**Design Document**

CIS 4911 – Senior Project U01

Virtual Job Fair 3.0

**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.

**ABSTRACT**

The Design Document gives an insight into the system structure of the Virtual Job Fair 3.0 project. By describing system architecture, subsystem decomposition, design methodology, hardware and software mapping, persistent data management and security/privacy gives the reader a better idea of the process that was used to design Virtual Job Fair 3.0. Chapter 1 introduces basic information about the project, such as problem definition, definitions, acronyms and overview of document. Chapter 2 introduces the structure of the system by describing its system design and subsystem decomposition.

Chapter 3 delves into to the detail design, including static models and dynamics models plus code specification with class interfaces. Then, a glossary is introduced with domain-specific terms. Then, finally, the appendixes contain miscellaneous information, such as use case diagrams, use cases being implemented, and document class interfaces and diary of meeting and tasks.

Contents

[1.      Introduction 5](#_Toc374337944)

[1.1.   Problem definition 5](#_Toc374337945)

[1.2.   Design methodology used 6](#_Toc374337946)

[1.3.   Definitions, acronyms, and abbreviations. 7](#_Toc374337947)

[1.4.   Overview of document 8](#_Toc374337948)

[2.      System Design (i.e., overall system design) 9](#_Toc374337949)

[2.1.   Overview 9](#_Toc374337950)

[2.2.   Subsystem Decomposition 10](#_Toc374337951)

[2.3.   Hardware and Software Mapping 14](#_Toc374337952)

[2.4.   Persistent Data Management– 15](#_Toc374337953)

[2.5.   Security/Privacy 35](#_Toc374337954)

[3.      Detailed Design 37](#_Toc374337955)

[3.1.   Overview 37](#_Toc374337956)

[3.2.   Static model 38](#_Toc374337957)

[3.3.   Dynamic model 42](#_Toc374337958)

[5.      Appendix 51](#_Toc374337959)

[5.2.   Appendix B 53](#_Toc374337960)

[5.3.   Appendix C 95](#_Toc374337961)

[5.4.   Appendix D - Diary of meeting and tasks. 96](#_Toc374337962)

[6. References 105](#_Toc374337963)

# 1.      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

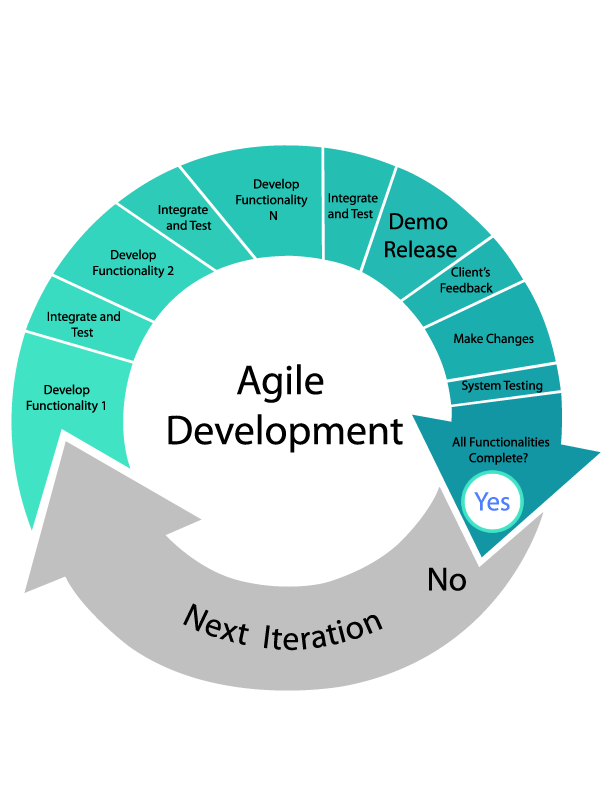
## Employers looking for talent are always interested in filling out positions with the best possible candidates. In order to accomplish this task, the most effective method to date is to tap local talent, whether it is at universities or job fairs. Given the increasing globalization trend, and the fact 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. This way the students do not need to go looking in different sites, he or she can find everything on our site. Also students will receive notification of jobs matching his/her saved search criteria, so that way the students will know as soon as an opportunity is available. Therefor providing the employer and potential employee a more easy and friendly way to find their match.

## 1.2.   Design methodology used

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 1.2.1 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.

## 1.3.   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.

## 1.4.   Overview of document

In chapter 1, the main problem is introduced, along with the design methodology used for the project, definitions, acronyms and abbreviations. In chapter 2, the system is introduced in terms of system architecture, with its subsystem decomposition, hardware and software mapping, persistent data management and security/privacy aspects explained. In chapter 3, the behavior of each subsystem is described, and the static models and dynamic models used are explained.

In chapter 4, a glossary with domain-specific terms is introduced. In the appendix, miscellaneous material, such as use case diagrams, use cases being implemented and documented class interfaces can be found. Finally, a diary of meetings and references can be found at the end of the document.

