***Enhanced Job and Candidate Application***

***Design Specification***

C:\Users\Simul\Desktop\Winter 2015\CSC 4996\MVC App\User Login\User Login\Content\Images\computech logo.png



Version: 1.0

03/10/2015

**Prepared by:**

Simul Kadakia

Wesley Trescott

Gagandeep Singh

Contents

[1. Introduction 1](#_Toc413715937)

[1.1 Purpose 1](#_Toc413715938)

[1.2 Scope 1](#_Toc413715939)

[1.3 Definitions, Acronyms and Abbreviations 1](#_Toc413715940)

[1.4 References 1](#_Toc413715941)

[2. Assumptions / Constraints / Standards 2](#_Toc413715942)

[2.1 Design Constraints: 2](#_Toc413715943)

[2.2 Assumptions and Dependencies 2](#_Toc413715944)

[2.3 Components: 2](#_Toc413715945)

[3. Architecture Design 3](#_Toc413715946)

[3.1 Logical View 3](#_Toc413715947)

[3.2 Hardware Architecture 3](#_Toc413715949)

[3.3 Software Architecture 4](#_Toc413715950)

[3.3 Security Architecture 4](#_Toc413715951)

[3.4 Communication Architecture 5](#_Toc413715952)

[3.5 Performance 5](#_Toc413715953)

[4. System Design 6](#_Toc413715954)

[4.1 Use-cases 6](#_Toc413715955)

[4.1.1 User class – Job seeker 6](#_Toc413715956)

[4.1.2 User class – Admin 12](#_Toc413715957)

[4.2 Sequence Diagram 14](#_Toc413715958)

[4.3 Data Flow Diagrams 16](#_Toc413715959)

[4.4 Database Design 20](#_Toc413715960)

[4.5 Application Program Interface 20](#_Toc413715961)

[4.5.1 Google Authentication 20](#_Toc413715962)

[4.5.2 Software Class Diagrams 21](#_Toc413715963)

[4.6 User Interface Design 22](#_Toc413715964)

[4.6.1 Home Page 22](#_Toc413715965)

[4.6.2 Register 23](#_Toc413715966)

[4.6.3 Login 25](#_Toc413715967)

[4.6.4 Profile Page 26](#_Toc413715968)

[4.6.5 Job Search Page 27](#_Toc413715969)

[4.6.6 Job Details Page 27](#_Toc413715970)

[5. Product Design Specification Approval 28](#_Toc413715971)

# 1. Introduction

This section gives a scope description and overview of everything included in this document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

## Purpose

The purpose of this Software Design Specification (SDS) document is to provide a detailed description of the design framework of the ‘Enhanced Job and Candidate Application’ system. It will also provide specific information about the input and desired output, software and hardware architecture, database, security, and sequence diagrams and test cases pertaining to the application.

## Scope

Enhanced Job and Search Candidate is an application developed by three students at Wayne State University for Computech Corporation. The goal of the application is to provide a web application service to job seekers to search for available jobs at Computech Corporation and apply to those that they are interested in. The application will provide filtering options to reduce the jobs displayed based on certain criteria. Job seekers will be able to store their profile information and resume path (resume will be stored in a folder) in the database, which can be used when they are applying for a job using a web interface. The application will also provide functionalities to an admin user to deactivate and delete users abusing the system.

## 1.3 Definitions, Acronyms and Abbreviations

* EJCA - Enhanced Job and Candidate Application
* User – Job seeker who uses the application
* Admin – Admin/administrator who manages the users
* Admin portal - Part of the web application that provides special facilities to Admin
* Front End - The part of the application the user interacts with
* Back End – The part of application that manages data and is managed by developers.
* UI – User Interface which is the front end of the application
* Server –Machine that will host the web application as well as database.

## References

“Using OAuth 2.0 to Access Google APIs.” *Google Developers*. Google Corporation, 21 November 2014. Web. 10 March 2015.

*Additional helpful resources not directly referenced in this document:*

* Microsoft ASP.NET MVC - <http://www.asp.net/mvc>
* Microsoft SQL Server - <https://msdn.microsoft.com/en-us/sqlserver/aa336270.aspx>
* Razor - <http://www.asp.net/web-pages/overview/getting-started/introducing-razor-syntax-(c)>
* JQuery - <http://jquery.com/>
* Bootstrap – <http://getbootstrap.com>

# 2. Assumptions / Constraints / Standards

## 2.1 Design Constraints:

As a web application, the largest design constraint is the differences in the rendering of the user interface based on the type of device used. End users will use the application from a modern internet browser such as Safari, Google Chrome, or Internet Explorer.

## 2.2 Assumptions and Dependencies

Assumptions to properly use the application include:

* Internet connection with enough bandwidth (about 1 to 2 mbps) to fully render all web application pages
* A modern internet browser with an up to date JavaScript engine and support for HTTP cookies to remember returning user logins

## 2.3 Components:

* **Microsoft ASP.NET MVC framework:** EJCA will be developed using MVC, which is a software [architectural pattern](http://en.wikipedia.org/wiki/Architectural_pattern) for implementing [user interfaces](http://en.wikipedia.org/wiki/User_interface). It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user
* **Microsoft SQL Server:** EJCA will use MS SQL server database which will store users, jobs and admin information.
* **Razor:** Razor is a view engine which will embed server-based code on the web page.
* **JQuery:** EJCA will be using JavaScript library for more user and mobile friendly EJCA.
* **Bootstrap:** EJCA needs to be compatible on mobile browsers along with traditional desktop and bootstrap framework will enable to use the same layout on multiple platforms.

# 3. Architecture Design

This section details the architectural designs of the various components of the (currently under development) EJCA.

## 3.1 Logical View

Below is a 4+1 architectural logic view designed to orient those interested in the web application’s logical design to the sections of this document dealing with class, development, process, and physical views of the system architecture. Following are specific subsections on the hardware, software, security, and communication architectures of the system, as well as a section detailing the system architecture’s impact on the web application’s performance.

## 

## 3.2 Hardware Architecture

The hardware architecture of the EJCA was given by the client and consists of two types of hardware interfaces: a webserver and the user’s machine. The webserver is the physical machine hosting the site, including the server software it runs as well as the database. The server to be used is a Windows Server 2008 R2 Standard with 16 GB of RAM, and the software it runs is an IIS 7.5 webserver and a Microsoft SQL Server 2008 R2 database. This piece of hardware is owned by Computech and is used to receive HTTP requests and provide HTTP responses, ensuring the constant availability of the application. Additionally, each user of the application will access it using his or her own machine, whether that be in a desktop environment, or through a laptop, smartphone, or tablet. The desktop or laptop devices supported are limited to any device running a modern internet browser with an up to date JavaScript engine and support for HTTP cookies, such as the latest versions of Google Chrome or Safari. The tablet and smartphone devices supported are the iPhone 6 and iPad Air running the iOS 8.1.3 operating system and the Samsung Galaxy S5 running the Android v5.0 Lollipop operating system. Below is a diagram showing an overview of the system hardware architecture.



## 3.3 Software Architecture

EJCA runs on the Model View Controller (MVC) software architecture, a design pattern offered as part of the Microsoft ASP.NET framework. Since this architecture provides a simple three-tier system for displaying pages and managing data, it is ideal for the needs of EJCA. Being a web application, EJCA will also require a web service, which will run on a web server. The EJCA has three controllers, which provide the logic and data processing functionality of the web application. They are the Home controller, handling the logic of the site’s homepage, the User controller, handling the logic of all user-related activities, and the JobSearch controller, handling the logic of searching for jobs. Each controller has views it controls, which are displayed to the user as .cshtml web pages. Additionally, the application uses three user model C# classes to interface with the database, accessing and manipulating user data. Also, the application makes use of Microsoft’s Entity Framework to provide a model for the jobs database so that users may filter and search for jobs. From a language perspective, the controllers and models are written in C# and the views are written in CSHTML using the Razor syntax. For the frontend user interface, the Bootstrap framework is used both for its professional and modern look and also for its mobile device compatibility. Bootstrap is accessed by using its Cascading Style Sheets (.css files) and JavaScript library, which allow for an easy mobile rendering when screen pixel widths of under 768px are detected. Below is a diagram showing an overview of the system software architecture.



