Final Deliverable

CIS 4911 – Senior Project U01

Virtual Job Fair

**Members:**

Justin Korah

Andres Gonzalez

Tomer Doar

Diego Perez

Enmanuel Corvo

**Mentors:**

Masoud Sadjadi

Juan Caraballo

Copyright

*The contents of the Virtual Job Fair system and all associated documents belong to Florida International University and are only to be used for the purposes decided by FIU faculty. Redistribution or commercial use is prohibited without permission by the appropriate faculty in charge of the project and its contents.*

Executive Summary

*Virtual Job Fair seeks to aid the process of recruitment for SCIS students at Florida International University. The project attempts to connect students and companies in a more personal way than your typical job posting and application site. More particularly, the system attempts to serve as a tool for companies, especially for those who are unable to make campus visits to evaluate the talent at the university. The following document contains a comprehensive description and analysis of the work carried out by the Virtual Job Fair team. It serves as a detailed guide of all the requirements gathered for the system, an analysis of those requirements, consequent specification and development of a design solution, and a description of the implementation.*

*Section 1 gives a brief overview of the scope of the system and also serves as an introduction for the rest of the document. Section 2 analyzes the feasibility of the proposed system based on the current system and how the proposed solution can significantly improve it within the allotted time. Section 3 outlines the project plan (main tasks to be completed and needed resources). Section 4 describes the high level system requirements and their corresponding analysis. Section 5 discusses the high level system architecture while section 6 analyzes it in greater detail. Section 7 is concerned with the testing of the whole system. Section 8 contains the glossary and section 9 contains the following appendixes: project plan, use cases, ui designs, analysis models, design models, documented class interfaces, test drivers and stubs, diary of meetings).*

Contents

[1. Introduction 6](#_Toc354467306)

[1.1 Problem definition 6](#_Toc354467307)

[1.2 Scope of system 6](#_Toc354467308)

[1.3 Over all development methodology 7](#_Toc354467309)

[1.4 Definitions, acronyms, and abbreviations 8](#_Toc354467310)

[1.5 Overview of document 9](#_Toc354467311)

[2. Feasibility Study 10](#_Toc354467312)

[2.1 Description of current system 10](#_Toc354467313)

[2.2 Description of alternative solutions considered 12](#_Toc354467314)

[2.3 Recommendation 13](#_Toc354467315)

[3. Project Plan 14](#_Toc354467316)

[3.1 Project Organization 14](#_Toc354467317)

[3.1.1 Project Personnel 14](#_Toc354467318)

[3.1.2 Hardware and Software Resources 14](#_Toc354467319)

[3.2 Identification of Tasks, Milestones and Deliverables 15](#_Toc354467320)

[3.3 Cost of the Project 15](#_Toc354467321)

[4. System Requirements 17](#_Toc354467322)

[4.1 Functional and Nonfunctional Requirements 17](#_Toc354467323)

[4.2 Requirements Analysis 21](#_Toc354467324)

[4.2.1 Use case model 21](#_Toc354467325)

[4.2.2 Static model 22](#_Toc354467326)

[4.2.3 Dynamic model 22](#_Toc354467327)

[5. System 23](#_Toc354467328)

[5.1 Overview 23](#_Toc354467329)

[5.2 Subsystem Decomposition 24](#_Toc354467330)

[5.3 Hardware and Software Mapping 27](#_Toc354467331)

[5.4 Persistent Data Management 28](#_Toc354467332)

[5.5 Security/Privacy 36](#_Toc354467333)

[6. Detailed Design 38](#_Toc354467334)

[6.1 Overview 38](#_Toc354467335)

[6.2 Static model 38](#_Toc354467336)

[6.3 Dynamic model 40](#_Toc354467337)

[6.4 Code Specification 41](#_Toc354467338)

[7. System Validation 42](#_Toc354467339)

[7.3 Evaluation of Tests 54](#_Toc354467340)

[8. Glossary - define terms used in document, especially domain specific terms. 57](#_Toc354467341)

[9. Appendix 58](#_Toc354467342)

[9.1 Appendix A - Project schedule 58](#_Toc354467343)

[9.2 Appendix B – All use cases with nonfunctional requirements 58](#_Toc354467344)

[9.3 Appendix C – User Interface designs 86](#_Toc354467345)

[9.4 Appendix D – Analysis models 86](#_Toc354467346)

[9.5 Appendix E – Design models 91](#_Toc354467347)

[9.6 Appendix F – Documented Class interfaces and constraints 100](#_Toc354467348)

[9.7 Appendix G – Documented code for test drivers and stubs 110](#_Toc354467349)

[9.8 Appendix H – Diary of meeting and tasks for the entire semester 113](#_Toc354467350)

[10. References 116](#_Toc354467351)

# 1. Introduction

The introduction gives context to the project in terms of what the project sets out to solve. It starts by stating the problem seen before starting the project, also the reason for the project. The introduction than goes into the scope of the system in terms of what the system will do and what it won’t do. Then development methodology is discussed as to what methodology the team used during the projects lifecycle. Following that, definitions for the deliverable are given so they can be referred to as the reader goes through the document. Then an overall explanation of the document is given.

## Problem definition

Recruiters everywhere are always looking for talent at the college level to fill in internship and entry level roles. Currently the most effective way for employers to recruit at schools is to visit them and set up presentations or attend career fairs at the school. Since not all employers have the resources to actively visit schools searching for talent, universities have provided career sites that enable employers to post jobs along with contact info, so that students can apply. This solution, however, is not sufficient and is not all that much different than online job boards. It is not nearly as effective as a campus visit where companies can connect with students on a face to face basis, which is what Virtual Job Fair will try to emulate.

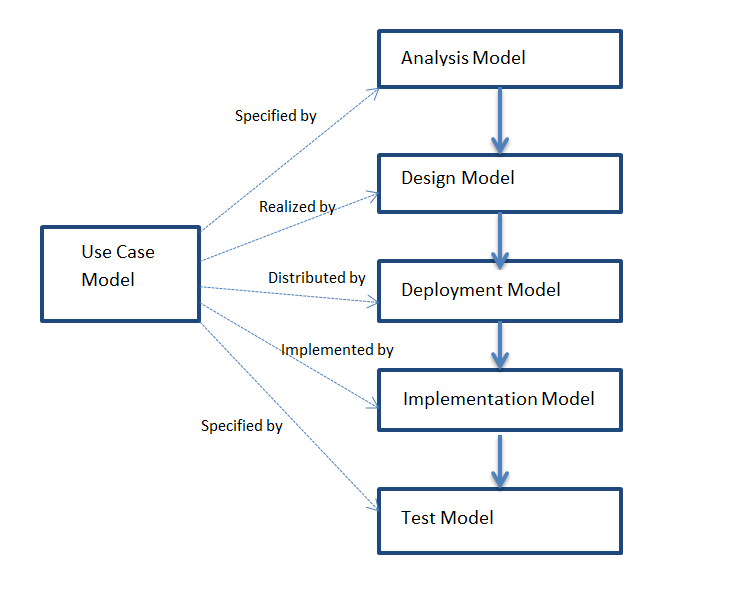
## Scope of system

The system will allow for employer and student registration. Employers will be able to register and give information about themselves as well as the company, which will be shown in a public profile available for student viewing. Likewise, students will create profiles which contain education information, experience information, and information about his or her skillset. These student profiles are viewable by employers and employers will have the functionality to search for students with certain criteria.

Additionally, employers are able to post job and internship openings which can be viewed and searched for by students. Students can then apply to these jobs and provide a cover letter for each application. Employers can view who applied to their postings and view their profile, where they could also download their resume. Furthermore, employers are able to schedule video interviews with students, which will take place on the system.

The system contains other functionality to make recruiting easier on the students and the faculty. These features include, but are not limited to: a messaging system, a notification system, video resumes, and job to student skill matching

## Over all development methodology



The team will utilize the Unified Software Development Methodology to develop and carry out the software deliverables. The team will benefit from using a methodology that has the characteristics of an iterative and incremental development process. Using USDP, the team will be presented with clear use cases that the system is required to support. By the end of the project, the team is expected to perform cycles which includes: inception, elaboration, construction, and transition. Each cycle should end with a software product ready to deliver.

The USDP methodology utilized many different UML models to aid in the analysis, design, and implementation phases. Such models can be referred during any stage of the product for clarification and familiarization of the system. These models are made easy to reference to with the use of many diagrams. Use case diagrams showcases the actors of the systems and the use cases associated with the actors. Sequence diagrams show the flow of the system not only through what the user sees, but how the system actually performs a use case. Deployment diagrams give an idea of how the system is run in terms of the hardware and software used to deploy the system. Class diagrams give a look into the design and implementation and how the objects are instantiated, which is very useful for future development.

## Definitions, acronyms, and abbreviations

|  |  |
| --- | --- |
| Term | Meaning |
| CareerPath | Current career page for SCIS department, shown in www.cis.fiu.edu/careerpath |
| USDP | Unified Software Development Process |
| SCIS | School of Computer and Information Sciences |
| CS | Computer Science |
| FIU | Florida International University |
| IT | Information Technology |
| CIS | Computer and information Sciences |
| TC | Test case |
| MVC | Model View Controller |
| API | Application programing interface |
| Term | Meaning |
| CareerPath | Current career page for SCIS department, shown in www.cis.fiu.edu/careerpath |
| SCIS | School of Computer and Information Sciences |
| CS | Computer Science |
| FIU | Florida International University |
| IT | – Information Technology |
|  |  |

## 1.5 Overview of document

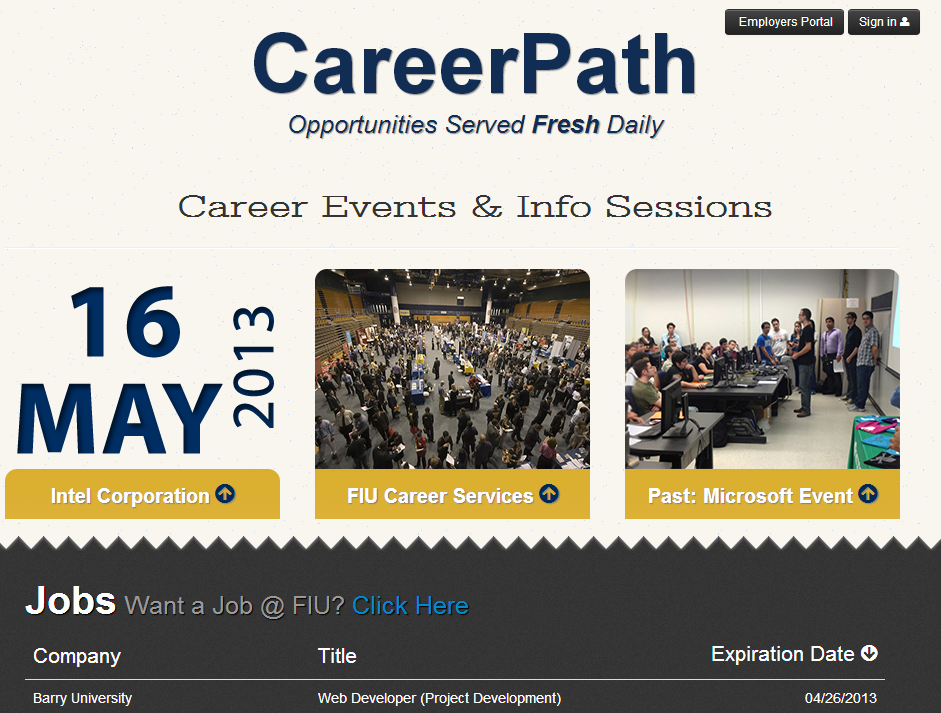
This is an overview of the content of this document, which briefly describes what is covered in each section.