# 2.      System Design (i.e., overall 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.

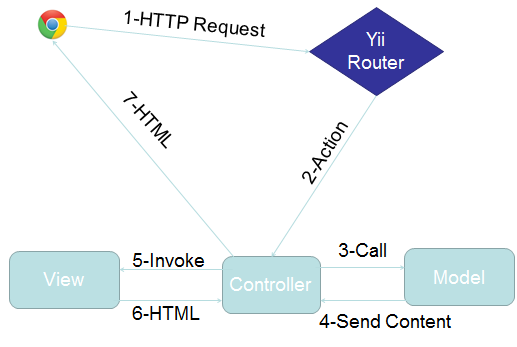
The new system shall…

* 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 an e-mail that is already in the system.
* Allow students to build their profile (basic information, skills, education, and 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

2.1 Overview

The previous team implemented Virtual Job Fair 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 2.1.1 MVC** **Yii Framework**

2.2.   Subsystem Decomposition

Below is the subsystems decomposition found in the 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

**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 our 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

**SMS message subsystem**

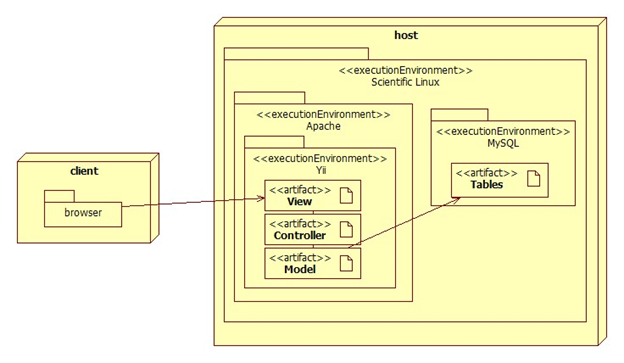
This subsystem gives the ability to send messages to students as well as allow the users to receive reminders about upcoming interviews directly in their SMS enabled phones. The SMS messaging subsystem is designed to allow this kind of functionality which can keep the users connected and active with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

## 2.3.   Hardware and Software Mapping

The previous team developed a deployment diagram which indicates the mapping of hardware and software. For our Virtual Job Fair 3.0, we are still 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. Therefore, the description below was that done by the previous team, and remains valid for Virtual Job Fair 3.0.



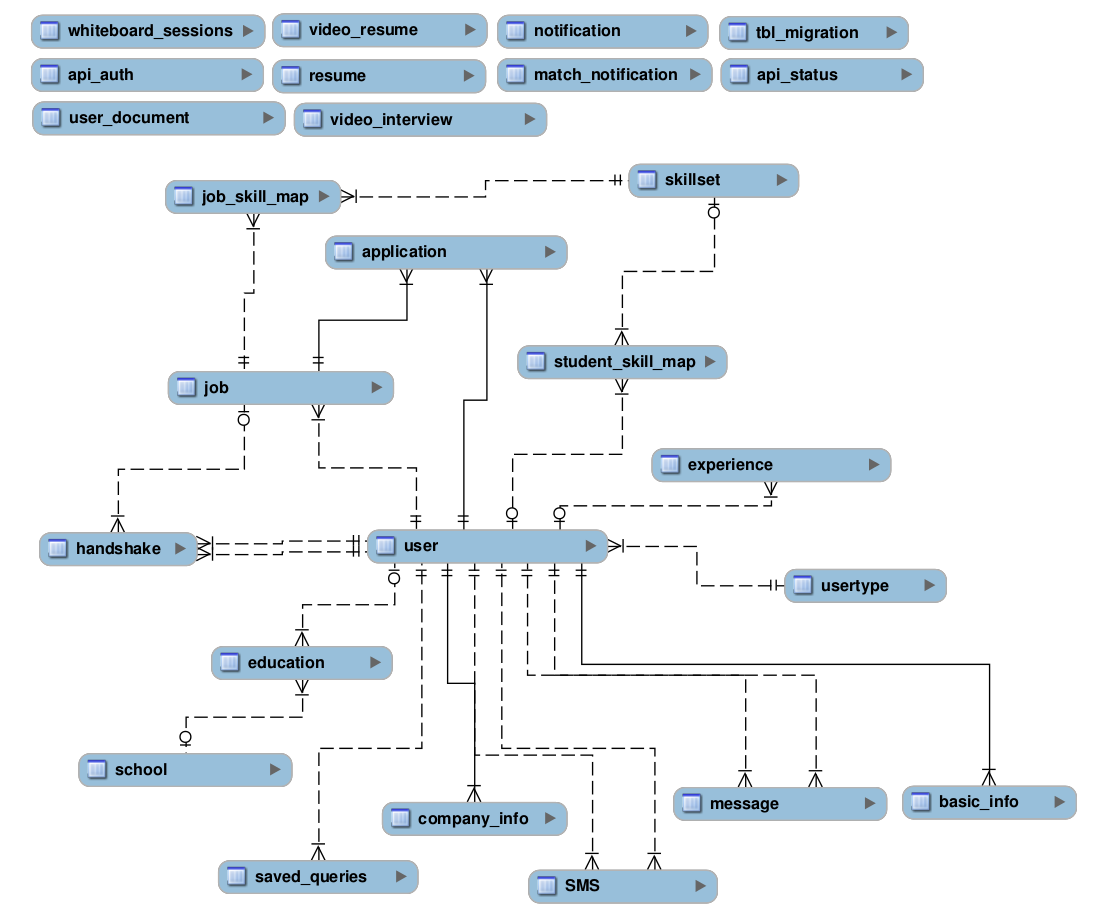
**Figure 2.3.1 Hardware and Software**

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.