## 3.3 Security Architecture

The security architecture of the EJCA is composed of four components: the admin mode, user password encryption, login validation, and SSL encryption. The functionality of the admin mode gives the administrator the ability to view and deactivate user accounts, ensuring that users abusing the application will be removed. Also, all user passwords will be encrypted using the SHA1 encryption algorithm before being stored in the database. SHA1 was developed by the NSA and is an example of a cryptographic hash function, considered nearly impossible to decode, thus ensuring the security of user password information from the system administrator. Users will receive an account validation email upon registering an account and will likewise have the ability to reset passwords with email validation, as system administrators will also not have access to passwords. Additionally, user logins are validated so that incorrect user credentials will not result in access to the application, thus maintaining a secure login portal for all users. Also, the Computech webserver, IIS 7.5, uses an SSL certificate to authenticate and encrypt data transmitted between users and the site using HTTPS, so that user data flows securely between the client and server. This is activated for each application use, as the EJCA is coded to request a secure connection channel from the server during the initial handshake. Below is a diagram showing an overview of the system security architecture.



## 3.4 Communication Architecture

The web application communication architecture consists of the communication between the different components of the software system and the communication between the running application on the user’s device and the Computech server. The software system utilizes Microsoft’s ASP.NET MVC architecture, so communication between the model, view, and controller components of the system is dictated by the workings of the framework. Generally, the C# controller contains a C# model object which it queries for data. This data is then passed to the .CSHTML view, where it is rendered for the user to view. Though communication between each of these components is handled via the .NET framework, communication between the application itself and users is handled by the HTTP protocol. The messages passed will be GET (in the case of requesting a web page) and POST (in the case of submitting a form) requests from the user or admin, which will trigger application calls to insert, delete, or select data from the database.

## 3.5 Performance

This performance of the system architecture provides metrics of how well it operates during user interaction with the software. This includes response time to user logins, registrations, job searches, and admin logins and activities, as well as web site availability. Included in this metric are page transition times, as for example from the home screen to the job search screen. Since database queries and requests in the application deal with either basic user information or job listings being retrieved from or entered into the database, load times will be a factor of how much RAM is available on the local machine and the speed of the server hosting the site. The webserver hosting EJCA has 16 GB of RAM, with a target speed, based on this configuration, of 100 KB per second. Additionally, given that the user machine has 500 MB of available RAM, and internet download speeds of at least 2 MBPS, overall loading quickness in the completed application will be at a maximum of 3 seconds. The system will also be available for use whenever it is run, barring any user internet connectivity problems. This means that Computech’s webserver will function properly and that the application will not require periods of unavailability due to maintenance needs.

# 4. System Design

## 4.1 Use-cases

### 4.1.1 User class – Job seeker

#### 4.1.1.1 Search

**Actors:** Job seeker

**Description:** User can search for various jobs available. When the user opens the website, the home page will show featured jobs with an option to conduct a job search. Users can also conduct a search for the same jobs once logged into their profiles.

**Trigger:** User clicks on **Search Jobs** button.

**Pre-conditions:** None

**Post-conditions:** System displays all the available jobs to the user.

**Normal Flow:**



**Alternative flow:** None

**Exception:** None

**Assumptions:** None

#### 4.1.1.2 Register

**Actors:** Job seeker

**Description:** Users can create a new account by providing personal information such as email id, password and first and last name. This information will be stored in database and email id and password will be used to log in user.

**Trigger:** User clicks on **Register** button under Users dropdown menu on the header.

**Pre-conditions:** None

**Post-conditions:** User can log in to the system next time and apply for job.

**Normal Flow:**



**Alternative flow:** N/A

**Exception:**



**Assumptions:** User has not previously registered with the same email address.

#### 4.1.1.3 Login

**Actors:** Job seeker

**Description:** User can login by entering email id and password entered during registration process. **Trigger:** User clicks on **Login** button under Users dropdown menu on the header.

**Pre-conditions:** User has registered with the same email address.

**Post-conditions:** User can view and update their profile as well as apply for jobs.

**Normal Flow:**



**Alternative flow:** N/A

**Exception:**



**Assumptions:** User email address and password exists in the database.

#### 4.1.1.4 Retrieve Password

**Actors:** Job seeker

**Description:** User should then be able to retrieve password through email. User can request to change password which will send further instruction to his or her email

**Trigger:** User clicks on **Forgot Password** link under Users dropdown menu on the header.

**Pre-conditions:** User has registered with the same email address.

**Post-conditions:** User can reset the password and will be able to login in to their account.

**Normal Flow:**



**Alternative flow:** N/A

**Exception:** For security purpose, there will be no exception in password retrieval controller. Even if an email id doesn’t exist in the database, user will redirected to Success page but email will not be sent.

**Assumptions:** User email id exists in the database.

#### 4.1.1.5 Apply

**Actors:** Job seeker

**Description:** User can apply to any jobs which they are interested in.

**Trigger:** User clicks on **Apply** button on **Details** page.

**Pre-conditions:** User has logged in.

**Post-conditions:** User is taken to page displaying ‘Application submitted successfully’ message.

**Normal Flow:**



**Alternative flow:**



**Exception:** User cannot apply if user is not logged in. As shown in alternate path above, if such a user clicks on Apply, user will be redirected to Login page and after validating credentials, user will be redirected back to Job Details page and then it follows the same process.

**Assumptions:** User has logged in.

#### 4.1.1.6 Filter by skills, location, pay rate, and experience:

**Actors:** Job seeker

**Description:** User can filter jobs based on skills, location, pay rate, and experience required for the job.

**Trigger:** User clicks on **Filter Jobs** button on Home page or Job Search page.

**Pre-conditions:** User selects at-least one criteria.

**Post-conditions:** System will display jobs based on selected criteria.

**Normal Flow:**



**Alternative flow:** None

**Exception:** None

**Assumptions:** User selects at-least one criteria.

#### 4.1.1.7 Profile Information

**Actors:** Job seeker

**Description:** User can enter his or her information such as address, phone number, skills and experience in his or her profile page.

**Trigger:** User clicks on **Update Profile** button on under the menu button on the header.

**Pre-conditions:** User has logged in or user has registered.

**Post-conditions:** User information will be saved in the database.

**Normal Flow:**



**Alternative flow:** None

**Exception:**



**Assumptions:** User has registered with the same email address.

#### 4.1.1.8 Change Information

**Actors:** Job seeker

**Description:** User can change information stored in his or her profile page. User should be able to change any fields stored in his or her profile page except the email ID.

**Trigger:** User clicks on **Update Profile** button on under the menu button on the header.

**Pre-conditions:** User has logged in.

**Post-conditions:** User information will be updated in the database.

**Normal Flow:**



**Alternative flow:** None

**Exception:** 

**Assumptions:** User has registered their account and saved information in their profile page.

#### 4.1.1.9 Upload Resume

**Actors:** Job seeker

**Description:** User can upload one resume and attach them to his or her profile.

**Trigger:** User clicks on **Upload Resume** button on **Profile** page.

**Pre-conditions:** User has logged in.

**Post-conditions:** None

**Normal Flow:**



**Alternative flow:** None

**Exception:** 

**Assumptions:** User has registered with the same email address.

### 4.1.2 User class – Admin

#### 4.1.2.1 Login

**Actors:** Admin

**Description:** Admin can login by entering email id and password stored in database.

**Trigger:** User clicks on **Admin** button on under the menu button on the header.

**Pre-conditions:** Admin email id is stored in the Tbl\_Admin table in database.

**Post-conditions:** Admin will be directed to Admin Dashboard.

**Normal Flow:**



**Alternative flow:** None

**Exception:**



**Assumptions:** Admin email address exists in the database.

#### 4.1.2.2 Search

**Actors:** Admin

**Description:** Admin can search for all registered users by their first and/or last names.

**Trigger:** User enters either first name or last name or both and click on **Search User** button.

**Pre-conditions:** Admin has logged in.

**Post-conditions:** Users matching the criteria will be displayed.

**Normal Flow:**



**Alternative flow:** None

**Exception:**



**Assumptions:** Admin’s email address and password exists in Tbl\_Admin.

#### 4.1.2.3 Deactivate Users

**Actors:** Admin

**Description:** Admin can deactivate users which will disable them to login into their account..

**Trigger:** Admin clicks on **Deactivate User** button next to user’s name.

**Pre-conditions:** Admin has logged in.

**Post-conditions:** User will be assigned 0 in **IsActive** entity in **Tbl\_User** which will lock the user and disable them to login again with the same email address.

