Design Document

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.*

Abstract

*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.*

*This design documents lays out the technical requirements for the Virtual Job Fair application. The information on this document is dedicated for a technically inclined audience that is familiar with software development concepts. The document outlines the tools used for development, the overall system design, the detailed system design, and system diagrams.*

Contents

[1. Introduction 5](#_Toc354467269)

[1.2 Problem definition 5](#_Toc354467270)

[1.3 Design methodology 6](#_Toc354467271)

[1.4 Definitions 7](#_Toc354467272)

[1.5 Overview of document 7](#_Toc354467273)

[2. System Design 7](#_Toc354467274)

[2.1 Overview 8](#_Toc354467275)

[2.2 Subsystem Decomposition 9](#_Toc354467276)

[2.3 Hardware and Software Mapping 13](#_Toc354467277)

[2.4 Persistent Data Management 13](#_Toc354467278)

[2.5 Security 21](#_Toc354467279)

[3. Detailed Design 23](#_Toc354467280)

[3.1 Overview 23](#_Toc354467281)

[3.2 Static model 25](#_Toc354467282)

[3.3 Dynamic model 30](#_Toc354467283)

[3.4 Code Specification 34](#_Toc354467284)

[4. Glossary 44](#_Toc354467285)

[5. Appendix 45](#_Toc354467286)

[5.1 Appendix A - Use case diagrams 45](#_Toc354467287)

[5.2 Appendix B - Use cases being implemented 45](#_Toc354467288)

[5.3 Appendix C – Documented class interfaces 67](#_Toc354467289)

[5.4 Appendix D - Diary of meeting and tasks 70](#_Toc354467290)

[6. References 71](#_Toc354467291)

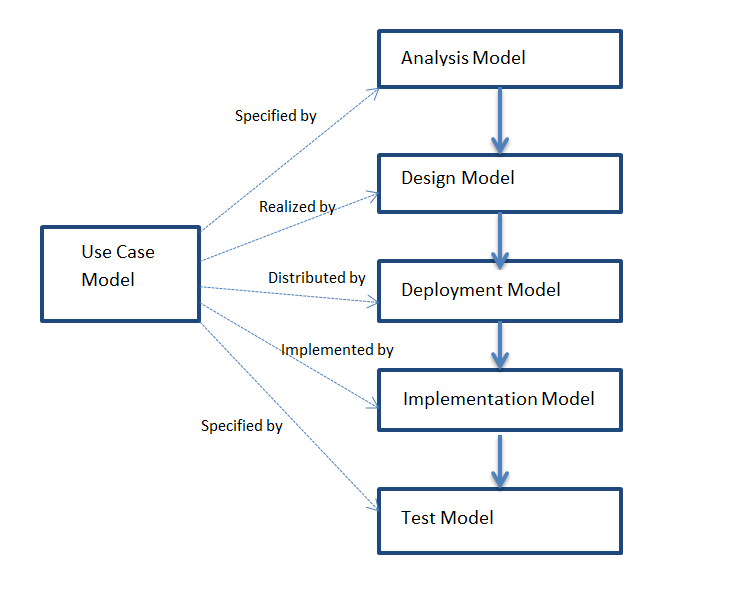
# 1. Introduction

The introduction provides the reader with background information regarding the project and some aspects of the design process. It starts off by giving the overall problem the project is trying to solve. It follows by giving a brief look into the design methodology used throughout the project and how it helps us in the design process. A brief description of the design methodology is also needed in terms of what kinds of models are needed to represent the design of the product.