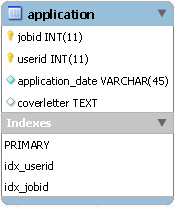
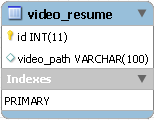
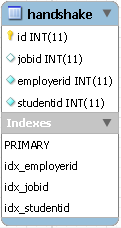
2.4.   Persistent Data Management

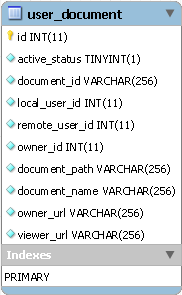
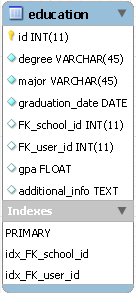
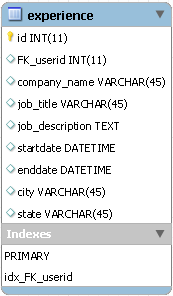
The previous group completed an ER diagram. Since we are adding new tables and columns into existing tables and database, respectively, below is the updated version of the ER diagram of the original group, which includes our new data, which is identified by a yellow background and dark red text.

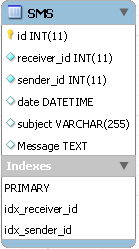
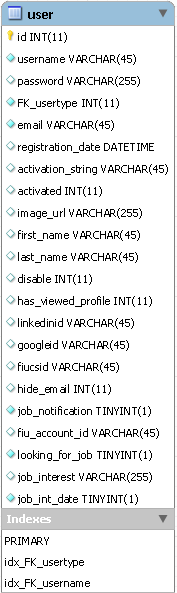
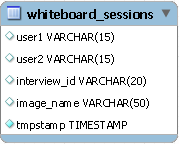
* **ER Diagram**

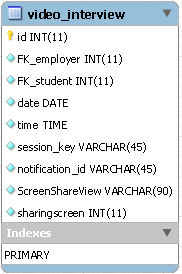
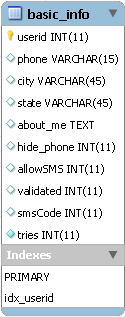
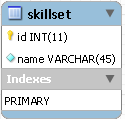


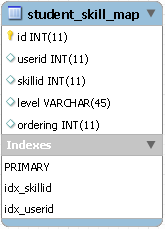
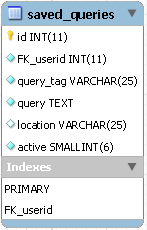
* **Database tables**

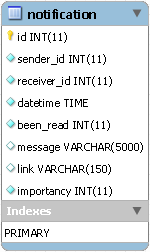
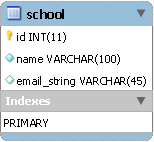
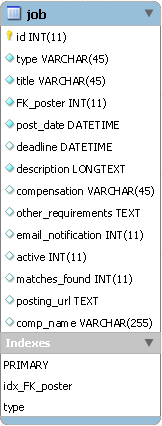
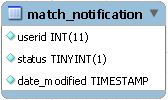
****  

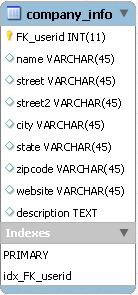
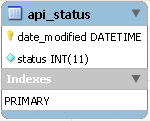
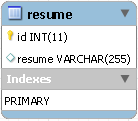
  

2.5.   Security/Privacy

The previous team defined security and privacy features for log in, 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 student’s profiles, since it may contain information which should not be shared, such as phone number or email.

# 3.      Detailed Design

The detailed design chapter introduces the system in terms of subsystems and the relationships among them. Initially, the system is decomposed into subsystem, with each subsystem described in terms of behavior and structure. Then, the static model is introduced in terms of subsystems with descriptions for each. After that, the dynamic model is presented in terms of state machine diagrams with the main control object for each subsystem. Finally, class interfaces and constraints for the main control object in each subsystem are presented.

3.1.   Overview

**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.

**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.

**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.

**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.

**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.

**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.

**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.

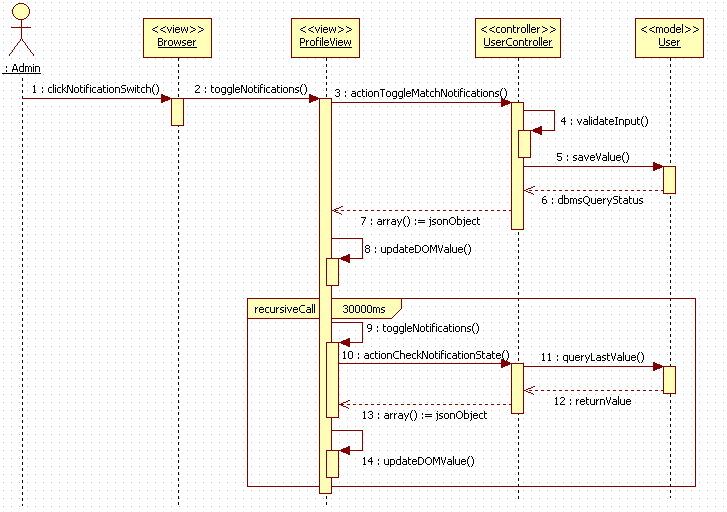
All of this is possible by using the Twilio cloud communications service. Twilio is a company that provides services such as SMS sending and receiving, speech and text recognition, conference calling etc.  The most compelling feature is that an extensive and well documented API is provided to developers,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

