A

Project Report On

Smart Recruitement System

Submitted to

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES RK VALLEY

On completion of Mini Project

Submitted By

Darsi Dharani[R200097]

Pamisetty Gopika[R200320]

Under the guidence of

Mrs. S.RAJESWARI, LECTURER

Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
(catering the Educational Needs of Gifted Rural Youth of AP)
R.K Valley, Vempalli(M), Kadapa (Dist.) 516330

2024 – 2025

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
(A.P. Government Act 18 of 2008)
RGUKT-RK Valley
Vempalli, Kadapa, Andhra Pradesh – 516330

DECLARATION

we , **Darsi Dharani** and **Pamisetty Gopika** hereby declare that the project report titled **"Smart Recruitement System-A Platform for Job Seekers to Find Jobs and Train for Interviews"** was conducted under the guidance of **Mrs. S Rajeswari** at the Department of Computer Science and Engineering, RGUKT-RK Valley, during the academic session **from December 2024 to May 2025**. The project was carried out as part of my academic curriculum, and I am now submitting this report to Mrs.S Rajeswari as a requirement for the successful completion of my mini/major project.

This report is the result of my dedicated efforts, and I affirm that it has not been copied, reproduced, or borrowed from any external source. Any references to external work, resources, or literature have been duly cited and are acknowledged in the references section of this report. Furthermore, to the best of my knowledge, the findings, analysis, and results presented in this dissertation have not been submitted elsewhere to any university, institute, or organization for the fulfillment of any academic degree, course, or project requirement.

Date: _____
Place: RGUKT-RK Valley

Darsi Dharani [R200097]

Pamisetty Gopika [R200320]

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES (AP IIIT)
R.K Valley, Vempalli(M), Kadapa(Dist) – 516330

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2024-2025

CERTIFICATE

This is to certify that the project report entitled "*SMART RECRUITEMENT SYSTEM*" being submitted by **D.Dharani[R200097]** and **P.Gopika[R200320**] under my guidance and supervision and is submitted to DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING in partial fulfilment of requirements for the award of Bachelor of Technology in Computer Science during the academic year 2024-2025 and it has been found worthy of Acceptance According to the requirements of the University.

Signature of Internal Guide
S.Rajeswari
Assistant Professor
Department of CSE

Signature of HOD
Dr.Ch. Ratna Kumari
Assistant Professor
Department of CSE

Signature of External Examiner

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude & respect to all those people behind the screen who guided, inspired and helped us crown all our efforts with success. We wish to express our gratitude to **Dr.S Rajeswari** for her valuable guidance at all stages of study, advice, constructive suggestions, supportive attitude and continuous encouragement, without which it would not be possible to complete this project.

We would also like to extend our deepest gratitude & reverence to the Director of RGUKT, RK Valley **Dr. A V S S Kumara Swami Gupta** and HOD of Computer Science and Engineering **Dr. CH Ratna Kumari** for their constant support and encouragement.

Last but not least we express our gratitude to our parents for their constant source of encouragement and inspiration for us to keep our morals high.

WITH SINCERE REGARDS
DARSI DHARANI-R200097
PAMISETTY GOPIKA - R200320

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The **Smart Recruitment System with Mock Interview Training and Resume Building** is a web-based platform developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js) to streamline and enhance the job application process for candidates and employers. The project aims to go beyond conventional job portals by integrating resume creation tools, intelligent mock interview preparation, and user-friendly job browsing features—all within a secure and scalable environment.

This system addresses the limitations of traditional recruitment methods by providing a centralized platform that combines job discovery with professional development. Key features for **job seekers** include secure registration and login, job search with filters (location, salary, and role), personalized dashboards, resume builder tools, application tracking, mock interviews tailored from job descriptions, and profile progress tracking. The **Resume Builder** enables users to create, edit, and download professional resumes aligned with industry standards.

Recruiters or **companies** can create accounts to post job listings, manage applications, and review applicant profiles, while an **admin panel** ensures platform moderation and user management. The role-based access control mechanism ensures that users only access features relevant to their roles (user, recruiter, admin), enhancing security and usability.

The platform implements **JWT-based authentication**, secure route protection, and responsive UI design for a smooth user experience across devices. Backend APIs handle job posting, application submission, mock interview generation, and profile updates efficiently.

In addition to job search and application features, the platform emphasizes **career readiness**. Mock interviews offer users a way to prepare for real-world interviews, and resume generation ensures users present themselves professionally.