Section 2 of this document covers the current systems available as solution to the problem we are trying to solve. It goes into details to explain why the current systems do not efficiently solve the problem and what they lack. Section 3 covers the project plan; this will include the project organization and the work breakdown between team members and their individual contribution. Section 4 covers the system requirements, which include functional requirements and analysis of the system requirements. This section will include diagrams covering the implementation requirements of the system, such as Use Case Model, Static Models, and Dynamic Models. Section 5 is a glossary where definition to specific terms used can be found. Section 6 consists of appendixes which contain mostly diagrams. Finally, sections 7 are the references used in this document.

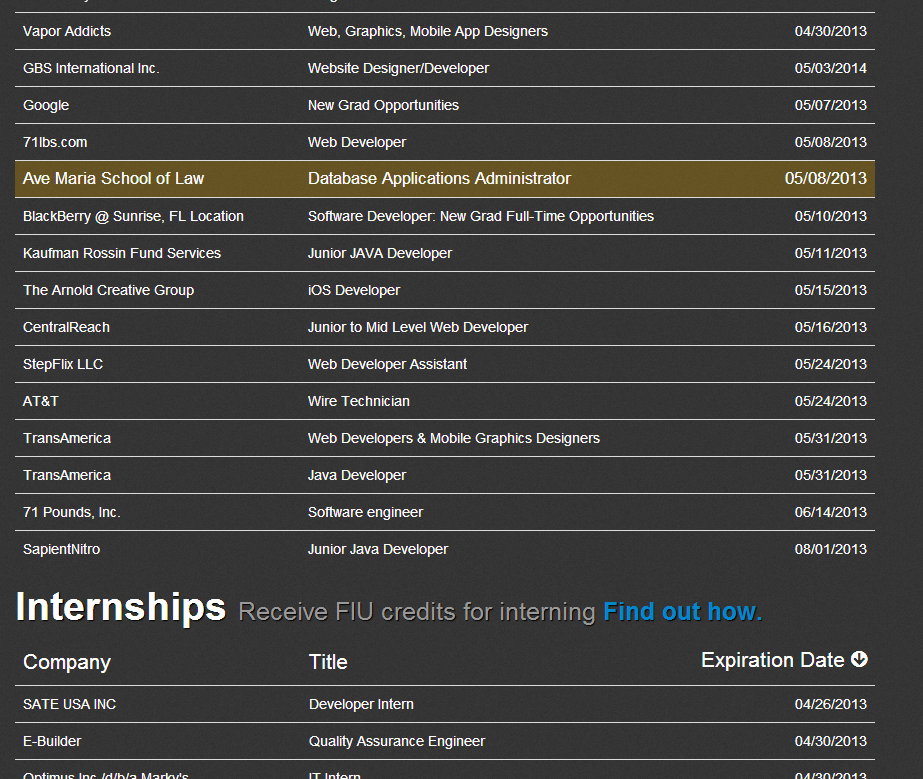
# 2. Feasibility Study

## 2.1 Description of current system

The CIS department currently has a web page set up with listings of internships and jobs. This current setup is known as CareerPath and is located at [www.cis.fiu.edu/careerpath](http://www.cis.fiu.edu/careerpath). In its current implementation, the system allows companies to post jobs and internships. These postings include information about the company, a description for the open position, the expected duties, and the qualifications needed to perform in the position. It also provides an email address to contact as well as a website to refer to. The system serves a basic need but is lacking features when compared to modern job sites.



Home page for CareerPath



Job list in CareerPath



Job description in CareerPath

## 2.2 Description of alternative solutions considered

While all of the solutions considered correspond to features that are not implemented in the current system, some of them were either not operationally feasible or would have taken more time than available. One of these alternative solutions was the online coding competition, which was originally proposed as one of the requirements and ended up being delegated to an alternative solution in favor of the online video interview.

Another alternative solution was enabling rating of students by professors, but it was not validated by the mentors.

## 2.3 Recommendation

The final, agreed upon solutions were selected because of the significance of the problem that each one of them tackles. They were validated by the mentors and it was decided that each team member would focus specifically on one of the solutions. For future work on a similar project, the Virtual Job Fair team recommends integrating an online coding compiler into the system as the means for supporting a coding competition controlled by employers.

# 3. Project Plan

## 3.1 Project Organization

### 3.1.1 Project Personnel

|  |  |  |
| --- | --- | --- |
| Team Member | Primary Task | Other Task |
| Justin Korah | Skill-based Student Job Match | General Functionality |
| Andres Gonzalez | Messaging System | General Functionality |
| Tomer Doar | Notification and Email System | General Functionality |
| Diego Perez | LinkedIn Integration and video resume | General Functionality |
| Enmanuel Corvo | Video Interview | General Functionality |

### 3.1.2 Hardware and Software Resources

Hardware

* Windows desktops and laptops
* Apple MacBook Pro laptops
* Panasonic Projector (team collaboration)

Software

* Eclipse Juno
* Scientific Linux Release 6.1
* Subversion v1.7
* Subclipse v1.0.7
* Apache v.2.2.15
* MySQL Workbench v5.2.45
* MySQL v5.0
* Yii Framework v1.1.13
* PHP v5.3.3
* Selenium v2.28

## 3.2 Identification of Tasks, Milestones and Deliverables

The project schedule in the form of a Gantt chart is found in Appendix A.

|  |  |  |
| --- | --- | --- |
| Task Name | Start Date | Due Date |
| Gather Requirements | 01/14/13 | 01/25/13 |
| Requirement Analysis | 01/24/13 | 01/28/13 |
| Setup Solution Stack | 01/21/13 | 01/28/13 |
| Deliverable 1 | 01/28/13 | 02/04/13 |
| Database Design and Setup | 01/28/13 | 02/11/13 |
| System Design | 02/11/13 | 02/25/13 |
| Deliverable 2 | 02/11/13 | 02/25/13 |
| Implementation of General Functionality | 02/25/13 | 03/15/13 |
| Implementation of Individual Features | 03/04/13 | 04/04/13 |
| Deliverable 3 | 03/22/13 | 04/04/13 |
| Test Case Write-up | 03/29/13 | 04/05/13 |
| Test Case Execution | 04/05/13 | 04/12/13 |
| Deliverable 4 | 04/04/13 | 04/19/13 |
| Code Improvement | 04/13/13 | 04/25/13 |

|  |  |
| --- | --- |
| Milestone Name | Expected Date |
| Requirements Complete | 02/04/13 |
| System Design Complete | 02/25/13 |
| Implementation Completed | 04/04/13 |
| Documentation Completed | 04/19/13 |
| Project Completed | 04/26/13 |

## 3.3 Cost of the Project

Given the open sourced nature of the technologies that will be used, there will be no costs associated with building our project other than the time factor.

# 4. System Requirements

The Virtual Job Fair is the name of the system that is proposed. The Virtual Job Fair is a website that will provide the opportunity for employers to post jobs and internships, and will allow students to apply to these jobs. Furthermore, the site will in many ways promote more interaction between employers and students. The site will allow for both students and employers to generate user profiles. Employer profiles will have basic information about the user and the company. Student profiles are a little more extensive and will contain past experience, education, and will showcase a list of skills the student claims to have obtained. The employer will also have access to the resume through the profile.

To solve the specific problem of recruiters not being able to host sessions at every school, a virtual interview system is set up to aid the talent search. Employers will be able to schedule interviews with students and the interview will be able to take place on the website. Such a system will allow employers and students to get to know each other on a face to face basis without actually being in the same room. This will be very beneficial to companies who recruit for talent nationwide.

## 4.1 Functional and Nonfunctional 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 integrate with their LinkedIn account to provide education and experience information (security)**
* 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 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.

## 4.2 Requirements Analysis

This section gives a brief overview of each of the requirement artifacts presented in Appendix B-D. It goes into a bit more detail regarding the requirements of the system, presenting the team of developers with the specific tasks, artifacts, and interactions that will need to be implemented to complete the system.

### 4.2.1 Use case model

REFER TO APPENDIX B

The Use Case diagram in appendix B gives a high-level overview of all the functional requirements of the system. It identifies the three types of users that our system interacts with: students, employers, and administrator. For each type of user, the model identifies the tasks that the user should be able to carry out upon completion of the system.

The tasks on the left-hand side encompass everything a student should be able to accomplish. Most of these tasks have to do with the creation and management of a student profile, while the last task listed represents the ability of students to apply for jobs.

The tasks on the right-hand side represent what the employers and administrator should be able to do upon completion of the system, while the tasks shown in the center represent functionality common to all types of users.

### 4.2.2 Static model

REFER TO APPENDIX C

The Static Model in appendix C provides developers with a specification of all the classes that should be implemented for the system. It does not differentiate between the different subsystems; it simply contains all of the necessary classes that were identified during the requirements analysis, i.e. the collection of all the classes from the different subsystems. Conventional UML notation was used to make it easier to differentiate between boundary, controller, and model classes, as per-required for the MVC architectural pattern.

### 4.2.3 Dynamic model

REFER TO APPENDIX D

The Dynamic model in appendix D contains the sequence diagrams extracted from analyzing the specific requirements of each of the Use Cases presented in Appendix A. It gives the team of developers a more specific view of the interaction between the user and the system for each specific piece of functionality from the Use Case Model.

By referring to this diagram, developers can see the main artifacts involved in the flow of information for each use case, and how they are associated with each other: boundary objects as the intermediaries between user requests and controllers, controllers accessing model objects and passing along the returned data to the boundary objects, etc.

Conventional UML notation was used to produce the sequence diagrams.

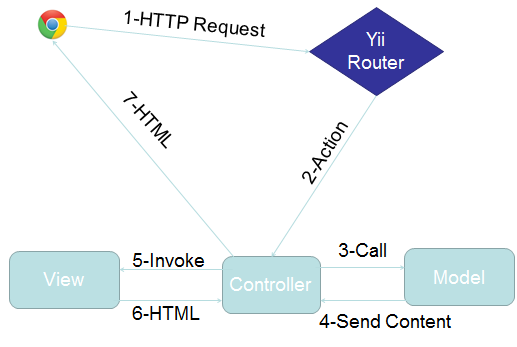
# 5. System

The system design section looks into the reasoning behind the decisions that were made during the design process. This includes a look into the subsystem, the hardware and software mapping, the persistent data design, and the security and privacy features. The subsystem decomposition gives a high-level explanation of the major subsystems (Video interview, messaging system, notification system, profile creation system, and student job match system). The hardware and software mapping provides a deployment diagram to visualize what brings the system to life. The persistent data management goes into the database design of the system. And lastly, the security/privacy section explains the security and privacy features implemented into the system.

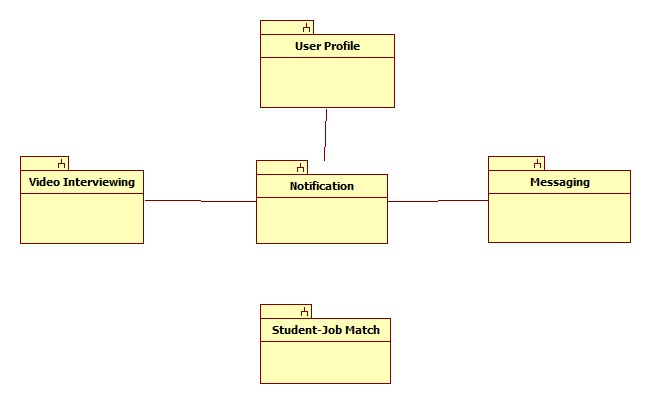
## 5.1 Overview

In order to break down the system into manageable parts, it was necessary to identify the main features that the system is going to support and how they differ in terms of technology used and functionality. Following this phase, it became evident that the system could be decomposed into 5 main subsystems, each one accomplishing many different tasks identified during Requirements Analysis. This section includes, among other things, an overview of each one of these subsystems.

The Architectural pattern used to built 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:



A depiction of the main subsystems is given below. Refer to section 2.2 for a description of the functionalities of each subsystem:



## 5.2 Subsystem Decomposition

The main subsystems of the Virtual Job Fair include the video interview subsystem, the messaging subsystem, the notification subsystem, the profile creation subsystem and the student job match subsystem.