3.2.   Static model

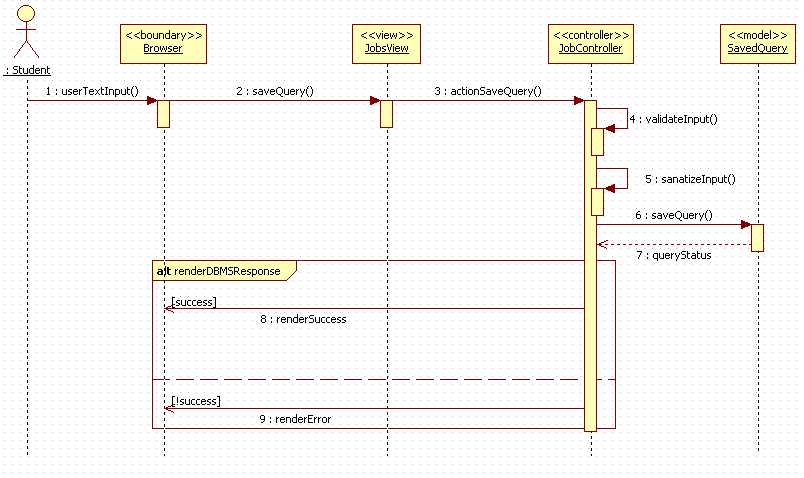
3.3.   Dynamic model

In the following section, are the main sequence diagrams for the above subsystem:

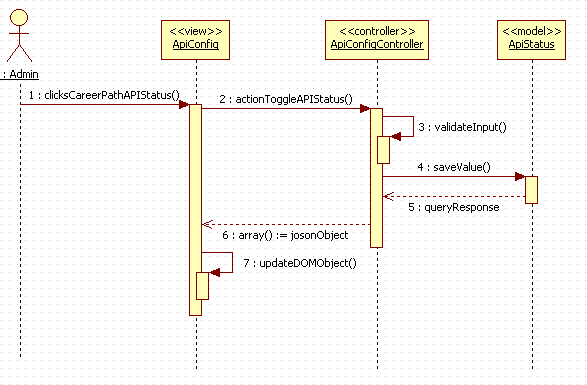
**Automated Notification - Admin Enable Notification**

