifies that users can successfully export a document |
| Setup | 1. click on Collaborative Editor 2. click on manage documents 3. click export document |
| Input | NONE |
| Expected Output | “User is notified that no active document has been selected to export” |

|  |  |
| --- | --- |
| Identifier | VJF-055 Login using FIU SCIS Credentials |
| Purpose | This test verifies that users can successfully login using FIU SCIS credentials |
| Setup | 1. click on FIU SCIS login 2. enter username 3. enter password 4. click login/submit |
| Input | FIU SCIS username  Panther id as password |
| Expected Output | “user successfully logs in” |

|  |  |
| --- | --- |
| Identifier | VJF-055 Login using FIU SCIS Credentials |
| Purpose | This test verifies that users can successfully login only using valid FIU SCIS credentials |
| Setup | 1. click on FIU SCIS login 2. enter non-valid username 3. enter non-valid password 4. click login/submit |
| Input | FIU SCIS username  Panther id as password |
| Expected Output | “login form resets itself, user can’t login user has to go through process again” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Upload\_Image\_TC01** |
| **Purpose** | This test verifies that the system does not non-image files to be uploaded |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  File “1. CGS 3095 - Course Overview.pdf” was the last uploaded file |
| **Input** | File name: “1. CGS 3095 - Course Overview.pdf” |
| **Expected Output** | This file is not an image. Please try another file. |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Upload\_Image\_TC02** |
| **Purpose** | This test verifies that the system allows .png images to be uploaded |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  User uploads the file “2a.png” |
| **Input** | File name: “2a.png” |
| **Expected Output** | “File upload was successful!” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Upload\_Image\_TC03** |
| **Purpose** | This test verifies that the system allows images with consecutive dots (.) to be uploaded |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  User uploads the file “2a..png” |
| **Input** | File name: “2a..png” |
| **Expected Output** | “File upload was successful!” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_View\_Uploaded\_Image\_TC01** |
| **Purpose** | This test verifies that the system does not display a non-image file, if it has been uploaded |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  The “8VJFID6675162.pdf” was the last uploaded file |
| **Input** | File name: “8VJFID6675162.pdf” |
| **Expected Output** | “There is no image to view in the system” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_View\_Uploaded\_Image\_TC02** |
| **Purpose** | This test verifies that the system that the system does not allow to view an empty upload |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  The “8VJFID6675162.pdf” was the last uploaded file |
| **Input** | File name: “” |
| **Expected Output** | “There is no image to view in the system” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_View\_Uploaded\_Image\_TC03** |
| **Purpose** | This test verifies that “.jpg” images are able to be viewed after they are uploaded |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started  The “test.jpg” was the last uploaded file |
| **Input** | File name: “test.jpg” |
| **Expected Output** | Display of image “test.jpg” |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Show\_or\_Restore\_Whiteboard\_TC1** |
| **Purpose** | This test verifies that the whiteboard can be shown after a document has been shared |
| **Setup** | User is logged in as an employer or student  A document has been shared with the document-sharing feature |
| **Input** | User clicks on the “Whiteboard” blue button on the left-hand side of the interview page |
| **Expected Output** | Buttons of the document sharing feature fade from view and the whiteboard buttons and whiteboard screen appears |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Show\_or\_Restore\_Whiteboard\_TC2** |
| **Purpose** | This test verifies that only one instance of the whiteboard is displayed after “Whiteboard” is clicked more than once in a row |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started |
| **Input** | User clicks on the “Whiteboard” blue button on the left-hand side of the interview page |
| **Expected Output** | Buttons of the document sharing feature fade from view and the whiteboard buttons and whiteboard screen appears |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Show\_or\_Restore\_Whiteboard\_TC3** |
| **Purpose** | This test verifies that only one instance of the whiteboard is displayed after “Whiteboard” is clicked more than once in a row |
| **Setup** | User is logged in as an employer or student  Whiteboard session has been started |
| **Input** | User clicks on the “Whiteboard” blue button on the left-hand side of the interview page |
| **Expected Output** | Buttons of the document sharing feature fade from view and the whiteboard buttons and whiteboard screen appears |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Select\_Upload\_Image\_TC01** |
| **Purpose** | This test case verifies that the system supports selecting an image whose file name contains extended ASCII code characters |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User right clicks on “Choose File”  User selects file “ŒšÆß.jpg” |
| **Expected Output** | The file name is displayed in the upload text box successfully |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Select\_Upload\_Image\_TC02** |
| **Purpose** | This test case verifies that the system supports selecting an image whose file name contains extended ASCII code characters |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User right clicks on “Choose File”  User selects file “ŒšÆß.jpg” |
| **Expected Output** | The file name is displayed in the upload text box successfully |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Draw\_With\_Pencil\_Tool\_TC1** |
| **Purpose** | This test case verifies that the right click also draws on the whiteboard |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User right clicks anywhere in the whiteboard screen to draw |
| **Expected Output** | The drawing made by the user appears on the whiteboard screen |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Draw\_With\_Pencil\_Tool\_TC2** |
| **Purpose** | This test case verifies that a drawing persists after a whiteboard screen goes in, out and into view again |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User draws on the whiteboard and clicks on another feature  User clicks on “Whiteboard” again |
| **Expected Output** | The drawing made by the user appears on the whiteboard screen after the whiteboard is in view |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Change\_Pencil\_Tool\_Color\_TC1** |
| **Purpose** | This test case  verifies that the selected pencil tool color is maintained when the whiteboard is substituted by another feature and then displayed again |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User changes the color of the pencil tool by clicking the “Color” button  User clicks on another feature  User clicks on “Whiteboard” again |
| **Expected Output** | The initial color selected should be maintained  as selected in the past whiteboard session |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Change\_Pencil\_Tool\_Color\_TC2** |
| **Purpose** | This test case  verifies that the selected pencil tool color is maintained after an image is uploaded and the whiteboard session restored |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User uploads a new image through the image-sharing tool  User clicks on the “Whiteboard” button again |
| **Expected Output** | The initial color of the pencil tool  is still the same after uploading an image |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Type\_Text\_Into\_Whiteboard\_TC01** |
| **Purpose** | This test case  verifies that the whiteboard container supports word wrapping with the text feature of the whiteboard |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User clicks on right border of the whiteboard container  Text box input: “123456789123456789123456789123456789123456789123456789” |
| **Expected Output** | The overflowing text is sent to the next line instead of disappearing in the non-visible part of the whiteboard |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Type\_Text\_Into\_Whiteboard\_TC02** |
| **Purpose** | This test case  verifies that the text font color is maintained after text is typed with the whiteboard |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started |
| **Input** | User selects the green color from the color palette in the “Color” tab of the whiteboard  Text box input: “Drawing on the text box”  User selects color palette to see which color is selected |
| **Expected Output** | The color selected is the same as the one which was selected before |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Clear\_Whiteboard\_Contents\_TC01** |
| **Purpose** | This test case  verifies that system tells the user that a whiteboard is already empty when clearing an empty whiteboard |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started  Whiteboard has not been written on or has been cleared |
| **Input** | User selects the “Clear” option from the whiteboard menu |
| **Expected Output** | The system lets the user know that the whiteboard is already empty |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Clear\_Whiteboard\_Contents\_TC02** |
| **Purpose** | This test case  verifies that a selected color is maintained after a whiteboard has been cleared |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started  User selects a color from the color palette |
| **Input** | User selects the “Clear” option from the whiteboard menu |
| **Expected Output** | The previous color selected by the user should be still selected when the color palette is displayed |

|  |  |
| --- | --- |
| **Identifier** | **VJF\_Erase\_From\_Whiteboard\_TC01** |
| **Purpose** | This test case verifies that a selected color is maintained after a user has erased from the whiteboard |
| **Setup** | User is logged in as an employer or student Whiteboard session has been started  User selects a color from the color palette  User draws on the whiteboard |
| **Input** | User uses the erase feature by clicking on the “Erase” button on the whiteboard menu and dragging it on top of the drawing |
| **Expected Output** | The previous color selected by the user should be still selected when the color palette is displayed |

## 7.3 Evaluation of Tests

**Subsystem**

|  |  |
| --- | --- |
| Identification | Result |
| VJF\_SubSystem\_SMSconfirm01 | PASS |
| VJF\_SubSystem\_SMSautocomplete01 | PASS |
| VJF\_SubSystem\_SMSautocomplete02 | PASS |
| VJF\_SubSystem\_SMSverify01 | PASS |
| VJF\_SubSystem\_SMSaverify02 | PASS |
| VJF\_SubSystem\_SMSverify03 | PASS |
| VJF\_SubSystem\_SMSvalidate01 | PASS |
| VJF\_SubSystem\_SMSvalidate02 | PASS |
| VJF\_SubSystem\_SMSvalidate03 | PASS |
| VJF\_SubSystem\_SMSvalidate04 | PASS |
| VJF\_SubSystem\_SMSchangePref\_01 | PASS |
| VJF-055-ss1 test login | PASS |
| VJF-055-ss2 test user is not duplicate | PASS |
| VJF-055-ss3 test user is activated | PASS |

**System**

|  |  |
| --- | --- |
| Identification | Result |
| VJF\_SYSTEM\_Send\_SMS\_TC01 | PASS |
| VJF\_SYSTEM\_Send\_SMS\_TC02 | PASS |
| VJF\_SYSTEM\_Send\_SMS\_TC03 | PASS |
| VJF\_SYSTEM\_Confirm\_Phone\_number\_TC01 | PASS |
| VJF\_SYSTEM\_Confirm\_Phone\_number\_TC02 | PASS |
| VJF\_SYSTEM\_Confirm\_Phone\_number\_TC03 | PASS |
| VJF\_SYSTEM\_Share\_Screen\_TC01 | PASS |
| VJF\_SYSTEM\_Share\_Screen\_TC02 | PASS |
| VJF\_SYSTEM\_Share\_Screen\_TC03 | PASS |
| VJF\_SYSTEM\_Share\_Screen\_TC04 | PASS |
| VJF\_SYSTEM\_View\_Screen\_Share\_TC01 | PASS |
| VJF\_SYSTEM\_View\_Screen\_Share\_TC02 | PASS |
| VJF\_SYSTEM\_View\_Screen\_Share\_TC03 | PASS |
| VJF\_SYSTEM\_End\_Screen\_Share\_TC01 | PASS |
| VJF\_SYSTEM\_End\_Screen\_Share\_TC02 | PASS |
| VJF\_SYSTEM\_End\_Screen\_Share\_TC02 | PASS |
| VJF\_SYSTEM\_End\_Screen\_Share\_TC03 | PASS |
| VJF\_Upload\_Image\_TC01 | PASS |
| VJF\_Upload\_Image\_TC02 | PASS |
| VJF\_Upload\_Image\_TC03 | PASS |
| VJF\_View\_Uploaded\_Image\_TC01 | FAIL |
| VJF\_View\_Uploaded\_Image\_TC02 | FAIL |
| VJF\_View\_Uploaded\_Image\_TC03 | PASS |
| VJF\_Show\_or\_Restore\_Whiteboard\_TC01 | PASS |
| VJF\_Show\_or\_Restore\_Whiteboard\_TC02 | PASS |
| VJF\_Show\_or\_Restore\_Whiteboard\_TC03 | PASS |
| VJF\_Select\_Upload\_Image\_TC01 | PASS |
| VJF\_Select\_Upload\_Image\_TC02 | PASS |
| VJF\_Draw\_With\_Pencil\_Tool\_TC01 | PASS |
| VJF-Create Document\_TC01 | PASS |
| VJF-Create Document\_TC01 | PASS |
| VJF-Share Active Document \_TC01 | PASS |
| VJF-Share Active Document \_TC01 | PASS |
| VJF-Share Active Document\_TC01 | PASS |
| VJF-Delete Document\_TC01 | PASS |
| VJF-Delete Document\_TC00 | PASS |
| VJF-Import Document\_TC01 | PASS |
| VJF-Import Document\_TC02 | PASS |
| VJF-Import Document\_TC03 | PASS |
| VJF-Rename Document\_TC01 | PASS |
| VJF- Rename Document\_TC02 | PASS |
| VJF-Save Document\_TC01 | PASS |
| VJF- Save Document\_TC02 | PASS |
| VJF-Save Document\_TC03 | PASS |
| VJF-Open Document\_TC01 | PASS |
| VJF-Open Document\_TC02 | PASS |
| VJF-Export Document\_TC01 | PASS |
| VJF-Export Document\_TC02 | PASS |
| VJF-Login using FIU SCIS Credentials \_TC01 | PASS |
| VJF-Login using FIU SCIS Credentials\_TC02 | PASS |

