Tribhuvan University

Faculty of Humanities and Social Science

Himalaya College of Engineering

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project is prepared under my supervision by Binit Maharjan entitled "**Jop-Portal System**" in the partial fulfillment for the degree of Bachelor of Computer Application is recommended for the final evaluation.

Signature

Er. Himal Chand Thapa

Supervisor

Humanities and Social Science

Himalaya College of Engineering

Chyasal-9, Lalitpur

Tribhuvan University

Faculty of Humanities and Social Science

Himalaya College of Engineering

LETTER OFAPPROVAL

This is to certify that this project is prepared by Binit Maharjan entitled "Job-Portal System" in the partial fulfillment for the degree of Bachelor of Computer Application has been evaluated. In our opinion it is satisfactory in scope and quality as a project for the required degree.

Er. Himal Chand Thapa Supervisor Department of Computer application Himalaya College of Engineering Chyasal-9, Lalitpur	Er. Himal Chand Thapa Co-Ordinator Department of Computer application Himalaya College of Engineering Chyasal-9, Lalitpur
Signature of Internal Examiner	**Signature of External Examiner**

ABSTRACT

The Job Portal System is a platform designed to streamline recruitment, improve accessibility, and provide personalized career opportunities in the job market. Focused on connecting companies and job seekers in Nepal, it caters to fresh graduates, professionals, and recruiters by enabling users to seamlessly act as both candidates and employers. The system features secure registration, a hybrid job recommendation algorithm leveraging content-based filtering with cosine similarity, and advanced job search options based on user skills, location, and experience level. Employers can post and manage job listings, review applications, shortlist candidates, while administrators can monitor users, approve or reject employers, manage fraudulent listings, and oversee platform activities. JWT authentication is implemented for secure session handling, with email OTP verification for account recovery. Built on MongoDB, React, Express, and Node.js (MERN stack).

Keywords: Jobs, Recruiters, Candidates, Applications, Recommendation, Nepal

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our supervisor Er. Himal Chand Thapa who gave us the golden opportunity to do this wonderful project on the topic of Job-Portal System, which also helped us in doing a lot of research and we came to know about so many new tools and technologies. We would like to express our sincere gratitude to everyone who had directly and indirectly helped us to complete this project.

I am highly indebted to Himalaya College of Engineering for their guidance and constant supervision as well as for providing necessary information regarding the Project and support in the completion.

We would also like to express our heartfelt gratitude to all the classmates who were always inspiring us to continue with the project and reminded us of the competition every time. I am also grateful to all our professors for their guidance and support throughout the project.

In the end, we would also like to thank Tribhuvan University for giving us this opportunity via the course of Computer Application to help us understand the project ethics at this early stage and helped us to evaluate my knowledge and expand it a little more.

Yours sincerely,
Binit Maharjan

Table of Contents

LIST OF ABBREVIATIONS

CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheet
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
HTML	Hyper Text Markup Language
JS	Java Script
MERN	Mongo Express React Node
UI	User Interface

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background

A job portal is an online platform that connects job seekers with employers, making the hiring process faster, more efficient, and more accessible. These platforms allow job seekers to search and apply for jobs, create professional profiles, and upload resumes, while employers can post job openings, review applications, and hire suitable candidates. Compared to traditional hiring methods, job portals offer a wider reach, instant communication, and streamlined recruitment workflows.

The Job Portal System aims to provide an easy-to-use, responsive, and secure platform that benefits both job seekers and employers. It will include features such as user authentication, job posting and application management, role-based access, and real-time updates. The system will ensure smooth navigation, efficient backend processing, and a user-friendly interface to make recruitment simple and effective.

In Nepal, the job market is growing rapidly, with an increasing demand for skilled professionals across various industries. However, many job seekers face challenges in finding the right opportunities, while employers struggle to reach qualified candidates efficiently. This system addresses these issues by providing a centralized platform that connects both parties, reduces hiring time, and promotes transparency in the recruitment process.

1.2 Problem Statement

In Nepal, the recruitment process still faces significant inefficiencies. Job seekers often rely on scattered sources such as newspaper advertisements, social media posts, and word of mouth to find opportunities, which can be time-consuming and unreliable. Similarly, employers face challenges in reaching qualified candidates quickly, as traditional hiring methods limit visibility and require extensive manual effort to screen applications.

The lack of a centralized, user-friendly, and secure online platform leads to missed opportunities for both job seekers and employers. Many existing platforms either have complex interfaces, limited filtering options, or fail to provide real-time updates, resulting in delays in communication and hiring decisions. Additionally, there is a gap in ensuring transparency, credibility of job postings, and efficient application tracking.

The Job Portal System seeks to solve these problems by offering a single, responsive, and secure platform where employers can post jobs, manage applications, and communicate with candidates, while job seekers can easily search, apply, and track their job applications in real time.

1.3 Objectives

The objectives of this project are:

- To create a user-friendly platform where job seekers can easily search, apply, and track job applications securely, while enabling employers to efficiently post job openings and manage candidate applications.

1.4 Scope and Limitation

1.4.1 Scope

The scope of this project covers the design and development of a responsive Job Portal System that facilitates interaction between job seekers and employers. Job seekers can register, create profiles, upload resumes, search for jobs using filters, and apply online. Employers can register, post job vacancies, review applications, and manage recruitment processes. The system will include role-based access, secure authentication, and real-time notifications, ensuring accessibility from both desktop and mobile devices

1.4.2 Limitations

The limitations of this project are:

- The system requires a stable internet connection for access and functionality.
- User experience may vary depending on device performance and browser compatibility.
- Job verification relies on employer-provided information, which may limit authenticity checks.
- The platform does not guarantee job placement; it only facilitates communication between employers and job seekers.

1.5 Development Methodology

This project follows the **Waterfall Model** of software development, a linear and sequential approach where each phase must be completed before moving to the next. This methodology was chosen for its structured process and clear documentation, making it suitable for projects with well-defined requirements.

The phases include:

- **Requirement Analysis** – Gathering and documenting functional and non-functional requirements from the perspective of both job seekers and employers.
- **System Design** – Creating system architecture, database schema, and user interface designs based on the gathered requirements.
- **Implementation** – Developing the frontend using React and the backend using Node.js/Express, with MongoDB as the database, following the prepared design.
- **Testing** – Conducting unit testing, integration testing, and system testing to identify and fix errors before deployment.
- **Deployment** – Hosting the Job Portal System and making it available for public access.
- **Maintenance** – Providing regular updates, fixing bugs, and adding new features as needed.

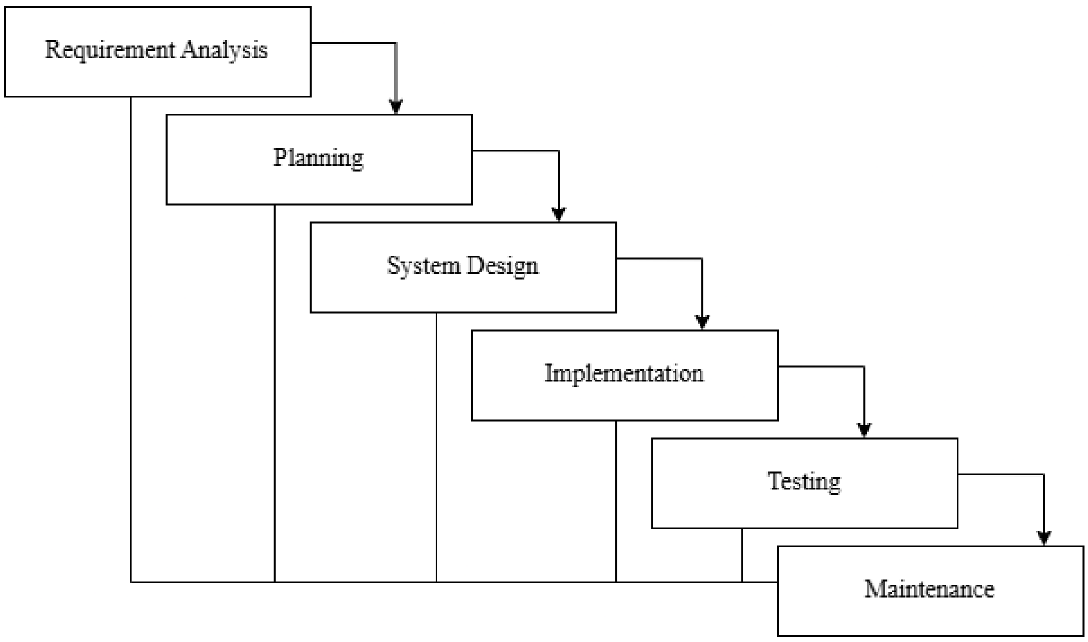


Figure 1.1 Methodology

1.6 Report Organization

Chapter 1: Introduction

This chapter provides an overview of the project, including a general introduction, the problem statement, objectives, scope, limitations, and the methodology adopted during development.

Chapter 2: Background Study and Literature Review

This chapter highlights the background study related to the project and reviews existing systems in the vacation rental domain. It discusses their working concepts, features, advantages, and drawbacks to identify gaps addressed by this project.

Chapter 3: System Analysis

This chapter details the system analysis performed for the project. It includes functional and non-functional requirements, feasibility analysis, and other aspects critical for system development. Additionally, it covers design aspects such as the Data Flow Diagram, and Entity-Relationship Diagram, illustrating the system's structure and functionality.

Chapter 4: System Design

This chapter describes the design process for the system, including detailed architectural diagrams, database schema design, and the algorithms used to develop various modules of the Vacation Home Rental System.

Chapter 5: Implementation and Testing

This chapter explains the implementation process, including the tools, frameworks, programming languages, and database platforms utilized. It also details the testing phase, including test cases, results, and the analysis of system outputs to ensure reliability and functionality.

Chapter 6: Conclusion and Future Enhancements

This chapter summarizes the outcomes of the project, highlighting key achievements and lessons learned during the development process. It also discusses potential future enhancements to improve the system's functionality and scalability.

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background

Job portals are online platforms that connect job seekers with employers by allowing them to post and apply for job opportunities. Typically, they include features such as job listings, resume uploads, profile creation, and application tracking. Employers can post vacancies, search for candidates, and manage applications, while job seekers can browse openings, apply directly, and receive notifications about relevant opportunities.

Some of the most popular global job portals are LinkedIn, Indeed, and Glassdoor. Job portal software functions similarly to other recruitment management systems, as it automates key processes such as posting job openings, filtering resumes, scheduling interviews, and managing communication between recruiters and applicants. Many platforms also support applicant tracking systems (ATS), job recommendation engines, analytics, and employer branding tools. In Nepal, notable job portals include merojob.com, ramrojob.com, and jobsnepal.com.

Online recruitment platforms have become an essential part of the hiring process for many organizations. They offer a centralized, time-efficient solution that benefits both recruiters and job seekers. With the rising demand for skilled professionals and the growth of digital infrastructure, the role of job portals has become even more significant.

Advancements in technology have made it possible to develop job portal systems with modern features such as AI-powered job matching, automated application screening, and secure cloud-based data storage. These systems help reduce the workload of HR departments, improve candidate targeting, and increase the speed of recruitment.

By offering user-friendly tools for posting, searching, and applying for jobs, job portal software serves as a bridge between talent and opportunity. It streamlines recruitment, enhances transparency, and supports the overall efficiency of the hiring process, creating benefits for businesses, job seekers, and the broader job market.

2.2Literature Review

Employment plays a critical role in economic growth and individual well-being. For many people, securing a stable job is essential for financial independence, career development, and improved quality of life. However, in many regions, including Nepal, the job search process is often challenging due to limited access to information, inefficient recruitment channels, and a mismatch between job seekers' skills and available opportunities. Traditional hiring methods such as newspaper advertisements, physical notice boards, and word-of-mouth recommendations often lack the reach, speed, and transparency needed in today's competitive market.

Globally, online job portals have transformed the recruitment process by providing a centralized platform where employers and job seekers can connect. These platforms typically allow employers to post vacancies, screen applicants, and communicate directly with candidates, while job seekers can search for jobs, submit applications, and track their progress. However, despite the benefits, challenges such as unverified job postings, spam applications, and limited personalization still persist

The COVID-19 pandemic further accelerated the adoption of digital hiring tools, as in-person recruitment events and interviews became less feasible. During this period, job portals and professional networking sites became essential for sustaining employment activities. For instance, LinkedIn reported a significant rise in virtual hiring processes, with AI-driven recommendation algorithms matching candidates to relevant positions more effectively

A study on the effectiveness of online recruitment in South Asia found that while job portals increased the reach of job advertisements, their impact on actual hiring outcomes depended heavily on the accuracy of the job–applicant matching process and the user-friendliness of the platform. In some cases, overly generic search algorithms led to mismatches, frustrating both employers and applicants.

These findings highlight that while job portals bring numerous advantages, there are still areas that require improvement, such as verification of job postings, enhanced filtering mechanisms, and better data-driven matching tools to improve recruitment efficiency and user trust.

2.3 Study of Existing Systems

- **Merojob.com** – Launched in 2009, Merojob is one of Nepal’s largest online job portals. It offers features such as job posting, resume database access, applicant tracking, and employer branding. Job seekers can create profiles, upload CVs, and apply directly through the platform. While it provides advanced search filters and career counseling services, some users have reported delays in application status updates and occasional outdated job listings .
- **Ramrojob.com** – Ramrojob focuses on providing vacancy announcements for a wide range of industries in Nepal. It offers a clean and simple interface for both job seekers and employers. The platform allows free job posting for certain categories and paid services for premium listings. However, it lacks advanced applicant tracking and analytics features found in larger global platforms .
- **Jobsnepal.com** – Established in 2000, Jobsnepal is among the earliest online recruitment platforms in the country. It connects employers and candidates through vacancy listings, resume uploads, and email alerts. While it has a broad reach and a large user base, the platform’s design and user interface are relatively outdated compared to modern portals.
- **LinkedIn** – A global professional networking platform that integrates job postings with networking features, skill endorsements, and AI-powered job recommendations. It allows employers to conduct targeted searches for candidates and promote professional branding for individuals. However, its premium recruitment tools are costly, which may limit adoption for smaller companies in Nepal.
- **Indeed** – One of the largest international job search engines, Indeed aggregates job postings from multiple sources and offers direct posting options for employers. It supports resume uploads, personalized job alerts, and employer reviews. While its reach is extensive, it may list jobs irrelevant to a specific region unless search filters are applied.

Chapter 3

System Analysis

3.1 System Analysis

To develop a successful and user-friendly job portal, Job Portal System requires a comprehensive analysis of both functional and non-functional requirements. These requirements ensure that the system meets the needs of all users, including recruiters, employees and administrators.

3.1.1 Requirement Analysis

3.1.1.1 Functional Requirements

A functional requirement describes what a system or application must do. It defines the specific behaviors, tasks, or functions the system should perform, such as user authentication, data processing, or generating reports. These requirements focus on how the system will meet the needs of its users and stakeholders.