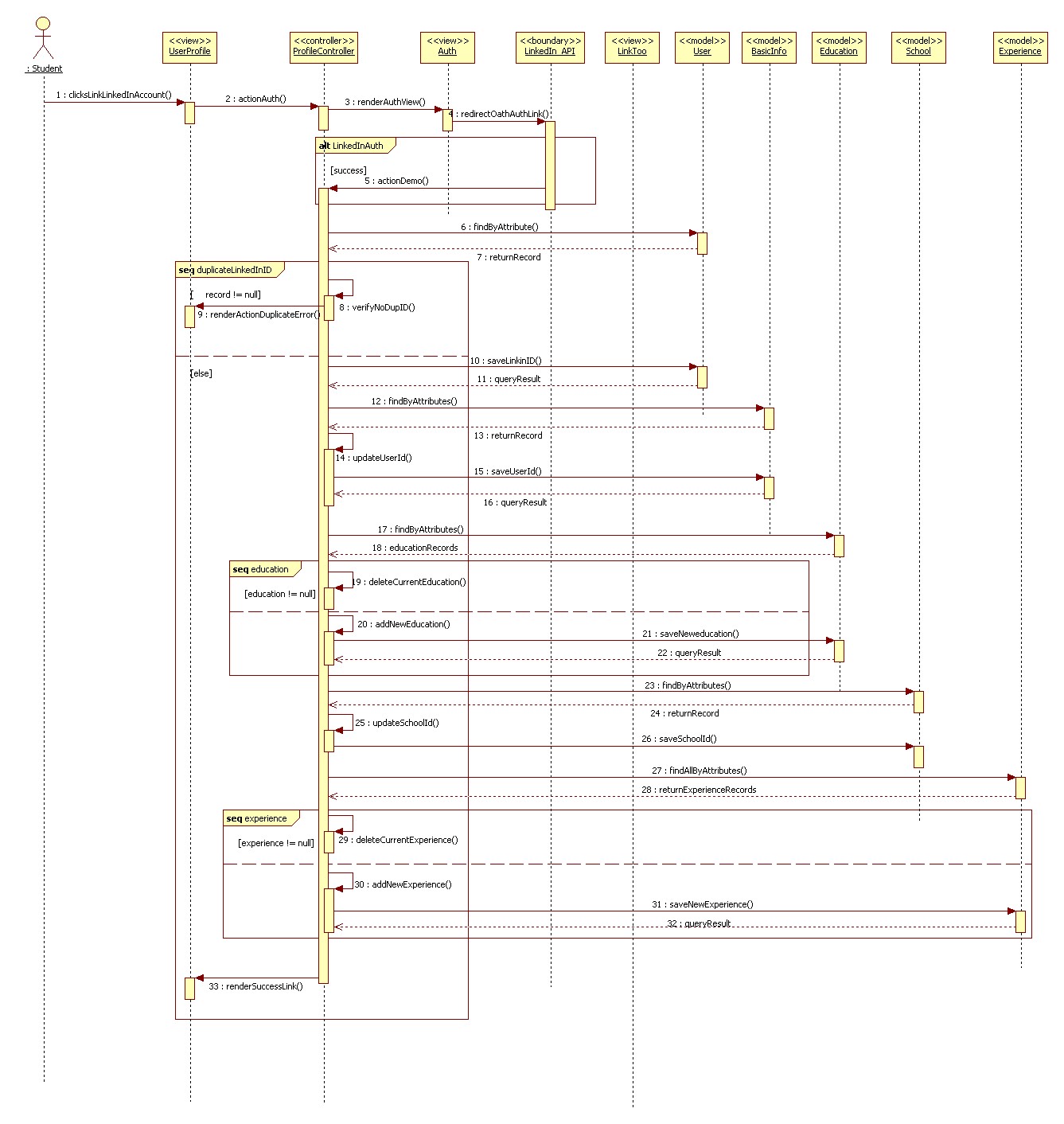
****

**Search Subsystem - Save Query**

****

**API subsystem - Import Jobs**

****

**Link Account Subsystem - Linking Account**

3.4.   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 C.

**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**

**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**

**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.

4. Glossary

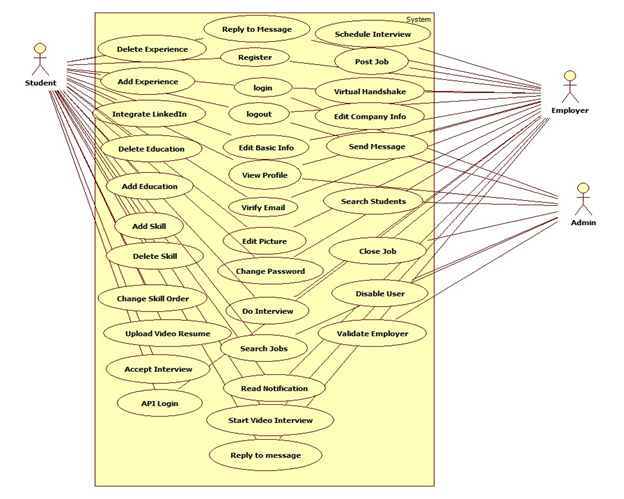
|  |  |
| --- | --- |
| Term | Meaning |
| Class Diagram | A pictorial representation of all the classes in the system |
| Functional Requirement | A function supported by the system, where a function is a set of inputs, the behavior, and outputs. |
| Non-Functional Requirement | A requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. |
| Object Diagram | A pictorial representation of an instance of a class with example of how the data of the class will be populated |
| Sequence Diagram: | A pictorial representation of how processes operate with one another and the user during the course of a specific piece of functionality. |
| Use Case | List of steps defining the interaction between the user and the system to achieve a goal |