## 1.2 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.

## 1.3 Design 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.

## 1.4 Definitions

|  |  |
| --- | --- |
| Term | Meaning |
| API | Application Programmer Interface |
| VJF | Virtual Job Fair |
| UML | Unified Modeling Language |
| SCIS | School of Computer and Information Sciences |

## 1.5 Overview of document

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

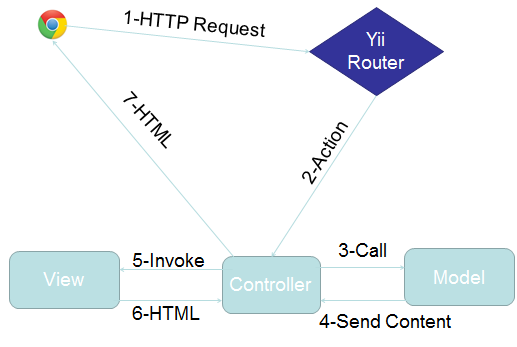
Section 2 of the document addresses the System Design. This section will explain the decomposition of the system. Moreover, this section will also go into the software and hardware mapping, explaining how the system is connected together. This is represented by a diagram which helps the reader visualize the idea. This section will also cover what was implemented for system security. Section 3 of this document covers the detailed design of the system. The reader can expect diagrams and explanation of pieces of code. Section 4 is a glossary where definition to specific terms used can be found. Section 5 consists of appendixes which contain mostly diagrams. Finally, sections 6 are the references used in this document.

# 2. System Design

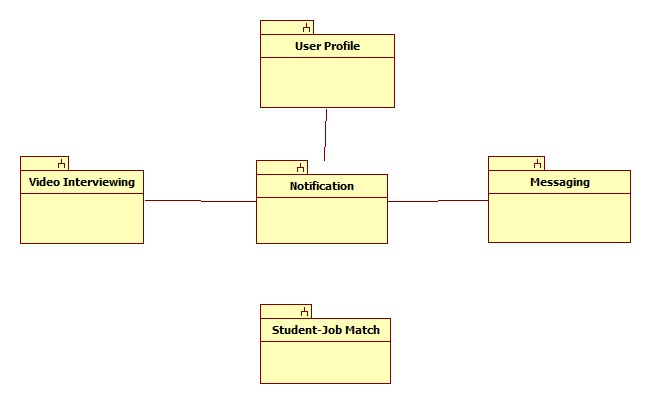
In order to break down the system into manageable parts, it was necessary to identify the main features that the system is going to support and how they differ in terms of technology used and functionality. Following this phase, it became evident that the system could be decomposed into 5 main subsystems,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

## 2.1 Overview

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:



## 2.2 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**

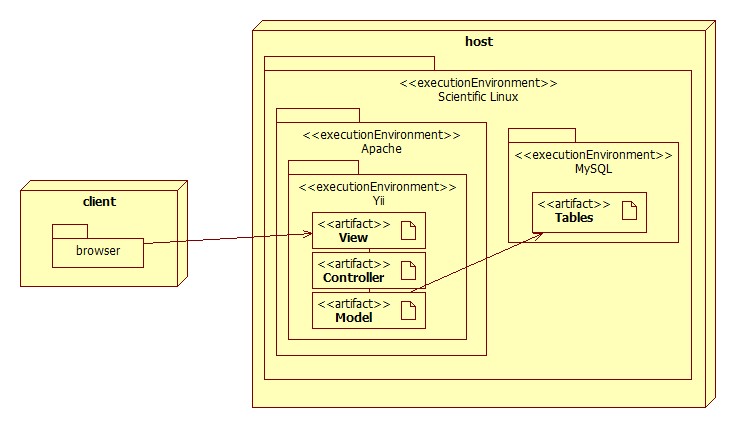
Making the right connection is what this web application is all about. Therefore, an efficient algorithm to match students to job openings is very important. The student job match subsystem takes care of matching students with the required skills to job post, making the job of the recruiters easier,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