In conclusion, this project bridges the gap between education and employment by offering not just job listings but also preparation tools within a single system. Future enhancements may include AI-driven job recommendations, video interview simulations, and mobile app deployment. The Smart Recruitment System provides a complete and practical solution for modern hiring needs, empowering users to achieve better employment outcomes.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The job recruitment process can often be complex, time-consuming, and intimidating for candidates, especially those who lack the necessary interview preparation or fail to present their qualifications effectively through resumes. Furthermore, employers face challenges in efficiently managing applications, reviewing candidates, and finding the best fit for their job openings. While many job portals exist, they lack comprehensive tools for candidates to prepare for interviews, build customized resumes, and track application progress in a unified platform.

The current solutions do not provide a seamless experience that connects job seekers with tailored career development resources and opportunities. There is a clear need for a web-based platform that streamlines the job application process, offers mock interview training, and allows users to create optimized resumes while ensuring secure access and smooth interaction between candidates and employers.

The Smart Recruitment System with Mock Interview Training and Resume Building seeks to address these challenges by creating a unified system that enhances both the job search experience for candidates and the recruitment process for employers, all while ensuring data privacy and security through modern authentication and role-based access control.

1.2 Scope

The Smart Recruitment System with Mock Interview Training and Resume Building is designed to provide a comprehensive platform for both candidates and employers. It focuses on enhancing the job search experience, empowering candidates with the tools they need to prepare for interviews and create professional resumes while offering companies an efficient platform to manage job applications. The project scope includes:

- User Authentication: Secure login and registration with token-based authentication.
- Candidate Features: Job search, application management, resume builder, and mock interview training.
- Employer Features: Job posting, application management, and candidate review.
- Admin Panel: User and platform management, including analytics and oversight.
- Frontend: Developed using React.js for a responsive and intuitive user interface.
- Backend: Node.js + Express.js to handle routing and logic.
- Database: MongoDB for storing user profiles, job listings, and applications.
- Security: Ensuring secure authentication, data encryption, and role-based access control.

- Scalability: Future enhancements could include AI job matching and advanced analytics.

1.3. Technological Impact

The Smart Recruitment System brings several technological advancements to the recruitment process, enhancing the efficiency, accessibility, and security of hiring. The impact includes:

Digital Transformation: Shifts traditional hiring processes to a digital platform, adapting to the modern needs of job seekers and employers.

Cost Efficiency: Reduces costs related to paper resumes, recruitment agencies, and manual interview scheduling, streamlining the hiring process.

Real-Time Operations: Job applications, resume reviews, and mock interview feedback are processed and delivered in real-time, increasing speed and accuracy.

Wider Participation: Encourages participation from candidates across various regions, including remote areas, students, and working professionals.

Environmentally Friendly: Minimizes the use of paper and physical resources, supporting eco-friendly, sustainable hiring practices.

1.4. Importance of the Smart Recruitment System

The adoption of the Smart Recruitment System offers numerous advantages for candidates, employers, and the overall recruitment process. Some of the major reasons highlighting its importance are:

Convenience: Candidates can apply for jobs, update profiles, and prepare for interviews from anywhere, saving time and effort.

Inclusiveness: Supports participation from diverse groups, including remote workers, differently-abled individuals, and those with busy schedules.

Reduced Errors: Automating resume creation, job applications, and interview scheduling minimizes human error and improves accuracy.

Security & Privacy: With secure authentication and encrypted data storage, the system ensures confidentiality and integrity in the recruitment process.

CHAPTER 2

LITERATURE SURVEY

2.1. Study of Existing Recruitment Platforms

Several recruitment platforms exist today that aim to simplify the hiring and job application process. These systems often provide basic functionalities such as user registration, job postings, and notifications. Some of the well-known solutions include:

- **Superset** – A cloud-based platform widely used for automating placements and internships, especially in campuses.
- **Custom Recruitment Portals** – Many organizations and educational institutions build their own in-house systems for managing recruitment internally.
- **ERP-Based Modules** – Some enterprise resource planning (ERP) systems offer placement and recruitment tracking as part of their functionality.

While these platforms address some aspects of the recruitment lifecycle, they often come with limitations such as high costs, lack of customization, complex interfaces, or poor scalability, especially for mid-sized institutions and companies. Furthermore, many of them do not offer integrated career development tools such as dynamic mock interview training or intelligent resume building, which are essential for holistic candidate preparation.

2.2. Traditional Methods

Traditionally, recruitment and career preparation involved manual processes that were time-consuming and less efficient. Key practices included:

- **Job Search:**
  Candidates relied on newspapers, referrals, or college notice boards to find job openings.
- **Application Submission:**
  Resumes and cover letters were submitted physically, often by mail or hand-delivery to companies.
- **Campus Recruitment:**
  Training and Placement Officers coordinated with companies through phone calls, letters, or direct visits without a centralized platform.
- **Resume Preparation:**
  Candidates created resumes individually, with no standard formats or industry-specific guidelines.