* **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

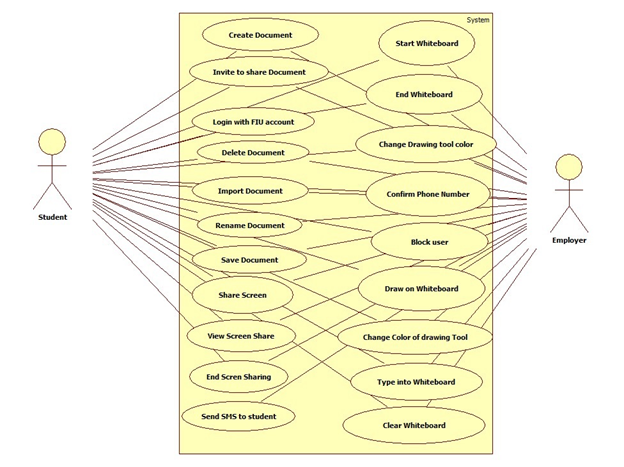
# 5.      Appendix

5.1.   Appendix A - Use case diagram for use cases being implemented.

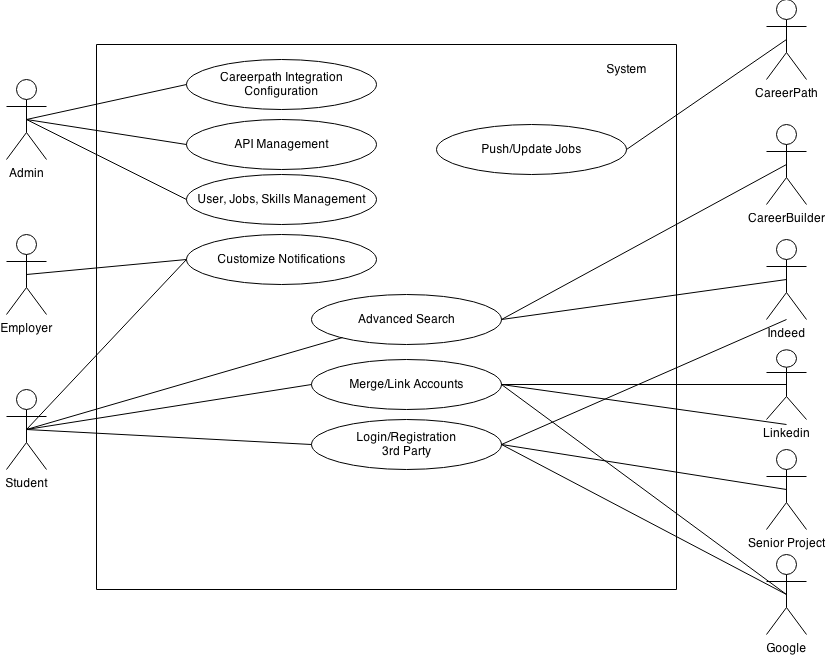
**Spring 2013 Use Case Diagram**



**Current System’s Use Case Diagram**



**New System’s Use Case Diagram**

****

**Figure 6.2.1 Use Case Diagram**

5.2.   Appendix B

* **New system’s use cases**

|  |  |
| --- | --- |
| 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 system notifications |
| Actor | Admin |
| Pre-conditions | 1. User must be logged in as Admin. 2. User must be in the Settings Page. |
| Steps | User clicks on the 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-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-067 Merge 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 | 1. User fails to provide an username enter the wrong username 2. User fails to provide a password or enter the wrong password |

* **Current system’s use cases**

|  |  |
| --- | --- |
| 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-0020 Start Video Interview |
| Description | Start Video Interview |
| Actor | Employer, Student |
| Pre-conditions | 1. Actors are logged in 2. A video interview has been previously scheduled 3. A notification for the video interview was sent to both parties and is displaying in the homepage 4. Actors are in homepage |
| Steps | 1. Actor clicks on link to video interview in the notifications window in homepage    1. Notification for employer: ([You schedule interview with Diego on 2013-03-13 at 3:00pm](http://srprog-spr13-01.aul.fiu.edu:8080/demos/videointerview.html?view=Diego&notificationRead=842&usertype=2) )    2. Notification for Student: ([Company IBM wants to have a video interview with you](http://srprog-spr13-01.aul.fiu.edu:8080/demos/videointerview.html?view=IBM) [2013-03-13 at 3:00pm](http://srprog-spr13-01.aul.fiu.edu:8080/demos/videointerview.html?view=Diego&notificationRead=842&usertype=2) .[Good Luck!](http://srprog-spr13-01.aul.fiu.edu:8080/demos/videointerview.html?view=IBM) ) 2. Actor is redirected to the video interview page where he/she sees to the left the video connection tools and the video window, and to the right: 3. For the employer:    1. the profile of the student participating in the interview. 4. For the student:    1. the profile of the employer conducting the interview. 5. Once the employer is ready to start the interview he/she clicks in the connect button which allows the student participating in the interview to connect right after 6. Once both parties are connect the video interview is started 7. Once the interview is finish, the employer and student clicks the finish button to be disconnected |
| Post-conditions | Both parties participated in a video interview |
| Exceptions | Actor ends the interview before it is finished |

|  |  |
| --- | --- |
| Use Case ID | VJF-0021 Accept Interview |
| Description | Student user accepts a video interview |
| Actor | Student |
| Pre-conditions | Actor is logged in and is at the home page |
| Steps | 1. Actor clicks on the video interview notification section. 2. Actor is notified that a video interview has been schedule for him 3. Actor is asked to confirm the video interview 4. Actor clicks the accept button |
| Post-conditions | The video interview is confirmed and a notification is sent back to the user who originated the interview (employer user) |
| Exceptions | Actor does not accept the interview and closes the notification |

|  |  |
| --- | --- |
| 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-0030 Virtual Handshake |
| Description | Employer gives student a virtual handshake to show interest |
| Actor | Employer, Student |
| Pre-conditions | 1. Employer has posted a job |
| Steps | 1. After employer posts job, he is taken to a student match page 2. Employer can review the list of students who matches with the job he posted 3. Employer clicks on “virtual handshake” for any student |
| Post-conditions | 1. Employer remains on student match page 2. Student receives a notification that the employer has shown interest in him for the position |
| Exceptions | None |

|  |  |
| --- | --- |
| 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-0035 Administrator Close Job |
| Description | Close a job from further applications |
| Actor | Admin |
| Pre-conditions | 1. Admin is logged in |
| Steps | 1. Admin goes to home page 2. Admin enters text included in a job title 3. Admin is redirected to search results with a list of relevant jobs 4. Admin clicks on “delete” for a job of his choosing |
| Post-conditions | 1. The respective job is closed 2. Admin is redirected to search page |
| Exceptions | Search may not return any results |

