# SMART JOB PORTAL USING MERN STACK

**A PROJECT REPORT**

***Submitted by***


**JYOTIPRIYA DAS**

**SAYAN PAL**

**INDRAJIT SAHU**

**HASANOOR ZAMAN**


***Supervised by***

**Prof. Sarasij Majumdar**

***in partial fulfillment for the award of the degree***

***of***

**MASTER OF COMPUTER APPLICATION**

**IN**

COMPUTER APPLICATION

Year: 2024-2025



**TECHNO INTERNATIONAL NEW TOWN**

**Rajarhat, New Town,**
**Kolkata – 700156,**
**West Bengal, India**

# BONAFIDE CERTIFICATE

Certified that this project report " Smart Job Portal using MERN Stack " is the bonafied work of " **JYOTIPRIYA DAS, SAYAN PAL, INDRAJIT SAHU, HASANOOR ZAMAN** ", who carried out the project work under my supervision.

---

**SIGNATURE**

Dr. Soma Chaterjee

**HEAD OF THE DEPARTMENT**

**MASTERS OF COMPUTER APPLICATION**

Techno International New Town

Rajarhat, New Town,

Kolkata – 700156,

West Bengal, India

---

**SIGNATURE**

Prof. Sarasij Majumdar

**PROJECT GUIDE & SUPERVISOR**

**MASTERS OF COMPUTER APPLICATION**

Techno International New Town

Rajarhat, New Town,

Kolkata – 700156,

West Bengal, India

---

**SIGNATURE**

**External Examiner**

# ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to everyone who supported and guided us throughout the completion of our project titled “Smart Job Portal Using Mern Stack.” First and foremost, we would like to convey our sincere thanks to our respected guide, Prof.Sarasij Majumdar and Mentor Prof.SoumenSantra for his valuable guidance, constant encouragement, and insightful suggestions that helped us at every stage of this project. His expertise and support have been a great source of inspiration and learning for us. We would also like to extend our gratitude to our college, TECHNO INTERNATIONAL NEW TOWN, for offering us the academic environment and resources necessary to carry out our research work successfully. It is such an interesting project to work with and without the continuous academic support we couldn’t have done it. Our heartfelt thanks also go to our friends, classmates, and family members for their continuous motivation, patience, and moral support during this project. Their encouragement kept us focused and dedicated throughout the process. Finally, we would like to express our appreciation to all those who directly or indirectly contributed to the successful completion of this project. This work would not have been possible without their help, support, and guidance.

Date: _____

Students:

1.

2.

3.

4.

Guide : _____

(Prof.Sarasij Majumdar)

Mentor : _____

(Prof.Soumen Santra)

# TABLE OF CONTENTS

# ABSTRACT

In today's competitive world, employment has become a critical issue for job seekers and recruiters alike. To address this challenge, we present a Job Portal system designed to the recruitment process by providing an interactive platform where job seekers and recruiters can connect efficiently. This project is a full-stack web application developed using **Node.js, Express.js, MongoDB, and React.js**, aimed at automating the job search and hiring process.
The portal enables **job seekers** to register, create profiles, upload resumes, and search or apply for jobs using filters such as job category, location, and experience level. On the other hand, **recruiters** can register, post job vacancies, manage applicants, and shortlist candidates based on specific requirements. The admin panel ensures that the platform remains organized, secure, and spam-free by providing administrative control over users, listings, and reports.

To enhance security and user experience, the system includes **authentication and authorization** using **JWT** and **NextAuth**, ensuring that data is protected and access is role-based (admin, recruiter, or job seeker). The **RESTful API architecture** provides a smooth backend structure that allows seamless data flow and integration with the frontend React components. The portal's responsive design ensures compatibility across various devices and screen sizes, delivering a modern UI and excellent usability.

This project's **key features** include User authentication and login system**,** Job seeker and recruiter role differentiation**,** Resume upload and job application tracker**,** Job posting, editing, and deletion options for recruiters**,** Admin panel for monitoring and moderation**,** Real-time search and filtering capabilities**,** End-to-End communication between job seekers and recruiters.
This system was created to reduce manual efforts and bridge the gap between skilled candidates and companies. It demonstrates real-world applicability of software engineering principles including **CRUD operations, REST APIs, MVC architecture, NoSQL database schema, responsive frontend design**, and **secure authentication mechanisms**.

The Job Portal project offers a scalable and reliable digital solution to the employment sector and can be extended further with features like AI-based job suggestions, interview scheduling, and resume parsing. It showcases the practical implementation of web development skills and project planning, making it a valuable asset for academic portfolios and professional growth.

# PROJECT REPORT

## SMART JOB PORTAL USING MERN STACK

---

- # CHAPTER 1

## INTRODUCTION

---

### 1.1 General

In today's world, finding jobs and hiring employees has moved online. Instead of using newspapers or handing out paper resumes, people now use websites called job portals. These portals make it easier for job seekers to find opportunities and for recruiters to hire the right candidates. However, many of the current job portals are complicated, expensive, or not well-suited for local or specific job needs. To solve this, we created a Job Portal web application. It is an easy-to-use platform where job seekers and employers can connect quickly and smoothly. This system is made to make the hiring process easier and faster. It helps job seekers and recruiters connect without much trouble. The platform has useful features for both types of users, a strong backend to handle data, and a user-friendly design. It shows how full-stack web development can be used in a real-world project.

The primary goal of this project is to design and develop a web-based job portal that allows job seekers to register, create and update profiles, and search/apply for jobs, enables recruiters to post job openings, view applications, and manage their listings, ensures that administrators can monitor the system, manage users, and prevent abuse, provides an intuitive, responsive UI that can be accessed from mobile, tablet, or desktop, offers a secure and scalable architecture to handle user data and authentication effectively.

Tools & Technologies Used for Frontend: React.js, Custom CSS, Axios. Backend: Node.js, Express.js, For database: MongoDB (NoSQL), For authentication: JWT, NextAuth, For Hosting & Cloud Services: Render/ Netlify/ Hostinger/ Microsoft Azure/ Amazon Web Services (AWS) (AWS can be used for services such as EC2 for hosting, S3 for files storage, and Route 53 for domain management), For Development Tools: Postman, Git, VS Code, MongoDB compass, Version control - Github.

The application uses **MVC (Model-View-Controller)** architecture. Where Model specifies Handling database schema and interaction using Mongoose for MongoDB, View specifies Developeing using React.js, rendering dynamic pages based upon user type and actions, Controller specifies Business logic written in Node.js manages API endpoints and server-side processing.The system supports **RESTful APIs** for seamless communication between frontend and backend, allowing the decoupled development of UI and logic layers.

The Job Portal system is divided into three main modules: Job Seeker, Recruiter, and Admin Panel, each offering a range of useful features. In the Job Seeker module, users can register and log in to the portal, create their profiles, and upload their resumes. They can browse and search for jobs using filters such as location, job category, and keywords. Once they apply for jobs, their application history is saved, allowing them to track their submissions. Additionally, users can easily reset or update their passwords whenever needed. The Recruiter module allows employers to register and log in as recruiters. They can create new job posts, as well as update or delete existing listings. Recruiters also have access to an applicant tracking system where

they can view all received applications, update statuses, and manage the hiring process through a dedicated recruiter dashboard. Finally, the Admin Panel gives system administrators complete control over the platform. Admins can manage all users, including both job seekers and recruiters, and take necessary actions such as deleting inappropriate job listings. They can monitor overall system usage and handle user complaints or reports to ensure the platform remains safe, clean, and functional.

The Job Portal system includes several essential functionalities that ensure a smooth and secure experience for all users. One of the core features is Authentication and Authorization, implemented using a JWT-based token system, which protects user data and ensures that only authorized users can access specific parts of the application based on their roles (job seeker, recruiter, or admin). This application is designed with a fully responsive layout using Tailwind CSS, making it mobile-friendly and easily accessible across various devices and screen sizes. The database design is well-structured, with collections such as Users, Jobs, Applications, and Admins. These collections are connected using ObjectId references, enabling efficient data management and retrieval. A powerful search and filter functionality allows job seekers to find relevant job listings quickly. They can filter jobs by keyword, location, company name, and job type (such as full-time, part-time, or internship), helping them narrow down their options with ease. Additionally, the platform features an application tracker, where job seekers can monitor the jobs they have applied for and check the current status of each application. This provides a transparent and organized experience, encouraging user engagement and making the job application process more efficient.

The Job Portal system has strong potential for future enhancement to make the platform even more intelligent, interactive, and user-friendly. One of the planned improvements is the integration of a real-time chat system between recruiters and job seekers, allowing instant communication and quicker decision-making during the hiring process. Additionally, the platform can be enhanced with AI-powered job recommendations, which would analyze users' skills, experience, and preferences to suggest the most suitable job opportunities. Another key feature for future development is a resume parser, which can automatically extract information from uploaded resumes and use it to auto-fill the job seeker's profile, saving time and reducing manual input errors. The system can also include interview scheduling with calendar synchronization, enabling recruiters to propose interview times and job seekers to accept or reschedule directly within the portal. Lastly, adding push and email notifications for job alerts will keep users informed about new job postings, application updates, and important announcements. These improvements aim to create a more efficient and personalized job search experience, making the platform more valuable for both recruiters and candidates.

This Job Portal Project not only bridges the gap between recruiters and candidates but also showcases practical knowledge of web development using the **MERN stack**. By implementing a modular, secure, and user-friendly platform, this system simplifies the employment process and provides a reliable tool for job search and recruitment. It is a strong representation of how modern web technologies can be applied to solve real-world problems, and it opens the door for future improvements and scaling in professional environments.

---

## 1.2 Background of Online Recruitment Systems

Online recruitment systems emerged as a solution to overcome the limitations of traditional hiring processes. Early job portals focused primarily on job listings and resume uploads. Over time, these platforms evolved to include advanced features such as applicant tracking, recruiter dashboards, and automated notifications.

Despite these advancements, many platforms still lack real-time communication facilities and rely heavily on external tools for interviews and messaging. Additionally, privacy issues and

restricted access to communication channels remain major concerns for both recruiters and job seekers.

---

## 1.3 Motivation for the Project

The motivation behind developing this project arises from the need to create a **direct, secure, and transparent recruitment platform** that:

- Reduces dependency on third-party services
- Enables instant communication
- Improves hiring efficiency
- Provides better control over user data

The **MERN Stack** was chosen for development due to its flexibility, performance, and suitability for building modern full-stack web applications.

---

## 1.4 Problem Statement

Traditional job portals often involve intermediaries that control communication between recruiters and job seekers. This results in delayed responses, lack of transparency, and privacy concerns. Recruiters face difficulties in managing large volumes of applications, while job seekers struggle to receive timely updates.

There is a need for a **web-based job portal** that allows **direct interaction**, supports real-time communication, and efficiently manages recruitment activities without third-party interference.

---

## 1.5 Objectives of the Project

The main objectives of the project are:

- To develop a web-based job portal using MERN stack
- To provide role-based access for job seekers, recruiters, and admin
- To enable secure 1-on-1 communication
- To support job posting and application tracking
- To design a responsive user interface using Custom CSS
- To ensure efficient data management using MongoDB

---

## 1.6 Scope of the Project

**Included Features**

- User registration and authentication
- Profile and resume management
- Job posting and search
- Application tracking
- Real-time chat between users
- Video interview functionality
- Admin panel for system monitoring

**Excluded Features**

- Cloud deployment
- Payment gateways
- Mobile application

---

## 1.7 Organization of the Report

- **Chapter 1** introduces the project and its objectives
- **Chapter 2** reviews existing systems and technologies
- **Chapter 3**analyzes system requirements
- **Chapter 4** explains system design
- **Chapter 5** discusses tools and technologies
- **Chapter 6** describes implementation
- **Chapter 7** covers testing
- **Chapter 8** presents results
- **Chapter 9** concludes the project

# CHAPTER 2

# LITERATURE REVIEW

---

## 2.1 Introduction

A literature review helps in understanding how similar systems have been developed in the past and what technologies are commonly used in the chosen domain. In the case of online recruitment systems, many job portals already exist, but they do not fully solve problems related to communication delays, privacy, and transparency.

This chapter reviews existing job portals, studies relevant web development technologies, and identifies the limitations of current systems. The analysis of previous work helps in justifying the need for the proposed **Smart Job Portal using MERN Stack**.

---

## 2.2 Existing Online Job Portals

Several job portals are currently used by job seekers and recruiters. Although these platforms provide basic recruitment features, they also introduce certain limitations.

---

### 2.2.1 LinkedIn

LinkedIn is widely used for professional networking and job searching. It allows recruiters to post job openings and search for potential candidates, while job seekers can apply using their online profiles.

**Limitations:**

- Direct communication is often restricted without premium access
- Platform focuses more on networking than recruitment
- User data is heavily analyzed by third-party services

---

### 2.2.2 Naukri.com

Naukri is one of the most popular job portals in India. It provides job listings, resume uploads, and recruiter dashboards.

**Limitations:**

- Communication between recruiters and candidates is limited
- Many features require paid subscriptions
- No built-in real-time chat or interview system

### 2.2.3 Indeed

Indeed collects job postings from multiple sources and allows users to apply through the platform.

**Limitations:**

- Limited personalization
- No direct recruiter–candidate interaction
- Dependence on external job sources

---

## 2.3 Studies on Recruitment Management Systems

Previous studies on recruitment systems emphasize the importance of:

- Efficient application management
- Secure storage of user data
- Fast communication between users

However, many systems focus mainly on data collection and filtering, while ignoring direct human interaction. This often leads to longer hiring cycles and reduced user satisfaction.

---

## 2.4 MERN Stack in Modern Web Applications

The MERN stack is a combination of four JavaScript-based technologies: MongoDB, Express.js, React.js, and Node.js. It is widely used for developing full-stack web applications due to its flexibility and performance.

- **React.js** helps in building dynamic and responsive user interfaces
- **Node.js** allows handling multiple user requests efficiently
- **Express.js** simplifies backend development
- **MongoDB** provides flexible data storage

Research and industry practices show that MERN stack applications are suitable for systems that require real-time updates and smooth user interaction.

---

## 2.5 Real-Time Communication in Web Systems

Modern web applications increasingly rely on real-time communication features such as instant messaging and video interaction. Technologies like WebSockets and WebRTC are commonly used to achieve this functionality.

Studies indicate that recruitment platforms with real-time communication reduce response delays and improve the overall hiring experience.

## 2.6 Identified Gaps in Existing Systems

From the literature review, the following gaps were identified:

- Lack of private and direct communication
- Absence of built-in chat and interview features
- Dependence on third-party platforms
- Limited transparency in application tracking

These limitations highlight the need for a more interactive and secure recruitment platform.

---

## 2.7 Summary

The literature review shows that although many job portals exist, they do not fully address the needs of modern recruitment. The lack of direct communication and privacy concerns remain major challenges. These gaps justify the development of the proposed **Smart Job Portal using MERN Stack**, which focuses on transparency, real-time interaction, and user control.

# CHAPTER 3

# SYSTEM ANALYSIS

---

## 3.1 Introduction

System analysis is the phase where the actual problem is studied in detail before designing and developing the software. It focuses on understanding how the current recruitment process works, what difficulties users face, and what improvements are required. This chapter explains the existing system, identifies its limitations, and defines the functional and non-functional requirements of the proposed Smart Job Portal.

The purpose of this analysis is to ensure that the final system is practical, user-oriented, and suitable for real-world usage.

---

## 3.2 Analysis of the Existing System

In the existing recruitment process, job seekers and recruiters mainly depend on third-party job portals. These portals act as intermediaries and control most aspects of the hiring workflow, including communication and data visibility.

**Problems in the Existing System**

- Communication between recruiters and job seekers is often delayed
- Direct interaction is limited or restricted
- Many important features require paid subscriptions
- User data privacy is not fully under user control
- No integrated chat or video interview system

Due to these issues, the hiring process becomes slow, inefficient, and less transparent.

---

## 3.3 Proposed System

The proposed system is a **Smart Job Portal using MERN Stack** that enables direct interaction between job seekers and recruiters. The system removes unnecessary intermediaries and provides built-in communication tools to improve the recruitment experience.

**Key Characteristics of the Proposed System**

- Web-based application
- Role-based access (Job Seeker, Recruiter, Admin)
- Direct 1-on-1 communication
- Real-time chat and video interview support
- Simple and responsive user interface
- Localhost or college server deployment

**1. The Core Workflow Phases**

The flow is generally divided into three major phases:

- **Authentication Phase:** The entry point where the system determines if the user is a guest, a returning user, or a new registrant.
- **Role-Based Redirection:** A critical decision diamond where the system checks the `user_role` (Job Seeker vs. Recruiter) to load the appropriate dashboard.
- **Functional Processing:** The actual tasks performed, such as job browsing, applying, or posting a job.

**2. Key Components to Highlight**

- **Start/End Points:** Represented by ovals, these show the browser session initiation and the logout/session termination.
- **Decision Diamonds:**
  - *Is Authenticated?* (If No -> Redirect to Login).
  - *Role Check?* (If Seeker -> Show Jobs; If Recruiter -> Show Applicant Pool).
  - *Application Success?* (Validation check for resume upload).
- **Process Rectangles:** These represent the MERN stack actions, such as "Fetch Jobs from MongoDB" or "Generate Video Call Link via WebRTC."