- **Interview Preparation:**
  Mock interviews were rare and usually arranged informally with seniors, friends, or local coaching centers.
- **Communication and Updates:**
  Updates regarding applications and interview results were slow, leading to uncertainty and delays.

These traditional methods lacked real-time updates, personalization, scalability, and easy access to career development resources, highlighting the need for a more efficient, digital solution.

2.3. Advantages of the Proposed System

The **Smart Recruitment System with Mock Interview Training and Resume Building** offers several advantages over traditional methods:

- **Centralized Platform:**
  Candidates, companies, and administrators can interact on a single, unified platform, streamlining the entire recruitment process.
- **Dynamic Mock Interview Preparation:**
  Personalized mock interview questions are generated based on job descriptions, helping candidates prepare more effectively.
- **Resume Building Tool:**
  Users can create professional, customized resumes aligned with industry standards, improving their chances of selection.
- **Real-Time Notifications:**
  Candidates receive instant updates about application status, interview schedules, and preparation tips.
- **Secure Authentication and Role Management:**
  The platform ensures secure login, data protection, and access control based on user roles (candidate, company, admin).
- **User-Friendly Interface:**
  A responsive and intuitive design makes it easy for users to navigate and manage their profiles and applications.
- **Scalability and Flexibility:**
  Built with the MERN stack, the system can handle growing numbers of users, job postings, and applications efficiently

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Functional Requirements

• User Authentication:

- Users can register and log in using email and password.
- Role-based access control is implemented (e.g., job seekers vs. employers).
- JWT (JSON Web Token) is used for secure, stateless authentication.
- Passwords are securely hashed before being stored.

• Profile Management:

- Users can update their profile details, including education, skills, and resume.
- Profile photo upload and removal functionality is supported.
- A profile completion percentage is displayed to guide users.

• Job Management (Employers):

- Employers can post jobs with details like title, description, location, salary, start and end dates, and number of positions.
- Jobs can be updated or soft deleted (isDeleted: true) without removing data permanently.

• Job Applications (Job Seekers):

- Users can apply for jobs directly from job listings.
- Users can view job details, remaining application days, and positions left.
- Duplicate applications are prevented.
- Users can delete previously applied jobs.

• Save for Later:

- Job seekers can save jobs to view/apply later.
- Save status persists across sessions and updates visually in the UI.

• Mock Interview:

- Job seekers can take mock interviews to prepare for real ones.

- Includes standard questions and a way to practice answering under simulated conditions.

• Search and Filter:

- Jobs can be searched by location, industry (role), and salary range.
- Filters are dynamically populated based on actual job data in the database.

3.2 Performance Requirements

- **Response Time:**

  All user actions (applying, saving jobs, updating profiles) should respond in under 2 seconds.

- **Scalability:**

  The platform should handle up to 5,000 concurrent users with smooth experience.

- **Database Efficiency:**

  Indexed fields (like location, industry, salary) ensure optimized query performance.

- **High Availability:**

  Hosted on reliable cloud services with automatic backups and failover mechanisms.

- **Error Handling and Logging:**

  Centralized logging for server errors and user activity is maintained for monitoring and debugging.

3.3 Technical Requirements

• Frontend:

- React.js for building the UI.
- Tailwind CSS for responsive design.
- React Router for navigation.
- Axios for API requests.

• Backend:

- Node.js with Express.js to handle server-side operations.
- MongoDB with Mongoose for database management.
- JWT for authentication.

- Multer for handling profile photo uploads.
- Nodemailer (optional) for account-related emails.

• **Hosting & Deployment:**
- Frontend deployed on **Vercel** or **Netlify**.
- Backend and database hosted via **Render** or **MongoDB Atlas**.

• **Version Control:**
- Git for tracking changes.
- GitHub for repository management and collaboration.

• **Testing:**
- Postman for API testing.
- Manual testing for frontend feature validation.

3.4 Non-Functional Requirements

• **Usability:**
- Simple and intuitive UI for both job seekers and employers.
- Clear call-to-action buttons and form feedback.

• **Security:**
- Passwords hashed using bcrypt.
- JWT tokens for secure route access.
- CORS enabled for safe frontend-backend communication.

• **Maintainability:**
- Code is modular and separated by concerns (routes, controllers, models, etc.).
- Detailed documentation is provided for APIs and environment setup.

• **Reliability:**
- Soft delete ensures data recovery.
- Resilient error handling prevents crashes and protects user actions.