1. User Registration and Authentication
    - Users must be able to create accounts, log in and manage their profiles
    - The system must allow users to log in securely using their credentials.
2. Profile Management
    - Job seekers must be able to create and update profiles, including personal details, educational background, work experience, and uploaded resumes.
    - Employers must be able to create company profiles with details such as company name, address, description, and logo.
3. Job Posting and Management
    - Employers must be able to post job vacancies with details such as title, description, requirements, salary, and location.
    - Employers should be able to edit, update, or delete job postings.
4. Application Management
    - Job seekers must be able to apply for jobs through the platform.

- Employers should be able to view, shortlist and reject applications.

5. Role-based Access Control
    - Job seekers, employers, and administrators should have separate access rights and functionalities based on their roles.
    - Administrators must be able to monitor platform activities and manage reported content.

Use Case Diagram

A Use Case Diagram is a visual representation that depicts the interactions between users (actors) and a system. It outlines the functional requirements of a system by showing various use cases, which are the actions or services that the system provides to fulfill user needs.

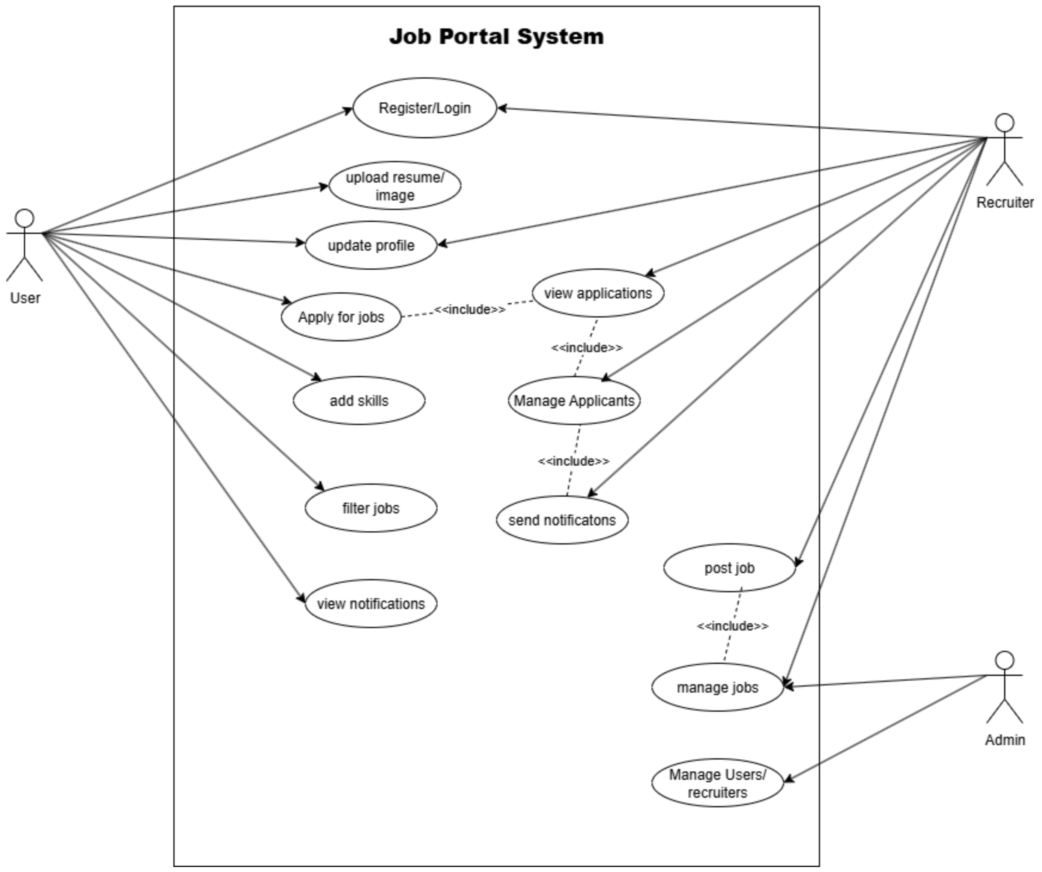


Figure 3.1 Use Case Diagram

3.1.1.2 Non Functional Requirements

A nonfunctional requirement refers to the attributes or qualities that a system must have, rather than the specific behaviors it must perform. These requirements focus on how well the system performs its tasks, such as performance, security, usability, reliability, and scalability. For example, a nonfunctional requirement might specify that a website should load in less than 2 seconds or that the system should handle upto 1,000 users simultaneously.

1. Scalability: The platform must support a growing number of users, listings, and transactions without performance degradation.

2. Security: Robust security measures to protect user data, including encryption, secure authentication, and regular security audits.

3. Performance: The system should load quickly and handle high traffic during peak times, ensuring a smooth user experience.

4. Usability: Intuitive and user-friendly interface designed for all user types, including those with limited technical skills.

5. Reliability: The platform should ensure high availability with minimal downtime and provide backups for data protection.

6. Hardware Requirements:
• Laptop, desktop, or smartphone with 4 GB RAM.
• Screen resolution of 1366x768 or higher.
• Intel Core i3 processor or better.
• Internet connection with 1Mbps speed.

3.1.2 Feasibility Study

A feasibility study is an assessment that evaluates whether a project or idea is practical and achievable. It looks at different factors like cost, resources, technology, and time to determine if the project can be successfully completed. It helps to identify potential problems and risks before starting the project, ensuring that it's worth pursuing.

3.1.2.1 Technical

The Job Portal System will be developed using modern and widely accepted web technologies. The frontend will be built with React.js to provide a responsive and user-friendly interface, while the backend will use Node.js with Express.js to handle server-side operations efficiently. MongoDB will be used as the database to store user data, job listings, and applications securely.
These technologies are well-supported, scalable, and suitable for handling features such as user authentication, job posting, application tracking, and notifications. The development team's experience with these tools ensures that technical challenges can be effectively addressed, making the project technically feasible.

3.1.2.2 Operational

The Job Portal is designed to meet the needs of both job seekers and employers by providing an intuitive platform for posting, searching, and applying for jobs. The interface will be simple to navigate, allowing users to quickly access relevant features such as advanced job filters, profile management, and application tracking. Integrated tools like secure authentication, resume uploads, and in-platform messaging will ensure smooth communication between candidates and recruiters. Given the increasing reliance on online recruitment and the similarity of the platform's workflow to widely used job sites, the system will be easily adopted by users seeking a more efficient and reliable hiring solution.

3.1.2.3 Schedule

Schedule feasibility is the degree to which a deadline for a strategy, plan, project or process is realistic and achievable.

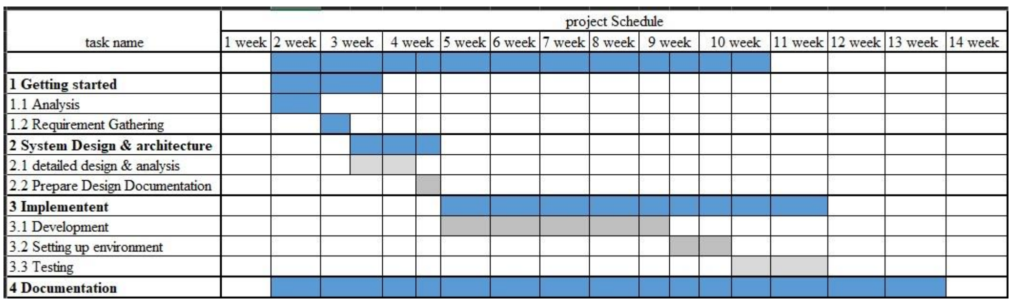


Figure 3.2 Gantt chart

3.1.3.1 Data modeling using ER Diagrams

An Entity-Relationship (ER) diagram is a visual representation used to model the structure of a database. It illustrates the relationships between entities (e.g., people, objects, or concepts) and their attributes. ER diagrams use symbols like rectangles to represent entities, diamonds for relationships, and ovals for attributes, helping to design and communicate the data structure clearly before actual implementation.

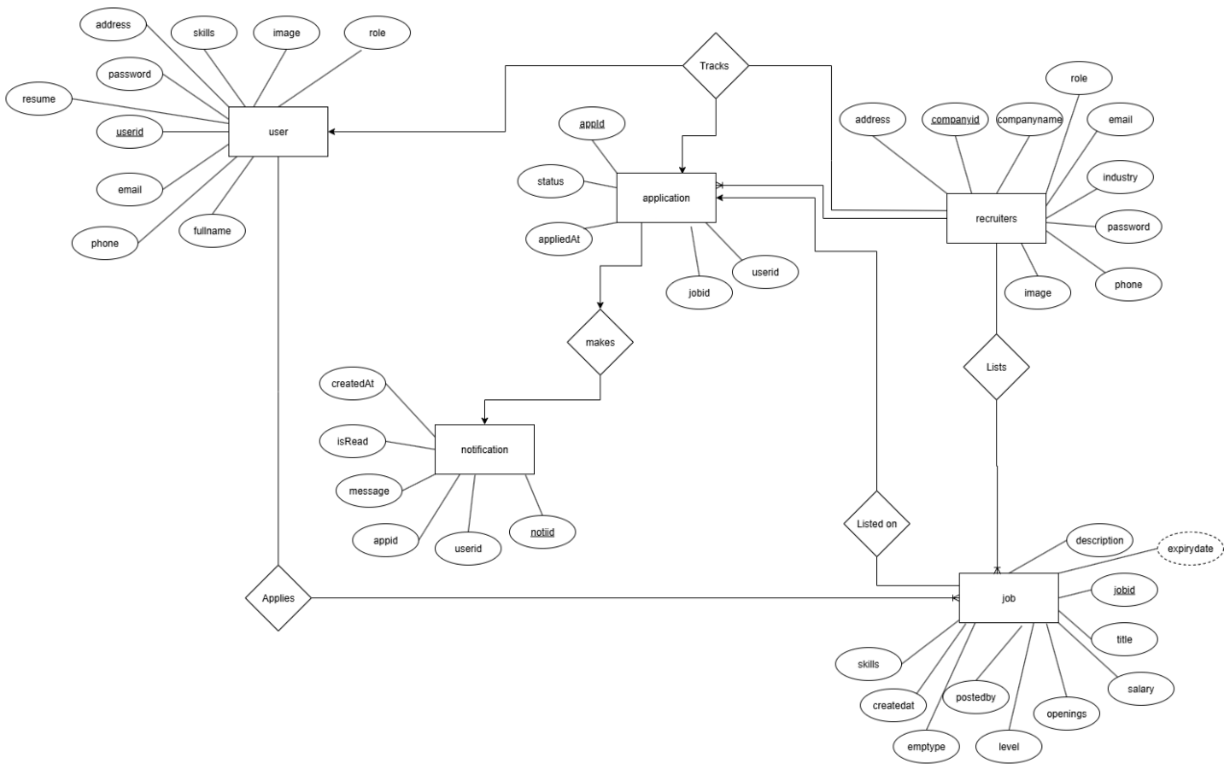


Figure 3.3 ER Diagram

3.1.3.2 Process modeling using DFD

The process modeling of the Job-portal system is done using Data Flow Diagram (DFD). DFDs shows how the data moves through the system and the interaction between users, recruiters, admins and different system components. It illustrates processes like registration, job listing, applying, tracking along with the data inputs, outputs and storage. DFDs help in understanding the system's functions and identifying areas for improvement

1. **Level 0 DFD**

   Level 0 DFD (Context Diagram) for Job-Portal System represents the highest-level view showing the entire system as a single process. It illustrates how the system interacts with external entities without showing any internal processes or data stores.

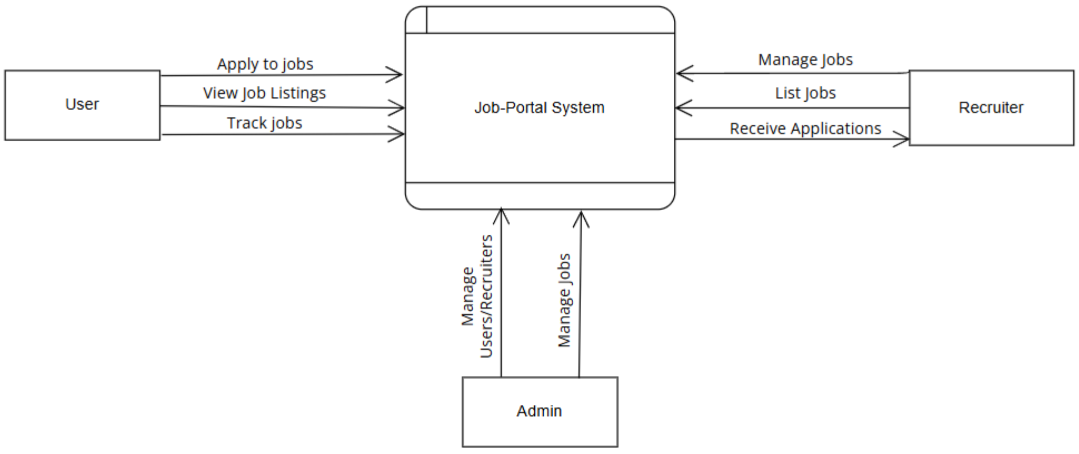


Figure 3.4 Level 0 DFD

2. **Level 1 DFD of User**

The Level 1 Data Flow Diagram for the User role illustrates the processes a job seeker can perform within the system. Users can register or log in, after which they can search and filter jobs using criteria such as keywords, salary range, or location. The system retrieves matching job listings from the job_details data store. Once a suitable job is found, the user can submit an application, which is stored in the application_details data store. Users can also view the status of their applications, which is retrieved from the same application_details store. This process ensures smooth job searching, application submission, and application tracking for user

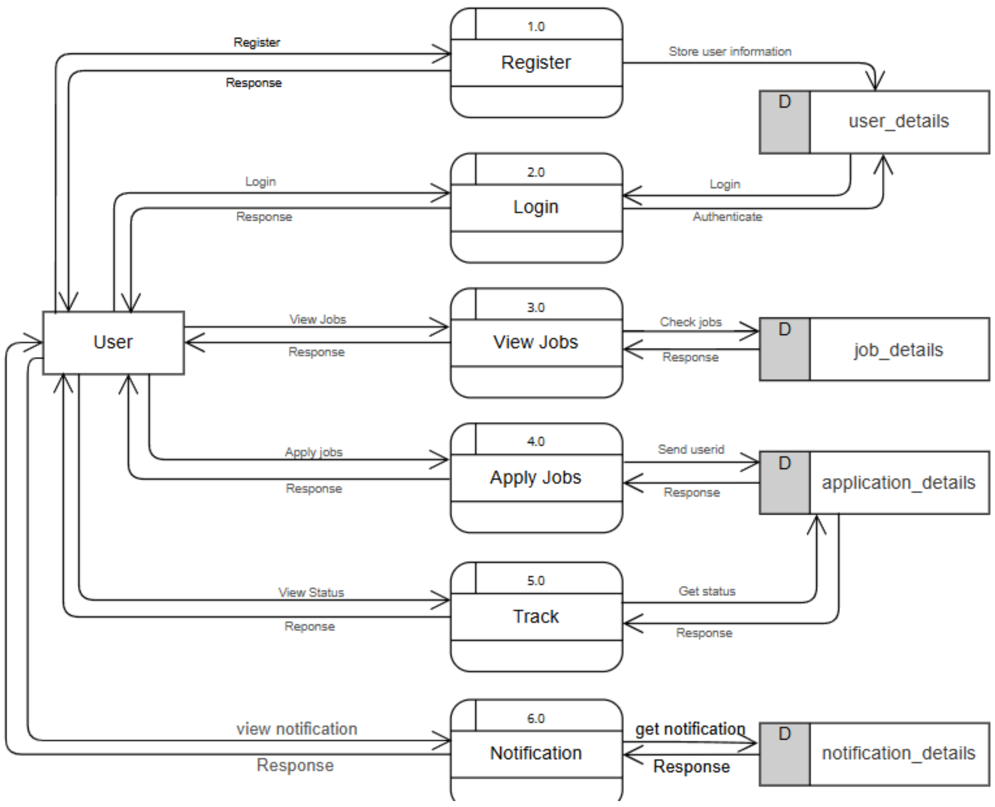


Figure 3.5 Level 1 DFD of User

3. Level 1 DFD of Recruiter

The Level 1 Data Flow Diagram for the Recruiter role focuses on how recruiters manage job postings and applications. Recruiters can register or log in, then list new job opportunities by adding data to the job_details store. They can manage their existing job postings by updating or removing them when necessary. Additionally, recruiters can review job applications retrieved from the application_details store and update the application status for each candidate. This ensures recruiters can efficiently post, manage, and monitor job vacancies while controlling the progress of applicants through the recruitment process.

