VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Said Abdirahman Mohamed

# ONLINE RECRUITMENT APPLICATION

Technology and Communication

2019

## ACKNOWLEDGEMENTS

I would like to thank to VAMK for providing with me the opportunity to study at this prestigious University. I would also like to express my gratitude to my supervisor, Dr. Ghodrat Moghadampour, for his support, patience, and advice throughout the thesis work.

Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models. Most importantly, I wish to thank my loving and supportive wife, and my wonderful children, who provide unending inspiration.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

# ABSTRACT

| | |
|---|---|
| Author | Said Abdirahman Mohamed |
| Title | Online Recruitment Application |
| Year | 2019 |
| Language | English |
| Pages | 47 |
| Name of Supervisor | Ghodrat Mohgadampour |

The purpose of this thesis was to develop an online recruitment application through which two ends can meet. The ends are freshly-graduated jobseekers and employers. This project introduces the best possible ways possible how a jobseeker can apply for a job and view his/her applied jobs through the system. At the same time the system provides the employers with a channel to post their vacancies in their efforts of hunting new employees and checking if there is any application to their posted jobs. apply for jobs.

The following four major parts were designed and implemented. Firstly, the analysis phase the overall direction of the project was identified. Secondly, the design of the graphical user interface was implemented. Thirdly, a MySQL database that connects and communicates with spring boot was built to store all required data. Finally, Thymeleaf which is a modern server-side java templete engine was written to display the data in a human-friendly way to the users of this application.

The core features of this application for employers are posting jobs, editing jobs, deleting jobs and viewing applications. Other core features for jobseekers are viewing listed jobs in the system and applying for available jobs. In this application almost any employer can post a job online, the process is clear, easy to understand, less timewasting and user-friendly. Conversely, the process is very simple for jobseekers too, it makes quick and painless to apply for jobs.

Keywords Spring boot, MySQL,Bootstrap,Thymeleaf, Online Recruitment Application

# CONTENTS

**LIST OF ABBREVIATIONS**

| | |
|---|---|
| **HTML** | Hyper Text Markup Language |
| **CSS** | Cascading Style Sheet |
| **JAVA EE** | Java Platform Enterprise Edition |
| **SQL** | Structured Query Language |
| **IDE** | Integrated Development Environment |
| **API** | Application Programming Interface |
| **SDK** | Software Development Kit |
| **GUI** | Graphical User Interface |
| **ER** | Entity Relationship |
| **VAMK** | Vaasan Ammattikorkeakoulu |
| **UI** | User Interface |
| **URL** | Uniform Resource Locator |
| **HTTP** | Hypertext Transfer Protocol |
| **JDBC** | Java Database Connectivity |
| **UML** | Unified Modeling Language |

## LIST OF FIGURES AND CODE SNIPPETS

# 1 INTRODUCTION

Somalia is an underdeveloped country in East Africa. The country suffered from civil wars for a long time and the recruitment process is very slow. With that being said, the country's telecommunication infrastructure is strong when compared with other countries in the region, such as Ethiopia and Djibouti. Somalia is one of the first East African countries that used Electronic Virtual Cash (EVC). /1/

The internet is a trend worldwide. In Somalia, more and more users are gaining internet access every day. On a visit to the country, I discovered that an online recruitment application would help Somali employers to gain a huge potential pool of applicants. I then decided to develop an application for Somali Organization in order to help them in the recruitment process. Employers and Jobseekers can use Online Recruitment Application with its full functionality.

Recruitment is the process of finding and recruiting the best qualified applicant from within or outside a company for job opening, in a timely and cost-effective manner. The recruitment process includes determining the requirements of a job, attracting candidates to that position, evaluating and selecting applicants, hiring and integrating the new employee into the organisation./2/

The first objective of this project is to build a common meeting ground for freshly-graduated job seekers and the employers where candidates find their dream jobs and recruiters hunt the right candidates. The second objective is that this recruitment application must be easy to use and secure at the same time.

This study contains five chapters. Chapter 1 introduces the study by outlining the background of the thesis. It states the objective that the research is set to achieve and explains the research problem and research questions. Chapter 2 reviews relevant literature for the research which includes: the technologies and tools used to build this application. Chapter 3 outlines the research methodology. In Chapter 4, the researcher designs the application’s GUI and database. Chapter 5 explains some of the implementation part of the application. Chapter 6 describes the test cases of all features of this application. In Chapter 7, conclusions are given.

# 2 REVELANT TECHNOLOGIES

This chapter explains tools and technlogies used to achieve the aim of the project, some of which are Java Programming Language Spring Boot, Bootstrap, Thymeleaf, MySQL and Eclipse IDE**.**

## 2.1 Java Programming Language

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. Java code can run on any Java supporting platform without the need for recompilation. Java applications are usually compiled to bytecodes that can run on any Java virtual machine (JVM) regardless of the computer's underlying architecture. As of 2019, Java has continued to grow and with an estimated 9 million developers it was one of the most common programming languages in use according to GitHub, particularly for client-server web applications. The Java language was created with five primary goals:

- It has to be simple, focused on the object, and familiar.
- It's must be robust and safe.
- It must be portable and architecture-neutral.
- It has to perform with high performance.
- It has to be interpreted, dynamic, and threaded.

One of the design goals of Java is portability, meaning that programs written for the Java platform run on any combination of hardware and operating system with adequate run time support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode instead of directly to the architecture-specific machine code./3/

### 2.1.1 Object-Oriented Programming

Java is an Object-oriented programming language (OOP). OOP is a methodology or paradigm to design using classes and objects. An object is an entity in the real world that has its own properties and behaviors, whereas classes are blueprints that define an object.

## 2.3 Spring Boot

Spring framework has been around for over a decade and a half, the first edition was written by Rod Johnson, who published the framework with his book Expert One-on-One Design and Development in October 2002. The Spring Framework is an application framework and inversion of control container for the Java platform. Spring Framework provides structure and common patterns to make the process of building applications easier.

The framework's features can be used by any Java application, but there are extensions for building web applications on top of the Java Enterprise Edition (EE) platform. The framework does not impose any specific programming model but, it gives comprehensive infrastructure support for developing Java applications. Spring Framework includes some good features and modules, such as Dependency Injection and they provide a range of services. Some of these models are Spring JDBC, Spring MVC, Spring Security, Spring AOP, Spring ORM, Spring Test,These modules can reduce the development time of an application significantly. In the early days of Java web development before Spring Framework was released, Java developers needed to write a lot of boilerplate code to insert a record into a data source. But by using the JDBCTemplate of the Spring JDBC module the process has been reduced to a few lines of code with only some configurations./4/

Spring Boot is a project built at the top of the Spring Framework. It provides a simpler and quicker way to setup, configure, and run applications that are both simple and web based. It smartly chooses your dependencies, auto-configures all the features required, and the application can be started with one click. Furthermore, it also simplifies the deployment process of the application. Using Spring Boot the

process of writing a Java-Based web application becomes easy, first the application needs to be packaged then run with a simple command, such as java -jar my-application.jar. Spring Boot provides plugins to work with embedded and in-memory databases very easily.

Spring Boot has the following core features that make it unique and easy to use:

1. Auto-configuration: Spring Boot can automatically provide configurations based on the application being built.
2. Standalone: Spring-boot-starter provides all the required dependencies needed and auto configurations to build standalone applications. The application simply needs to be started by clicking a button or giving a run command.
3. Opinionated: The framework helps developers to setup a working application quickly by giving default configurations that are most likely to satisfy developers./5/

### 2.3.1 Spring Initializr

Spring Initializr is a web-based tool to bootstrap a Spring Boot application. With the assistance of Spring Initializr we can easily generate the structure of the Spring Boot project. It provides extensible API to build projects based on JVM. It provides the project with various options that are presented in a model of metadata. The metadata model enables us to configure the list of JVM and platform for example, version-supported dependencies, It represents the metadata in a well-known allowing third-party clients to obtain the requisite assistance. It supports IDE STS, IntelliJ IDEA Ultimate, NetBeans and Eclipse. The Spring Team has provided these three approaches to create Spring Boot Application, using Spring Boot CLI Tool, using Spring STS IDE and using Spring Initializr Website. The last approach Spring Initializr UI has been used to create this Online Recruitment Application./6/ The screenshot below displays the Spring Initializr UI.

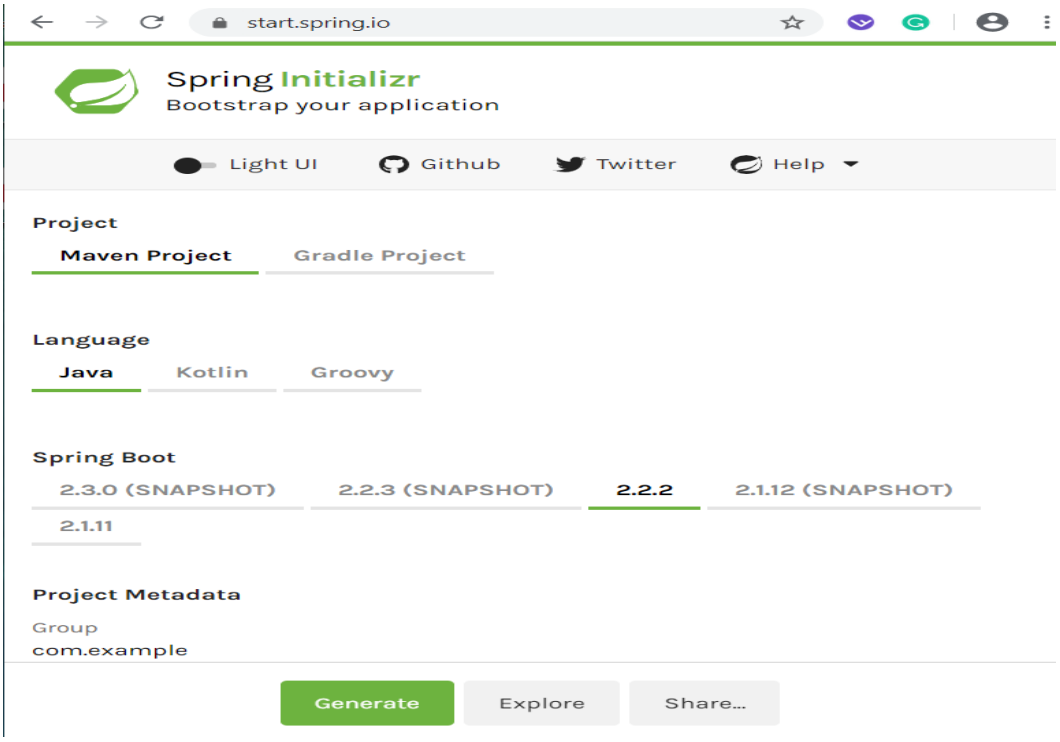


**Figure 1.** Spring Initializr UI

A spring boot application can easily be bootstrapped by entering the kind of project, language, the Spring boot version and project metadata in the Spring Initializr UI fields and then clicking the Generate button. This will start packing the project and download the selected Jar or War file.