• **Scalability:**
- API and database designed to handle growing number of users, jobs, and applications efficiently.

3.5 Third – Party Packages and Libraries

Frontend:

1. React Router:

- **Purpose:** React Router enables dynamic routing in React applications, allowing users to navigate between different views (e.g., Home, Job Listings, Login, Profile, etc.) without reloading the entire page.
- **Why Used:** It simplifies navigation within the single-page application (SPA) and efficiently manages routes for user-specific pages like job applications, saved jobs, and user dashboards.

2. React Hook Form:

- **Purpose:** A form handling library that works with React's hooks API to manage forms with minimal re-renders and high performance.
- **Why Used:** Ideal for managing forms such as login, registration, job application, and profile update, ensuring faster validation and better UX.

3. React Toastify:

- **Purpose:** React Toastify is used to display toast notifications for events like success, error, or info messages.
- **Why Used:** Provides real-time feedback for user actions like "Login Successful," "Job Applied," "Profile Updated," or error messages, enhancing user interaction and clarity.

4. Formik:

- **Purpose:** Formik is another popular form-handling library that manages form state, validation, and submission.
- **Why Used:** Useful for handling more complex forms such as resume submission or admin-level job creation, where detailed validations and dynamic fields are needed.

5. React Select:

- **Purpose:** A customizable dropdown component for React with features like search, multi-select, and async options.
- **Why Used:** Enhances filtering in job listings based on location, industry, roles, or salary range, providing a modern and intuitive user experience.

Backend:

1. Bcrypt.js:

- **Purpose:** Bcrypt is a hashing library used to securely hash passwords before saving them in the database.
- **Why Used:** Ensures that passwords are never stored as plain text, adding a critical layer of security to the user authentication system.

2. JWT (jsonwebtoken):

- **Purpose:** A library used for creating and verifying JSON Web Tokens (JWT) for stateless authentication.
- **Why Used:** Used to manage user sessions securely by issuing tokens on login and verifying them on protected routes such as applying to a job or updating profiles.

3. CORS (Cross-Origin Resource Sharing):

- **Purpose:** Middleware that enables cross-origin requests between different domains or ports.
- **Why Used:** Necessary to allow the frontend React app to communicate securely with the backend Express server during development and deployment.

4. Joi / Express-validator (choose one used in your project):

- **Purpose:** Libraries used to validate incoming data on the backend to ensure it meets defined formats and constraints.
- **Why Used:** Prevents invalid or malicious data from entering the system by validating user input during registration, job posting, and profile updates.

5. Mongoose:

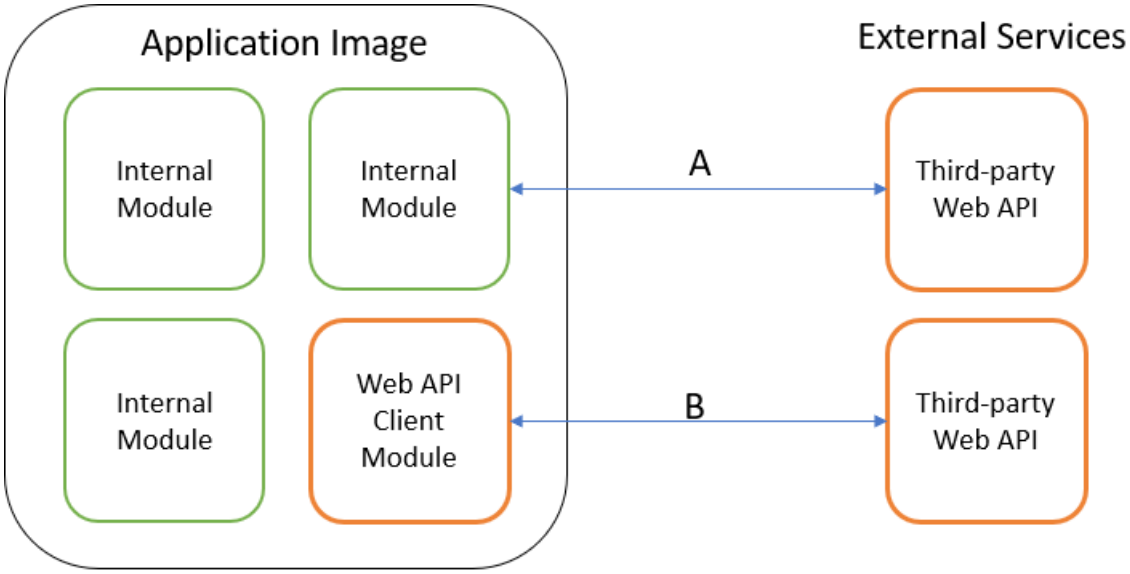
- **Purpose:** Mongoose is an ODM (Object Data Modeling) library that provides a schema-based solution to model application data in MongoDB.
- **Why Used:** Simplifies interaction with the MongoDB database, enforces schemas for collections (users, jobs, applications), and offers built-in data validation.