## 2.3 Hardware and Software Mapping

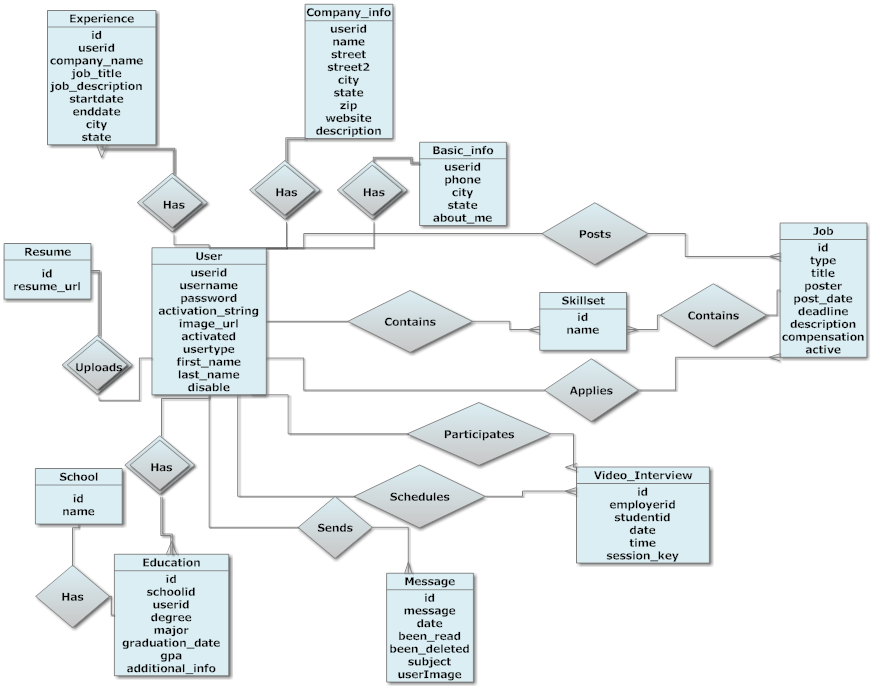


The deployment diagram shown above represents the hardware and software mapping in the Virtual Job Fair system. The main components of the system are the Apache and MySQL environment hosted on a Linux operating system. The Yii framework environment is using apache to execute, and contains our various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

## 2.4 Persistent Data Management

Data is 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, 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.

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 |  |  |

## 2.5 Security

**Security Features**

* User password will be hashed in the database.

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

* Administrator will be able to disable users and delete jobs

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

* Yii access control rules

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

* Cross-site Scripting Prevention

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

* Secure registration process

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

**Privacy**

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

# 3. Detailed Design

This chapter focuses on the behavior and structure of each subsystem. The static model contains a description of design patters used for the subsystems. The dynamic model presents the state machine diagrams for each of the subsystems and describes the main algorithms used in the problem solution.

## 3.1 Overview

**Video Subsystem**

The video subsystem structure is built on top of easy-RTC framework which relies on Web-RTC technology to deliver a browser to browser communication experience to the end user. This is what makes possible to have video communication without the need to rely on a dedicated media server. All these configurations are running on port 8080 of the server by node.js which takes care of mapping users to each other, so that the communication can be established between the desired users.

**Messaging Subsystem**

The messaging subsystem structure relies heavily on AJAX to reduce the need of waiting for whole page loads for tasks such as accessing inbox, sent messages, replying to messages, etc. This feature is key to a user friendly interface and a dynamic user interaction, providing users with an almost desktop-like experience when interacting with their messages. JQuery is also heavily used to manage the data returned by the Ajax calls.

**Student job match subsystem**

The student job match subsystem structure relies on the database relationships. It compares the skills related to a specific user to the skills related to a job post and matches them. It does not only match skills to skills blindly, but it looks at skills presence. The interface of this subsystem allows students to build hierarchies with their skills, by putting the skills they are most familiar with on top and those they are not very good at on the bottom. This allows the subsystem to effectively predict who will be a good candidate for a job based on their skills hierarchy.