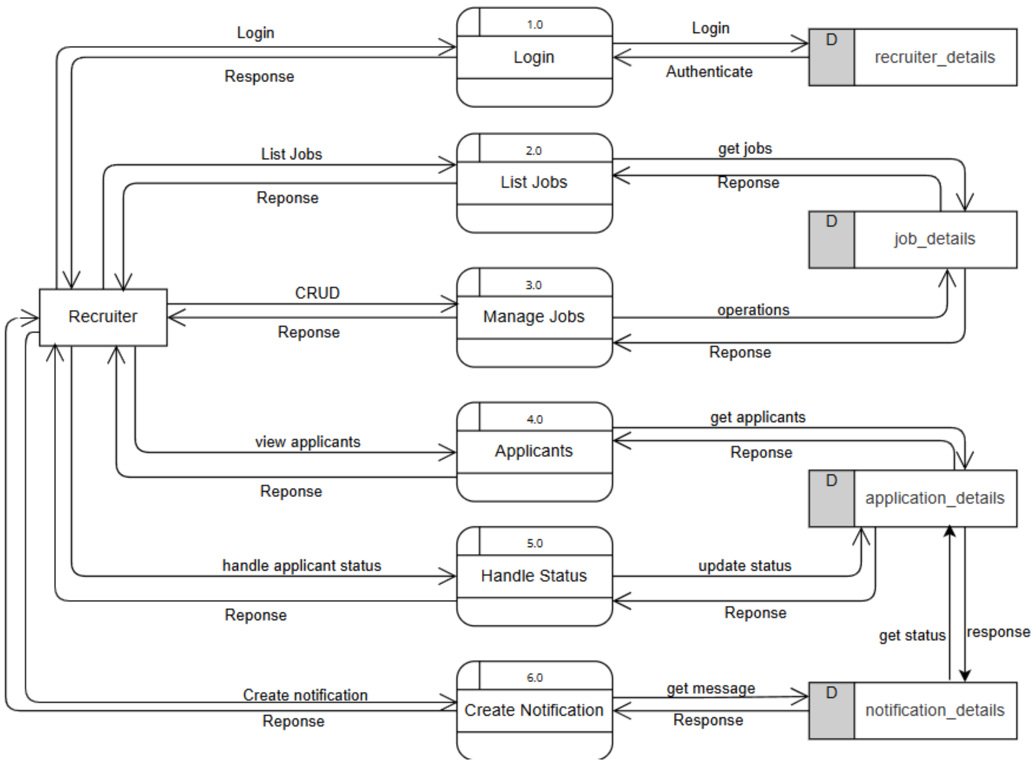


Figure 3.6 Level 1 DFD of Recruiter

CHAPTER 4

SYSTEM DESIGN

4.1 Design

System design defines the architecture, components, modules, interfaces, and data for the Job-Portal System to meet the specified requirements. It involves translating user needs into a detailed blueprint that guides the development and implementation of the system. The system design covers both the high-level structure of the system, such as its overall flow and user interactions, and the low-level details, including the implementation of specific modules. This ensures a well-organized, efficient, and scalable solution for the vacation home rental platform.

4.1.1 Database Design

The figure below represents the database schema design of the Job Portal system. Database schema design defines the logical structure of the database, showing how different entities are related. In the Job Portal, there are four main collections: users, recruiters, jobs, and applications. Each collection has its own fields where the _id field serves as the primary key. If the primary key of one collection is referenced in another, it becomes a foreign key, represented in the schema with a connecting line. For example, the applications collection uses userId (foreign key from users) and jobId (foreign key from jobs) to maintain relationships. Similarly, the jobs collection has a postedBy field that links to recruiters. Each attribute also has a specified data type such as string, objectId, date, or double. This schema ensures proper normalization, data integrity, and establishes relationships between applicants, recruiters, and job postings in the Job Portal system.

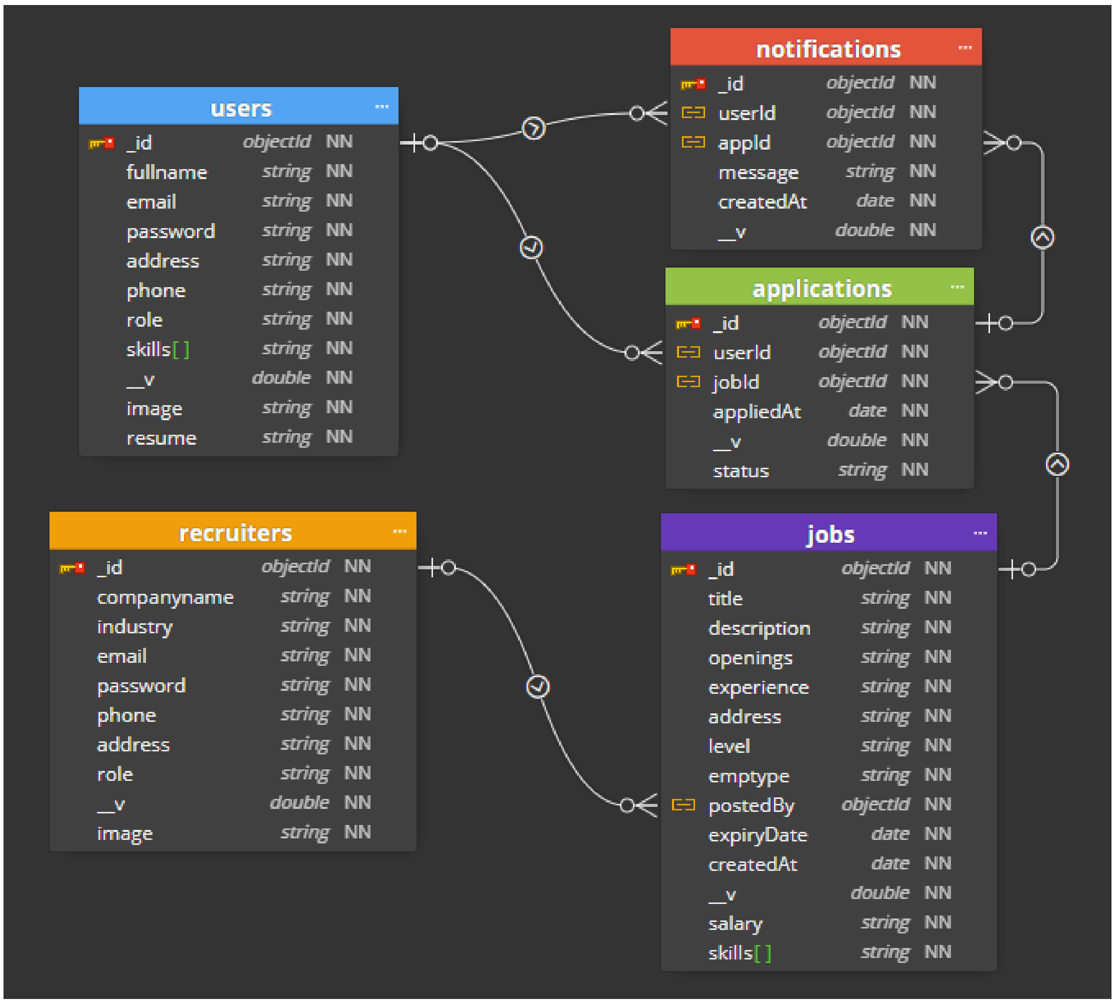


Figure 4.1 Database Schema

4.1.2 System Design

The Job Portal system is designed to connect job seekers with recruiters in an efficient and user-friendly manner. The front-end is developed using React JS, and CSS, ensuring a clean design and smooth navigation for both applicants and recruiters. The authentication and authorization module manages user registration, login, and secure access control, using JWT for token-based authentication and Bcrypt for password encryption. The server-side application is built on Node.js and Express.js, handling all backend operations, data validation, and business logic. MongoDB serves as the primary database for structured and semi-structured data storage, maintaining collections for users, recruiters, jobs, and applications. This architecture provides an end-to-end solution for managing job postings, applications, and recruitment workflows.

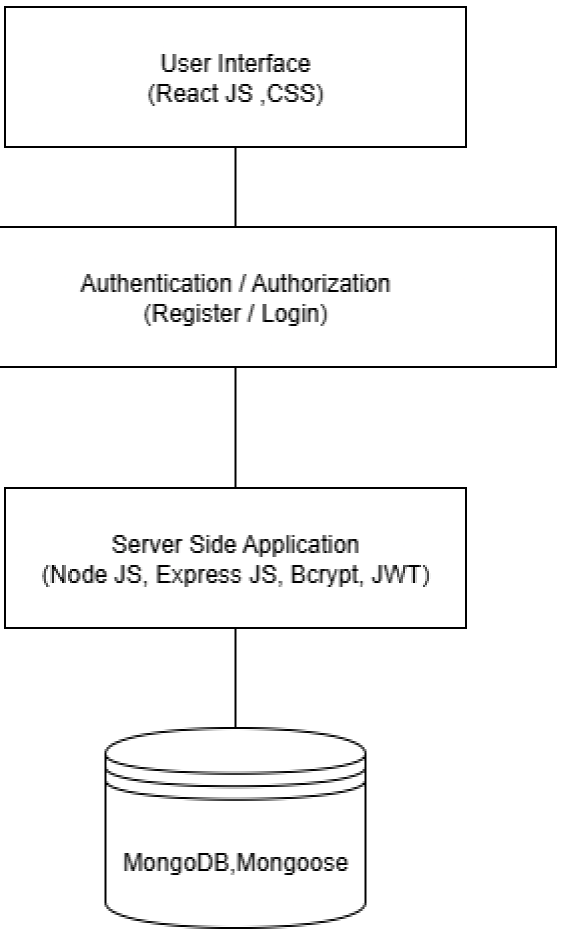


Figure 4.2 System Design

4.1.3 Flowchart

The flowchart of the Job Portal System illustrates the process flow among its primary users: Recruiters, Users, and Admins. Users can register, create profiles, upload resumes, search for jobs, apply to openings, and track their application status. Recruiters can register, post job listings, manage applications, and shortlist or reject candidates. Admins oversee the entire platform by reviewing recruiter’s registrations, monitoring job postings, and ensuring compliance with policies. The flowchart visually connects these actions, showing how data flows between different users and the system to provide a smooth and efficient recruitment process.

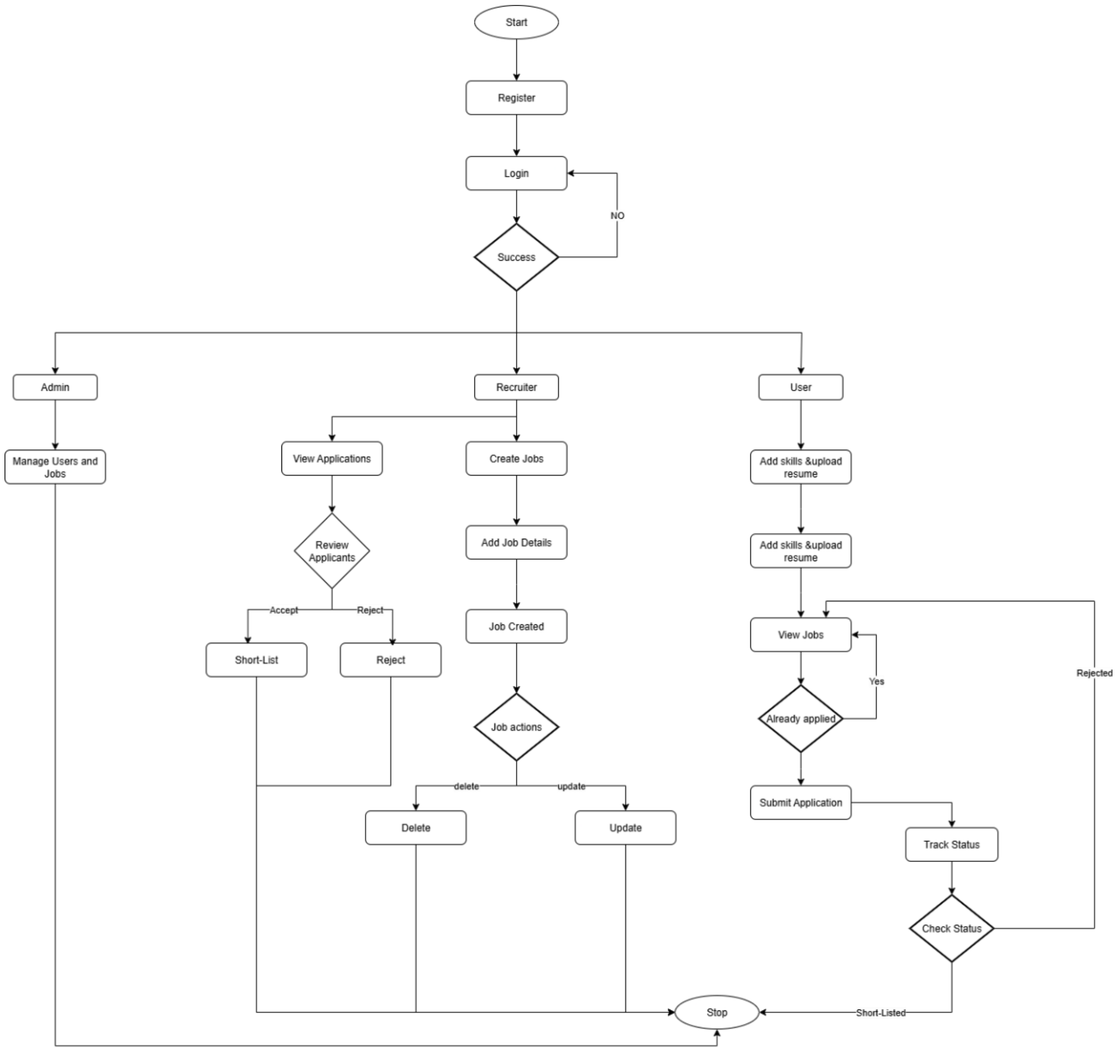


Figure 4.3 System Flowchart

4.1.4 Interface Design

The UI of Job-Portal System is developed using React JS, CSS. The overall interface of the application is meant to be as satisfactory, interactive and user-friendly as possible. The system encompasses essential pages such as Signup, Login, Home, Account Setting, Dashboard, Job Listing, Create Job, Apply Page.