## 2.4 Thymeleaf

Thymeleaf is a modern server-side Java template engine for rendering pages (HTML5, XML, XHTML) for both web and standalone environments. It is open source, free, and downloadable without difficulties. Thymeleaf is so simple to learn and its syntax is easy to understand. Thymeleaf's main goal is to bring the development workflow with beautiful natural templates – HTML that can be viewed correctly in browsers and also act as static prototypes, allowing for better collaboration in development teams. HTML templates written in Thymeleaf look and work like HTML. Big companies all over the world are using Thymleaf such as Across framework, Auchan Retail France, Sahibinden and PPI AG./7/

### 2.4.1 Using Thymeleaf

Thymeleaf is a library for Java. It is a template engine and capable of applying a set of transformations to template files to display the data and/or text generated by the applications. It is best suited to serve XHTML / HTML5 in web applications, but it can process any XML file, whether in web applications or in standalone applications. Thymeleaf's main goal is to create models in an elegant and well-formed way. To achieve this, XML tags and attributes are used to define the execution of predefined logic on the DOM (Document Object Model) instead of writing that logic explicitly as code within the template.

Its architecture allows templates to be processed quickly, relying on smart caching of parsed files to use as few I / O operations as possible during execution. Finally, Thymeleaf has been designed with XML and Web standards in mind from the beginning, allowing to create templates that are fully validated if required./8/

### 2.4.2 Thymeleaf Standard Dialect

Thymeleaf is an incredibly extensible template engine (actually it has to be best referred to as a template engine framework) that enables the full description of the DOM nodes to be transformed into templates and, in addition, how they are to be processed.

An object that adds any common sense to a DOM node is known as a processor, and a hard and quick of those processors — plus a few larger artifacts — is called a dialect of which Thymeleaf's core library provides one out — the Standard Dialect, which must be sufficient for a large percentage of users ' desires.

The Thymeleaf Standard Dialect can perfom templates in any mode, however it is especially appropriate for web-oriented template modes (XHTML and HTML5 ones). Besides HTML5, it mainly supports and validates the following XHTML specifications: XHTML 1.0 Transitional, XHTML 1.0 Strict, XHTML 1.0 Frameset, and XHTML 1.1. Most of the processors of the Standard Dialect are characteristic processors. This lets in browsers to successfully display XHTML/HTML5 template documents even before being processed, because they will without a doubt ignore

the extra attributes. For example, while a JSP the usage of tag libraries could consist of a fraction of code not at once displayable through a browser.

Thymeleaf's center is a tool for storing DOMs. In particular, it uses its own high-performance DOM implementation— not the standard DOM API — to build in-memory tree representations of templates on which it later operates through its nodes and executes processors on them which modify the DOM according to the current configuration and the collection of statistics passed to the temp.

The use of a DOM template representation makes it thoroughly suitable for net programs due to the fact net files are very frequently represented as item timber; in fact DOM trees are the manner browsers represent internet pages in reminiscence. Also, building on the idea that most web applications use only a few dozen templates, that these are no longer huge documents and they don not generally change whilst the application is running, Thymeleaf’s utilization of an in-reminiscence cache of parsed template DOM bushes permits it to be speedy in manufacturing environments, Because most template processing operations require very little I/O./8/

## 2.5 Bootstrap

Bootstrap is a free and open source front-end framework for designing websites and web applications. It contains HTML and CSS-based design templates typography, forms, buttons, navigation and other interface components. Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter. Bootstrap 4 is the newest version of Bootstrap with new components, faster stylesheet and more responsiveness. Bootstrap 4 supports the latest versions of all major browsers. The most common components of Bootstrap are its layout components, its basic layout component is called Container and every other element in the page is put in it. The Container has four predefined fixed width, depending on the size of the screen. Once a container is in place, other Bootstrap layout components enforce a CSS grid layout through specifying rows and columns. /9/

Bootstrap helps us to make modern, sleek, responsive and mobile first websites. It provides a solid basis for any website, regardless of the size of the project. It includes Reboot, which is based on Normalize.css and helps level out browser differences for different page elements. Here top four reasons that make Bootsrap one of the most popular front-end frameworks on the web are explained:

The powerful Grid System: Bootstrap comes with one of the best mobile-first, responsive grid systems. It's built with Flexbox, and its simple to use. While developers have a CSS Grid layout for building models, the Bootstrap Grid section can still be useful for fast prototyping.

Rapid Development: Bootstrap has many common CSS and JavaScript components that can help any website achieve the desired functionality.

Browser Compatibility: Bootstrap supports all major browsers and platforms, with the latest stable updates. By following Bootstrap's instructions carefully, a website design can be built that works fully in all those major browsers.

Open Source: Bootstrap is an open source project hosted on Github and published under the license of MIT. This is one of the major reasons Bootstrap became famous. As a developer there is no need deal with purchasing and licensing issues./10/

## 2.6 MySQL

MySQL is an open source relational database management system (RDBMS). MySQL was developed by a Swedish company, MYSQL AB, founded by David Axmark, Allan Larsson and Micheal Widenius. Original development of MySQL launched in 1994 by Widenius and Axmark. MySQL's first version came out on 23 May 1994. It is written in C and C++ and the last version of MySQL Server is 8.0 which was announced in April 2018. /11/

With over ten million installations. MySQL is the most popular database management system. One reason for MySQL's success must be the fact that it can be used with PHP and it is free. But it is also extremely powerful, fast and highly scalable, which means that it can grow with the application. There are three different

ways in which a user can interract with MySQL: using a command line, via a web interface such as phpMyAdmin, and through a programming language, such as Java.

The data in a MySQL database are stored in tables. A table is collection of related data, and it consists of columns and rows. Since the 1980s, RDBMSs have been a suitable choice for storing information, personnel data, and other applications. Relational databases have often substituted traditional hierarchical databases and network databases as they have been easier to implement and maintain.

The aim of using MySQL for this application is to make it possible to dynamically pull contents from the database to create web pages for regular browser viewing. So, at one end of the system we have a visitor to our application using a web browser to request a page. That browser expects to receive standard HTML document in return. One the other hand, we have the contents of our application, which is located in one or more tables in a MySQL database that only understands how to respond to SQL queries./12/

## 2.7 Eclipse IDE

Eclipse is an integrated framework (IDE) for the development of Java programming language applications and other programming languages such as C / C++, Python, and Ruby. The Eclipse architecture that provides the basis for the Eclipse IDE is composed of plug-ins and is built with additional plug-ins to be extensible. The software Eclipse can be used for the development of rich client applications, integrated development environments and other tools. The Java Development Tools (JDT) project offers a plug-in enabling for the use of Eclipse as a Java IDE. The Eclipse Public License(EPL) is the license under which Eclipse projects are released. EPL ensures that Eclipse is free to download and install, it also allows Eclipse to be distributed. Since 2006, the Foundation releases new versions, the newest version of Eclipse is 2019-12 which released 18 December 2019 /13/

Eclipse Foundation's mission is to enable development through the provision of infrastructure (version control systems, code review systems, server building, download sites, and a structured process. The Eclipse Foundation is not operating

on the basis of the Eclipse code, i.e. it has no employee developers working on Eclipse projects.

Eclipse's open source community nowadays consists of over 150 projects covering various aspects of software development. Eclipse projects, for example, host the Jakarta EE project (formerly known as Java EE), the JavaScript development framework and the Jetty webserver. As an Integrated Development Environment (IDE) for Java, most people know Eclipse. The Eclipse IDE is Java's leading development environment with a market share of approximately 40.5% in 2019. There are several modules in the Eclipse IDE. The website of Eclipse.org provides pre-packaged Eclipse distributions for typical use cases to provide downloads. The distribution of the Eclipse IDE for Java Developers is specifically designed for regular development of Java. This includes typical packages required, such as Maven and Gradle build system support and Git version control system support./14/

# 3. APPLICATION DESCRIPTIONS

This chapter presents the detailed description of the project and its requirements. This project has two main parts server-side, which is hidden from the users, and client-side functionalities, which can be seen and used by the clients. Following, the researcher covers on the client side in detail.

## 3.1 Quality Function Deployment

The following sections provide a detailed description of the application's key features and functionalities. The properties are classified into three major categories: Must-have, Should-have, and Nice-to-have. Must-have is any requirement that absolutely has to be delivered for the project to be considered successful whereas Nice-to-have are the complement of objectives or requirements that are considered desired or even important the overall deliverable, but can be considered as optional or nice-to-have in the overall completion of the project. These may alternatively be classified as optional, non-critical or auxiliary requirements.

### 3.1.1 Normal Requirements (Must Have)

A job seeker must be able to:

1. Register his/her personal information into the system
2. Login with his/her registered username and password
3. See available jobs
4. Apply available jobs

The employer must be able to:

1. Register his/her personal information into the system
2. Login with his/her registered username and password
3. Post available jobs
4. See applicants who applied his/her open jobs
5. See the jobs his/her has posted

### 3.1.2 Expected Requirements(Should Have)

The system should be secure, reliable and at the sametime user friendly. Job seekers and employers should be able to login with their existing username and password easily and then play with the rest features on the system.

### 3.1.3 Optional Requirements(Nice To Have)

Job seekers can add their CV's and resumes while applying for the job which can be reviewed by the employer any time needed to proceed the application process. Employers can store the information of the company like when it is founded, what kind of projects they are working on and the work culture in their workplace.

## 3.2 Use Case Diagram

A use case is a structure for documenting the functional requirements for a system, usually involving software. Each use case provides a set of scenarios that shows how the system should interact with a human user or another system. In this project there are two actors identified for this system - jobseeker and employer. They have their own pages and to login they should use their existing email and password for authentication which authorizes their existing data.

### 3.2.1 Job Seeker Use Case

The following figure shows the Jobseeker's use case diagram. The diagram describes how the Jobseeker interacts with the application and also shows all the actions that the Jobseeker can perform on this application.

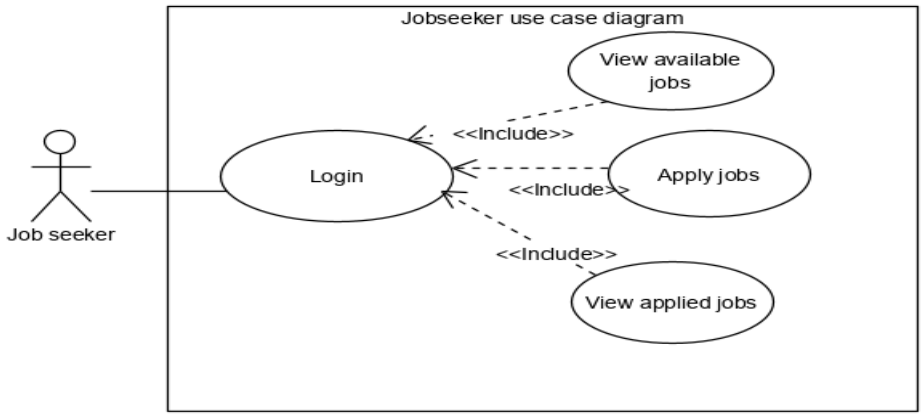


**Figure 2.** Job seeker use case diagram

The Jobseeker has these lists of use cases in details.

### 3.2.2 Job Seeker Login Review


**Figure 3.** Jobseeker Login Use Case

The Jobseeker's login Description:

1. The system prompts the job seeker to enter his/her email and password after clicking the job seeker's page.
2. The Job seeker clicks the sign-in button.
3. The system verifies if the provided email and password are correct.