**Notification Subsystem**

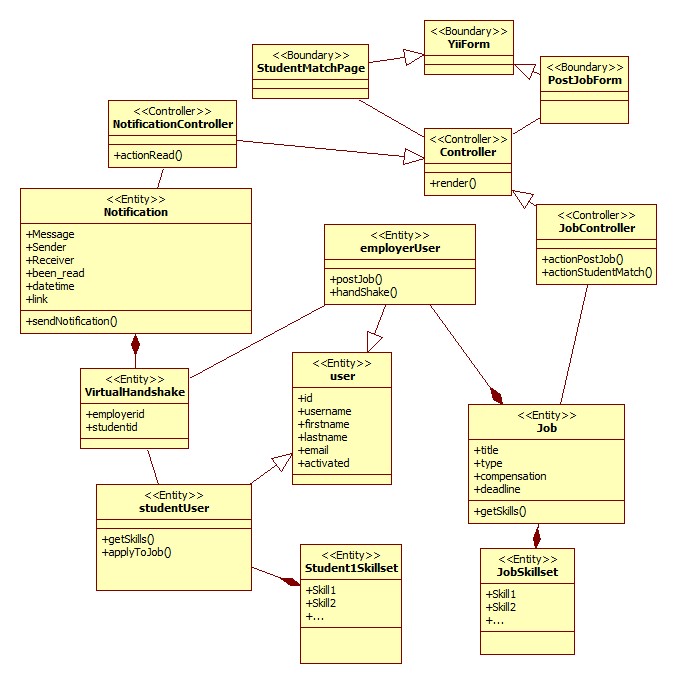
The notification subsystem behaves as the information control for the whole system. It allows users to stay informed of current events and news. Whenever a user interacts somehow with other user the notification subsystem is responsible for tracking that interaction and notifying the respective users. The notification subsystem structure is based on the database relationship, which allows the subsystem to easily build notification based on the user relationships propagation through the database.

**Profile creation subsystem**

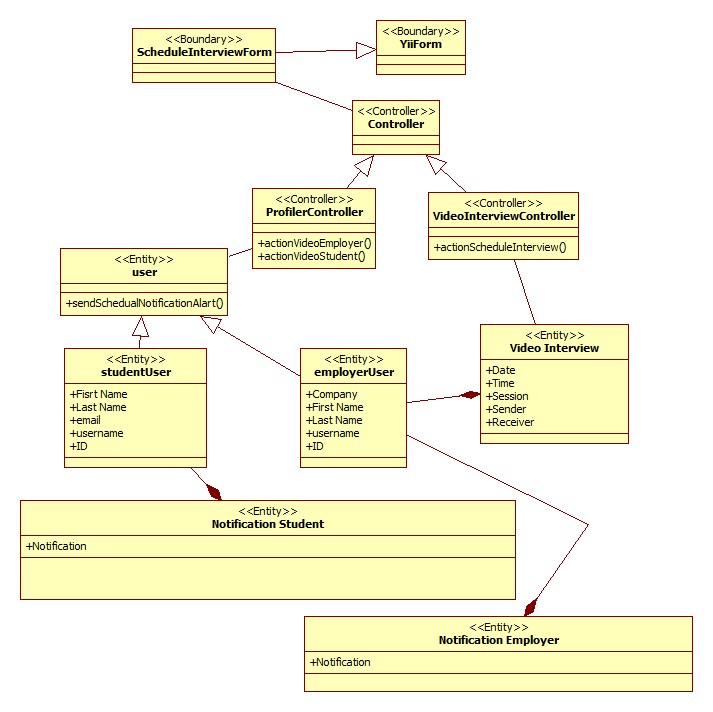
The profile creation subsystem structure is built around APIs of third party website such as LinkedIn, to allow users to easily populate their profiles with the data they already have in their LinkedIn account. In addition the profile creation subsystem extends to the credentialing of users, by allowing users to authenticate through third party websites such as Google, LinkedIn and FIU Panther Mail. This is all possible by the various API calls to the outside websites and it provides a reliable and secure way of authentication.