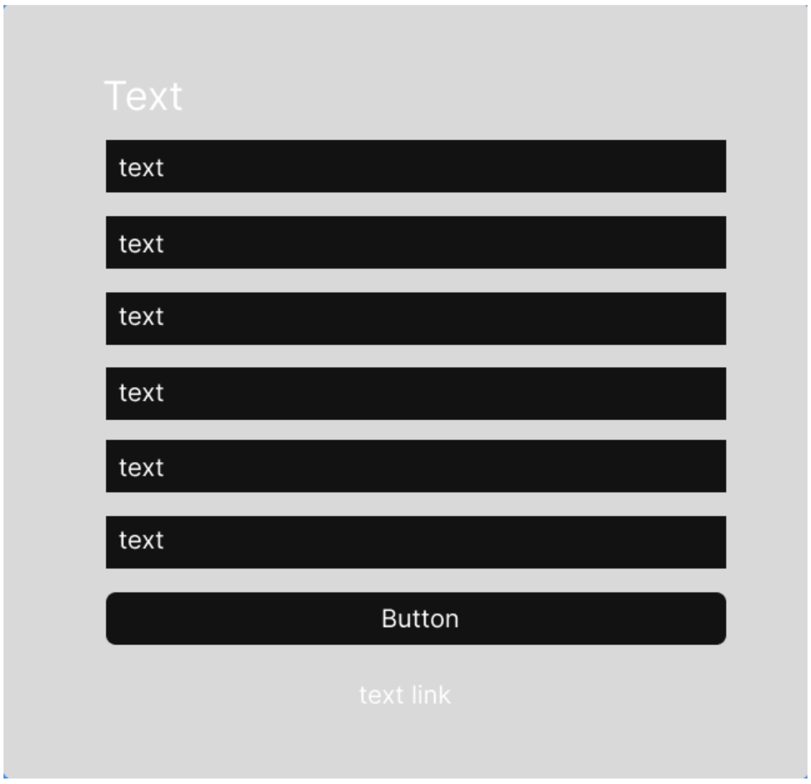


Figure 4.4 Registration Form

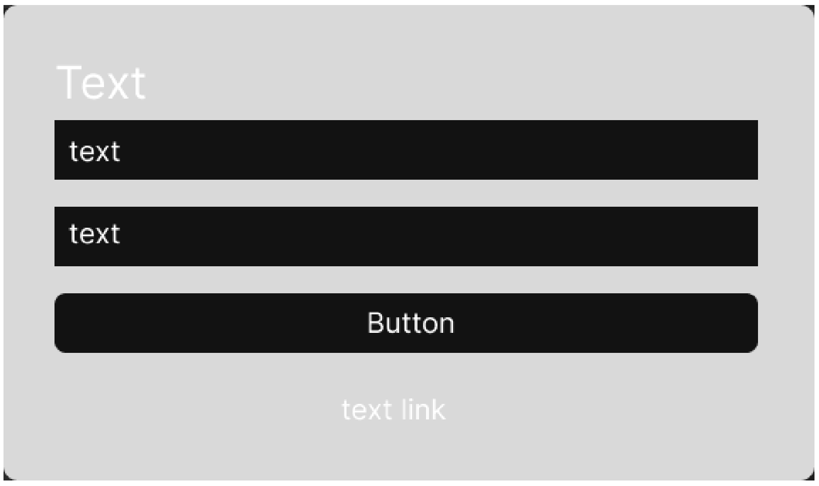


Figure 4.5 Login Form

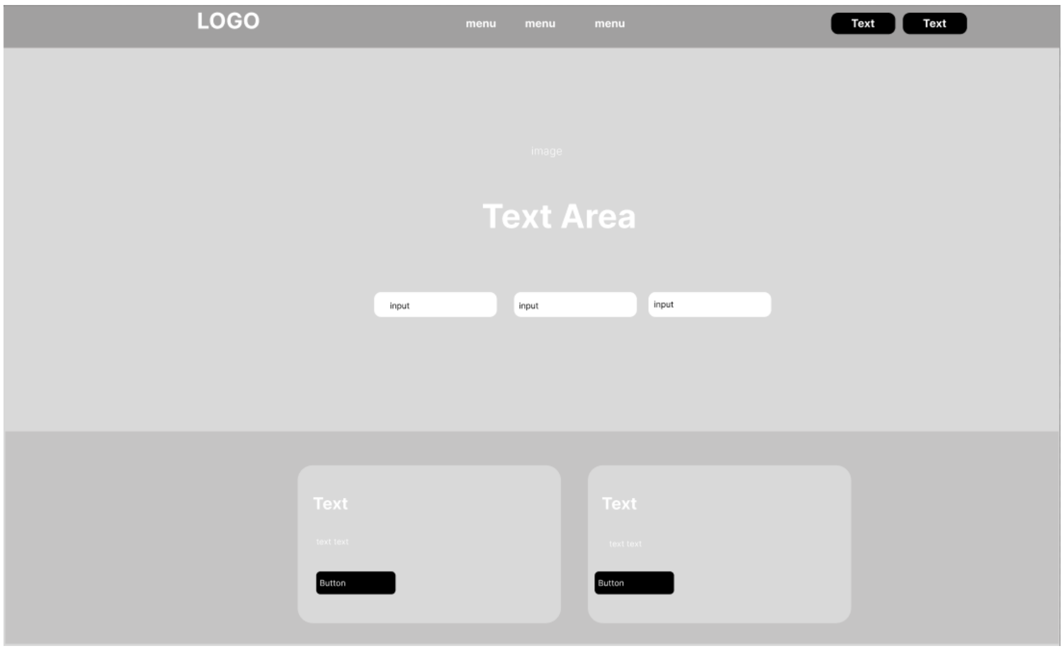


Figure 4.6 Landing Page

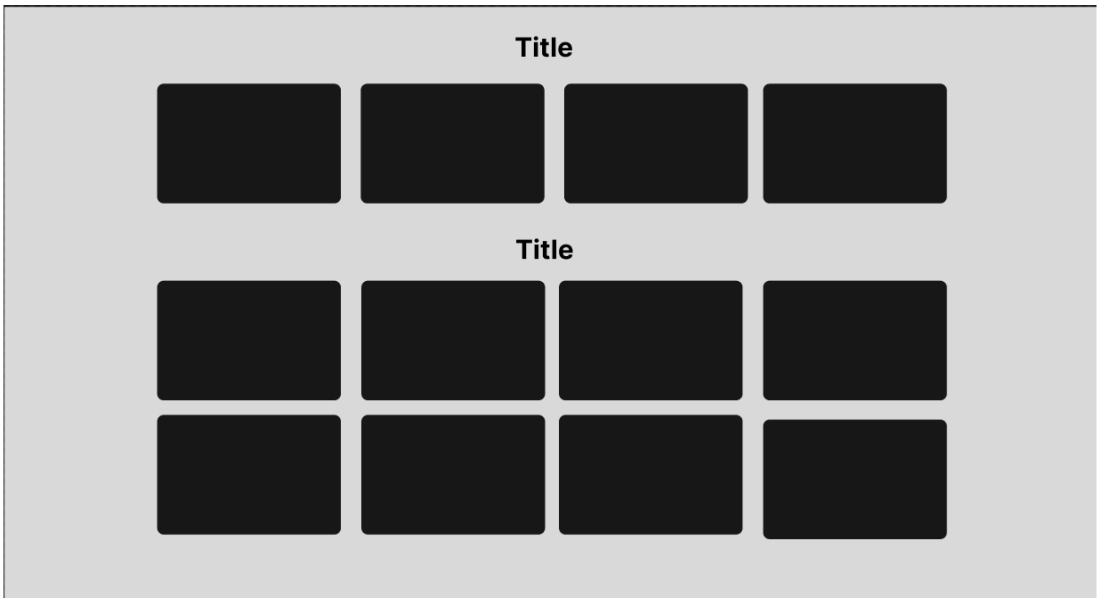


Figure 4.7 Interface for homepage

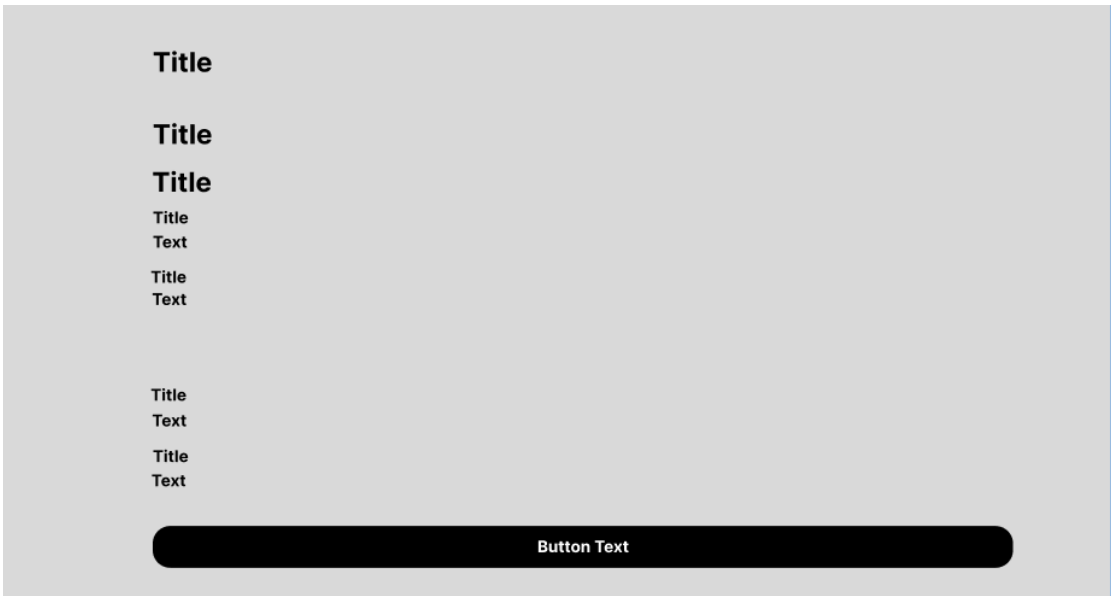


Figure 4.8 Job Apply

4.2 Algorithm Details

4.2.1 Introduction

The job portal system implements matches users with the most relevant job postings by comparing skills of users with skills required by jobs. To measure similarity, the Cosine Similarity algorithm is used, which calculates the angle between two vectors in multi-dimensional space.

- If similarity = 1, user and job skill sets are identical.
- If similarity = 0, no skills match.
- Values between 0 and 1 indicate the degree of similarity.

4.2.2 Cosine Similarity Formula

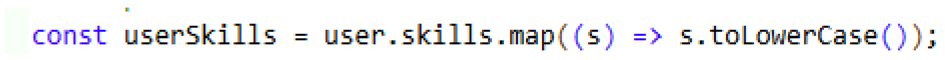
Cosine Similarity Formula= $\cos\theta = \frac{\sum_{i=1}^{n} U_i J_i}{\sqrt{\sum_{i=1}^{n} U_i^2} \cdot \sqrt{\sum_{i=1}^{n} J_i^2}}$

Where:

- Ui= value of skill iii in the user vector
- Ji = value of skill iii in the job vector

4.2.3 Implementation Steps

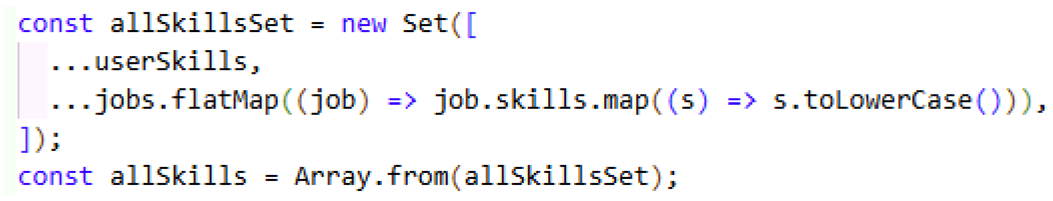
Step 1: All skills are converted to lowercase to avoid duplication.



```
const userSkills = user.skills.map((s) => s.toLowerCase());
```

Step 2: Create Global skill set

A universal set of skills is built by combining user and job skills.



```
const allSkillsSet = new Set([
  ...userSkills,
  ...jobs.flatMap((job) => job.skills.map((s) => s.toLowerCase())),
]);
const allSkills = Array.from(allSkillsSet);
```

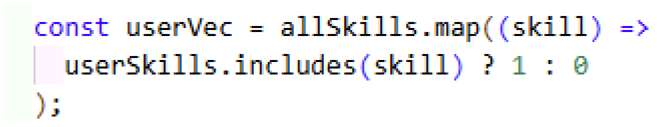
Example:

Skills Set	**Values**
User	["javascript", "nodejs"]
Job A	["javascript", "react"]
Job B	["java", "spring"]
All	["javascript", "nodejs", "react", "java", "spring"]

Step 3: Create Vectors

Vectors are created for user and each job.

- If a skill exists → 1
- If not → 0



```
const userVec = allSkills.map((skill) =>
  userSkills.includes(skill) ? 1 : 0
);
```

Skills	**javascript**	**nodejs**	**react**	**java**	**spring**
User	1	1	0	0	0
Job A	1	0	1	0	0
Job B	0	0	0	1	1

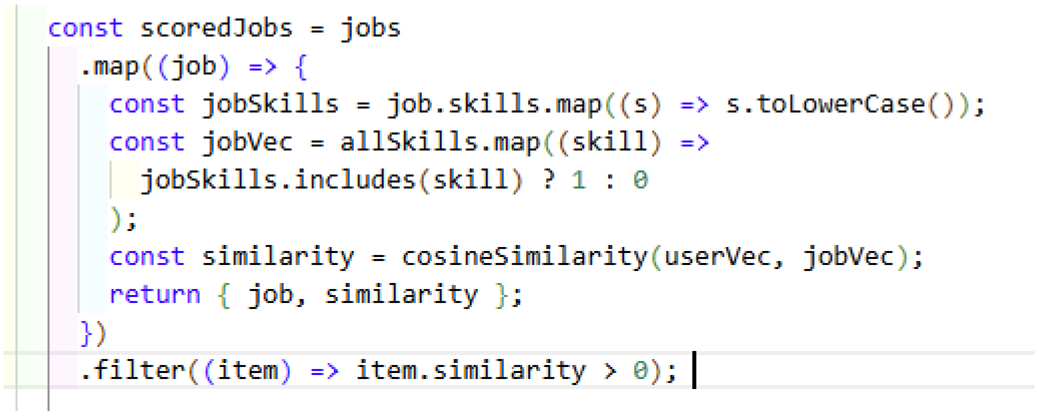
Step 4: Cosine Similarity Calculation

For Job A:

- Dot product = $(1*1)+(1*0)+(0*1)+(0*0)+(0*0)=1$
- User magnitude = $\sqrt{1^2 + 1^2} = \sqrt{2}$
- Job A magnitude = $\sqrt{1^2 + 1^2} = \sqrt{2}$
- Cosine similarity = $\frac{1}{\sqrt{2}\times\sqrt{2}} = 0.5$

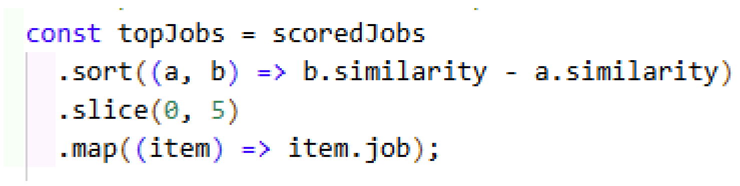
For Job B:

- Dot product = $(1*0)+(1*0)+(0*0)+(0*1)+(0*1)=0$
- Cosine similarity = **0**



```
const scoredJobs = jobs
  .map((job) => {
    const jobSkills = job.skills.map((s) => s.toLowerCase());
    const jobVec = allSkills.map((skill) =>
      jobSkills.includes(skill) ? 1 : 0
    );
    const similarity = cosineSimilarity(userVec, jobVec);
    return { job, similarity };
  })
  .filter((item) => item.similarity > 0);
```

Step 5: Ranking Jobs



```
const topJobs = scoredJobs
  .sort((a, b) => b.similarity - a.similarity)
  .slice(0, 5)
  .map((item) => item.job);
```

Jobs are ranked by similarity score in descending order. Top 5 jobs are returned.

