PROJECT REPORT

Advanced Programming Concepts (23CS007)

# BACHELOR OF ENGINEERING

### in

**COMPUTER SCIENCE AND ENGINEERING**

## Submitted by: Supervised By:

Aryan Singhal (2310990133) Mr. Venkatesh K

Aakriti Kheterpal (2310990087) Technical Trainer Anishka Srivastava (2310990119)

Ritik Goyal (2310991110)



**CHITKARA UNIVERSITY**

**CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB,-140401 (INDIA)**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Practical No.** | **Practical Name** | **Page No.** |
|  | **Declaration** | **3** |
|  | **Acknowledgement** | **4** |
| **1.** | **Introduction** | **5 - 10** |
| **2.** | **Problem Statement** | **11 – 34** |
| **3.** | **Results** | **35 – 38** |
| **4.** | **Future Scope** | **39 – 40** |
| **5.** | **Appendices** | **41 – 45** |
| **6.** | **References** | **46** |

**DECLARATION**

We hereby declare that the project work titled, “**Job Recruitment and Application Management System**” submitted as part of Bachelor’s degree in CSE, at Chitkara University, Punjab, is an authentic record of our own work carried out under the supervision of **Venkatesh k.**

**Name :** Aryan Singhal Aakriti kheterpal Anishka Srivastava

Ritik Goyal

### Signature(s):

**ACKNOWLEDGEMENT**

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of my study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as part of the curriculum.

We are thankful to **Mr. Venkatesh K** for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence, and blessings.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to-day experience and received lots of suggestions that improve our quality of work.

# INTRODUCTION TO WEB DESIGN FRAMEWORKS AND DEVELOPMENT

### Background

The evolution of web development has revolutionized the way people interact with technology. In the early days of the internet, static web pages built with HTML were the norm, providing little more than basic information. Over time, the demand for more dynamic, interactive, and user-friendly systems led to the emergence of advanced web technologies and frameworks.

Modern applications are no longer limited to simple information sharing. They provide complete solutions to complex problems such as e-commerce, social networking, online learning, and digital recruitment. Among these, job application and recruitment portals have gained significant importance as they provide a digital bridge between employers and job seekers. Portals like **LinkedIn, Indeed, and Naukri** are popular today, but smaller customized solutions also play a major role in academic and organizational contexts.

In this project, we have developed a **Job Application Portal** that makes use of advanced frameworks such as **Spring Boot** on the backend and **JSP with CSS** on the frontend. The system allows users to sign up, log in, and view job postings, while administrators can add, edit, and delete job opportunities.

### Need for Web Design Frameworks

Developing a full-stack web application without a structured framework can be challenging due to issues such as scalability, maintainability, and security. Frameworks such as **Spring Boot** provide a ready- made structure for application development, enabling developers to focus on solving business problems rather than reinventing the wheel.

In the context of our project:

* + - **Spring Boot** manages backend logic such as user authentication and job posting workflows.
    - **JSP (JavaServer Pages)** handles the frontend presentation layer where users and admins interact with the system.
    - **Spring Data JPA** simplifies database interactions with MySQL by automatically handling queries and transactions.

Using frameworks ensures:

* + - **Consistency** – All modules follow the same coding patterns.
    - **Security** – Built-in authentication and role-based access control.
    - **Efficiency** – Faster development cycle due to reusable components.
    - **Scalability** – The system can be easily extended to add more features in the future.

### Importance of Web Applications in Recruitment

Traditional recruitment processes are time-consuming and involve significant paperwork. Employers need to advertise jobs through newspapers or offline channels, and job seekers must manually apply, often with physical resumes. This leads to inefficiency, delays, and limited reach.

Digital recruitment portals overcome these limitations by:

1. **Wider Reach:** A job posting becomes visible to multiple applicants instantly.
2. **Efficiency:** Employers can add, update, or remove postings in real-time.
3. **Accessibility:** Applicants can access opportunities anytime, anywhere.
4. **Automation:** Reduced manual intervention through databases and dynamic content rendering.

Our project replicates these advantages in a simplified manner, demonstrating the concepts of modern recruitment portals in an academic setting.

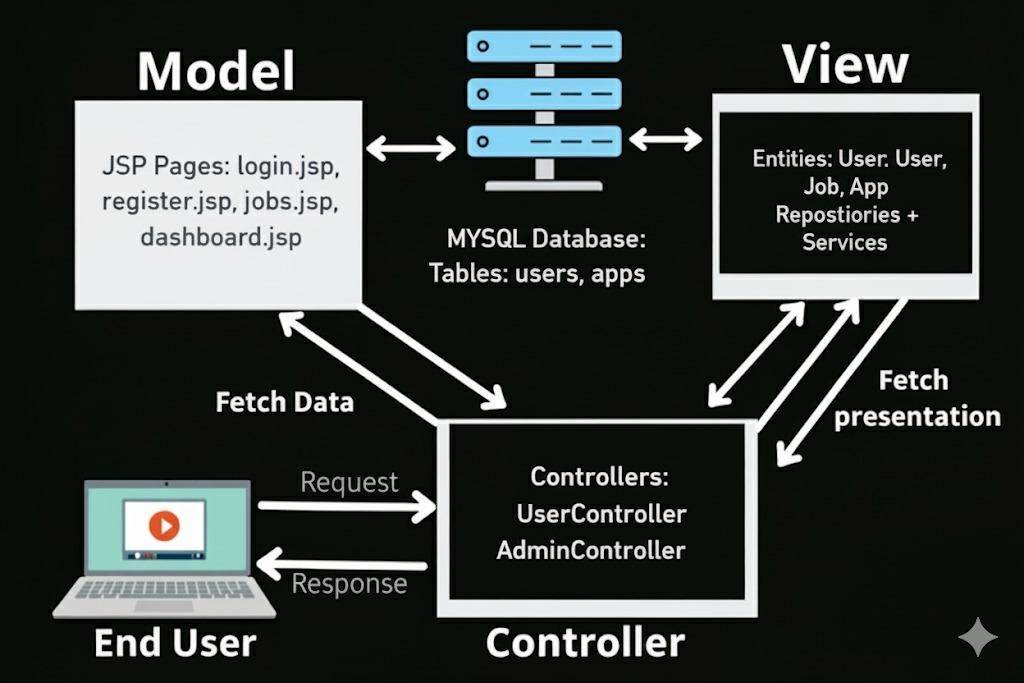
### Project Overview – Job Application Portal

The **Job Application Portal** developed as part of this project is a full-stack Java web application. It is designed to:

* + - Allow users to **sign up, log in, and browse jobs**.
    - Provide admins with the ability to **add, edit, and delete job posts**.
    - Implement **role-based access control**, where normal users and admins have different privileges.
    - Store all data in a **MySQL database**.

The system architecture follows the **Model-View-Controller (MVC)** pattern:

* + - **Model:** Represents database entities (User, JobPost).
    - **View:** JSP pages styled with CSS for user interaction.
    - **Controller:** Handles requests and responses, linking the frontend with backend services.



### Advantages of Our Approach

The implementation of the Job Application Portal using modern frameworks provides several key benefits that make the system secure, user-friendly, and future-ready.

### Security

The system includes password-protected login and session management to ensure that only authorized users can access the portal. Role-based access adds another layer of security, giving administrators exclusive rights to manage job postings while restricting normal users to viewing opportunities.

### User-Friendliness

The portal uses simple and intuitive JSP pages, making navigation easy even for non-technical users. Clear menus, straightforward login and signup forms, and well-structured job listings enhance the overall experience for both job seekers and administrators.

### Flexibility

Built on Spring Boot and MVC architecture, the system is modular in design. This makes it easy to add new features in the future, such as resume upload, search filters, or applicant notifications, without disrupting the existing structure.

### Maintainability

The separation of presentation, business logic, and data handling through MVC ensures that changes can be made to one part of the system without affecting others. For example, user interface updates can be applied without altering backend code.

### Scalability

The application is designed to handle multiple users and job postings simultaneously. Its layered structure allows for smooth scaling, so the system can be extended to support a larger user base and additional functionalities if required.

### Real-World Applications and Relevance

The Job Application Portal developed in this project is not only a part of academic learning but also a practical demonstration of how modern recruitment systems function in the real world. Today, job portals have become an integral part of the hiring process, as they bridge the gap between employers and job seekers in an efficient and accessible manner. Large multinational corporations, universities, and independent hiring agencies rely on similar platforms to advertise vacancies, filter applications, and manage large volumes of candidates.

Our project reflects these industry practices on a smaller scale, using frameworks such as Spring Boot and JSP to create a simplified yet functional version of such portals. By working on this system, we not only gain hands-on experience in full-stack web application development but also showcase our ability to design solutions that can be adapted and scaled for practical use. This makes the project highly relevant, as it aligns closely with the digital transformation trends in recruitment and workforce management.

Some possible real-world extensions of this project include:

### Connecting Job Seekers with Companies

The portal can evolve into a direct communication channel between job seekers and recruiters. Instead of relying solely on third-party advertisements, companies can register on the platform, post jobs, and interact with applicants directly.

### Resume Management and Applicant Tracking

A professional extension of this portal could include features for applicants to upload resumes and for recruiters to track the status of applications. This would make the system comparable to Applicant Tracking Systems (ATS) used by many organizations.

### Recommendation Systems

Advanced portals often use algorithms to recommend jobs based on a candidate’s profile, past searches, and preferences. Our system could be enhanced with such intelligence to increase engagement and improve the chances of suitable matches between employers and applicants.

### Integration with Third-Party APIs

The portal could also connect with external job networks such as LinkedIn or Indeed through APIs, enabling users to access a larger database of opportunities without leaving the platform. This would greatly expand its usefulness in real-world scenarios.

### Chapter Summary

In this chapter, we explored the fundamental concepts of web design frameworks and their growing importance in shaping modern application development. Beginning with the evolution of web

technologies, we emphasized how the transition from static web pages to dynamic, framework-based applications has transformed the way digital platforms are built and used. This context helped establish the relevance of choosing advanced tools such as **Spring Boot** for backend development and **JSP with CSS** for the frontend.

We further discussed the necessity of frameworks in addressing common challenges like scalability, maintainability, and security, which are crucial for building reliable and user-friendly systems. To illustrate these ideas, the chapter highlighted the role of digital recruitment portals in bridging the gap between employers and job seekers. By examining real-world parallels with platforms like LinkedIn, Naukri, and Indeed, we demonstrated the value of such portals in today’s competitive environment.

The chapter also provided a detailed overview of our **Job Application Portal project**, explaining its architecture, core functionalities, and relevance in both academic and real-world contexts. We emphasized how the system is designed with modularity, role-based access, and database integration, ensuring it is both practical and extendable.

Overall, this chapter set the foundation for the rest of the report by presenting the motivation behind the project, the significance of using web frameworks, and the broader context of recruitment systems. The following chapters will build upon this base, focusing on the **problem statement, objectives, chosen tools, methodology, implementation details, and the outcomes** of the project.

# PROBLEM STATEMENT

### Abstract

Employment plays a crucial role in the economic and social well-being of individuals, families, and societies as a whole. In today’s competitive environment, the process of finding suitable job opportunities has become increasingly challenging for job seekers. Similarly, employers face continuous difficulties in identifying, screening, and hiring skilled candidates who match the requirements of their organizations. Despite the digital transformation in several domains, the field of recruitment in many cases still relies on traditional or partially manual processes, which are inefficient and fail to meet the demands of a fast- moving job market.