**3. Detailed Step-by-Step Description**

You can use this list for your "Description of Flow Chart" section:

1. **User Access:** User hits the landing page (React Frontend).
2. **Login/Signup:** User enters credentials. The system communicates with the Node.js backend to verify data in MongoDB.
3. **Dashboard Loading:** Upon successful JWT verification, the user is redirected based on their profile type.
4. **Job Seeker Path:** * Search/Filter jobs.
   - View Job Details.
   - Submit Application (Triggers a state change in the database).
5. **Recruiter Path:**
   - Post Job (Adds a new document to the Jobs collection).
   - Manage Applications (View seeker profiles).
   - Initiate Chat/Video (Opens a Socket.io connection).
6. **Termination:** The user logs out, clearing the local storage token.

**4. Technical Significance for Documentation**

- **Data Integrity:** Mention that every process step involving "Saving" or "Updating" involves **Mongoose schemas** to ensure data consistency.
- **State Management:** Explain that the flow is managed on the frontend using **Redux or React Context API** to keep the user's progress visible across different pages without refreshing.

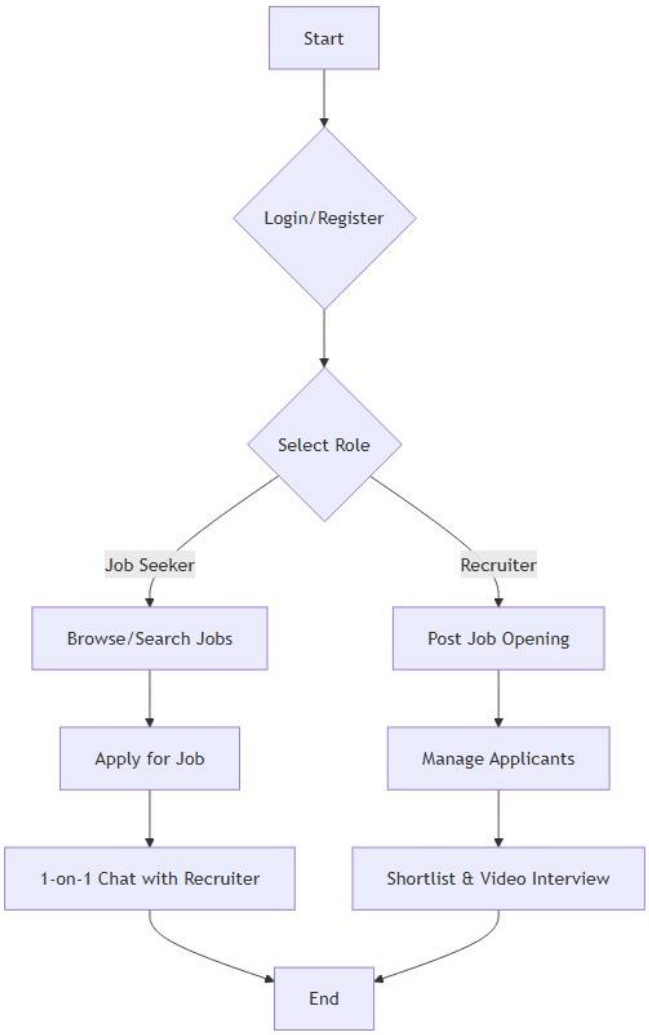


**Figure3.1 : Flow Chart**

---

## 3.4 Functional Requirements

Functional requirements describe the actions that the system must perform.

### 3.4.1 Job Seeker Requirements

- User registration and login
- Profile creation and update
- Resume upload and management
- Job search and job application
- Direct chat with recruiters
- Participation in video interviews

### 3.4.2 Recruiter Requirements

- Secure login and authentication
- Job posting and editing
- Viewing applicant details
- Shortlisting candidates

- Chat communication with job seekers
- Conducting video interviews

### 3.4.3 Admin Requirements

- User management
- Monitoring job postings
- Handling misuse or violations
- Overall system supervision

---

## 3.5 Non-Functional Requirements

Non-functional requirements define how well the system performs.

- **Security:** Secure login and role-based access
- **Performance:** Fast response time for user requests
- **Usability:** Simple and easy-to-use interface
- **Scalability:** Ability to handle increasing users
- **Reliability:** Stable operation with minimal errors
- **Maintainability:** Easy to update and modify

---

## 3.6 Use Case Analysis

### Actors

- Job Seeker
- Recruiter
- Admin

### Major Use Cases

- User Registration and Login
- Job Posting
- Job Search and Application
- Resume Management
- Chat Communication
- Video Interview
- Admin Monitoring

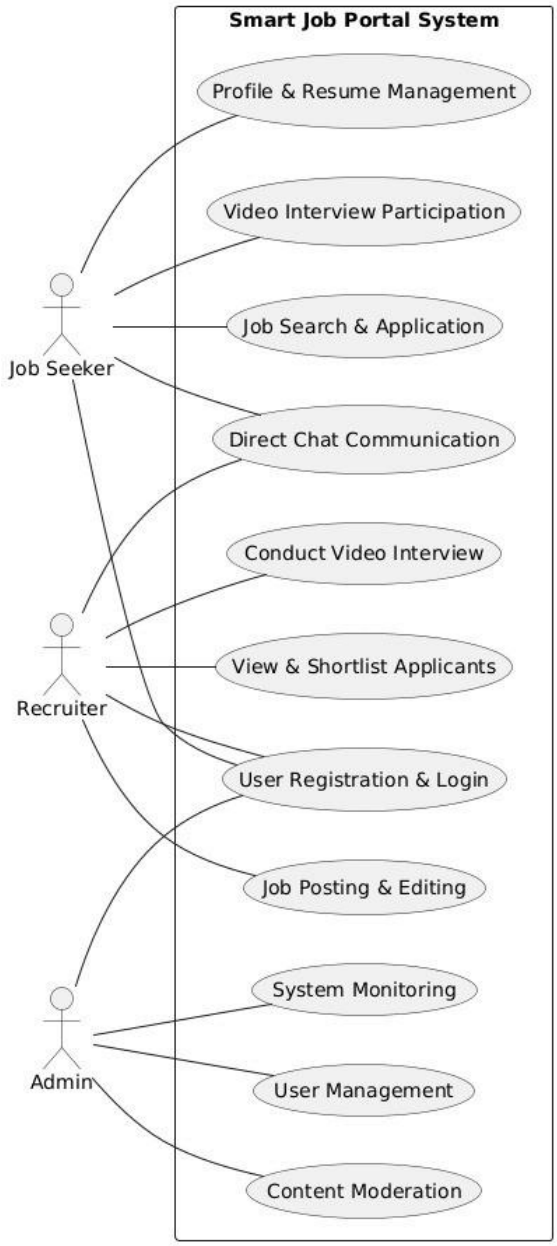


**Figure3.2 : Use Case Diagram**

## 3.7 Use Case Description

- A **Job Seeker** can register, log in, search for jobs, apply for jobs, chat with recruiters, and attend interviews.
- A **Recruiter** can log in, post jobs, view applications, communicate with candidates, and conduct interviews.
- The **Admin** oversees all activities and manages users and content.

## 3.8 Feasibility Study

### 3.8.1 Technical Feasibility

The system is technically feasible as it uses commonly available technologies such as React.js, Node.js, Express.js, and MongoDB, which are well-supported and widely used.

### 3.8.2 Economic Feasibility

The project uses open-source tools, making development cost-effective. No expensive hardware or licensed software is required.

### 3.8.3 Operational Feasibility

The system is easy to operate and does not require special training for users, making it suitable for real-world deployment.

---

## 3.9 Requirement Summary Table

| Category | Description |
|---|---|
| Authentication | Secure role-based login |
| Job Management | Posting, searching, applying |
| Communication | Chat and video interview |
| Database | MongoDB |
| Interface | React + Custom CSS |
| Deployment | Localhost / College Server |

---

## 3.10 Conclusion

System analysis confirms that the existing recruitment systems have several limitations related to communication, transparency, and privacy. The proposed Smart Job Portal effectively addresses these issues and provides a strong foundation for system design and implementation.

# CHAPTER 4

# SYSTEM DESIGN

---

## 4.1 Introduction

System design converts the requirements identified during system analysis into a structured plan for implementation. This chapter explains how the Smart Job Portal is architected, how different modules interact with each other, and how data flows through the system. The design focuses on simplicity, modularity, security, and ease of maintenance.

The system follows a layered architecture that separates the user interface, application logic, and database, ensuring better performance and scalability.

---

## 4.2 Overall System Architecture

The Smart Job Portal follows a **three-tier architecture**:

1. **Presentation Layer (Frontend)**
   - Developed using React.js
   - Styled using Custom CSS with Flexbox and Media Queries
   - Provides interfaces for Job Seekers, Recruiters, and Admin
2. **Application Layer (Backend)**
   - Built using Node.js and Express.js
   - Handles business logic, authentication, APIs, and communication
   - Manages chat and video interview signaling
3. **Data Layer (Database)**
   - MongoDB used for storing user data, jobs, applications, and messages

**Architecture Flow**

User → React Frontend → Express Backend → MongoDB → Response to User

This separation ensures better control over each part of the system and simplifies debugging and future upgrades.

---

## 4.3 Module Design

The system is divided into independent modules. Each module performs a specific task and communicates with other modules through well-defined interfaces.

---

### 4.3.1 Authentication Module

This module manages user registration and login.

Functions:

- User registration with role selection
- Secure login using encrypted passwords
- Role-based access control
- Session handling using tokens

This module ensures that only authorized users can access system features.

---

### 4.3.2 Profile Management Module

This module allows users to manage personal and professional information.

Functions:

- Job seeker profile creation and update
- Resume upload and modification
- Recruiter profile management
- Viewing applicant profiles

---

### 4.3.3 Job Management Module

This module handles job-related activities.

Functions:

- Recruiters can post, edit, and delete job listings
- Job seekers can search and filter jobs
- Job application submission
- Application status tracking

---

### 4.3.4 Communication Module (Chat)

This module enables direct communication between recruiters and job seekers.

Functions:

- One-to-one private messaging
- Real-time message delivery
- Message storage in database

This module removes dependency on external communication platforms.

---

#### 4.3.5 Video Interview Module

This module supports online interviews.

Functions:

- Real-time video and audio communication
- Peer-to-peer interaction
- Used during interview scheduling

The module improves hiring efficiency by enabling remote interviews.

---

#### 4.3.6 Admin Module

The admin module is used for monitoring and control.

Functions:

- User management
- Job listing moderation
- System monitoring
- Handling misuse or violations

---

## 4.4 Database Design

MongoDB is used as the database due to its flexibility and ability to store structured and semi-structured data.

**Major Collections**

- **Users** – Stores login credentials and roles
- **Profiles** – Stores user profile details
- **Jobs** – Stores job postings
- **Applications** – Links job seekers to jobs
- **Messages** – Stores chat data

---

## 4.5 Entity Relationship (ER) Diagram – Description

- One user has one profile
- One recruiter can post multiple jobs
- One job can receive multiple applications
- One application connects a job seeker with a job
- Messages are exchanged between recruiters and job seekers

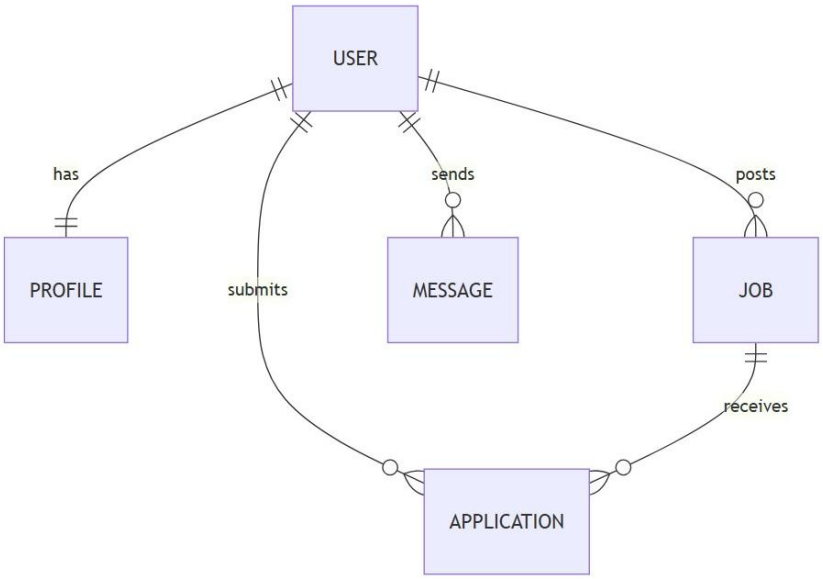


**Figure4.1 : Entity Relationship Diagram (ERD)**

---

## 4.6 Data Flow Diagram (DFD)

### Level 0 (Context Diagram)

- Users interact with the Smart Job Portal
- The system communicates with the database

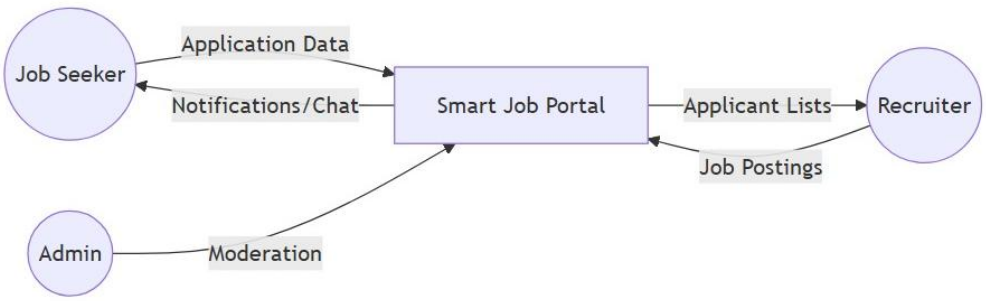


**Figure4.2.1 : DFD (level 0) Diagram**

### Level 1 DFD

- Login and authentication process
- Job posting and job search process
- Application submission process
- Chat and video communication process