### 3.2.3 Job Seeker View Jobs Review

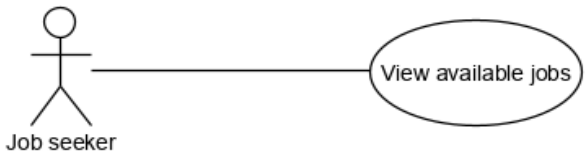


**Figure 4.** Jobseeker view jobs use case

1. The Jobseeker's view jobs Description:
2. The Jobseeker clicks view jobs page.
3. The system lists all available jobs in detail.

### 3.2.4 Job Seeker Apply to Job Review

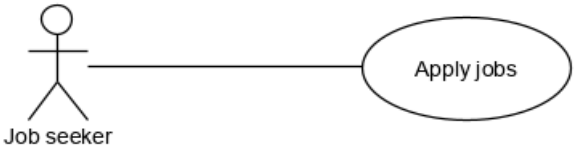


**Figure 5.** Jobseeker apply to job use case

The Jobseeker's apply jobs Description:

1. The Job seeker clicks apply button to apply after sees his/her suitable job.
2. The system verifies the information and saves to the database.
3. The system returns the job seeker to the view jobs page.

### 3.2.5 Job Seeker View Applied Jobs Review

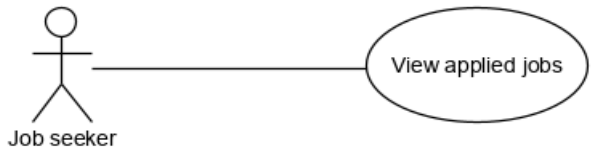


**Figure 6.** Jobseeker view applied jobs use case

The Job seeker's applied jobs Description:

1. The Jobseeker clicks applied jobs button to see his/her applied vacancies.
2. The system fetches these data from the database and displays them in a user friendly way.

### 3.2.6 Employer Use Case

The following figure shows the Employer use case diagram. The diagram presents how the Employer deals with the application and also shows all the actions that his/her can perform on this application. The Employer has two features more than the job seeker, which are edit and delete jobs.

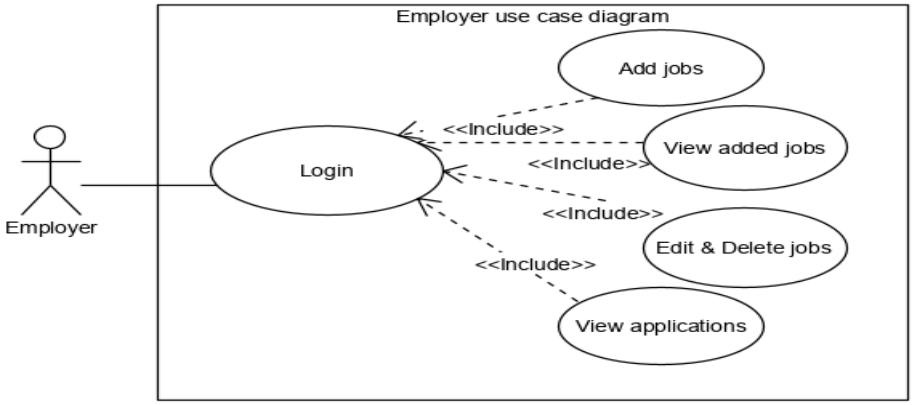


**Figure 7.** Employer use case diagram

The Employer has these lists of use cases in details.

### 3.2.7 Employer Login Review

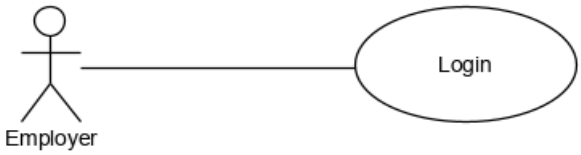


**Figure 8.** Employer login use case

The Employer's login Description:

1. The system prompts the employer to enter his/her email and password after clicking the employer's page.
2. The Employer clicks the sign-in button.
3. The system verifies if the provided email and password are correct.

### 3.2.8 Employer Add Job Review


**Figure 9.** Employer add job use case

The Employer's add job Description:

1. The Employer clicks the addjob page in order to add available jobs into the system.
2. The system prompts the employer to enter all necessary information about adding new jobs
3. The system verifies provided information and then saves into the database.

4. The system returns the employer to the same page.

### 3.2.9 Employer View Jobs Review

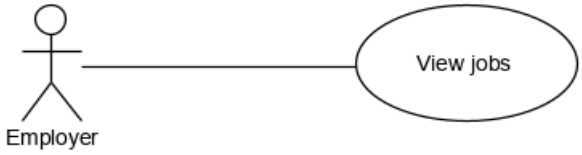


**Figure 10.** Employer view job use case

The Employer's see job Description:

1. The Employer clicks the  view jobs page.
2. The system lists all jobs that the employer has already added into the system.
3. The Employer can edit or delete all his/her jobs on the same page.

### 3.2.10 Employer Edit and Delete Jobs Review

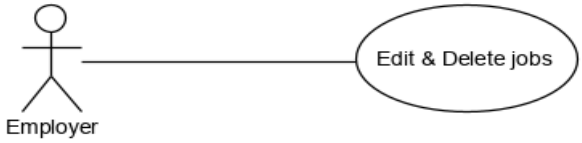


**Figure 11.** Employer edit and delete job use case

The Employer's edit and  delete job Description:

1. The Employer clicks view the jobs page and then clicks delete or edit icon.
2. The Employer clicks the delete icon, then the system deletes that particular job from the database.

3. The Employer clicks the edit icon, then the system prompts the employer to edit the information.
4. The system verifies the edited information and then saves into the database.

### 3.2.11 Employer View Applicants Review

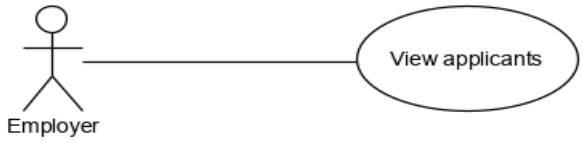


**Figure 12.** Employer view applicants use case

The Employer's view applicants Description:

1. The Employer clicks the view applicants page.
2. The system fetches all applicants from the database and displays them in a user friendly way.

## 3.3 Sequence Diagram

The sequence diagram is a type of UML diagram that shows how objects in a system or classes within a code interact with each other. Particularly these diagrams show interactions in the order they take place. Messages in the sequence diagram show the information being sent between the objects, the sequence diagrams show the order of interactions or sequences.

### 3.3.1 User Login Sequence Diagram

The following sequence diagram describes how the user interacts with the login instance on the application and how many steps must be taken before the user proceeds with other features of the application.

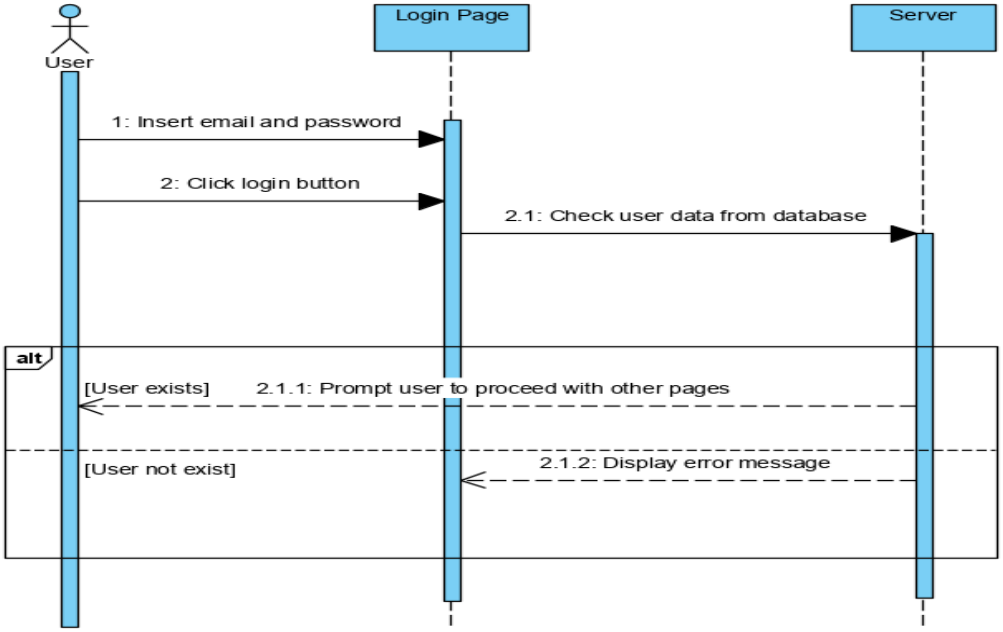


**Figure 13.** Employer login sequence diagram

### 3.3.2 Add Job Sequence Diagram

The following sequence diagram explains how the employer interacts with the add job feature in the application and how many steps must be taken in order to add data to the system. The Employer fills the form and then clicks the submit button, the controller will save the data into the database and then the server will show the saved data to the employer.

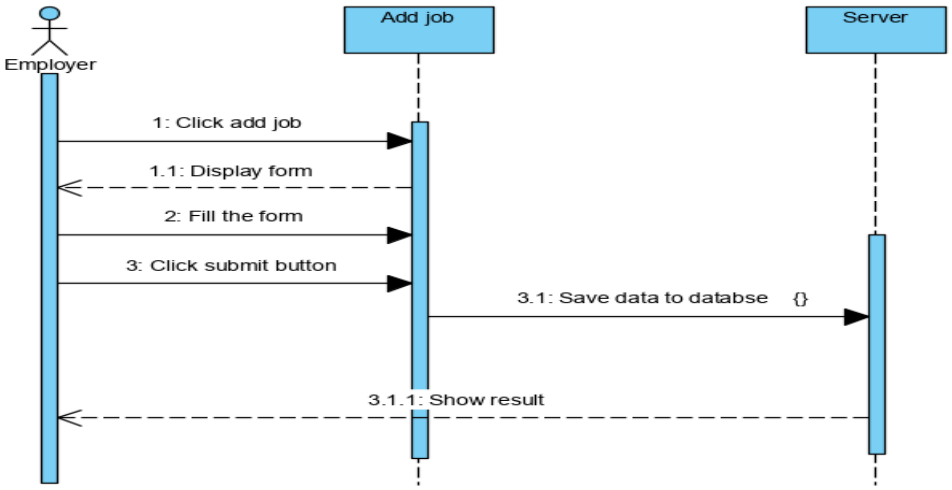


**Figure 14.** Add job sequence diagram

### 3.3.3 View Job Sequence Diagram

This sequence diagram explains how the user views jobs in the system. It shows how many steps must be taken to view available jobs. The user clicks the view jobs page and then the controller will check the user's role. If the user's role is the employer, the system displays only his/her posted jobs but, if the user's role is the job seeker the system displays all available jobs.