## 8. Glossary

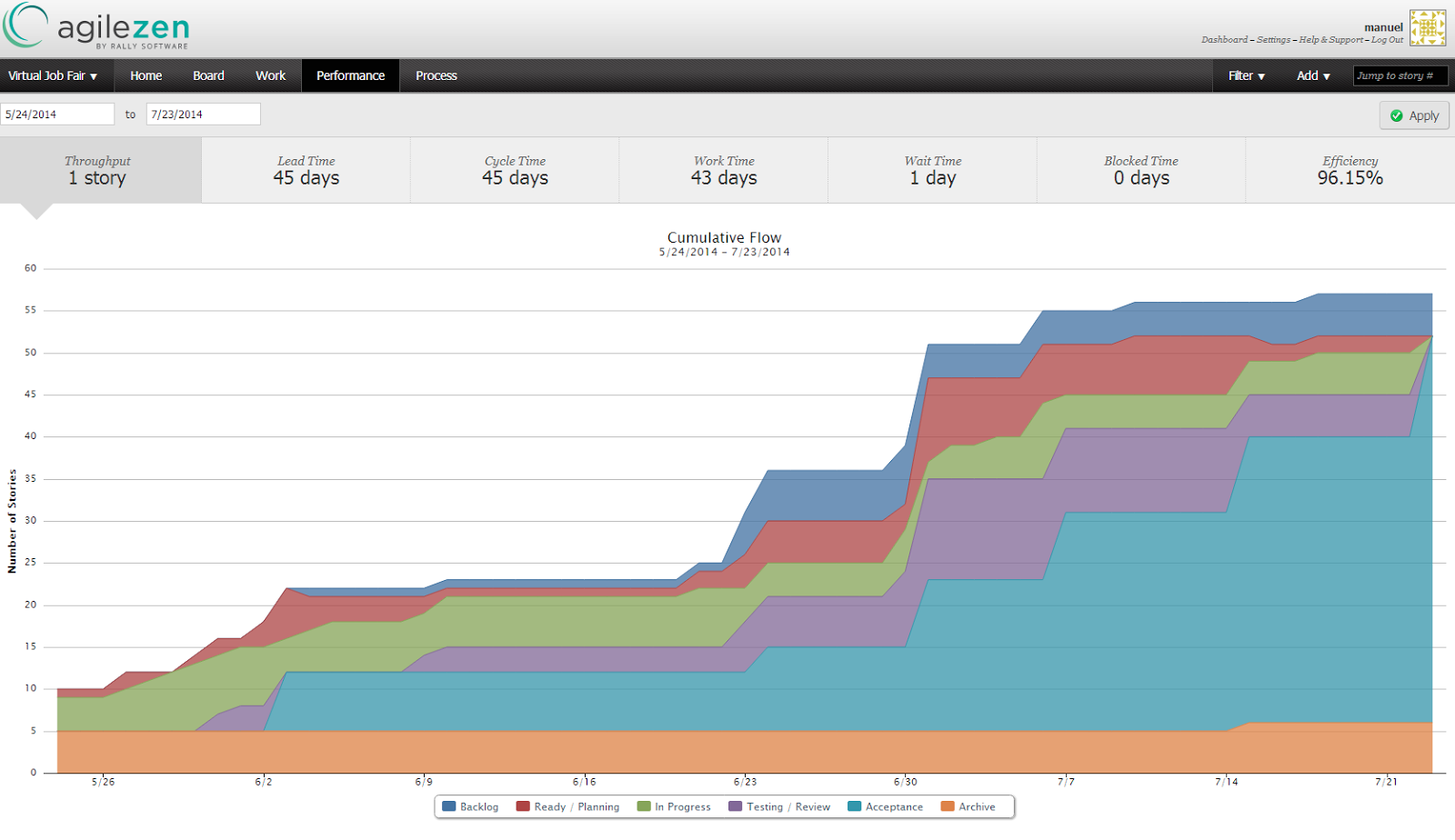
**- Salary:** a periodic payment made to an employee in exchange for services provided. Salaries are provided in yearly terms.

- **Résumé:** a document which describes a student’s qualifications, skills and education

- **Cover Letter:** a document which is used by students to introduce themselves to the companies that they are applying to. It usually goes together with a résumé

# 9. Appendix

## 9.1 Appendix A - Project Performance



**Figure 9.1-1 Project overall performance**

## 9.2 Appendix B - All Use Case with Nonfunctional Requirements

**The following subsection present the textual specification of the use cases for the VFJ system.**

|  |  |
| --- | --- |
| Use Case ID | VJF-001 Registration |
| Description | Registration process for a student type |
| Actor | Student |
| Pre-conditions | ·         User has navigated to Virtual Job Fair |
| Steps | 1. User clicks on register link 2. User selects registration 3. User selects student or employer 4. User fills in required details 5. User clicks submit |
| Post-conditions | 1. User is redirected to page where he/she is asked to check email for verification link 2. Verification email is sent |
| Exceptions | 1. User fails to fill out one of the details in the form 2. User fails to provide an FIU email address |

|  |  |
| --- | --- |
| Use Case ID | VJF-002 View Profile |
| Description | Viewing profiles for student and employers |
| Actor | All Actors |
| Pre-conditions | 1. User is logged in |
| Steps | 1. User clicks on username hyperlink 2. User is redirected to the user profile |
| Post-conditions | 1. User is on profile page |
| Exceptions | 1. If a student is viewing another student’s profile, certain information is withheld |

|  |  |
| --- | --- |
| Use Case ID | VJF-003 Edit Basic Info |
| Description | Allow a user to edit their profile |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in |
| Steps | 1. User clicks on “My Profile” 2. User clicks on “Edit Basic Info” 3. User changes necessary data 4. User clicks “Save” |
| Post-conditions | 1. User is redirected to “My Profile” page |
| Exceptions | 1. User inputs invalid/empty data |

|  |  |
| --- | --- |
| Use Case ID | VJF-004 Verify Email |
| Description | Allow a user to verify his account |
| Actor | Student, Employer, Faculty |
| Pre-conditions | 1. User has registered for an account and email has been sent |
| Steps | 1. User navigates to verification email sent by system 2. User clicks on verification link 3. User is redirected to Virtual Job Fair page to verify Email |
| Post-conditions | 1. Access is granted to user with correct verification link |
| Exceptions | 1. Verification link does not match system’s expectectation |

|  |  |
| --- | --- |
| Use Case ID | VJF-005 Login |
| Description | Allow a user to login to his account |
| Actor | All user types |
| Pre-conditions | 1. User has navigated to Virtual Job Fair website |
| Steps | 1. User enters username and password 2. User clicks “Login” 3. User is redirected to his home page |
| Post-conditions | 1. User is on homepage |
| Exceptions | 1. User entered incorrect username/password combination |

|  |  |
| --- | --- |
| Use Case ID | VJF-006 Logout |
| Description | Allow a user to logout from his account |
| Actor | All user types |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks Logout 2. User is redirected to main page |
| Post-conditions | 1. User is on main page |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-007 Home Page |
| Description | Allow a user to visit Home Page |
| Actor | All user types |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks Home 2. User is redirected to his home page |
| Post-conditions | user is his home page |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-008 Upload Picture |
| Description | Allow user to change his profile picture |
| Actor | all user types |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | User clicks My Profile                 User is redirected to his profile page                 User clicks on the edit picture buttom                 User pick his picture and then click save. |
| Post-conditions | user post his picture |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-009 Upload Resume |
| Description | Allow user to Upload his resume |
| Actor | Student |
| Pre-conditions | User is on Virtual Job Fair                 User is logged in |
| Steps | User clicks My Profile                 User is redirected to his profile page                 User clicks on the edit resume button                 User pick his resume file and then click save. |
| Post-conditions | user post his resume |
| Exceptions | User file invalid/empty data |

|  |  |
| --- | --- |
| Use Case ID | VJF-0011 Add Education |
| Description | Allow user to add education to his profile |
| Actor | Student |
| Pre-conditions | User is on Virtual Job Fair                 User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the add education 4. User add his education info then click save |
| Post-conditions | user add education to his profile |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0012 Delete Education |
| Description | allow user to delete education from his profile |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the delete education button |
| Post-conditions | user delete the education that has been chosen to be deleted |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0013 Add Experience |
| Description | user can add experience to his profile |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the add experience 4. User add his experience info then click save |
| Post-conditions | user add experience to his profile |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0014 Delete Experience |
| Description | user can delete experience from his profile |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the delete experience button |
| Post-conditions | user delete the experience that has been chosen to be deleted |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0015 Change password |
| Description | user can change his password |
| Actor | All Actors |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the change password button 4. User is redirect to a new page for changing his password 5. User is typing his old, new password and retype the new password 6. User click submit and redirect to the login page |
| Post-conditions | user change his password |
| Exceptions | - Old Password was incorrect.  - Passwords do not match |
| Use Case ID | VJF-0016 Add Skill |
| Description | user can add skill to his profile |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User type a new skill 4. User clicks add skill |
| Post-conditions | user add a new skill |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0017 Delete skill |
| Description | user can delete skill from his profile |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User is clicking on the skill delete button |
| Post-conditions | Skill is deleted |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0018 Change skills Order |
| Description | user can change the order of each skill |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User drags the skill to any position 4. User clicks save skills |
| Post-conditions | User change the order of the skill |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0019 Integrate LinkedIn |
| Description | Get user information from LinkedIn |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on LinkedIn link 4. User is redirected to page where prompted for username and password for LinkedIn 5. User enters username and password for LinkedIn and clicks continue. 6. User is redirected to My Profile with complete information from LinkedIn |
| Post-conditions | User profile is built |
| Exceptions | User cancels the action |

|  |  |
| --- | --- |
| Use Case ID | VJF-0022 View Student Profile |
| Description | Allow Employer to view student profile |
| Actor | Employer |
| Pre-conditions | Employer is logged in and is at the home page |
| Steps | 1. Employer is typing student name on the search input. 2. Employer is clicking on the student that he want to view |
| Post-conditions | The employer is on the student profile view |
| Exceptions | The employer is typing a wrong student name that doesn't exist |
| Use Case ID | VJF-0023 Send Message |
| Description | Send a message to a user |
| Actor | A user |
| Pre-conditions | -User is in the compose message page |
| Steps | 1-User populates the ‘To’ field  2-User populates the ‘Subject’ field  3-User types in the message in the text area  4-User clicks ‘Send’ |
| Post-conditions | The system sends the message. The message appears in the inbox of the target user |
| Exceptions | -Inexistent username selected as the receiver of the message  -Wrong username format typed in the ‘To’ field |

|  |  |
| --- | --- |
| Use Case ID | VJF-0024 Reply to Message |
| Description | Reply to a message from some user |
| Actor | A user |
| Pre-conditions | -User has selected a message to be seen |
| Steps | 1-User clicks on the Reply button that appears when reading a message |
| Post-conditions | 2- The System redirects the user to the compose a message page, and the original message the user had selected appears in the text area in the format:  On <Date> <User> wrote:  <message> |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0025 Get Inbox |
| Description | User requests to see all the received messages |
| Actor | A User |
| Pre-conditions | -User is logged in |
| Steps | 1- User navigates to the messages page |
| Post-conditions | User is shown with a list of all the received messages in the format: <Sender>  <Subject> |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0026 Get Sent Messages |
| Description | User requests to see all the messages he/she has sent |
| Actor | A User |
| Pre-conditions | -User is logged in |
| Steps | 1- User selects the ‘Sent” messages from the messages page |
| Post-conditions | User is shown with a list of all the sent messages in the format: <Receiver>  <Subject> |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0027 Get Trashed Messages |
| Description | User requests to see all the trashed messages |
| Actor | A User |
| Pre-conditions | -User is logged in |
| Steps | 1- User selects the ‘Trash” messages from the messages page |
| Post-conditions | User is shown with a list of all the trashed messages in the format: <Sender/Receiver>  <Subject> |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0028 Delete Messages |
| Description | User checks all the messages he/she wants to send to the trash |
| Actor | A User |
| Pre-conditions | -User is logged in |
| Steps | 1-User checks the messages to be sent to the trash  2-User clicks on the trash icon |
| Post-conditions | The selected messages are sent to the trash |
| Exceptions | User does not select any messages before clicking on the trash icon. The System invokes an alert message |

|  |  |
| --- | --- |
| Use Case ID | VJF-0029 Post Job |
| Description | Employer posts a job for students to apply to |
| Actor | Employer |
| Pre-conditions | 1. Employer is logged in 2. Employer is on Home Page |
| Steps | 1. Employer clicks on Post Job Menu Item 2. Employer fills in job details (type, description, compensation, expire date) 3. Employer adds skills to posting if necessary 4. Employer clicks post job |
| Post-conditions | 1. Employer is taken to student match page to view students whose skillset is aligned with the job skillset |
| Exceptions | 1. Employer fills in job details incorrectly, is given an error |

|  |  |
| --- | --- |
| Use Case ID | VJF-0031 Edit Company Info |
| Description | Employer changes his company Information |
| Actor | Employer |
| Pre-conditions | 1. Employer is logged in 2. Employer is viewing his profile |
| Steps | 1. Employer clicks on edit image in company info section 2. Employer changes text in any of the text boxes 3. Employer clicks on checkmark |
| Post-conditions | 1. New company info is saved 2. Employer is redirected back to profile page |
| Exceptions | Employer fills in invalid values for the fields |
| Use Case ID | VJF-0032 Search Jobs |
| Description | Student searches for jobs by skill |
| Actor | Student |
| Pre-conditions | 1. Student is logged in 2. Student is on home page |
| Steps | 1. Student clicks on job search text box 2. Student begins typing a skill 3. Student either selects skill from auto complete or fully types out skill 4. Student clicks submit |
| Post-conditions | 1. Student is redirected to search result page with relevant jobs, and option to search more jobs |
| Exceptions | None |
| Use Case ID | VJF-0033 Search Students |
| Description | Student searches for students by skill |
| Actor | Employer |
| Pre-conditions | 1. Employer is loggedin 2. Employer is on home page |
| Steps | 1. Employer clicks on job search text box 2. Employer begins typing a skill 3. Employer either selects skill from auto complete or fully types out skill 4. Employer clicks submit |
| Post-conditions | Employer is redirected to search result page with relevant students, and option to search more students |
| Exceptions | None |
| Use Case ID | VJF-0034 Close Job |
| Description | Close a job from further applications |
| Actor | Employer |
| Pre-conditions | 1. Employer is logged in |
| Steps | 1. Employer views his own profile 2. Employer selects a job from one of his own postings 3. Employer clicks on “Close Job” |
| Post-conditions | 1. User is redirected back to the job page 2. The job is closed |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0036 Disable User |
| Description | Disable a user from the website |
| Actor | Admin |
| Pre-conditions | 1. Admin is logged in 2. Admin is on home page |
| Steps | 1. Admin enters a search for a username 2. Admin is taken to results page with list of users 3. Admin can disable users by clicking on “delete” |
| Post-conditions | 1. User is disabled 2. Admin is taken back to search page for more users |
| Exceptions | Search may not return any results |