## 3.2 Static model

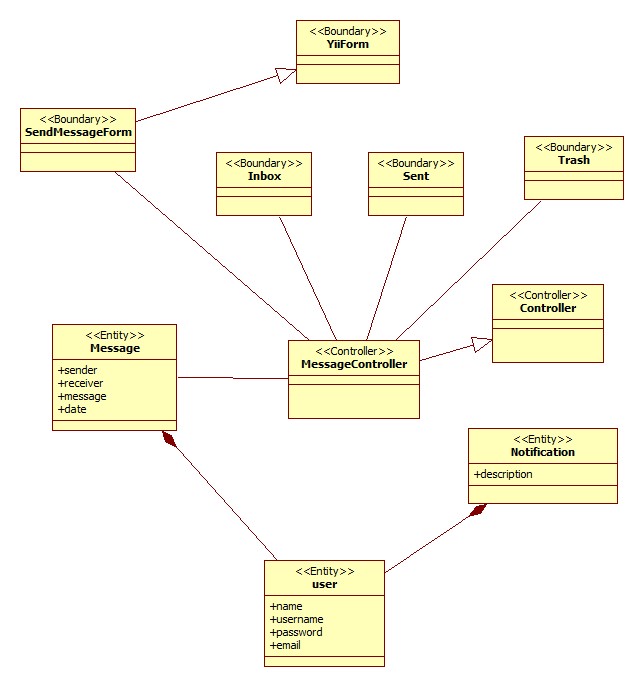
**Student Job Match Subsystem**



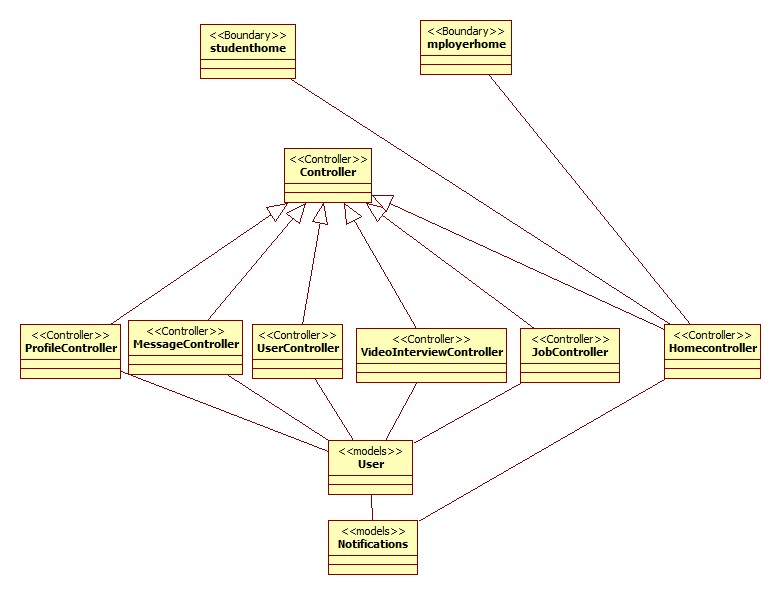
**Video Interview Subsystem**

****

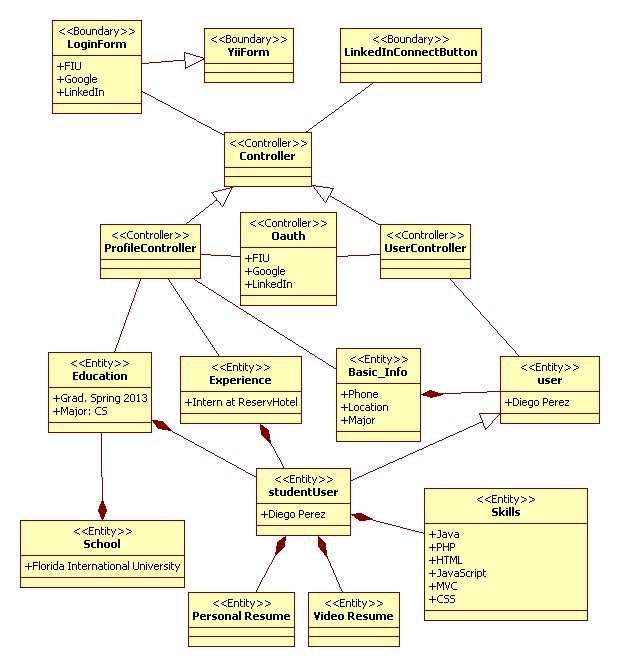
**Messaging Subsystem:**



**Notification Subsystem:**

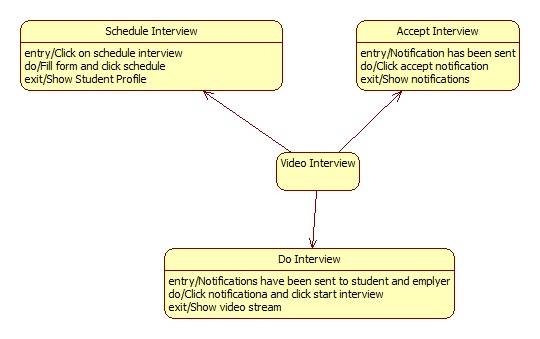


**Profile Creation Subsystem:**

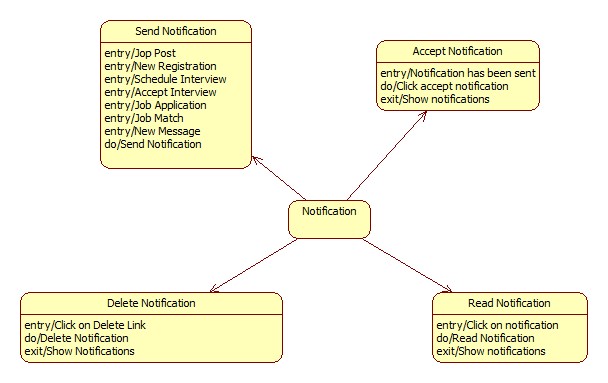


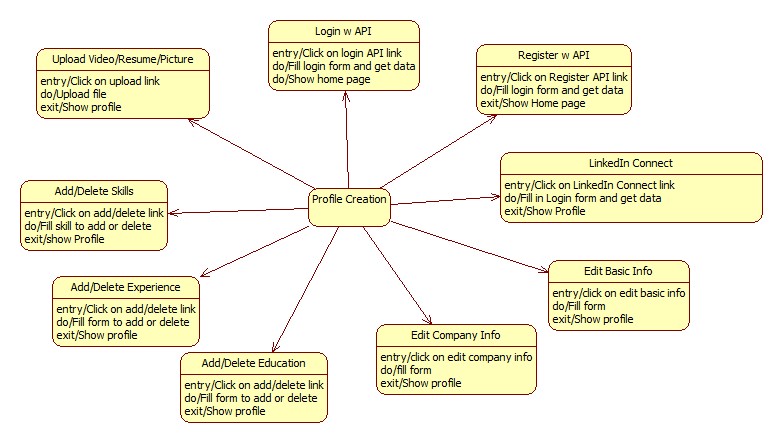
## 3.3 Dynamic model

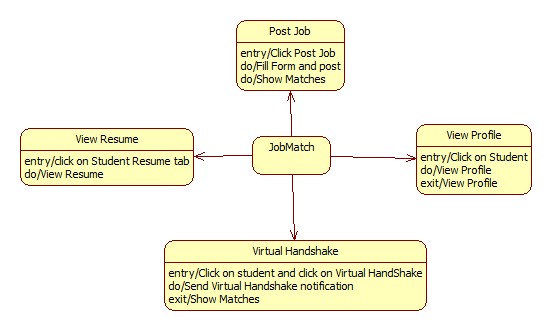
**Video Interview Subsystem:**



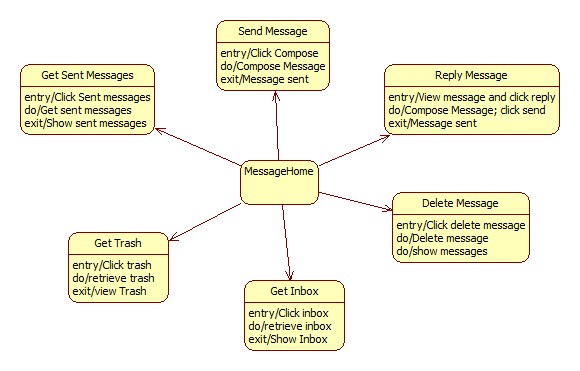
**Notification Subsystem:**



**Profile Creation Subsystem:**

**Job Match Subsystem:**

**Message Subsystem:**



## 3.4 Code Specification

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

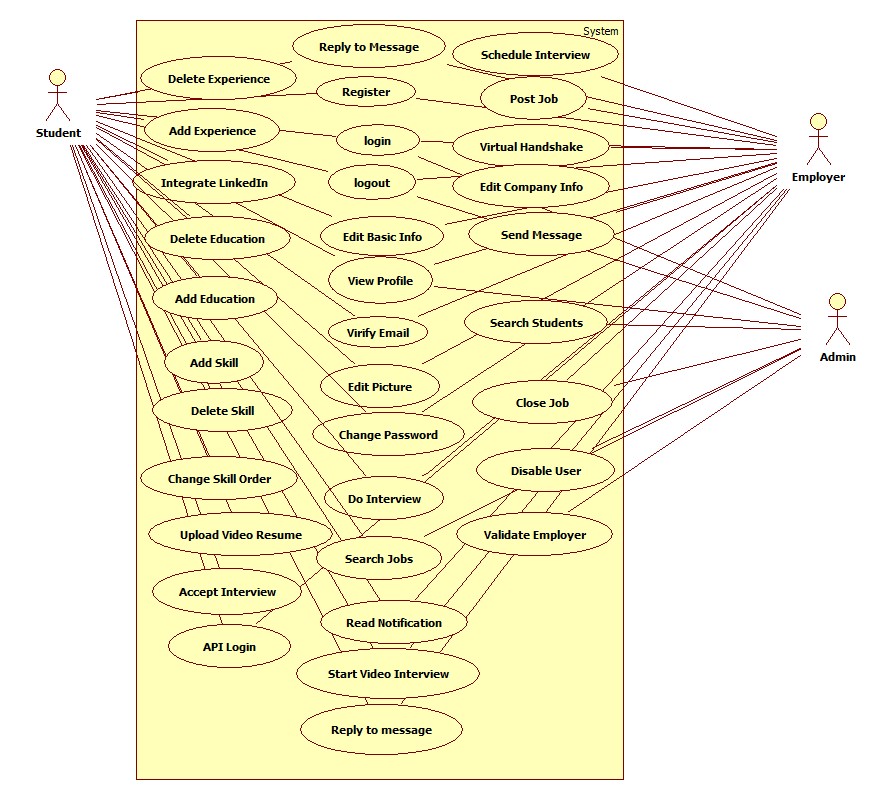
**public function** actionScheduleInterview()

# 4. Glossary

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

# 5. Appendix

## 5.1 Appendix A - Use case diagrams



## 5.2 Appendix B - Use cases being implemented

|  |  |
| --- | --- |
| 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 |

