



# **Software Design & Architecture**

**COMSATS UNIVERSITY ISLAMABAD, WAH  
CAMPUS**

**Architecture Document (Recruitee)**

**Submission To: Maam Samra Siddiqui**

**Submission Date: Dec 22, 2022**

**(Thursday)**

# GROUP MEMBERS

**SP20-BSE-047**  
HAMZA ALTAF

**SP20-BSE-051**  
TALHA NAEEM BUTT

**SP20-BSE-069**  
OSSAMA MEHMOOD

**SP20-BSE-081**  
ZAEEM TAHIR

**SP20-BSE-091**  
SAAD ZIA

## “Table of Content”

<b>1. Project Context .....</b>	<b>3</b>
1.1. <i>Introduction .....</i>	<i>3</i>
1.2. <i>Purpose.....</i>	<i>3</i>
1.3. <i>Scope .....</i>	<i>4</i>
<b>2. Architecture Requirements .....</b>	<b>5</b>
2.1. <i>Overview of Key Objectives .....</i>	<i>5</i>
2.2. <i>Architecture Use Cases.....</i>	<i>5</i>
2.3. <i>Stakeholder Architectural Requirements.....</i>	<i>6</i>
2.4. <i>Constraints.....</i>	<i>8</i>
2.5. <i>Non-functional Requirements .....</i>	<i>8</i>
2.6. <i>Risks.....</i>	<i>9</i>
<b>3. Solution .....</b>	<b>10</b>
3.1. <i>Relevant Architectural Patterns .....</i>	<i>10</i>
3.2. <i>Architecture Overview .....</i>	<i>12</i>
3.3. <i>Structural Views.....</i>	<i>15</i>
3.4. <i>Behavioral Views .....</i>	<i>18</i>
3.5. <i>Implementation Issues .....</i>	<i>21</i>
<b>4. Architecture Analysis .....</b>	<b>21</b>
4.1. <i>Scenario analysis.....</i>	<i>21</i>
4.2. <i>Risks.....</i>	<i>22</i>
<b>5. Architecture Diagram &amp; Design Pattern .....</b>	<b>24</b>
5.1. <i>Architecture Diagram &amp; Working.....</i>	<i>24</i>
5.2. <i>Design Pattern .....</i>	<i>25</i>

# 1. Project Context

## 1.1. Introduction

Recruitee is a job portal for finding your dream job here Various career opportunities await you. Finding the right career and connecting with companies anytime and anywhere also known as a job portal or employment website, is a platform that allows job seekers to search and apply for job openings, and employers to post job listings and receive applications from potential candidates.

Recruitee can be general, covering a wide range of industries and job types, or specialized, focusing on a specific field or type of work. Many job portals also offer additional resources and tools for job seekers, such as uploading the resume.

## 1.2. Purpose

Recruitee is a job portal that is designed to help companies streamline their recruitment process and find top talent. It offers a range of tools and features to assist with job posting, applicant tracking, and candidate evaluation. Some of the key purposes of Recruitee include:



**Job posting:** Recruitee allows companies to post job openings and specify the required qualifications and skills.


**Applicant tracking:** The recruitee tracks the status of each job application and provides tools to help companies keep track of the hiring process.

**Candidate evaluation:** Recruitee offers features to help companies evaluate candidates, such as the ability to create custom interview guides and review resumes and cover letters.

Overall, the purpose of Recruitee is to make it easier for companies to find, evaluate, and hire the best candidates for their open positions.

### 1.3. Scope

The portal offers a range of features including job posting, applicant tracking, resume management, candidate communication, and collaborative hiring tools. It is designed to be used by small and medium-sized businesses as well as large enterprises and is built on the MVC module and based on Laravel.



However, Recruitee allows businesses to easily manage their recruitment efforts in one place, identify the most qualified candidates, and make better hiring decisions.