|  |  |
| --- | --- |
| Use Case ID | VJF-0038 Read notification |
| Description | User read notification from his/home page |
| Actor | All user types |
| Pre-conditions | 1. User is logged in 2. User is on home page |
| Steps | 1. User is clicking on the notification section that he or she will like to read from. 2. User gets list of notifications |
| Post-conditions | User read his notifications |
| Exceptions | None |

|  |  |
| --- | --- |
| Use Case ID | VJF-0040 Validate an Employer Register |
| Description | Admin validate a new employer that register |
| Actor | Admin |
| Pre-conditions | 1. Admin is logged in. 2. Admin is on his home page. |
| Steps | Admin is clicking on the notification like that validate the new employer. |
| Post-conditions | The new employer got validate. |
| Exceptions | noon |

|  |  |
| --- | --- |
| Use Case ID | VJF-041 Upload Video Resume |
| Description | Allow user to Upload his resume |
| Actor | Student |
| Pre-conditions | 1. User is on Virtual Job Fair 2. User is logged in |
| Steps | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the edit video resume button 4. User pick his resume file and then click save. |
| Post-conditions | user post his video resume |
| Exceptions | User file invalid/empty data |

|  |  |
| --- | --- |
| Use Case ID | VJF-045 Send SMS to student |
| Description | Allows Employer to send a text message to student |
| Actor | Employer |
| Pre-conditions | 1. User is logged in. 2. User is in the homepage 3. Student has a phone number associated with his account 4. Student has allowed employers to contact him through SMS |
| Steps | 1.      User clicks on send SMS  2.      User is redirected to SMS page  3.   User enters user name of student to contact  4.   User enters message  5.   User presses send |
| Post-conditions | Selected Student receives text message on his phone. |
| Exceptions | The student has not allowed contact by sms  Student has not entered a phone number |

|  |  |
| --- | --- |
| Use Case ID | VJF-056 Log in using FIU SCIS Credentials |
| Description | Allow certain users, FIU SCIS, to login to the system using their FIU SCIS Credentials / UNIX account, Provided by the Senior Project API |
| Actor | Student |
| Pre-conditions | 1. User is at the Login page |
| Steps | 1. User enters his / her username 2. User enters his / her password 3. User is logged in |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. User fails to fill the login form 2. Connection Error |

**Use Cases VJF v3.0**

|  |  |
| --- | --- |
| Use Case ID | VJF-057 Advance search |
| Description | Allow students to search for job. |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. 2. User must be in the Job Page. |
| Steps | 1. User fills the following fields:    * **these words:** +java    * **this exact word or phrase:** “developer”    * **any of these words:** full-time OR part-time    * **none of these words:** -php 2. User checks the ‘**Include jobs from outside sources’** checkbox    * Enters ‘Miami, Florida’ in input box 3. User clicks Search button |
| Post-conditions | Page reloads with job results from CareerPath, Indeed.com, and CareerBuilder.com |
| Exceptions | No jobs matched criteria. |

|  |  |
| --- | --- |
| Use Case ID | VJF-058 Disable API Querying |
| Description | To disable the push of jobs from CareerPath. |
| Actor | Admin |
| Pre-conditions | 1. User must be logged in as Admin. 2. User must be in the Import Job Page. |
| Steps | User clicks on the ‘CareerPath API Status: On‘ button. |
| Post-conditions | The button will switch to say ‘Off’ |
| Exceptions | N/A |

|  |  |
| --- | --- |
| Use Case ID | VJF-059 Set Notifications On |
| Description | Disables Email Job Notifications notifications |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. 2. User must be in the Profile Page. |
| Steps | User clicks on the ‘Email Job Notifications’ Off button |
| Post-conditions | The button changes to On |
| Exceptions | N/A |

|  |  |
| --- | --- |
| Use Case ID | VJF-060 Navigation bar Search |
| Description | Allow students to search for job anywhere on the site. |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. |
| Steps | 1. User clicks on the search box. 2. User enters “java developer” 3. User clicks on the search icon |
| Post-conditions | User is taken to the job page with render job results from CareerPath, Indeed.com, CareerBuilder.com |
| Exceptions | 1. Database connection exception. 2. Internet connection exception. |

|  |  |
| --- | --- |
| Use Case ID | VJF-061 Reset Advance Search Inputs |
| Description | Allows user to clear all the fields in the search form. |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. 2. User must be in the Job Page. 3. User should have the form filled. |
| Steps | User clicks on the Reset Field button |
| Post-conditions | The page reloads with all the fields reset. |
| Exceptions | N/A |

|  |  |
| --- | --- |
| Use Case ID | VJF-062 Set Job Search Status On |
| Description | Allows student to enable job search status. |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. 2. User must be in their Profile Page. |
| Steps | User clicks on ‘Looking For Job:’ Off button |
| Post-conditions | ‘Looking For Job:’ button changes to On |
| Exceptions | N/A |

|  |  |
| --- | --- |
| Use Case ID | VJF-063Linking Account Google |
| Description | User is able to link their system account with third party account. |
| Actor | Student |
| Pre-conditions | 1. User must be logged in as Student. 2. User must be in their Profile Page. 3. User is already logged in Google. |
| Steps | * User clicks on the Google link under Linking Accounts * User clicks on the Accept button when prompt for access. * User selects which name to keep from Merge conflict(s) found popup Page. * User selects which email to keep from Merge conflict(s) found popup Page. * User selects which name to keep from Merge conflict(s) found popup Page. * User clicks on Fix Conflicts * User clicks on View my Profile |
| Post-conditions | Profile pages is reloaded with new information. |
| Exceptions | N/A |

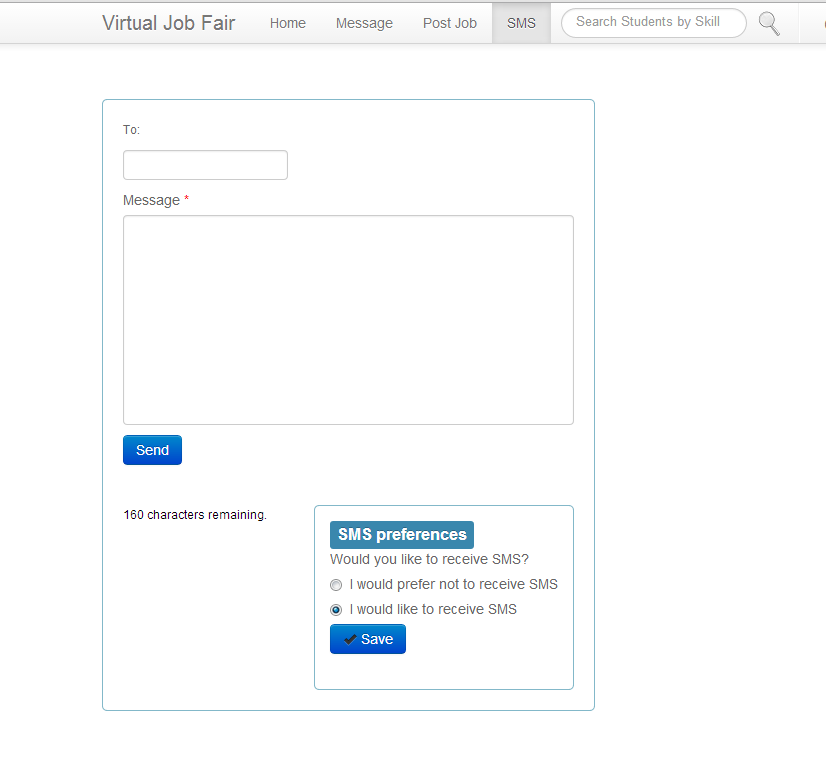
|  |  |
| --- | --- |
| Use Case ID | VJF-064 Admin Enable Notification |
| Description | Admin turns on the system-wide notifications. |
| Actor | Admin |
| Pre-conditions | 1. User must be logged in as Admin  2. User is on the Notifications settings page  3. Notification status button toggle is set to ‘OFF’ |
| Steps | 1. User clicks on the notification toggle button |
| Post-conditions | 1. Last modified by field is populated with the username of the currently logged in administrator user.  2. Last modified date is populated with the current date to indicate the button has been recently modified.  3. Notification status button toggle is set to ‘ON’ |
| Exceptions | Connection is lost exception |

|  |  |
| --- | --- |
| Use Case ID | VJF-065 Import Jobs |
| Description | Bring jobs from the CareerPath API endpoint, including expired jobs. |
| Actor | Admin |
| Pre-conditions | 1. User must be logged as an admin 2. User must be in the import jobs page |
| Steps | 1. User clicks on start date an selects a date 2. User clicks on send date an selects a date 3. User clicks on imports jobs 4. User clicks on the ok button of the pop up |
| Post-conditions | 1. Jobs are persisted to the VJF database.  2. User stays on the import jobs page |
| Exceptions | Database lost connection |

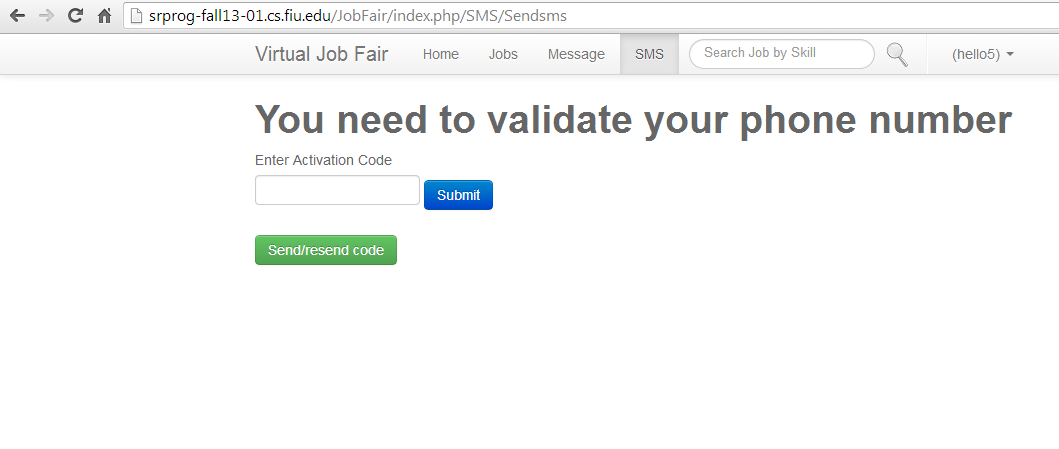
|  |  |
| --- | --- |
| Use Case ID | VJF-066 Save Query |
| Description | Allows student to save search query. |
| Actor | Student |
| Pre-conditions | 1. User must be logged as an student 2. User must be in the jobs page 3. User must have filled the advanced search form |
| Steps | 1. User clicks on the save query button 2. User enters query name on pop up 3. User clicks on Save name button. |
| Post-conditions | Job page reloads. |
| Exceptions | Database Connection Exception. |

|  |  |
| --- | --- |
| **Use Case ID** | VJF-067Merge Account |
| **Description** | Merge two account into one, and provide the user with a view containing any merge conflict generated by the merge of accounts. |
| **Actor** | Student |
| **Pre-conditions** | User must be logged in as Student. |
| **Steps** | 1. User clicks on the drop down button in the navigation bar 2. User click on the Merge Accounts button 3. User enter the username and password of the account that he or she wants merge with 4. User chose the information that he wants to keep 5. User clicks on the fix Conflict button |
| **Post-conditions** | **1** User is prompt with a notification let him or her know that the merge was successful. |
| **Exceptions** | * User fails to provide an username enter the wrong username * User fails to provide a password or enter the wrong password |

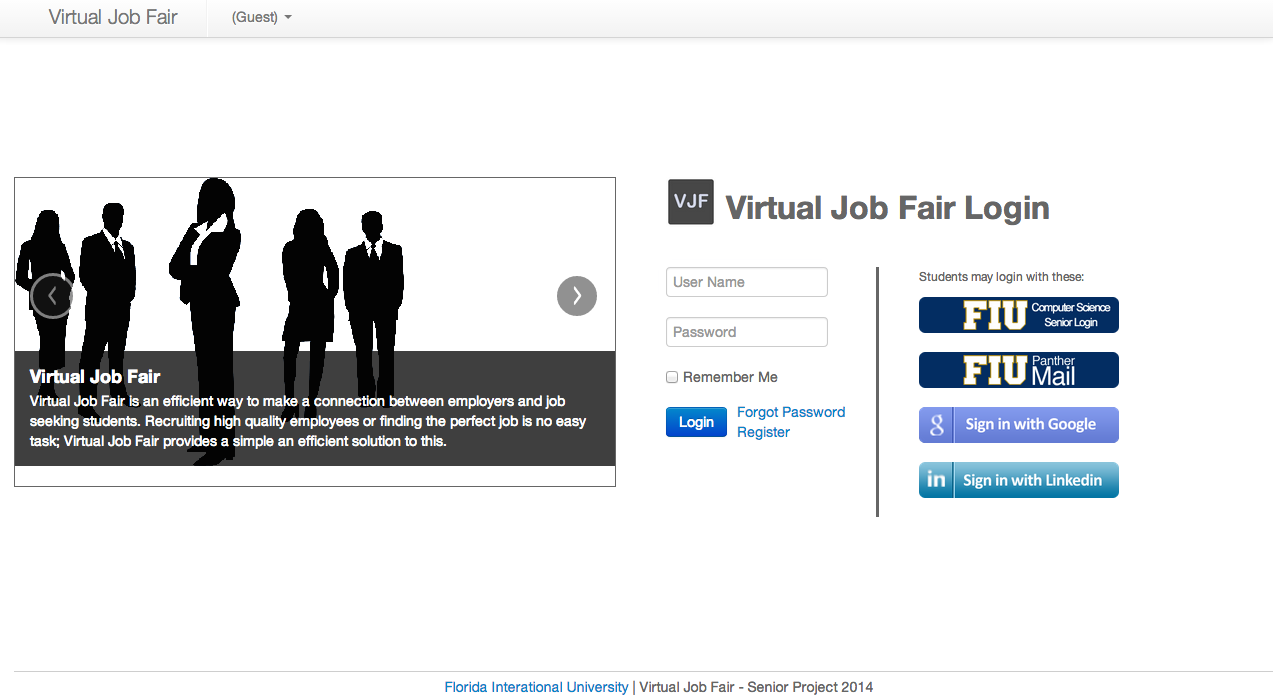
**9.3 Appendix C - User Interface Designs**