**Video Interview Subsystem**

The Video Interview Subsystem allows users to have a virtual interview. This is possible thanks to Web-RTC technology. Web-RTC is an open source project that allows web browsers to communicate directly with each other with the aid of Java Scrip API calls and HTML5. This new technology makes the communication between internet users easier than traditional methods. Thanks to Web-RTC one can share video feed with other users without the need for media servers or plug-ins.

The synchronization of video interviews is handled in a table in the database. This is very important because we need to make sure only users scheduled to have an interview can be in the interview page. This is done by using a session key which is unique for each video interview scheduled. Users that arrive at the interview page would do so via a link which has many parameters such the session key. Once both users arrive at the interview page, the application will check for the session key parameter and match them, and only those users that have matching session keys would be able to connect to each other.

The uses cases related to this subsystem are:

* VJF-0020 Start Video Interview
* VJF-0021 Accept Interview
* VJF-0030 Schedule Video Interview

**Messaging Subsystem**

The messaging subsystem is a very important one because it 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 notification subsystem is a very important part of the system. This 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.

The uses cases related to this subsystem are:

* VJF-0021 Accept Interview
* VJF-0025 Post Job
* VJF-001 Registration
* VJF-0033 Apply to Job
* VJF-0034 Read notification

**Profile creation 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.

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

**Student job match subsystem**

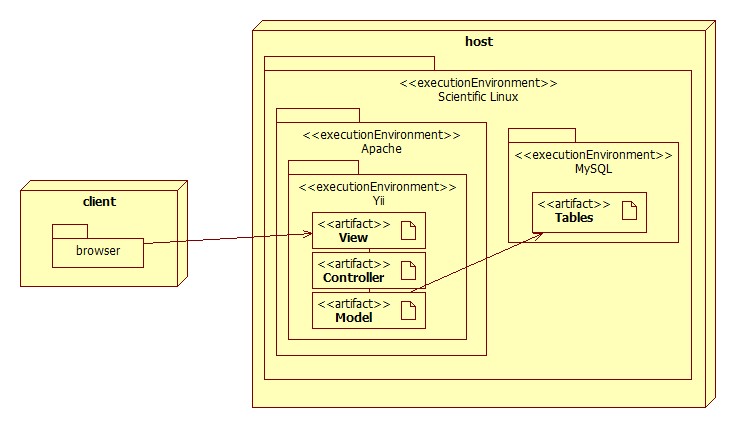
Making the right connection is what this web application is all about. Therefore, an efficient algorithm to match students to job openings is very important. The student job match subsystem takes care of matching students with the required skills to job post, making the job of the recruiters easier, 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
* VJF-0026 Virtual Handshake

## 5.3 Hardware and Software Mapping

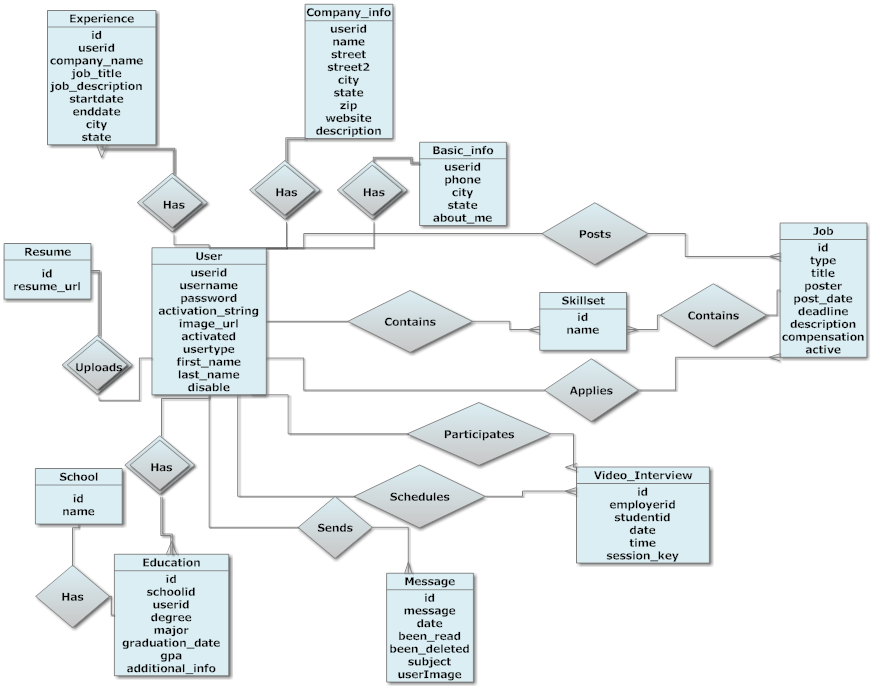


The deployment diagram shown above represents the hardware and software mapping in the Virtual Job Fair system. The main components of the system are the Apache and MySQL environment hosted on a Linux operating system. The Yii framework environment is using apache to execute, and contains our various 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

Data is primarily being stored in a MySQL database. It is necessary to store all persistent data in tables to hold information such as: user profiles, job postings, applications, interview schedules, system messages, notifications, and much more. The file system is also being used to hold certain files such as profile images and student resumes. The database contains URL references to such images and resumes in order to display them.

The design of the database can be visualized in the ER diagram below:



The database was built with extensive use of foreign keys to benefit the development framework we chose. The framework we chose automatically generates models based on tables and relations based on foreign keys, eliminating the need for almost all SQL in the application code.

The following data dictionary provides information on each database table used throughout the system. It gives field names as well as field types and extra information such as primary keys and default values.

**application**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| jobid | int(11) | NO | PRI |  |  |
| userid | int(11) | NO | PRI |  |  |
| application\_date | varchar(45) | NO |  |  |  |
| coverletter | text | YES |  |  |  |

**basic\_info**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| userid | int(11) | NO | PRI |  |  |
| phone | varchar(15) | YES |  |  |  |
| city | varchar(45) | YES |  |  |  |
| state | varchar(45) | YES |  |  |  |
| about\_me | text | YES |  |  |  |
| hide\_phone | int(11) | YES |  |  |  |

**company\_info**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| FK\_userid | int(11) | NO | PRI |  | auto\_increment |
| name | varchar(45) | YES |  |  |  |
| street | varchar(45) | YES |  |  |  |
| street2 | varchar(45) | YES |  |  |  |
| city | varchar(45) | YES |  |  |  |
| state | varchar(45) | YES |  |  |  |
| zipcode | varchar(45) | YES |  |  |  |
| website | varchar(45) | YES |  |  |  |
| description | text | YES |  |  |  |

**education**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| degree | varchar(45) | NO |  |  |  |
| major | varchar(45) | NO |  |  |  |
| graduation\_date | date | NO |  |  |  |
| FK\_school\_id | int(11) | YES | MUL |  |  |
| FK\_user\_id | int(11) | YES | MUL |  |  |
| gpa | float | YES |  |  |  |
| additional\_info | text | YES |  |  |  |

**experience**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| FK\_userid | int(11) | YES | MUL |  |  |
| company\_name | varchar(45) | YES |  |  |  |
| job\_title | varchar(45) | YES |  |  |  |
| job\_description | text | YES |  |  |  |
| startdate | datetime | YES |  |  |  |
| enddate | datetime | YES |  |  |  |
| city | varchar(45) | YES |  |  |  |
| state | varchar(45) | YES |  |  |  |

**handshake**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| jobid | int(11) | YES | MUL |  |  |
| employerid | int(11) | NO | MUL |  |  |
| studentid | int(11) | NO | MUL |  |  |

**job**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| type | varchar(45) | NO |  |  |  |
| title | varchar(45) | NO |  |  |  |
| FK\_poster | int(11) | NO | MUL |  |  |
| post\_date | datetime | NO |  |  |  |
| deadline | datetime | YES |  |  |  |
| description | longtext | NO |  |  |  |
| compensation | varchar(45) | YES |  |  |  |
| other\_requirements | text | YES |  |  |  |
| email\_notification | int(11) | YES |  |  |  |
| active | int(11) | YES |  | 1 |  |
| matches\_found | int(11) | YES |  |  |  |

**job\_skill\_map**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| jobid | int(11) | NO | MUL |  |  |
| skillid | int(11) | NO | MUL |  |  |
| level | varchar(45) | YES |  |  |  |
| ordering | int(11) | YES |  |  |  |

**message**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| FK\_receiver | varchar(45) | NO | MUL |  |  |
| FK\_sender | varchar(45) | NO | MUL |  |  |
| message | text | YES |  |  |  |
| date | datetime | YES |  |  |  |
| been\_read | int(11) | YES |  | 0 |  |
| been\_deleted | int(11) | NO |  | 0 |  |
| subject | varchar(255) | YES |  |  |  |
| userImage | varchar(255) | YES |  |  |  |

**notification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| sender\_id | int(11) | NO | MUL |  |  |
| receiver\_id | int(11) | NO |  |  |  |
| datetime | time | NO |  |  |  |
| been\_read | int(11) | NO |  | 0 |  |
| message | varchar(5000) | YES |  |  |  |
| link | varchar(150) | YES |  |  |  |
| importancy | int(11) | NO |  | 0 |  |

**resume**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  |  |
| resume | varchar(255) | YES |  |  |  |

**school**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| name | varchar(100) | NO |  |  |  |
| email\_string | varchar(45) | YES |  |  |  |

**skillset**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| name | varchar(45) | NO | UNI |  |  |

**student\_skill\_map**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| userid | int(11) | YES | MUL |  |  |
| skillid | int(11) | YES | MUL |  |  |
| level | varchar(45) | YES |  |  |  |
| ordering | int(11) | YES |  |  |  |

**user**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| username | varchar(45) | NO | UNI |  |  |
| password | varchar(255) | YES |  |  |  |
| FK\_usertype | int(11) | NO | MUL |  |  |
| email | varchar(45) | NO | UNI |  |  |
| registration\_date | datetime | NO |  |  |  |
| activation\_string | varchar(45) | NO |  |  |  |
| activated | int(11) | YES |  |  |  |
| image\_url | varchar(255) | YES |  |  |  |
| first\_name | varchar(45) | NO |  |  |  |
| last\_name | varchar(45) | NO |  |  |  |
| disable | int(11) | YES |  |  |  |
| has\_viewed\_profile | int(11) | YES |  |  |  |
| linkedinid | varchar(45) | YES |  |  |  |
| googleid | varchar(45) | YES |  |  |  |
| hide\_email | int(11) | YES |  |  |  |

**usertype**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| type | varchar(45) | NO |  |  |  |

**video\_interview**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  | auto\_increment |
| FK\_employer | int(11) | NO | MUL |  |  |
| FK\_student | int(11) | NO | MUL |  |  |
| date | date | NO |  |  |  |
| time | time | NO |  |  |  |
| session\_key | varchar(45) | NO |  |  |  |
| notification\_id | varchar(45) | NO |  |  |  |

**video\_resume**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI |  |  |
| video\_path | varchar(100) | YES | UNI |  |  |

## 5.5 Security/Privacy

**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 can not. 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 employers 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.

# 6. Detailed Design

This section breaks down the design of the system into a higher level of detail, specifying exactly how each of the subsystems is broken down into associated classes, along with their attributes and inheritance hierarchy. It also lays out the dynamic model of the system by a series of sequence diagrams, and shows the method signatures of the main classes of each subsystem.

## 6.1 Overview

For each one of the subsystems presented in section 5.2, the Active Record pattern is the most evidently used. Each one of the model classes in each subsystem is an Active Record, i.e., they inherit from the Active Record Base class. This provides a very simple interface to the persistent data of the system, since each attribute of a class is mapped to a column of a row in a database table. The Active Records also come packed with a lot of useful functionality provided by the Yii Framework.