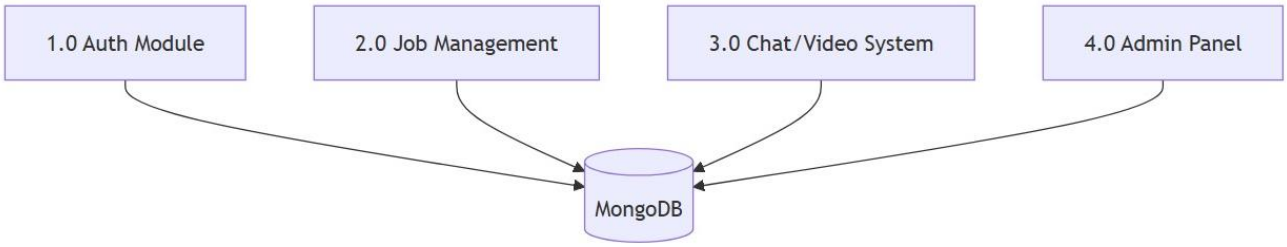


**Figure4.2.2 : DFD (level 1) Diagram**

**Level 2 DFD**

- **Admin**

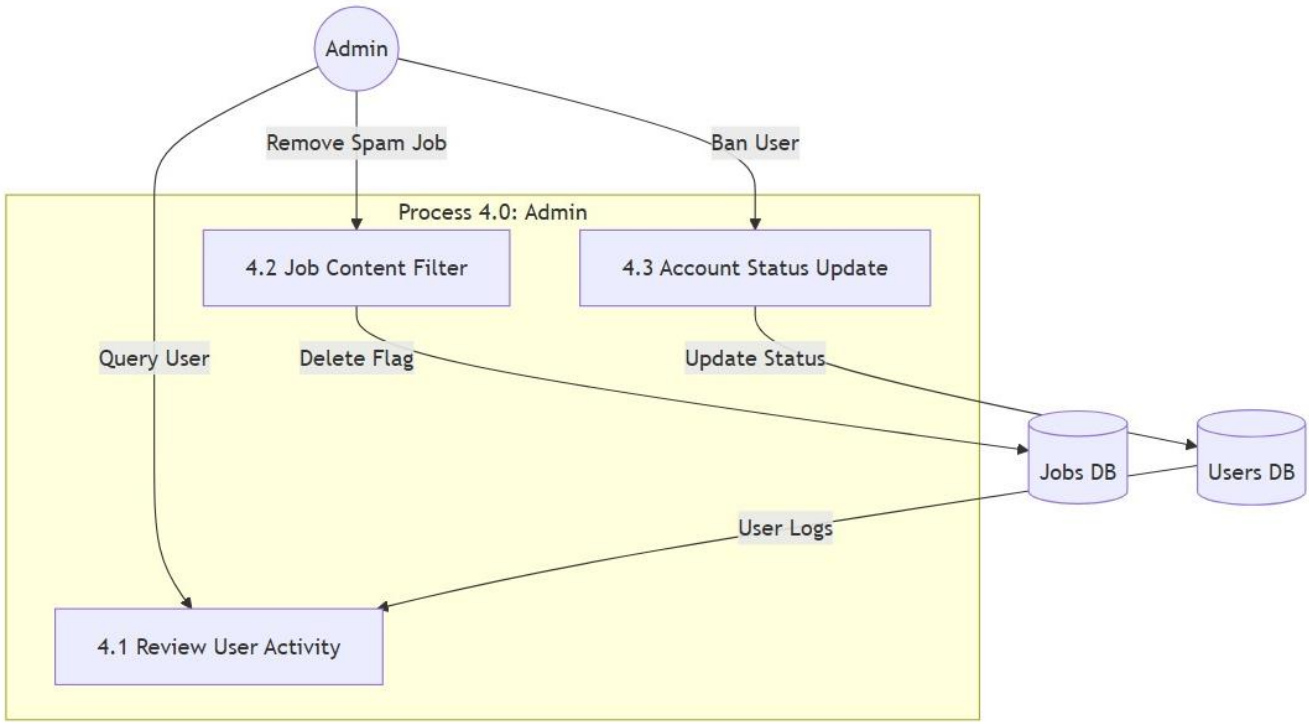


**Figure4.2.3 : DFD (level 2 - Admin) Diagram**

- **Chat and Communication**

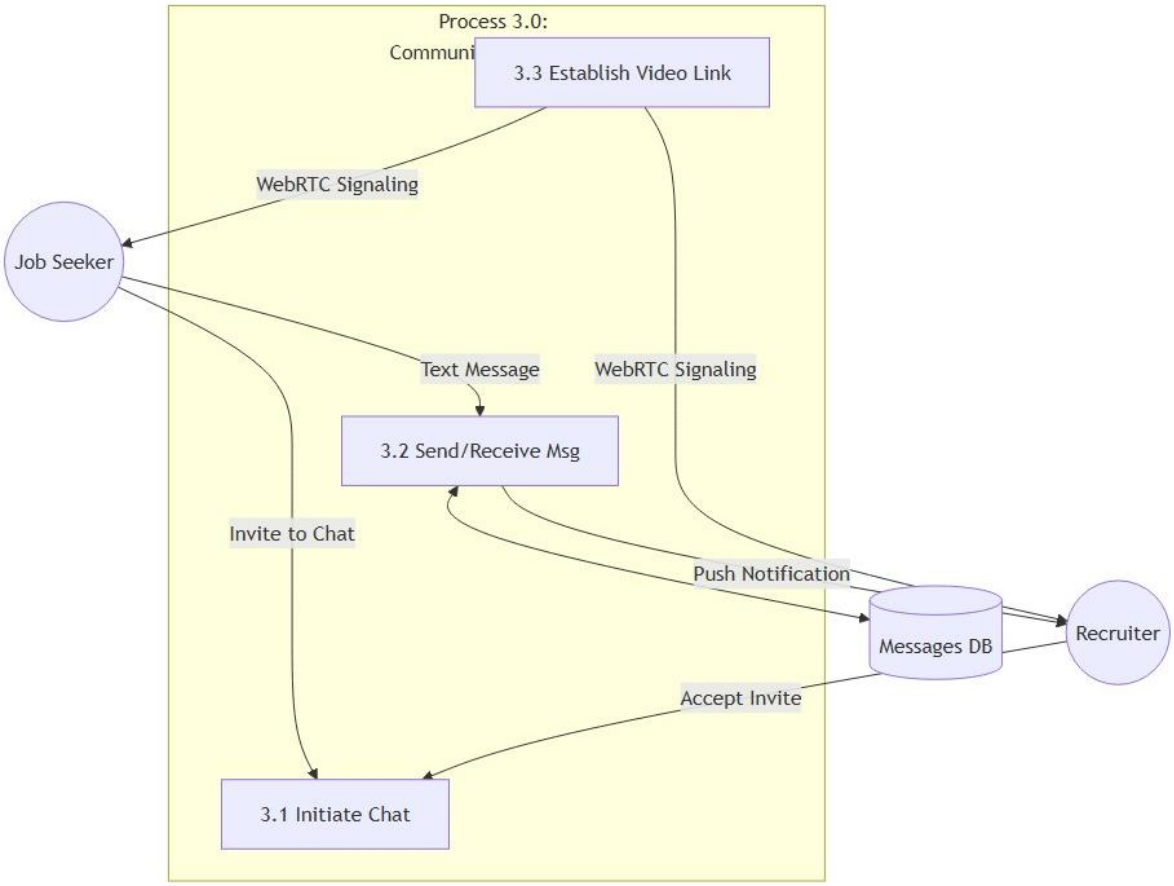


**Figure4.2.4 : DFD (level 2 – Chat and Communication) Diagram**

- **Job Management and Application**

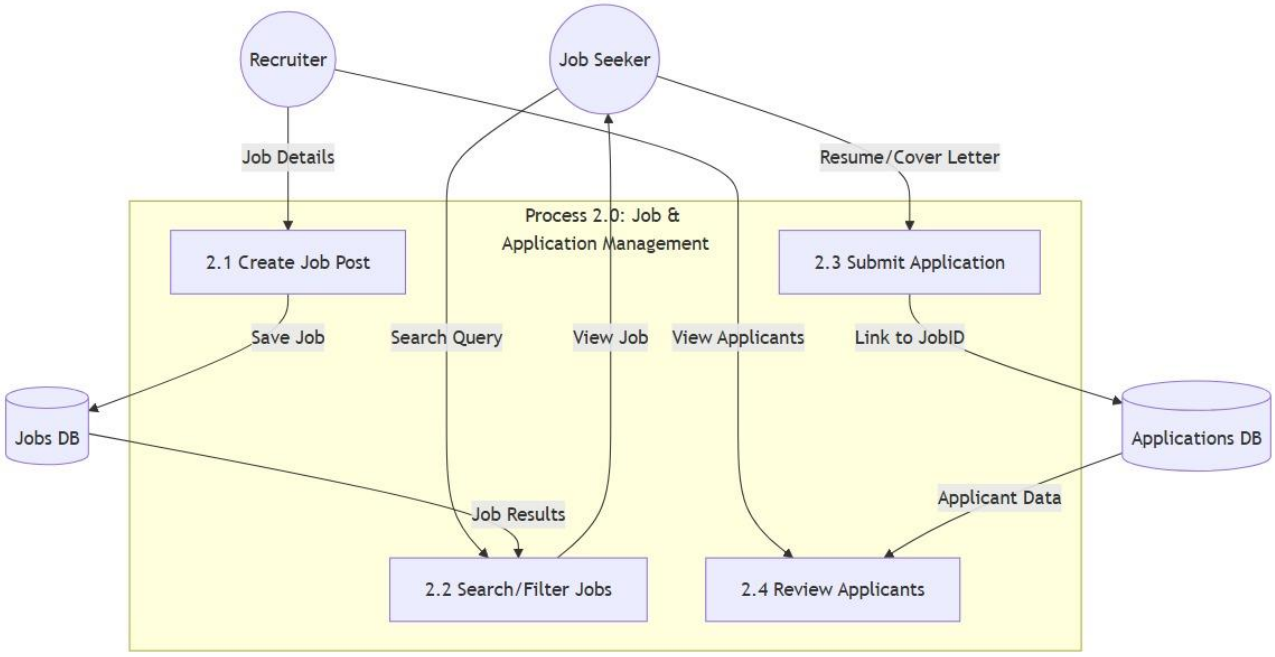


**Figure4.2.5 : DFD (level 2 – Job Management and Application) Diagram**

## 4.7 UML Diagrams

### 4.7.1 Structural Diagrams

#### 4.7.1.1 Class Diagram

The Class Diagram provides a structural view of the system, defining the objects (classes) and their relationships.

- **Major Classes:** User, Profile, Job, Application, Video and Message.
- **Relationships:**
  - A **User** has one **Profile**.
  - A **Recruiter** can post multiple **Jobs**.
  - A **Job** can receive multiple **Applications**.
  - An **Application** links a Job Seeker to a specific Job.
  - **Messages** are exchanged between recruiters and seekers.
  - **Video** call can be Initiated or Joined by recruiters and seekers.

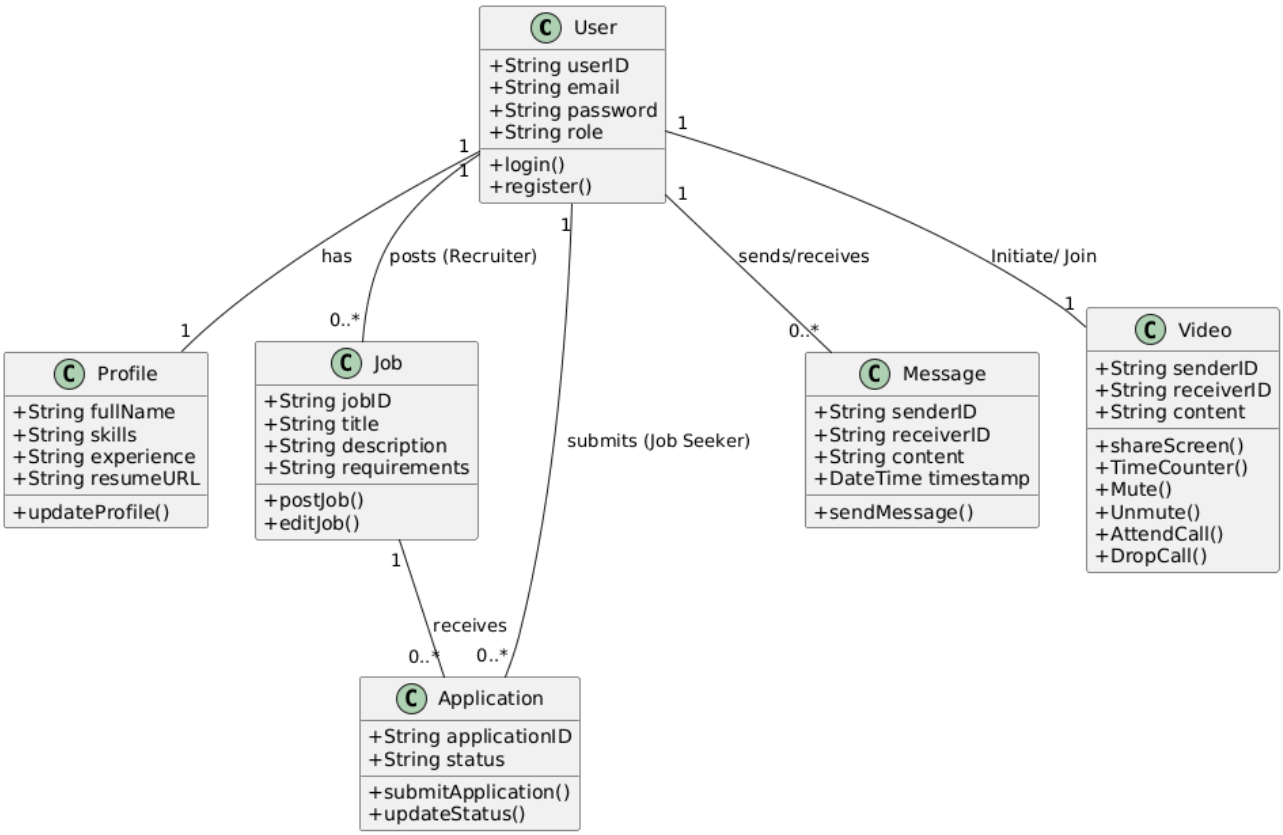


**Figure 4.3.1 : Class Diagram**

#### 4.7.1.2 Object Diagram

This is an instance-level view of the Class Diagram, showing specific examples of data.

- **Instances:** User: "Sayan pal", Job: "MERN Stack Developer", Application: "App_01".

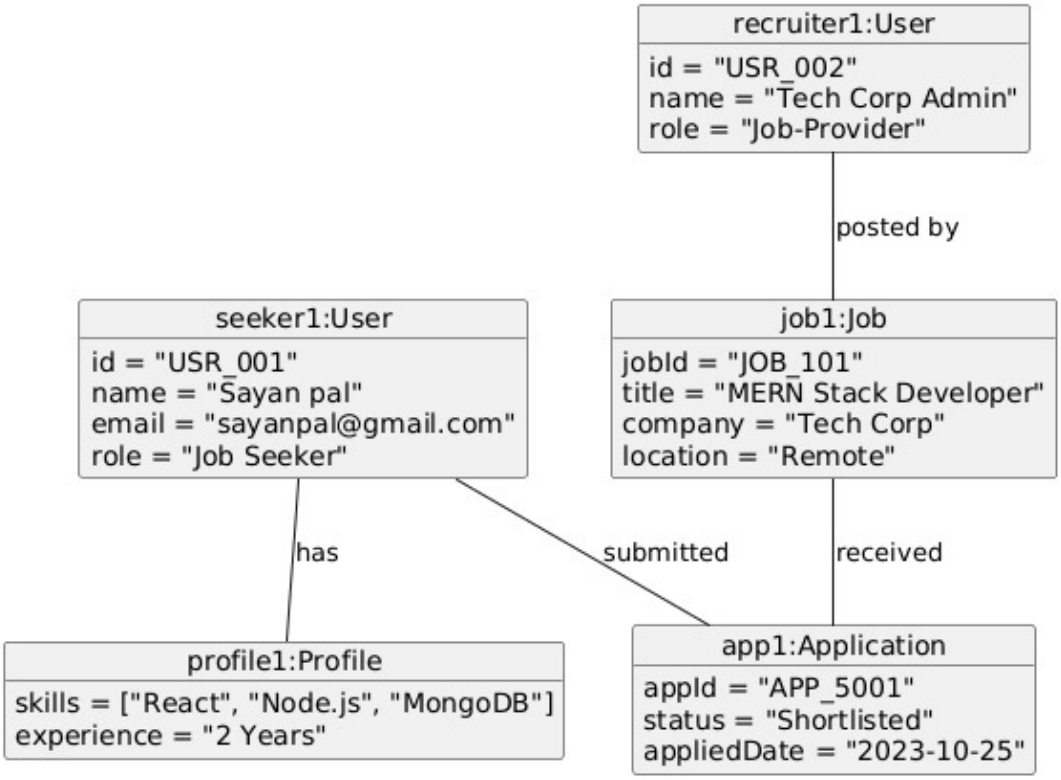


**Figure 4.3.2 : Object Diagram**

### 4.7.1.3 Component Diagram

This diagram describes how the system is divided into independent modules that communicate through interfaces.

- **Frontend Components:** React UI, Authentication UI, Chat UI, Video UI.
- **Backend Components:** Express Server, API Routes, Video/Chat Signaling.
- **Data Component:** MongoDB database.

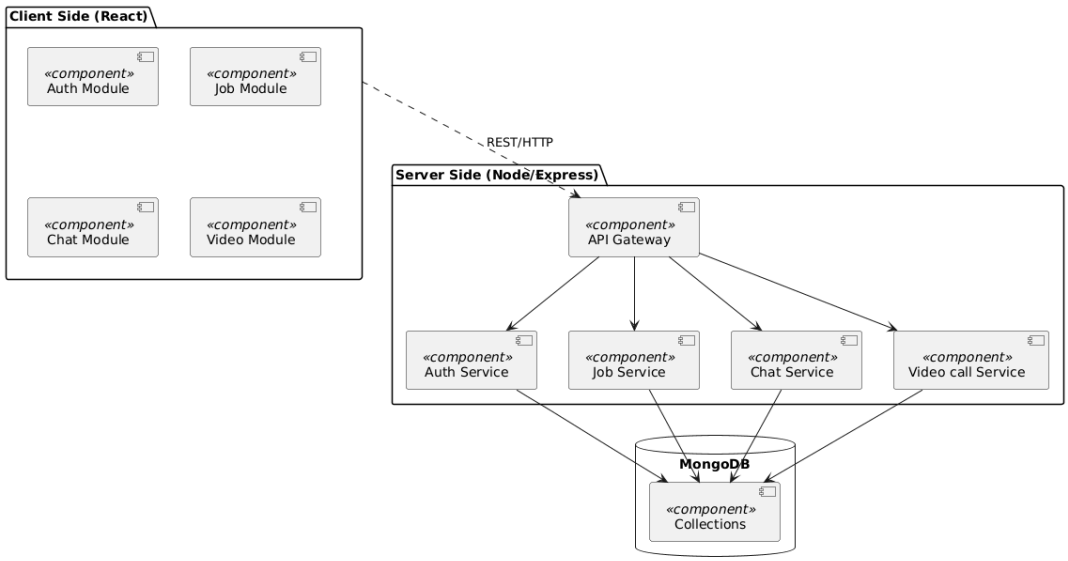


**Figure 4.3.3 : Component Diagram**

#### 4.7.1.4 Deployment Diagram

Illustrates the physical hardware and software environment. The current system is designed for localhost Deployment.

- **Client Node:** Web browser (Chrome) used for testing.
- **Server Node:** Windows OS running Node.js and MongoDB.