The background of employment challenges highlights that job seekers are often forced to browse through multiple job portals, social media platforms, and newspaper advertisements in search of opportunities. This fragmented nature of information leads to confusion, redundancy, and missed chances for employment. Additionally, small and medium-sized businesses that cannot afford premium recruitment services often struggle to connect with potential candidates, limiting their access to talent. On the other hand, employers must spend significant time and resources in manually sorting applications, shortlisting resumes, and scheduling interviews. These inefficiencies highlight the urgent need for a structured and automated solution.

The problems with manual job applications are even more evident in scenarios where organizations still rely on offline submissions. Candidates are required to physically visit offices, fill out application forms, or submit resumes in person. This process is not only time-consuming but also excludes candidates who are geographically distant from potential employers. Furthermore, manual systems are prone to errors such as misplaced documents, incomplete data, and duplication of applications. For employers, the lack of a centralized system makes it difficult to track applicants, manage job postings, and analyze recruitment data effectively. Such issues create unnecessary delays in the hiring cycle and negatively impact both candidates and recruiters.

In this context, automation is needed to address the inefficiencies and bridge the gap between job seekers and recruiters. An automated job portal provides a centralized, structured, and accessible platform where candidates can register, search for jobs, and apply with just a few clicks. Recruiters benefit by being able to post job openings instantly, edit or remove outdated listings, and view candidate applications in real time. Security and data integrity are ensured through the use of authentication mechanisms and relational databases. Role-based dashboards make the system even more efficient, allowing users, recruiters, and administrators to access functionalities tailored to their specific needs.

The expected outcome from the JobApp portal is to create a seamless, transparent, and efficient recruitment system. For job seekers, the platform reduces the time and effort needed to find and apply for jobs while increasing their exposure to opportunities that match their skills. For recruiters, the portal provides tools to manage vacancies, streamline the selection process, and reduce hiring costs. Administrators gain control over monitoring activities, ensuring fair usage and maintaining the integrity of the system. Ultimately, the project aims to modernize the hiring process by automating tasks, reducing redundancy, and improving the overall recruitment experience for all stakeholders.

In conclusion, the JobApp project is designed not only as a technological solution but also as a bridge to overcome fundamental challenges in the employment sector. By leveraging modern frameworks such as Spring Boot, JSP, and MySQL, the system integrates functionality, security, and usability into one platform. The adoption of such an automated portal is expected to result in faster hiring cycles, improved employer-employee connections, and enhanced accessibility for candidates, thereby contributing positively to the employment ecosystem .

# 2.2. TOOLS AND TECHNOLOGY

The JobApp Project is developed using a combination of modern tools and technologies that make the system robust, scalable, and user-friendly. Each technology was carefully chosen to support the overall goal of creating a reliable Job Portal Application.

### Programming Language – Java (Version 17)

-Java is the backbone of this project.

-It is object-oriented, platform-independent, and highly secure, making it suitable for enterprise-level web applications.

-Java provides inbuilt features like exception handling, collections, multithreading, and JDBC support, which make it effective for backend development.

### Use in JobApp:

* + Writing controllers, services, and entity classes.
  + Implementing business logic such as authentication, validation, and job search filters.

### Framework – Spring Boot (Version 3.2)

-Spring Boot is a Java-based framework that simplifies application development by reducing boilerplate code.

-It supports the Model–View–Controller (MVC) architecture.

-Comes with an embedded Tomcat server, eliminating manual deployment.

### Use in JobApp:

* + Manages Controllers (UserController, JobController, AdminController).
  + Provides dependency injection for better modularity.
  + Handles connection to MySQL using Spring Data JPA.

### Frontend Technology – JSP (Java Server Pages)

-JSP is used to design the View layer of the MVC architecture.

-It allows embedding Java code in HTML to dynamically fetch and display data.

### Use in JobApp:

* + login.jsp, register.jsp – Authentication pages.
  + jobs.jsp, home.jsp – Display job postings.
  + dashboard.jsp – Recruiter/Admin dashboard for job management.

### Database – MySQL

-MySQL is a relational database management system (RDBMS) used for structured storage of data.

-In JobApp, the database consists of two main tables:

### Table 1: user

-Stores details of all users (job seekers, recruiters, admins).

-Columns:

* + id (Primary Key)
  + name
  + email
  + password
  + role

### Table 2: jobPost

-Stores details of jobs posted by recruiters.

-Columns:

* + id (Primary Key)
  + title
  + description
  + location
  + company
  + postedBy (foreign key → user.id)

### Use in JobApp:

* + The user table ensures secure authentication and role-based access control.
  + The jobPost table allows recruiters to post, update, and delete jobs, while job seekers can view and apply.

### Build Automation Tool – Maven

-Maven is a dependency management and build automation tool.

-The project is structured as a Maven Project with a pom.xml file.

### Use in JobApp:

* + Manages dependencies like Spring Boot starter modules and MySQL connector.
  + Ensures project consistency across different systems.

### Integrated Development Environment (IDE) – IntelliJ IDEA / Eclipse

-Provides intelligent code suggestions, debugging, and Maven integration.

-Allows developers to run the project with the embedded Tomcat server.

### Use in JobApp:

* + Writing, testing, and debugging Java code.
  + Managing database configurations (application.properties).

### Web Server – Apache Tomcat (Embedded in Spring Boot)

-Handles HTTP requests and responses between the client (browser) and the application.

### Use in JobApp:

* + Runs the JSP pages.
  + Serves as the backbone for handling web requests.

### Version Control – GitHub (Optional)

-Provides version control for maintaining code history.

-Useful for collaboration and backup.

### Methodology

The methodology adopted for the JobApp project ensures a structured and systematic development approach, guaranteeing the creation of a robust, maintainable, and user-friendly application. The methodology covers the entire software development lifecycle (SDLC), including requirement gathering, design, implementation, testing, and deployment.

### Software Development Life Cycle (SDLC) Model

For the JobApp project, we adopted the Waterfall Model due to its sequential and well-defined stages, which is suitable for projects with clearly defined requirements. The stages followed are:

* + - * Requirement Analysis:

Detailed interactions with stakeholders (job seekers, recruiters, and system administrators) were conducted to gather both functional and non-functional requirements. Functional requirements include user registration, login, job posting, and job search. Non-functional requirements emphasize security, scalability, and performance.

* + - * System Design:

This stage focused on translating requirements into design specifications, including database schema, system architecture, and workflows. The MVC (Model-View-Controller) pattern was selected for modularity.

* + - * Implementation:

Controllers, services, entities, and JSP pages were developed according to the design specifications. Spring Boot was used for backend development, JSP for the frontend, and MySQL for the database.

* + - * Testing:

Unit, integration, and user acceptance testing ensured that the application functions as intended and meets stakeholder expectations.

* + - * Deployment and Maintenance:

The application was deployed on Spring Boot’s embedded Tomcat server. Database backups, log

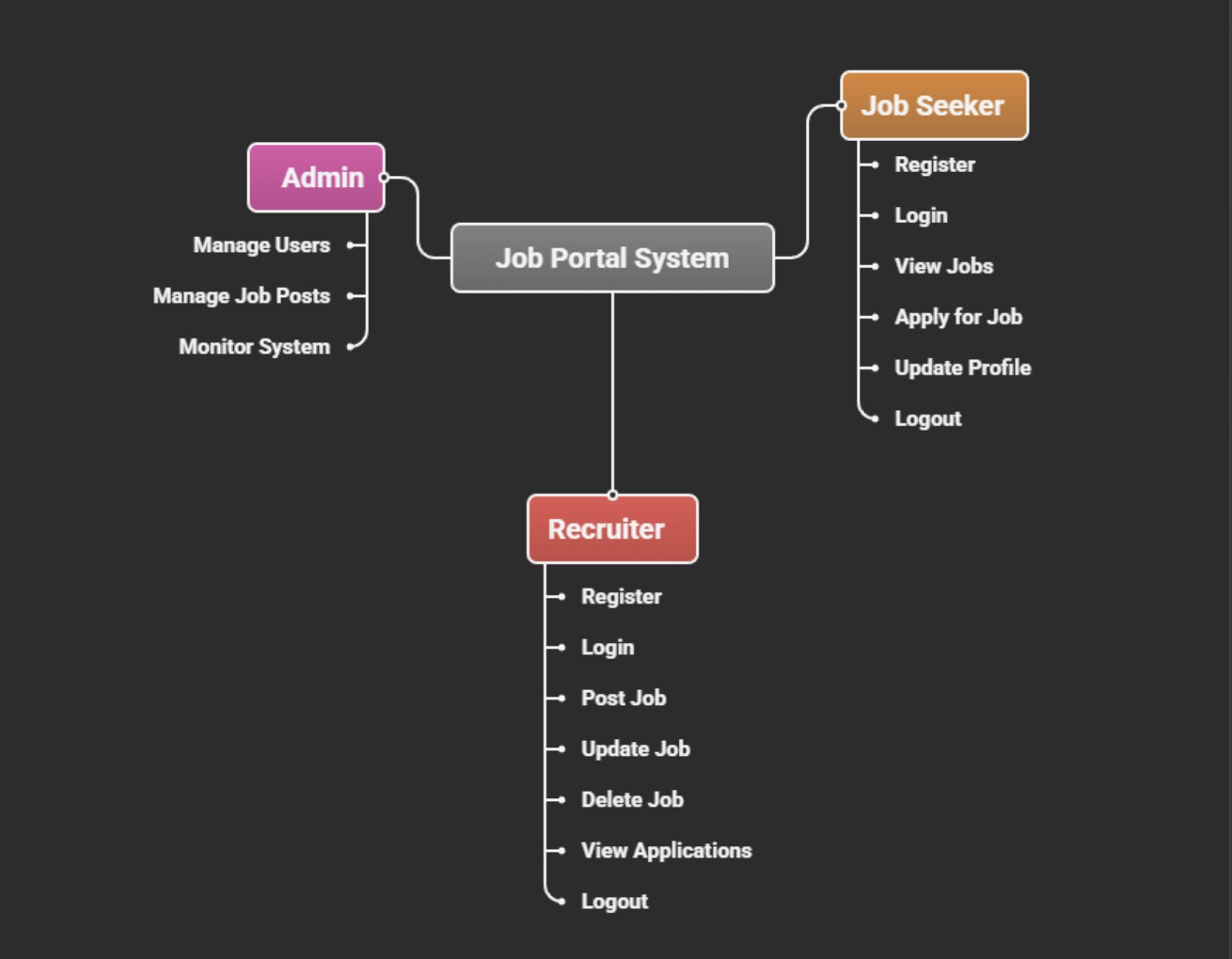
monitoring, and potential future enhancements were planned for long-term maintenance.

### System Architecture

The system architecture follows the MVC pattern:

* + - * **Model Layer:** Contains entities representing the database tables (User and JobPost) and JPA repositories for CRUD operations.
      * **View Layer:** JSP pages provide the user interface for login, registration, job listings, and dashboards.
      * **Controller Layer:** AuthController and JobController handle requests, interact with services, and render responses.
      * **Database Layer:** MySQL stores all user and job-related information securely.
      * The architecture ensures a clear separation of concerns, easy maintainability, and scalability.

### Use Case Diagram

****

**Key actors:**

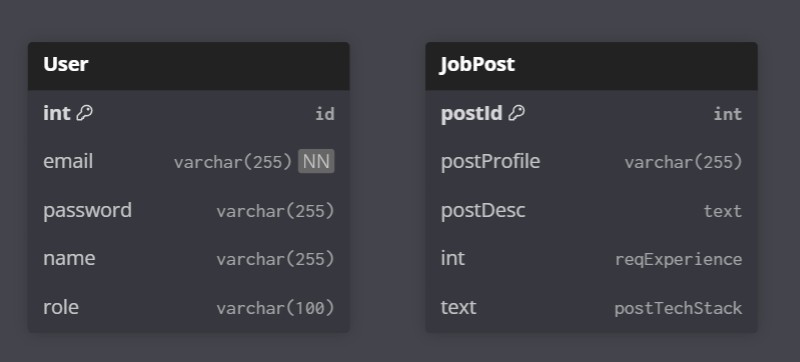
-Job Seeker, Recruiter, Administrator

### Use Cases:

* + - * Register/Login/Logout
      * View Job Listings
      * Post Job
      * Update/Delete Job Posting
      * Admin Monitoring (planned for future versions)
      * Each use case is connected to the relevant actor with clear associations.