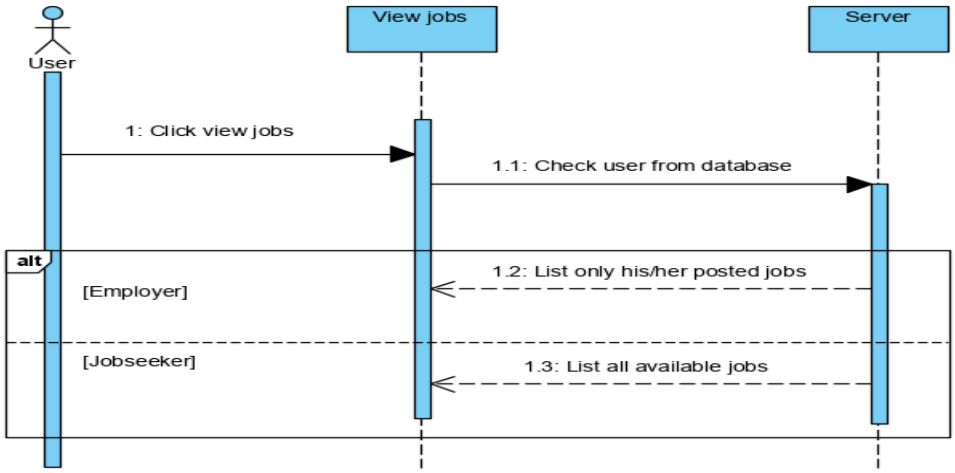


**Figure 15.** View job sequence diagram

### 3.3.4 Apply Job Sequence Diagram

The following sequence diagram explains how the employer interacts with the add job feature on the application and how many steps must be taken in order to add data to the system. The Employer fills the form and then clicks the submit button, the controller will save the data into the database and then the server will show the saved data to the employer.

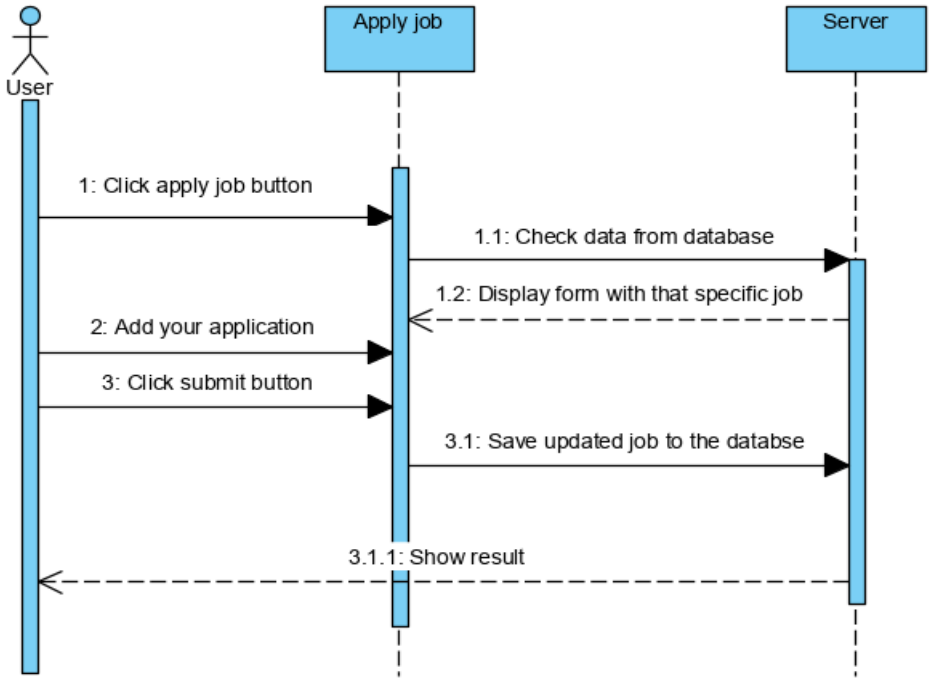


**Figure 16.** Apply job sequence diagram

### 3.3.5 Update Job Sequence Diagram

The below sequence diagram briefly illustrates how an employer interacts with the update job feature in the application and explains all the steps that must be taken in order to update data in the system. The Employer clicks the job that he/she wants to update then the system prompts that specific jobs form, then the employer updates the form and then clicks the submit button. The controller will save the data into the database and then the server will show the updated data to the employer.

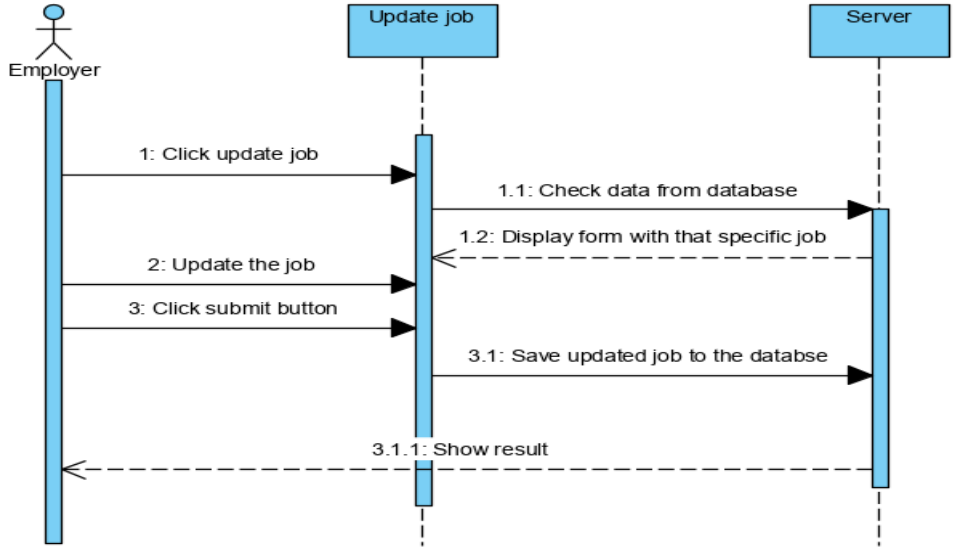


**Figure 17.** Update job sequence diagram

### 3.3.4 Delete Job Sequence Diagram

This is a sequence diagram which shows the messages involved in deleting a job in the application. The employer will click the delete job button, then the feature will delete only that specific job in the database.

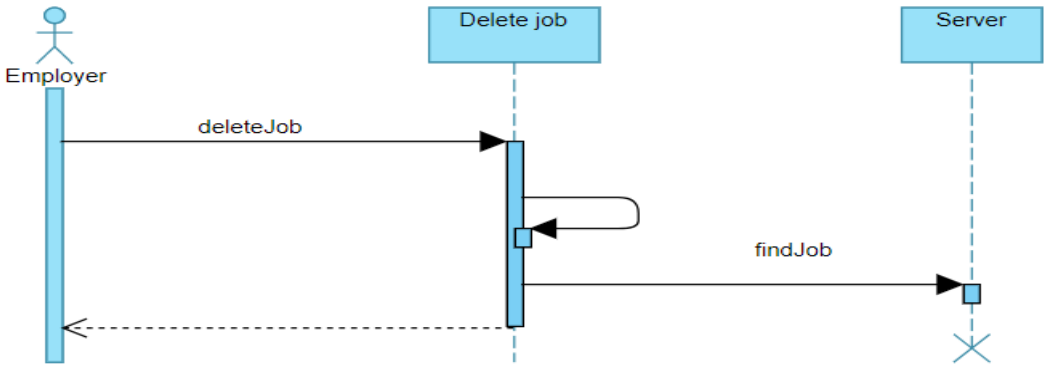


**Figure 18.** Apply job sequence diagram

# 4. DATABASE AND GUI DESIGN

This chapter will show us how the graphical user interface of the application was implemented and how it was built and configured for the database. This online recruitment application has a launch page for all and two different registration pages, one for the employer and the other for jobseeker and login page. Thymeleaf and Bootstrap were used to create the graphical user interface for this application.

## 4.1 Database Design

Database design is the process of producing a detailed data model of database and involves classifying data and identifying relationships between tables. To model and create a MySQL database, MySQL workbench was used. The model below contains tables that hold all the information on the Online Recruitment Application. The employers will be in the main table which will have all the information about employers. The job table will have all the information about jobs while the application table will have all the data related to the applications. There will be four joining tables in this application which will be many-to-many relationships as many jobseekers can have many jobs and vice versa.

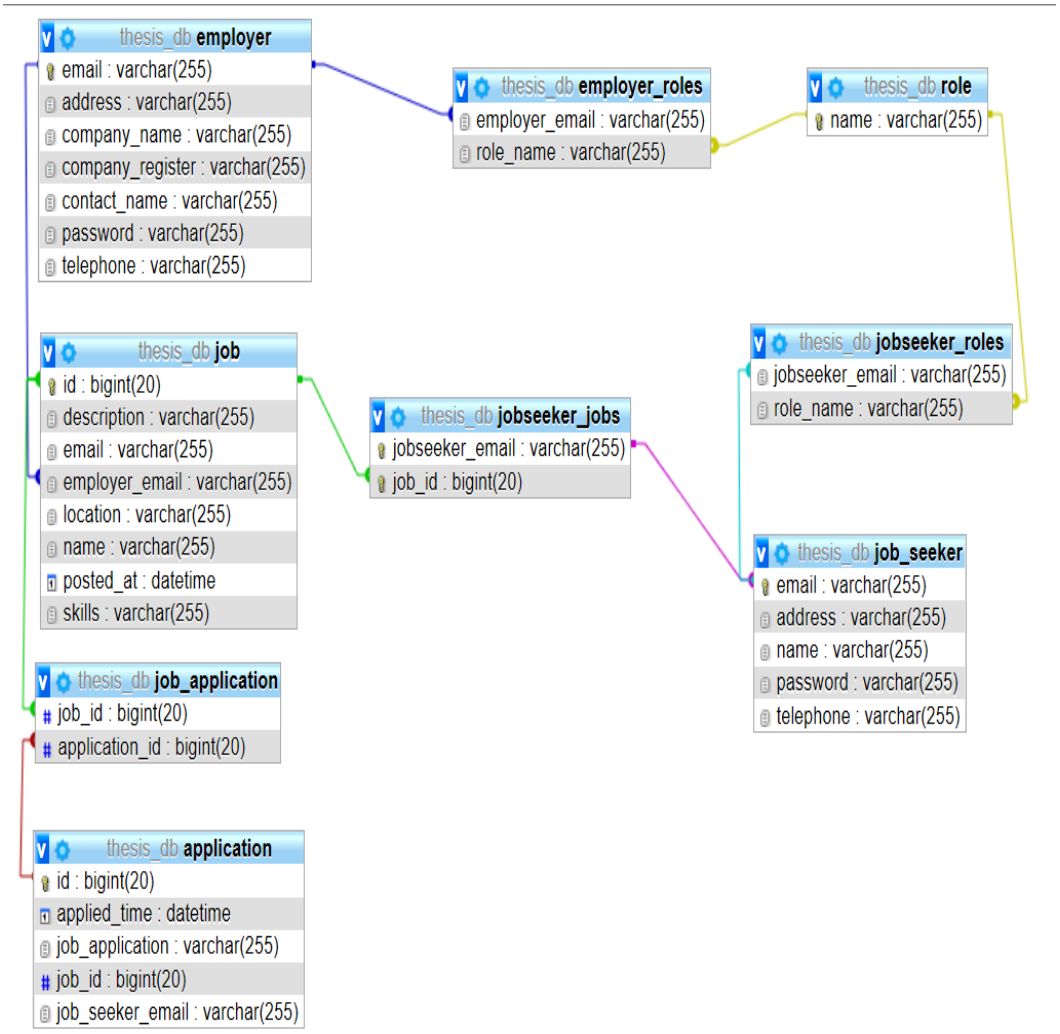


**Figure 19.** Database design model

## 4.2 Database Configuration

Spring Boot provides ready-made auto configuration to create datasource beans using application properties file. The code snippet below configuration shows how the MySQL database was configured. The application properties file starts by setting the path of the application and then specifies the database URL, which basically represents the location of the database. It also specifies the database username and password and some other configurations related to the applicaton. When the application is executed, Spring Boot will read this file and creates the data source bean.

```
#  The path of the application
server.servlet.context-path=/RecruitmentApplication
 # Connection url for the database
 spring.datasource.url=jdbc:mysql://localhost/thesis_db
 # username and password for the databse
 spring.datasource.username=root
 spring.datasource.password=
 # show each sql queries
 spring.jpa.show-sql=true
 # Hibernate ddl auto
 spring.jpa.hibernate.ddl-auto=update
 # sql statements and parameters
 logging.level.com.thesis.recruitment=TRACE
 logging.level.org.hibernate.SQL=DEBUG
 logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
 # login level
 logging.level.org.springframework.web=INFO
 logging.level.org.hibernate=INFO
 logging.file=../logs/mylog.log
```

**Code Snippet 1.** Database configuration and communication

## 4.3 Launch Page

The launch page is the starting point for this application any time a user needs to use this system. It has two pages one for employer and the other for jobseeker. This page does not need any authentication to be viewed but to proceeds to the application every user needs to login or register if he/she is a new user.