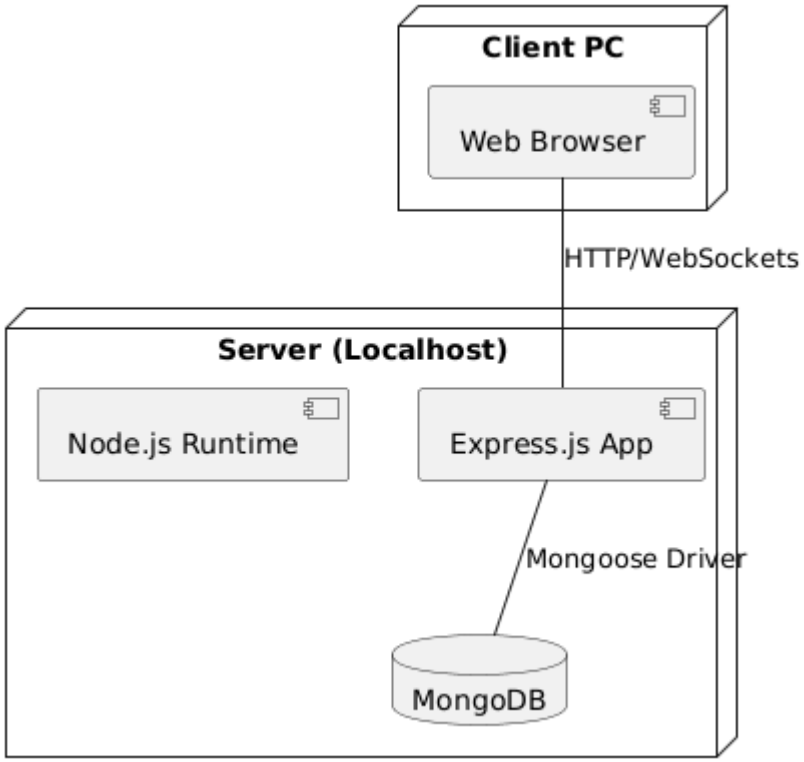


**Figure 4.3.4 : Deployment Diagram**

### 4.7.2 Behavioural Diagrams

#### 4.7.2.1 Use Case Diagram

The Use Case diagram illustrates the functional requirements of the system by showing the interactions between different actors and their specific actions.

- **Actors:** Job Seeker, Recruiter, and Admin.
- **Key Use Cases:**
  - **Job Seeker:** Registration, login, profile management, resume upload, job search, job application, and participating in chat/video interviews.
  - **Recruiter:** Job posting, viewing/shortlisting applicants, and conducting real-time communication via chat or video.
  - **Admin:** User management, monitoring job postings, and overall system supervision.

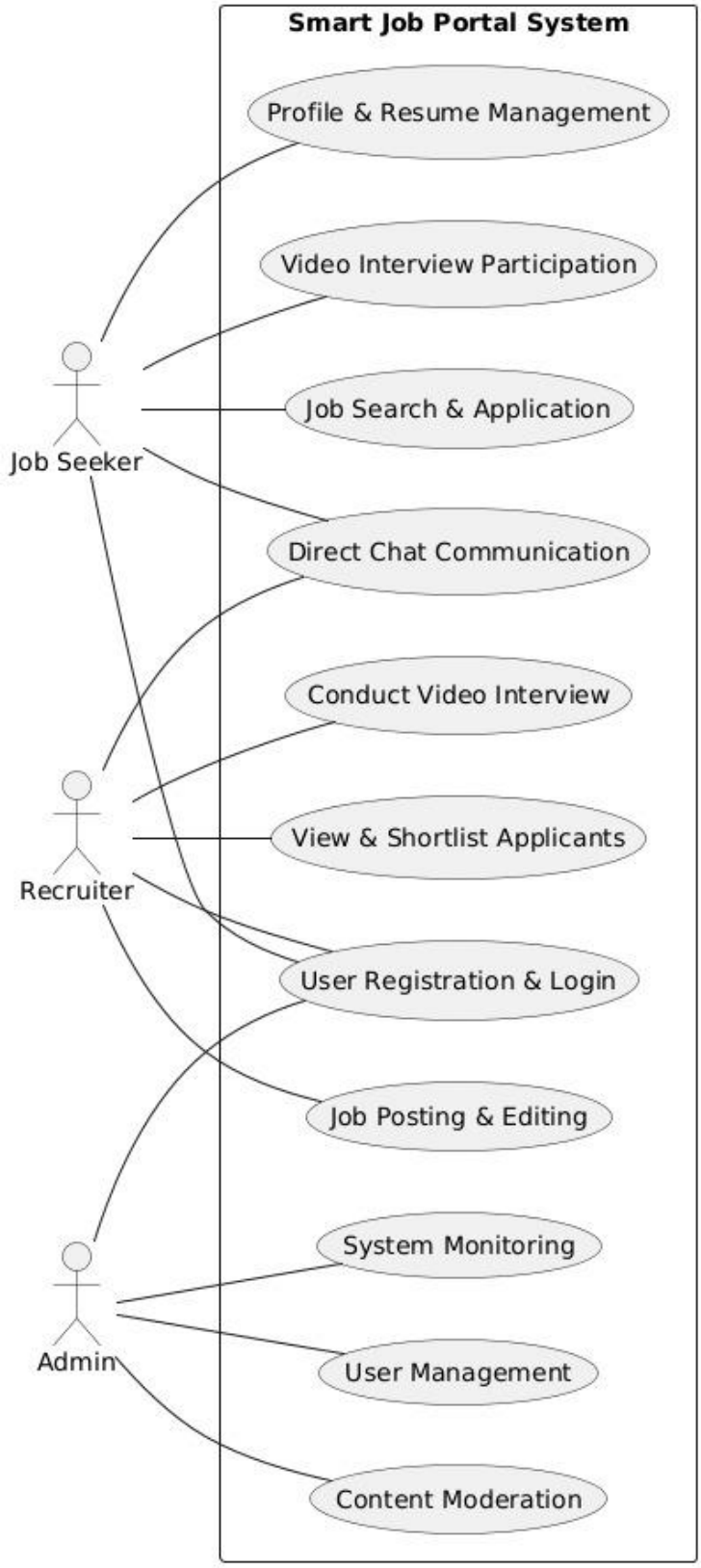


**Figure 4.3.5 : Use Case Diagram**

### 4.7.2.2 Activity Diagram

The dynamic flow of control between activities. For example, the **Job Application Flow**:

1. Job Seeker logs in.
2. Searches for a job.
3. System checks if profile is complete.
4. Submits application.
5. Recruiter is notified.

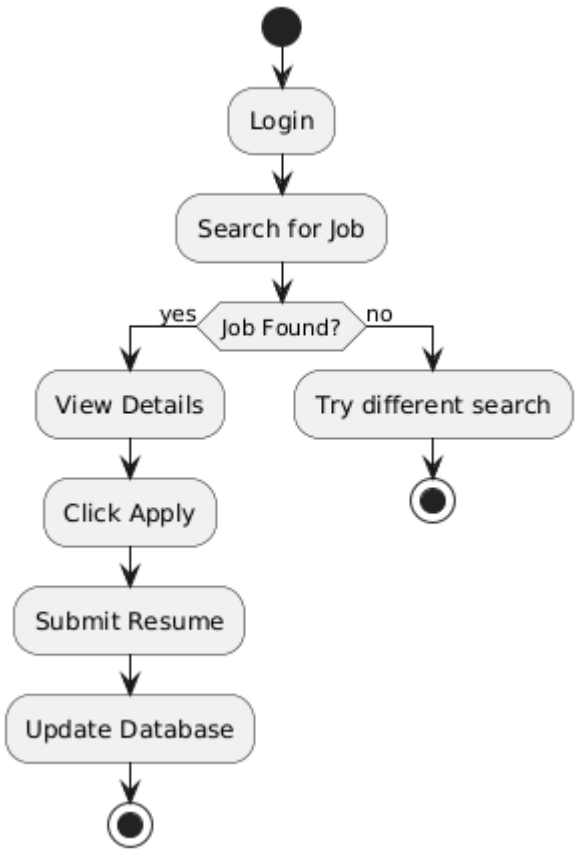


**Figure 4.3.6.1 : Activity Diagram (Job Application Flow)**

The dynamic flow of control between activities. For example, the **Job Post Flow**:

1. Recruiter Logs in.
2. Navigate to “Post Job” or “Dashboard” page and open post Job form.
3. Enter Job Details.
4. Click submit.
5. If data is valid It saves in Database by showing success message.
   Or,
   Shows Error messege and get back to the form.

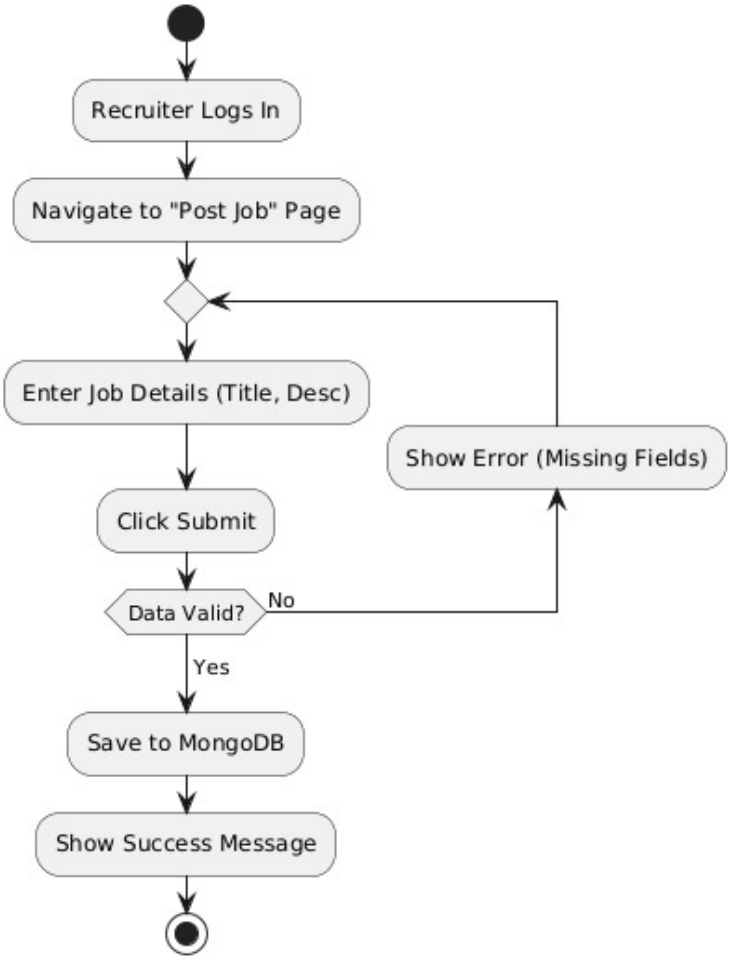


**Figure 4.3.6.2 : Activity Diagram (Job Post Flow)**

The Activity Diagram for the Smart Job Portal illustrates the dynamic flow of control and data between the **Job Seeker**, the **Recruiter**, and the **System** (Backend). It focuses on the sequential and conditional steps involved in the recruitment lifecycle.

**Key Highlights of the Diagram:**

- **Decision Points:** The diagram highlights critical logic gates, such as checking if a user's profile is complete before allowing an application, and the recruiter's decision-making process (Shortlist vs. Reject).
- **Swimlanes:** The flow is divided into swimlanes to clearly distinguish between frontend actions (Job Seeker), backend processing (System), and administrative actions (Recruiter).
- **Concurrency & Real-time Integration:** It captures the transition from a standard CRUD operation (applying for a job) to a real-time interaction (Video Interview via WebRTC).
- **State Updates:** It shows how the database state changes from "Pending" to "Shortlisted" or "Hired" based on user interactions.

**Purpose in Project:** This diagram was used during the development phase to map out the **Redux state transitions** and **API endpoint requirements**. It ensures that the "Happy Path" (successful hire) and "Alternative Paths" (rejection or incomplete profile) are handled gracefully by the MERN stack architecture.

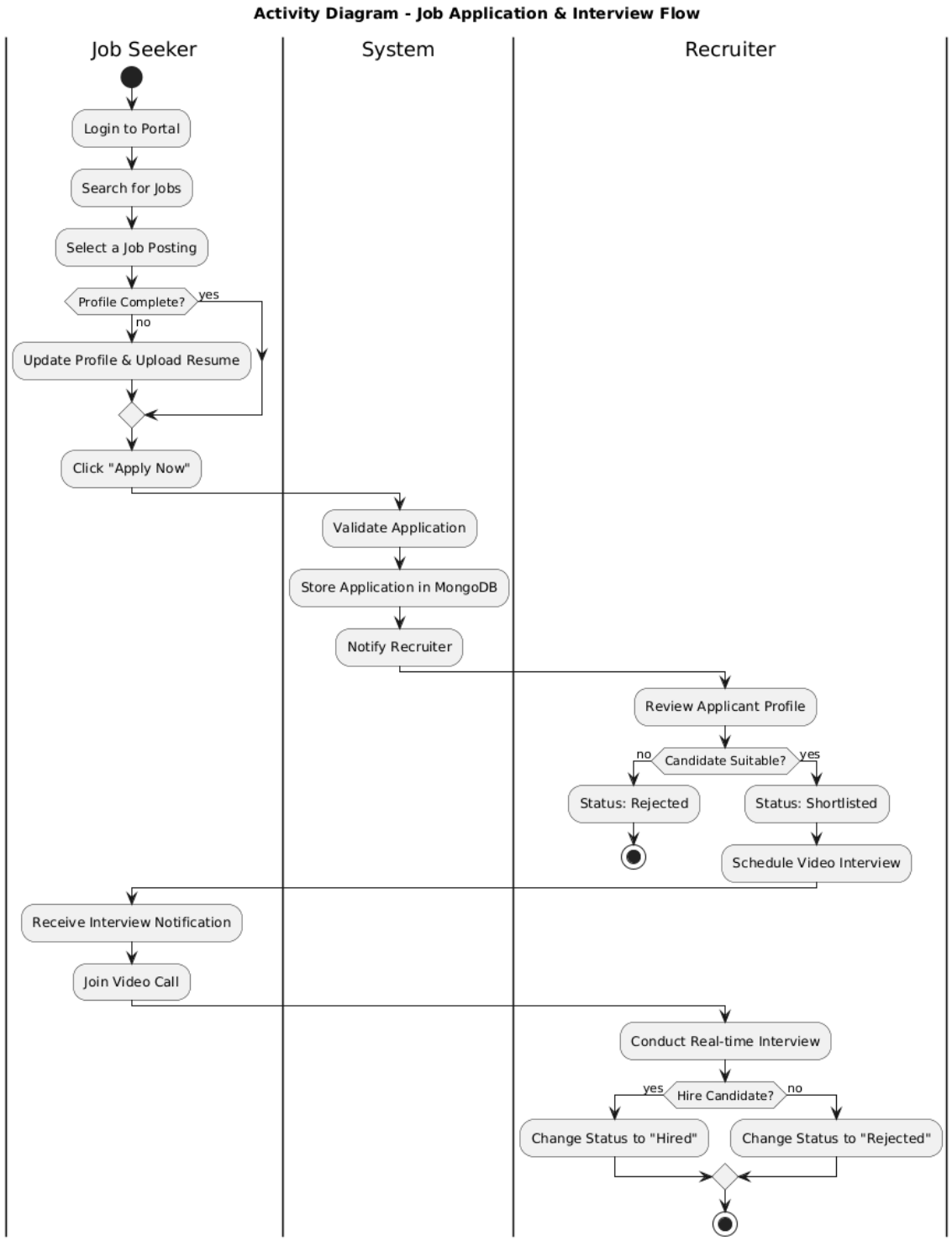


**Figure 4.3.6.3 : Activity Diagram**

## 4.7.2.3 State Machine Diagram

Tracks the status changes of an object, such as the **Job Application Status**:

- **States:** Submitted → Shortlisted / Rejected → Interviewed → Hired / Rejected.

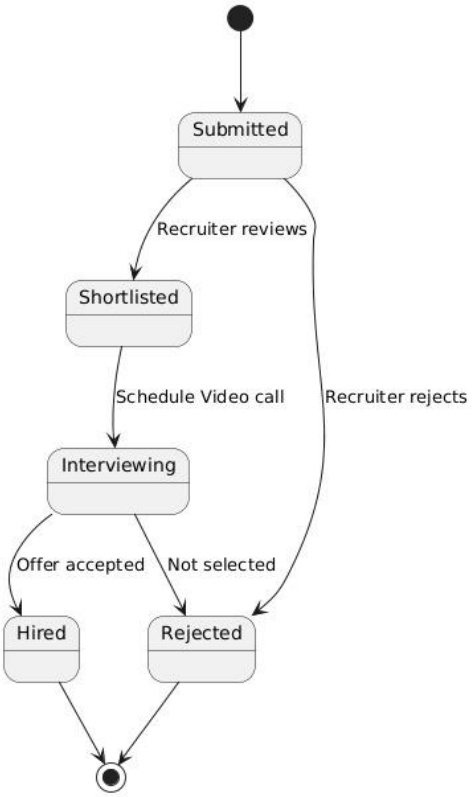


**Figure 4.3.7 : State Machine Diagram**

### 4.7.2.4 Sequence Diagram

Visualizes the interaction between objects over time. A common sequence in this portal is the **Chat Communication**:

Seeker sends message via React UI.

1. React calls Express API.
2. Express saves message in MongoDB.
3. Express (via WebSockets) pushes message to Recruiter.