### ER Diagram

ER Diagram



### Entities:

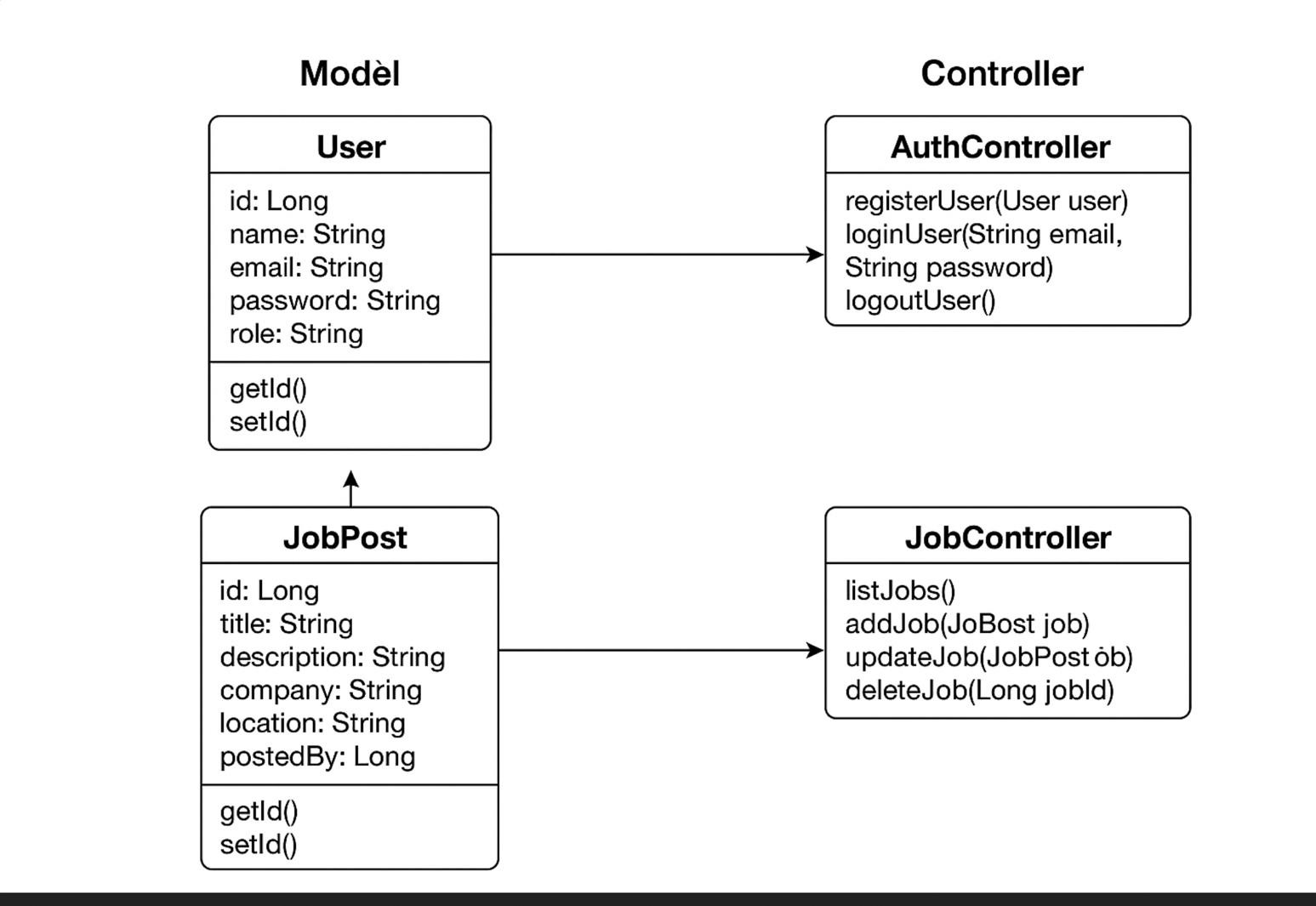
* + - * User (id, name, email, password, role)
      * JobPost (id, title, description, company, location, postedBy)

### Relationships:

* + - * One user (recruiter) can post multiple job listings;
      * users can view multiple jobs.

### Class Diagram

Class Diagram



### Key classes:

* + - * **User –** attributes: id, name, email, password, role
      * **JobPost** – attributes: id, title, description, company, location, postedBy
      * **AuthController** – handles registration, login, and logout
      * **JobController** – manages job creation, viewing, updating, and deletion
      * **Service Classes** – implement business logic
      * **Repository Interfaces –** manage database operations

### 2.3.7 Workflow Explanation

The JobApp workflow can be described step by step:

### User Registration and Login:

-A new user fills the registration form on register.jsp.

-The AuthController validates input, encrypts the password, and stores the user in the database.

-Login involves inputting credentials on login.jsp, where the controller authenticates via service and repository layers.

### Job Posting (Recruiter):

-The recruiter accesses dashboard.jsp to create a job.

-JobController validates data, creates a JobPost entity, and saves it to the database.

-Upon successful creation, the system confirms and updates the dashboard dynamically.

### Job Viewing (Job Seeker):

-Job Seekers access jobs.jsp, which triggers a request to JobController.

-The controller fetches all job postings from the database via the service layer.

-Listings are dynamically displayed using JSP and JSTL tags.

### Job Modification and Deletion:

-Recruiters can update or delete jobs from dashboard.jsp.

-The controller ensures only authorized users can modify their own posts.

1. Logout:

-Users can log out, which invalidates the session and redirects to login.jsp.

-Each step emphasizes security, validation, and modular design, ensuring the system is reliable - and user-friendly.

### Implementation

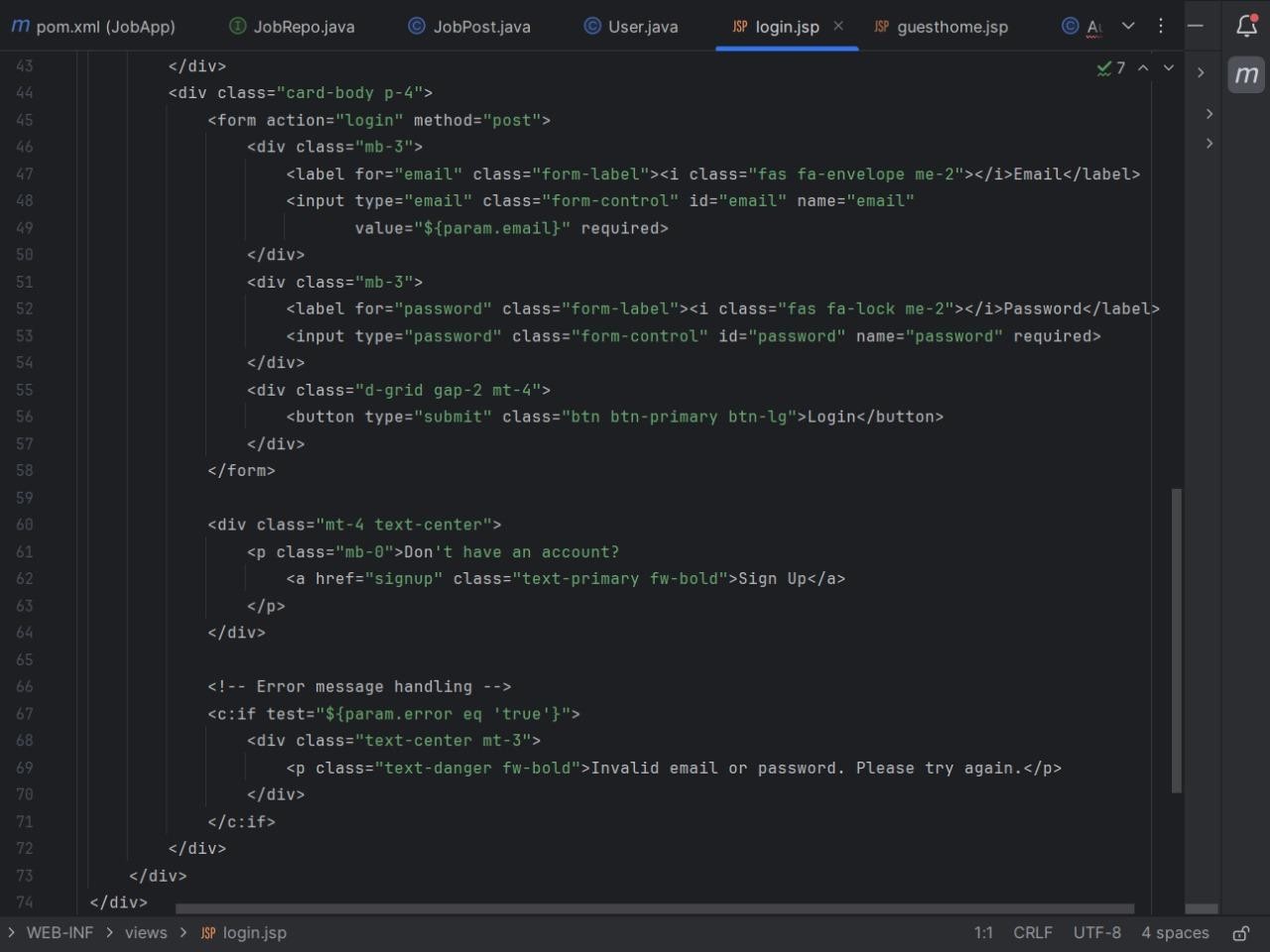
The implementation phase converts the methodology into a fully functional JobApp system. This section covers frontend development, backend development, database integration, and testing using Spring Boot, JSP, and MySQL within the MVC architecture.