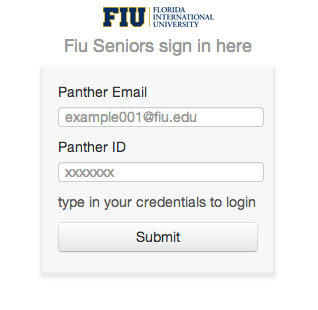
**Figure 9.3-1 Send SMS**



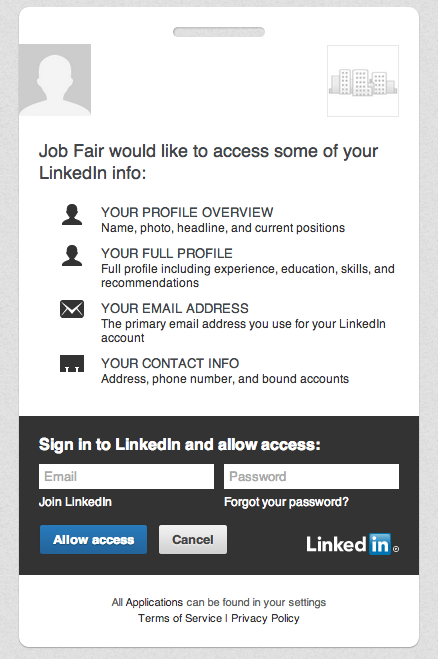
**Figure 9.3-2 Validate Phone Number**



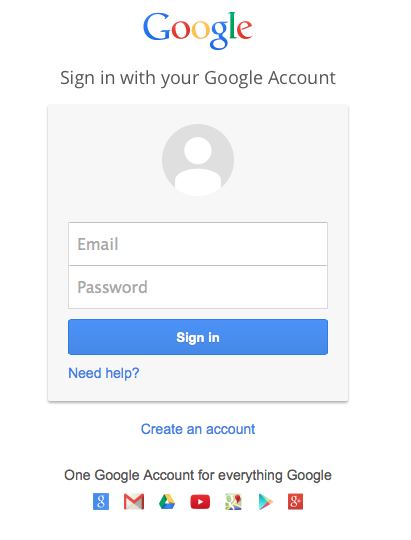
**Figure 9.3-3 Log in Page**



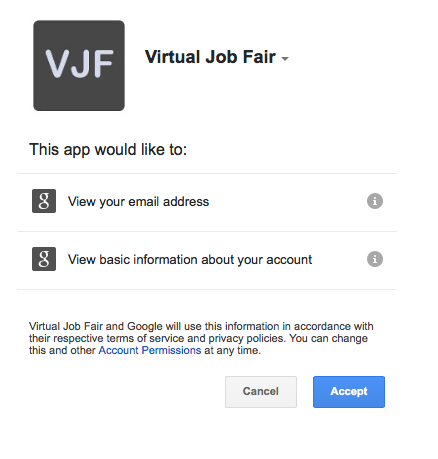
**Figure 9.3-4 Log In Using FIU SCIS Credentials**



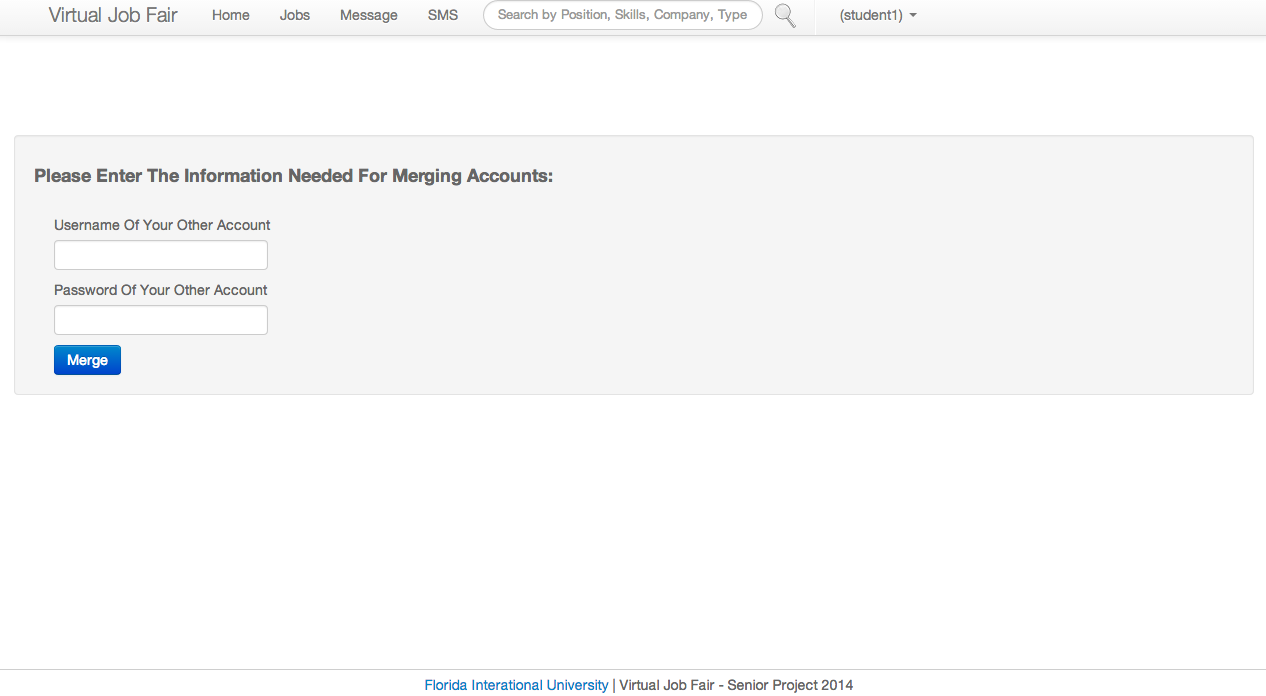
**Figure 9.3-5 Log In Using LinkedIn Credentials**



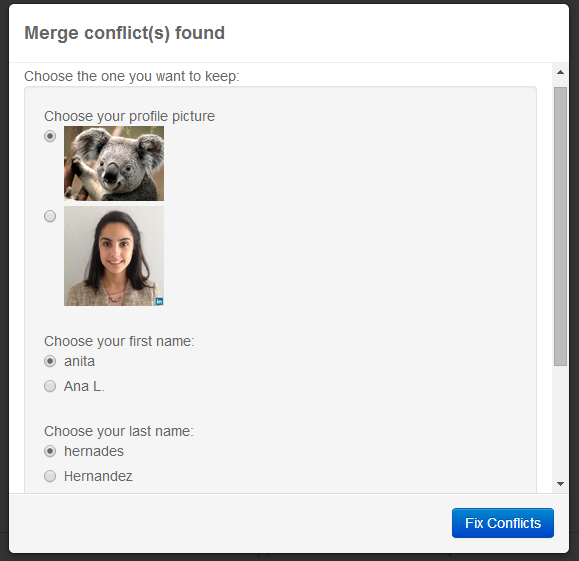
**Figure 9.3-6.1 Log In Using FIU and Google Credentials**



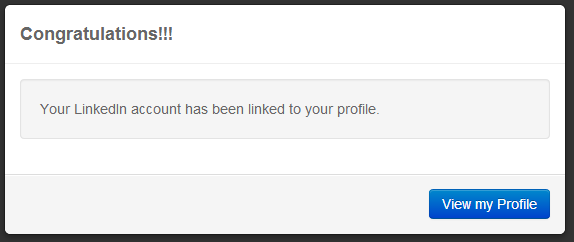
**Figure 9.3-6.2 Log In Using FIU and Google Credentials**



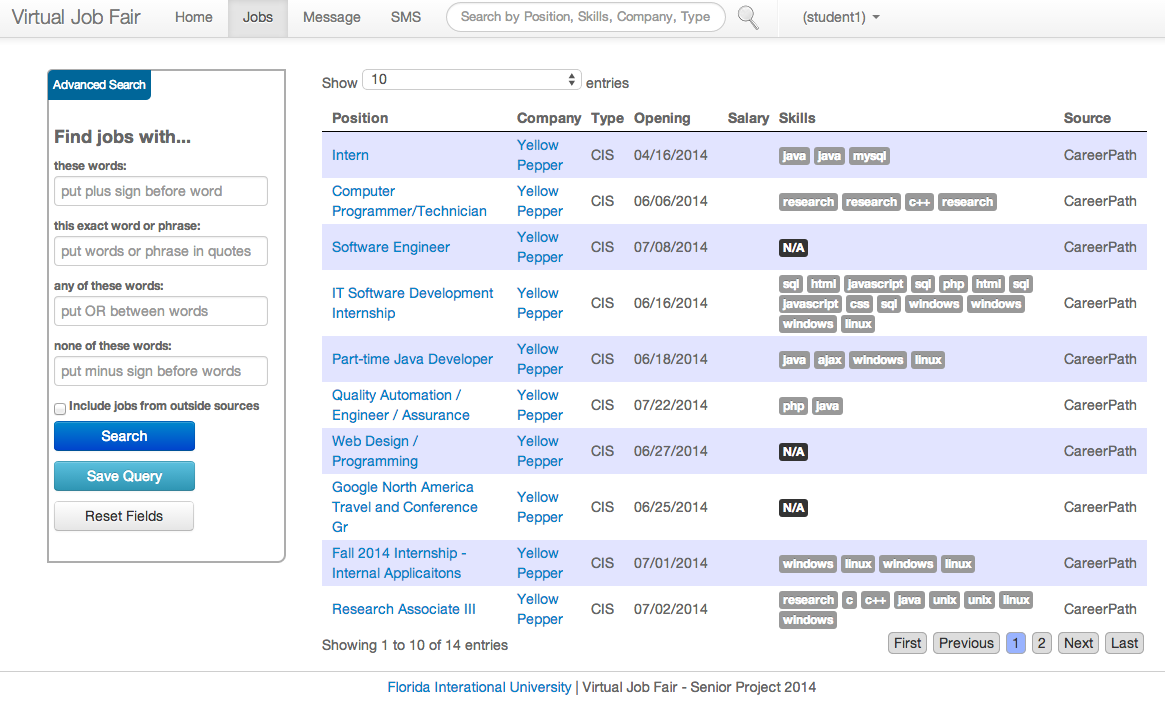
**Figure 9.3-7 Merge Accounts from Profile Drop Down Menu**



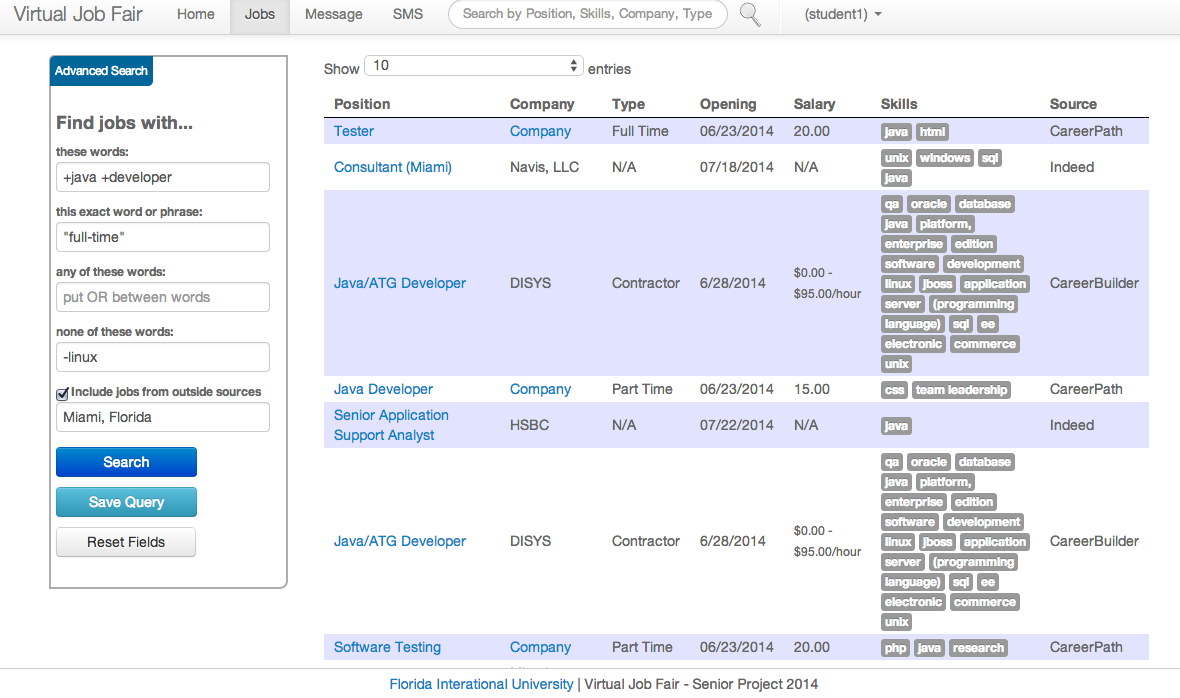
**Figure 9.3-8.1 Merge Conflicts when Linking / Merging Accounts**