## 5.3 Appendix C – Documented class interfaces

**MessageController:**

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

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

**public function actionGetInbox()**

**public function actionGetMessage($id)**

**public function actionGetSent()**

**public function actionGetTrash()**

**public function actionSetAsRead($id)**

**public function actionSentToTrash()**

**public function actionDeleteMessages()**

**public function actionAutoComplete()**

**public function actionGetReceiver($string)**

**public function accessRules()**

**public function filters()**

**User Model – Notification subsystem:**

**public function** sendVerificationEmail()

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Profile Creation Subsystem:**

Class: *ProfileController*

**public function** actionView()

**public function** actionViewEmployer()

**public function** actionVideoEmployer()

**public function** actionVideoStudent()

**public function** actionSaveSkills()

**public function** actionDeleteEducation()

**public function** actionAddEducation()

**public function** actionDeleteExperience()

**public function** actionAddExperience(){

**public function** actionUploadImage()

**public function** actionUploadVideo()

**public function** actionUploadResume()

**public function** actionEditBasicInfo()

**public function** actionEditCompanyInfo()

**public function** actionDemo()

**public function** actionGoogleAuth()

**Job Matching Subsystem:**

Class: *JobController*

**public function** actionPost()

**public function** actionSaveSkills($jobid)

**public function** actionQuerySkill($name)

**public function** actionApply($jobid)

**public function** actionClose($jobid)

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

**public function** actionStudentMatch($jobid)

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

**Video Interview Subsystem:**

Class: *VideoInterviewController*

**public function** actionScheduleInterview()

## 5.4 Appendix D - Diary of meeting and tasks

|  |  |  |
| --- | --- | --- |
| Date | Activities | Participants |
| 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 |

# 6. 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.