### Frontend Implementation

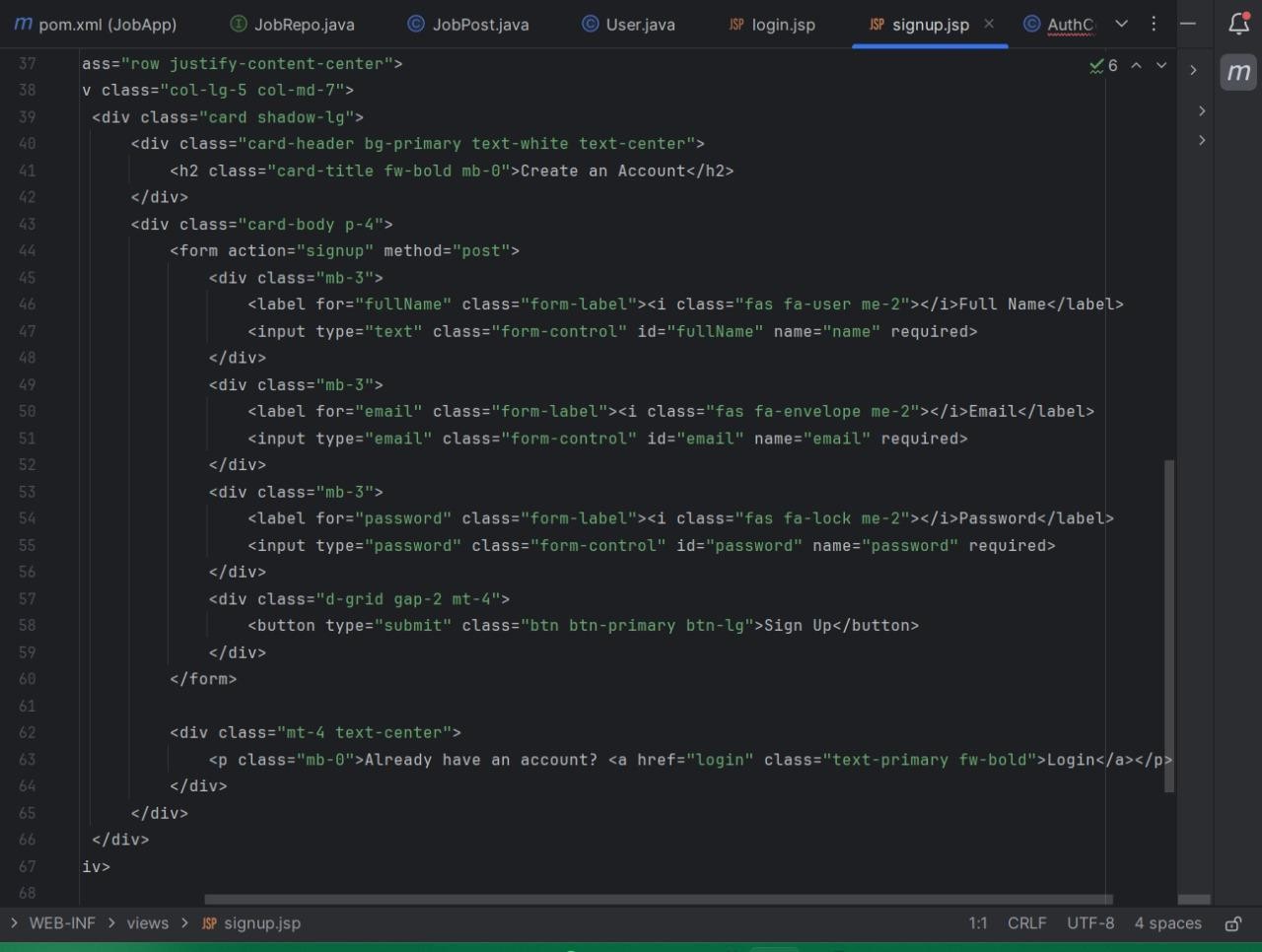
The frontend was developed using JSP, HTML, and CSS, providing interactive interfaces for both job seekers and administrators/recruiters.