**Normal Flow:**



**Alternative flow:** None

**Exception:** None

**Assumptions:** Admin’s email address and password exists in Tbl\_Admin.

#### 4.1.2.4 Delete Users

**Actors:** Job seeker

**Description:** Admin can delete users which will delete all of their information from the database.

**Trigger:** User clicks on **Delete** button next to the user’s name.

**Pre-conditions:** Admin has logged in.

**Post-conditions:** None

**Normal Flow:**

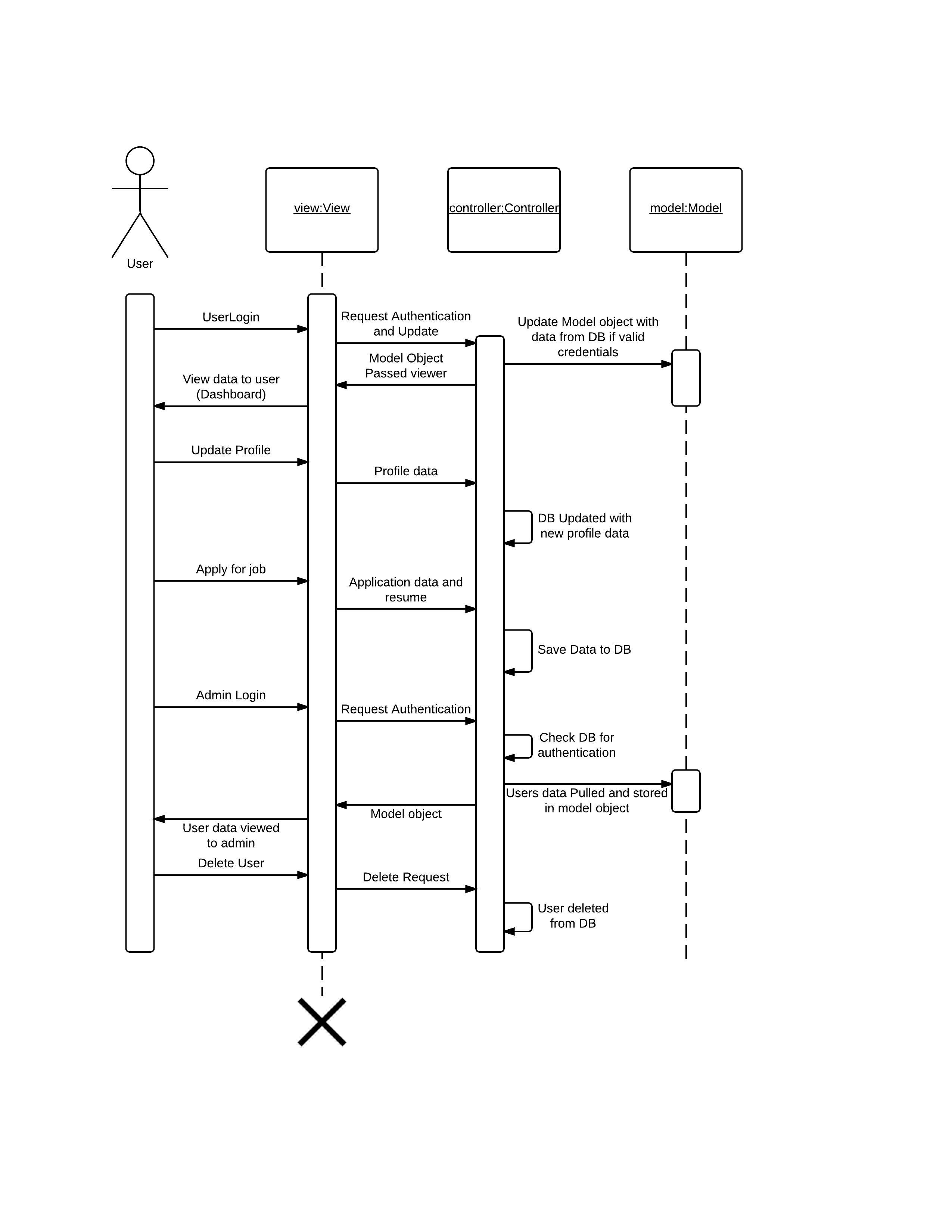


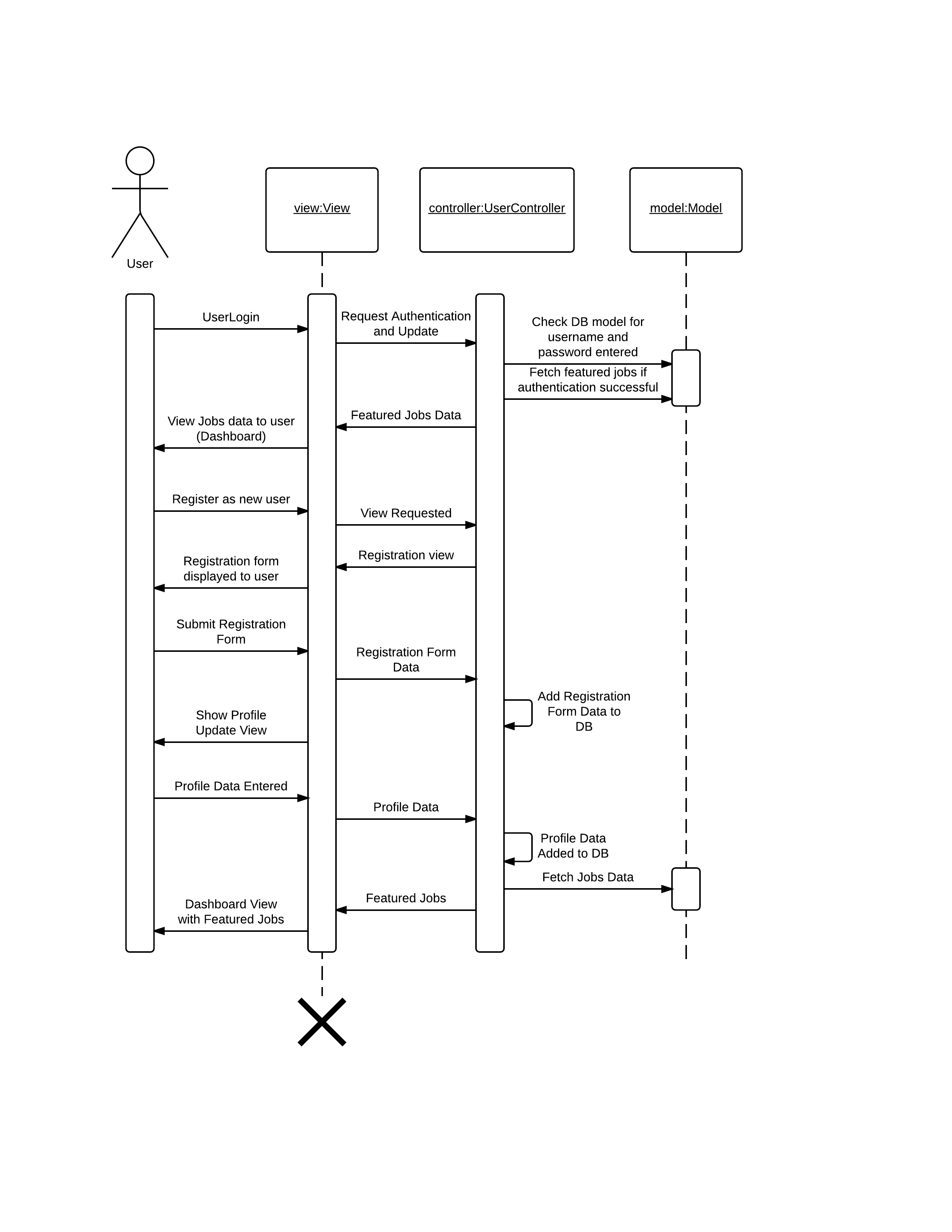
**Alternative flow:** None

**Exception:** None

**Assumptions:** Admin’s email address exists in the database.

## 4.2 Sequence Diagram



The following System Sequence Diagram covers the user functionality such as login and registration. 