Job	**Similarity Score**
Job A	0.50
Job B	0.00

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

System implementation is a key phase in the development of our project, where the system design is transformed into a fully functional and operational solution. This phase ensures that the system meets the required specifications and works as intended. In our project, we followed the Waterfall development methodology, which proceeds in a linear and sequential manner. Each stage—requirement analysis, system design, development, testing, and deployment—was completed before moving to the next. By using this structured approach, we ensured that all requirements were clearly defined from the beginning and that the system was developed systematically, resulting in a stable and reliable final product.

5.1.1 Tools Used

Frontend Technologies

- HTML: HTML (HyperText Markup Language) is the standard language used to structure the content of web pages. It defines elements like headings, paragraphs, links, images, and forms, allowing browsers to display web content correctly. HTML serves as the backbone of any website's structure.

- CSS: CSS (Cascading Style Sheets) is used to style and layout web pages. It controls the appearance of HTML elements, such as fonts, colors, spacing, and positioning. CSS ensures that web pages are visually appealing and responsive across various devices.

- JavaScript: JavaScript is a programming language that adds interactivity and dynamic behavior to web pages. It enables features like form validation, animations, and data fetching, making websites more interactive and responsive to user actions.

JavaScript Frameworks and Libraries

- React: React is a JavaScript library used to build user interfaces, especially single-page applications. It allows developers to create reusable UI components that update efficiently, making applications faster and easier to maintain.

- React Router DOM: React Router DOM is a library used in React applications to handle routing. It enables navigation between different components or pages within a single-page application (SPA), enhancing user experience by loading content dynamically without reloading the page.

Backend Technologies

- Node.js: Node.js is a JavaScript runtime environment that allows developers to run JavaScript on the server side. It is known for its speed and scalability, making it suitable for building real-time applications and APIs.

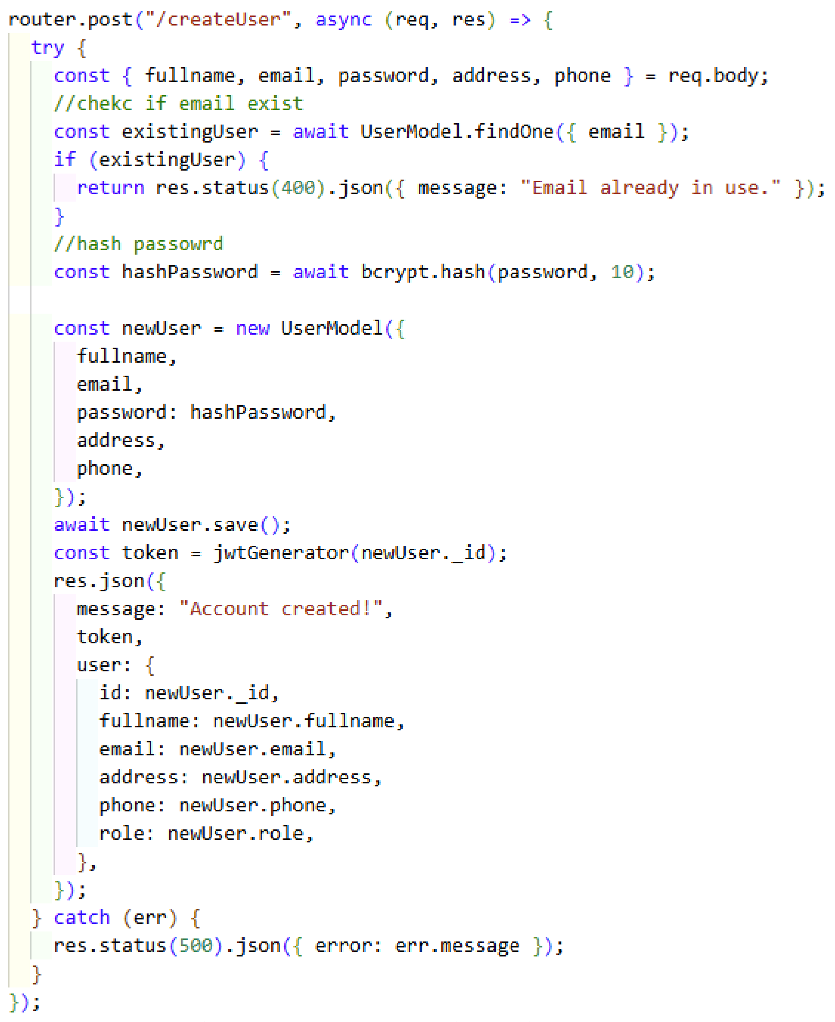
- Express.js: Express.js is a minimal and flexible Node.js web application framework. It simplifies the process of handling HTTP requests, routing, and middleware, making backend development faster and more efficient.

Database

- MongoDB: MongoDB is an open-source, NoSQL database system. It stores data in a flexible, document-oriented format using JSON-like structures, which allows for handling unstructured or semi-structured data efficiently. It supports scalability, high availability, and fast query performance.

5.2 Implementation Details of Modules

5.2.1 User Registration and Authentication Module



```
router.post("/createUser", async (req, res) => {
  try {
    const { fullname, email, password, address, phone } = req.body;
    //chekc if email exist
    const existingUser = await UserModel.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: "Email already in use." });
    }
    //hash passowrd
    const hashPassword = await bcrypt.hash(password, 10);

    const newUser = new UserModel({
      fullname,
      email,
      password: hashPassword,
      address,
      phone,
    });
    await newUser.save();
    const token = jwtGenerator(newUser._id);
    res.json({
      message: "Account created!",
      token,
      user: {
        id: newUser._id,
        fullname: newUser.fullname,
        email: newUser.email,
        address: newUser.address,
        phone: newUser.phone,
        role: newUser.role,
      },
    });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

Figure 5.1 User Registration Module

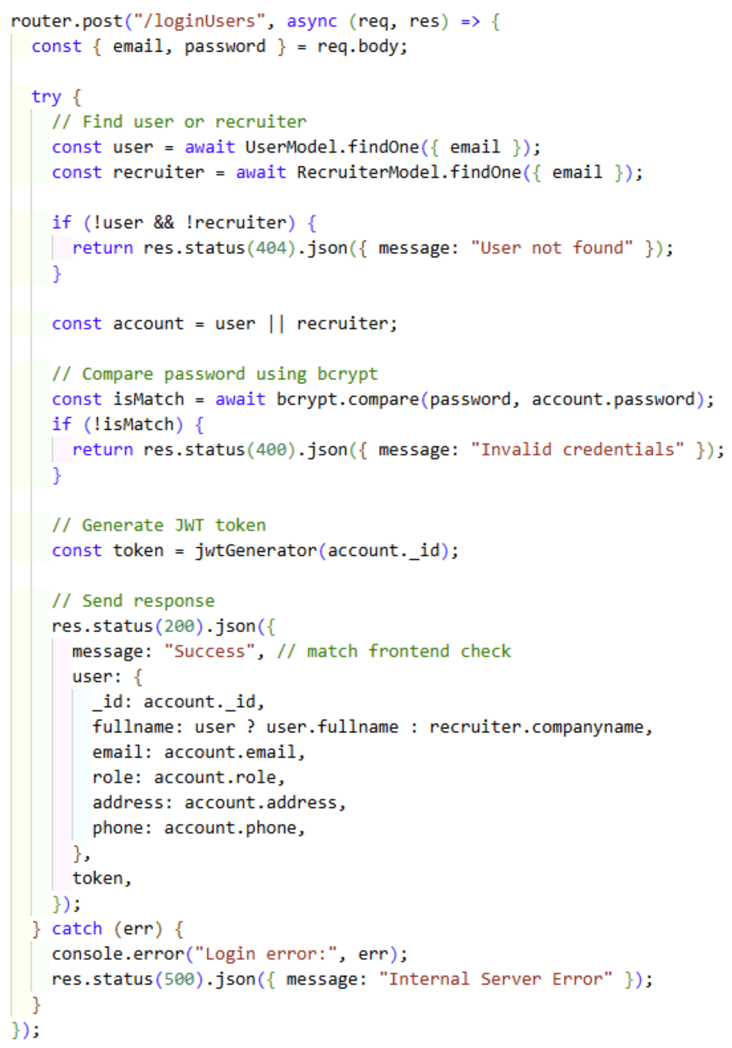
Purpose:

- Provides user registration by storing user details securely.
- Provides hashing of password.
- Prevents duplicate emails.

Implementation

- Accepts data from the frontend.
- Checks if email already exists in database.
- If not, hashes the password and stores the user data.
- Generates a JWT token upon successful registration.

5.2.2 Login and Authentication Module



```
router.post("/loginUsers", async (req, res) => {
  const { email, password } = req.body;

  try {
    // Find user or recruiter
    const user = await UserModel.findOne({ email });
    const recruiter = await RecruiterModel.findOne({ email });

    if (!user && !recruiter) {
      return res.status(404).json({ message: "User not found" });
    }

    const account = user || recruiter;

    // Compare password using bcrypt
    const isMatch = await bcrypt.compare(password, account.password);
    if (!isMatch) {
      return res.status(400).json({ message: "Invalid credentials" });
    }

    // Generate JWT token
    const token = jwtGenerator(account._id);

    // Send response
    res.status(200).json({
      message: "Success", // match frontend check
      user: {
        _id: account._id,
        fullname: user ? user.fullname : recruiter.companyname,
        email: account.email,
        role: account.role,
        address: account.address,
        phone: account.phone,
      },
      token,
    });
  } catch (err) {
    console.error("Login error:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
});
```

Figure 5.2 Login and Authentication Module

Purpose

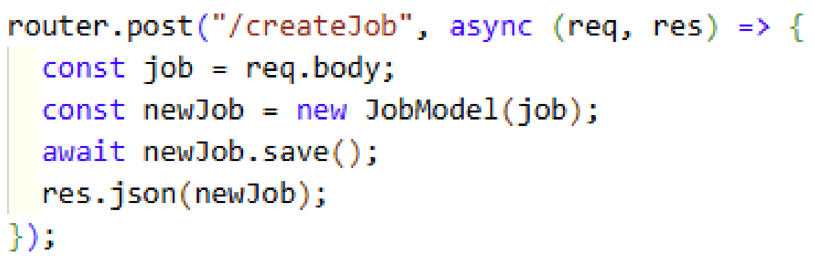
- Authenticates users by verifying email and password.

Implementation:

- Accepts email and password from the frontend.
- Validates the email against the database.
- Compares the provided password with the stored hashed password.
- Generates a JWT token for valid users.

5.3 Job Listing Module

5.3.1 Job Listing Creation



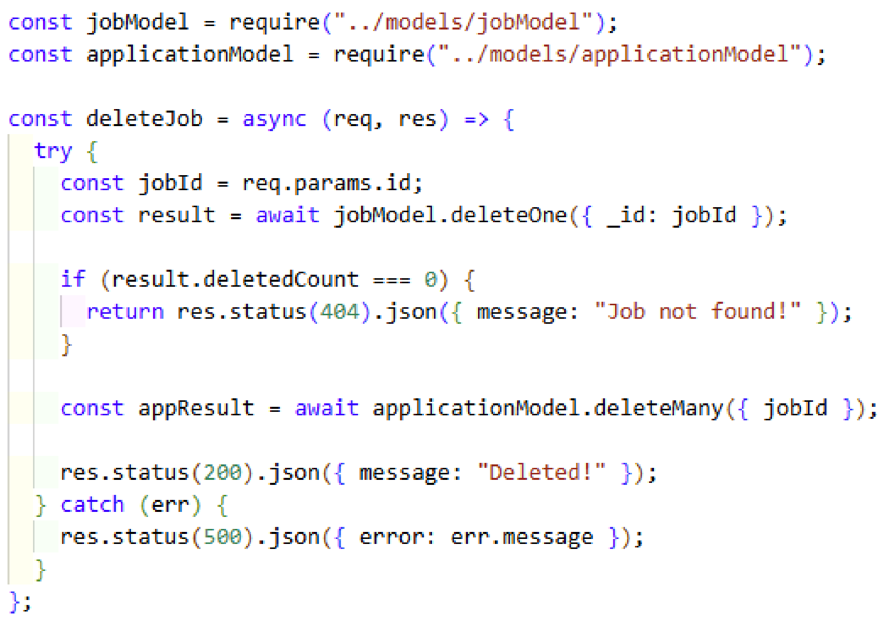
```
router.post("/createJob", async (req, res) => {
  const job = req.body;
  const newJob = new JobModel(job);
  await newJob.save();
  res.json(newJob);
});
```

Figure 5.3 Job Listing Module

Purpose:

- Allow recruiters to create job listings with detailed information.

5.3.2 Job Listing Deletion Module



```
const jobModel = require("../models/jobModel");
const applicationModel = require("../models/applicationModel");

const deleteJob = async (req, res) => {
  try {
    const jobId = req.params.id;
    const result = await jobModel.deleteOne({ _id: jobId });

    if (result.deletedCount === 0) {
      return res.status(404).json({ message: "Job not found!" });
    }

    const appResult = await applicationModel.deleteMany({ jobId });

    res.status(200).json({ message: "Deleted!" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