**JSP Pages**

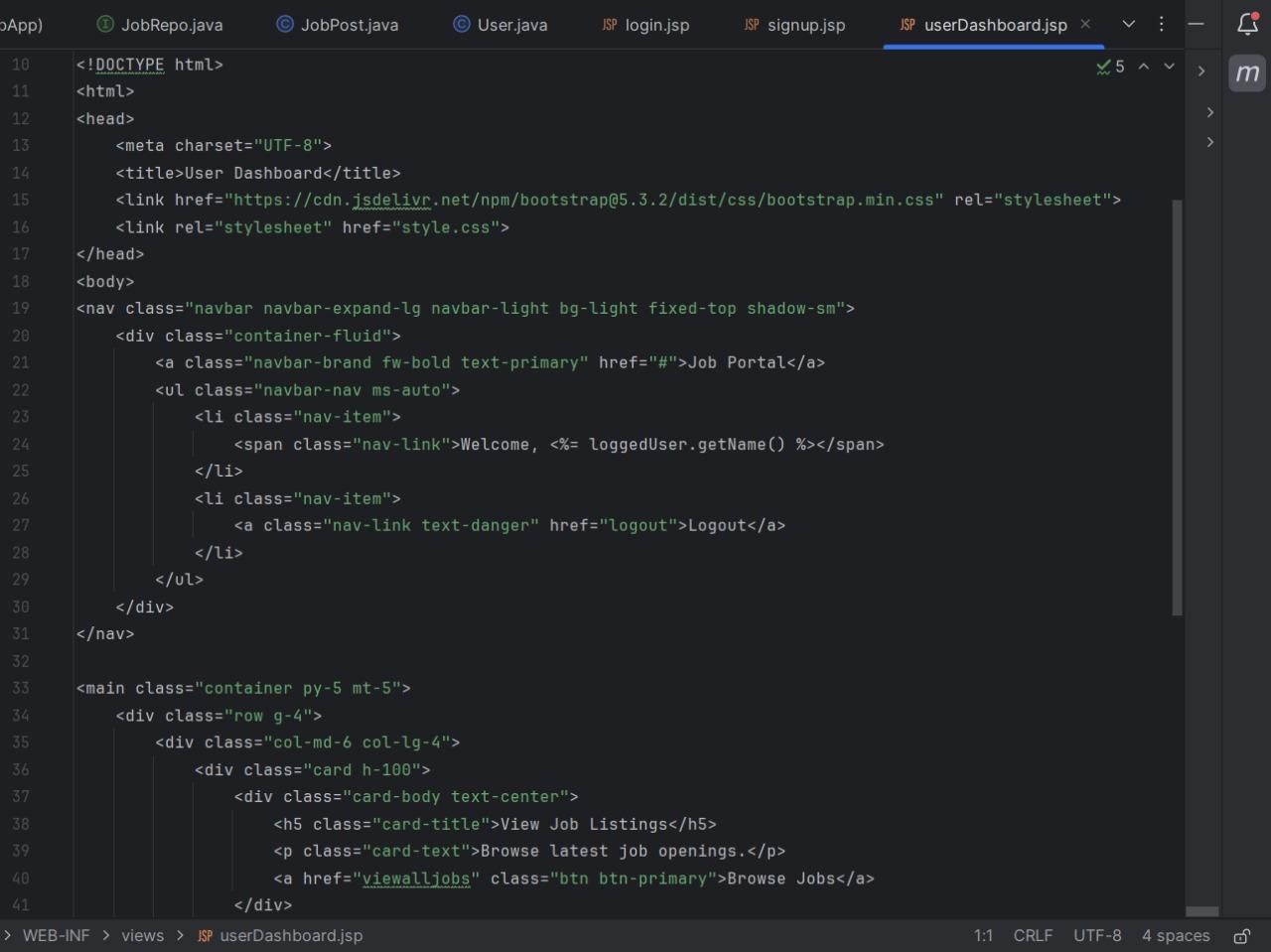
1. **login.jsp**

****

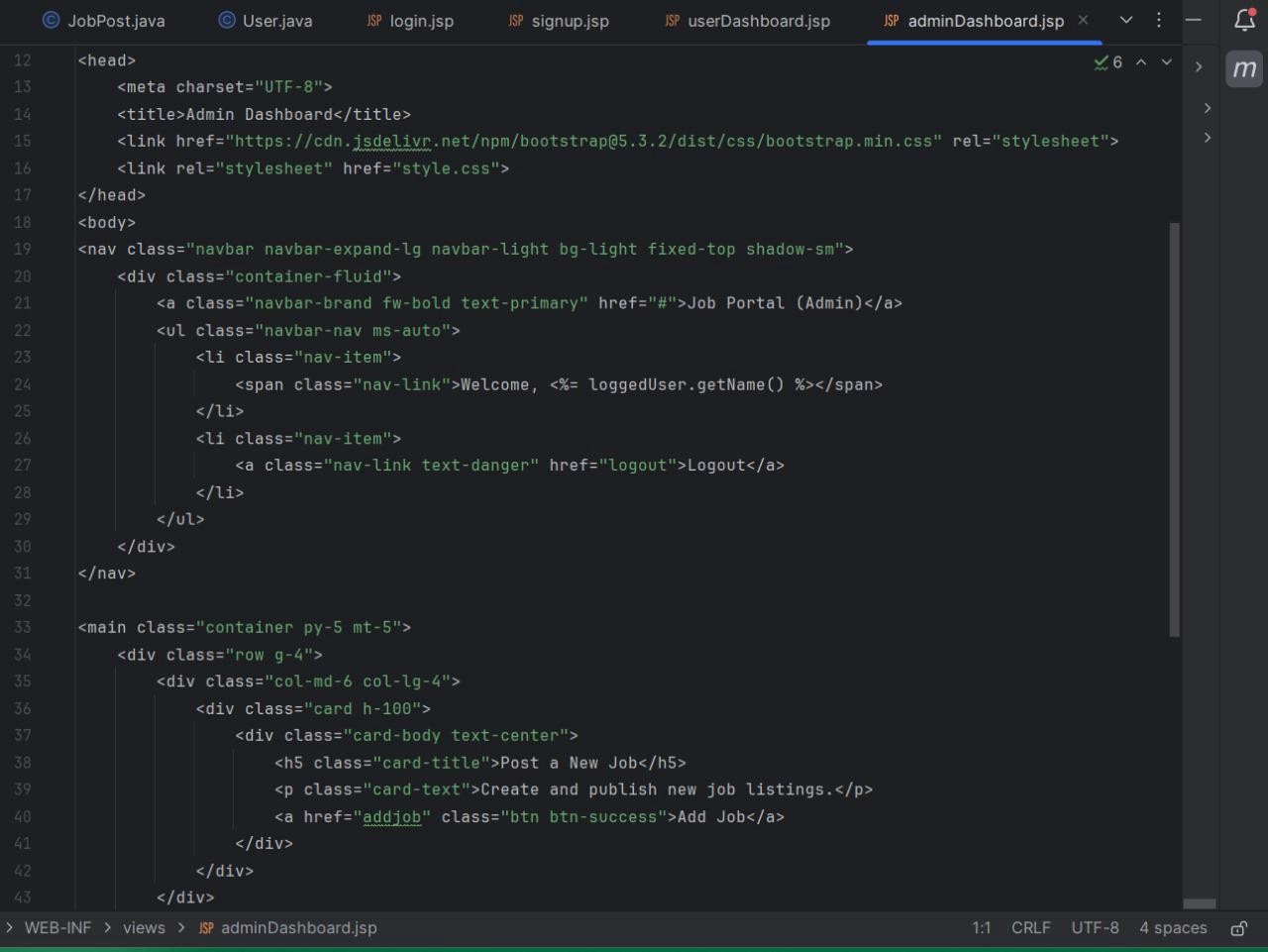
1. **signup.jsp**

****

1. **userDashboard.jsp**

****

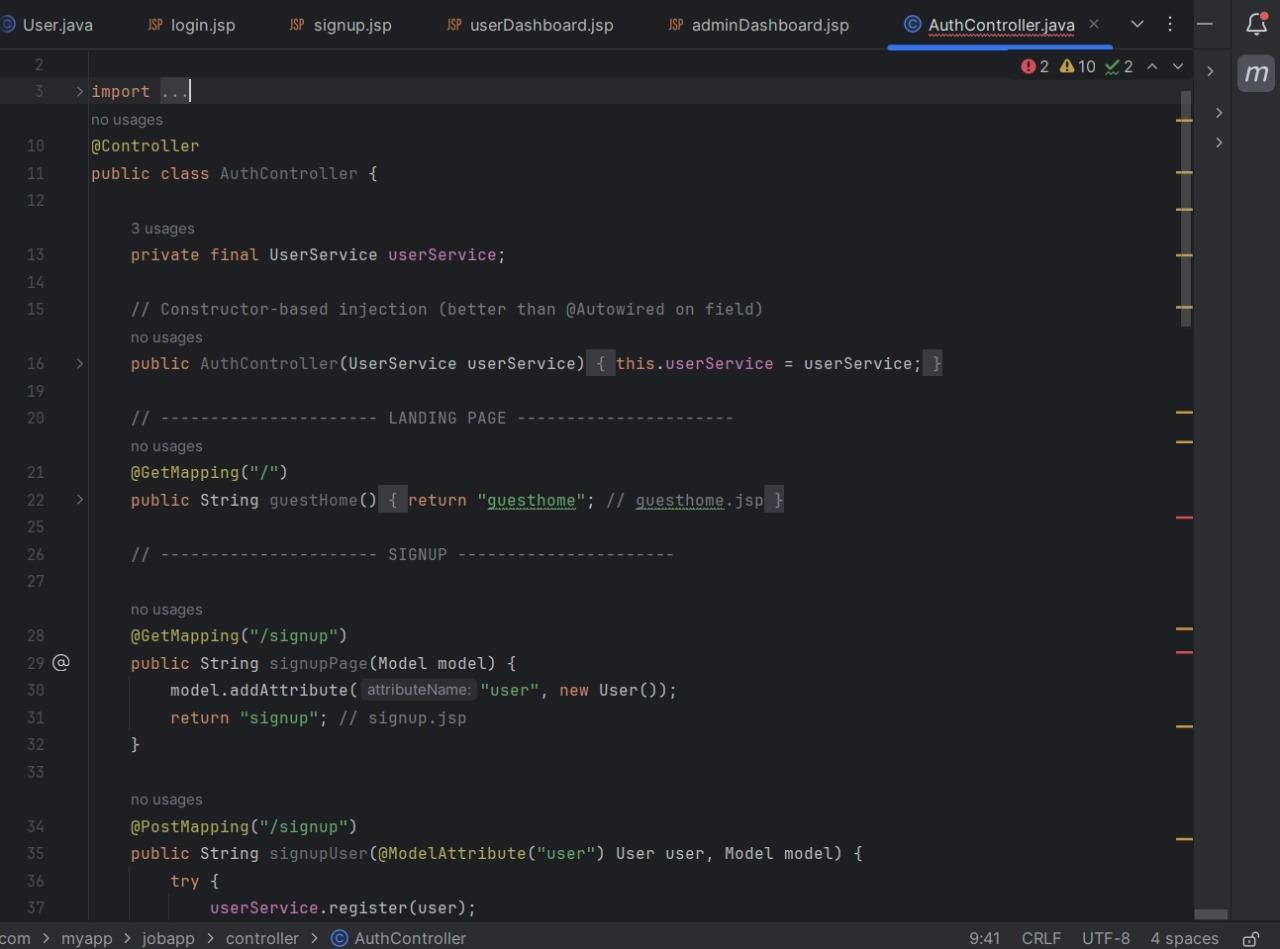
1. **adminDashboard.jsp**

****

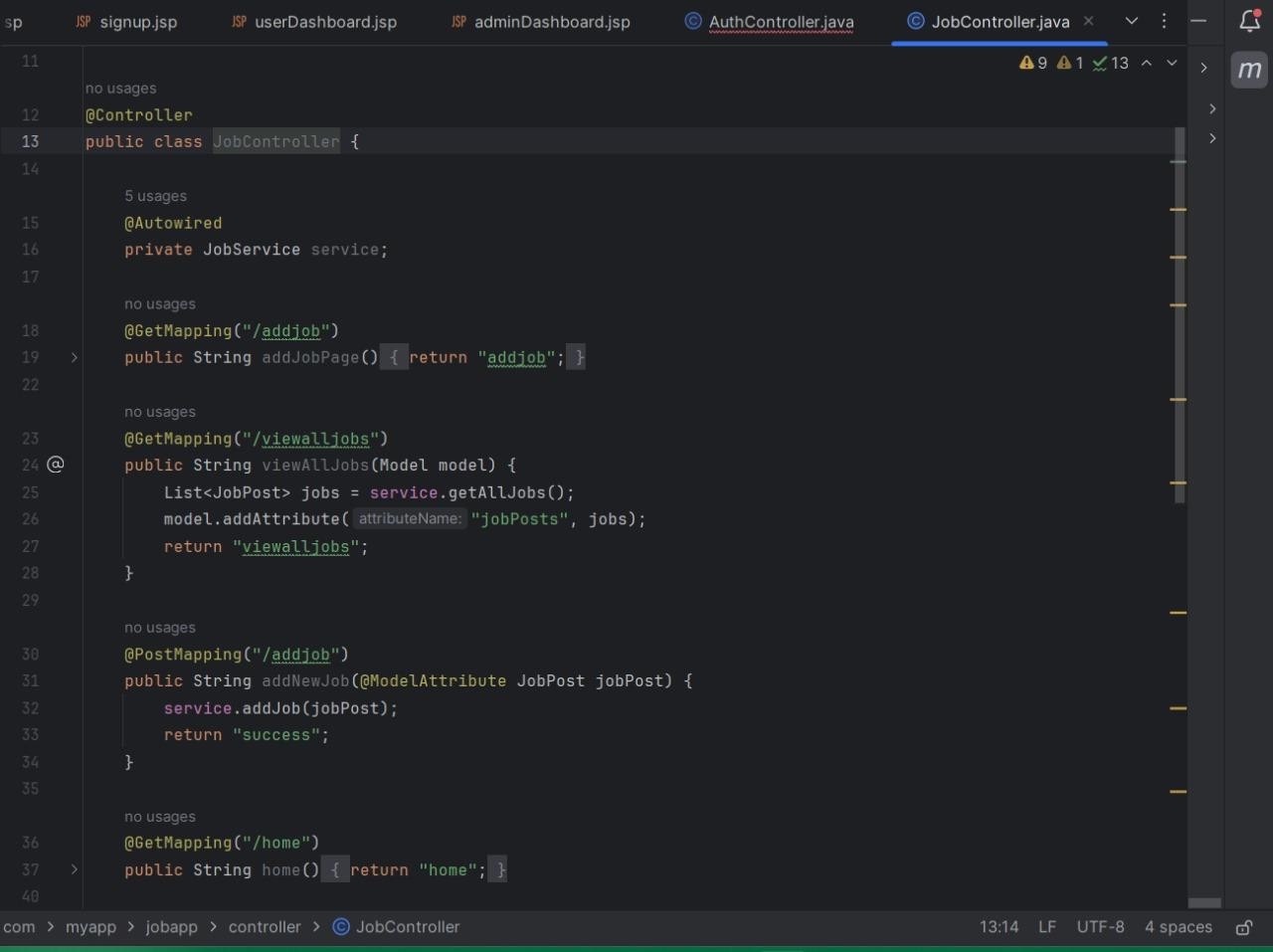
## Backend Implementation

The backend uses Spring Boot with Controllers, Services, and Repositories, following the MVC architecture.