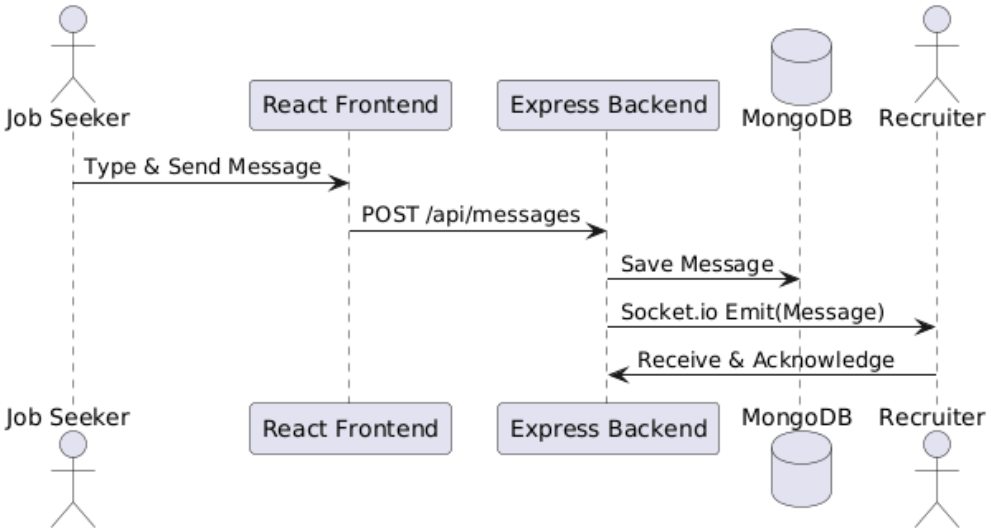


**Figure 4.3.8.1 : Sequence Diagram (Chat Communication)**

Seeker Apply in Jobs via React UI.

The Sequence Diagram illustrates the time-ordered interaction between the system's objects and components during the **User Authentication** and **Job Search** workflows. It highlights how the MERN stack handles security and data retrieval sequentially.

**A. Authentication Workflow :**

- **Trigger:** The process begins when a user submits their login credentials through the React-based login form.
- **Security:** The Backend (Express) does not store plain-text passwords. It retrieves the hashed password from **MongoDB** and uses the **Bcrypt** library to verify the match.
- **Session Management:** Upon successful verification, a **JSON Web Token (JWT)** is generated. This token is sent back to the frontend and stored locally, ensuring that subsequent requests are authenticated without requiring the user to log in again.

**B. Data Retrieval Workflow :**

- **Stateless Communication:** When a user searches for a job, the React frontend sends a GET request. The sequence shows that the JWT must be validated by the backend before the database is queried.
- **Database Interaction:** The diagram depicts the asynchronous nature of the MERN stack, where the Backend waits for the MongoDB query to resolve before formatting the data as a JSON response for the Frontend.

**C. Technical Significance:**

- **Inter-component Communication:** Shows the clear separation between the Client Side (React) and Server Side (Node/Express).
- **Real-time Performance:** The sequential flow demonstrates the efficiency of using a non-blocking I/O (Node.js) environment to handle database requests.

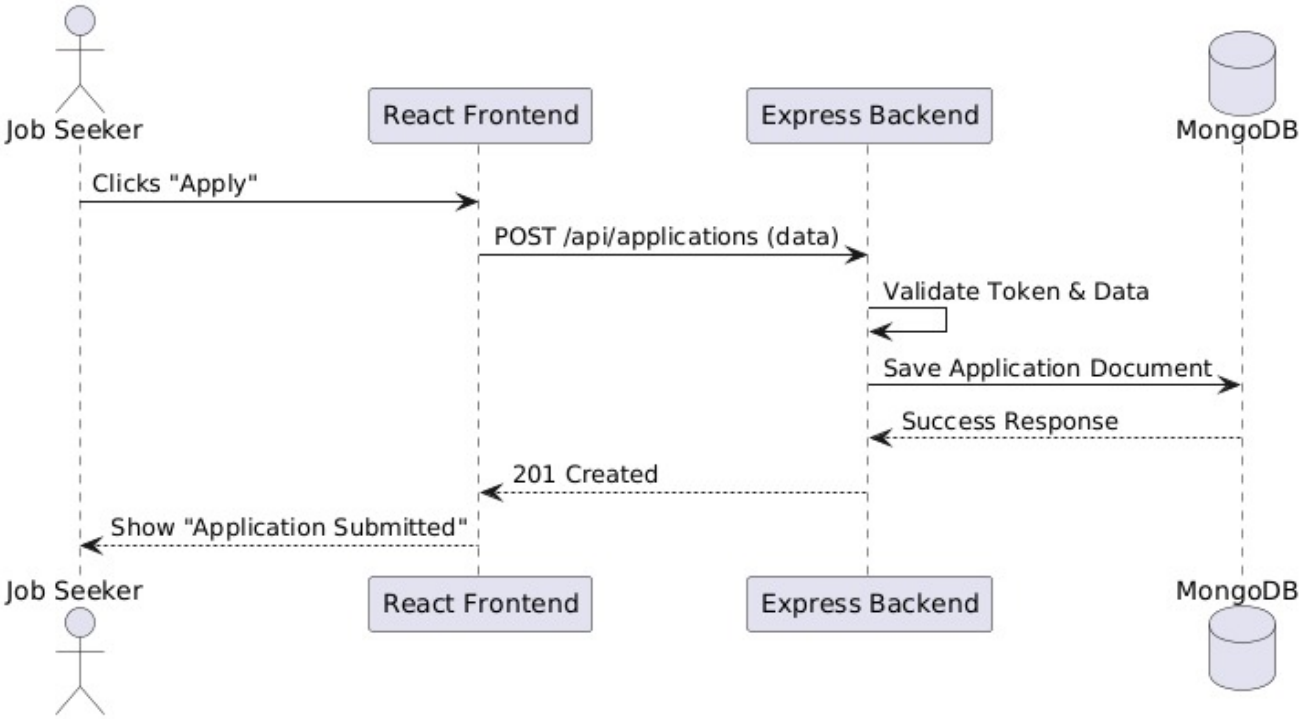


**Figure 4.3.8.2 : Sequence Diagram (Job Application)**

### 4.7.2.5 Timing Diagram

Focuses on the timing constraints during real-time communication, such as the signaling process for a **Video Interview**.

- **Events:** Initialize Call → Signaling (Offer/Answer) → P2P Connection established.


**Figure 4.3.9 : Timing Diagram**

---

## 4.8 Gnatt Chart

• **Overview:** The Gantt Chart provides a graphical illustration of the project schedule, showing the start and finish dates of the various phases of the **Software Development Life Cycle (SDLC)** for the 2024-2025 academic year.

• **Phases Breakdown:**

- **Requirement Analysis:** Initial weeks spent gathering needs for Job Seekers and Recruiters.
- **System Design:** Drafting UML diagrams (Use Case, Sequence, etc.) and database schema design.
- **Implementation (Coding):** The longest phase, involving frontend development with React and backend development with Node/Express.
- **Testing & Quality Assurance:** Final weeks dedicated to unit testing, integration testing, and bug fixing.
- **Documentation:** Finalizing the MCA project report and preparing for the viva.

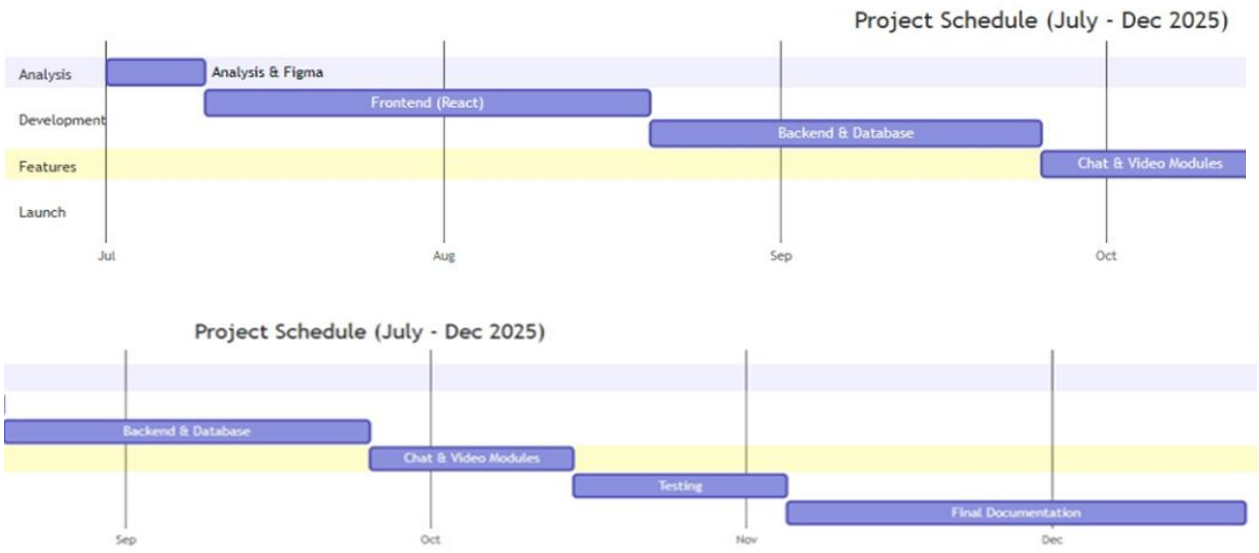


**Figure 4.3.10 : Gnatt Chart**

---

## 4.9 Security Design

Security is an important part of the system design.

Measures used:

- Encrypted passwords
- Token-based authentication
- Role-based access control
- Restricted API access

These measures protect user data and prevent unauthorized access.

---

## 4.10 Design Advantages

- Modular structure
- Easy maintenance
- Secure communication
- Scalable architecture
- Clear separation of frontend and backend

---

## 4.11 Conclusion

The system design provides a clear and structured blueprint for developing the Smart Job Portal. It ensures that the application is secure, efficient, and easy to maintain while meeting all functional requirements.

# CHAPTER 5

# TOOLS AND TECHNOLOGIES USED

---

## 5.1 Introduction

The selection of appropriate tools and technologies plays a vital role in the success of any software project. For the development of the Smart Job Portal, modern and widely accepted web technologies were chosen to ensure reliability, performance, and ease of development. This chapter describes the technologies used in both frontend and backend development, along with the database and supporting tools.

---

## 5.2 MERN Stack Overview

The MERN stack is a modern web development stack composed entirely of JavaScript-based technologies: **MongoDB**, **Express.js**, **React.js**, and **Node.js**. This stack enables developers to build full-stack web applications using a single programming language—JavaScript—across both client-side and server-side components.

The MERN stack follows a **three-tier architecture**:

- **Frontend (Client Layer):** Developed using React.js to create interactive and responsive user interfaces.
- **Backend (Application Layer):** Built with Node.js and Express.js to handle business logic, routing, authentication, and API management.
- **Database Layer:** Implemented using MongoDB to store and manage application data in a flexible, document-oriented format.

Using the MERN stack improves development efficiency, reduces context switching between different languages, and ensures smooth communication between frontend and backend through RESTful APIs. The stack is highly scalable, making it suitable for applications with growing user bases, such as job portals and recruitment management systems.

---

## 5.3 React.js

React.js is a JavaScript library developed by Facebook for building fast and interactive user interfaces, particularly for single-page applications (SPAs). It focuses on the **view layer** of the application and allows developers to build complex UIs using small, independent, and reusable components.

One of the core features of React.js is its **component-based architecture**, where the user interface is divided into reusable components such as forms, buttons, cards, and dashboards. This modular approach improves code readability, maintainability, and reusability across the application.

React.js uses a **Virtual DOM**, which is an in-memory representation of the actual DOM. When the application state changes, React updates only the affected components instead of reloading the entire page. This significantly improves performance and provides a smooth user experience.

In this project, React.js is used to develop:

- User interfaces for **job seekers** (profile creation, job browsing, applications)
- Dashboards for **recruiters** (job posting, candidate management)
- The **admin panel** (user management, monitoring activities)

React's efficient state management and reusable components make it ideal for handling dynamic and data-driven interfaces required in a recruitment platform.

---

## 5.4 Custom CSS (Flexbox and Media Queries)

Custom Cascading Style Sheets (CSS) are used to design and style the application interface. Instead of relying on external UI frameworks, custom CSS provides complete control over the look and feel of the application.

**Flexbox** is used extensively for layout alignment. It allows elements to be arranged dynamically in rows or columns and helps in creating flexible layouts that adapt to different screen sizes. Flexbox simplifies alignment, spacing, and distribution of UI components, especially in dashboards and forms.

**Media queries** are used to implement responsive design. They enable the application layout to adjust automatically based on screen size, resolution, and device type (desktop, tablet, or mobile). This ensures consistent usability across different devices.

Advantages of using custom CSS in this project include:

- Lightweight and faster loading styles
- Easy customization and modification
- No dependency on third-party styling libraries
- Better understanding and control of UI behavior

Overall, custom CSS ensures a clean, responsive, and user-friendly interface tailored specifically to the project's requirements.

---

## 5.5 Node.js

Node.js is a server-side JavaScript runtime built on Chrome's V8 engine. It allows JavaScript code to run outside the browser, enabling the development of scalable backend applications.

A key feature of Node.js is its **non-blocking, event-driven I/O model**, which allows the server to handle multiple client requests concurrently without waiting for one request to complete before processing another. This makes Node.js highly efficient and suitable for real-time and high-traffic applications.

Advantages of using Node.js include:

- High performance due to asynchronous execution
- Efficient handling of concurrent requests
- Large ecosystem of open-source packages via npm
- Seamless integration with frontend JavaScript

In this project, Node.js is responsible for handling server-side logic, managing API requests, processing authentication, and coordinating communication between the frontend and the database.

---

## 5.6 Express.js

Express.js is a lightweight and flexible web application framework built on top of Node.js. It simplifies the process of building web servers and RESTful APIs by providing a robust set of features for routing and middleware integration.

Express.js helps in:

- Defining API endpoints (GET, POST, PUT, DELETE)
- Handling HTTP requests and responses
- Managing middleware for authentication, validation, and error handling
- Structuring backend code in a clean and modular way

In this project, Express.js acts as the **bridge between the frontend and the database**. It processes requests from the React frontend, applies business logic, interacts with MongoDB, and returns appropriate responses. This separation of concerns improves code organization and maintainability.

---

## 5.7 MongoDB

MongoDB is a NoSQL, document-oriented database that stores data in flexible, JSON-like documents called BSON. Unlike traditional relational databases, MongoDB does not require a fixed schema, making it highly adaptable to changing data requirements.

Key advantages of MongoDB include:

- Flexible schema design
- Easy scalability and horizontal scaling
- Efficient handling of large and unstructured data
- Seamless integration with Node.js applications

In this project, MongoDB is used to store:

- User profiles (job seekers, recruiters, admins)
- Job listings and job descriptions
- Job applications and application statuses
- Chat messages and communication data

The ability of MongoDB to manage dynamic and evolving data makes it an ideal choice for a recruitment platform where data structures may change frequently.

---

## 5.8 Additional Tools and Software

| Tool | Purpose |
|---|---|
| Visual Studio Code | Code editor |
| Postman | API testing |
| Git | Version control |
| GitHub | Code repository |
| Browser (Chrome) | Testing and debugging |

---

## 5.9 Platform Requirements

### Hardware Requirements

- Minimum 4 GB RAM
- Intel i3 processor or higher
- Standard desktop or laptop

### Software Requirements

- Windows Operating System
- Node.js
- MongoDB
- Web Browser

---

## 5.10 Conclusion

The tools and technologies selected for this project are reliable, efficient, and suitable for developing a full-stack web application. The MERN stack combined with Custom CSS provides a strong foundation for building a responsive and user-friendly job portal.

# CHAPTER 6

# SYSTEM IMPLEMENTATION

## 6.1 Introduction

System implementation is the phase where the system design is converted into a working application. This chapter explains how the Smart Job Portal was developed using the MERN stack, covering frontend implementation, backend development, database integration, and communication features such as chat and video interviews.

The implementation follows a modular approach, allowing different components of the system to be developed and tested independently.

---

## 6.2 Frontend Implementation

The frontend of the application is developed using React.js. It provides interactive and responsive user interfaces for different user roles.

**Key Frontend Components**

- Login and Registration pages
- Job Seeker dashboard
- Recruiter dashboard
- Job listing and search pages
- Chat interface
- Video interview interface
- Admin panel

Custom CSS is used for styling the components. Flexbox is applied for layout alignment, and media queries are used to ensure responsiveness across different screen sizes.