|  |  |
| --- | --- |
| 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-0037 Apply to Job |
| Description | Student applies to an employers job posting |
| Actor | Student |
| Pre-conditions | Student is viewing a job |
| Steps | 1. Student clicks on apply 2. System displays a popup box 3. Student fills in a cover letter 4. Student clicks submit |
| Post-conditions | 1. System notifies employer of new application 2. User is redirected back to job page |
| Exceptions | Student has already applied for the job |
| 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-0039 Schedule interview |
| Description | Employer is schedule interview with the student |
| Actor | Employer |
| Pre-conditions | 1. Employer is logged in 2. Employer is on student profile view. |
| Steps | 1. Employer is clicking on the button video interview. 2. Employer is choosing the date and time. 3. Employer clicks submit |
| Post-conditions | 1. System is notifies the employer for a new schedule interview that he posted 2. System is notifies the student for a new schedule interview that he has been invited for. |
| Exceptions | employer type wrong input for time and date |
| 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-042 Share Screen |
| Description | Allow a user to share his screen |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in. 2. User is in the homepage 3. Interview has been scheduled. 4. Notification for the interview is displayed for both involved parties 5. Other user involved in interview is not sharing screen |
| Steps | 1. User clicks on scheduled interview notification 2. User is redirected to the interview page. 3. User clicks on share screen |
| Post-conditions | User is able to share his screen and database is update with required information. |
| Exceptions | User tries to share screen while the other party is sharing. |
| Use Case ID | VJF-043 View Screen Share |
| Description | Allow a user to see a shared screen |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in. 2. User is in the homepage 3. Interview has been scheduled. 4. Notification for the interview is displayed for both involved parties 5. Other user involved in interview is sharing a screen |
| Steps | 1.      User clicks on scheduled interview notification  2.      User is redirected to the interview page.  3.      User clicks on view screen share |
| Post-conditions | User is able to view screen shared by other party |
| Exceptions | The other user is not sharing a screen |

|  |  |
| --- | --- |
| Use Case ID | VJF-044 End Screen Sharing |
| Description | Allow a user to end current screen sharing |
| Actor | Student, Employer |
| Pre-conditions | 1.       User is logged in.   1. User is in the interview portal   3.      User is sharing screen |
| Steps | 1.      User clicks on end screen sharing |
| Post-conditions | 1.   User is not sharing screen  2.   System information is updated to allow other party to share screen  3.   Other party involved will not continue to see live feed from user screen |
| 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-046 Send interview reminder |
| Description | An interview reminder is sent to involved parties |
| Actor | Time |
| Pre-conditions | 1. An employer has set up an interview appointment 2. Interview starts in the next 30 minutes. |
| Steps | 1.      Database is continuously checked for interviews starting in the next 30 minutes  2.      After finding jobs that meet this criteria an email message is sent to involved parties about the event.  3.      If any of the accounts is set up to receive SMS then a text message will be sent as well. |
| Post-conditions | Parties involved in the interview will receive an email reminder and SMS according to set up permissions |
| Use Case ID | VJF-047 Confirm phone number |
| Description | Allows user to confirm and validate a phone number |
| Actor | Employee, Student |
| Pre-conditions | 1.      User entered a phone number in the system  2.      User is logged in.  3.      User is in the homepage  4. User phone has not been validated |
| Steps | 1.      User clicks on SMS page  2.      User is redirected to validate number page  3.      User clicks on validate phone  4.      An SMS message is sent to user’ phone  5.      User enters received code  6.      User presses validate button |
| Post-conditions | The system is updated to reflect phone validation |
| Exceptions | User enters a wrong authentication code |

|  |  |
| --- | --- |
| Use Case ID | VJF-048 Create new document |
| Description | Allow user to create a new document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on create new document button 3. The New document editing session is started |
| Post-conditions | 1. User is on the interview portal 2. A new document is displayed |
| Exceptions | 1. Connection Error |

|  |  |
| --- | --- |
| Use Case ID | VJF-049 Share active document |
| Description | Allow users to invite another user to a shared document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one document to share |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on share document button 3. Invitation is sent to other user 4. Other user receives notification 5. Other user joins the shared document session |
| Post-conditions | 1. User is on the interview portal 2. User is on the shared document session |
| Exceptions | 1. Connection Error 2. Other user not available |

|  |  |
| --- | --- |
| Use Case ID | VJF-050 Delete shared document |
| Description | Allow user to delete a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one document to delete |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on the manage documents button 3. User selects a document from documents list 4. User clicks on the delete document button 5. User is presented with a confirmation dialog 6. User confirms deletion of file 7. Document is deleted |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error 2. User did not select a document to delete |

|  |  |
| --- | --- |
| Use Case ID | VJF-051 Import document |
| Description | Allow user to import a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one document to import |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on import document button 3. User chooses file to import and drags it over the import document area 4. The document is imported into the system |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error 2. File is not a valid document |

|  |  |
| --- | --- |
| Use Case ID | VJF-052 Rename document |
| Description | Allow user to rename a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one document to rename |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on the manage documents button 3. User clicks on document to rename 4. User clicks on the rename document button 5. Rename dialog appears 6. User chooses new document name 7. The document name is changed |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error |

|  |  |
| --- | --- |
| Use Case ID | VJF-053 Save document |
| Description | Allow user to save a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one active document to save |
| Steps | 1. User clicks on save document link 2. The document is saved 3. User is notified |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error 2. There are no active documents to save |

|  |  |
| --- | --- |
| Use Case ID | VJF-054 Open document |
| Description | Allow user to open a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one active document to open |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on manage documents 3. User clicks on document to open 4. User clicks on the open document button 5. Document is loaded into the Editor |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error 2. There are no active documents to open |

|  |  |
| --- | --- |
| Use Case ID | VJF-055 Export document |
| Description | Allow user to export a document |
| Actor | Student, Employer |
| Pre-conditions | 1. User is logged in 2. User is on the interview portal 3. User has at least one active document to export |
| Steps | 1. User clicks on Collaborative Editor 2. User clicks on the manage documents button 3. User clicks on document to export 4. User clicks on export document 5. User receives the document as a file |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error 2. There are no active documents to export |