1. **Controllers**
   1. **AuthController**

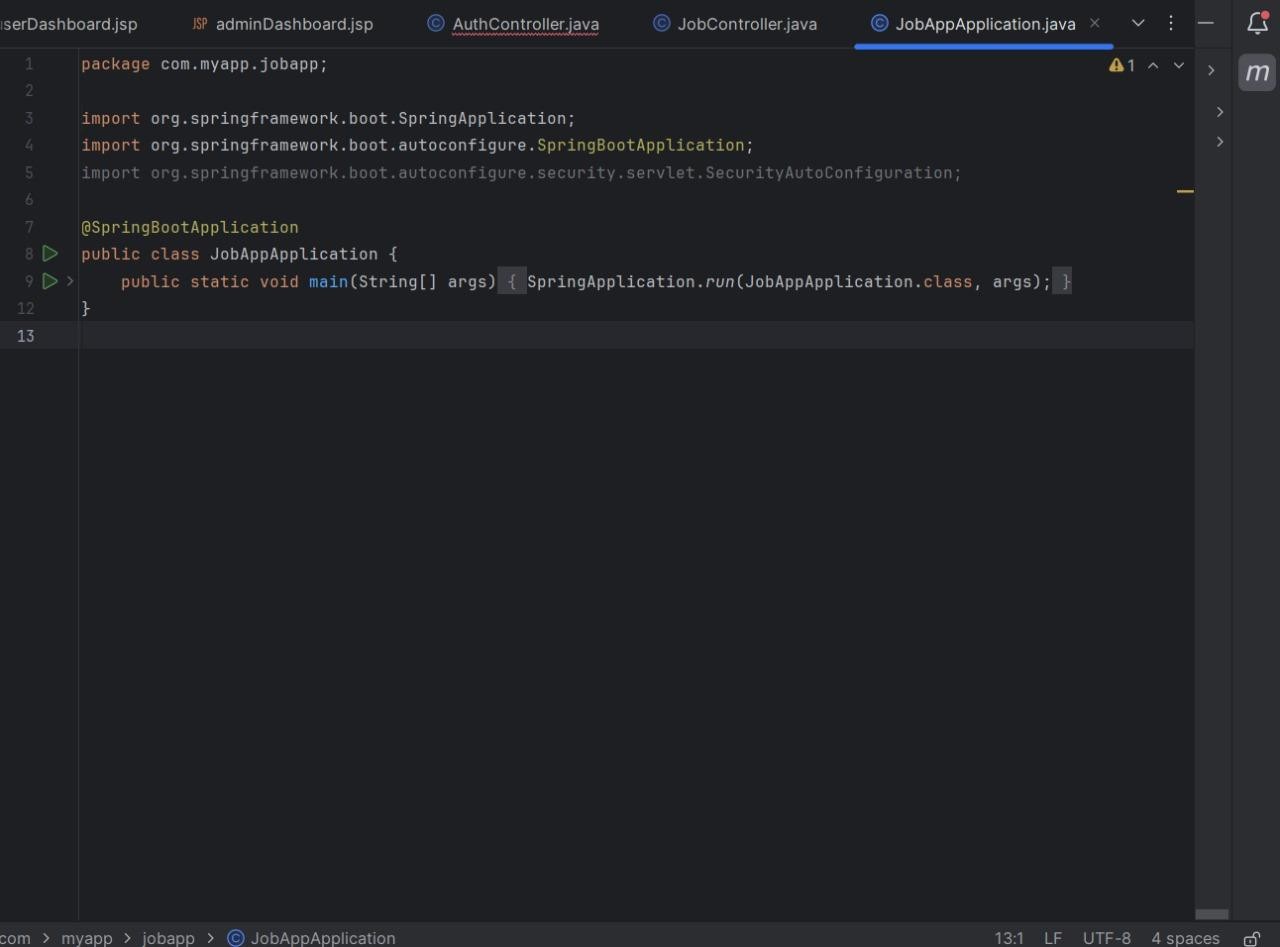
****

* 1. **JobController**

****

1. **Services**

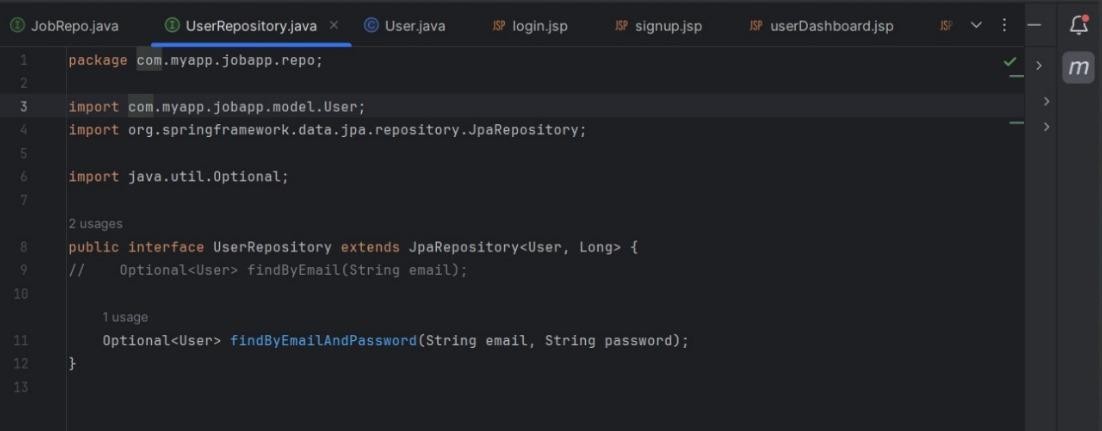
* **JonApplication.java**

****

1. **Repositories**
   1. **JobRepo.java**

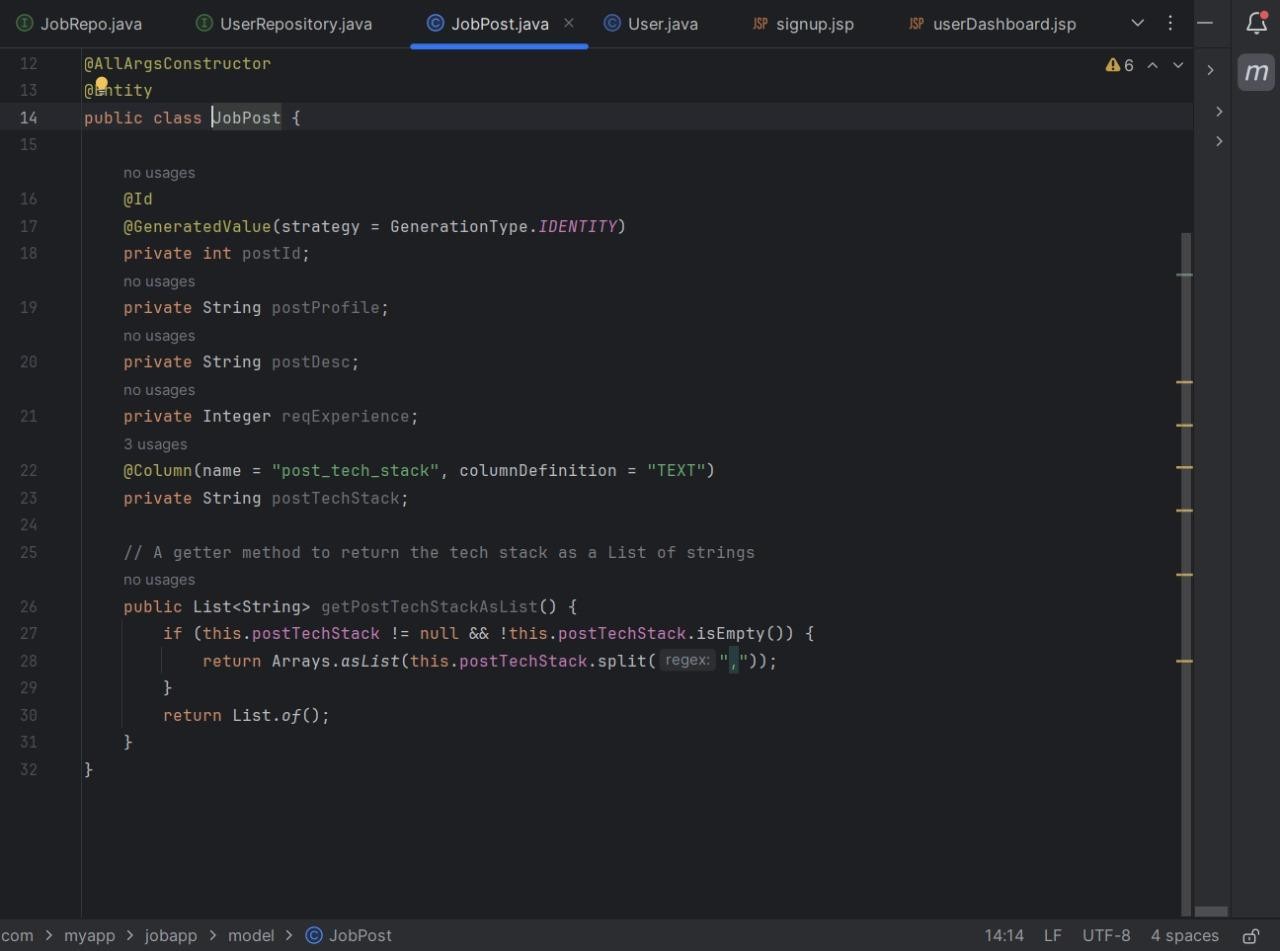
****

* 1. **UserRepo.java**

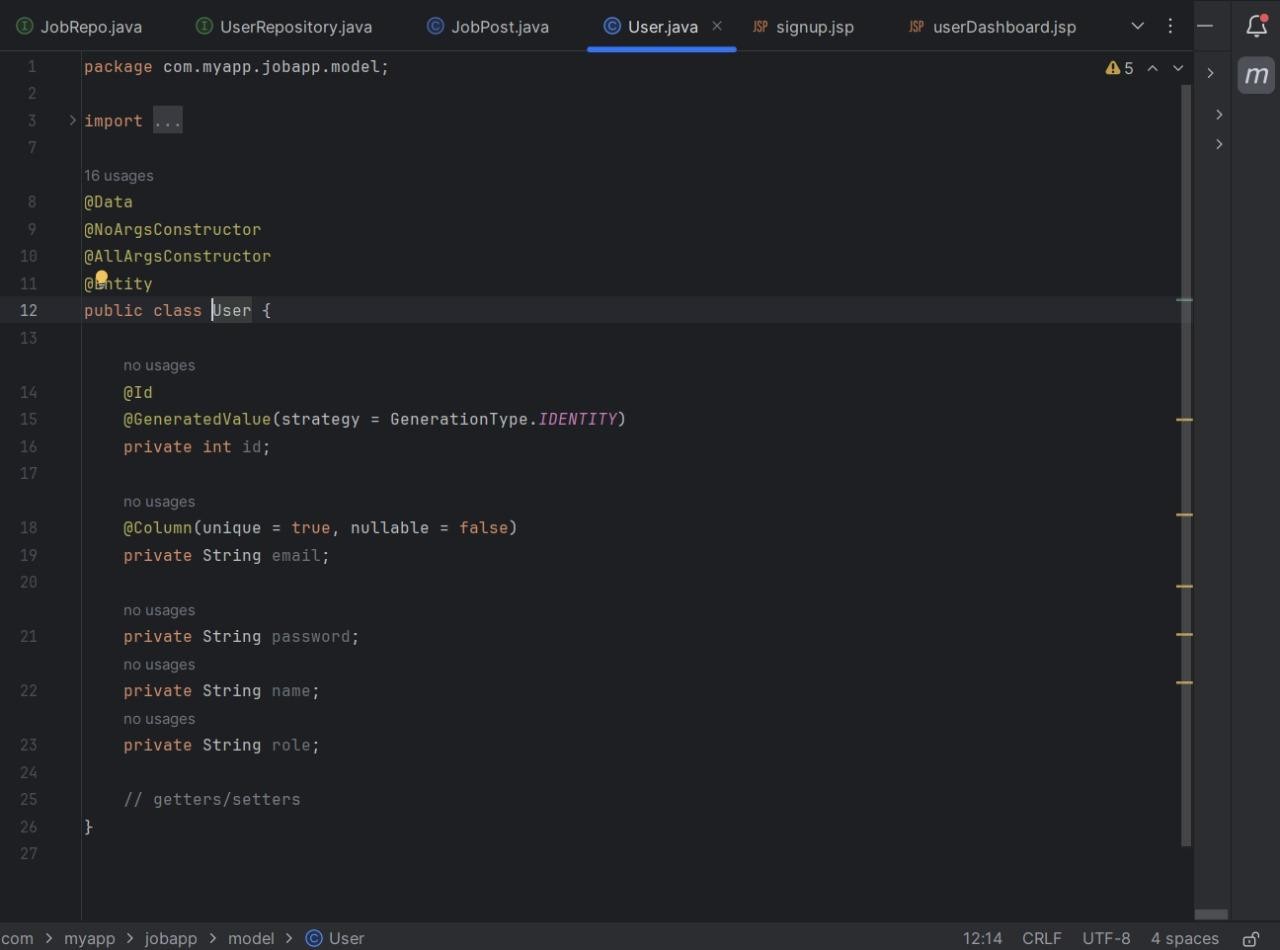
****

## Database Implementation

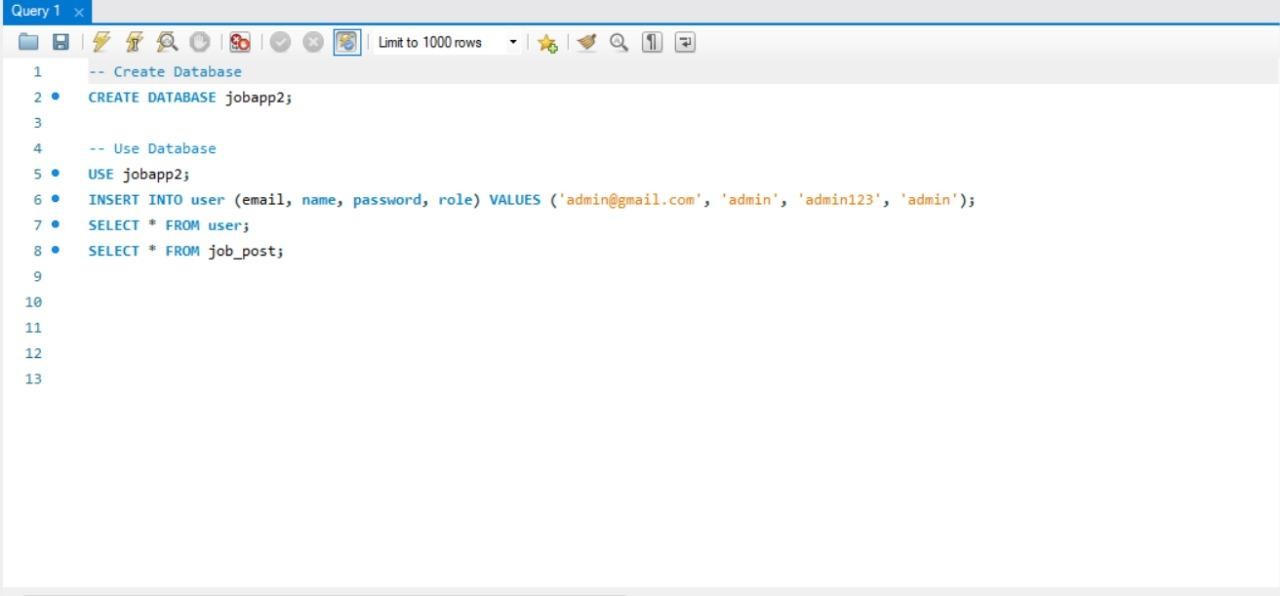
1. **Entity Classes (Models)**
   1. **JobPost**