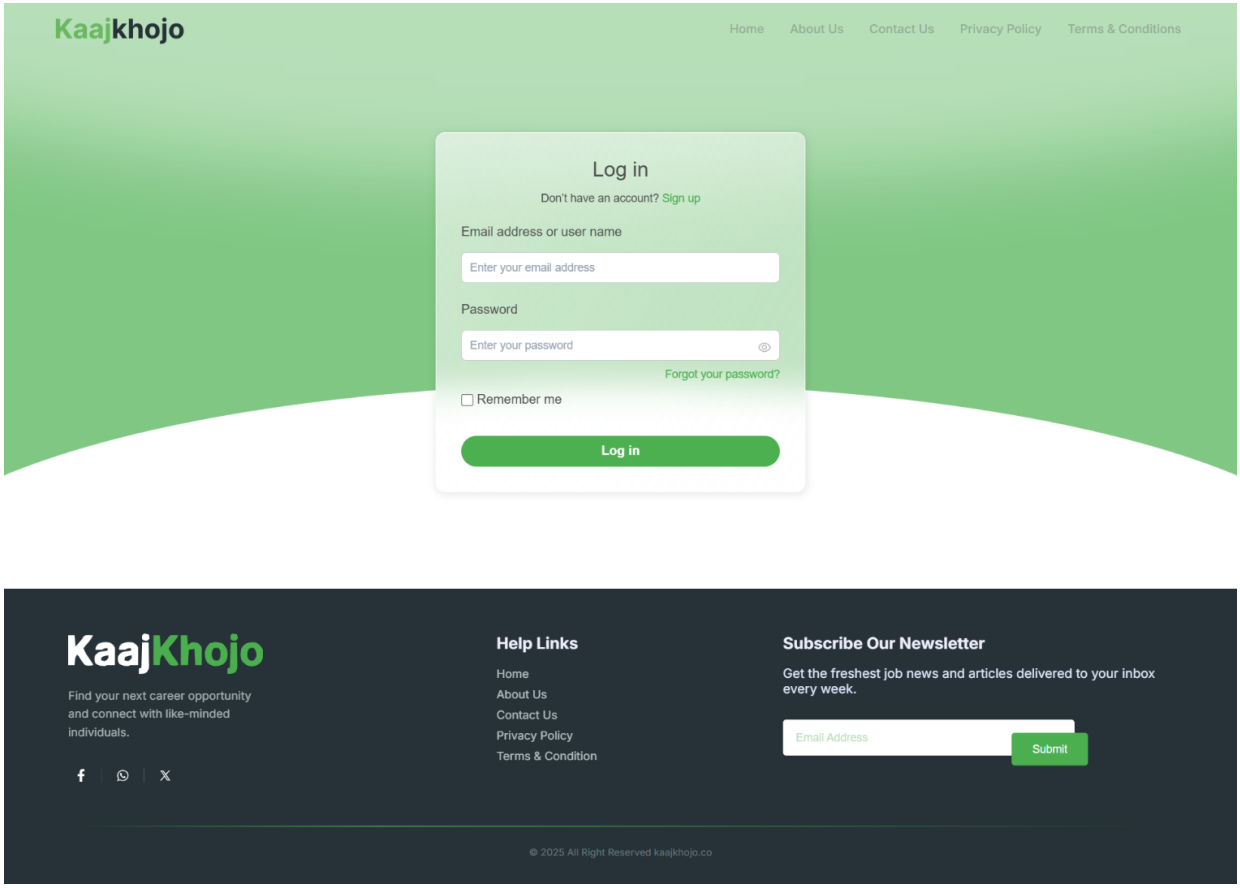


**Figure 6.1 : User Login**

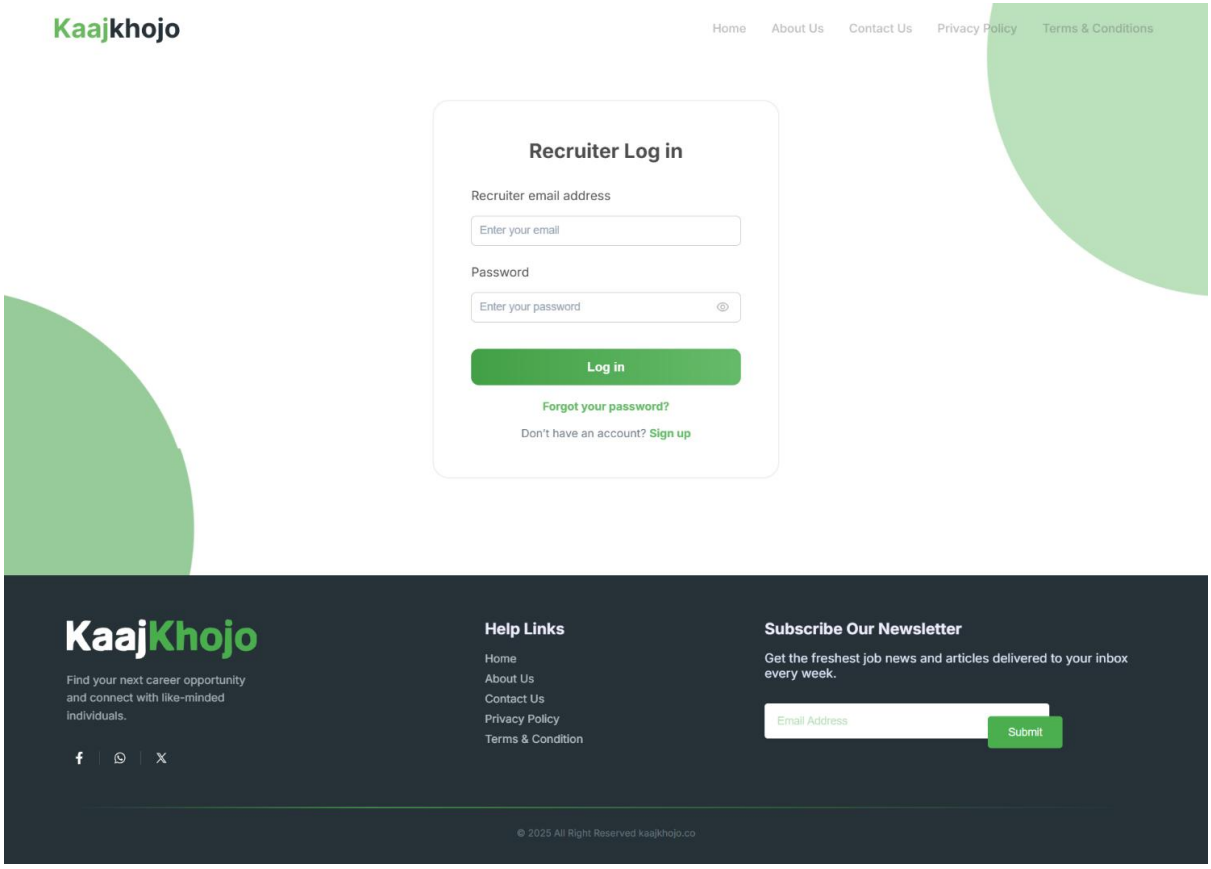


**Figure 6.2 : Recruiter Login**

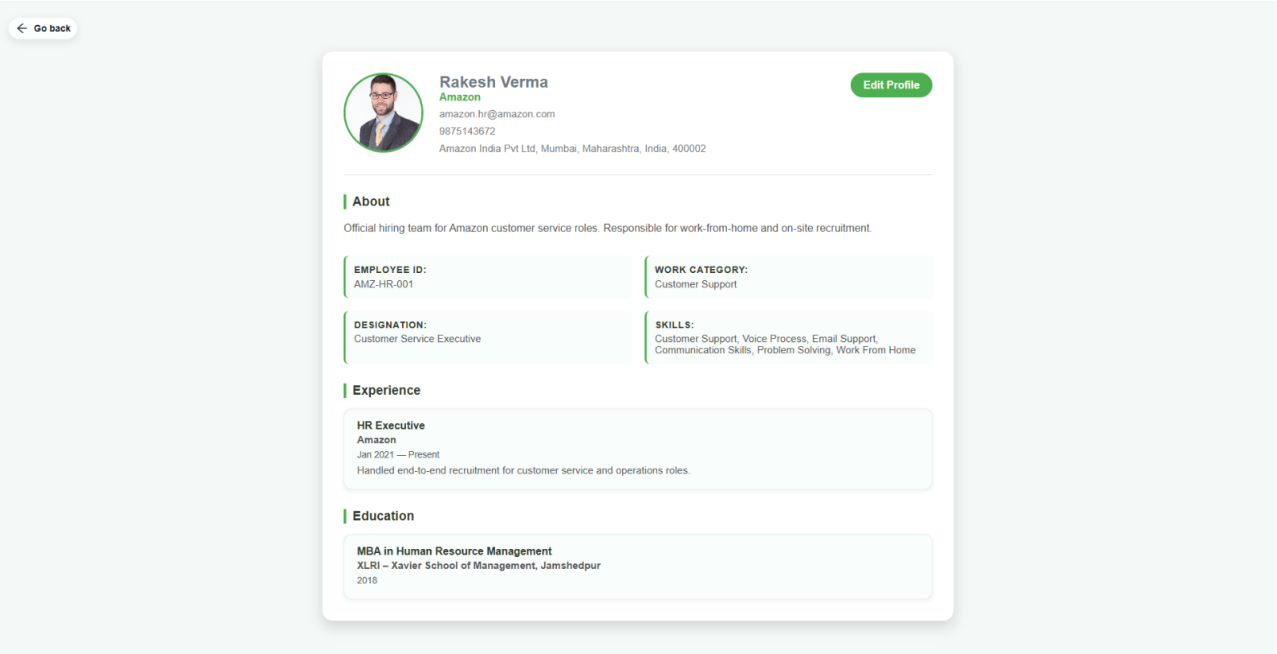


**Figure 6.3 : Recruiter Profile**

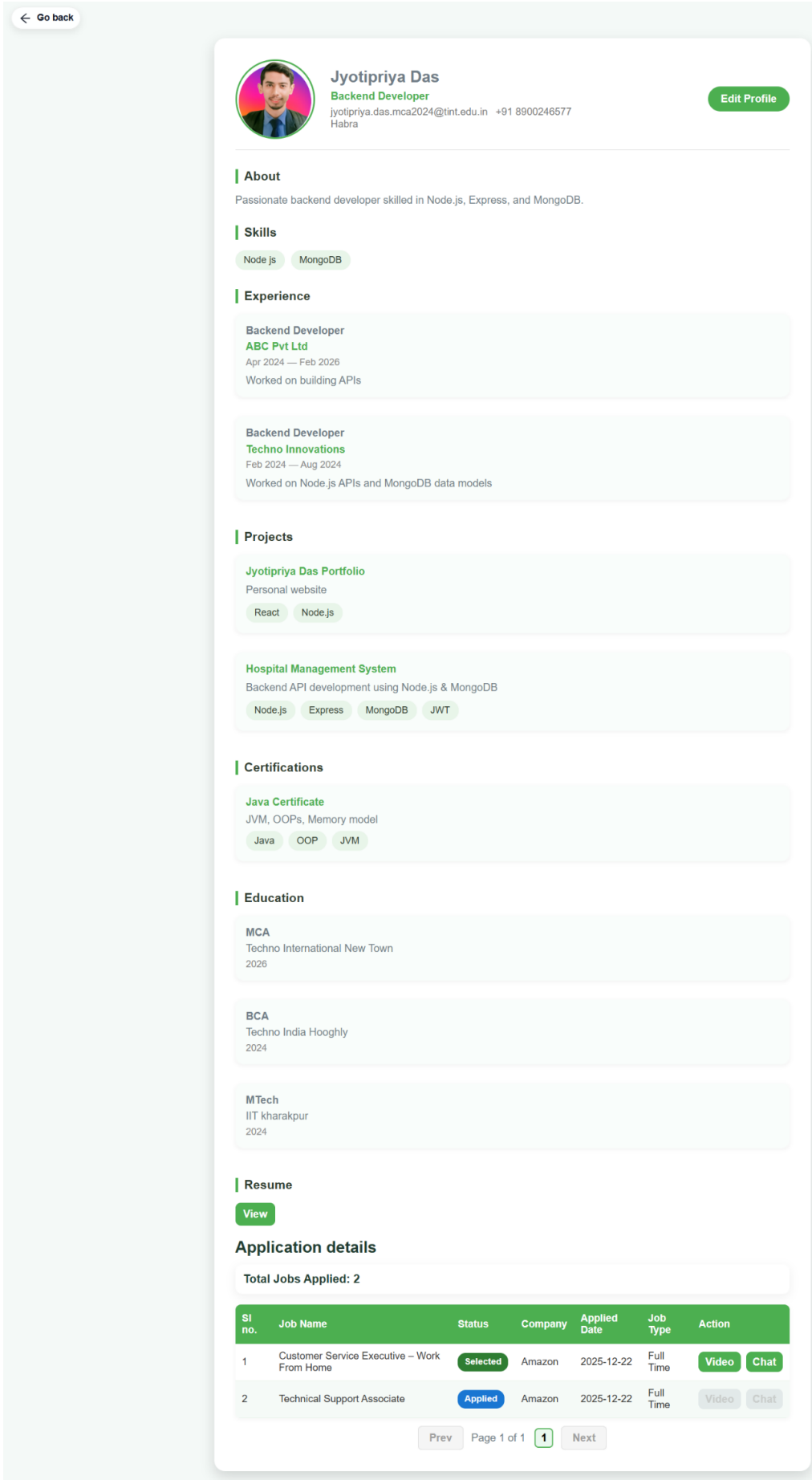


**Figure 6.4 : User Profile**

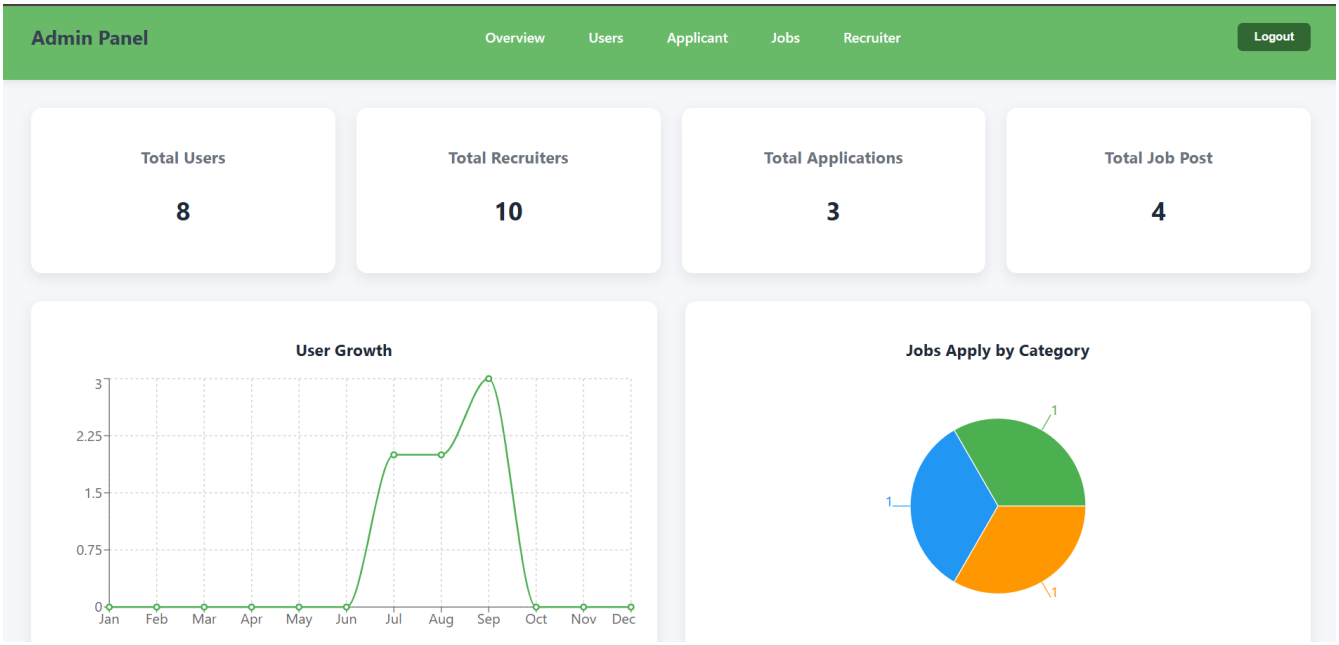


**Figure 6.5 : Admin Dashboard**

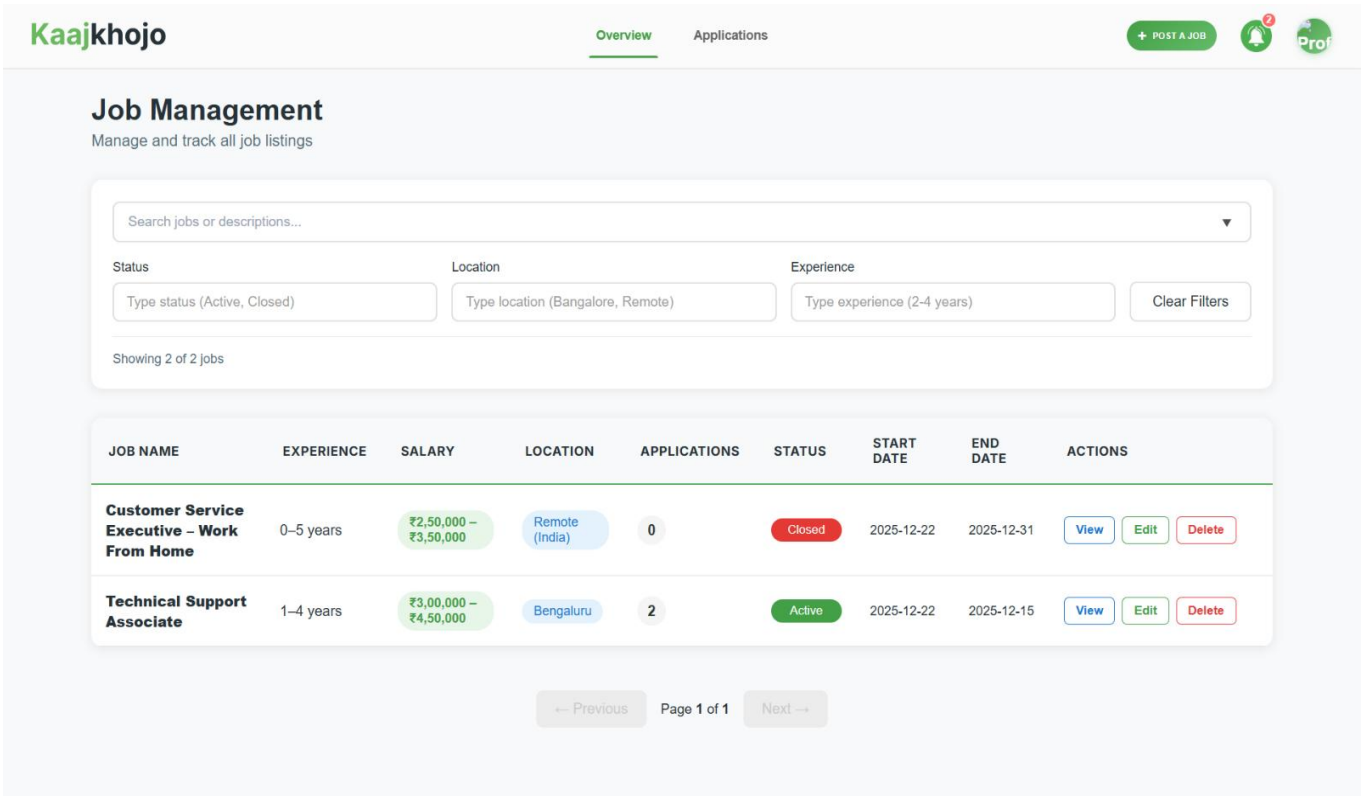


**Figure 6.6 : Recruiter Dashboard**

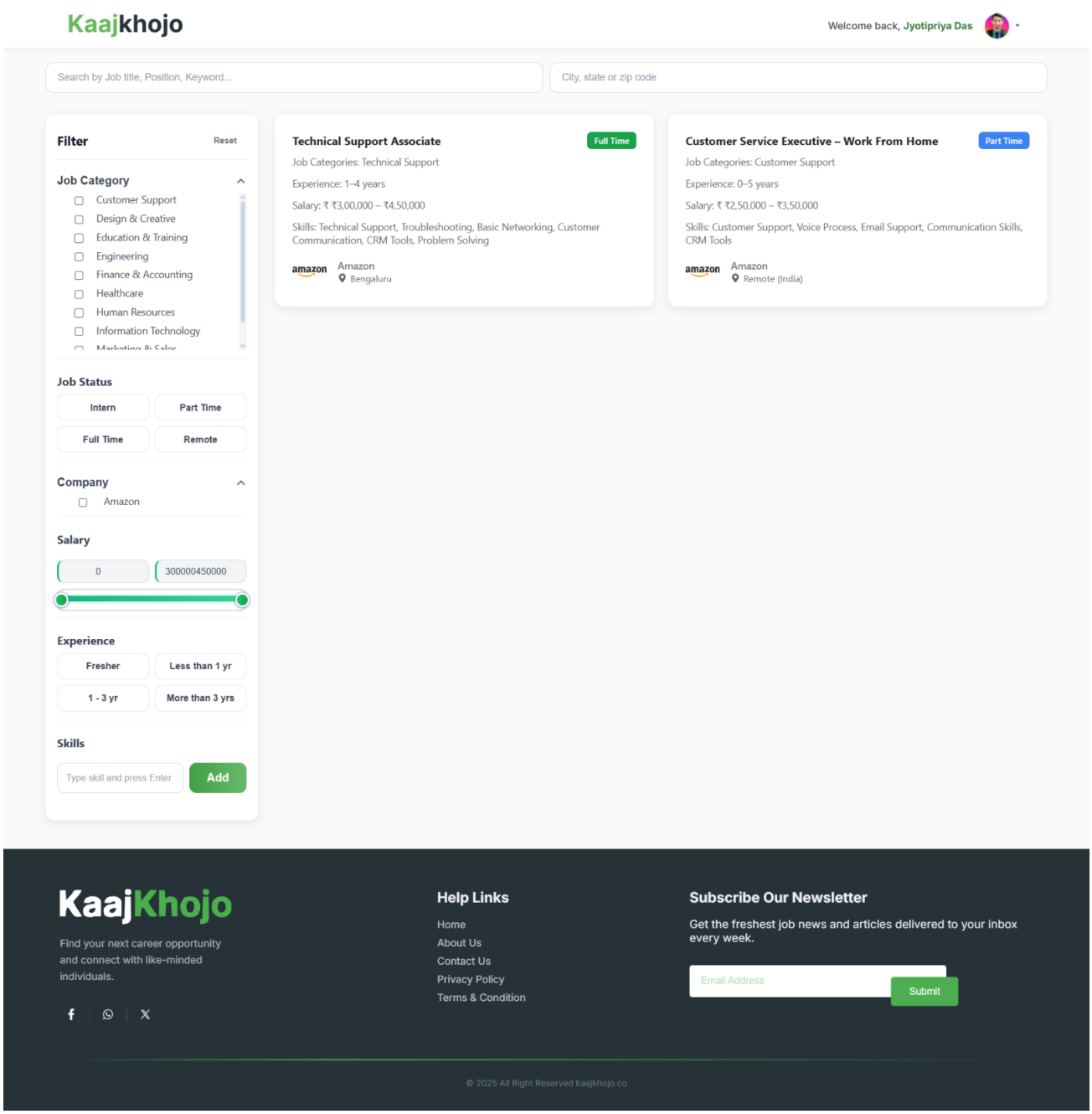


**Figure 6.7 : User Dashboard**

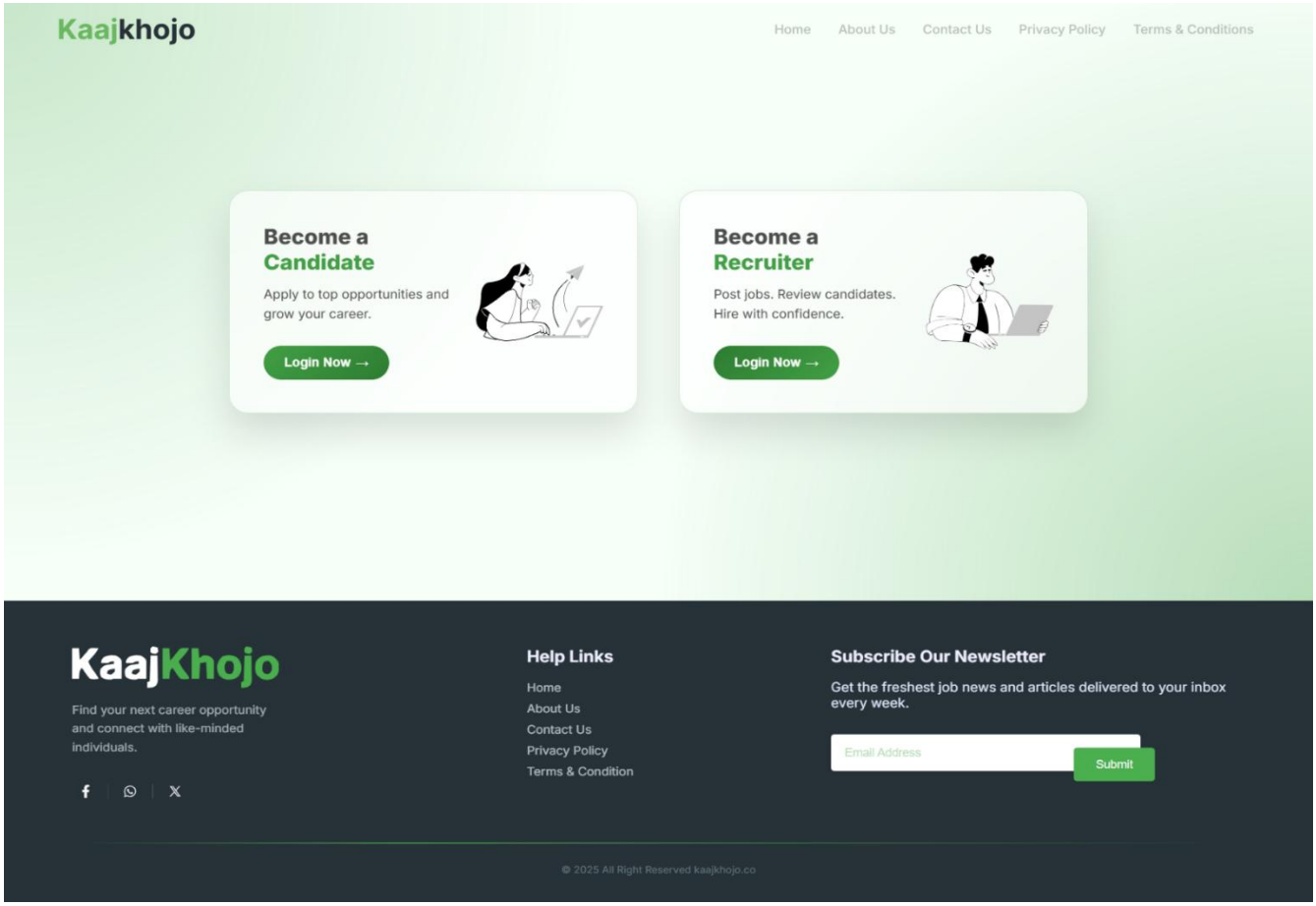


**Figure 6.8 : Choose Role Login**

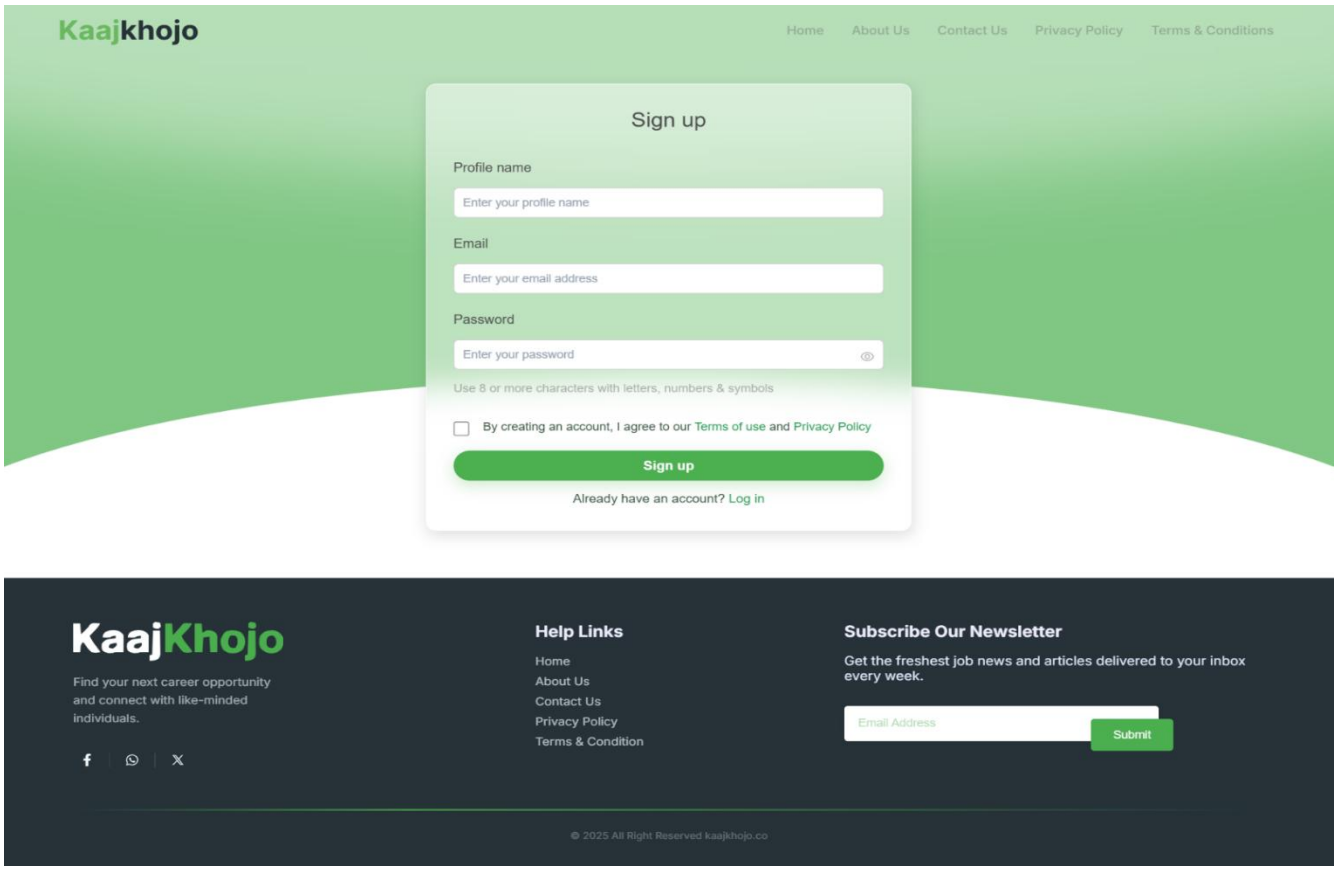


**Figure 6.9 : User Signup**

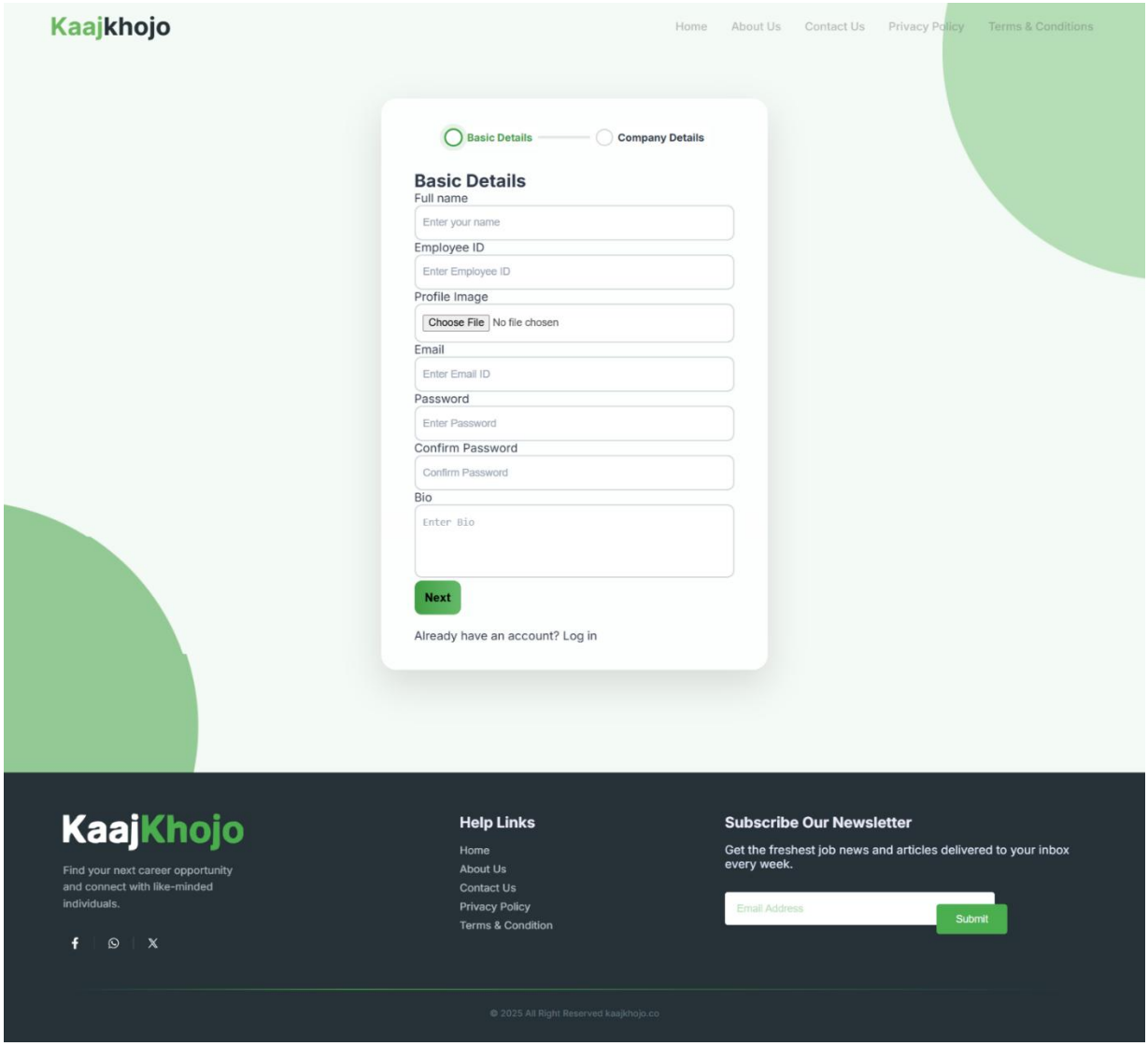


**Figure 6.10 : Recruiter Signup**

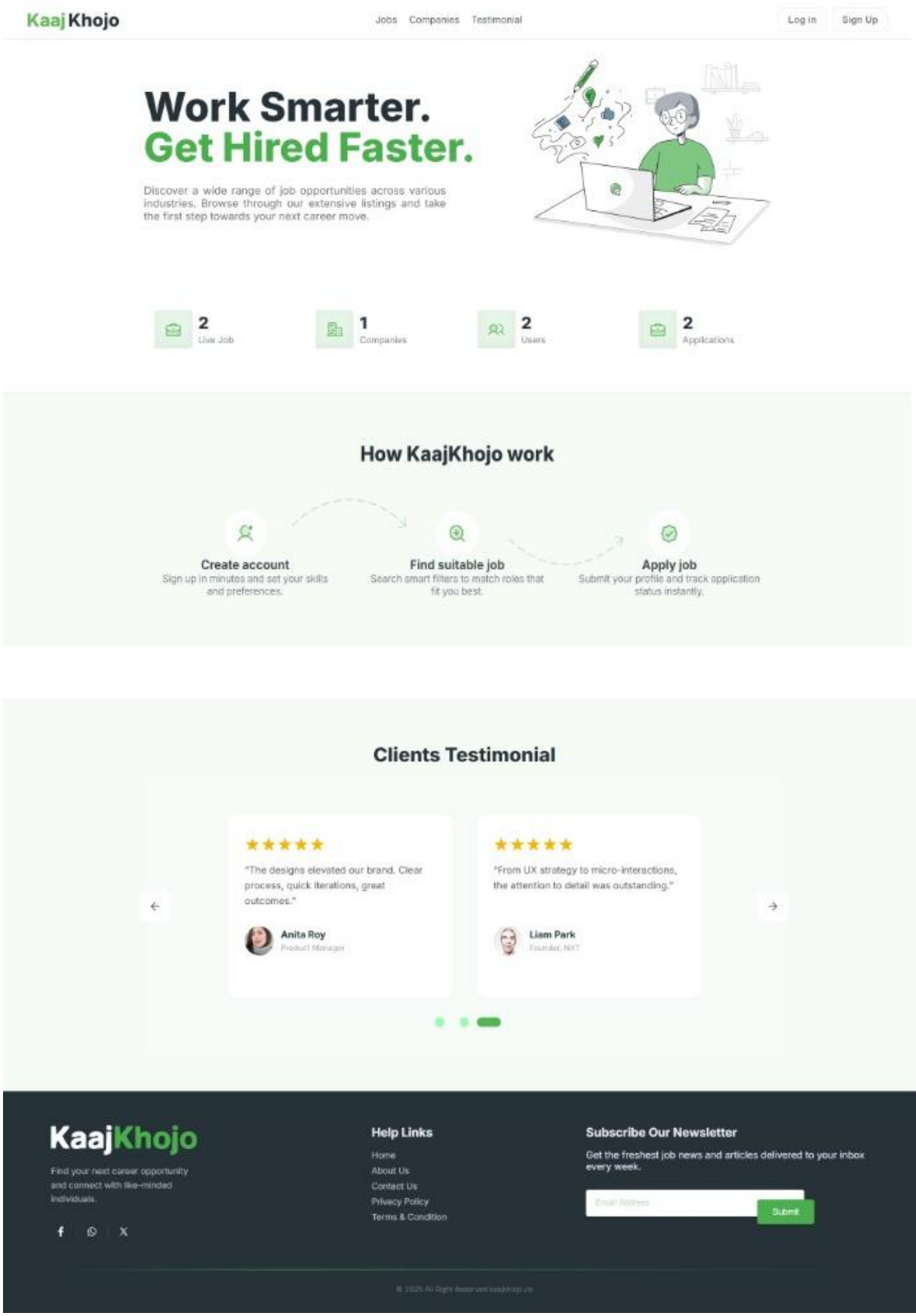


**Figure 6.11 : Home page**

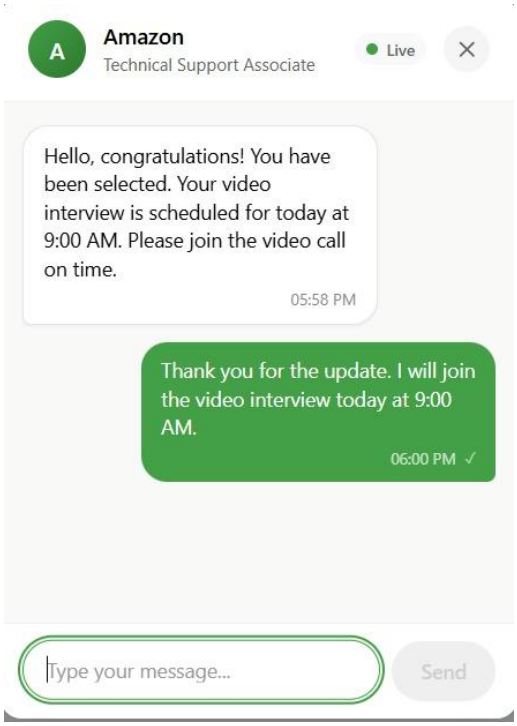


**Figure 6.13 : User Chat**

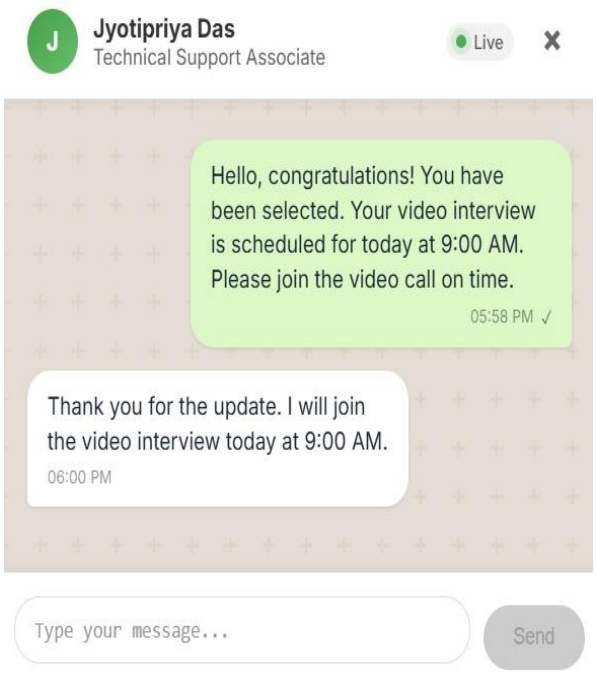


**Figure 6.12 : Recruiter Chat**

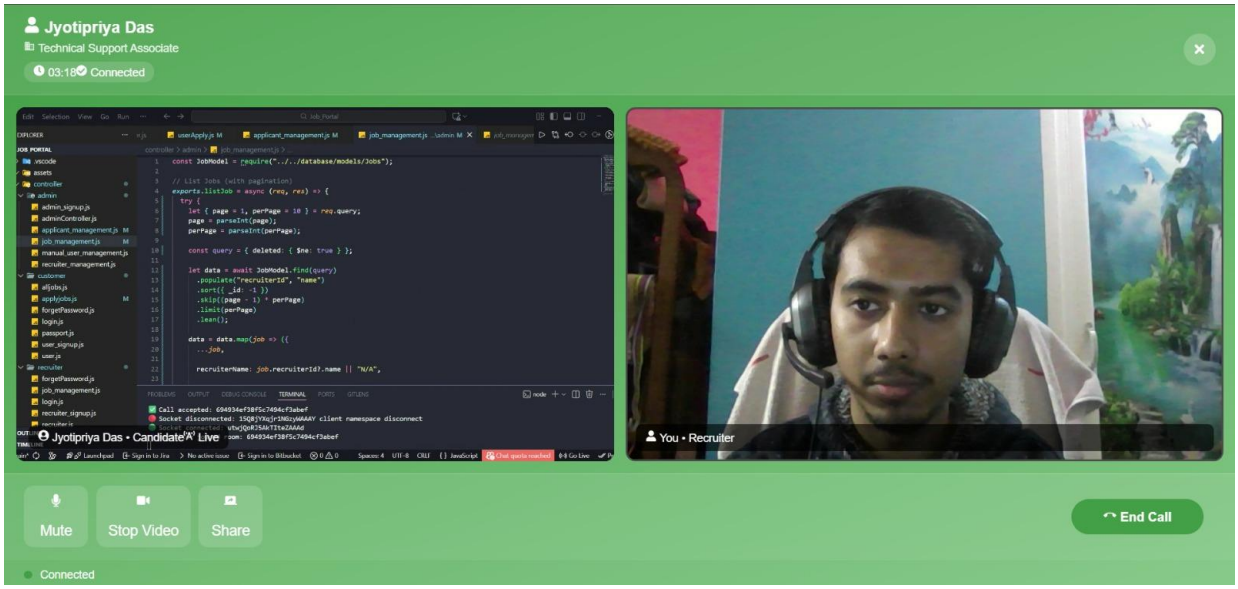


**Figure 6.14 : Video Call**

---

## 6.3 Backend Implementation

The backend is developed using Node.js and Express.js. It handles all server-side operations and business logic.

### Backend Responsibilities

- Handling user authentication

- Managing job postings and applications
- Processing API requests
- Managing chat and video signaling
- Communicating with the database

RESTful APIs are used to connect the frontend with the backend.

---

## 6.4 User Authentication and Authorization

Authentication is implemented using a secure login mechanism.

Implementation Details:

- Passwords are encrypted before storage
- Role-based access control is applied
- Tokens are used to manage user sessions
- Unauthorized access is restricted

This ensures secure access to system resources.

---

## 6.5 Database Implementation

MongoDB is used as the backend database.

**Data Storage Details**

- User credentials and roles
- Job seeker profiles
- Job postings
- Job applications
- Chat messages

Collections are designed to minimize data redundancy and support fast data retrieval.

---

## 6.6 Job Posting and Application Module

Recruiters can post job openings by providing job details such as title, description, and requirements.

Implementation Steps:

- Job data is submitted from frontend
- Backend validates and stores job details
- Job seekers can search and apply for jobs
- Application status is updated in the database

## 6.7 Chat Communication Implementation

The chat feature allows direct communication between job seekers and recruiters.

Key Features:

- One-to-one private messaging
- Real-time message delivery
- Messages stored in database
- Simple and user-friendly interface

This feature improves communication efficiency and reduces dependency on external platforms.

---

## 6.8 Video Interview Implementation

The video interview feature enables remote interviews.

Implementation Overview:

- Peer-to-peer video communication
- Real-time audio and video streaming
- Used for interview sessions

This module allows recruiters to conduct interviews directly through the platform.

---

## 6.9 Error Handling and Validation

Proper validation and error handling are implemented to ensure system stability.

Examples:

- Input validation for forms
- Error messages for invalid actions
- Server-side exception handling

---

## 6.10 Conclusion

The implementation phase successfully converts the system design into a functional application. The Smart Job Portal provides secure authentication, efficient job management, and real-time communication features, making it suitable for real-world use.

# CHAPTER 7

# SYSTEM TESTING

## 7.1 Introduction

System testing is an essential phase of software development that ensures the application works as expected and meets all specified requirements. The main objective of testing is to identify errors, verify system functionality, and ensure reliability before final deployment. This chapter describes the testing approach used for the Smart Job Portal and presents the test cases executed during development.

---

## 7.2 Testing Strategy

The testing process was carried out at different levels to ensure that each module of the system functions correctly. Both manual and functional testing methods were used during development.