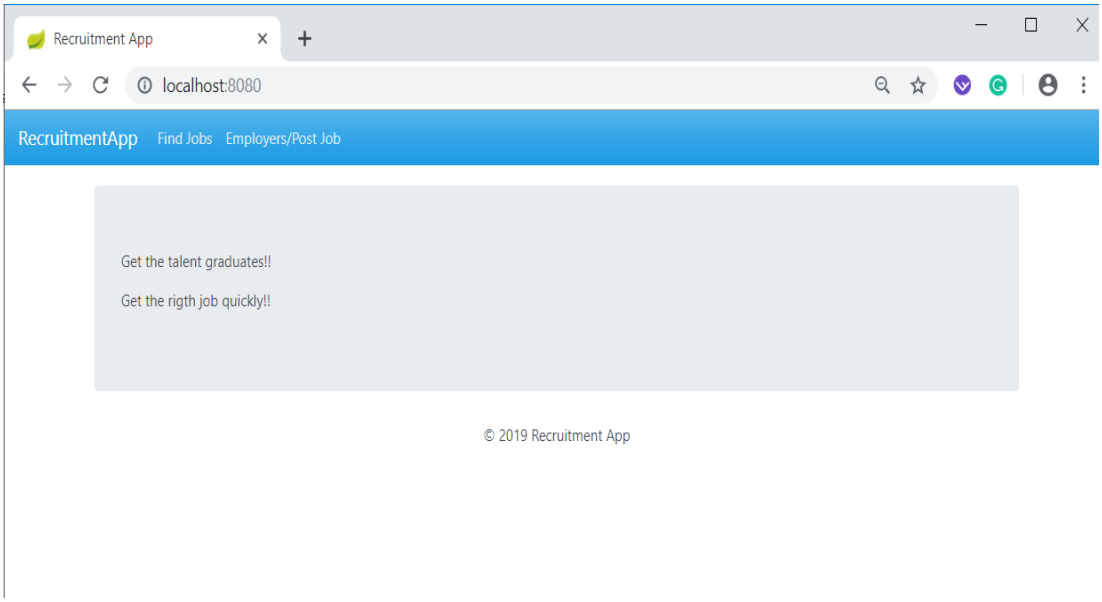


**Figure 20.** Launch page

## 4.4 Employer Home Page

After the employer has logged in successfully into the system the employer's home page shown below will be displayed so that employer can proceed with features, such as adding jobs, editing or deleting jobs and viewing added jobs or applications.

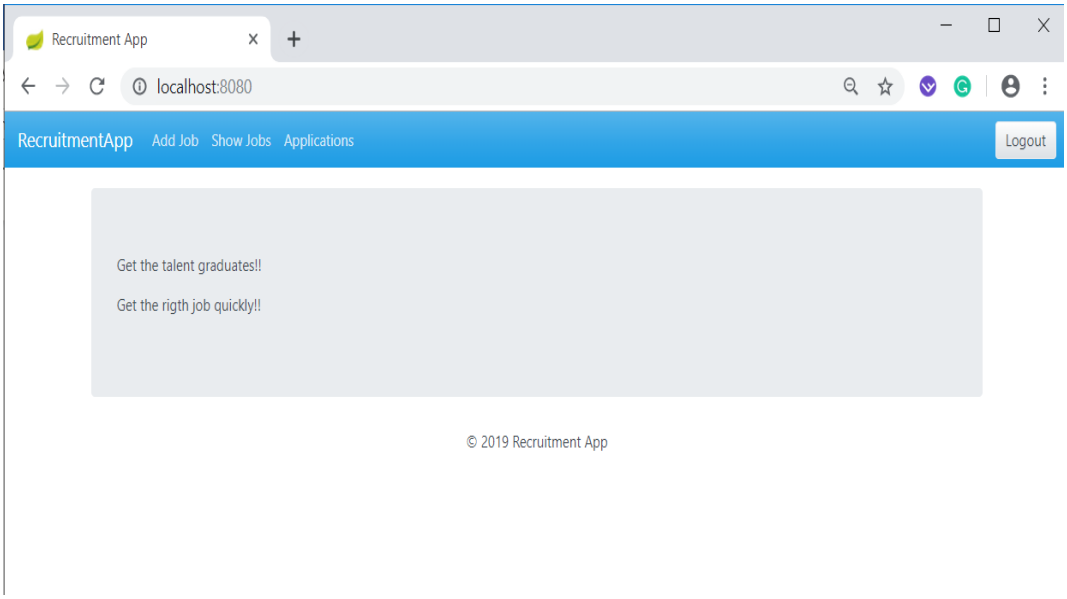


**Figure 21.** Employer home page

## 4.5 Jobseeker Home Page

After the Jobseeker logged in successfully into the system, the jobseeker home page shown below will be displayed, so that the jobseeker can proceed with features, such viewing jobs, applying jobs and viewing his/her applied jobs.

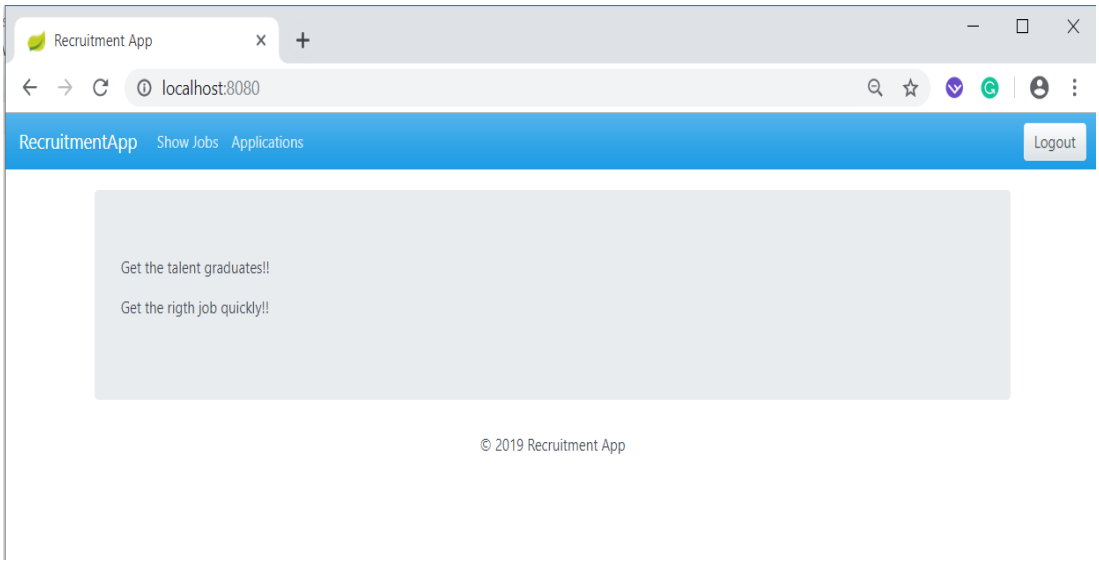


**Figure 22.** Jobseeker home page

# 5. IMPLEMENTATION

This section illustrates how the back-end of this application was implemented. Before writing any code, a Spring Boot project was created using the Spring tool suite IDE and all the important dependencies has been added to the project  This section describes and explains major classes and its methods in this application. The technologies used to implement this application is SPRING BOOT, BOOTSRAP, TYMELEAF and MySQL.

## 5.1 Spring Boot Application (Main Method)

The code below snippet is the entry point of this online recruitment application. The user can run this application by right-clicking the main Java file and clicking on run as Java application or Spring Boot application.

```
@SpringBootApplication
public class RecruitmentApplication extends SpringBootServletInitializer{
// override configure method of the SpringBootServletInitializer Class
@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
    return application.sources(RecruitmentApplication.class);
   }
//main method
public static void main(String[] args) {
      configureApplication(new SpringApplicationBuilder()).run(args);
            }
// turn of the banner
private static SpringApplicationBuilder configureApplication(SpringApplicationBuilder builder) {
      return builder.sources(RecruitmentApplication.class).bannerMode(Banner.Mode.OFF);
   }
}
```

**Code Snippet 2.** Main method of the application

## 5.2 Employer Controller

This Employer Controller class handles the registration process for employer. The Register Form Method returns the registration form shown in Figure 15 when a registration request is submitted to the server. The Register Employer Method creates an employer after checking that there is no validation error and the user does not exist using  isUserPresent" method in the EmployerService class and then returns success page as shown in the code

```
@Controller
public class EmployerController {
@Autowired
   private EmployerService employerService;
   // get the registration page
@GetMapping("/register")
   public String registerForm(Model model) {
   model.addAttribute("employer", new Employer());
   return "views/employer-registration";
}
   // registration for a new employer
   @PostMapping("/register")
public String registerEmployer(@Valid Employer employer, BindingResult bindingResult, Model
model) {
   if (bindingResult.hasErrors()) {
                        return "views/employer-registration";
   }
   //check if user exits already
   if (employerService.isUserPresent(employer.getEmail())) {
                        model.addAttribute("exist", true);
              return "views/employer-registration";
   }
   employerService.createEmployer(employer);
   return "views/success";
         }
}
```

**Code Snippet 3.** Employer registration code

The registration process for the jobseeker is the same as the employer registration process so the researcher will escape to avoid repeating the code that was already explained.