Figure 5.4 Job Deletion Module

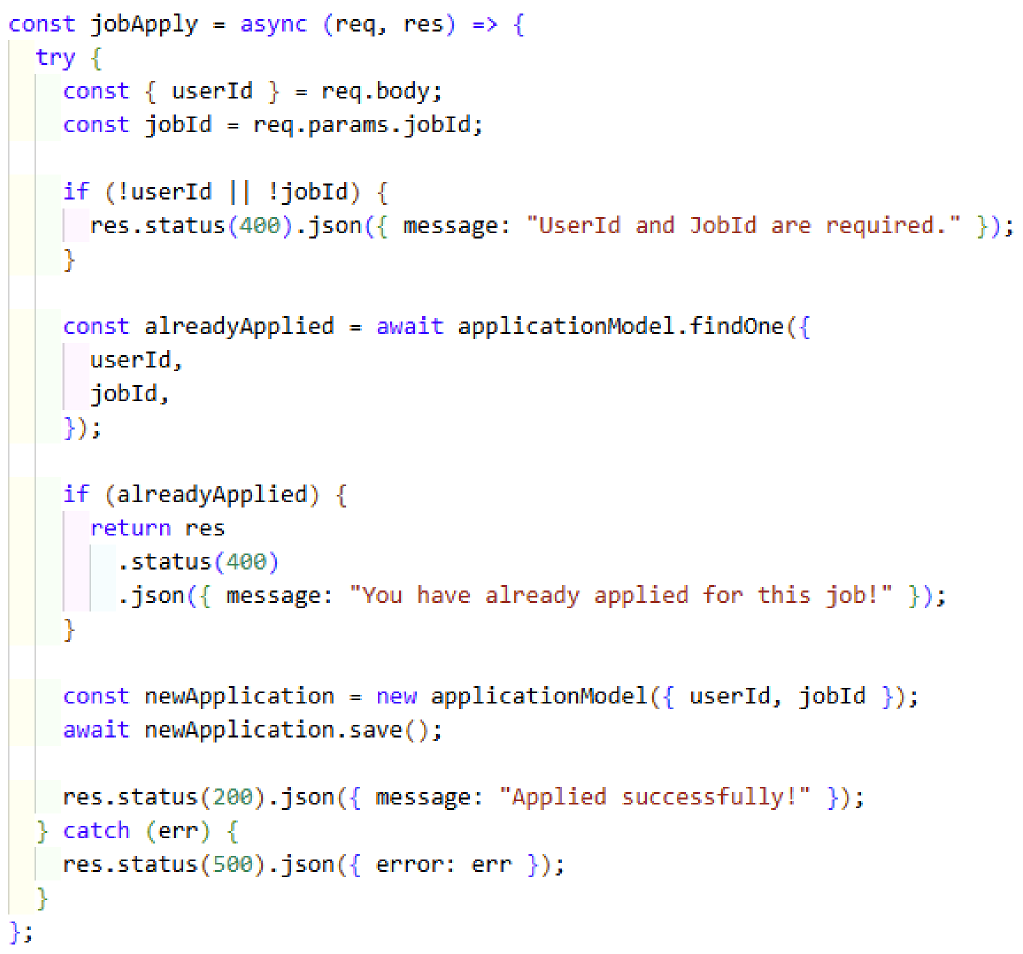
Purpose:

- Enables recruiters to delete their job listings.
- Ensures applications are also deleted.

5.3.3 Job Listing Update

5.4 Application and Applicants Management Module

5.4.1 Applying



```
const jobApply = async (req, res) => {
  try {
    const { userId } = req.body;
    const jobId = req.params.jobId;

    if (!userId || !jobId) {
      res.status(400).json({ message: "UserId and JobId are required." });
    }

    const alreadyApplied = await applicationModel.findOne({
      userId,
      jobId,
    });

    if (alreadyApplied) {
      return res
        .status(400)
        .json({ message: "You have already applied for this job!" });
    }

    const newApplication = new applicationModel({ userId, jobId });
    await newApplication.save();

    res.status(200).json({ message: "Applied successfully!" });
  } catch (err) {
    res.status(500).json({ error: err });
  }
};
```

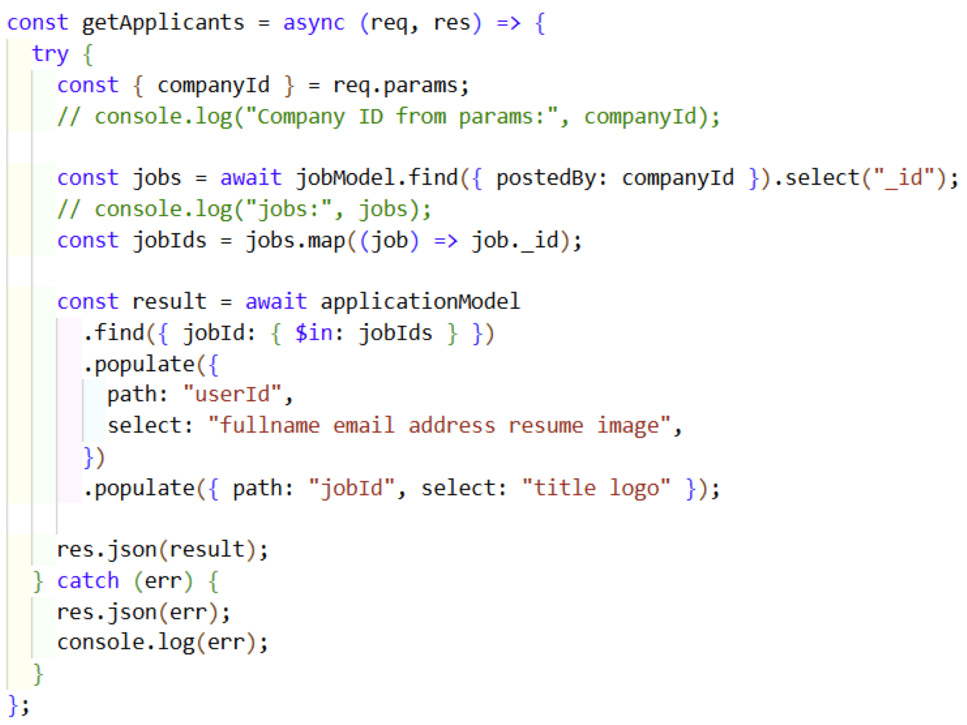
Figure 5.5 Job Apply

Purpose:

- Creates applications records after applying.
- Stores applications details including user and job.
- Ensures there is no multiple application from a user per job

5.4.2 Tracking and Status Update Modules

Getting Applicants Details



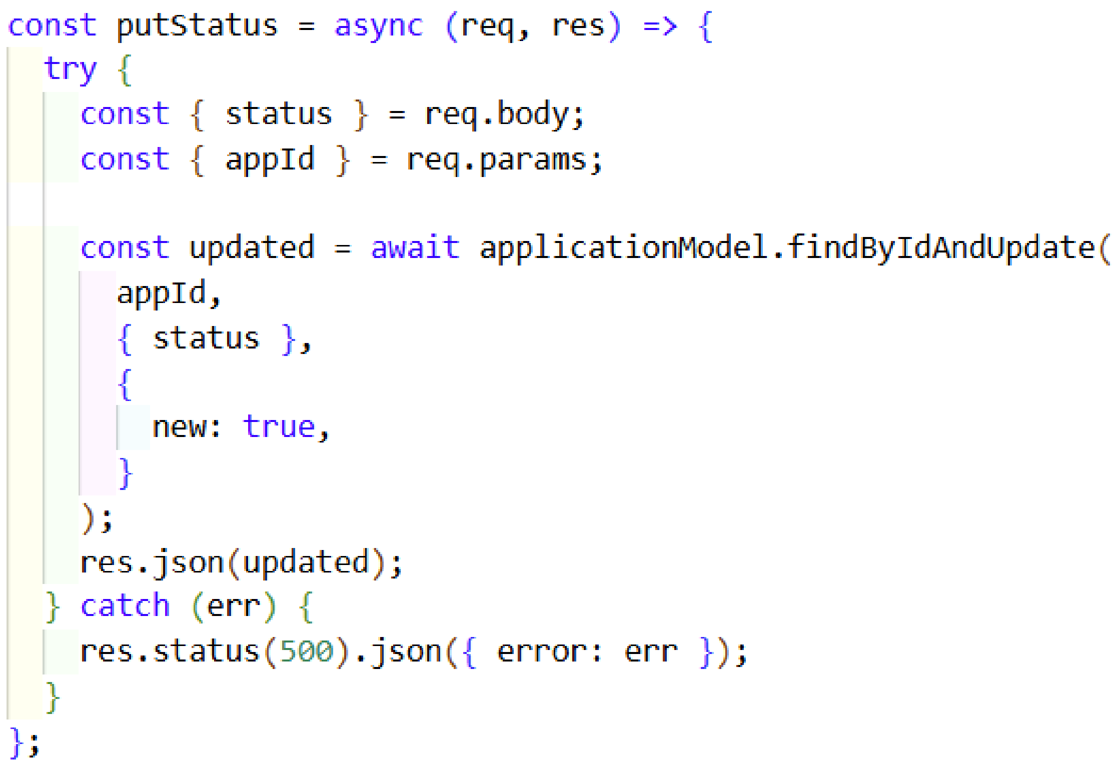
```
const getApplicants = async (req, res) => {
  try {
    const { companyId } = req.params;
    // console.log("Company ID from params:", companyId);

    const jobs = await jobModel.find({ postedBy: companyId }).select("_id");
    // console.log("jobs:", jobs);
    const jobIds = jobs.map((job) => job._id);

    const result = await applicationModel
      .find({ jobId: { $in: jobIds } })
      .populate({
        path: "userId",
        select: "fullname email address resume image",
      })
      .populate({ path: "jobId", select: "title logo" });

    res.json(result);
  } catch (err) {
    res.json(err);
    console.log(err);
  }
};
```

Figure 5.6 Applicants Details

Purpose:

- Recruiters can view applications to their posted job.
- Can view resume.
- Enables to update status.

Updating Status



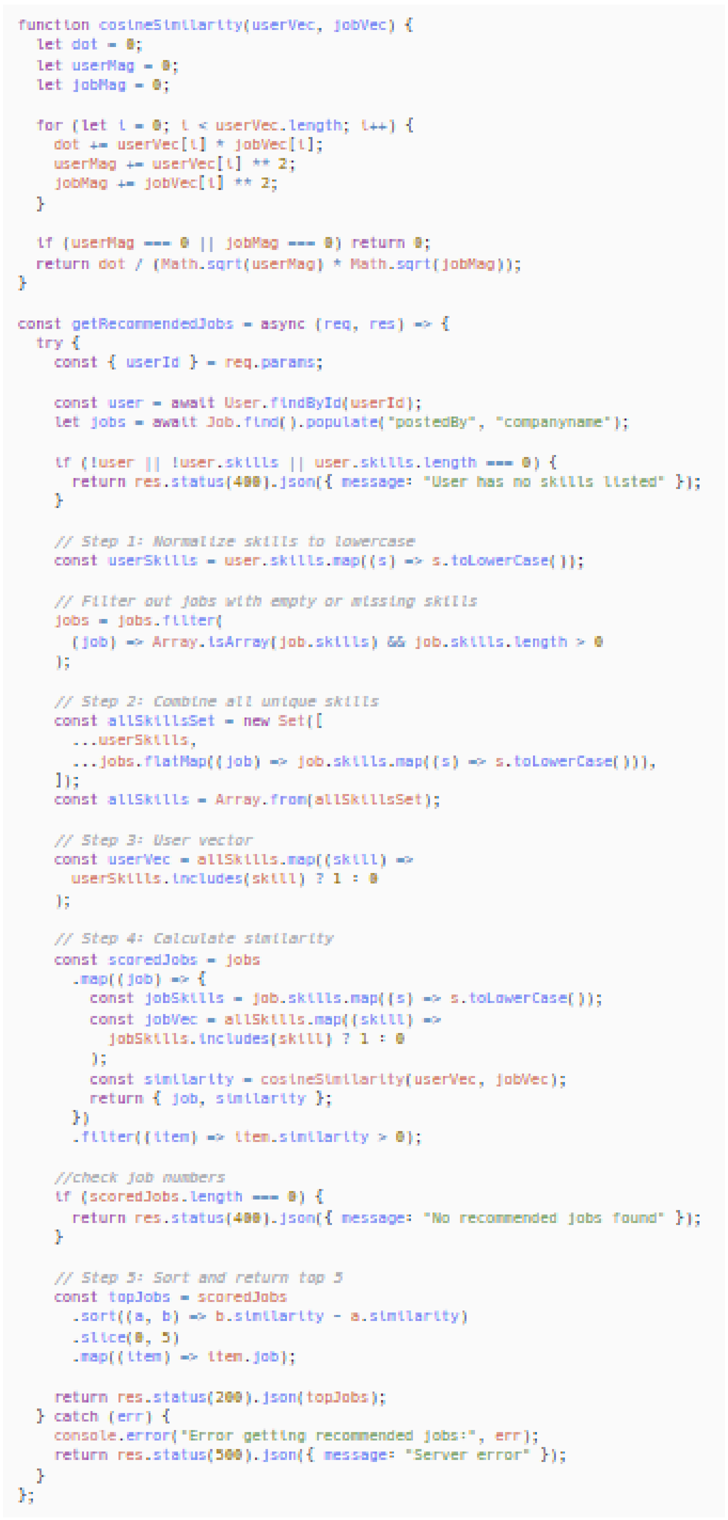
```
const putStatus = async (req, res) => {
  try {
    const { status } = req.body;
    const { appId } = req.params;

    const updated = await applicationModel.findByIdAndUpdate(
      appId,
      { status },
      {
        new: true,
      }
    );
    res.json(updated);
  } catch (err) {
    res.status(500).json({ error: err });
  }
};
```

Figure 5.7 Updating Status

Purpose:

- Updates status for job tracking.
- Recruiters can reject or shortlist candidates.
- Users can view their status of a job.

5.5 Job Recommendations Module



```
function cosineSimilarity(userVec, jobVec) {
  let dot = 0;
  let userMag = 0;
  let jobMag = 0;

  for (let i = 0; i < userVec.length; i++) {
    dot += userVec[i] * jobVec[i];
    userMag += userVec[i] ** 2;
    jobMag += jobVec[i] ** 2;
  }

  if (userMag === 0 || jobMag === 0) return 0;
  return dot / (Math.sqrt(userMag) * Math.sqrt(jobMag));
}

const getRecommendedJobs = async (req, res) => {
  try {
    const { userId } = req.params;

    const user = await User.findById(userId);
    let jobs = await Job.find().populate("postedBy", "companyname");

    if (!user || !user.skills || user.skills.length === 0) {
      return res.status(400).json({ message: "User has no skills listed" });
    }

    // Step 1: Normalize skills to lowercase
    const userSkills = user.skills.map((s) => s.toLowerCase());

    // Filter out jobs with empty or missing skills
    jobs = jobs.filter(
      (job) => Array.isArray(job.skills) && job.skills.length > 0
    );

    // Step 2: Combine all unique skills
    const allSkillsSet = new Set([
      ...userSkills,
      ...jobs.flatMap((job) => job.skills.map((s) => s.toLowerCase())),
    ]);
    const allSkills = Array.from(allSkillsSet);

    // Step 3: User vector
    const userVec = allSkills.map((skill) =>
      userSkills.includes(skill) ? 1 : 0
    );

    // Step 4: Calculate similarity
    const scoredJobs = jobs
      .map((job) => {
        const jobSkills = job.skills.map((s) => s.toLowerCase());
        const jobVec = allSkills.map((skill) =>
          jobSkills.includes(skill) ? 1 : 0
        );
        const similarity = cosineSimilarity(userVec, jobVec);
        return { job, similarity };
      })
      .filter((item) => item.similarity > 0);

    //check job numbers
    if (scoredJobs.length === 0) {
      return res.status(400).json({ message: "No recommended jobs found" });
    }

    // Step 5: Sort and return top 5
    const topJobs = scoredJobs
      .sort((a, b) => b.similarity - a.similarity)
      .slice(0, 5)
      .map((item) => item.job);

    return res.status(200).json(topJobs);
  } catch (err) {
    console.error("Error getting recommended jobs:", err);
    return res.status(500).json({ message: "Server error" });
  }
};
```

Figure 5.8 Job Recommendations Module

Purpose:

- Provides personalized job recommendations for users based on skills and filtering.
- Enhances user experience by suggesting job relevant to their interests.