## 4.3 Data Flow Diagrams

#### Level 1

Shows the flow of data among the user, web application, and the SQL database.



#### Level 2

Provides in-depth view of the data flow among the various components of the web application such as the UI, Application Core i.e. Model and Controller components, Viewer, and the SQL database.



#### Level 3-1

##### Shows the flow of user data in EJCA



#### Level 3-2

##### Shows the flow of jobs data in EJCA



## 4.4 Database Design



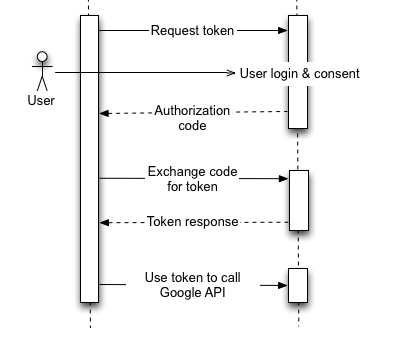
## 4.5 Application Program Interface

## 4.5.1 Google Authentication

The user has the ability to log into the EJCA with his/her Google account. This is achieved using the Google API (Microsoft.Owin.Security.Google). The Google API uses the OAuth 2.0 protocol for authentication and authorization. EJCA Web App is registered with Google and OAuth 2.0 credentials were obtained from the Google Developers Console.  
  
When a user selects the option to log in via Google, the user is prompted to log in using Google credentials. After logging in, the user is asked whether he or she is willing to grant the permissions that the application is requesting. This process is called *user consent*.

If the user grants the permission, the Google Authorization Server sends EJCA an access token. If the user does not grant the permission, the server returns an error and the Application goes back to the login page. Once an access token is received, it is passed to the Google API to complete the authentication.

EJCA Google Servers



(For image citation, see section 1.4 “References”)

### 4.5.2 Software Class Diagrams

Below is a partial UML class diagram illustrating the design of the controllers in the software system. Note that the lack of relational edged between each class is due to the ASP.NET framework’s low coupling of unrelated software components. Also note that each method is public, and most return an ActionResult associated with a .cshtml web page, allowing the user to access the content of the application. The model classes in the application closely follow the field layout of the database diagrammed above.

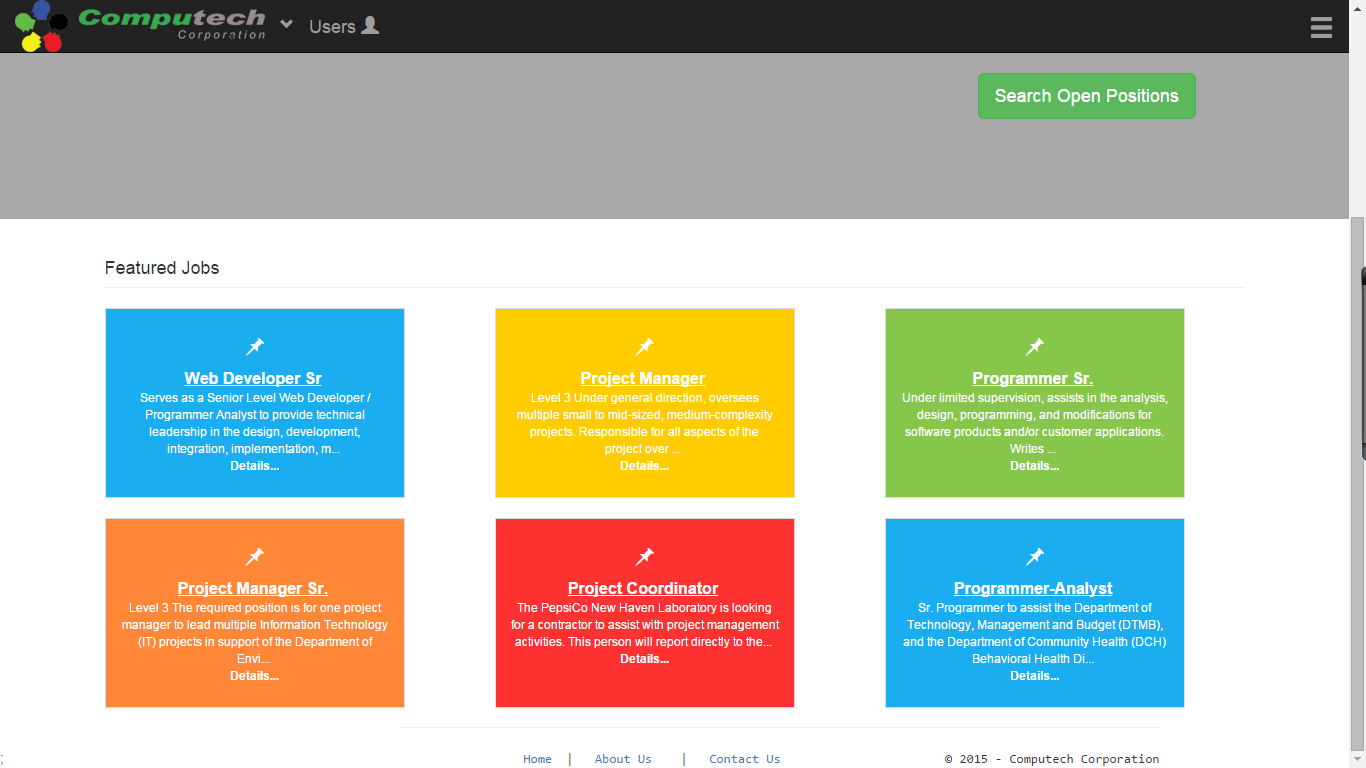
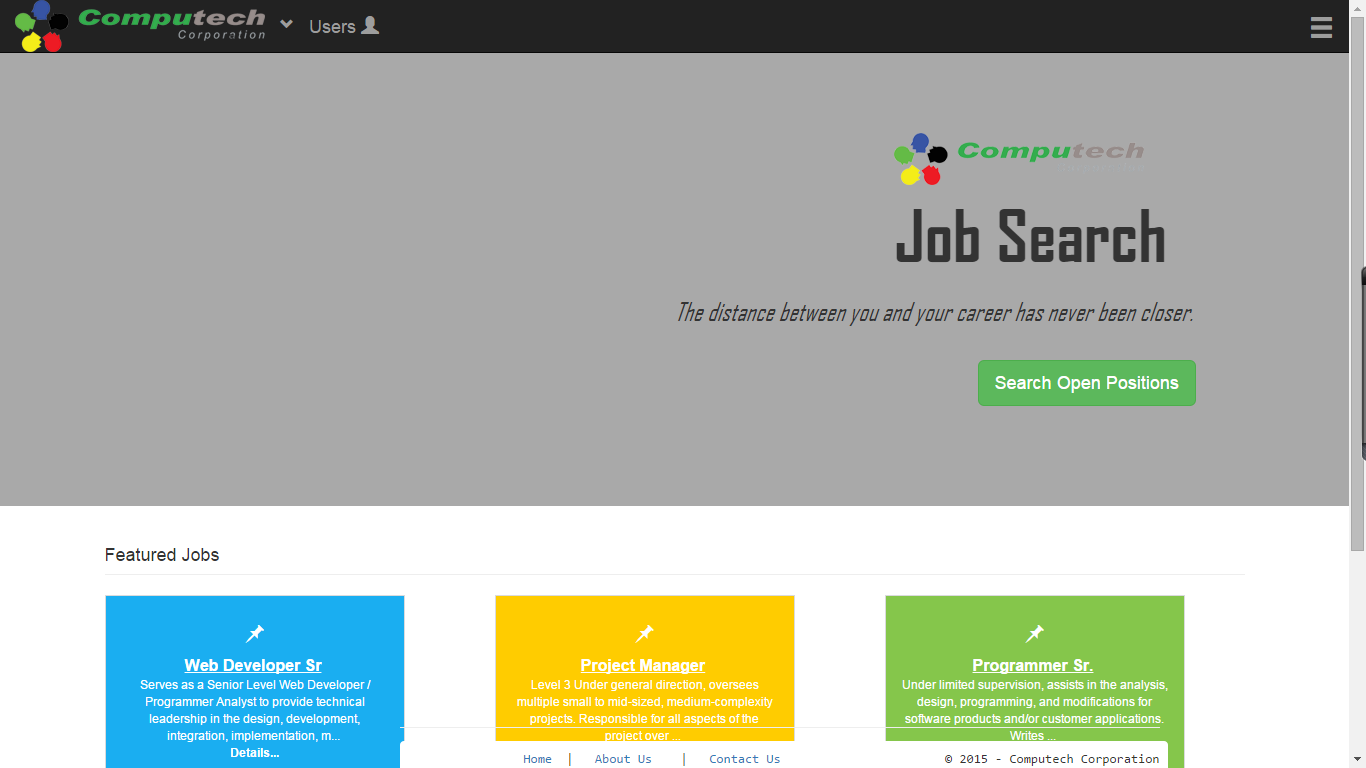