## **2. Architecture Requirements**

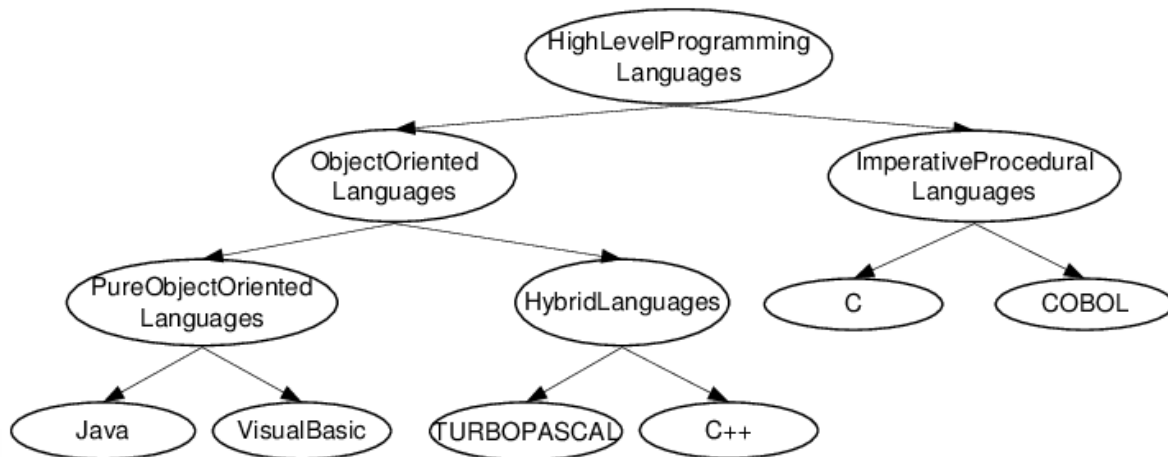
### **2.1. Overview of Key Objectives**

The main objective of the Job Portal System is to manage the details of Interviews, Employer, Call Letters, Employer Registration, and Search Jobs. It manages all the information about Interviews, Post Jobs, Search Jobs, and interviews. The project is totally built at the administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Interview, Employer, Post Job, and Call Letter. It tracks all the details about the Call Letter, Employer Registration, and Search Job.

### **2.2. Architecture Use Cases**

The architecture of the Job Portal Information Providers who publish open positions and company background information in the RDF format using controlled vocabularies.

They have two different approaches for publishing annotated job postings depending on their existing software infrastructure. If they use database standard software in the back end, they can export related data directly into RDF using mapping tools like D2RQ. If they do not use any enterprise software



## 2.3. Stakeholder Architectural Requirements

There are a number of stakeholders that may have specific architectural requirements for a job portal, including:

### **Job seekers:**

Job seekers may require the ability to easily search and apply for jobs, as well as to create and update their resumes and profiles. They may also want to be able to receive notifications of new job opportunities and to save and track their job search activities.



### **Employers:**

Employers may want to be able to post job openings, search and filter resumes and profiles, and communicate with job seekers. They may also want to be able to manage their recruitment efforts and track the status of job applicants.

### **Job portal administrators:**

The administrators of the job portal may have requirements for managing the site and its content, including the ability to add and edit job listings and resumes, as well as to track and analyze site usage and performance.

**Technical stakeholders:** The technical stakeholders in a job portal may include the developers and IT staff responsible for building and maintaining the site. They may have specific requirements related to the technology and infrastructure used to power the site, as well as security and privacy considerations.

**Regulatory stakeholders:** Depending on the location and industry of the job portal, there may be regulatory stakeholders who have requirements related to the handling of personal data, accessibility, and other legal considerations.



Overall, the architectural requirements for a job portal will depend on the specific needs and goals of these stakeholders, as well as any industry-specific or legal requirements that must be met.

## 2.4. Constraints

The following are the Constraints of the project:

- ✓ The interface is provided only in English, so the user should know English.
- ✓ A registered user only has the right to access the facilities provided by the system.
- ✓ The user can access the job recruiter system on any computer that has an internet connection.

## 2.5. Non-functional Requirements

### **Efficiency Requirement:**

When a “Job portal and recruitment system” is implemented job seekers and employees can view and upload job vacancies easily from their homes.



### **Usability Requirement:**

The system is designed for a user-friendly environment and ease of use.

### **Reliability Requirement:**

The system should provide a reliable environment for both job seekers and employers. All jobs should be reaching the admin without any errors and be shown to the viewers.

### **Database security:**

An unauthorized person cannot access the database and cannot read and write information.

### **Availability:**

The system should be available 24/7.

## **2.6. Risks**

### **Misrepresentation of job candidates:**

Dishonest job candidates can fabricate academic records, work experience, skills, and expertise online as they can on paper.

## Overlooking employee fraud:

Employee fraud can occur when an employee commits criminal acts against the company for the purpose of personal or financial gain.

## 3. Solution

### 3.1. Relevant Architectural Patterns

Relevant architecture patterns for a job portal will depend on the specific needs and goals of the application. Consider factors such as performance, scalability, cost, and maintainability when deciding which architecture is most appropriate.

#### Model View Controller Architecture Pattern

MVC is suitable for the website “Recruitee” as our website

As we break it down into **three (3)** main components as your code is easier to maintain.

People from the Front-End and Back-End Development Teams can focus only on the View and Model components respectively as it will not only avoid duplication but also make it easier to write code.

- ❖ MVC supports reusability

- ❖ MVC has a clean and clear structure that any developer is easily accustomed to and picks it up rather quickly.

However, the two (2) most relevant architectures for Recruitee are given below with their justification.