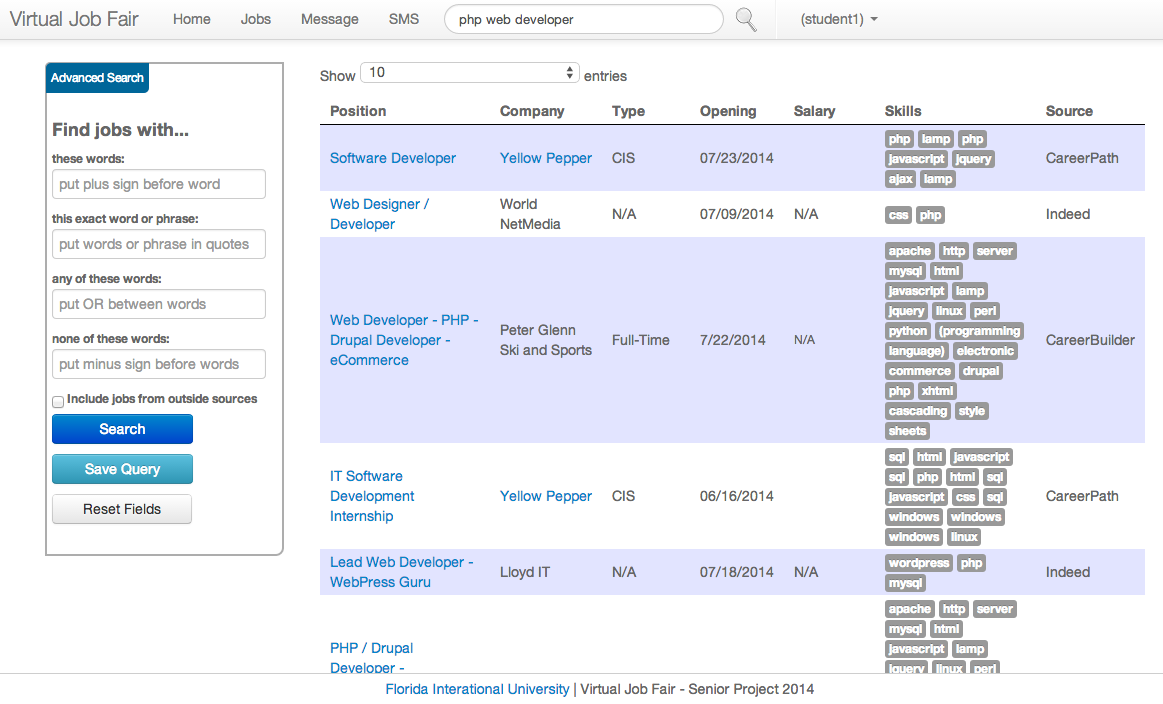
**Figure 9.3-8.2 Merge Conflicts when Linking / Merging Accounts**



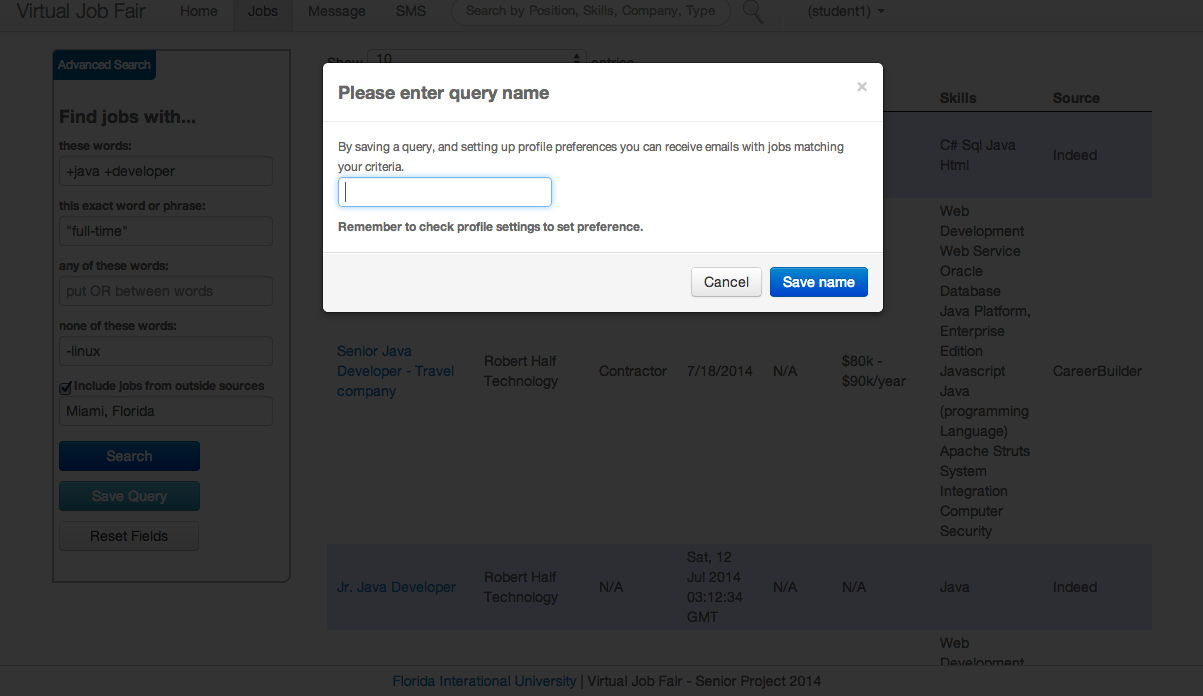
**Figure 9.3-9 Jobs Page**



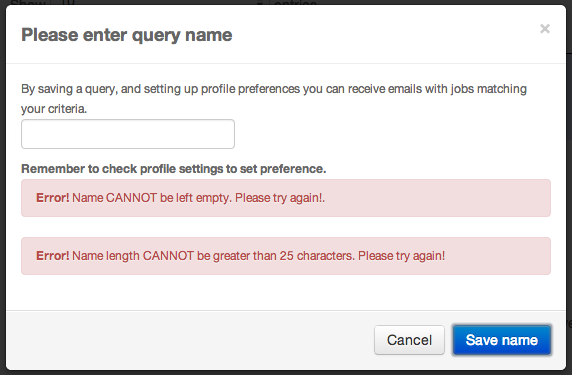
**Figure 9.3-10 Jobs Advanced Search**



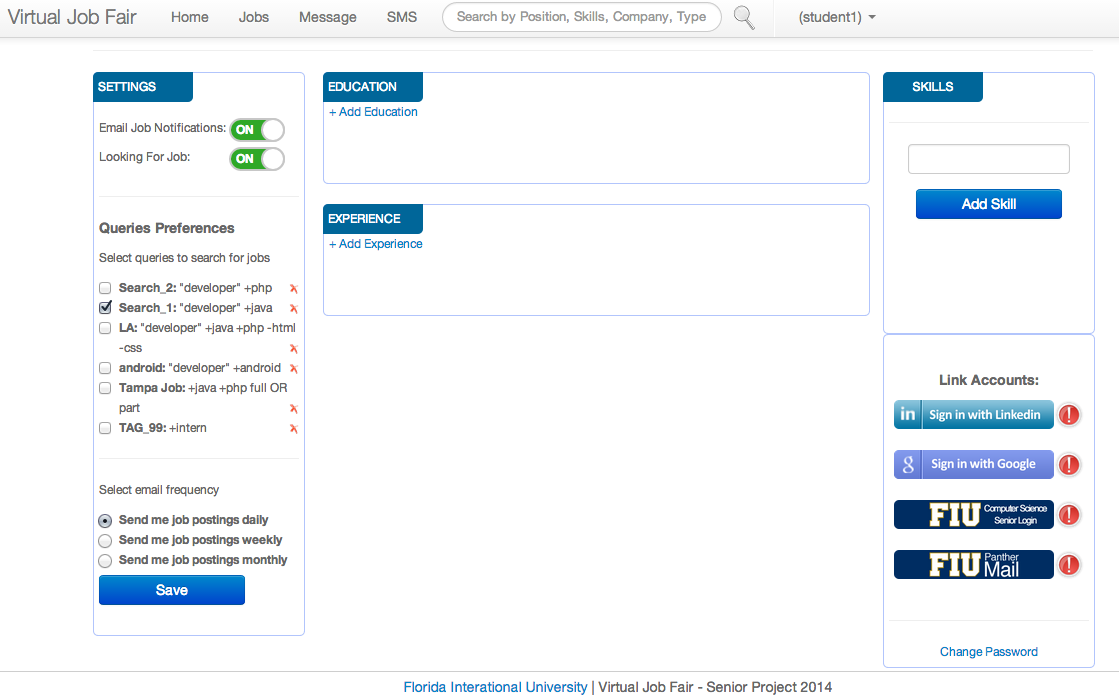
**Figure 9.3-11 Job Search from Navigation Bar**



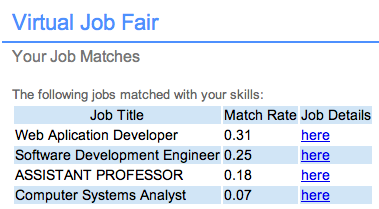
**Figure 9.3-12 Save Query from Advanced Search**



**Figure 9.3-13 Save Query Name Validation**



**Figure 9.3-14 Profile Settings for Query Preference**

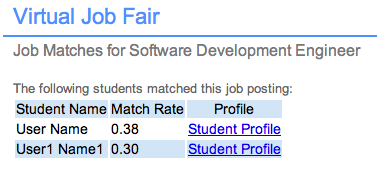


**Figure 9.3-15 Student Email Notification based on Skills**

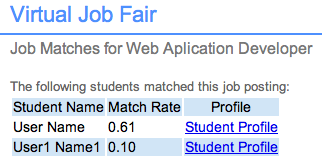




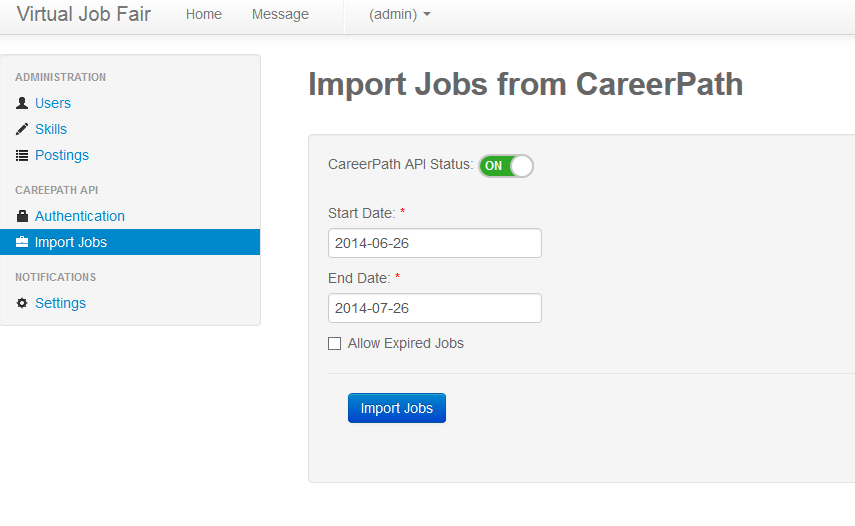
**Figure 9.3-16 Student Email Notification based on Saved Queries**



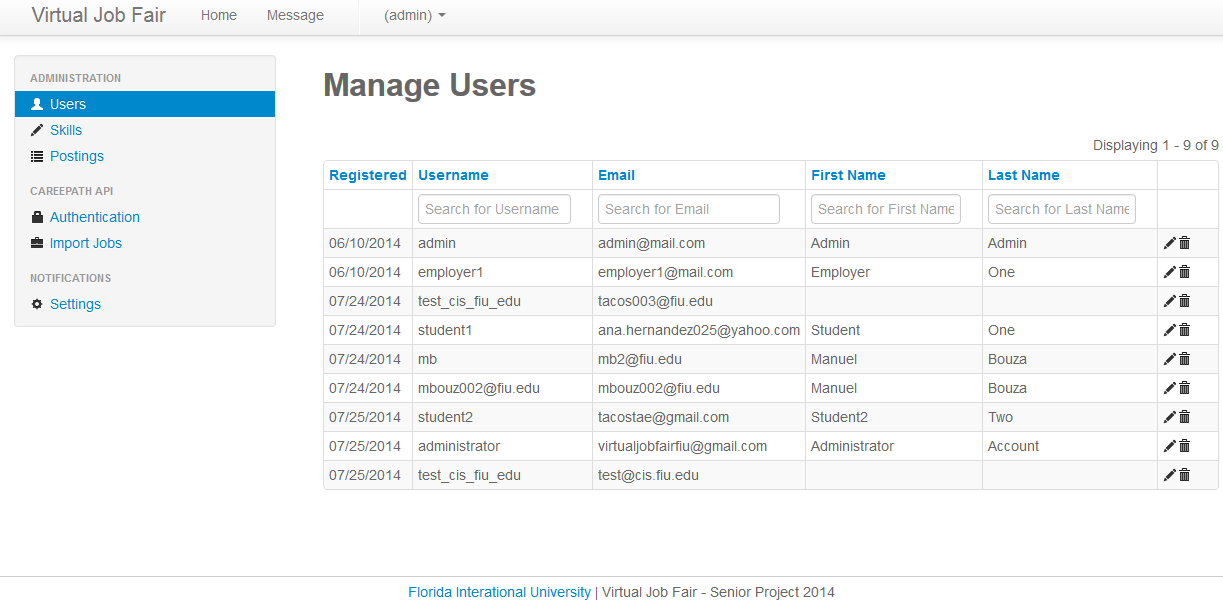
**Figure 9.3-17.1 Employer Email Notification based on Skills Matching per Job Posting**