Refer to Section 5.2 for a description of each subsystem

## 6.2 Static model

Refer to appendix E for class diagrams

**Student Job Match Subsystem**

The student job match subsystem structure is made up of eight database tables, which share common data such as skills. It starts with the employer entity which contains a job post and such job post contains a skill set which consist of specific skills. Following, the student entity comes into play when a match is done. The system basically matches the skills in the skill set of a job post belonging to an employer and then it raises a notification for the notification subsystem to deliver. This subsystem also includes a handshake use case, which allows the employer to virtually handshake students who are matches for a job. The virtual handshake acts as a midpoint between an interview. This also raises a notification to be delivered by the notification subsystem.

**Video Interview Subsystem**

The virtual interview subsystem structure relies of external technologies such WebRTC, easyrtc and Node.js. These frameworks allow the video interview subsystem to function properly by providing the most basic and necessary functionalities. All these subsystems within the video interview subsystem need to be set up and synchronized to work in harmony with each other; part of this synchronization takes place when a video interview is schedule. The scheduling of a video interview is store in a database table which holds relation to the employers and students. This allows the system to identify the two person that will be on the interview and limit the access for other users that might be at the page at the same time.

When a video interview is taking place, Node.js is configured to send and receive data from both users. This might sound like an easy task but synchronizing the send and receive of data is a tricky thing. Node.js makes sure that the server response to request by both users and each response goes respectively to the receiver from the origination point. To ensure that the server is always active and listening, Node.js must always be running; therefore, it was configured to run automatically on port 8080.

**Messaging Subsystem:**

The messaging subsystem structure relies heavily on Ajax to transfer json data between the client and the server. This type of interaction between the user and the system was implemented in most of the messaging use cases and results in a dynamic user experience (desktop-like), but it is a bit more difficult to implement than a pure Restful implementation for instance. Given the allotted time, it was decided that it could be implemented this way for the purposes of improving the user experience. The tradeoff was implementation time and implementation simplicity.

**Notification Subsystem:**

The notification subsystem acts as the mailman for the whole system. It is interconnected with all other subsystems to receive information and propagate that information through the system to the respective users. The notification subsystem is connected to the main controllers of all other subsystem. This connectivity allows it to keep track of all interactions between all the other subsystems and stay up to date with what is going on.

**Profile Creation Subsystem:**

The profile creation subsystem structure relies on API calls to third-party website such LinkedIn. These API calls allows the users to import their LinkedIn Information to Virtual Job Fair. As the information if being fetch into the system, the profile creation subsystem checks the data to make sure it is coherent. For example, when importing skills from LinkedIn, the profile creation subsystem checks for duplicates in the user profile, so it does not import redundant information.

## 6.3 Dynamic model

Refer to appendix E for state machine diagrams

**Video Interview Subsystem**

Video interviews involve three actions: schedule Interview, accept interview, and do interview. In the schedule interview process, entry is done by clicking on schedule on a student profile and filling out the form. Upon exit, the data is entered into the video\_interview table which is consulted throughout the whole subsystem. During the accept interview process, a notification is created to let the employer know the interview has been accepted. The do interview design makes use of WebRTC functionality and redirects the user away from the main site temporarily. Since WebRTC makes use of a node.js server, we were unable to use apache to display this page like the rest of the site. To display profile information on the interview page, iframes are used with reference to the main site since PHP files cannot be served on the node.js pages.

**Notification Subsystem**

The notification subsystem relies on the notifications table to function. Methods are created within model classes (mostly User) to handle events such as registration, message creation, and job postings. These methods create notifications and if necessary send out emails to users about the event. The subsystem also relies on the mail function which is provided by a library in Yii. Read notification is handled when a notification link is clicked. The link contains a parameter containing the notification id and upon page completion, the parameter will be picked up and the notification entry in the database will be marked as read. Delete notification simply deletes a row from the database.

**Profile Creation Subsystem**

The profile creation subsystem accesses many tables: user, basic\_info, company\_info, education, and experience. All of these tables are added or updated within the profile page. In addition, the profile creation subsystem relies heavily on API calls from LinkedIn. The profile is allowed to be created through LinkedIn profiles. To do this, LinkedIn API provides API access to get account information about education, experience and skills. This is done after authentication through LinkedIn’s website. Upon clicking LinkedIn connect, the user id redirected to a LinkedIn login. Upon logging in, the API returns information which is parsed automatically through a library provided by LinkedIn for PHP.

**Job Match Subsystem**

The Job match subsystem makes use of a comparison algorithm to give skill ratings to students compared to jobs. Students assign themselves skills in their profiles and order them from greatest to least. These skills may also have been pulled from the LinkedIn API. When employers post jobs, they assign skills in a similar manner, and order them. This action makes use of jQuery to allow sortable draggable lists in html. This also requires a lot of AJAX calls to query the database while filling out skills for a job. These AJAX calls also must check if the skill is a new skill, and will handle it appropriately. After the comparison, the top 4 students are returned to the student match page. This page is designed for the employer to view the profile and resume. The resume viewer makes use of a Google PDF viewer in an iframe. The virtual handshake also uses an AJAX call to send a notification the student, and upon completion it disables the button to restrict the call to a one-time function.

**Messaging Subsystem**

For the messaging subsystem, there’s a main message table on the backend which holds information regarding a message along with foreign keys to the receiver and sender of the message. When a user selects a message on the client side, or clicks on the list of ‘sent’ messages for instance, an Ajax call gets executed and handled by a controller method who accesses the corresponding data in the model, transforms the data into json, and sends it back to the javascript method in a callback function.

Once the data is received by the client, it is processed by a number of jquery methods who extensively manipulate the DOM of the page to present this data while destroying previously requested content. This pattern of execution was used in most of the messaging use cases and it is necessary for dealing with the Ajax calls.

## 6.4 Code Specification

Refer to appendix F for code interfaces

# 7. System Validation

Since Virtual Job Fair is a web-based application, the appropriate testing approach that was taken was to a user a program that automates browsers. Due to its highly available documentation and support, Selenium was chosen to test the application. Selenium is a suite of tools to automate web browsers. It can run on many browsers, operating systems, and can be controlled by many programming languages and frameworks. Testing was performed using Firefox browser and Java.

Our test cases focused on the most important use cases for the application. There are a few test cases that are in some way untestable using Selenium; these were excluded from the test cases but were extensively tested by hand inspection during development. Because of time constraints, subsystem testing was not performed completely but our system tests were able to test some aspects of the subsystems.

**7.1 Subsystem Tests**

Due to time constraints for the project, subsystem testing was omitted. However, the nature of the testing tool, Selenium, enabled us to test the subsystems while performing system testing. We present the test cases in the next section and provide the test code in Appendix G.

**7.2 System Tests**

|  |  |
| --- | --- |
| Test Case ID | VJF-TC1 |
| **Purpose** | Test **VJF-TC1** – Test successful login functionality |
| **Test Setup** | Username and password exist in the database that will enable the user to login. |
| **Input** | User navigates to VJF site and clicks Login.  User enters username and password. |
| **Expected Output** | User is redirected to Home page |
| **Actual output** | User is redirected to Home page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC2 |
| **Purpose** | Test **VJF-TC2** – Test unsuccessful login functionality |
| **Test Setup** | Username and password combination that user will input does not exist in the database |
| **Input** | User navigates to VJF site and clicks Login.  User enters username and password that does not exist. |
| **Expected Output** | The system displays “Incorrect username or password.” |
| **Actual output** | The system displays “Incorrect username or password.” |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC3 |
| **Purpose** | Test **VJF-TC3** – Test logout |
| **Test Setup** | User is logged in the system |
| **Input** | User clicks “Logout” |
| **Expected Output** | User is redirected to Main Page |
| **Actual output** | User is redirected to Main Page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC4 |
| **Purpose** | Test **VJD-TC4** – Test student search functionality by skill name |
| **Test Setup** | User is logged in as a student and will search for an existing skill by name |
| **Input** | User enters “PHP” in search box |
| **Expected Output** | Page displays job posts that require PHP as a skill |
| **Actual output** | Page displays job posts that require PHP as a skill |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC5 |
| **Purpose** | Test **VJF-TC5** – Test student search functionality by company name |
| **Test Setup** | User is logged in as a student and will search for an existing company by name |
| **Input** | User enters “IBM” in search box |
| **Expected Output** | Page displays job posts by IBM |
| **Actual output** | Page displays job posts by IBM |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC6 |
| **Purpose** | Test **VJF-TC6** – Test student search functionality with no results |
| **Test Setup** | User is logged in as a student and will search for an non existing record |
| **Input** | User enters “abcdefg” in search box |
| **Expected Output** | Page displays “No Results” |
| **Actual output** | Page displays “No Results” |