## 5.3 Add Job

The code below describes the process of adding jobs into the system. The addJobForm method returns the form shown in Figure 19 when add job request is submitted to the server. The addJobForm method creates a new job after checking that there is no validation error when submitting the form. In this class Logger is used in order to get the employer's email so that every employer will see only his/her posted jobs.

```
  @Controller
public class JobController {
  //initialize the logger
private static final Logger LOGGER = LoggerFactory.getLogger(JobController.class);
@Autowired
private JobService jobService;
@Autowired
private JobRepository jobRepository;
@Autowired
  private EmployerService employerService;
  // if the user has employer role then show this page
// get job-new page
  @PreAuthorize("hasRole('EMPLOYER')")
@GetMapping("/jobs/new")
  public String addJobForm(Model model) {
            model.addAttribute("job", new Job());
  return "views/job-new";
  }
  // check if the user has employer role
// adding new job
// get user's information by using getUserPrincipal
  @PreAuthorize("hasRole('EMPLOYER')")
  @PostMapping("/jobs/new")
```

```
  public String addJobForm(Principal principal, @Valid Job job, BindingResult bindingResult, Model
model){
  if (bindingResult.hasErrors()) {
            return "views/job-new";
  }
  String username = ((User) ((UsernamePasswordAuthenticationToken)
principal).getPrincipal()).getUsername();
          LOGGER.debug("username:{}", username);
          Employer employer = employerService.findOneByEmail(username);
          job.setEmployer(employer);
          jobService.addJob(job);
          return "redirect:/jobs";
          }
}
```

**Code Snippet 4.** add job code

## 5.4 View Job

The code below in the job controller class describes the process of viewing the jobs in the system when clicking the show jobs link. First the method checks the employer's email so that every employer will only see his/her posted jobs but jobseekers will see all available jobs. Employers can edit or delete jobs and jobseekers can apply for jobs by clicking the apply button.

```
  // get view job page
  // check first the user's role if is employer show only his/her posted jobs, show all jobs to jobseeker
   @GetMapping("/jobs")
public String showJobList(HttpServletRequest request, Model model, @RequestParam(defaultValue
= "")   String name) {
          boolean isEmployer = request.isUserInRole("ROLE_EMPLOYER");
            LOGGER.debug("Is Employer:{}", isEmployer);
            List<Job> jobs = null;
            if (isEmployer) {
String username = ((User) ((UsernamePasswordAuthenticationToken)
request.getUserPrincipal()).getPrincipal()).getUsername();
                    LOGGER.trace("username:{}", username);
            jobs = jobService.findByEmployerEmailAndName(username, name);
```

```
            } else {
            jobs = jobService.findByName(name);          }
            model.addAttribute("jobs", jobs);
        return "views/job-list";
    }
```

**Code Snippet 5.** View job code

## 5.5 Implementing Update and Delete Job

The code below in the job controller class describes the process of deleting or updating the job in the system. Here the @PreAuthorize annotation was used to verify the authorization before entering into this method. Only users who have an employer role will be allowed to delete or update jobs.

```
  // check if the user is employer
  //get a specific job's form
  @PreAuthorize("hasRole('EMPLOYER')")
  @GetMapping("jobs/{id}/update")
  public String showUpdateForm(@PathVariable("id") long id, Model model) {
Job job = jobRepository.findById(id).orElseThrow(()-> new IllegalArgumentException("Invalid Job Id:"
+ id));
            model.addAttribute("job", job);
            return "views/job-edit";
  }
  // update a specific job
  @PreAuthorize("hasRole('EMPLOYER')")
  @PostMapping("jobs/{id}/update")
  public String updateJob(@PathVariable("id") long id, @Valid Job job, BindingResult result, Model
model) {
            if (result.hasErrors()) {
                        job.setId(id);
                        return "views/job-edit";
            }
        jobRepository.save(job);
            model.addAttribute("jobs", jobRepository.findAll());
            return "redirect:/jobs";
    }
```

```
        // delete the specific job
   @PreAuthorize("hasRole('EMPLOYER')")
   @RequestMapping(value = "/jobs/{id}/delete", method = RequestMethod.GET)
   public String deleteJob(Model model, @PathVariable("id") Long id) {
            jobService.deleteJob(id);
            return "redirect:/jobs";
}
```

**Code Snippet 6.** Delete and update job code

## 5.6 Apply Job

The code below in the application controller class describes the process of applying for jobs in the system. Here the @PreAuthorize annotation was used to verify the authorization before entering into this method. Only users who have a jobseeker role will be allowed to apply. When the jobseeker clicks the apply button, the system prompts only that particular job.

```
// check if the user is jobseeker by using PreAuthorize annotation
// apply for job
// get user's information like name, telephone and email by using getUserPrincipal
   @PreAuthorize("hasRole('ROLE_JOBSEEKER')")
   @GetMapping("/jobs/{jobId}/apply")
   public String showApplyForm(HttpServletRequest request, @PathVariable("jobId") long jobId,
Model model) {
            String username = ((User) ((UsernamePasswordAuthenticationToken)
request.getUserPrincipal()).getPrincipal())
                        .getUsername();
            LOGGER.trace("username:{}", username);
         model.addAttribute("jobId", jobId);
         Job job = jobService.findById(jobId).orElseThrow(() -> new
IllegalArgumentException("Invalid Job Id:" + jobId));
            model.addAttribute("job", job);
         Application application = new Application();
            application.setJobSeekerEmail(username);
            application.setJobId(jobId);

            model.addAttribute("application", application);
```

```
            return "views/application-new";
}
```

**Code Snippet 7.** Apply job code

## 5.7 View Applications

The code below in the application controller class describes the process of viewing the applications in the system when clicking the Applications link. The Method checks both the employer and jobseeker username so that employers will only see applications for their posted jobs and jobseekers will only see applications for their applied jobs.

```
// get application's page, check user by using LOGGER and show every user only his data
// get user's information by using getUserPrincipal
@GetMapping("/applications")
  public String showApplications(HttpServletRequest request, Model model) {
            String username = ((User) ((UsernamePasswordAuthenticationToken)
request.getUserPrincipal()).getPrincipal())
                        .getUsername();
            LOGGER.trace("username:{}", username);
          boolean isEmployer = request.isUserInRole("ROLE_EMPLOYER");
            LOGGER.debug("Is Employer:{}", isEmployer);
          List<Application> applications = null;
            // if the user is employer then display only his/her posted job's application
            if (isEmployer) {
                        applications = applicationService.findAllByEmployer(username);
            }
        // for jobseekers display all jobs
        else {
                        applications = applicationService.findAllByJobSeeker(username);
            }
          model.addAttribute("applicashuns", applications);
          return "views/application-list";
}
```

**Code Snippet 8.** View applications code

## 5.8 Security Configuration

In the login process jdbc authentication and configure method were used. The configure method takes AuthenticationManagerBuilder as an argument. The security configuration class has been modified in order to use the jdbc datasource. Two queries were set up, one for authentication in usersByUsernameQuery and the other for authorization in authoritiesByUsernameQuery. As shown in the code snippet below passwords are saved in an encoded format.

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
        @Autowired
  private DataSource dataSource;

  // jdbc authentication and defining data source
      @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication().dataSource(dataSource)
                .usersByUsernameQuery(
"select email as principal, password as credentails, true from employer where  email=?")
                .authoritiesByUsernameQuery(
"select employer_email as principal, role_name as role from employer_roles where employer_email=?")
                .passwordEncoder(passwordEncoder()).rolePrefix("ROLE_");
          auth.jdbcAuthentication().dataSource(dataSource)
                .usersByUsernameQuery(
"select email as principal, password as credentails, true from job_seeker where email=?")
                .authoritiesByUsernameQuery(
"select jobseeker_email as principal, role_name as role from jobseeker_roles where jobseeker_email=?")
                .passwordEncoder(passwordEncoder()).rolePrefix("ROLE_");
        }
        // password hashing
      @Bean
          public PasswordEncoder passwordEncoder() {
                return new BCryptPasswordEncoder();}
```