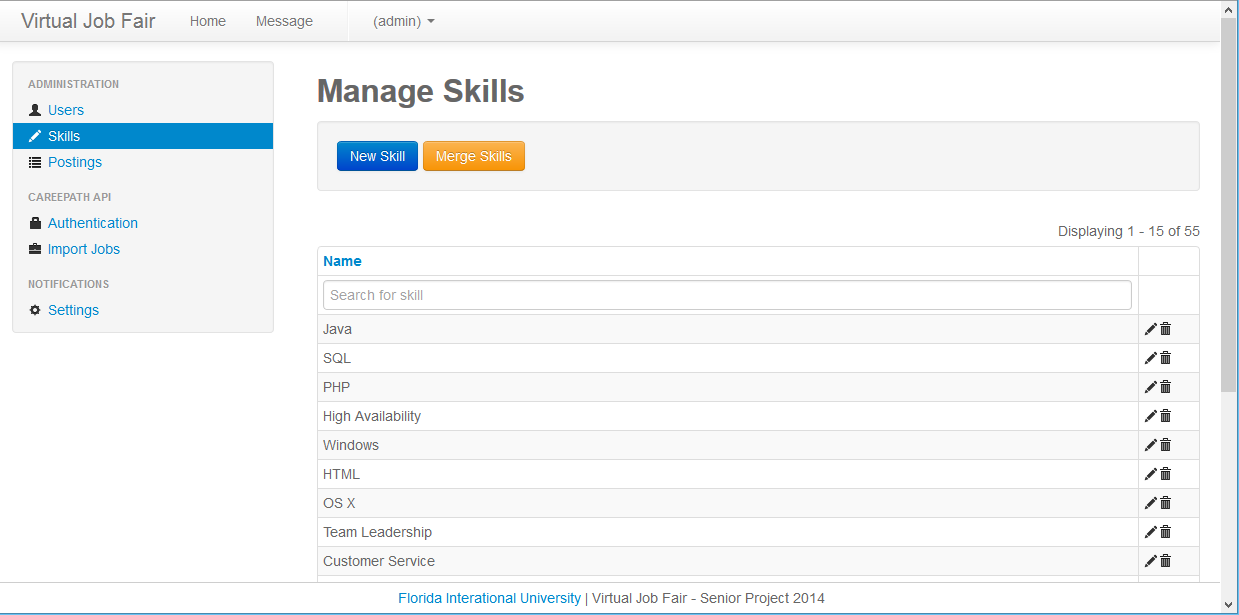
**Figure 9.3-17.2 Employer Email Notification based on Skills Matching per Job Posting**



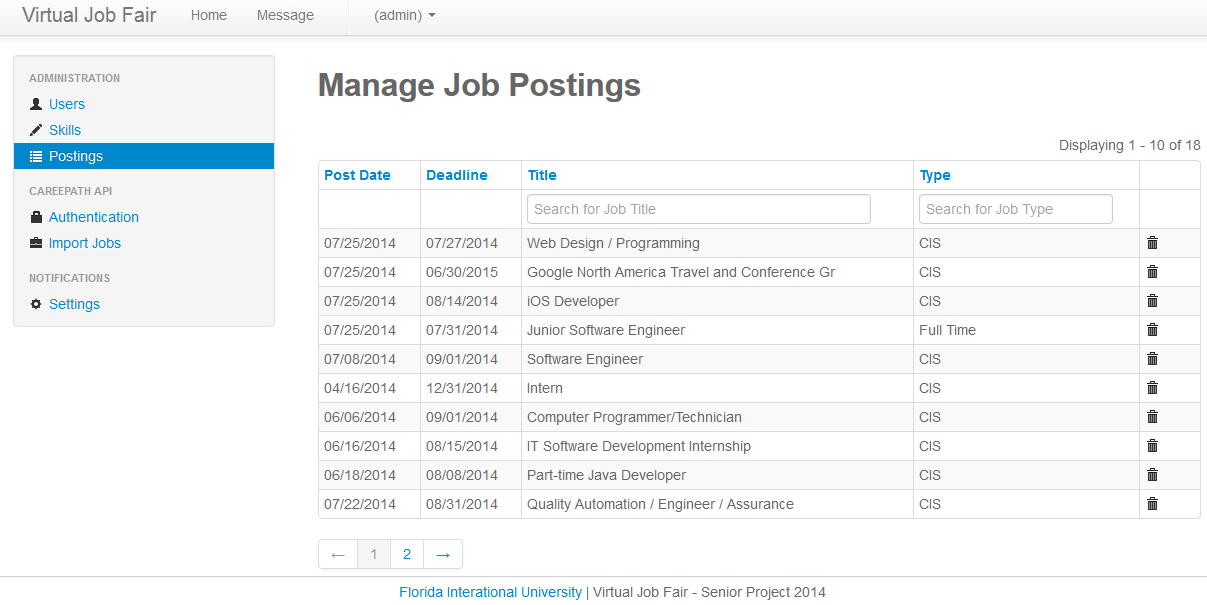
**Figure 9.3-18 Admin Import Jobs**



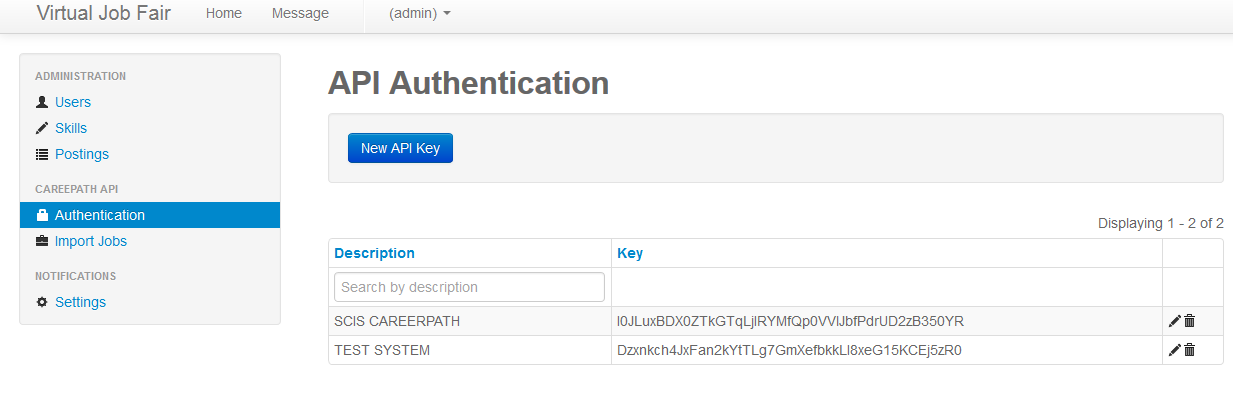
**Figure 9.3-19 Admin Manage Users**



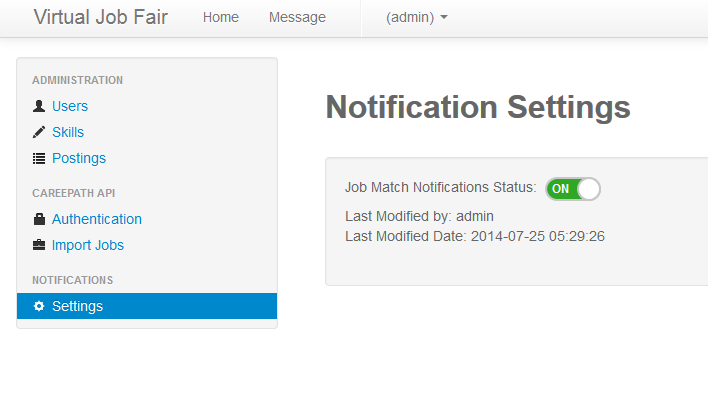
**Figure 9.3-20 Admin Manage Skills**



**Figure 9.3-21 Admin Manage Job Postings**



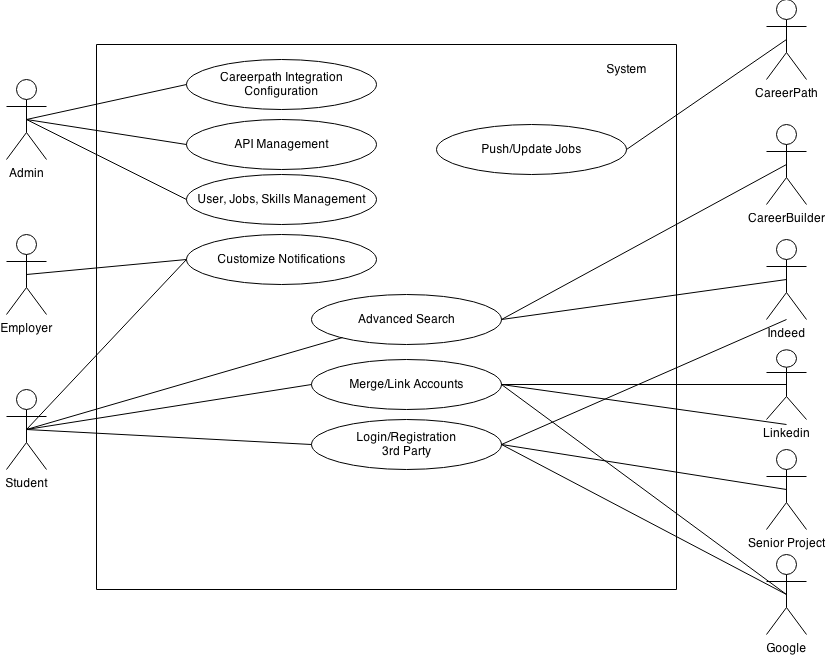
**Figure 9.3-22 Admin API Authentication**



**Figure 9.3-23 Admin Notification Settings**

**9.4 Appendix D - Analysis Models**

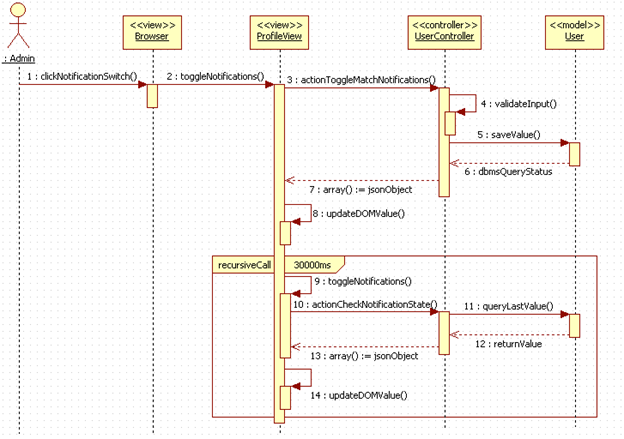
Use Case Diagram with main functionalities.