|  |  |
| --- | --- |
| Test Case ID | VJDF-TC7 |
| **Purpose** | Test **VJF-TC7**– Test successful employer job post functionality |
| **Test Setup** | User is logged in as an employer |
| **Input** | User clicks Post Job and enters required information for a job posting |
| **Expected Output** | User is redirected to the studentmatch page |
| **Actual output** | User is redirected to the studentmatch page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC8 |
| **Purpose** | Test **VJF-TC8**– Test unsuccessful employer job post functionality |
| **Test Setup** | User is logged in as an employer |
| **Input** | User clicks Post Job, fails to enter post title and clicks submit. |
| **Expected Output** | Page displays “You must input a job title” |
| **Actual output** | Page displays “You must input a job title” |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC9 |
| **Purpose** | Test **VJF-TC9** – Test successful student application for a job |
| **Test Setup** | User is logged in as a student |
| **Input** | User searches for a job post and clicks on its link. User clicks Apply |
| **Expected Output** | Page displays “Already applied” |
| **Actual output** | Page displays “Already applied” |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC10 |
| **Purpose** | Test **VJF-TC10** – Test successful employer search |
| **Test Setup** | User is logged in as employer and will search for an existing skill by name |
| **Input** | User enters “PHP” in search box |
| **Expected Output** | Page displays students that have PHP as a skill |
| **Actual output** | Page displays job posts that require PHP as a skill |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC11 |
| **Purpose** | Test **VJF-TC11** – Test unsuccessful employer search |
| **Test Setup** | User is logged in as employer; will search for an existing skill by name |
| **Input** | User enters “abcdefg” in search box (or any non available skill) |
| **Expected Output** | Page displays ‘No Results’ |
| **Actual output** | Page displays ‘No Results’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC12 |
| **Purpose** | Test **VJF-TC12** – Test employer close posting |
| **Test Setup** | User is logged in as employer and is viewing an existing post |
| **Input** | User clicks ‘Close Post’ |
| **Expected Output** | Page displays ‘This posting is closed’ |
| **Actual output** | Page displays ‘This posting is closed’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC13 |
| **Purpose** | Test **VJF-TC13** – Test employer close posting when already closed |
| **Test Setup** | User is logged in as employer and is viewing an existing closed post |
| **Input** | N/A |
| **Expected Output** | Page displays ‘This posting is closed’ |
| **Actual output** | Page displays ‘This posting is closed’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC14 |
| **Purpose** | Test **VJF-TC14** – Test student registration |
| **Test Setup** | User is located at the student register page |
| **Input** | User enters all required information to register and information is valid |
| **Expected Output** | Page displays ‘An email verification was sent’ |
| **Actual output** | Page displays ‘An email verification was sent’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC15 |
| **Purpose** | Test **VJF-TC15**– Test unsuccessful student registration |
| **Test Setup** | User is located at the student registration page |
| **Input** | User enters required information to register but forgets to enter username |
| **Expected Output** | Page displays ‘username must be at least 4 characters’ |
| **Actual output** | Page displays ‘username must be at least 4 characters’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC16 |
| **Purpose** | Test **VJF-TC16** – Test successful employer registration |
| **Test Setup** | User is located at employer registration page |
| **Input** | User enters all required information to register and information is valid |
| **Expected Output** | Page displays ‘You will receive an email soon when your account has been verified and activated’ |
| **Actual output** | Page displays ‘You will receive an email soon when your account has been verified and activated’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC17 |
| **Purpose** | Test **VJF-TC17** – Test unsuccessful employer registration |
| **Test Setup** | User is located at employer registration page |
| **Input** | User enters required information to register but forgets to enter username |
| **Expected Output** | Page displays ‘username must be at least 4 characters’ |
| **Actual output** | Page displays ‘username must be at least 4 characters’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC18 |
| **Purpose** | Test **VJF-TC18** – Test edit basic information for student |
| **Test Setup** | User is logged in as student and located at profile page |
| **Input** | User clicks ‘Edit Info’ and Enters ‘Florida’ for State. |
| **Expected Output** | Page displays ‘Florida’ in State field |
| **Actual output** | Page displays ‘Florida’ in State field |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC19 |
| **Purpose** | Test **VJF-TC19** – Test student edit basic information for employer |
| **Test Setup** | User is logged in as employer and located at profile page |
| **Input** | User clicks ‘Edit Info’ and Enters ‘Florida’ for State. User clicks ‘Edit Info’ again. |
| **Expected Output** | Page displays ‘Florida’ in State field |
| **Actual output** | Page displays ‘Florida’ in State field |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC20 |
| **Purpose** | Test **VJF-TC20** – Test remove student skill |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User clicks on the ‘x’ link next to the skill the user wants to delete |
| **Expected Output** | Skill is no longer displayed on profile |
| **Actual output** | Skill is no longer displayed on profile |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC21 |
| **Purpose** | Test **VJF-TC21** – Test add student skill |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User enters new skill in the add skill textbox and clicks ‘add skill’ |
| **Expected Output** | Page displays new skill on skill list |
| **Actual output** | Page displays new skill on skill list |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC22 |
| **Purpose** | Test **VJF-TC22** – Test student delete education |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User clicks on the ‘x’ link next to the education the user wants to delete |
| **Expected Output** | Page does not display the education |
| **Actual output** | Page does not display the education |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC23 |
| **Purpose** | Test **VJF-TC23** – Test student delete experience |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User clicks on the ‘x’ link next to the experience the user wants to delete |
| **Expected Output** | Page does not display the experience |
| **Actual output** | Page does not display the experience |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC24 |
| **Purpose** | Test **VJF-TC24** – Test student add experience |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User enters required information and information is valid |
| **Expected Output** | Page displays new experience on profile |
| **Actual output** | Page displays new experience on profile |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC25 |
| **Purpose** | Test **VJF-TC25** – Test student add education |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User enters required information and information is valid |
| **Expected Output** | Page displays new education on profile |
| **Actual output** | Page displays new education on profile |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC26 |
| **Purpose** | Test **VJF-TC26** – Test student change password successfully |
| **Test Setup** | User is logged in as student and has clicked ‘change password’ link |
| **Input** | User enters old password, new password, and new password confirmation correctly |
| **Expected Output** | User is redirected to profile page |
| **Actual output** | User is redirected to profile page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC27 |
| **Purpose** | Test **VJF-TC27** – Test student change password unsuccessfully |
| **Test Setup** | User is logged in as student and has clicked ‘change password’ link |
| **Input** | User enters old password (incorrectly), new password, and new password confirmation |
| **Expected Output** | Page displays ‘Old password was incorrect’ |
| **Actual output** | Page displays ‘Old password was incorrect’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC28 |
| **Purpose** | Test **VJF-TC28** – Test employer change password successfully |
| **Test Setup** | User is logged in as employer and has clicked ‘change password’ link |
| **Input** | User enters old password, new password, and new password confirmation correctly |
| **Expected Output** | User is redirected to profile page |
| **Actual output** | User is redirected to profile page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC29 |
| **Purpose** | Test **VJF-TC29** – Test employer change password unsuccessfully |
| **Test Setup** | User is logged in as employer and has clicked ‘change password’ link |
| **Input** | User enters old password (incorrectly), new password, and new password confirmation |
| **Expected Output** | Page displays ‘Old password was incorrect’ |
| **Actual output** | Page displays ‘Old password was incorrect’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC30 |
| **Purpose** | Test **VJF-TC30** – Test student login with Google successfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘Google login’ link and enters correct username and password for a Google account |
| **Expected Output** | User is redirected to student home page |
| **Actual output** | User is redirected to student home page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC31 |
| **Purpose** | Test **VJF-TC31** – Test student login with Google unsuccessfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘Google login’ link and enters incorrect username and password for a Google account |
| **Expected Output** | Page displays ‘The username or password you entered is incorrect’ |
| **Actual output** | Page displays ‘The username or password you entered is incorrect’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC32 |
| **Purpose** | Test **VJF-TC32** – Test student login with Panthermail (FIU) successfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘FIU login’ link and enters correct username and password for a Google (FIU) account |
| **Expected Output** | User is redirected to student home page |
| **Actual output** | User is redirected to student home page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC33 |
| **Purpose** | Test **VJF-TC33** – Test student login with Panthermail (FIU) unsuccessfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘FIU login’ link and enters incorrect username and password for a Google (FIU) account |
| **Expected Output** | Page displays ‘The username or password you entered is incorrect’ |
| **Actual output** | Page displays ‘The username or password you entered is incorrect’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC34 |
| **Purpose** | Test **VJF-TC34** – Test student login with LinkedIn successfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘LinkedIn login’ link and enters correct username and password for a LinkedIn account |
| **Expected Output** | User is redirected to student home page |
| **Actual output** | User is redirected to student home page |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC35 |
| **Purpose** | Test **VJF-TC35** – Test student login with LinkedIn unsuccessfully |
| **Test Setup** | User is a VJB site |
| **Input** | User clicks on ‘LinkedIn login’ link and enters incorrect username and password for a LinkedIn account |
| **Expected Output** | Page displays ‘Please correct the marked field(s) bellow’ |
| **Actual output** | Page displays ‘Please correct the marked field(s) bellow’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC36 |
| **Purpose** | Test **VJF-TC36** – Test student LinkedIn connect successfully |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User clicks on ‘LinkedIn connect’ link and enters correct username and password for a LinkedIn account |
| **Expected Output** | User is directed to profile page with complete information from LinkedIn |
| **Actual output** | User is directed to profile page with complete information from LinkedIn |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC37 |
| **Purpose** | Test **VJF-TC3**7– Test student LinkedIn connect unsuccessfully |
| **Test Setup** | User is logged in as student and at profile page |
| **Input** | User clicks on ‘LinkedIn connect’ link and enters incorrect username and password for a LinkedIn account |
| **Expected Output** | Page displays ‘Please correct the marked field(s) bellow’ |
| **Actual output** | Page displays ‘Please correct the marked field(s) bellow’ |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC38 |
| **Purpose** | Test **VJF-TC38**– Test Schedule Video Interview Functionality |
| **Test Setup** | User is logged in as employer and will search for a student by skill name |
| **Input** | User enters “PHP” in search box  Page displays students that have PHP listed as a skill  User clicks on a desired student to view his/her profile  User is taken to the student profile  User clicks the Video Interview button and the schedule video interview box shows up  The user enters the date and time for the interview and clicks submit  User logs off  The student user that the interview was scheduled for logins and clicks on the video interview notification section |
| **Expected Output** | A video Interview notification is displayed |
| **Actual output** | A video Interview notification is displayed |

|  |  |
| --- | --- |
| Test Case ID | VJF-TC39 |
| **Purpose** | Test **VJF-TC39** – Test Notification to Accept Video Interview Functionality |
| **Test Setup** | User is logged in as Student at home page |
| **Input** | User clicks on the Video Interviews notification section.  The notification interviews will toggle down. User picks the interview that he/she would like to accept.  User clicks on the accept button.  User logs out.  User logs in as an employer.  User clicks on the Video Interviews notification section.  The notification interviews will toggle down.  The user gets a new notification that the student accepted the scheduled interview. |
| **Expected Output** | An accept Interview notification should display |
| **Actual output** | An accept Interview notification should display |

## 7.3 Evaluation of Tests

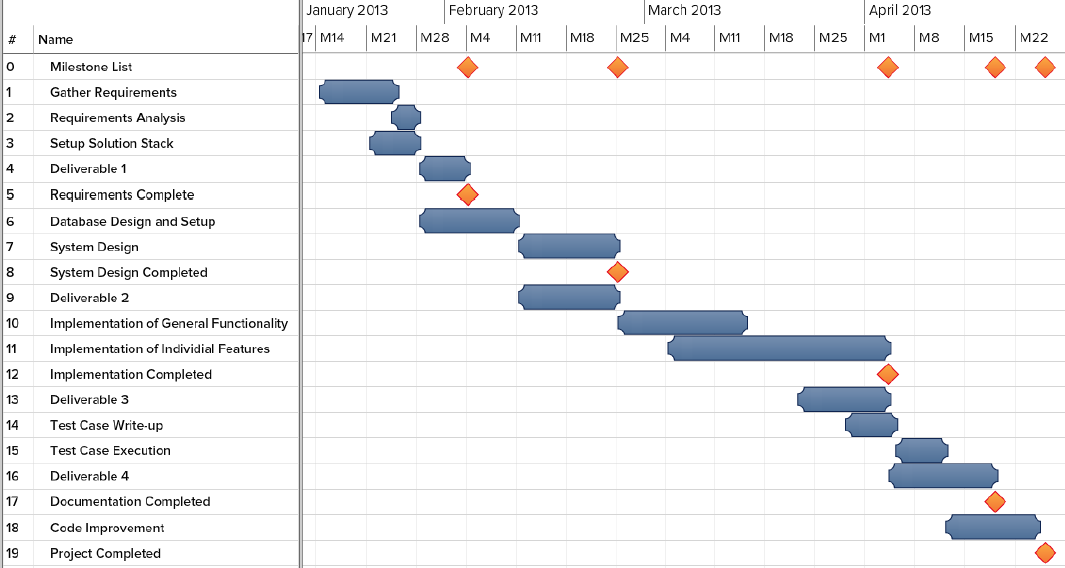
|  |  |
| --- | --- |
| Test Case ID | Result |
| VJF-TC1 | PASS |
| VJF-TC2 | PASS |
| VJF-TC3 | PASS |
| VJF-TC4x | PASS |
| VJF-TC5 | PASS |
| VJF-TC6 | PASS |
| VJF-TC7 | PASS |
| VJF-TC8 | PASS |
| VJF-TC9 | PASS |
| VJF-TC10 | PASS |
| VJF-TC11 | PASS |
| VJF-TC12 | PASS |
| VJF-TC13 | PASS |
| VJF-TC14 | PASS |
| VJF-TC15 | PASS |
| VJF-TC16 | PASS |
| VJF-TC17 | PASS |
| VJF-TC18 | PASS |
| VJF-TC19 | PASS |
| VJF-TC20 | PASS |
| VJF-TC21 | PASS |
| VJF-TC22 | PASS |
| VJF-TC23 | PASS |
| VJF-TC24 | PASS |
| VJF-TC25 | PASS |
| VJF-TC26 | PASS |
| VJF-TC27 | PASS |
| VJF-TC28 | PASS |
| VJF-TC29 | PASS |
| VJF-TC30 | PASS |
| VJF-TC31 | PASS |
| VJF-TC32 | PASS |
| VJF-TC33 | PASS |
| VJF-TC34 | PASS |
| VJF-TC35 | PASS |
| VJF-TC36 | PASS |
| VJF-TC37 | PASS |
| VJF-TC38 | PASS |
| VJF-TC39 | PASS |
| VJF-TC40 | PASS |
| VJF-TC41 | PASS |
| VJF-TC42 | PASS |

# 8. Glossary - define terms used in document, especially domain specific terms.

|  |  |
| --- | --- |
| Term | Meaning |
| Webrtc | Web with real time communication |
| Easyrtc | Framework of Webrtc |
| Selenium | Selenium automates browser |
| PHP | Server side scripting language for web development |
| MySql | Open source rotational database management system |
| Apache | HTTP server software program notable for playing a key role growth of the world wide. |
| Eclipse | A multi language software development composing a base workspace and an extensible plug-in system for customizing the environment. |