### Client-Server Architecture

Client-server architecture is a computing model in which a server provides resources and services to one or more clients over a network.

- ❖ The client is responsible for presenting information to the user and handling user input,
- ❖ While the server is responsible for storing and processing data and providing access to resources and services.

Overall, while the client-server architecture may be suitable for some types of applications, it may not be the best choice for a job portal due to its potential scalability, complexity, and reliability issues.

### Layer Architecture

The potential downside of layer architecture is that it can add complexity to the application,

- ❖ As it requires the implementation of multiple layers with distinct responsibilities and interfaces.

- ❖ Make the application more difficult to understand and maintain, particularly for developers who are not familiar with the architecture.

In addition, layer architecture may not be the most efficient choice for job portal websites with very high levels of traffic or very large numbers of users.

### 3.2. Architecture Overview

#### Model View Controller

We will choose MVC Architecture for our website “Recruitee” as It helps you to avoid complexity and simplifies the testing process by dividing our application into the three units Model, view, and controller and also provides developers the freedom to write and add functionality with ease.

It will increase development speed as Developers can work on one section of code while others can code on the other section.

For our website we used PHP Framework Laravel that is based upon MVC Architecture Pattern.

#### View

On the View Component the Front-End of the website will be displayed.

### Includes:

- Job Search
- Post Job
- Other Views (Home, About, Contact)

### Designing:

The Designing of Front-End will be done by Front-End Developers.

### **Model**

The model contains all the data-related logic that the user works with, like the schemas and interfaces of a project, the databases, and their fields.

### Includes (Database):

- List of Jobs
- Registered Users

### **Controller**

Controllers act as an intermediary between the model and View. Model and View don't interact with each other directly. It handles all the incoming requests.

- Contains all the business-related logic.

### Application Logic:

Part of the program which encodes the real-world application rules.

### Post Jobs:

#### Inputs:

Title: Required

Website: Required

Description: Required

location: Required

#### Conditions:

User posting job should be an authenticated user.

Company id should be verified.

Registration:

#### Input:

Username: Required

Email: Required

Password: Required

### Condition:

Email needs to be verified by sending a link to the email.

### Others:

Other business logic will be the basic CRUD Operations.

### **Working of MVC**

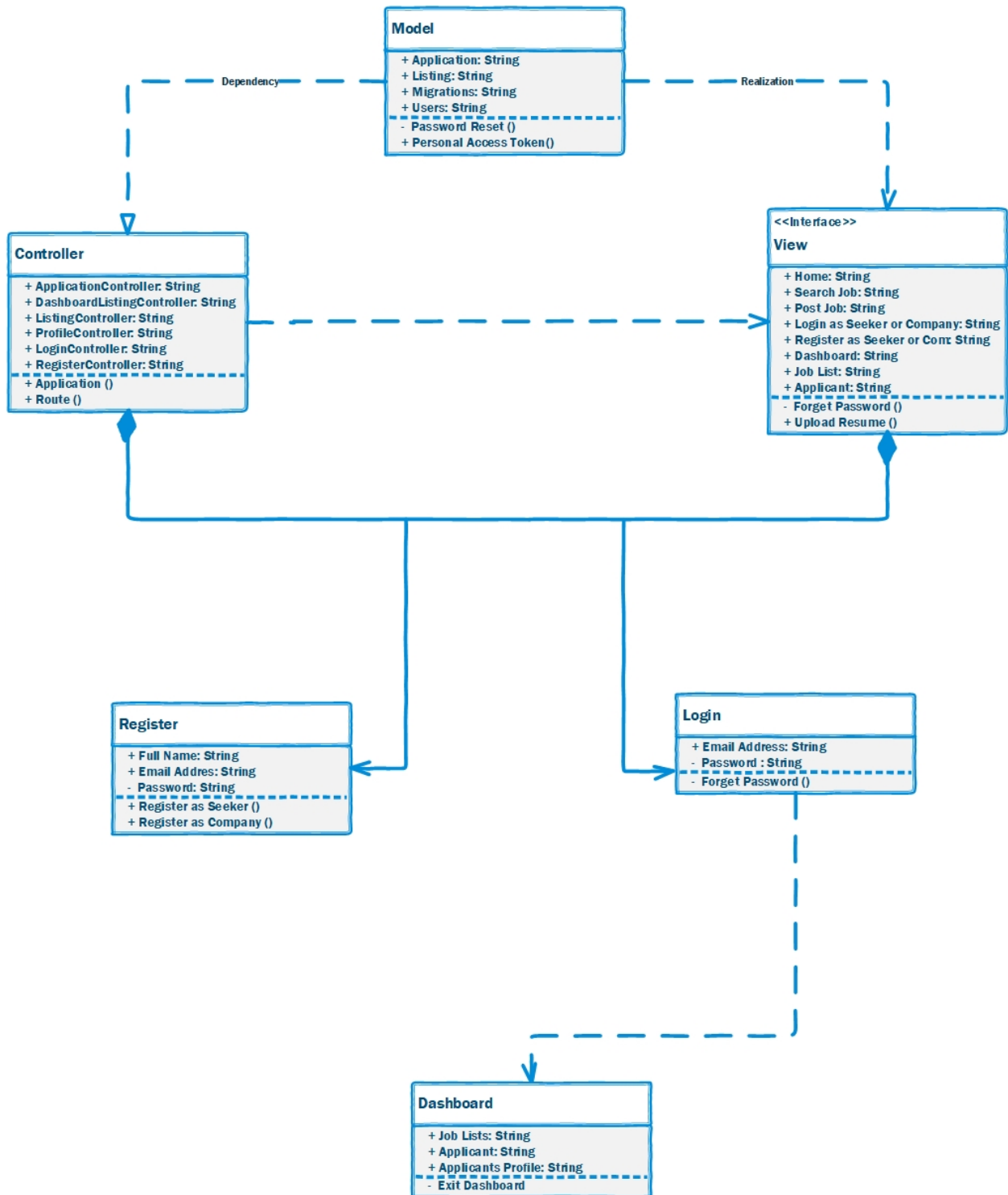
First, the browser sends a request to the Controller. Then, the Controller interacts with the Model to send and receive data.