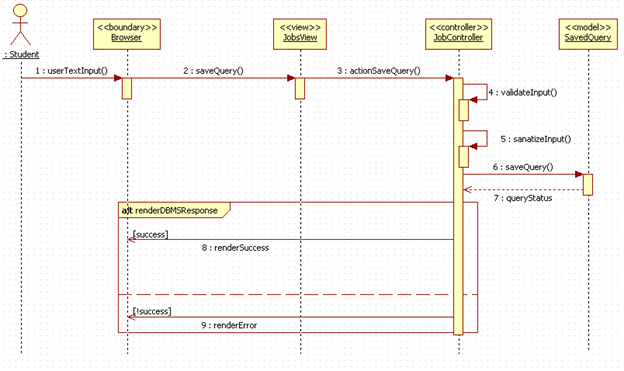
**Figure 9.4 Use Case Diagram**

**New System’s Sequence Diagrams**

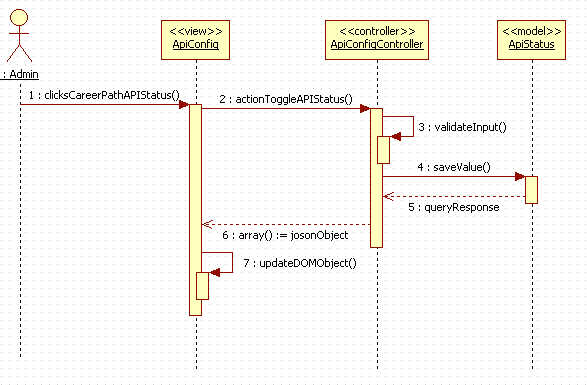
**Automated Notification - Admin Enable Notification**



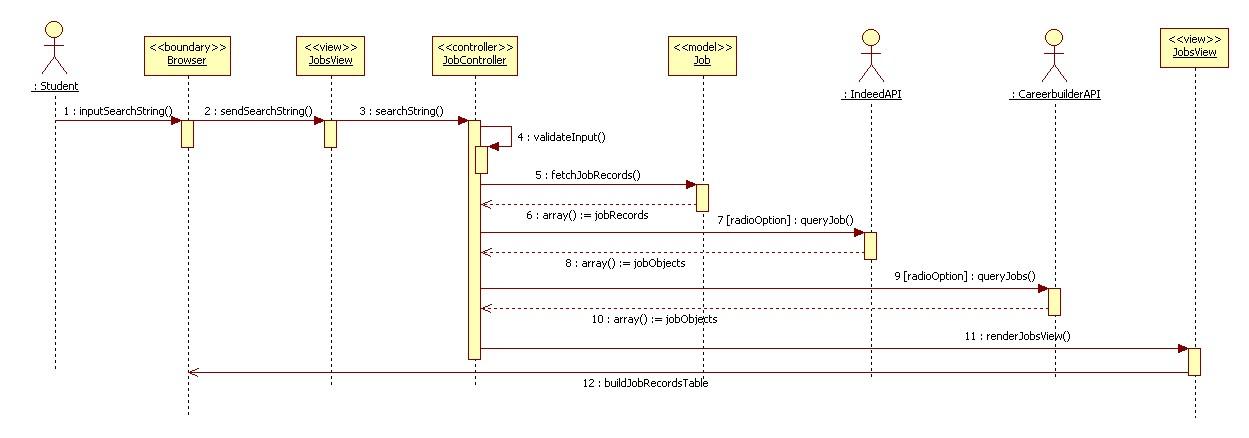
**Search Subsystem - Save Query**



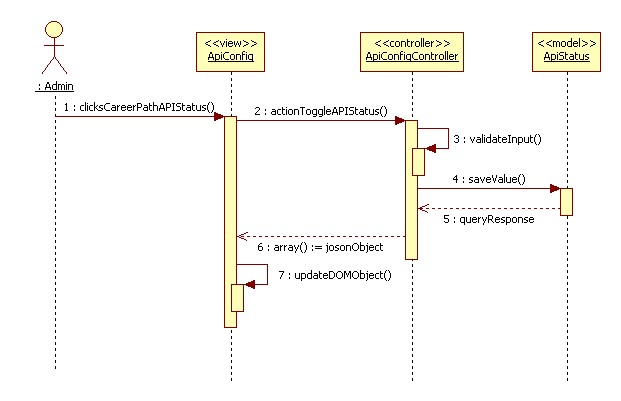
**API subsystem - Import Jobs**



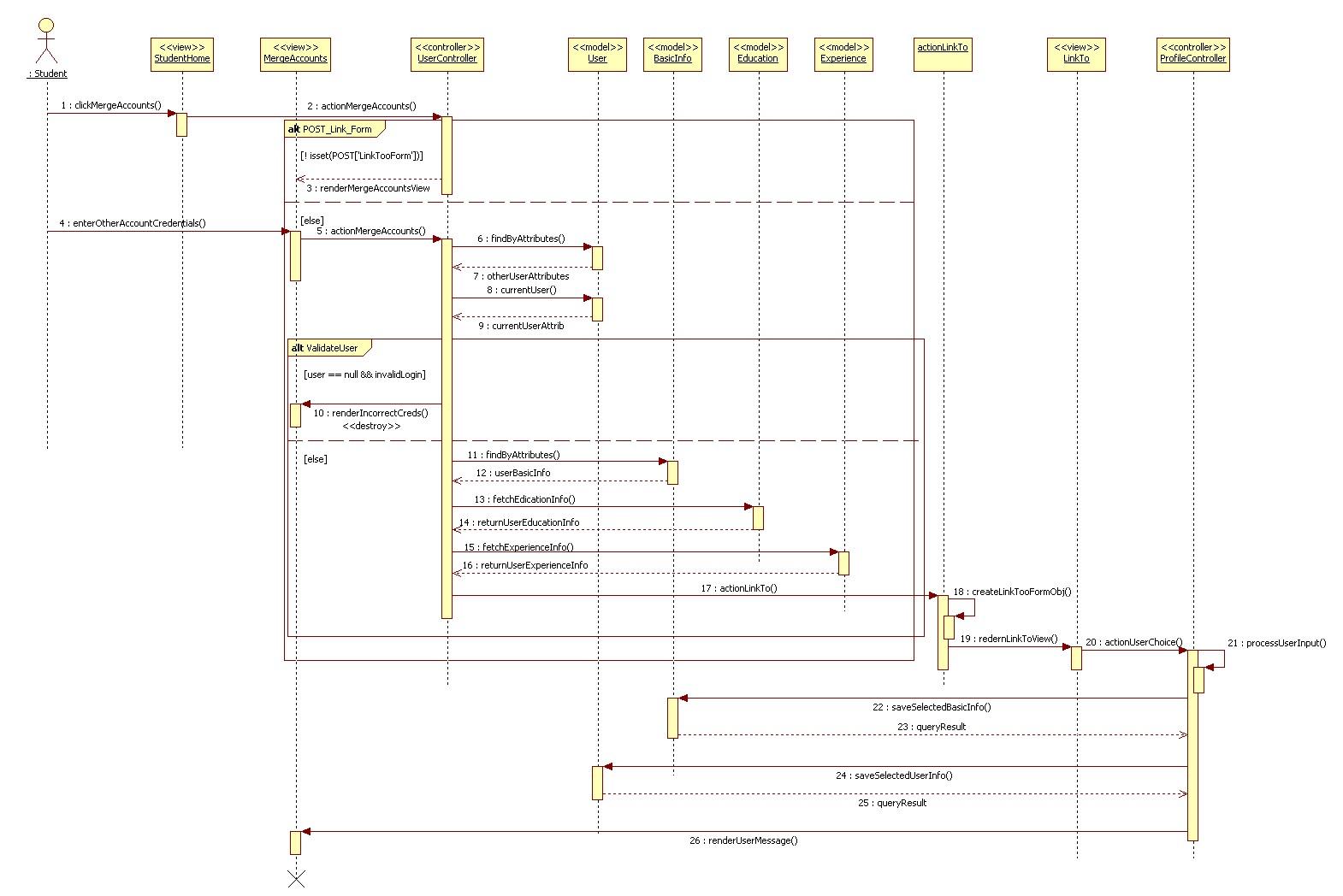
**Search Subsystem – Advance search**



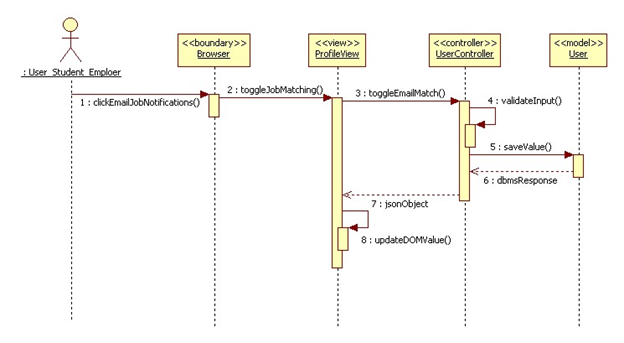
**API Subsystem – Disable API Querying**



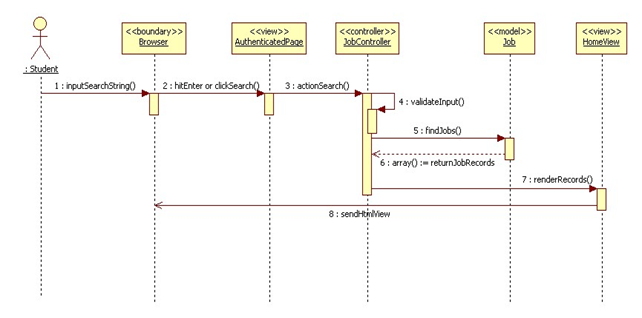
**Account – Account Merge Action**



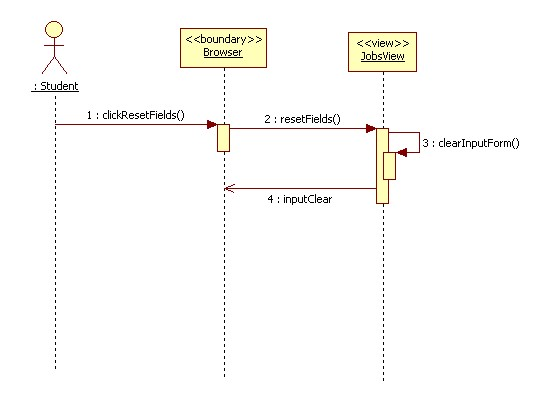
**Automated Notification – Set Notifications On/Off**



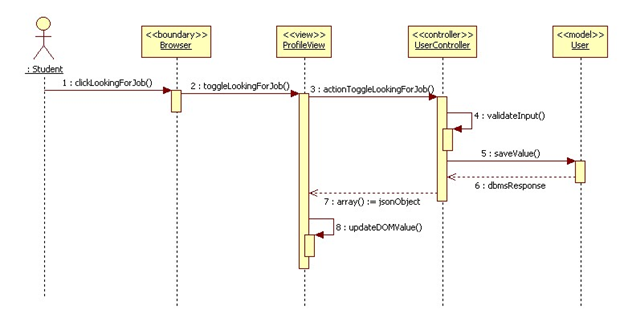
**Search Subsystem – Navigation bar Search**



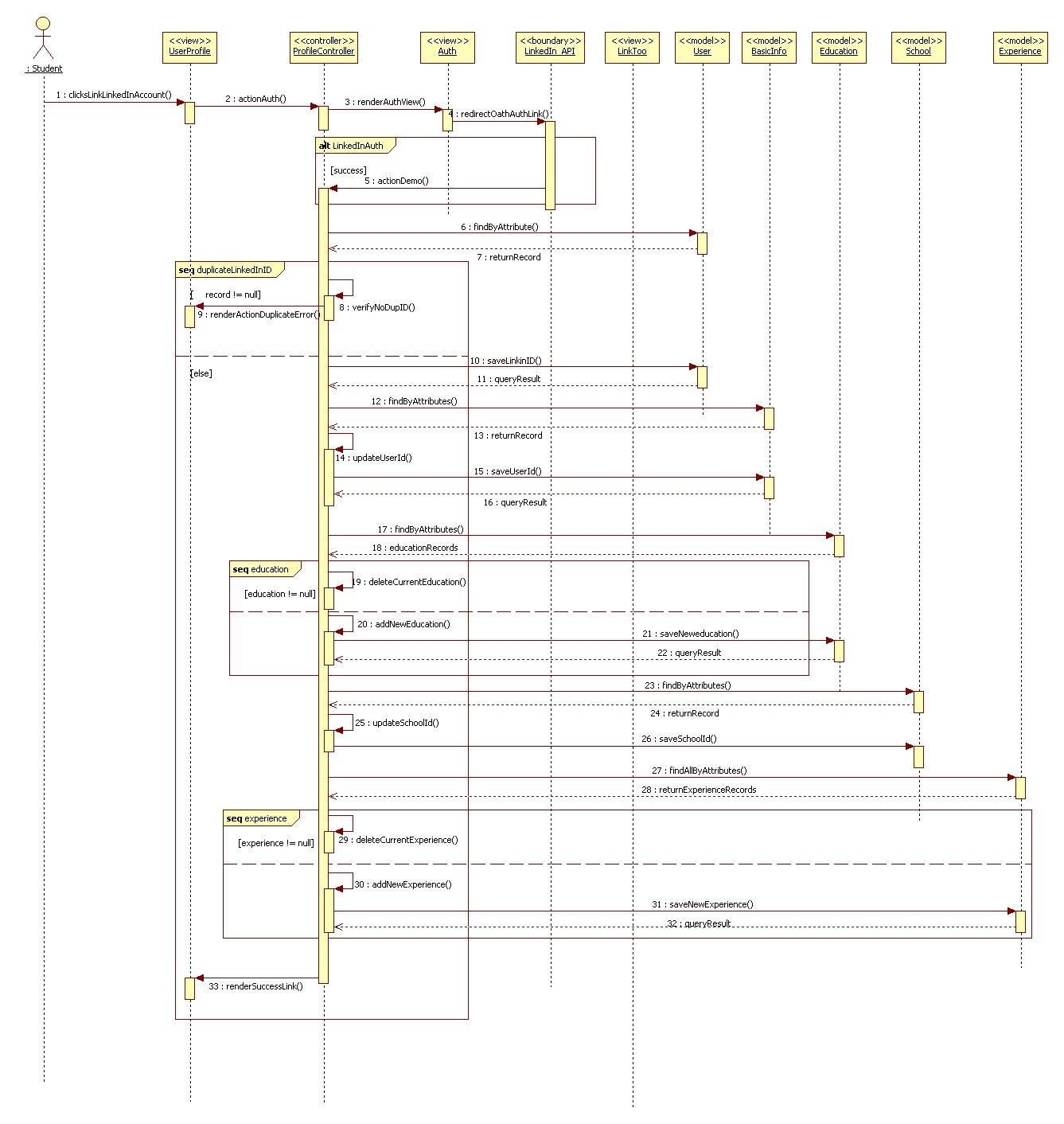
**Search Subsystem – Reset Advance Search Inputs**