# 9. Appendix

## 9.1 Appendix A - Project schedule



## 9.2 Appendix B – All use cases with nonfunctional requirements

|  |  |
| --- | --- |
| 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 | 1. User clicks My Profile 2. User is redirected to his profile page 3. User clicks on the edit picture buttom 4. 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 | 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 resume button 4. 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 | 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 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:    1. For the employer:       1. the profile of the student participating in the interview.    2. For the student:       1. the profile of the employer conducting the interview. 3. 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 4. Once both parties are connect the video interview is started 5. 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 | 1. 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 |

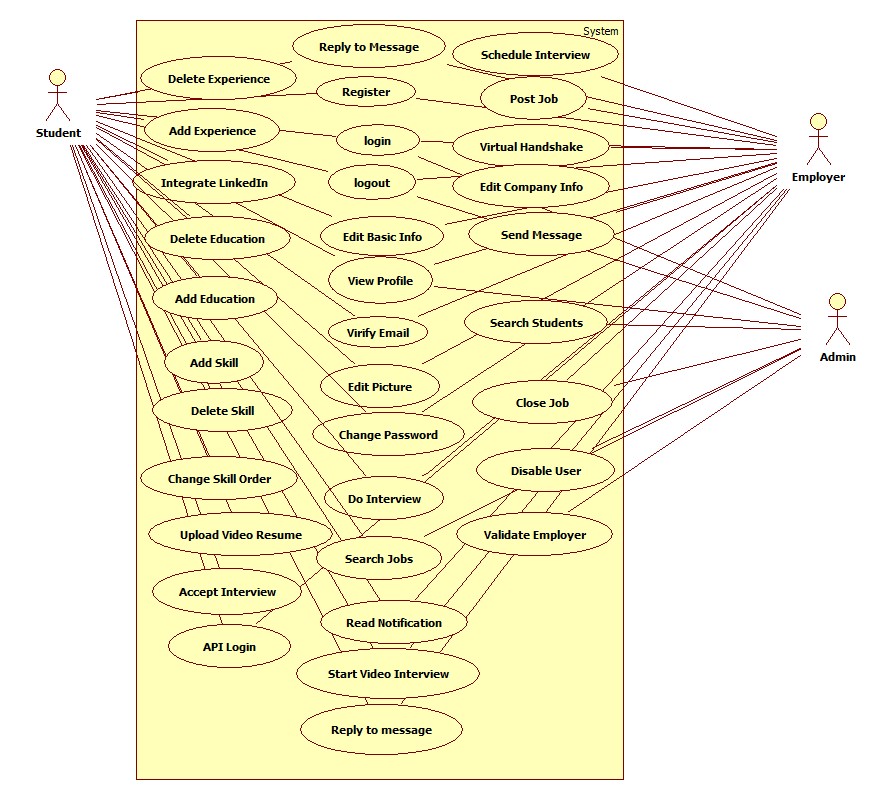
The system shall…

* **Use Case VJF 001 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 002 View Profile**
* Usability: Data displayed in profiles is easy to follow. Students are able to see their own and employer profiles only. Employers can see all student profiles.
* 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.
* **Use Case VJF 003 Edit Basic info**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 004 Verify Email**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 005 Login**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 006 Logout**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 007 Homepage**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0011 Add Education**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0012 Delete Education**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0013 Add Experience**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0014 Delete Experience**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 015 Change Password**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 017 Delete Skill**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 018 Change Skill Order**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 021 Accept Interview**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 020 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 three (3) seconds when connecting.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 009 Upload 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 one (5) seconds, depending on the file size.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 008 Upload Picture**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0038 Upload Video Resume**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 016 Add Skill**
* 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.
* **Use Case VJF 011 Add Education**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 013 Experience**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 019 Integrate 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 033 Apply to Job**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 023 Send 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 028 Search job**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 025 Post Job**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 030 Close a Job**
* Usability: The closing of a post is easy complete.
* 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.
* **Use Case VJF 0029 Search student**
* Usability: The search form is simple to submit.
* 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.
* **Use Case VJF 0022 View Student Profile**
* 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.
* **Use Case VJF 0026 Virtual Handshake**
* Usability: The virtual handshake form is easy complete.
* 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.
* **Use Case VJF 0032 Disable User**
* Usability: Disabling a user is simple to complete.
* 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.
* **Use Case VJF 0030 Administration Close Job**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0037 Validate an Employer Register**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0027 Edit Company Info**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0034 Read Notification**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0035 Schedule Interview**
* 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 one (1) second.
* Supportability: The system should be easy to maintain and make appropriate changes.
* **Use Case VJF 0024 Replay to Message**
* 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 one (1) second.

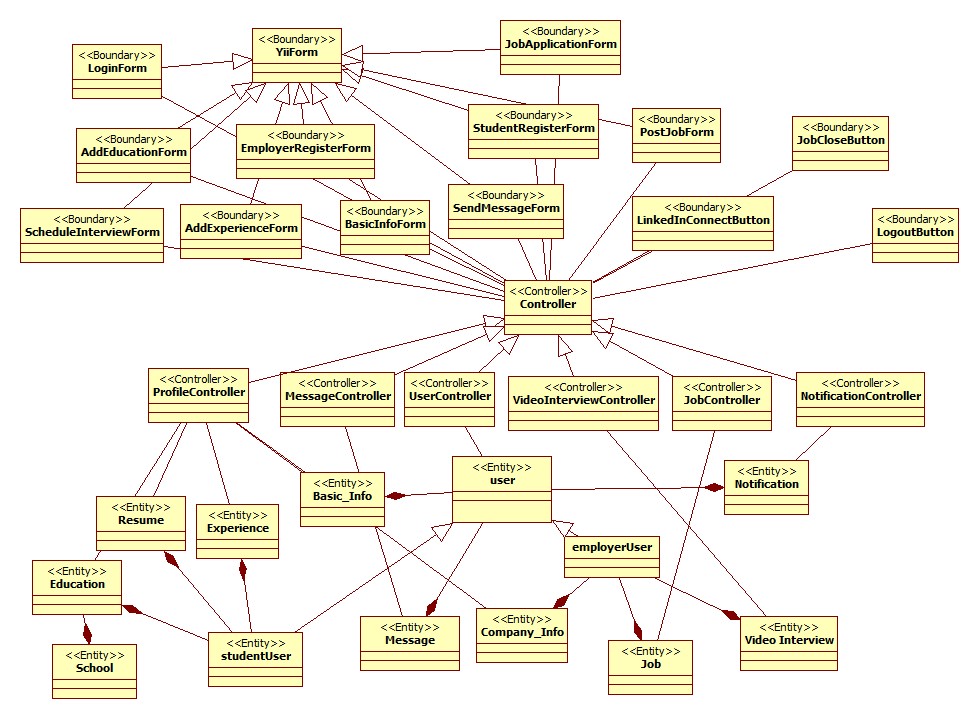
Supportability: The system should be easy to maintain and make appropriate changes.

## 9.3 Appendix C – User Interface designs

## 9.4 Appendix D – Analysis models

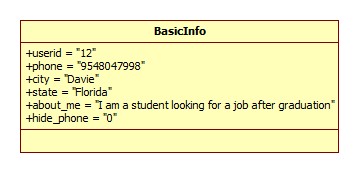


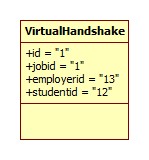
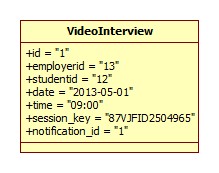
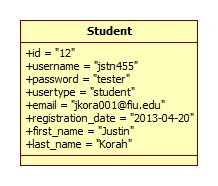
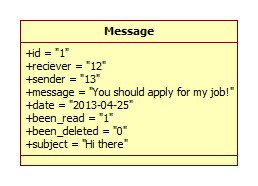
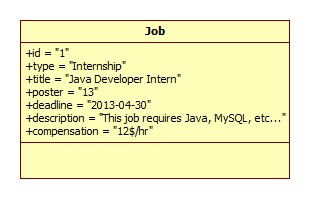
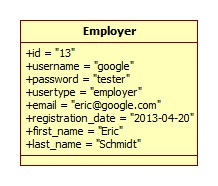
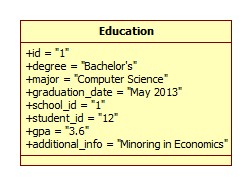
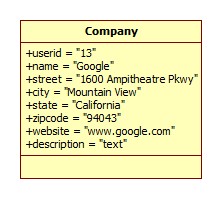
Use Case Diagram



Class diagram of the VJF System

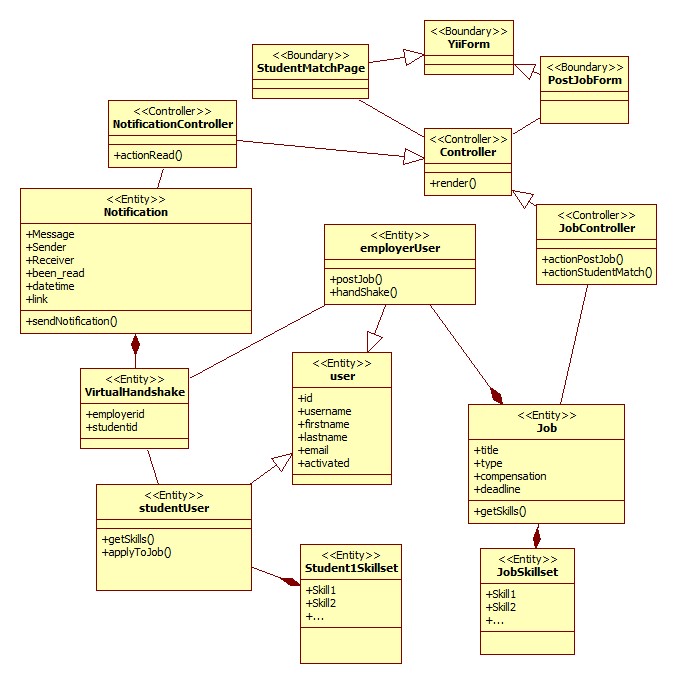
Below are object diagrams for entities found in the Virtual Job Fair system