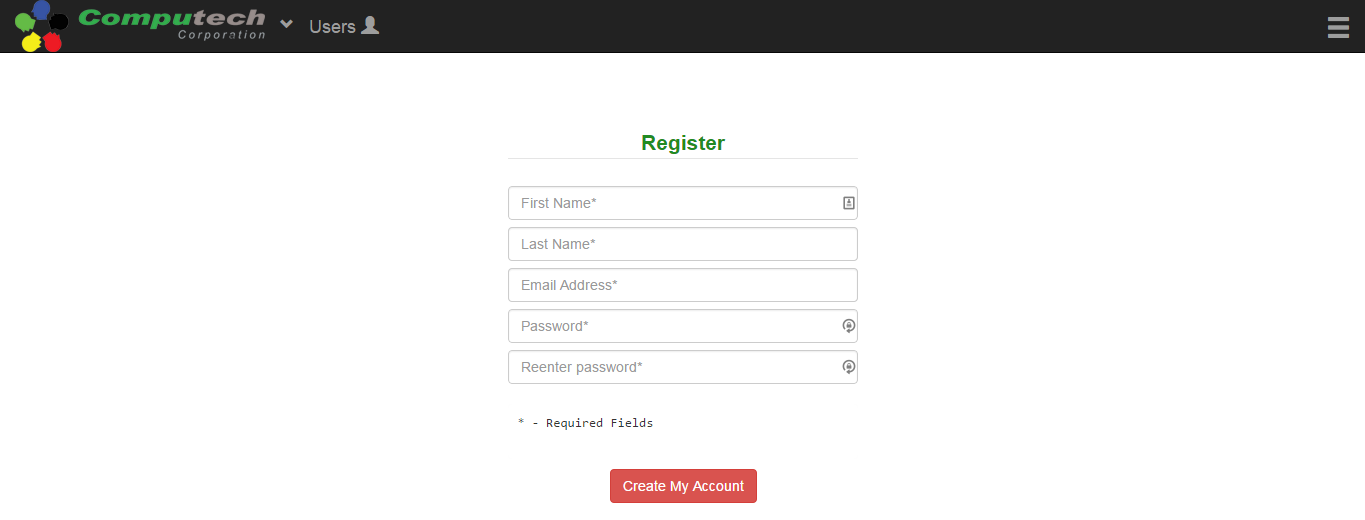
## 4.6 User Interface Design

### 4.6.1 Home Page

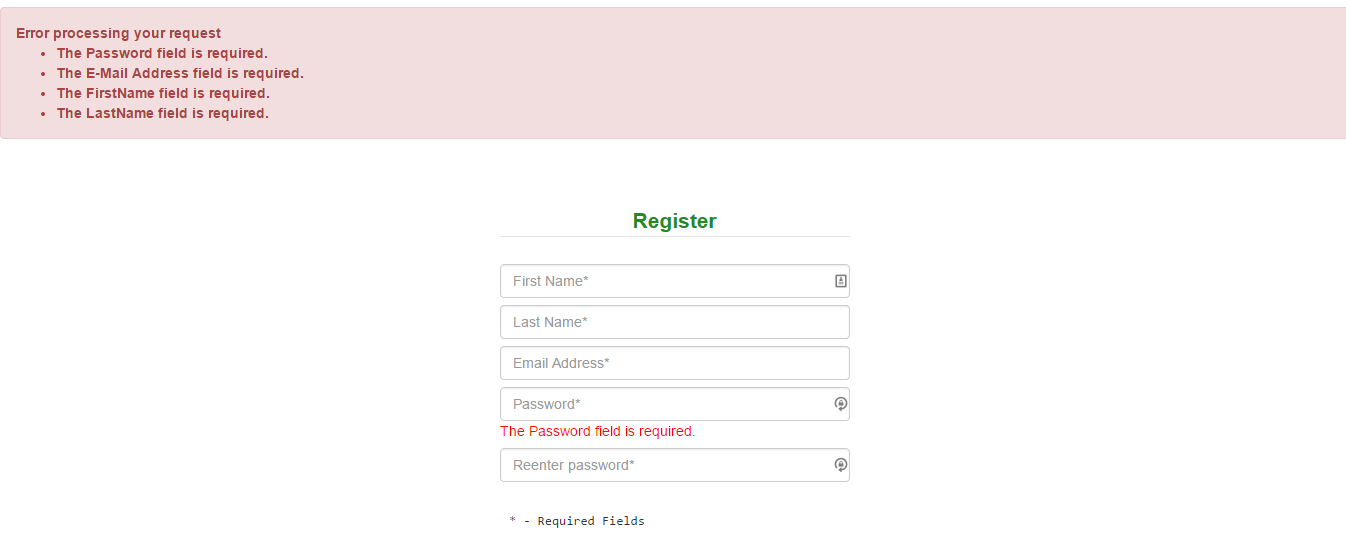


### 4.6.2 Register

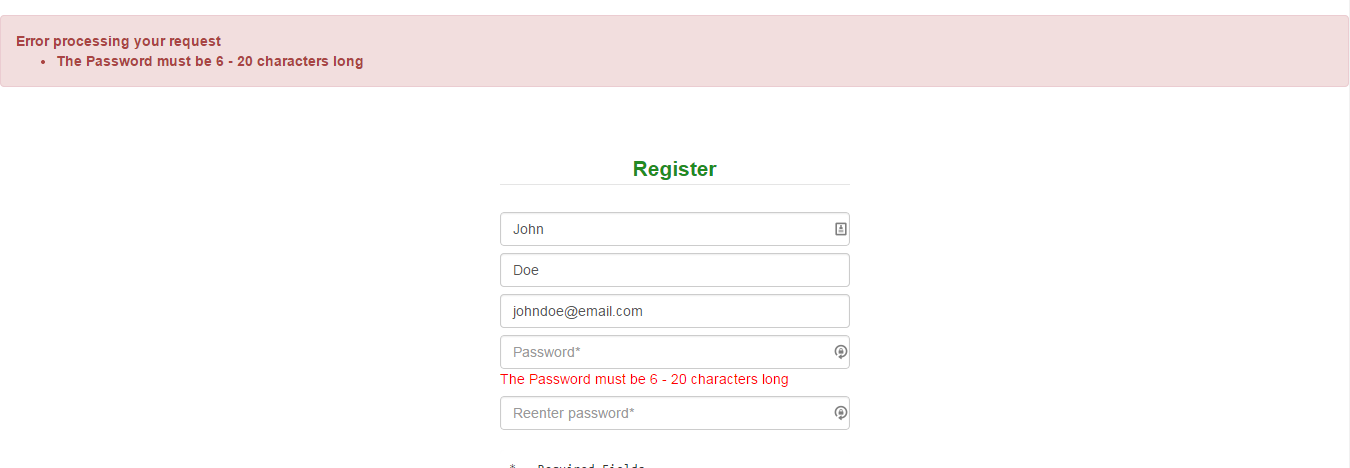
#### 4.6.2.1 Empty register form



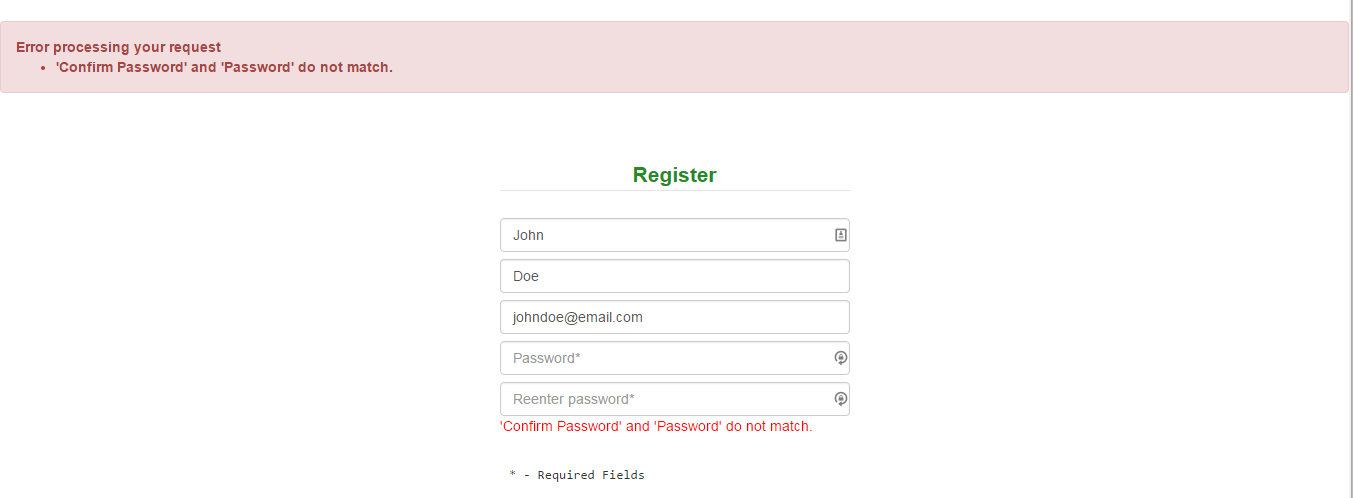