Implementation:

- Retrieves the user's skills from the database based on the given userId.
- Collects all available jobs along with their associated skills and company details.
- Normalizes skills (converts to lowercase) to ensure accurate comparison between user and job skills.
- Constructs a universal skill set combining user and job skills to form a common comparison basis.
- Represents both user and job skills as binary vectors over the universal skill set.
- Calculates similarity between user and each job using the cosine similarity algorithm.
- Assigns a similarity score to each job and filters out jobs with zero similarity.
- Sorts jobs in descending order of similarity scores.
- Selects the top 5 most relevant jobs for recommendation.
- Returns the recommended jobs as the API response.

5.6 Testing

Testing is a vital phase in software development that ensures the system operates as intended and meets user requirements. It involves different types of testing, such as unit testing, which focuses on verifying individual components or modules for correctness, and system testing, which evaluates the entire system's functionality and performance as an integrated whole. These testing methods help identify defects, ensure reliability, and improve overall software quality, reducing risks before deployment.

5.6.1 Test Cases for Unit Testing

1. Unit Test For SignUp

Table 1 Test Cases for SignUp

Test ID	**Test Cases**	**Test Input**	**Expected Outcome**	**Actual Outcome**	**Test Result**
T01	Empty User input	null	Registration Failed	Registration Failed	Pass
T02	New User	fullname: Ram Kumar, email: ram@example.com, password:ram123 confirm password: ram123 address: lagankhel phone:9800000000	Registration Successful	Registration Successfull	Pass
T03	Password mismatch	fullname: Ram Kumar, email: ram@example.com, password:ram123 confirm password: ram1234 address: lagankhel phone:9800000000	Passwords do not match	Passwords do not match	Pass
T04	Existing User	fullname: Ram Kumar, email: ram@example.com, password:ram123 confirm password: ram123 address: lagankhel phone:9800000000	Email already in use	Email already in use	Pass

2. Unit Test for Login

Table 2 Unit Test for Login

Test ID	**Test Cases**	**Test Input**	**Expected Outcome**	**Actual Outcome**	**Test Result**
T05	Empty user input	null	Login Failed	Login Failed	Pass
T06	Incorrect Email	Email: 123@gmail.com Password: abc	Login Failed	Login Failed	Pass
T07	Incorrect Password	Email:test@gmail.com Password:a	Login Failed	Login Failed	Pass
T08	Correct credentials	Email: user@gmail.com Password: abc	Login Success	Login Suucess	Pass

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

The job portal system developed successfully addresses the need for an efficient, user-friendly platform for both job seekers and employers. Through its seamless user interface and robust backend, the system enables job seekers to register, create profiles, upload resumes, search and apply for jobs, and track application statuses. Employers can post job vacancies, manage applications, shortlist candidates, and communicate with potential hires, while administrators can monitor and manage overall platform activities. The system was built using modern web technologies and adheres to best practices in design and development, ensuring reliability, scalability, and security. Overall, the system provides a complete solution for job recruitment, offering convenience, transparency, and flexibility for all users

6.2 Future Recommendations

While the current system provides a solid foundation, several enhancements can be made to further its functionality and user experience:

1. **AI-Powered Job Matching**
   Implement artificial intelligence (AI) algorithms to recommend the most relevant jobs to seekers and suggest suitable candidates to employers based on skills, experience, and preferences.

2. **Mobile Application Development**
   Develop dedicated Android and iOS mobile applications to improve accessibility and provide a seamless job search and recruitment experience on the go.

3. **Advanced Search and Filtering**

   Introduce smart filters such as salary range, remote/hybrid options, company ratings, and skill-based matching for more personalized job searches.

4. **Automated Resume Screening**

   Enable employers to use AI-driven resume screening tools to quickly identify top candidates, reducing hiring time and effort.

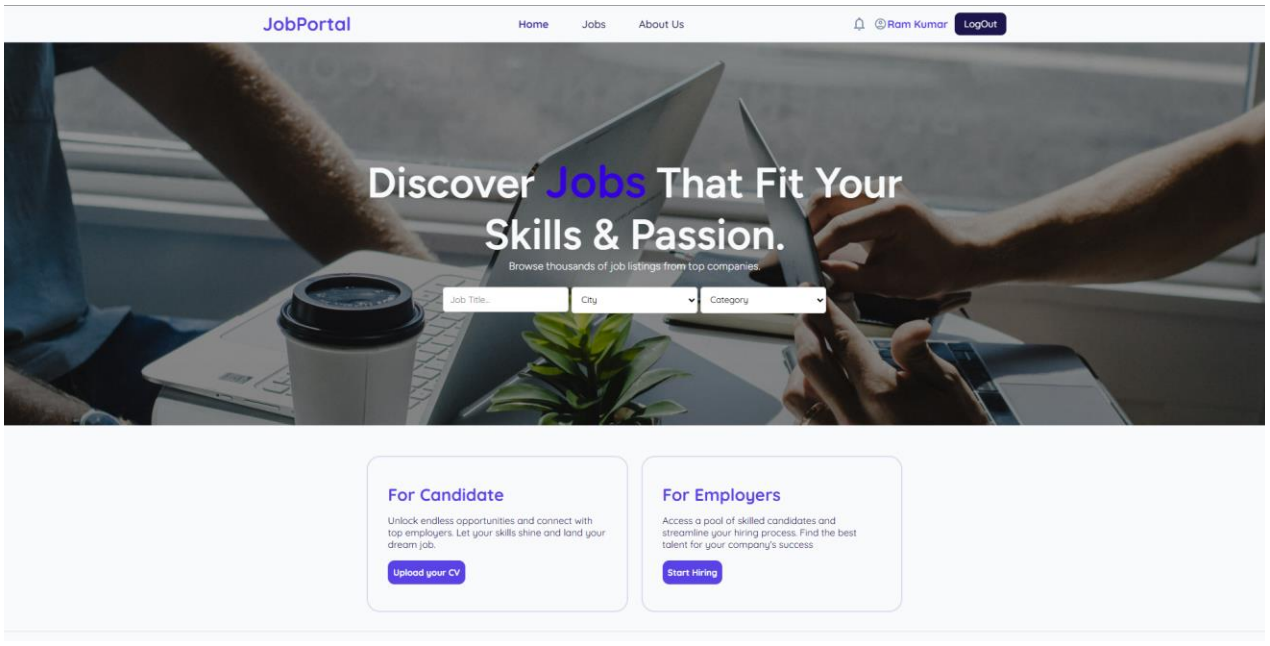
These improvements would not only enhance the user experience but also broaden the system’s reach and functionality in the competitive vacation rental market.

REFERENCES

- LinkedIn. (2022). *Hiring Trends and Insights*. Retrieved from https://www.linkedin.com/business/talent/blog
- Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
- Indeed. (2023). *Job Search and Hiring Insights*. Retrieved from https://www.indeed.com/

APPENDIX

Home Page



Discover Jobs That Fit Your Skills & Passion.

Browse thousands of job listings from top companies.

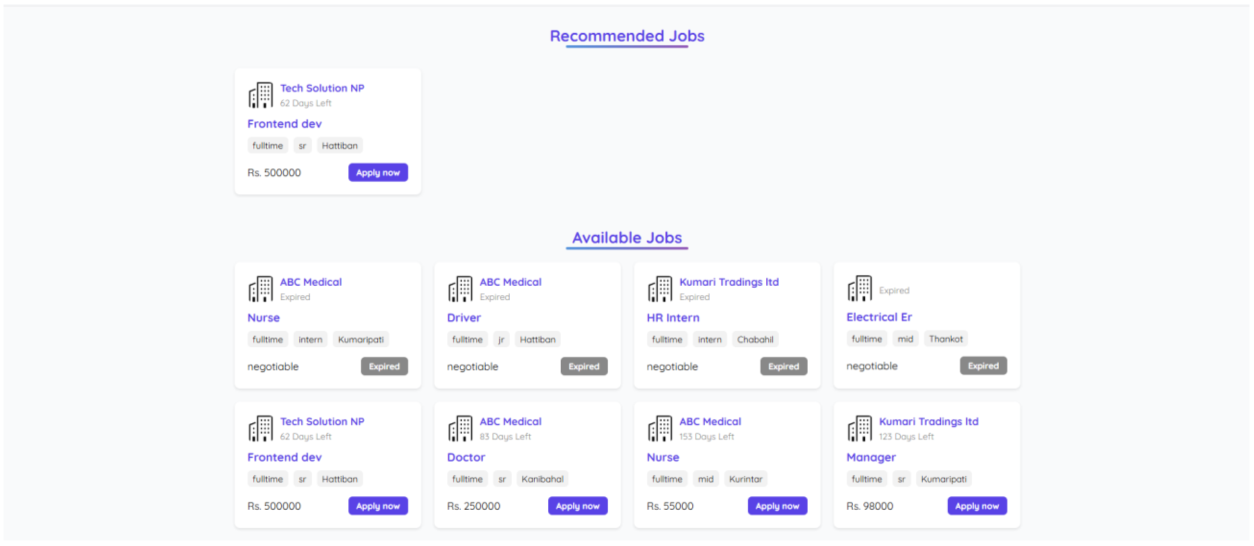
For Candidate

Unlock endless opportunities and connect with top employers. Let your skills shine and land your dream job.

For Employers

Access a pool of skilled candidates and streamline your hiring process. Find the best talent for your company's success

Recommended Jobs & Available Jobs



Recommended Jobs

Tech Solution NP — 62 Days Left — Frontend dev — fulltime, sr, Hattiban — Rs. 500000 — Apply now

Available Jobs

ABC Medical — Expired — Nurse — fulltime, intern, Kumaripati — negotiable — Expired

ABC Medical — Expired — Driver — fulltime, jr, Hattiban — negotiable — Expired

Kumari Tradings ltd — Expired — HR Intern — fulltime, intern, Chabahil — negotiable — Expired

Expired — Electrical Er — fulltime, mid, Thankot — negotiable — Expired

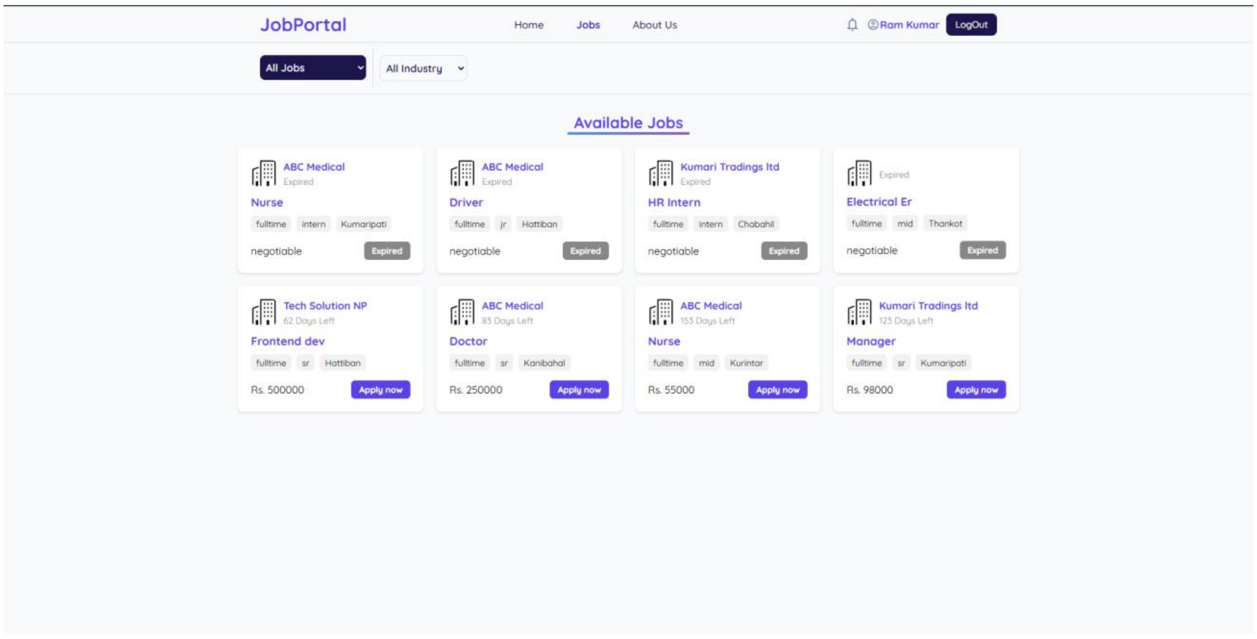
Tech Solution NP — 62 Days Left — Frontend dev — fulltime, sr, Hattiban — Rs. 500000 — Apply now

ABC Medical — 83 Days Left — Doctor — fulltime, sr, Kanibahal — Rs. 250000 — Apply now

ABC Medical — 153 Days Left — Nurse — fulltime, mid, Kurintar — Rs. 55000 — Apply now

Kumari Tradings ltd — 123 Days Left — Manager — fulltime, sr, Kumaripati — Rs. 98000 — Apply now

Jobs Page



Available Jobs

ABC Medical
Expired
Nurse
fulltime intern Kumaripati
negotiable Expired

ABC Medical
Expired
Driver
fulltime jr Hattiban
negotiable Expired

Kumari Tradings ltd
Expired
HR Intern
fulltime intern Chabahil
negotiable Expired

Expired
Electrical Er
fulltime mid Thankot
negotiable Expired

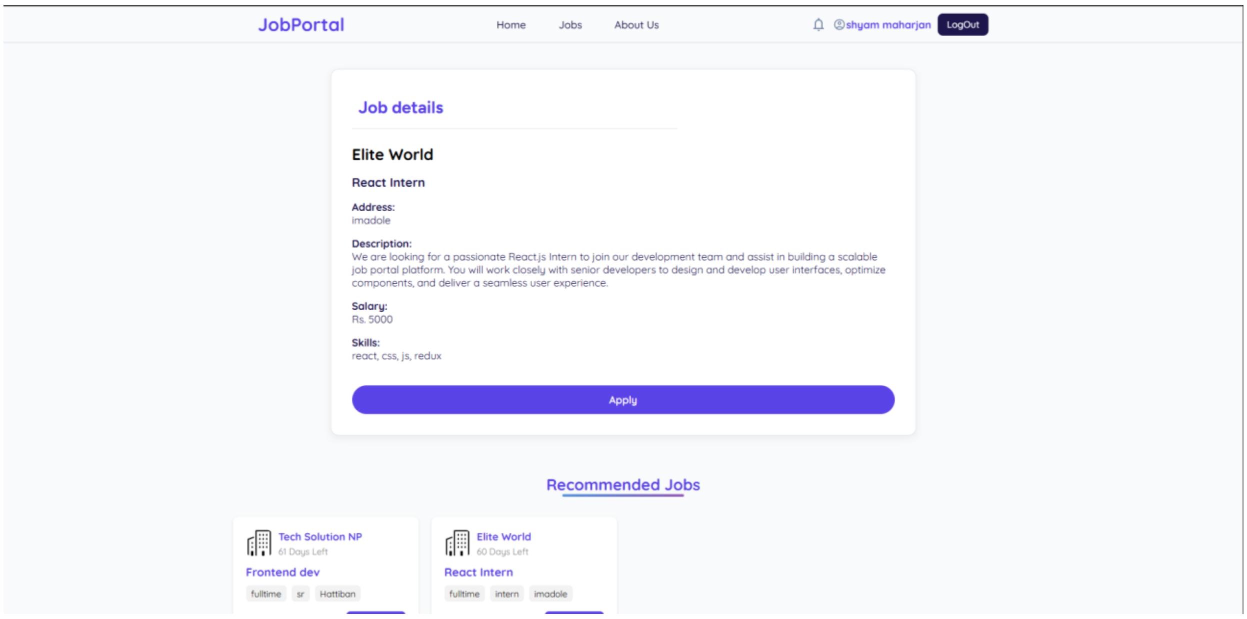
Tech Solution NP
62 Days Left
Frontend dev
fulltime sr Hattiban
Rs. 500000 Apply now

ABC Medical
83 Days Left
Doctor
fulltime sr Kanibahal
Rs. 250000 Apply now

ABC Medical
153 Days Left
Nurse
fulltime mid Kurintar
Rs. 55000 Apply now

Kumari Tradings ltd
123 Days Left
Manager
fulltime sr Kumaripati
Rs. 98000 Apply now

Job Card



Job details

Elite World

React Intern

Address:
imadole

Description:
We are looking for a passionate React.js Intern to join our development team and assist in building a scalable job portal platform. You will work closely with senior developers to design and develop user interfaces, optimize components, and deliver a seamless user experience.