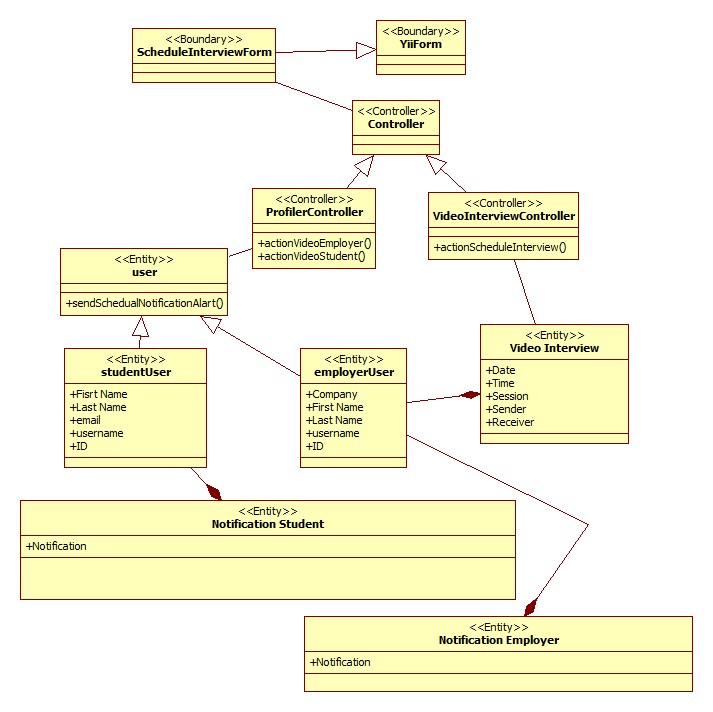


## 9.5 Appendix E – Design models

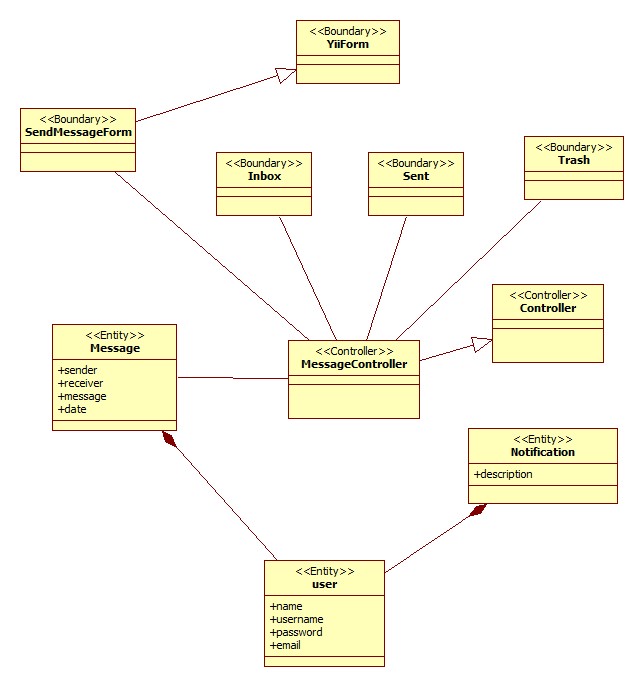
**Student Job Match Subsystem**



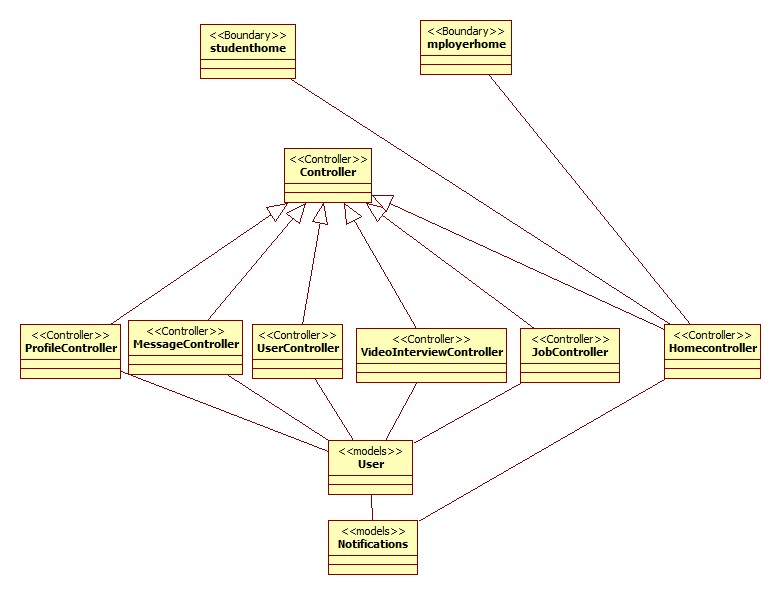
**Video Interview Subsystem**

****

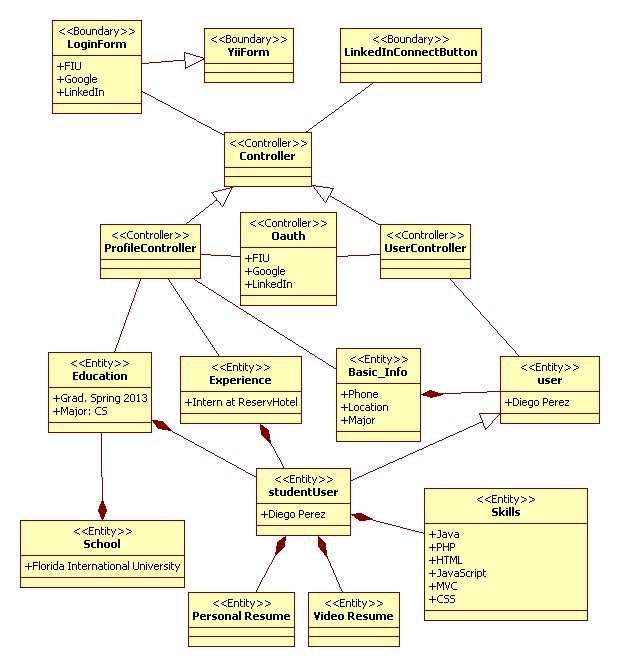
**Messaging Subsystem:**



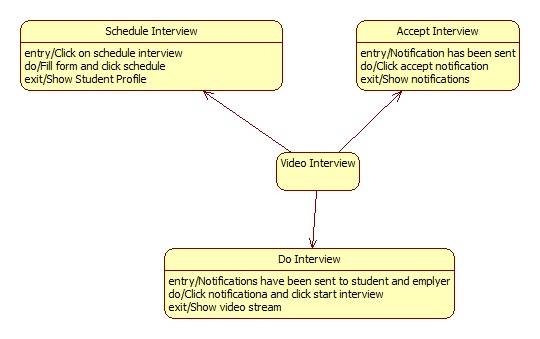
**Notification Subsystem:**



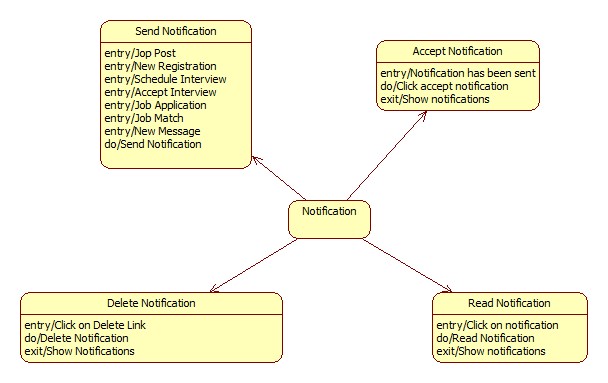
**Profile Creation Subsystem:**

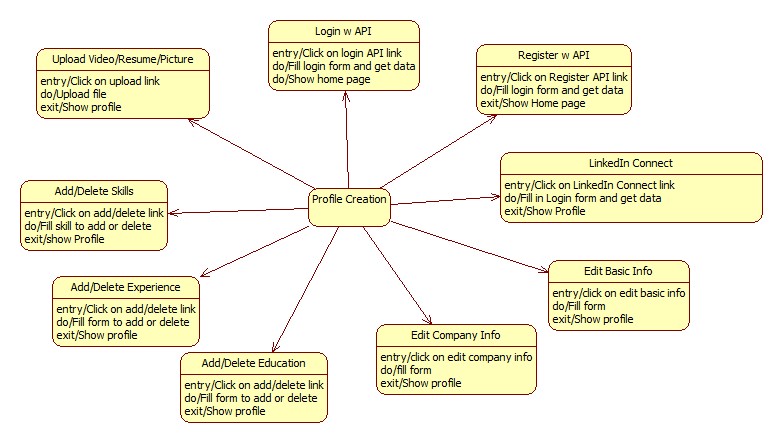


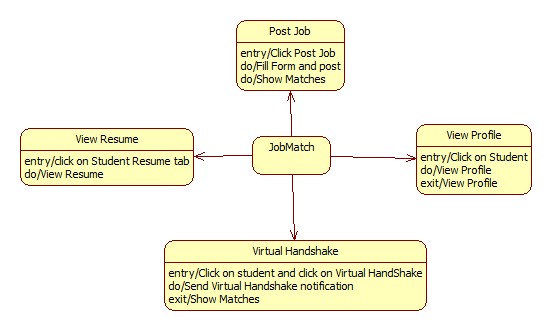
**Video Interview Subsystem:**



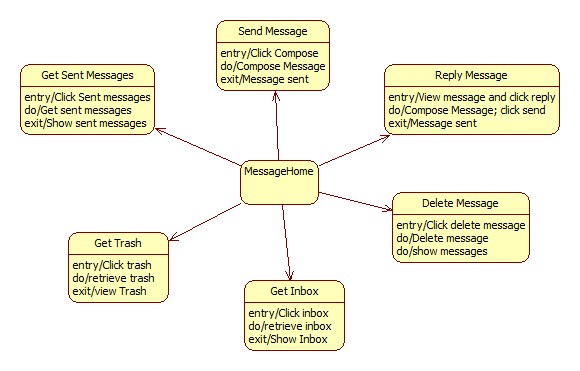
**Notification Subsystem:**



**Profile Creation Subsystem:**

**Job Match Subsystem:**

**Message Subsystem:**



## 9.6 Appendix F – Documented Class interfaces and constraints

The following are the interfaces for the main classes in each subsystem:

**Messaging Subsystem:**

Class: *MessageController*

**Method Signatures:**

@invokes message view

@precondition: user is logged in

@postcondition: user is redirected to message view

**public function actionIndex($target = null)**

@invokes compose view/ sends a message to a user

@precondition: user has requested the send message page / user has requested to send a message

@postcondition: user is redirected to the compose page / user is redirected to his/her inbox

**public function actionSend($username = null, $reply = null, $selfReply = null)**

@accesses message model to get all the received-message headers for a particular user

@precondition: user is on the message page

@postcondition: Contents given by the message model are passed to the view as json

**public function actionGetInbox()**

@accesses message model to get a received message for a particular user

@precondition: user has requested the message

@postcondition: Contents of a message are passed to the view as json

**public function actionGetMessage()**

@accesses message model to get all the sent-message headers for a particular user

@precondition: user has requested the messages sent

@postcondition: Contents given by the message model are passed to the view as json

**public function actionGetSent()**

@accesses message model to get all the trashed-message headers for a particular user

@precondition: user has requested the trash messages

@postcondition: Contents given by the message model are passed to the view as json

**public function actionGetTrash()**

@sets a message as read

@precondition: user has seen the message

@postcondition: Message headers no longer appear bold

**public function actionSetAsRead($id)**

@Sends selected messages to the trash

@precondition: user has selected some messages to be sent to the trash

@postcondition: Messages are sent to the trash.

**public function actionSentToTrash()**

@deletes some messages

@precondition: user has selected the messages to be deleted

@postcondition: the messages are removed from the database

**public function actionDeleteMessages()**

@enables autocomplete feature to display users

@precondition: user is populating the ‘TO’ field in the send a message page

@postcondition: user name is autopopulated

**public function actionAutoComplete()**

@prepares the username for the user who is going to receive the message

@precondition: user has sent a message

@postcondition: username of the user is extracted from string parameter

**public function actionGetReceiver($string)**

@specifies access rules

**public function accessRules**

@returns filter configuration for the MessageController

**public function filters**

**Notification Subsystem:**

Class: *NotificationController*

**Method signatures:**

@Send verification email to a new user to verify his/her new account.

@Precondition: user has been register

@Postcondition: user get a verification email

**public function** sendVerificationEmail()

@Send email with a new password for login.

@Precondition: user has been clicking change password/ forgotpassword

@Postcondition: user get a new password email

**public static function** *sendEmailWithNewPassword*($address, $password, $username)

@Send verification email to a new employer to verify his/her new account.