#### 4.6.2.2 Empty fields error



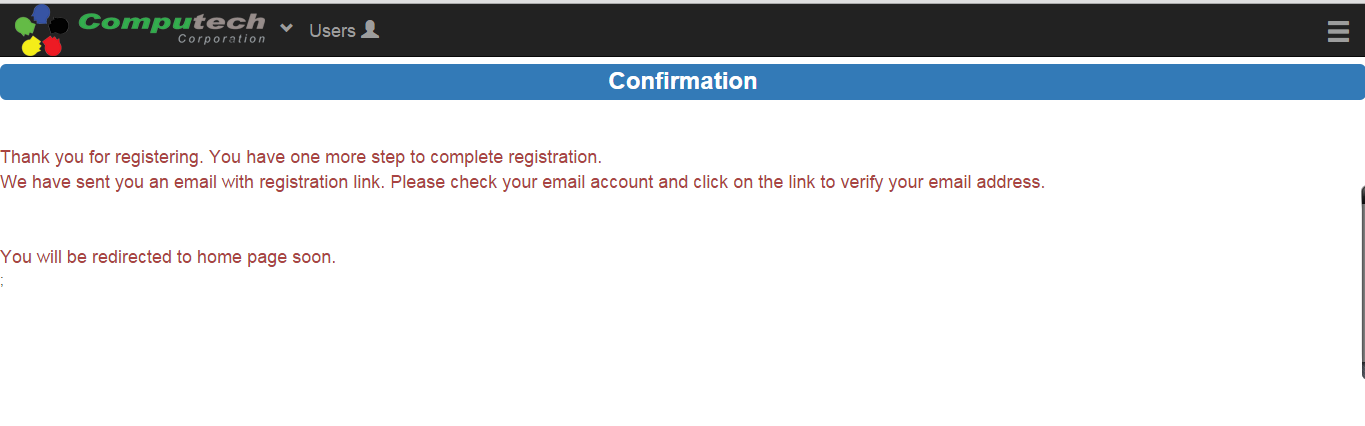
#### 4.6.2.3 Password length error



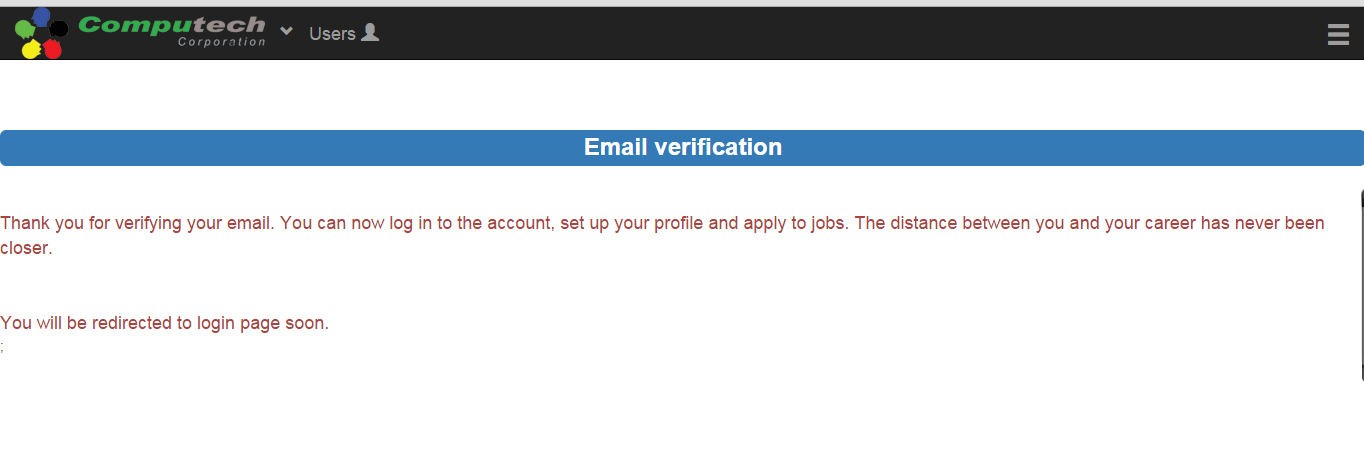
#### 4.6.2.4 Password and confirm password do not match error



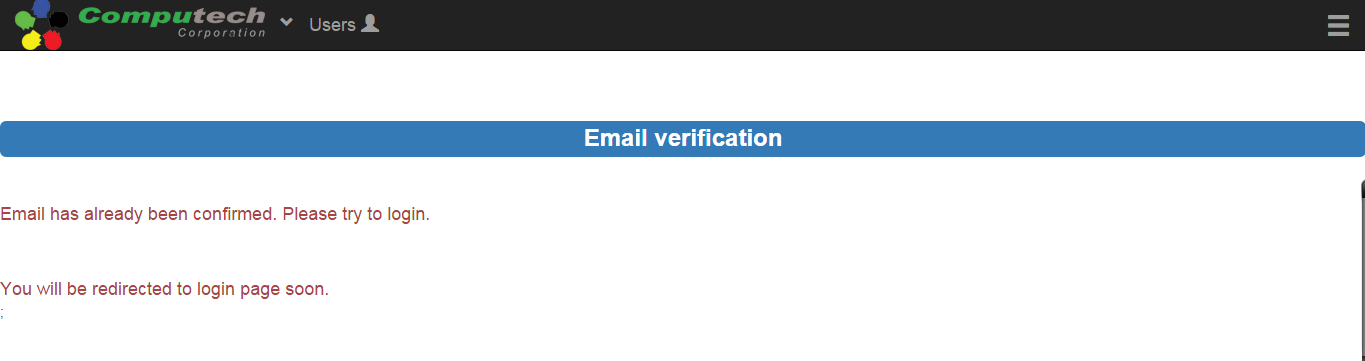
#### 4.6.2.5 Post registration email confirmation message