The Controller then interacts with the View to render the data. The View is only concerned about how to present the information and not the final presentation. It will be a dynamic HTML file that renders data based on what the Controller sends it.

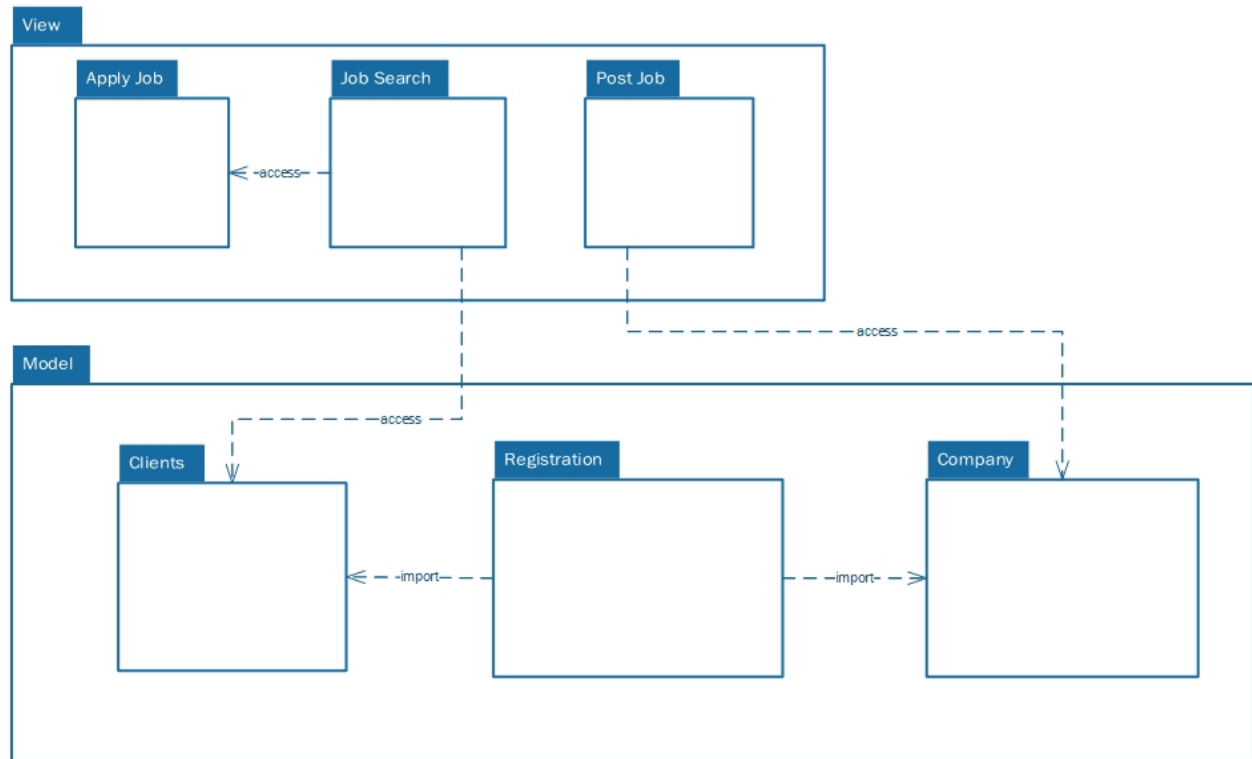
### **3.3. Structural Views**

### **Class Diagram**

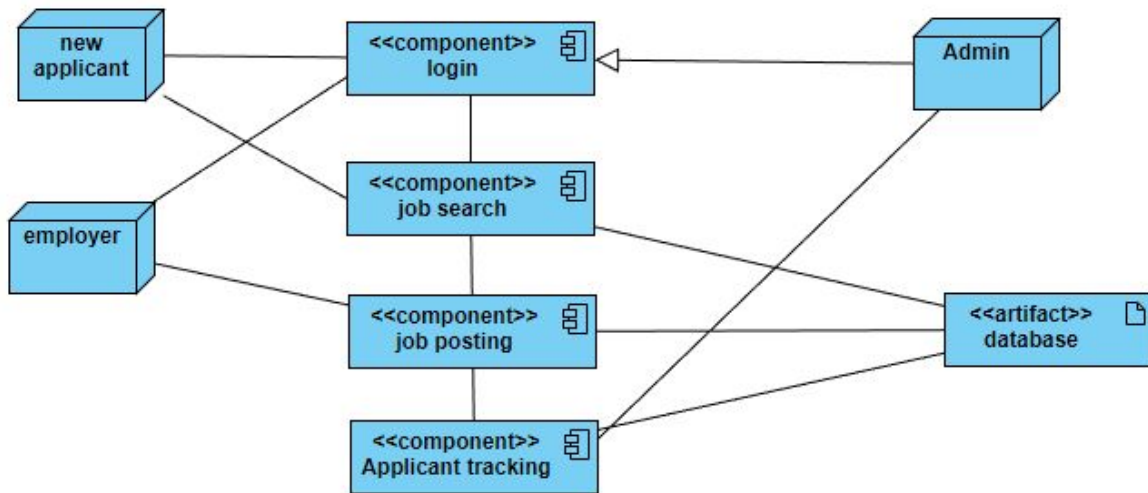




## Package Diagram

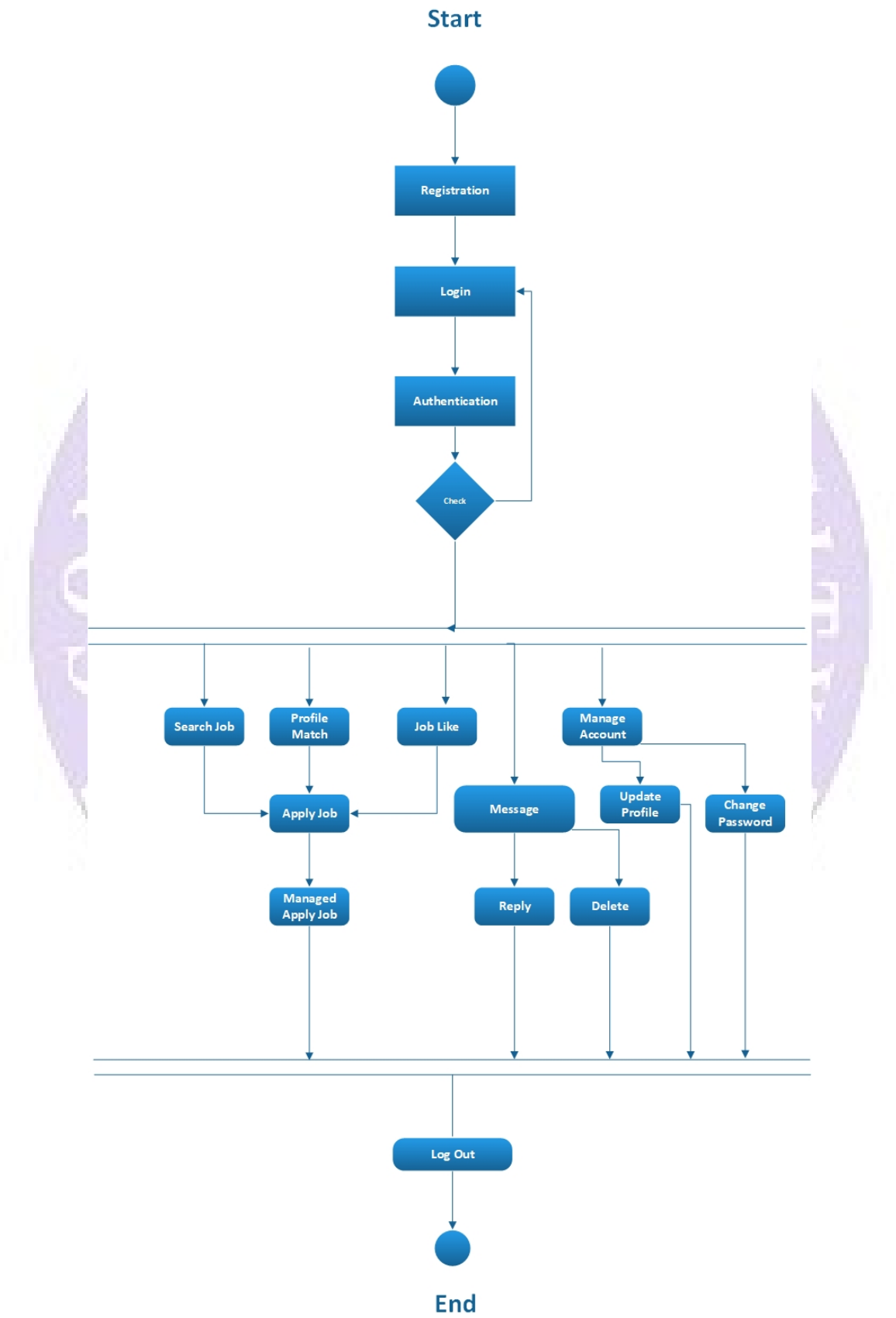


## Deployment Diagram

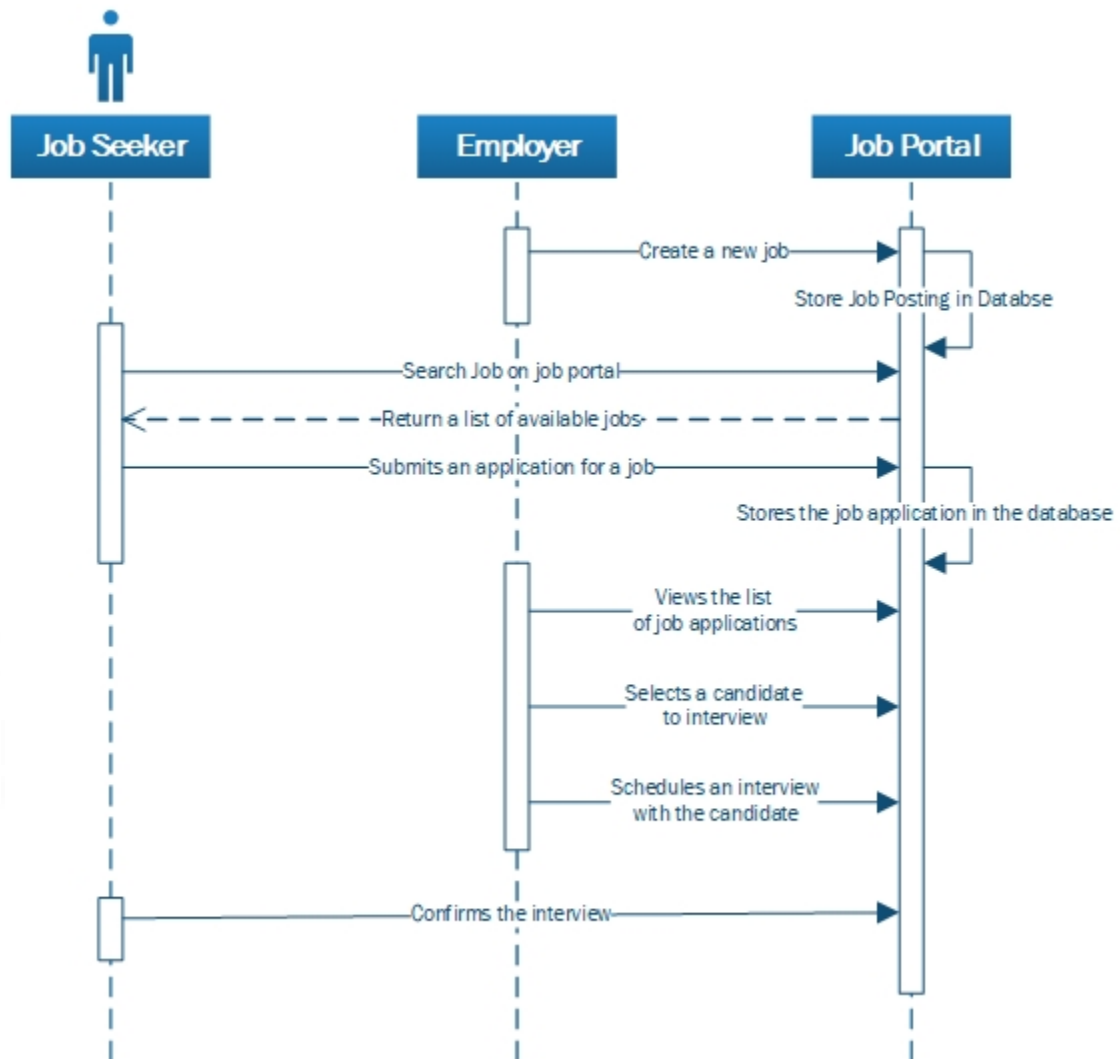


### 3.4. Behavioral Views

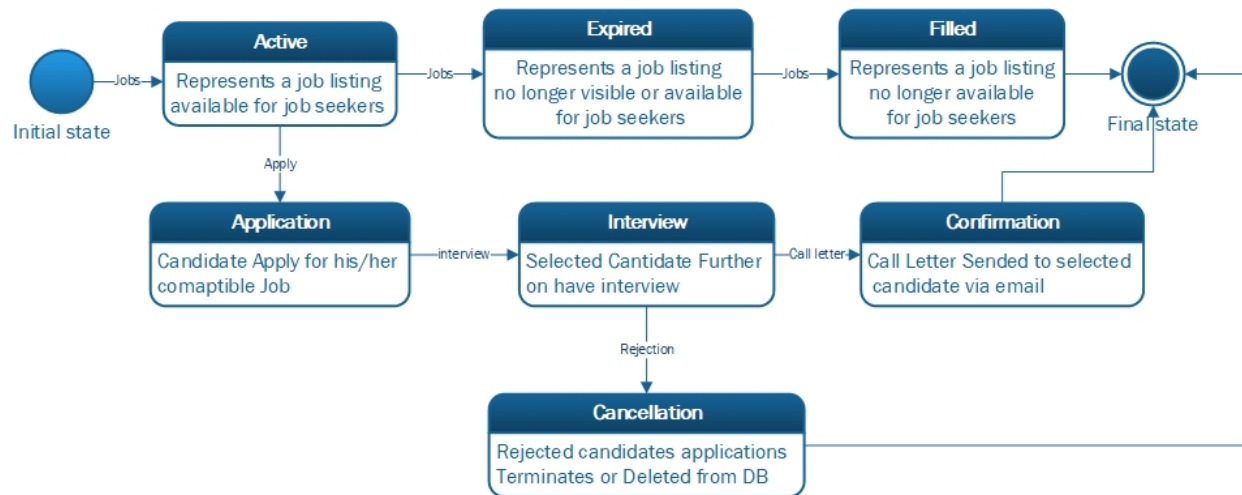
#### Activity Diagram



## Sequence Diagram



## State Machine Diagram



### 3.5. Implementation Issues


In MVC there are issues related to Scalability when we need to add new functionalities, we need to update the whole components as a whole instead of individual components.