****

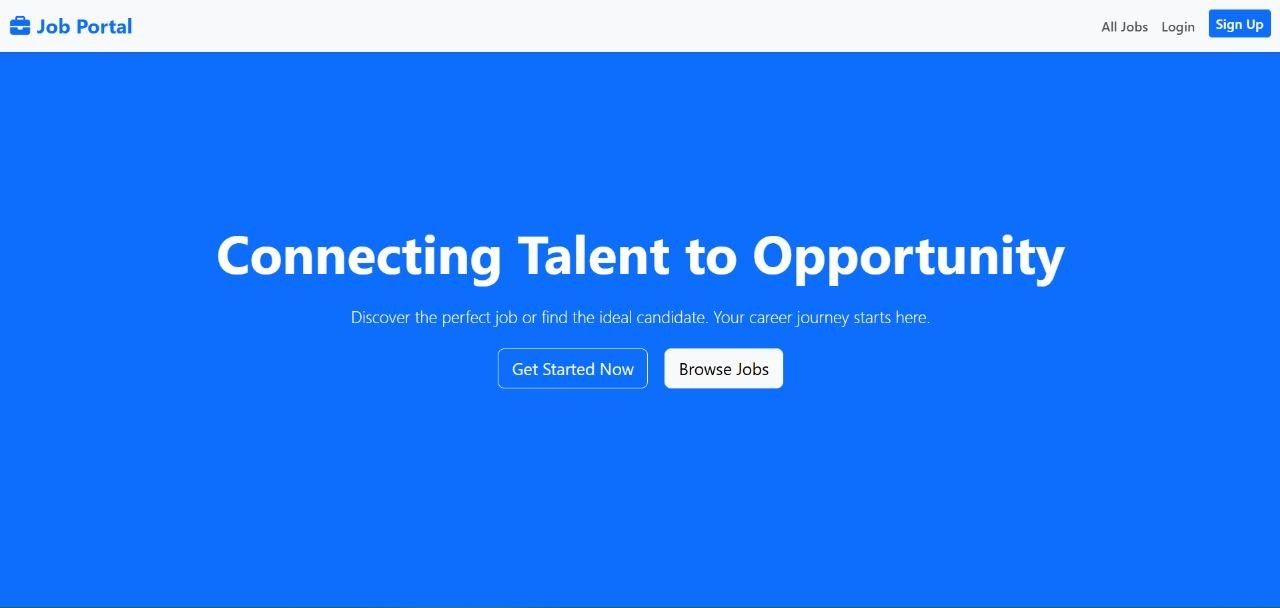
* 1. **User**

****

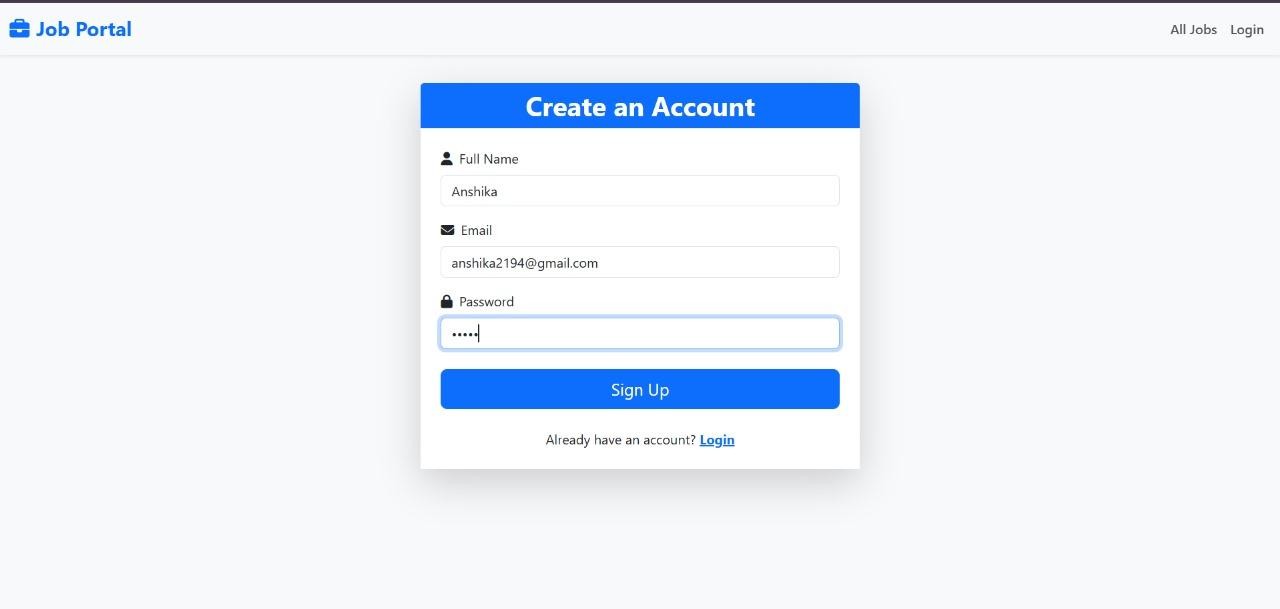
1. **My SQL workbench**

****

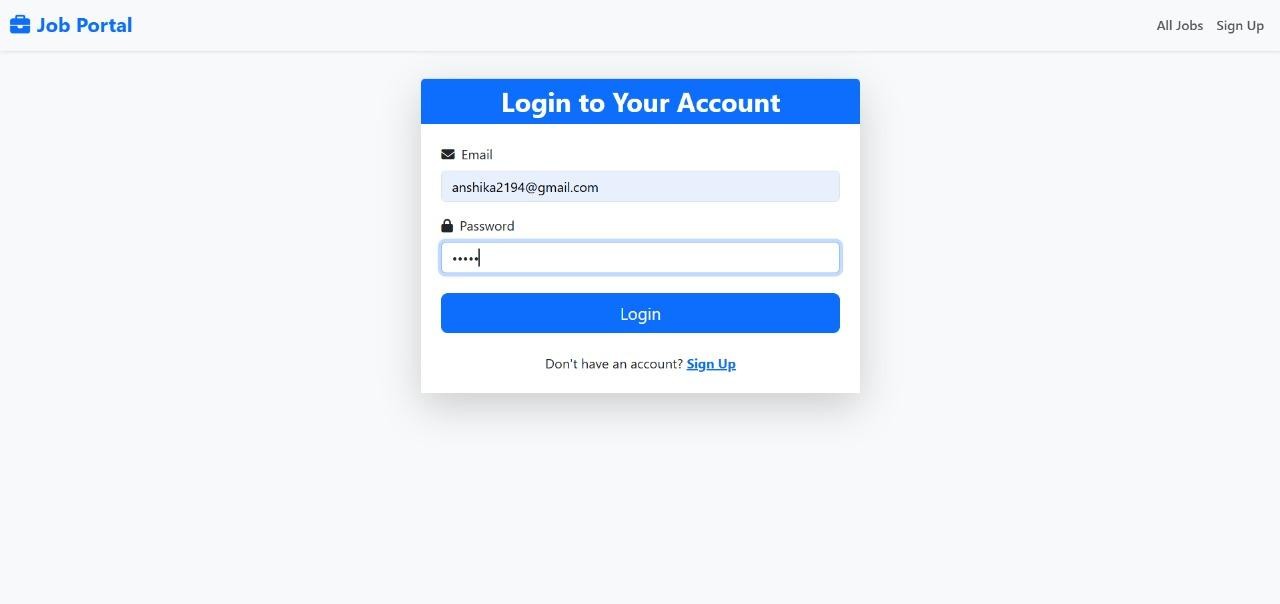
1. **Results**
   1. **Guest Dashboard**