#### 4.6.2.6 Email verification

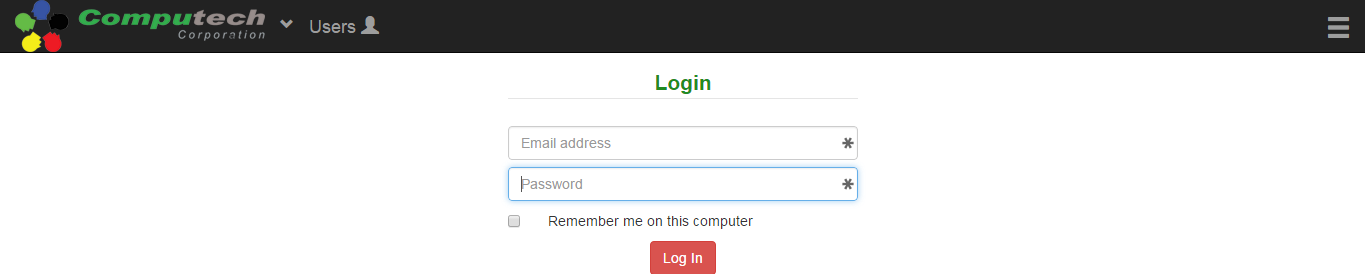


#### 4.6.2.7 Email reconfirmation error

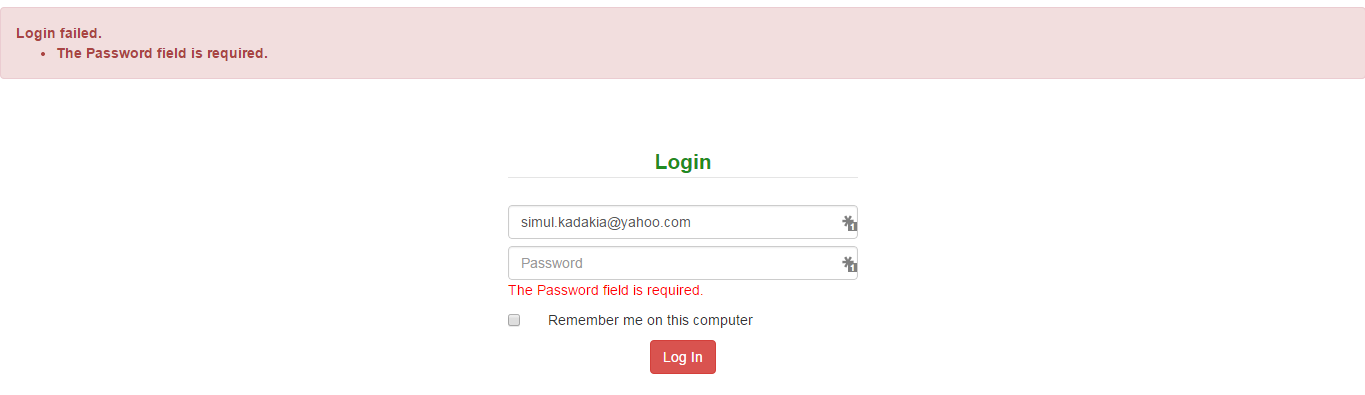


### 4.6.3 Login

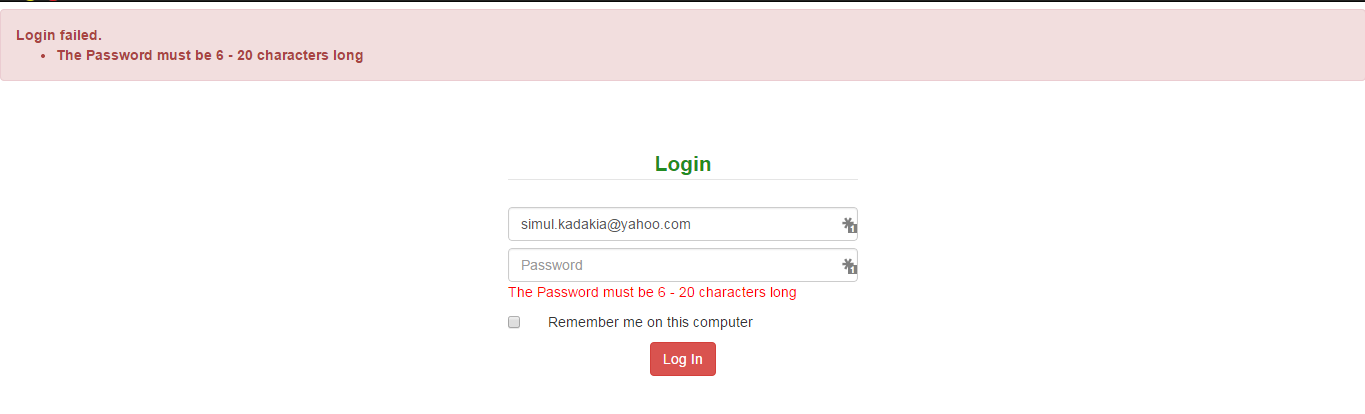
#### 4.6.3.1 Empty login form



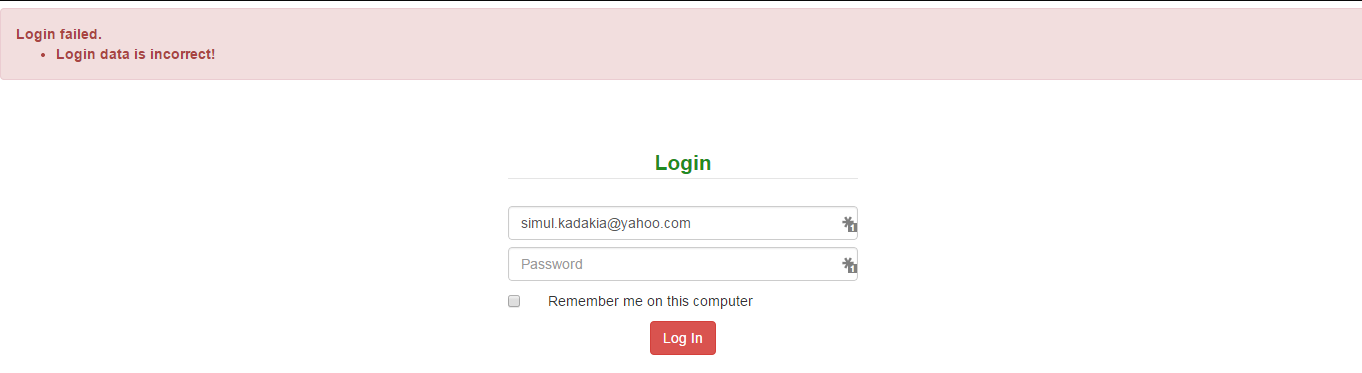
#### 4.6.3.2 Empty Password error



#### 4.6.3.3 Password length error

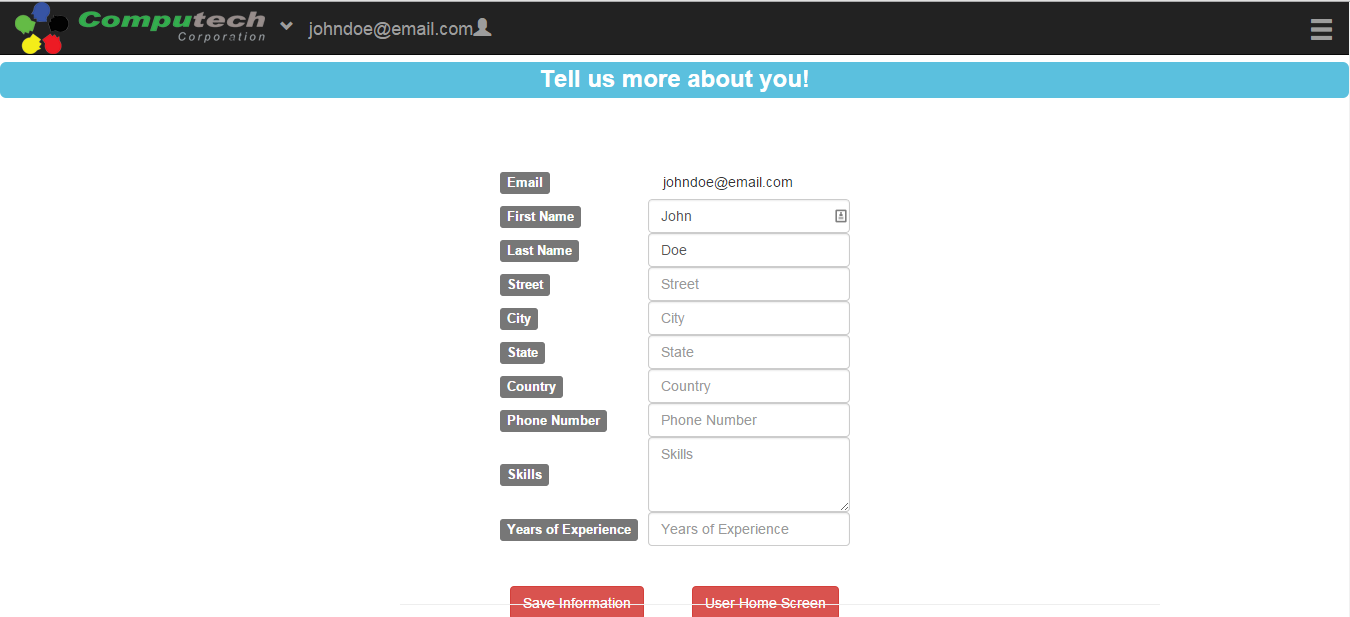