```
            // enable HTTP security
            @Override
            protected void configure(HttpSecurity http) throws Exception {
  http.authorizeRequests()
.antMatchers("/", "/css/**", "/webjars/**", "/register", "/registerJobseekers", "/login").permitAll()
                        .anyRequest().authenticated().and().formLogin().loginPage("/login").permitAll()
                        .defaultSuccessUrl("/jobs").and().logout().logoutSuccessUrl("/login");
  }
}
```

**Code Snippet 9.** Security Configuration code

The last method in the SecurityConfig class was modified in order to define the request that should be authenticated and the one that should not be authenticated.

# 6. TESTING

This chapter briefly describes the test cases for both the employer and jobseeker features of this application.

## 6.1 Employer Registration Page

The following registration form will be shown after clicking the register link, then the new employer fills up all the fields of this form and clicks submit button, the registration controller will then save the data into MySQL. If nothing or wrong data has been entered, then the system will display error message as shown in the figure below.

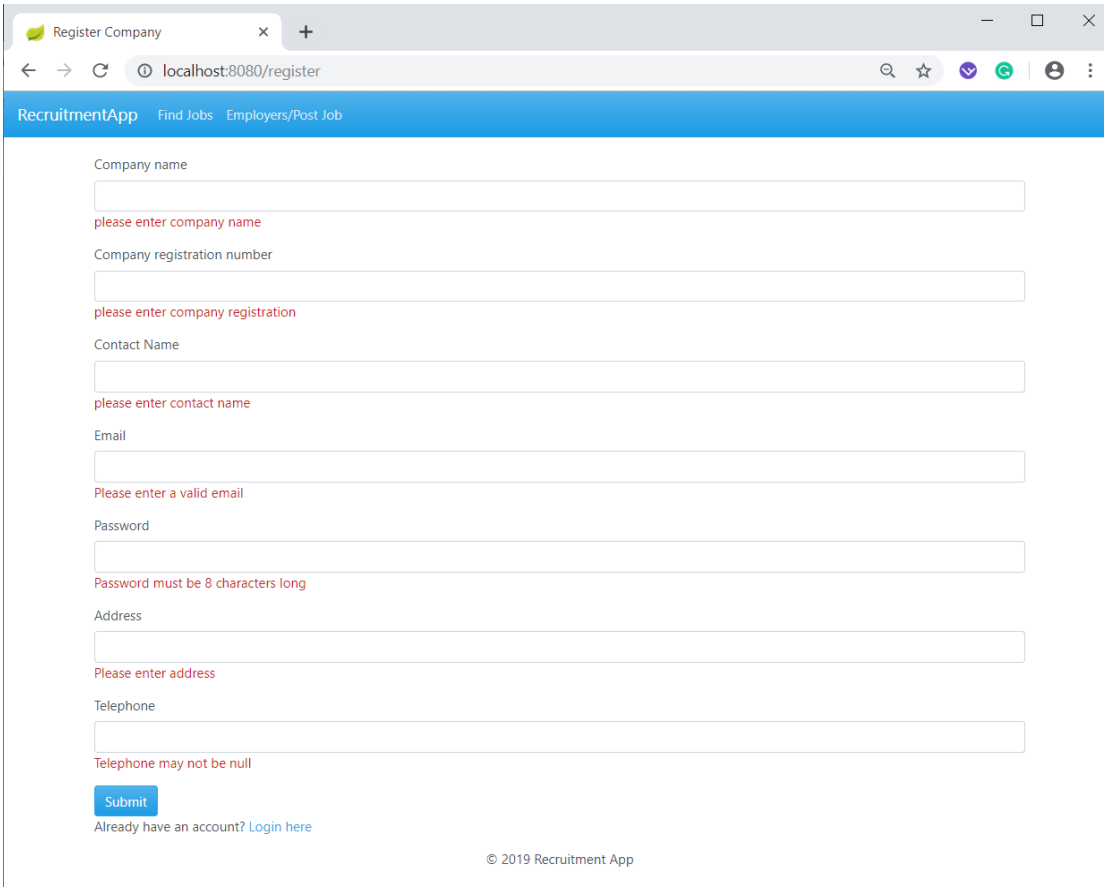


**Figure 23.** Employer registration page

If every thing goes well when a new user registers into the system this success registration message will be shown.

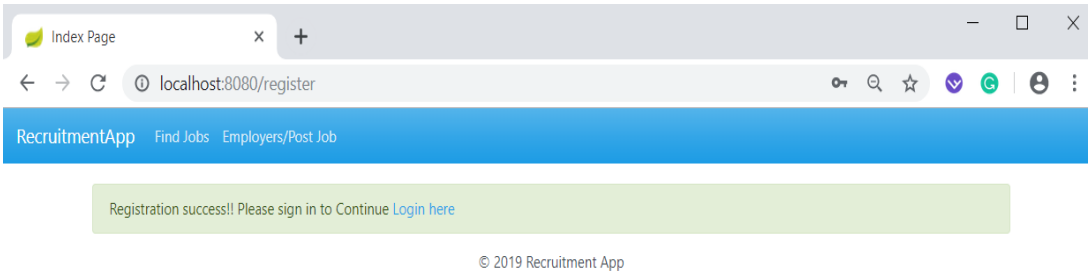


**Figure 24.** Success registration message

## 6.2 Jobseeker and Employer Login Page

The login page is for both the jobseeker and employer. It was designed using Thymeleaf and Bootstrap and allows both the jobseeker and the employer to enter their valid email and password, then the login controller will redirect to their home pages respectively. If nothing or wrong data has been entered, then the system will display error message as shown in the figure below.

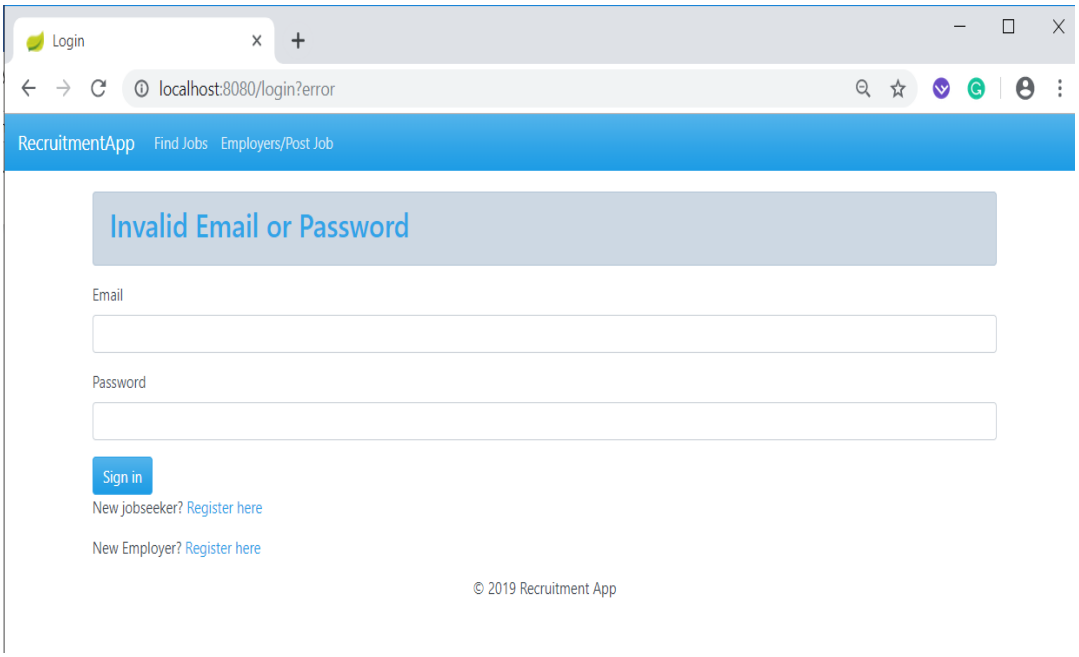


**Figure 25.** Login page

## 6.3 Add Job

This add job form will be shown after clicking the add job button. If the employer clicks the submit button without filling up any of these fields, the system will

prompt validation error messages as shown below but if the employer fills up all these required fields of this form and clicks the submit button, the job controller will then save the data into MySQL.

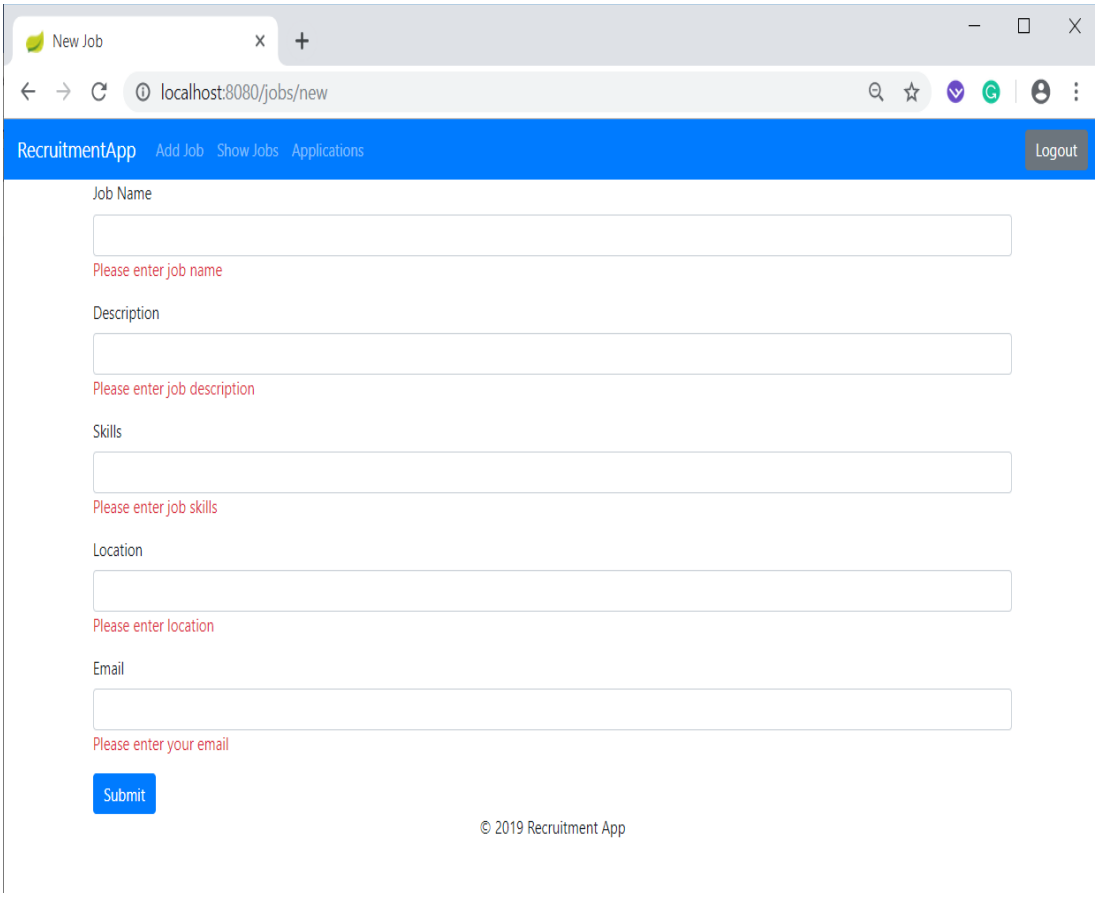


**Figure 26.** Add job page

## 6.4 Show Jobs

The page below shows the list of available jobs in details with editing and deleting features. After the employer clicks the edit button the system prompts editable form, the employer can edit every field of the form. The employer can also delete the job by clicking the delete button. Only those with an employer role can edit or delete jobs.

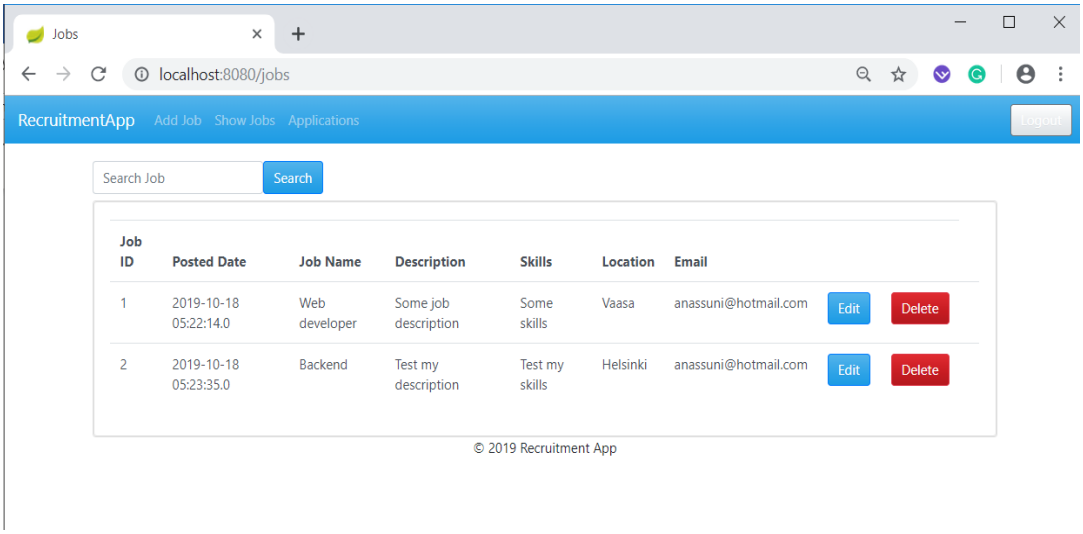


**Figure 27.** View jobs page

## 6.5 Jobseeker Registration Page

The following registration form will be shown after clicking the register link, then the new jobseeker fills up all the fields of this form and clicks the submit button. The registration controller will then save the data to MySQL.

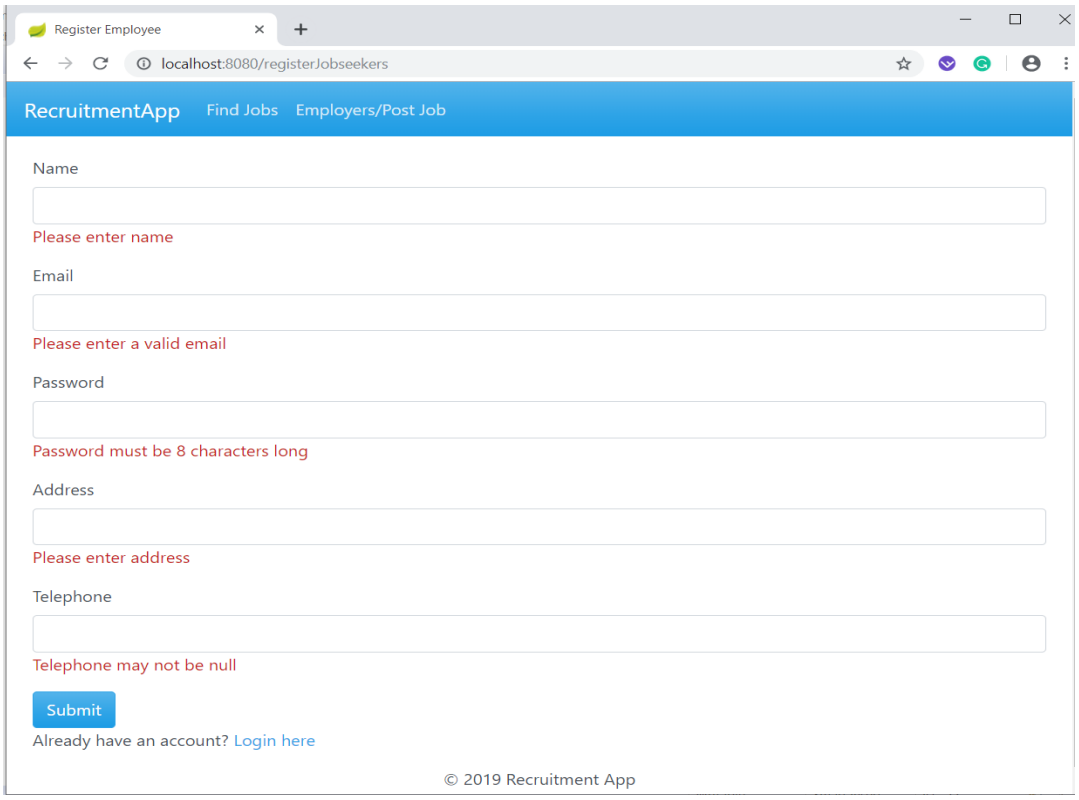


**Figure 28.** Jobseeker registration page

## 6.6 Jobseeker Show Jobs Page

When the jobseeker clicks the show jobs link, the system will display the list of available jobs in details with the apply feature that allows the jobseeker to apply his/her desired job in the list. The show jobs page also has a search feature which can be used to filter jobs by typing the name of the job. Figure 29 shows how the jobseeker show jobs page looks like after the link has been clicked.

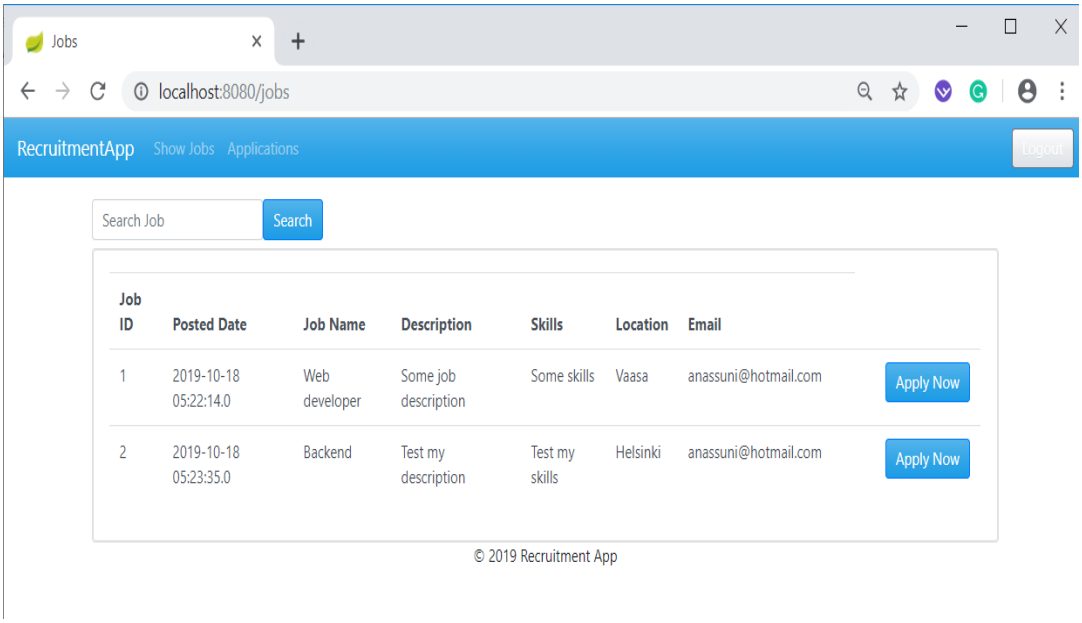


**Figure 29.** Jobseeker show jobs page

## 6.7 Jobseeker Apply Job Page

If the jobseeker clicks the Apply now button in the job list, then the system will display the clicked job with its uneditable unique details and one editable tex tarea. The jobseeker can add more information about his/her application if needed and click submit, then the application will be added to the list of applications. Figure 30 illustrates the view when the Apply Now button is clicked.