**Search Subsystem –Set Job Search Status**

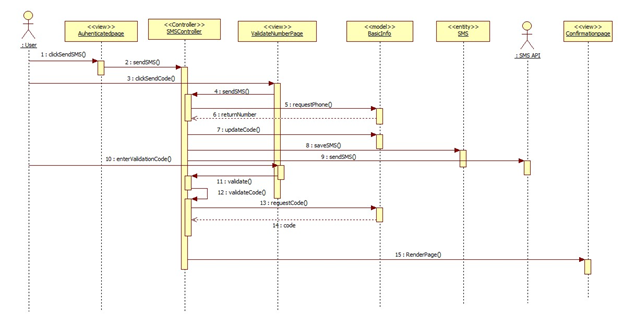


**Link Account Subsystem - Linking Account**

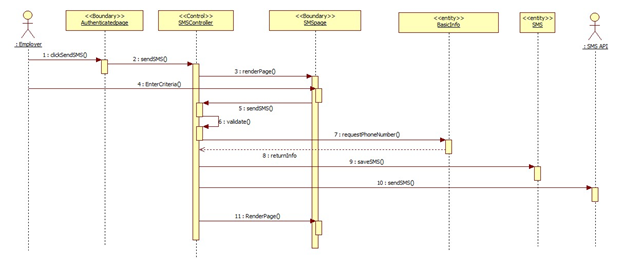


**Current System’s Sequence Diagrams**

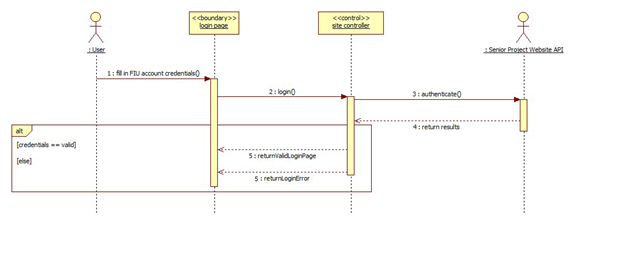
**Confirm Phone Number**



**Send SMS**

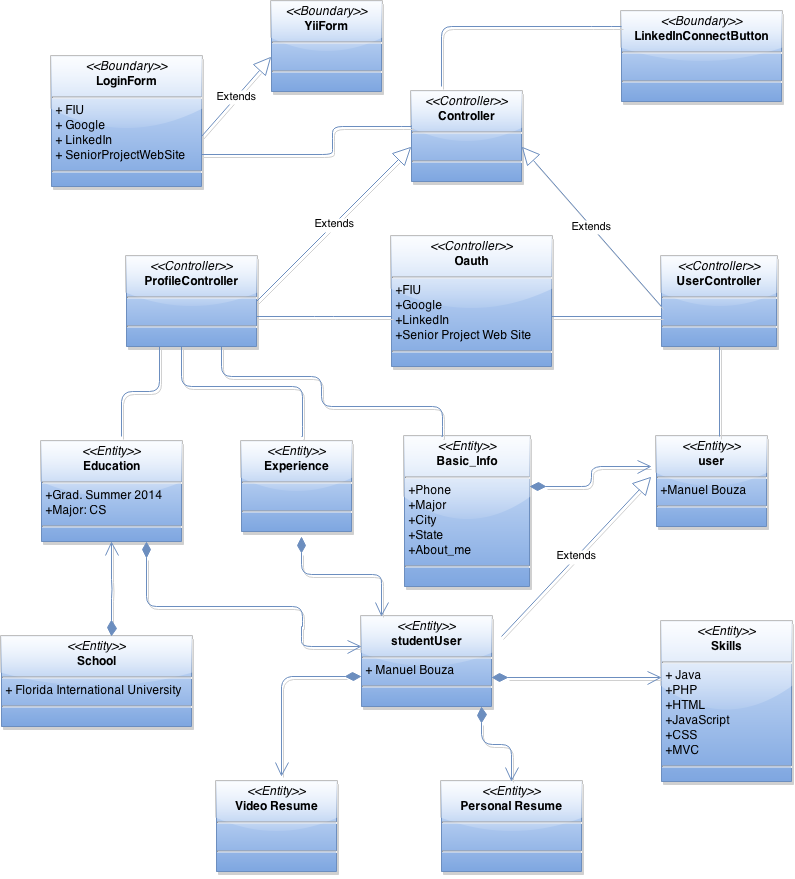


**Login with FIU credentials**

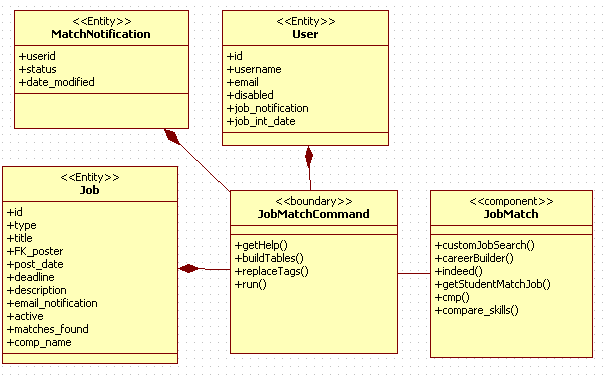


* 1. **Appendix E - Design Models**

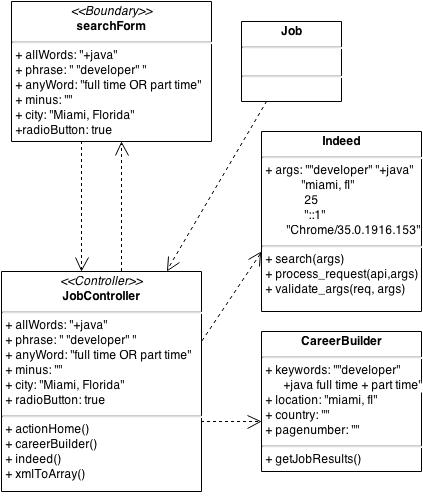
**Third-party login Object Design**

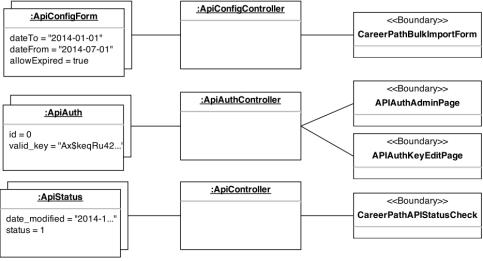
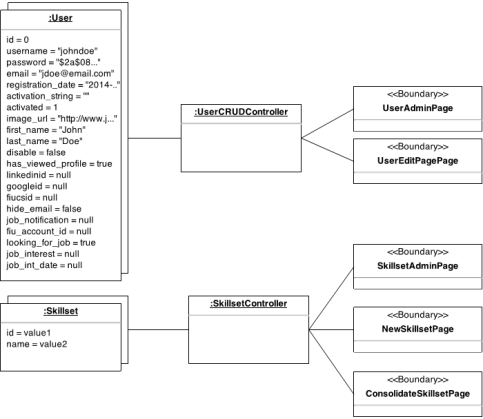


**Notifications Sub-system Object Design**



**Advanced Job Search Object Design**



**API Subsystem Object Design**

## 9.6 Appendix F - Documented Code

**Automated Notification Subsystem**

/\*

\* $type String

\* $ar Array

\* $interval Integer

\*/

public function buildTable($type, $ar, $interval)

/\*

\* $query String

\* $city String

\*/

public function customJobSearch($query = null, $city = null)

**Merge Subsystem**

public function actionMergeAccounts()

public function actionFiuAuth()

public function actionGoogleAuth()

public function actionFiuCsSeniorAuth()

public function actionDemo() //LinkedIn

**Search**

/\*

\* $allWords String

\* $phrase String

\* $anyWord String

\* $minus String

\* $city String

\* $tagName String

\*/

public function actionHome($allWords = null, $phrase = null, $anyWord = null, $minus = null, $radioOption = null, $city = null)

/\*

\* $allWords String

\* $phrase String

\* $anyWord String

\* $minus String

\* $city String

\* $tagName String

\*/

public function actionSaveQuery($allWords = null, $phrase = null, $anyWord = null, $minus = null, $city = null, $tagName = null)

**API**

public function actionPost()

/\*

\* $startDate String

\* $endDate String

\* $allowExpired Boolean

\*/

protected function careerPathSync($startDate, $endDate, $allowExpired)

## 9.7 Appendix G - Diary of Meeting and Tasks

During the duration of the project, our team did daily conference call stand ups with project manager, Rolando Vicaria to ensure adequate progress and discuss any problems we had encounter, as well as to receive feedback from our project manager.

**Meeting 1:**

**Date:** May 13, 2014

**Start Time:** 6:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Read documentation, and get familiar with the code.

**Meeting 2:**

**Date:** May 15, 2014

**Start Time:** 7:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Set up of the local environment.

**Meeting 3:**

**Date:** May 18, 2014

**Start Time:** 2:00PM

**End Time:** 4:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Re-Install IDEs for local environment. Setup BitBucket for code control.

**Assigned Tasks:**

* **Manuel B. :** Install PHPStorm, configure HTTP server, install local server WAMP, setup MySQL Workbench, setup git.
* **Ana H. :** Install PHPStorm, configure Apache, setup mysql command line interface, setup git.
* **Enio P. :** Configure IntelliJ, configure HTTP server, configure local mysql server, setup git.
* **Tomas A. :** Configure HTTP server, configure local mysql server, setup git.

**Meeting 4:**

**Date:** May 20, 2014

**Start Time:** 7:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Move bitbucket repo to Gitlab, and set up local branches.

**Meeting 5:**

**Date:** May 22, 2014

**Start Time:** 7:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta, Rolando Vicaria.

**Agenda:** Create tentative schedule and requirements.

**Assigned Tasks:**

* **Manuel B. :** Fix broken links in application.
* **Ana H. :** Remove search result page and research generalize search by keyword.
* **Enio P. :** Research existing SCIS job posting system.
* **Tomas A. :** Research existing notifications.

**Meeting 6:**

**Date:** June 3, 2014

**Start Time:** 7:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta

**Agenda:** Check local branch integration with development branch and admin to production mode.

**Meeting 7:**

**Date:** June 11, 2014

**Start Time:** 6:00PM

**End Time:** 8:30PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta

**Agenda:** Automate deployment of MySQL DB schema. Evaluate using Yii migrations if not design another method.

**Meeting 8:**

**Date:** June 17, 2014

**Start Time:** 6:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta

**Agenda:** PerformIntegrationtesting, and document problems.

**Meeting 9:**

**Date:** June 20, 2014

**Start Time:** 6:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Rolando Vicaria.

**Agenda:** Arrange AgileZen work schedule based on priorities discussed during class presentation.

**Meeting 10:**

**Date:** June 22, 2014

**Start Time:** 2:00PM

**End Time:** 4:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena.

**Agenda:** Discuss requirement progress, problems, and tentative schedule for the remaining requirements.

**Meeting 11:**

**Date:** June 27, 2014

**Start Time:** 6:00PM

**End Time:** 9:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Discuss development repo problems, and agreed on approaches to avoid further problems with the development branch.

**Meeting 12:**

**Date:** July 3, 2014

**Start Time:** 5:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** PerformIntegrationtesting, and document problems. Discuss development progress, and updated AgileZen.

**Meeting 13:**

**Date:** July 10, 2014

**Start Time:** 6:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Discuss local branches integration issues, work on current requirements, and do debugging.

**Meeting 14:**

**Date:** July 13, 2014

**Start Time:** 3:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta, Rolando Vicaria.

**Agenda:** Work on finalization of requirements, integration testing, and poster presentation.

**Meeting 15:**

**Date:** July 17, 2014

**Start Time:** 6:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Discuss final requirements progress, and uploaded documents to Google Drive to be able to share the work more efficiently.

**Meeting 16:**

**Date:** July 18, 2014

**Start Time:** 6:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, and final powerpoint presentation.

**Meeting 17:**

**Date:** July 19, 2014

**Start Time:** 12:00PM

**End Time:** 8:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, and final powerpoint presentation.

**Meeting 18:**

**Date:** July 20, 2014

**Start Time:** 12:00PM

**End Time:** 10:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, and final powerpoint presentation.

**Meeting 19:**

**Date:** July 21, 2014

**Start Time:** 4:00PM

**End Time:** 11:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, finalization of requirements.

**Meeting 19:**

**Date:** July 22, 2014

**Start Time:** 10:00AM

**End Time:** 2:30PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, finalization of requirements.

**Meeting 20:**

**Date:** July 23, 2014

**Start Time:** 10:00AM

**End Time:** 11:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, finalization of requirements, fixing powerpoint presentation, poster.

**Meeting 21:**

**Date:** July 24, 2014

**Start Time:** 10:00AM

**End Time:** 11:00PM

**In Attendance:** Manuel Bouza, Ana Hernandez, Enio Pena, Tomas Acosta.

**Agenda:** Work on final documentation, finalization of requirements, fixing powerpoint presentation, poster. Rehearsal of final presentation.

## 10. References

1. Feasibility Study & Project Plan Document v1.0. Fall 2013.
2. Feasibility Study & Project Plan Document v2.0. Spring 2013.
3. Requirement Document v1.0. Fall 2013.
4. Requirement Document v2.0. Spring 2013.
5. Design Document v1.0. Fall 2013.
6. Design Document v2.0. Spring 2013.
7. Final Deliverable v1.0. Fall 2013.
8. Final Deliverable v2.0. Spring 2013.
9. "Bootstrap." *2.2.2 Documentation*. N.p., n.d. Web. 24 July 2014. <http://bootstrapdocs.com/v2.2.2/docs/>.
10. "Documentation." *Yii PHP Framework: Best for Web 2.0 Development*. N.p., n.d. Web. 24 July 2014. <http://www.yiiframework.com/doc/>
11. Chacon, Scott. *Pro Git*. Berkeley, CA: Apress, 2009. Online. <http://git-scm.com/book/en/Getting-Started>