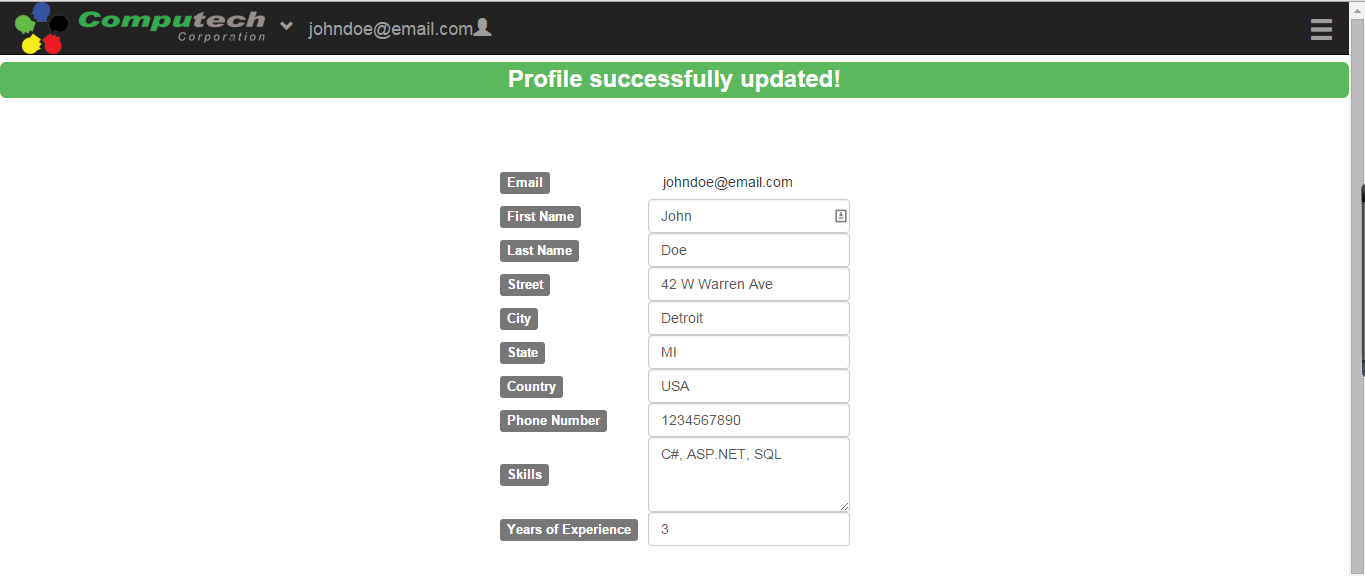
#### 4.6.3.4 Incorrect password/email/login without verifying



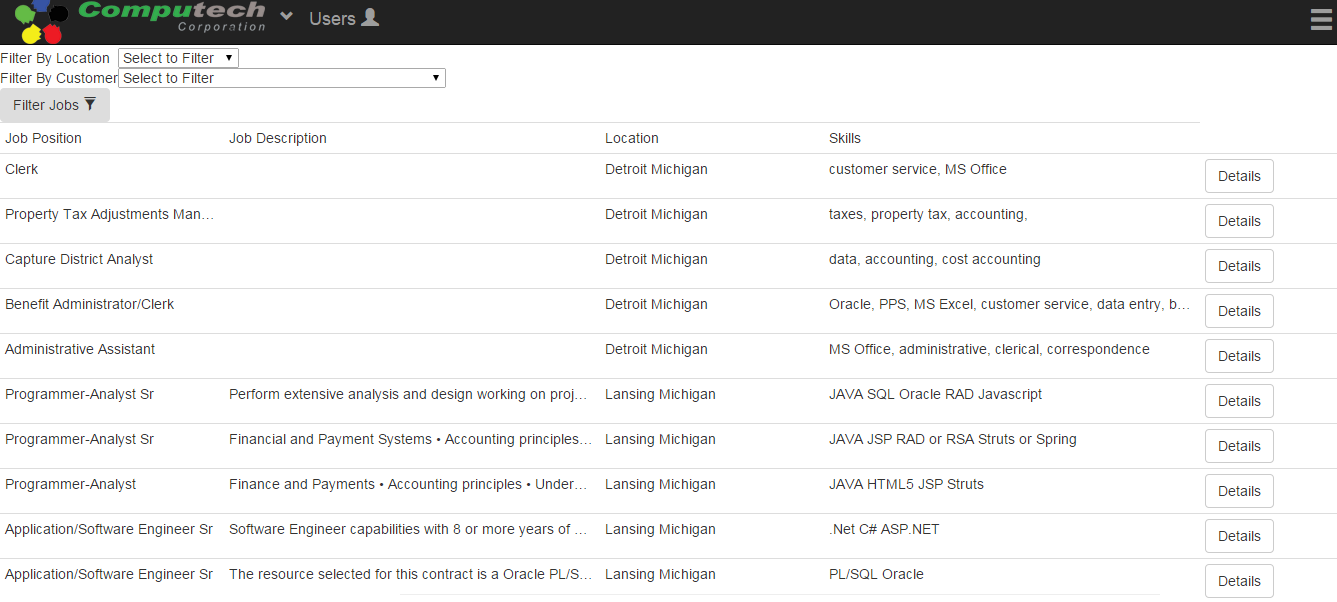
### 4.6.4 Profile Page

#### 4.6.4.1 Update Profile

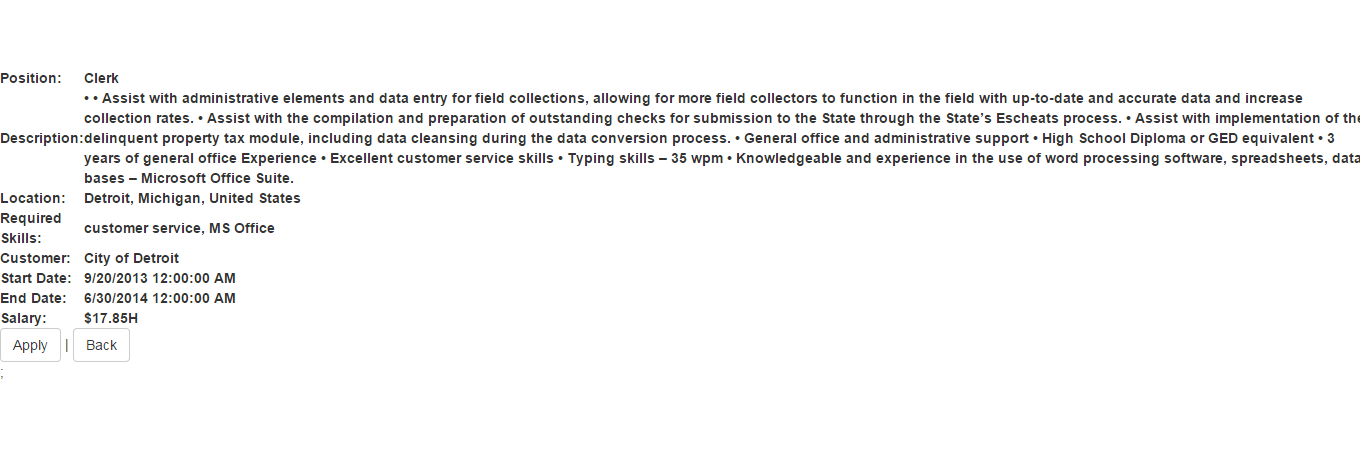




### 4.6.5 Job Search Page



### 4.6.6 Job Details Page



# 5. Product Design Specification Approval