|  |  |
| --- | --- |
| 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 Case ID | VJF-057 Maintain document access boundaries |
| Description | Users accessing the system using their personal accounts will have no access to shared documents in the interview session |
| Actor | System |
| Pre-conditions | 3. User is logged in using their personal account |
| Steps | 1. User clicks on create new document link 2. User is informed of the restriction 3. User is required to click on the create temporary document editing session |
| Post-conditions | 1. User is on the interview portal |
| Exceptions | 1. Connection Error |

|  |  |
| --- | --- |
| Use Case ID | VJF-057  Start whiteboard functionality |
| Description | Allows user to start the whiteboard functionality |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in and is in the interview portal page |
| Steps | 1. Actor clicks on “Start Whiteboard” 2. Use case ends when whiteboard becomes visible to the actor |
| Post-conditions | The user will have either an embedded whiteboard or a whiteboard on a new window |
| Exceptions | 1.   There is a connection error connecting with the whiteboard |

|  |  |
| --- | --- |
| Use Case ID | VJF-058 Draw on whiteboard |
| Description | Allows actor to draw on the whiteboard |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in and currently has the whiteboard functionality open |
| Steps | 1. On the whiteboard screen, the actor clicks on the drawing tool 2. On the blank space in the whiteboard screen, actor clicks and holds the left click of the mouse 3. Actor drags the mouse around the whiteboard screen 4. Use case ends when actor releases the left click |
| Post-conditions | There will be a drawing on the whiteboard |
| Exceptions | None |
| Use Case ID | VJF-059 Change color of drawing tool |
| Description | Allows actor to change color of drawing tool |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in and currently has the whiteboard functionality open |
| Steps | 1. Use case begins when, on the whiteboard screen, the actor clicks on the color palette tool 2. From the options that pop up, the actor selects the color he or she wants to use 3. Use case ends when the new color shows up \_\_\_\_\_\_ |
| Post-conditions | There will be a drawing on the whiteboard |
| Exceptions | 1.   User clicks outside of the color palette, and the color is not changed |

|  |  |
| --- | --- |
| Use Case ID | VJF-060 Type text into whiteboard |
| Description | Allows to type text into the whiteboard |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in, is interviewing and  has the whiteboard functionality open |
| Steps | 1. Use case begins when, on the whiteboard screen, the actor clicks on the “Pencil” option on the left 2. From the sub-menu that pops up, the user shall click on “Text” 3. The user shall click anywhere on the whiteboard screen where he/she wants the text to show 4. The system shall display a window with a textbox, and the user shall type the text that will appear 5. The user shall press “OK” after entering the text |
| Post-conditions | 1. The text the user typed will appear on the screen where the user initially clicked before typing the text |
| Exceptions | 1.   In step 2, if the user clicks on the whiteboard when the sub-menu is showing, the current whiteboard feature selected will be used  2.      In step 4, if the user clicks “Cancel” instead of “OK”, the window will disappear, and not text will be shown |
| Use Case ID | VJF-061 Clear contents of whiteboad |
| Description | Allows actor to clear the current contents of the whiteboard |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in, is interviewing and  has the whiteboard functionality open |
| Steps | 1.      Use case begins when, on the whiteboard screen, the actor clicks on the “Menu” option on the left  2.      From the sub-menu that pops up, the user shall click on “Clear”  3.      The system shall display a pop-up message confirming if the user wants to continue   1. Use case ends when user clicks “OK” on the pop-up |
| Post-conditions | 1.         The whiteboard screen is cleared |
| Exceptions | 1.      In step 4, if the user clicks on “Cancel”, the whiteboard contents will not be cleared, and the whiteboard screen will be shown again |

|  |  |
| --- | --- |
| Use Case ID | VJF-062    Block an user |
| Description | Allows a student to block an employer (and vice versa) |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in and is viewing the profile of the user he/she wants to block |
| Steps | 1. Use case begins when the actor clicks on “Block User” 2. The system shall present a pop-up asking the actor whether he/she really wants to block the other user 3. The actor shall click “Yes” to confirm the blocking 4. Use case ends when system gives confirmation of blocking with a pop-up message |
| Post-conditions | 1. The blocking user will not be found when searched by the user who was blocked  2. Actor will be sent to profile of the actor who was blocked |
| Exceptions | In step 3, if the actor clicks on “No”, then no blocking will be done, and the actor will remain on the same website |

|  |  |
| --- | --- |
| Use Case ID | VJF-063    Unblock an user |
| Description | Allows a student to unblock an employer that was previously blocked (and vice versa) |
| Actor | Employer or student |
| Pre-conditions | Actor is logged in and is viewing the profile of the user he/she wants to unblock |
| Steps | 1.      Use case begins when the actor clicks on “Unblock User”  2.      The system shall present a pop-up asking the actor whether he/she really wants to unblock the other user   1. The actor shall click “Yes” to confirm the blocking 2. Use case ends when system gives confirmation of unblocking with a pop-up message |
| Post-conditions | 1. The blocking user will be able to be found when searched by the user who was blocked  2. Actor will be sent to the profile of the user who has unblocked |
| Exceptions | In step 1, if the user that is being unblocked had not been blocked before, then the actor will not see the “Unblock user” option |

5.3.   Appendix C

**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)

## 

## 5.4.   Appendix D - 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.

# 6. 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>