****

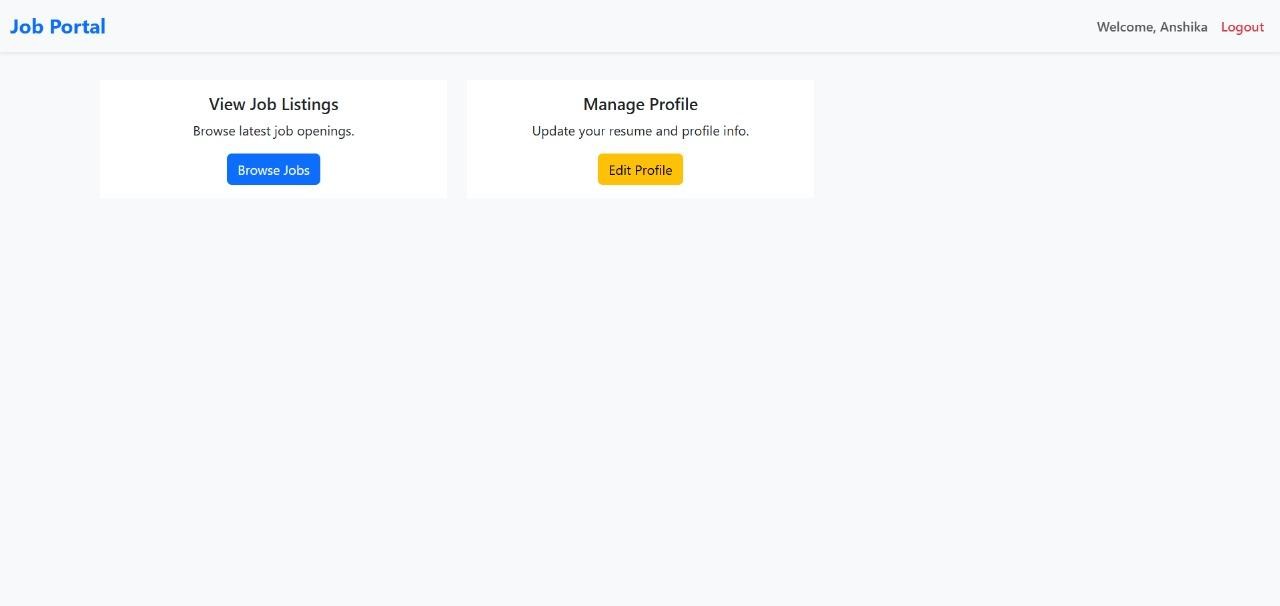
* 1. **User Registration**

****

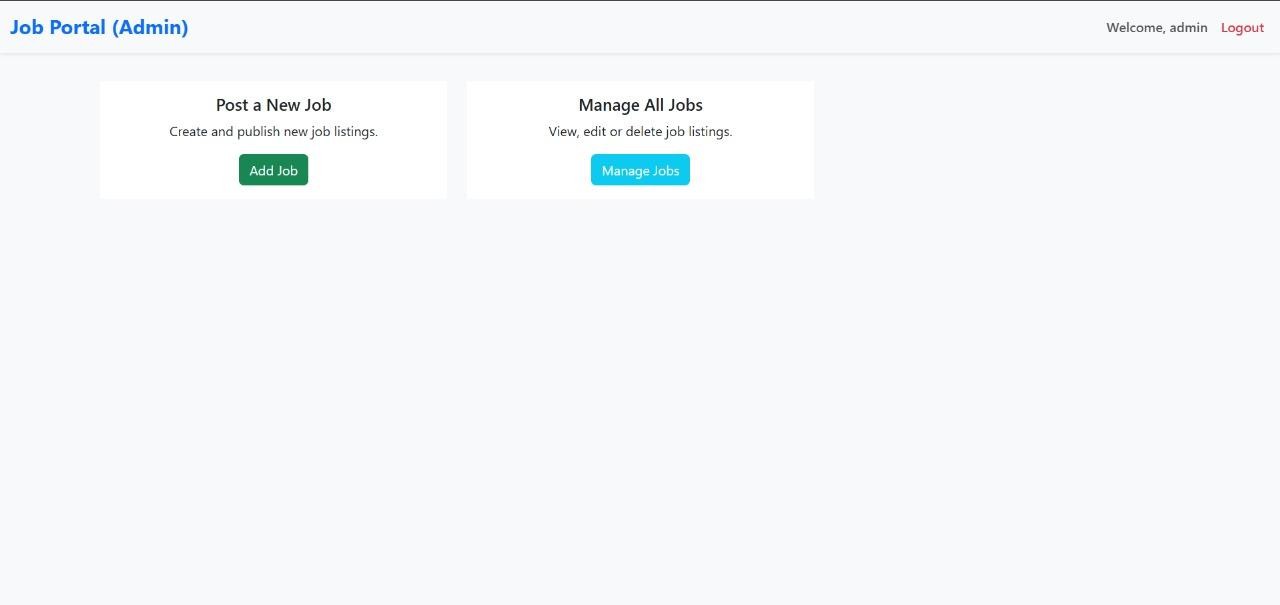
* 1. **User Login**

****

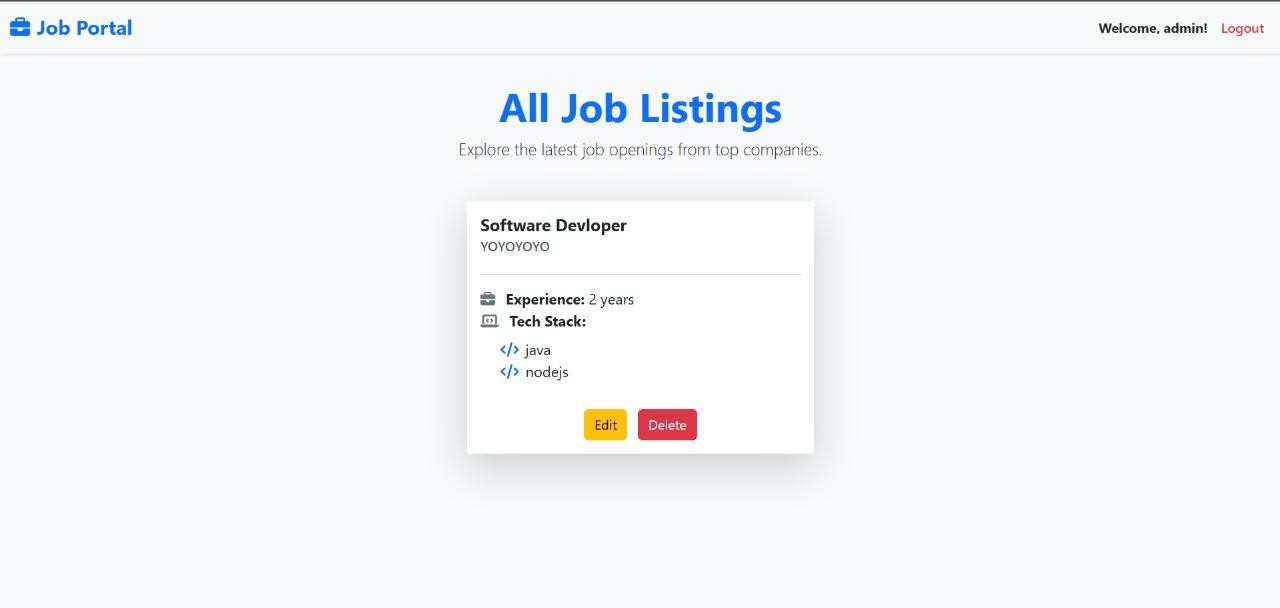
* 1. **User Dashboard**

****

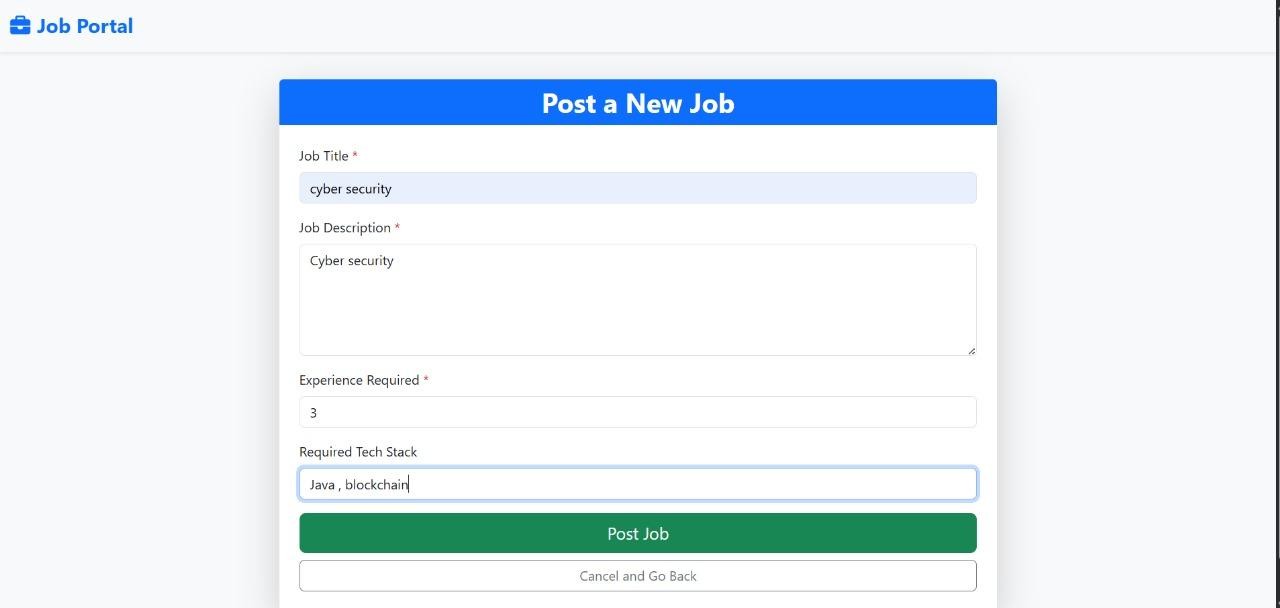
* 1. **Admin Dashboard**

****

* 1. **Job Listing**

****

* 1. **Job Posting**

****

## Future scope

### Analytics and Reporting

* + Provide dashboards and reports to track user activity, job postings, and application statistics.
  + Generate insights for recruiters on popular job types, application trends, and user engagement.

### Job Application Tracking

* + Enable job seekers to track their applications in real-time.
  + Allow recruiters to view applicants, update application status, and provide feedback.
  + Integrate notifications for application status changes.

### Email and Notification System

* + Implement email notifications for:
  + Successful registration
  + New job postings matching user preferences
  + Application status updates
  + Add push notifications for mobile users in future mobile app versions.

### Advanced Search and Filtering

* + Introduce keyword-based search, location filters, and skill-based filters.
  + Enable users to save job preferences and receive personalized recommendations.
  + Add sorting options by date, salary, or relevance.

### Mobile and Responsive Platforms

* + Develop a mobile-friendly web interface for job seekers and recruiters.
  + Create a native mobile app (Android/iOS) to improve accessibility and user engagement.

### Integration with External Systems

* + Integrate with LinkedIn, Indeed, or other job portals for expanded listings.
  + Implement single sign-on (SSO) with Gmail, LinkedIn, or corporate accounts.

### AI and Machine Learning Enhancements

* + Use AI to match job seekers with suitable jobs based on resume, skills, and experience.
  + Provide automated resume parsing and scoring for recruiters.
  + Predict job trends and demand in specific sectors for better insights.

### Security and Performance Improvements

* + Implement two-factor authentication for enhanced login security.
  + Optimize database queries and server response times for large-scale usage.
  + Use encryption and secure communication protocols for sensitive data.

## Appendices

This section provides supplementary information that supports the understanding, usage, and deployment of the JobApp system. It acts as a mini manual for developers, testers, recruiters, and job seekers.

### Appendix A: System Requirements

To successfully run the JobApp project, the following hardware and software requirements must be met:

1. Hardware Requirements

-Minimum 4 GB RAM (8 GB recommended for smooth IDE usage)

-Processor: Intel i3 or above

-Storage: At least 2 GB free space

1. Software Requirements

-Operating System: Windows 10 / Linux (Ubuntu 20.04 or later)

-JDK: Java Development Kit 17 or higher

-IDE: IntelliJ IDEA / Eclipse IDE (latest version recommended)

-Build Tool: Apache Maven 3.8 or later

-Database: MySQL Server 8.0+

-Web Browser: Google Chrome or Mozilla Firefox

### Appendix B: Deployment Steps (Mini Manual)

1. Download/Clone the Project
2. Obtain the project files from the repository or shared folder.
3. Import into IDE
4. Open Eclipse or IntelliJ IDEA.
5. Import the project as a Maven Project.
6. Configure Database
7. Start MySQL Server.
8. Create a database named jobapp.
9. Update application.properties with:
   1. -Database URL
   2. -Username
   3. -Password
10. Build the Project
11. Run mvn clean install to resolve dependencies and build the project.
12. Run the Application

-Use the command: mvn spring-boot:run OR

* run the main class JobAppApplication.java directly from the IDE.
* Access the System
* Open browser and go to:
* <http://localhost:8080/>

### Appendix C: User Guide for Job Seekers

* Job seekers form a key user group of the system. Their interactions include registration, login, job search, and application.
* Registering a New Account
* Navigate to the registration page.
* Provide name, email, password, and select User role.
* Submit details; confirmation is displayed upon successful registration.
* Logging In
* Navigate to the login page.
* Enter registered email and password.
* Upon successful login, the user dashboard is displayed.
* Viewing Job Listings
* On the dashboard, users can see available job postings.
* Each job displays title, company, location, and description.
* Applying for Jobs (Future Enhancement)
* Currently, job seekers can view postings.
* In future versions, seekers will be able to submit resumes directly through the platform.

### Appendix D: User Guide for Recruiters

* Recruiters post and manage job opportunities. Their workflow is slightly different from job seekers.
* Registering as a Recruiter
* Go to the registration page.
* Provide name, email, password, and select Recruiter role.
* Logging In
* Enter recruiter credentials on the login page.
* Successful login redirects to the recruiter dashboard.
* Adding a Job Post
* On the dashboard, select Add Job.
* Enter title, description, company name, and location.
* Submit to save the job in the system.
* Managing Jobs
* Recruiters can view their own job postings.
* Options include updating descriptions or deleting expired jobs.

### Appendix E: Workflow Summary

* To illustrate the combined experience of job seekers and recruiters:
* Job seekers register, log in, and browse opportunities.
* Recruiters register, log in, and add job postings.
* The system ensures that seekers can only browse and recruiters can only post.
* Role-based access creates a clear separation of responsibilities.
* This workflow simplifies user interaction and mirrors real-world job portal functionality.

### Appendix F: Maintenance Guidelines

* Maintaining the JobApp system ensures long-term reliability.
* Database Backups: Schedule weekly backups of the MySQL database.
* Log Monitoring: Regularly check application logs for errors.
* Dependency Updates: Keep Maven dependencies updated for security patches.
* User Data Security: Use strong encryption algorithms for passwords.
* Server Monitoring: Ensure the application server has sufficient memory and CPU capacity.

## References

* *Spring Boot Tutorials* – https:/[/www.baeldung.com/spring-boot](http://www.baeldung.com/spring-boot)
* GeeksforGeeks, *Spring Boot and JSP Integration* – https:/[/www.geeksforgeeks.org/](http://www.geeksforgeeks.org/)
* W3Schools, *HTML & CSS Basics for JSP Styling* – https:/[/www.w3schools.com/](http://www.w3schools.com/)
* TutorialsPoint, *Java Database Connectivity (JDBC)* – <https://www.tutorialspoint.com/jdbc/index.html>
* Spring Boot Documentation – <https://spring.io/projects/spring-boot>
* Spring Data JPA Documentation – <https://spring.io/projects/spring-data-jpa>
* JSP (JavaServer Pages) Technology – Oracle Docs –

https://docs.oracle.com/javaee/7/tutorial/servlets.htm

* MySQL 8.0 Reference Manual – https://dev.mysql.com/doc/
* Apache Maven Project – https://maven.apache.org/