The following types of testing were performed:

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing

---

## 7.3 Unit Testing

Unit testing focuses on testing individual components or modules of the system.

Examples:

- Login and registration module
- Job posting module
- Job search functionality
- Chat message handling

Each module was tested independently to ensure it worked correctly before integration with other modules.

---

## 7.4 Integration Testing

Integration testing checks the interaction between different modules of the system.

Key areas tested:

- Frontend and backend communication
- Database integration
- Authentication with protected routes
- Chat module with user accounts

This testing ensured smooth data flow between system components.

---

## 7.5 System Testing

System testing evaluates the complete system as a whole.

Test scenarios included:

- User login with valid and invalid credentials
- Job posting and application flow
- Chat communication between users
- Video interview functionality
- Admin operations

The system was tested in a real usage environment to verify overall performance.

---

## 7.6 User Acceptance Testing (UAT)

User acceptance testing was conducted to verify that the system meets user expectations.

Participants:

- Sample job seekers
- Sample recruiters

Feedback confirmed that the system was easy to use and fulfilled recruitment requirements.

---

## 7.7 Sample Test Cases

| Test Case ID | Description | Input | Expected Output | Result |
|---|---|---|---|---|
| TC01 | User Login | Valid credentials | Successful login | Pass |
| TC02 | Invalid Login | Wrong password | Error message | Pass |
| TC03 | Job Posting | Job details | Job posted | Pass |

| | | | | |
|---|---|---|---|---|
| TC04 | Job Application | Apply job | Application submitted | Pass |
| TC05 | Chat Message | Text message | Message delivered | Pass |

---

## 7.8 Bug Tracking and Resolution

During testing, minor issues related to input validation and UI responsiveness were identified. These bugs were resolved by refining the code and improving error handling.

No major system failures were observed after fixes.

---

## 7.9 Conclusion

Testing confirmed that the Smart Job Portal performs reliably and meets all functional requirements. The system successfully handles user operations, job management, and communication features with minimal errors.

# CHAPTER 8

# RESULTS AND DISCUSSION

## 8.1 Introduction

This chapter presents the results obtained after implementing and testing the Smart Job Portal. It discusses system performance, user interaction, and overall efficiency. The goal is to evaluate how well the system meets the requirements and objectives defined in earlier chapters.

---

## 8.2 System Screens and Features

The developed portal provides separate interfaces for Job Seekers, Recruiters, and Admin.

**Job Seeker Features:**

- Secure login and registration
- Profile creation and resume upload
- Job search and application submission
- Real-time chat with recruiters
- Video interview participation

**Recruiter Features:**

- Secure login
- Job posting and editing
- Viewing and shortlisting applicants
- Chat and video interviews with candidates

**Admin Features:**

- User management
- Monitoring job postings and applications
- Content moderation

---

## 8.3 Performance Analysis

The system was tested on localhost under typical load conditions.

Observations:

- Fast response times for login, search, and messaging
- Efficient handling of multiple simultaneous chat messages
- Smooth video interview performance under normal internet conditions
- Minimal lag in user interface updates

The performance indicates that the system is reliable for real-world deployment within a local or college server environment.

## 8.4 Discussion on Real-Time Communication

The chat and video modules significantly reduce the communication gap between job seekers and recruiters. Real-time messaging allows instant clarification of queries, while video interviews remove the need for physical presence, saving time for both parties.