## 4. Architecture Analysis

### 4.1. Scenario analysis

Here are some examples of scenarios that a job portal like Recrutee might consider in a scenario analysis:

**Increase in job postings:** If the job portal experiences a sudden increase in job postings, it may need to consider how to handle the additional workload and ensure that all postings are processed and made available to job seekers in a timely manner.



**Decrease in job postings:** If the job portal experiences a decrease in job postings, it may need to consider how to maintain revenue and profitability. This could involve finding new ways to generate income, such as through advertising or premium services.


**Increased competition:** If the job portal faces increased competition from other job portals, it may need to consider how to differentiate itself and maintain its market share. This could involve improving the user experience, adding new features, or promoting the benefits of using the portal.

**Changes in employment laws:** If there are changes in employment laws or regulations that affect the job portal's operations, it may need to consider how to comply with these changes and ensure that it is operating within the law.

By considering a range of potential scenarios and their potential impacts, a job portal like Recrutee can be better prepared to respond to challenges and opportunities as they arise.

## 4.2. Risks

Each of the scenarios mentioned above carries with it certain risks that the job portal should consider. Here are some examples of risks associated with each scenario:



**Increase in job postings:** The risk associated with an increase in job postings is the potential for a workload that is too high to handle efficiently. This could lead to delays in posting new jobs or responding to job seekers, which could result in a negative user experience.

**Decrease in job postings:** The risk associated with a decrease in job postings is the potential for a decrease in revenue and profitability. This could lead to financial strain on the job portal and may require cost-cutting measures or the need to find new sources of income.

**Increased competition:** The risk associated with increased competition is the potential for the job portal to lose market share to its competitors. This could result in a decrease in revenue and profitability, as well as a decrease in the number of job seekers and employers using the portal.

**Changes in employment laws:** The risk associated with changes in employment laws is the potential for non-compliance, which could result in fines, legal action, or damage to the job portal's reputation.