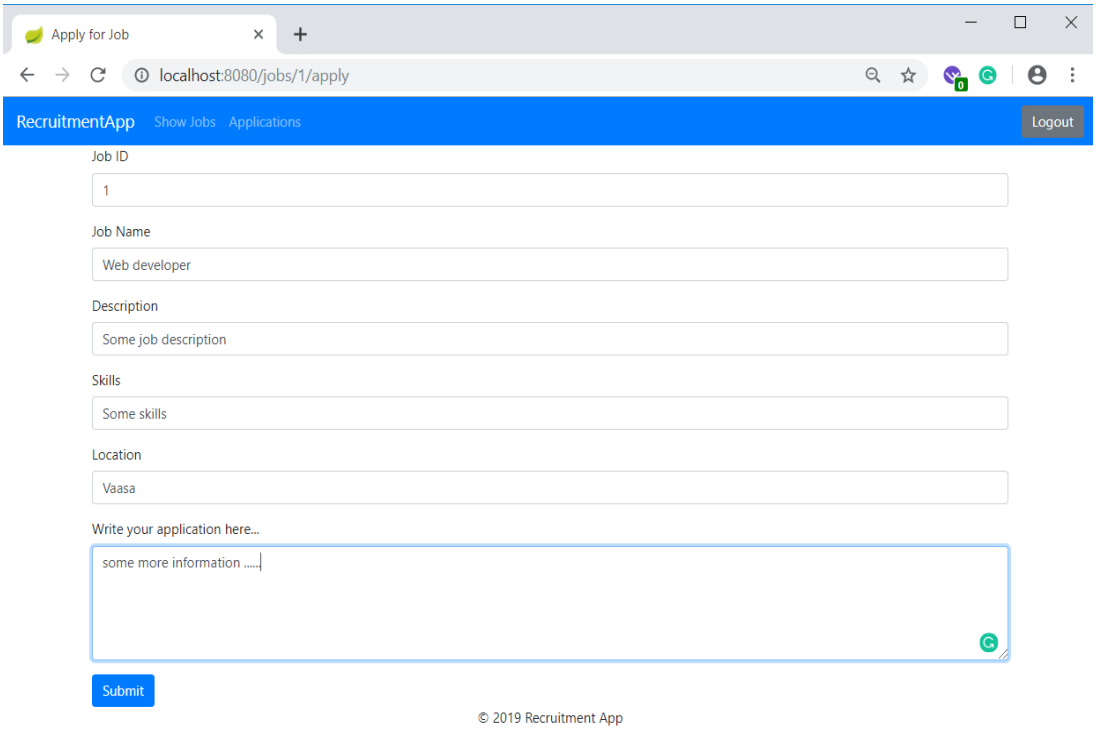


**Figure 30.** Apply to Job

## 6.8 Applications

The Applications page displays the list of applied jobs. After clicking the applications page the system displays to the jobseeker only the lists of his/her applied jobs, while the employer views applications of only his/her posted jobs based on their roles and emails.

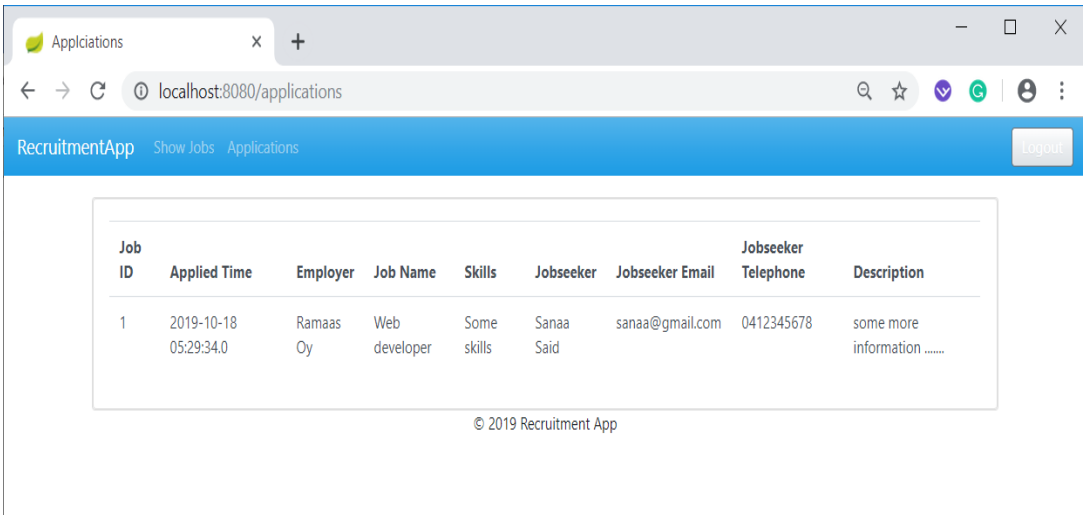


**Figure 31.** Applications

# 7. CONCLUSION

This recruitment application allows both the employer and jobseeker to sign up, sign in and log out of the application. When the employer registers to the system, the inputted data is first validated. The employer can add jobs to the system and his/her posted jobs are stored in the database and the employer can view and filter his/her job history at any time. On the other hand, when the jobseeker registers to the system, the inputted data is also validated. The jobseeker can apply for jobs through the system and his/her applied jobs are stored in the database and finally can view his/her applied job history.

The challenges faced while creating this application included that the used technology was not familiar to me before engaging in the research. I started learning Spring Boot and Thymeleaf from scratch which was a huge task and time consuming and then using them to build the application. This was the first time that I have created a complete application from start to finish for real use. However, the application is set to be simple for both employers and job seekers and all the required steps that had to be delivered for the project in order to be considered successful were achieved and tested.

## 7.1 Future work

This application can be improved by adding all optional requirement features in the application description chapter. Specific features should also be allocated, implemented and tested. For instance, features, such as a communication tool between the employers and job seekers could help in minimizing the processing time. Before submitting the application, the job seeker might need details about the specific requirements and tasks. At the same time, the employer could have inquiries about the submitted application. Therefore, creating a communication tool such as a chatting window could help both sides.

# REFERENCES

/1/ Hormuud Telecom Somalia. Accessed 24.01.2019 https://www.hormuud.com/personal/services/evc-plus.aspx

/2/ Recruitment definition. Accessed 29.01.2019. http://www.businessdictionary.com/definition/recruitment.html

/3/ Java programming language. Wikipedia. Accessed 30.1.2019. https://en.wikipedia.org/wiki/Java_(programming_language)

/4/ What is spring framework. Wikipedia. Accessed 01.01.2020 https://en.wikipedia.org/wiki/Spring_Framework

/5/ What is Spring boot. Accessed 10.2.2019 https://dzone.com/articles/what-is-spring-boot

/6/ Spring Initializr. Accessed 09.01.2020 https://www.javatpoint.com/spring-initializr

/7/ Thymeleaf. Accessed 10.2.2019 https://www.thymeleaf.org/

/8/ Turorial: Using Thymeleaf https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html

/9/ Bootstrap. Wikipedia. Accessd 10.3.2019 https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

/10/ Bootstrap. Sitepoint. Accessed 10.1.2020 https://www.sitepoint.com/why-i-love-bootstrap-you-should/

/11/ MySQL. Wikipedia. Accessed 14.4.2019 https://en.wikipedia.org/wiki/MySQL

/12/ Relational database management system. Wikipedia. Accessed 12.5.2019 https://en.wikipedia.org/wiki/Relational_database_management_system

/13/ Eclipse Overview. Accessed 12.7.2019 https://www.tutorialspoint.com/eclipse/eclipse_overview.htm

/14/ Eclipse tutorials. Accessed 09.01.2020 https://www.vogella.com/tutorials/Eclipse/article.html