This real-time interaction improves user satisfaction and reduces delays in the recruitment process compared to traditional portals.

---

## 8.5 User Feedback

During testing with sample users:

- Job seekers found the portal easy to navigate
- Recruiters appreciated the direct communication channels
- Admins reported effective monitoring and control

Overall feedback confirmed that the system meets functional expectations and provides a user-friendly experience.

---

## 8.6 Advantages of the System

- Direct 1-on-1 communication without intermediaries
- Real-time messaging and video interview support
- Easy-to-use interfaces for all user roles
- Simple deployment on localhost or college server
- Secure data management using MongoDB

---

## 8.7 Limitations

- Currently, the system is web-based and does not have a mobile application
- Video quality depends on user internet speed
- System scalability is limited to local or college server deployment

---

## 8.8 Conclusion

The Smart Job Portal successfully demonstrates a secure, interactive, and user-friendly recruitment platform. It achieves faster communication between job seekers and recruiters, efficient application management, and overall improved recruitment experience compared to traditional portals.

# CHAPTER 9

# CONCLUSION AND FUTURE SCOPE

## 9.1 Conclusion

The Smart Job Portal using MERN Stack successfully addresses the limitations of traditional job portals. The system provides:

- Secure, role-based authentication for job seekers, recruiters, and admin
- Efficient job posting, search, and application management
- Direct 1-on-1 communication through chat and video interviews
- Easy-to-use, responsive interfaces built with React.js and Custom CSS
- Reliable data storage using MongoDB

The project demonstrates how modern web technologies can simplify recruitment, reduce dependency on intermediaries, and improve user satisfaction. Testing and user feedback confirmed that the portal is functional, reliable, and user-friendly.

Overall, the project meets the objectives defined at the beginning and provides a solid framework for real-world deployment in a college or small organization environment.

---

## 9.2 Future Scope

The system can be extended and improved in the following ways:

1. **Mobile Application:**
   Developing an Android/iOS app for job seekers and recruiters to access the portal on-the-go.
2. **AI-Powered Job Matching:**
   Integrating machine learning algorithms to suggest relevant jobs to candidates based on their skills and experience.
3. **Resume Parsing:**
   Automating resume parsing and keyword matching to simplify shortlisting for recruiters.
4. **Analytics Dashboard:**
   Adding analytics for recruiters and admins to monitor job postings, applications, and user activity.
5. **Enhanced Video Features:**
   Adding recording, scheduling, and screen-sharing capabilities for interviews.
6. **Multi-server Deployment:**
   Future deployment on cloud or VPS to support scalability and remote access for larger user bases.

## 9.3 Final Remarks

The Smart Job Portal demonstrates that a well-designed, full-stack web application can effectively modernize recruitment processes. The project provides a foundation for further enhancements and serves as a practical MCA-level implementation of real-time web technologies.

# ▪ REFERENCES


1. Welling, L., & Thomson, L. (2017). *PHP and MySQL Web Development*. Addison-Wesley.
2. Freeman, E., Robson, E., Bates, B., & Sierra, K. (2014). *Head First HTML and CSS*. O'Reilly Media.
3. MongoDB, Inc. (2025). *MongoDB Manual*. Available: https://www.mongodb.com/docs
4. Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media.
5. Express.js Official Documentation. (2025). https://expressjs.com/
6. React.js Official Documentation. (2025). https://reactjs.org/docs/getting-started.html
7. WebRTC Official Guide. (2025). https://webrtc.org/
8. Sommerville, I. (2016). *Software Engineering, 10th Edition*. Pearson Education.
9. Laudon, K.C., & Laudon, J.P. (2019). *Management Information Systems*. Pearson Education.
10. Online Articles:

- "Building a Real-Time Chat App with React and Node.js," GeeksforGeeks, 2024.
- "Full-Stack Development with MERN Stack," Tutorialspoint, 2025.