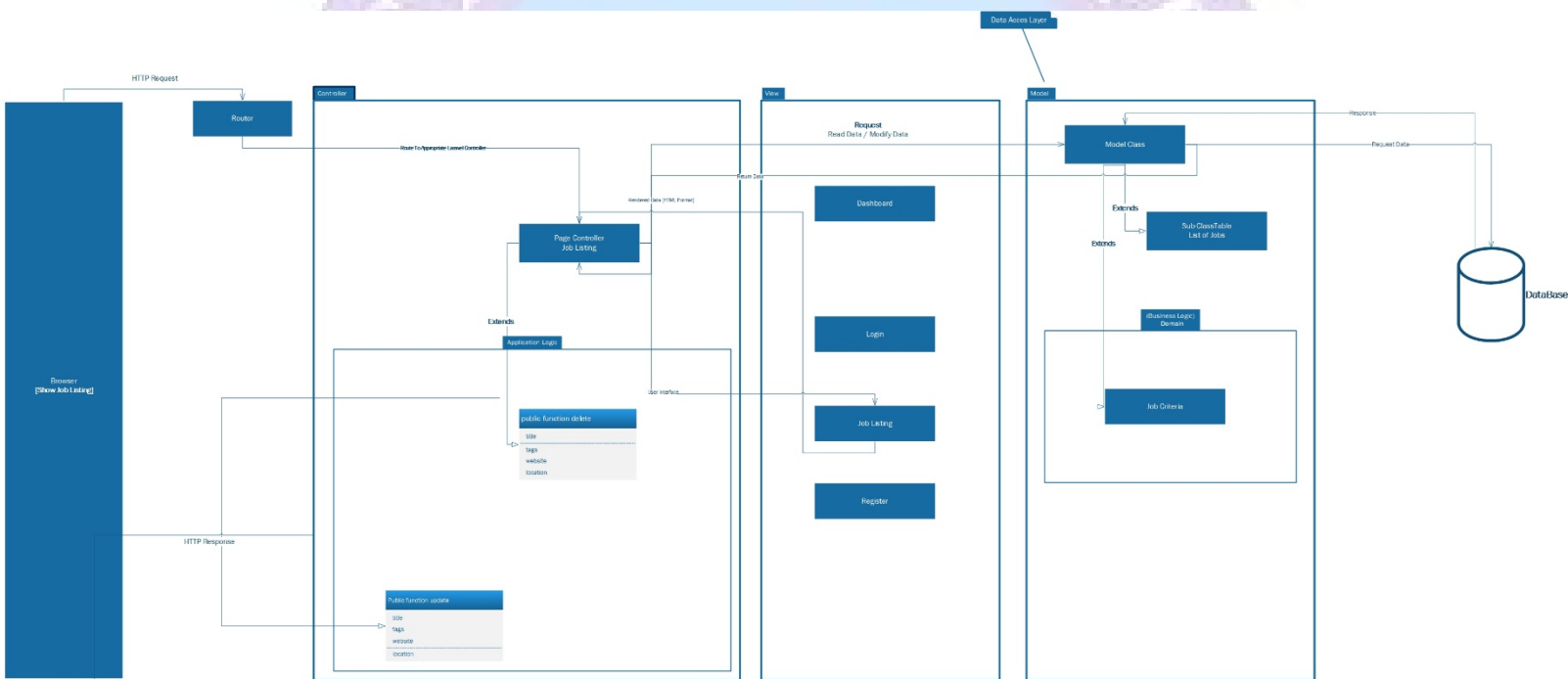


By considering the risks associated with each scenario, the job portal can develop strategies to mitigate these risks and protect itself against negative outcomes.

**Missing:** Architecture Diagram & Design Patterns

## 5. Architecture Diagram & Design Pattern

### 5.1. Architecture Diagram & Working



### Working Steps of Architecture

**1) Request:** First, a user inputs that they want a list of jobs through a web browser or a mobile application.

- 2) Then the request will be sent to the server and from the router will find matching route to the appropriate controller “Job Listing”.
- 3) Controller will ask the model to find the list of jobs, model will return the requested data to the controller.
- 4) **Request:** Controller will ask the View to present the list of jobs in html.
- 5) **Response:** Return the rendered list of data in html that is requested to the Controller.

Lastly, the Controller will take that Html and return it back to the user and display it on browser, thus getting the list of Jobs as the output.

## 5.2. Design Pattern

The Model-View-Controller (MVC) design pattern is a software architectural pattern that separates the representation of information from the user's interaction with it.

- **Model** consists of data and the business logic that operates on that data.
- **View** is the user interface that displays the data to the user and allows them to interact with it.
- **Controller** is the mediator between the model and the view, handling user input and updating the view based on the model's state.

- ❖ Overall, the MVC design pattern helps to structure the codebase of a job portal built in **Laravel** by separating the data, business logic, and user interface into distinct components. This can make the code easier to understand, maintain, and extend over time.

**END**

*“Architecture Document Completed  
(JAZAK-ALLAH)”*

*“Thank You So Much,  
Respectful Ma’am, Samra Siddiqui”*