Salary:
Rs. 5000

Skills:
react, css, js, redux

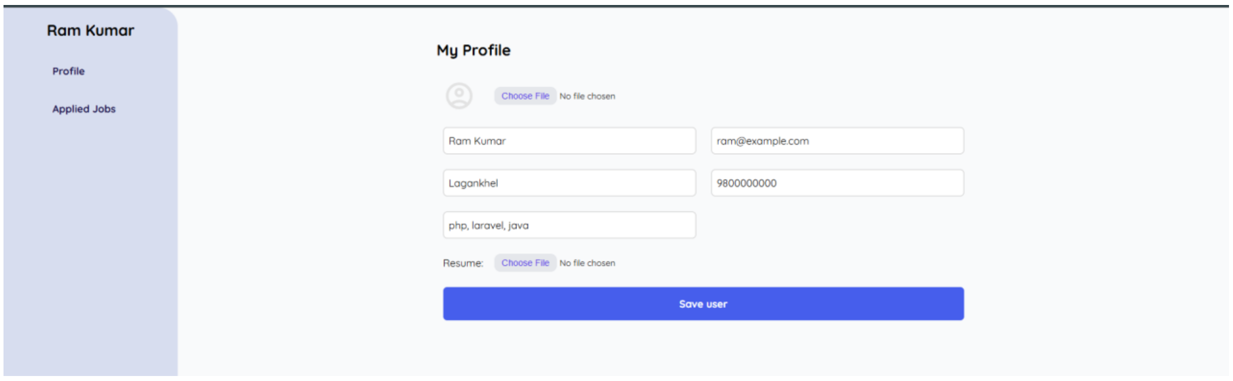
Apply

Recommended Jobs

Tech Solution NP
61 Days Left
Frontend dev
fulltime sr Hattiban

Elite World
60 Days Left
React Intern
fulltime intern imadole

Users Dashboard



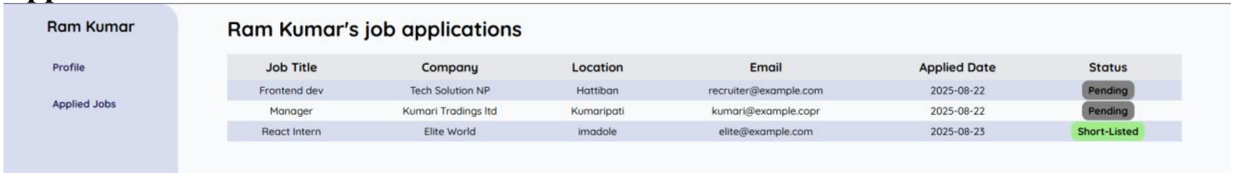
My Profile

No file chosen

Ram Kumar	ram@example.com
Lagankhel	9800000000
php, laravel, java	

Resume: No file chosen

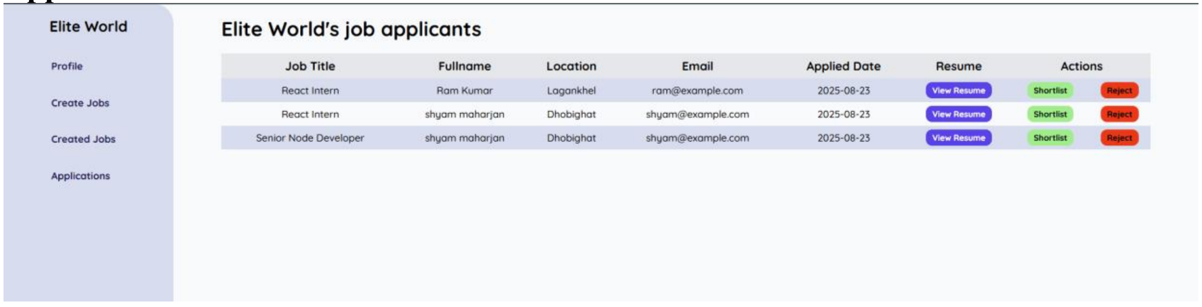
Applied Jobs



Ram Kumar's job applications

Job Title	Company	Location	Email	Applied Date	Status
Frontend dev	Tech Solution NP	Hattiban	recruiter@example.com	2025-08-22	Pending
Manager	Kumari Tradings ltd	Kumaripati	kumari@example.copr	2025-08-22	Pending
React Intern	Elite World	imadole	elite@example.com	2025-08-23	Short-Listed

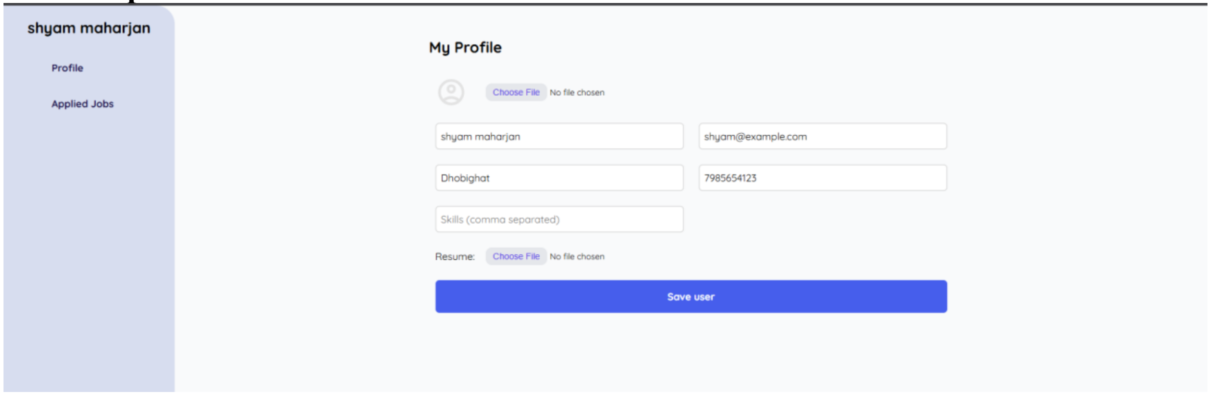
Applicants



Elite World's job applicants

Job Title	Fullname	Location	Email	Applied Date	Resume	Actions	
React Intern	Ram Kumar	Lagankhel	ram@example.com	2025-08-23	View Resume	Shortlist	Reject
React Intern	shyam maharjan	Dhobighat	shyam@example.com	2025-08-23	View Resume	Shortlist	Reject
Senior Node Developer	shyam maharjan	Dhobighat	shyam@example.com	2025-08-23	View Resume	Shortlist	Reject

Profile Update



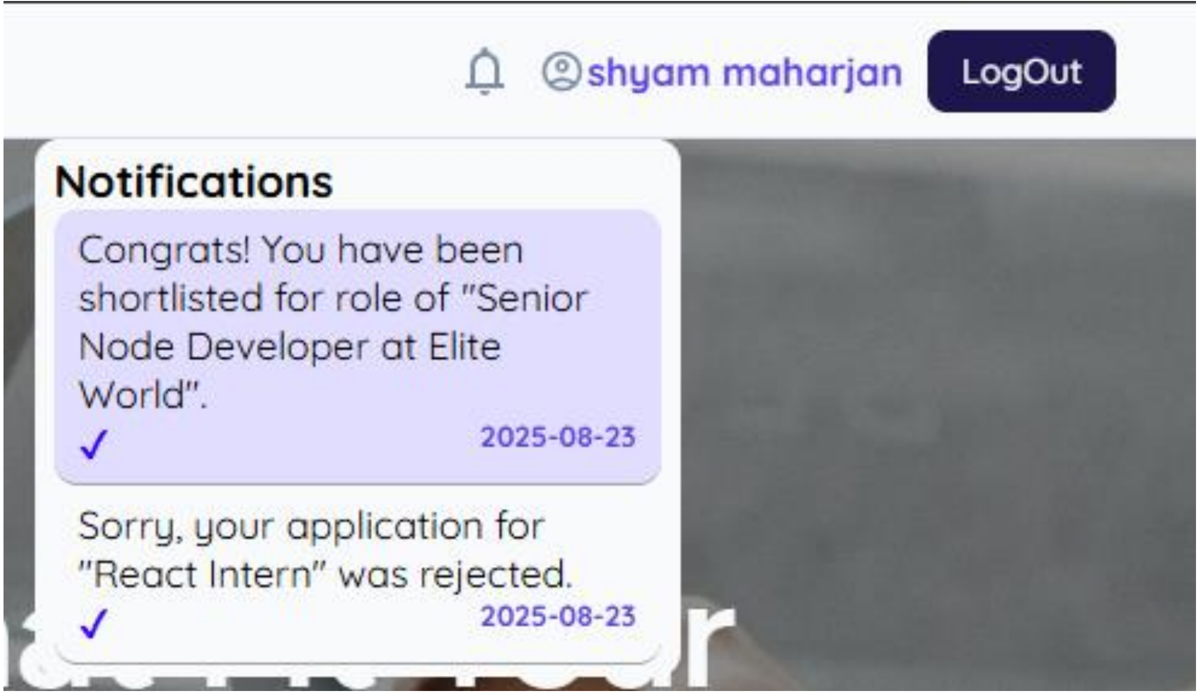
My Profile

No file chosen

shyam maharjan	shyam@example.com
Dhobighat	7985654123
Skills (comma separated)	

Resume: No file chosen

Notifications

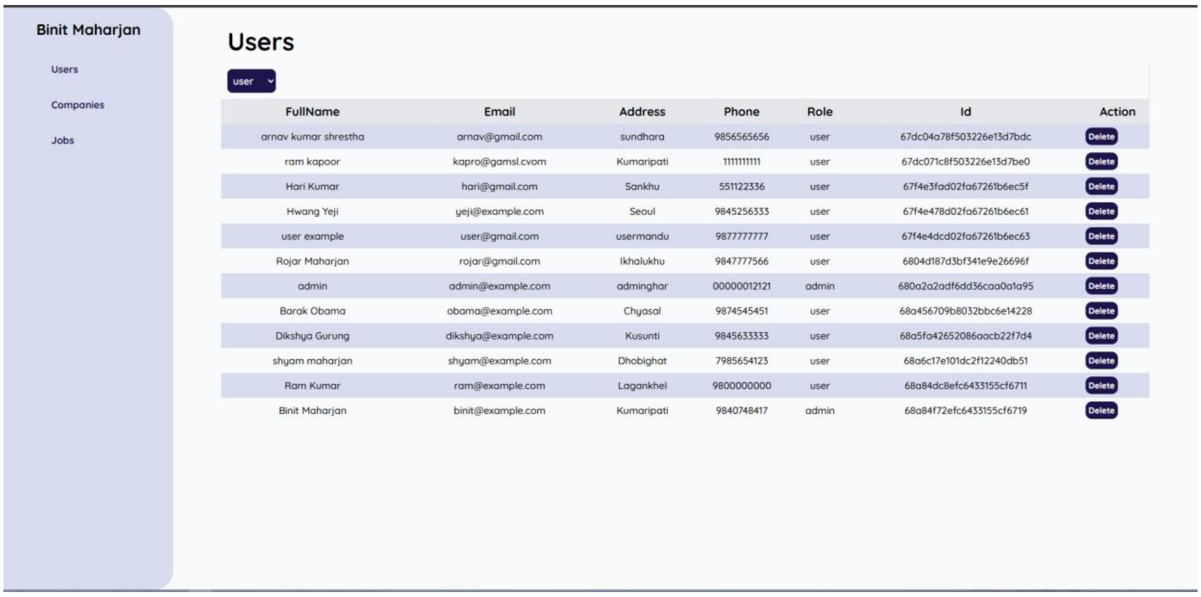


Notifications

Congrats! You have been shortlisted for role of "Senior Node Developer at Elite World".
✓ 2025-08-23

Sorry, your application for "React Intern" was rejected.
✓ 2025-08-23

Admin Dashboard



Users

FullName	Email	Address	Phone	Role	Id	Action
arnav kumar shrestha	arnav@gmail.com	sundhara	9856565656	user	67dc04a78f503226e13d7bdc	Delete
ram kapoor	kapro@gamsl.cvom	Kumaripati	1111111111	user	67dc071c8f503226e13d7be0	Delete
Hari Kumar	hari@gmail.com	Sankhu	551122336	user	67f4e3fad02fa67261b6ec5f	Delete
Hwang Yeji	yeji@example.com	Seoul	9845256333	user	67f4e478d02fa67261b6ec61	Delete
user example	user@gmail.com	usermandu	9877777777	user	67f4e4dcd02fa67261b6ec63	Delete
Rojar Maharjan	rojar@gmail.com	Ikhalukhu	9847777566	user	6804d187d3bf341e9e26696f	Delete
admin	admin@example.com	adminghar	00000012121	admin	680a2a2adf6dd36caa0a1a95	Delete
Barak Obama	obama@example.com	Chyasal	9874545451	user	68a456709b8032bbc6e14228	Delete
Dikshya Gurung	dikshya@example.com	Kusunti	9845633333	user	68a5fa42652086aacb22f7d4	Delete
shyam maharjan	shyam@example.com	Dhobighat	7985654123	user	68a6c17e101dc2f12240db51	Delete
Ram Kumar	ram@example.com	Lagankhel	9800000000	user	68a84dc8efc6433155cf6711	Delete
Binit Maharjan	binit@example.com	Kumaripati	9840748417	admin	68a84f72efc6433155cf6719	Delete