@Precondition: employer has been register

@Postcondition: employer get a verification email

**public static function** *sendEmployerVerificationEmail*($to)

@Send email notification alert

@Precondition: new notification has been crated

@Postcondition: user get an email with new notification alert

**public static function** *sendEmailNotificationAlart*($address, $to, $from, $message)

@send email message notification alert

@Precondition: a new message has been sent

@Postcondition: user get an email for new message

**public static function** *sendEmailMessageNotificationAlart*($address, $to, $from, $message)

@send email when student accepted schedule interview

@Precondition: a student click on the accept button

@Postcondition: an employer get an email that said that the student accepted the schedule interview

**public static function** *sendEmailEmployerAcceptingInterviewNotificationAlart*($address, $to, $from, $message)

@Send email when employer is interesting on student

@Precondition: an employer was clicking the handshake button

@Postcondition: a student get an email of hand shake alert

**public static function** *sendEmailStudentNotificationVirtualHandshakeAlart*($address, $to, $from, $message)

@Send notification to all student when new employer register and when there is a new job post

@Precondition: employer register/ new job has been post

@Postcondition: student get notification

**public static function** *sendAllStudentVerificationAlart*($id, $username, $email, $message, $link){

@Send notification to a student when employer schedule interview with him

@Precondition: employer schedual interview

@Postcondition: student get notification

**public static function** *sendSchedualNotificationAlart*($sender, $reciver, $message, $link)

@Send notification to employer after he schedule an interview

@Precondition: employer schedule interview

@Postcondition: employer get notification alert

**public static function** *sendEmployerNotificationAlart*($sender, $reciver, $message, $link, $level)

@send notification message alert

@Precondition: a new message has been sent

@Postcondition: user get an notification for new message

**public static function** *sendUserNotificationMessageAlart*($sender, $reciver, $link, $level

@Send notification when employer is interesting on student

@Precondition: an employer was clicking the handshake button

@Postcondition: a student get an notification of hand shake alert

**public static function** *sendUserNotificationHandshakeAlart*($sender, $reciver, $link, $message

@Send notification when a new job post and match student skills

@Precondition: an employer posted a new job

@Postcondition: a student get an notification of a new match job/skill alert

**public static function** *sendStudentNotificationMatchJobAlart*($sender, $reciver, $link, $message

@Send notification to admin when employer is register

@Precondition: an Employer registered

@Postcondition: an admin get notification alert

**public static function** *sendAdminNotificationNewEmpolyer*($employer, $admins, $link, $message)

@Send notification to employers when student accept schedule interview

@Precondition: student click accept

@Postcondition: employer get notification

**public static function** *sendEmployerNotificationStudentAcceptIntervie*($sender, $receiver)

**Profile Creation Subsystem:**

Class: *ProfileController*

**Method signatures:**

@View Student Profile

@Precondition: student clicks My Profile

@Postcondition: Profile view is displayed

**public function** actionView()

@View Employer Profile

@Precondition: employer click My Profile

@Postcondition: Profile view is displayed

**public function** actionViewEmployer()

@Go to Video Interview Page

@Precondition: Employer clicks Notification

@Postcondition: video interview page is displayed

**public function** actionVideoEmployer()

@Go to Video Interview Page

@Precondition: student clicks Notification

@Postcondition: video interview page is displayed

**public function** actionVideoStudent()

@Save student skills

@Precondition: student clicks save skills

@Postcondition: skills are saved and profile is displayed

**public function** actionSaveSkills()

@Delete education

@Precondition: student clicks delete education

@Postcondition: Profile view is displayed

**public function** actionDeleteEducation()

@Add Education

@Precondition: student enters data and clicks add education

@Postcondition: Profile view is displayed

**public function** actionAddEducation()

@View Profile

@Precondition: student click My Profile

@Postcondition: Profile view is displayed

@Delete Experience

@Precondition: student clicks delete experience

@Postcondition: Profile view is displayed

**public function** actionDeleteExperience()

@Add Experience

@Precondition: student enters information and clicks add experience

@Postcondition: Profile view is displayed

**public function** actionAddExperience(){

@Add Image

@Precondition: user clicks add picture and selects an image

@Postcondition: Profile view is displayed

**public function** actionUploadImage()

@Add video resume

@Precondition: student clicks upload video resume and selects video

@Postcondition: Profile view is displayed

**public function** actionUploadVideo()

@Add resume

@Precondition: student clicks add resume and selected file

@Postcondition: Profile view is displayed

**public function** actionUploadResume()

@Edit basic information

@Precondition: student clicks edit basic information and enters data

@Postcondition: Profile view is displayed

**public function** actionEditBasicInfo()

@Edit company info

@Precondition: Employer clicks edit company info and enters data

@Postcondition: Profile view is displayed

**public function** actionEditCompanyInfo()

@Integrate LinkedIn

@Precondition: student clicks LinkedIn connect and enters credentials

@Postcondition: Profile view is displayed

**public function** actionDemo()

@Login/Register with Google API

@Precondition: Student clicks on Google Link and enters credentials

@Postcondition: Home page is displayed

**public function** actionGoogleAuth()

@Login/Register with FIU account

@Precondition: student clicks on FIU link and enters credentials

@Postcondition: Home Page is displayed

**public function** actionFiuAuth()

**Job Matching Subsystem:**

Class: *JobController*

**Method signatures:**

@Post a Job

@Precondition: Employer fill in post form and click Post

@Postcondition: StudentMatch page is displayed

**public function** actionPost()

@Save job skills

@Precondition: User is on post job page

@Postcondition: Skills are associated with posted job

**public function** actionSaveSkills($jobid)

@Automatically fill skills when posting job

@Precondition: Job description is filled out

@Postcondition: Any relevant skills are added to skill list

**public function** actionQuerySkill($name)

@Apply to Job

@Precondition: student has not applied to job

@Postcondition: Student has applied to job, employer is notified

**public function** actionApply($jobid)

@Close Job Post

@Precondition: Post is open

@Postcondition: Post is closed, no more applications allowed

**public function** actionClose($jobid)

@Assigns skill rating for a student compared to a job

@Precondition: student and job have skills

@Postcondition: student is assigned a rating

**function** compare\_skills($jobskillmaps, $studentskillmaps)

@Match student with job

@Precondition: job is posted

@Postcondition: list of matching students is given

**public function** actionStudentMatch($jobid)

@Notifies student of employer interest

@Precondition: student has been matched for a job

@Postcondition: student is notified

**public function** actionVirtualHandshake($jobid, $studentid)

**Video Interview Subsystem:**

Class: *VideoInterviewController*

**Method signatures:**

@Schedules an interview

@Precondition: student is matched with a jon=b post

@Postcondition: student receives a notification

**public function** actionScheduleInterview()

## 9.7 Appendix G – Documented code for test drivers and stubs

Following are two codes test cases for the system. The complete testing code can be found in the testing folder along with the software solution.

// VJF-TC38

// Test schedule Interview

@Test

**public** **void** testScheduleInterview() {

**try** {

// go to login page

driver.get("http://srprog-spr13-01.aul.fiu.edu/JobFair/index.php/site/login");

delay.Del(2000);

driver.findElement(By.*id*("LoginForm\_username")).sendKeys("IBM2");

driver.findElement(By.*id*("LoginForm\_password")).sendKeys("tester");

delay.Del(3000);

driver.findElement(By.*className*("btn")).click();

delay.Del(2000);

// go to user

driver.get("http://srprog-spr13-01.aul.fiu.edu/JobFair/index.php/profile/student/user/dpere070@fiu.edu");

delay.Del(2000);

// click video interview

driver.findElement(By.*xpath*("//\*[@id='contactlinks']/div/a[1]")).click();

delay.Del(2000);

// enter date

driver.findElement(By.*xpath*("//\*[@id='VideoInterview\_date']")).click();

driver.findElement(By.*xpath*("//\*[@id='VideoInterview\_date']")).sendKeys("time");

delay.Del(2000);

List <WebElement> dates = driver.findElements(By.*xpath*("//\*[@class='ui-datepicker-calendar']//\*[@href='#']"));

delay.Del(1000);

dates.get(29).click();

delay.Del(1000);

driver.findElement(By.*id*("VideoInterview\_time")).sendKeys("3:28am");

delay.Del(2000);

driver.findElement(By.*xpath*("//\*[@class='modal-footer']/button")).click();

*assertTrue*(driver.getCurrentUrl().contains("dpere070@fiu.edu"));

//driver.findElement(By.xpath("//\*[@id='yw0']/li[5]/a")).click(); // logout

} **catch** (Exception e) {

e.printStackTrace();

}

}

// VJF-TC34

// login with linkedin succesfully

@Test

**public** **void** testLoginLinked1() {

**try** {

// go to login page

driver.get("http://srprog-spr13-01.aul.fiu.edu/JobFair/index.php/site/login");

delay.Del(2000);

driver.findElement(By.*xpath*("//\*[@id='altlogin']/a[3]")).click();

delay.Del(3000);

// enter login information for LinkedIn and click login

driver.findElement(By.*xpath*("//\*[@name='session\_key']")).sendKeys("xxxx@gmail.com");

driver.findElement(By.*xpath*("//\*[@name='session\_password']")).sendKeys("xxxxxxxx");

driver.findElement(By.*xpath*("//\*[@name='authorize']")).click();

delay.Del(2000);

// student is redirected to student home page

*assertTrue*(driver.getCurrentUrl().contains("home/studenthome"));

driver.findElement(By.*xpath*("//\*[@id='yw0']/li[5]/a")).click(); // logout

} **catch** (Exception e) {

e.printStackTrace();

}

}

## 9.8 Appendix H – Diary of meeting and tasks for the entire semester

|  |  |  |
| --- | --- | --- |
| Date | Activities | Participants |
| 01/08/2013 | Outlined the problem and potential solution | Every member of the team |
| 01/12/2013 | Decided on the solution and associated features to individual team members | Every member of the team |
| 01/15/2013 | Identified the main user requirements at a high level | Every member of the team |
| 01/19/2013 | Identified the HW/SW requirements and came up with alternative solutions/features to the overall problem | Every member of the team |
| 01/22/2013 | Identified the main tasks and milestones. Also worked on 1st deliverable documentation | Every member of the team |
| 01/26/2013 | Worked on 1st deliverable documentation | Every member of the team |
| 01/29/2013 | Worked on the database design | Every member of the team |
| 02/02/2013 | Worked on Deliverable 2 Project Plan. Decided on work breakdown | Every member of the team |
| 02/05/2013 | Finished the database design | Every member of the team |
| 02/09/2013 | Identified all the Use Cases. | Every member of the team |
| 02/12/2013 | Started working on the system design, identifying the main classes required | Every member of the team |
| 02/16/2013 | Finished System Design. Worked on the Dynamic Model of the System (sequence diagrams) | Every member of the team |
| 02/19/2013 | Finished 2nd Deliverable | Every member of the team |
| 02/23/2013 | Worked on the implementation of general system functionality | Every member of the team |
| 02/26/2013 | Worked on the implementation of general system functionality | Every member of the team |
| 03/02/2013 | Worked on the implementation of general system functionality | Every member of the team |
| 03/05/2013 | Worked on the implementation of individual features | Every member of the team |
| 03/09/2013 | Worked on the implementation of individual features | Every member of the team |
| 03/12/2013 | Worked on the implementation of individual features | Every member of the team |
| 03/16/2013 | Worked on the implementation of individual features | Every member of the team |
| 03/19/2013 | Worked on third deliverable documentation | Every member of the team |
| 03/23/2013 | Worked on third deliverable documentation | Every member of the team |
| 03/26/2013 | Completed third deliverable documentation.  Worked on the implementation of individual features | Every member of the team |
| 03/30/2013 | Polish user interfaces, compile bug list | Every member of the team |
| 04/02/2013 | Polish user interface, begin fixing bug list | Every member of the team |
| 04/06/2013 | Polish user interface, introduction to Selenium test cases | Every member of the team |
| 04/09/2013 | Start selenium test cases | Every member of the team |
| 04/13/2013 | Continue selenium test cases, start final deliverable | Every member of the team |
| 04/16/2013 | Review presentation, Review documentation, start user manual | Every member of the team |
| 04/20/2013 | Review presentation, Review documentation | Every member of the team |
| 04/21/2013 | Finalize presentation, finalize all deliverables | Every member of the team |

|  |  |  |
| --- | --- | --- |
| Date | Activities | Participants |
| 01/08/2013 | Outlined the problem and potential solution | Every member of the team |
| 01/12/2013 | Decided on the solution and associated features to individual team members | Every member of the team |
| 01/15/2013 | Identified the main user requirements at a high level | Every member of the team |
| 01/19/2013 | Identified the HW/SW requirements and came up with alternative solutions/features to the overall problem | Every member of the team |
| 01/22/2013 | Identified the main tasks and milestones. Also worked on 1st deliverable documentation | Every member of the team |
| 01/26/2013 | Worked on 1st deliverable documentation | Every member of the team |

# 10. References

In order to carry out the Login/Register with an outside API, the Oauth API was integrated from both Google and LinkedIn.

EasyRTC was used in order to complete the video interview functionality.

The Yii framework was extensively for coding.