6. Dotenv:

- **Purpose:** A zero-dependency module that loads environment variables from a .env file into process.env.
- **Why Used:** Keeps sensitive credentials (e.g., MongoDB URI, JWT secret) secure and out of the main codebase, aiding in configuration management.

7. Multer:

- **Purpose:** A middleware for handling multipart/form-data, typically used for uploading files.
- **Why Used:** Enables users to upload profile pictures and resumes, which are stored either on the server or in a cloud storage system, enhancing the user's job application experience.



CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

Introduction

The successful development of a Smart Recruitement System requires a well-structured, modular, and scalable approach to ensure seamless functionality, high performance, and ease of use for both job seekers and employers. This chapter provides an in-depth explanation of the system's architecture, design principles, and development methodology. The primary objective was to build a reliable, responsive, and secure platform for managing job listings, applications, company registrations, and user profiles with role-based access control and real-time feedback.

The system was strategically divided into the following core modules to manage complexity, enhance maintainability, and enable collaborative development:

- **Module 1: Requirement Analysis and Database Design**
- **Module 2: Authentication and Role-Based Access Control**
- **Module 3: Job Posting and Application System**
- **Module 4: Frontend Development**
- **Module 5: Backend Development**
- **Module 6: Profile Management and Resume Handling**
- **Module 7: Job Filters and Search Optimization**
- **Module 8: Mock Interview System**
- **Module 9: UI/UX Enhancements and Notifications**

Each module was key in building a Smart Recruitment System for both job seekers and recruiters. From authentication to applications and mock interviews, the system

ensures usability, scalability, and responsiveness. This chapter outlines each module's purpose, implementation, and challenges.

Module 1: Requirement Analysis and Database Design

Objectives:

- Understand user requirements for job seekers and employers.
- Identify key entities, relationships, and attributes.
- Design an efficient and scalable database schema using MongoDB.

Implementation:

The requirement analysis phase involved gathering both functional and non-functional needs of the job portal system. We identified two primary user roles: **Job Seekers** and **Recruiters**, and mapped their interactions with the platform. Core features were derived, including:

- Job posting by companies,
- Job application by users,
- Resume uploads and downloads,
- User profile and company profile management,
- Interview scheduling.

The database design centered around a **document-oriented approach using MongoDB**. Key collections created include:

- User: storing personal data, role, resume, profile picture, saved jobs, and applications.
- Company: storing company profiles, job postings, and recruiter accounts.
- Job: detailing job information such as title, salary, location, deadline, and open positions.
- Application: linking User and Job with timestamps and status (applied, shortlisted, rejected, etc.).
- Interview: storing mock interview records, questions, answers.

Relationships were maintained through **ObjectId references** between documents, and isDeleted flags were used for **soft deletion** of companies and jobs.

UI/UX Features:

- Dynamic rendering of job and company data from the backend schema.
- Clean input forms for job posting, profile updates, and resume upload that align with backend validation.
- Logical organization of dashboards based on role-specific data (e.g., recruiter sees posted jobs; job seeker sees applied jobs).
- Form validation reflecting schema rules like required fields, data types, and constraints (e.g., deadlines, LPA format).

Tools: MongoDB, Mongoose

Module 2: Authentication and Role-Based Access Control

Objectives:

- Implement secure authentication.
- Define roles: Admin, Job Seeker, Recruiter.
- Restrict access based on roles.

Implementation:

The authentication module was designed to ensure secure login and registration for users. **JWT-based authentication** was implemented to handle session management. During registration, user passwords were hashed using **bcrypt** to ensure that sensitive information was securely stored in the database. Upon successful login, a **JWT token** was generated and stored in **HTTP-only cookies** to prevent cross-site scripting (XSS) attacks.

Role-based middleware was integrated to restrict route access depending on the user's role. This involved creating role checks for the three primary user types: **Admin**, **Job Seeker**, and **Recruiter**. Each route was protected by middleware that verified the JWT token and ensured the user had the necessary role to access specific resources or perform certain actions. For example:

- Admins had access to manage users and view all job applications.
- Recruiters could post jobs, manage applications, and view their specific company profile.
- Job Seekers had limited access to apply for jobs and manage their profiles.

UI/UX Features:

- Clear error and success messages for login and registration flows (e.g., "Invalid credentials", "Account created successfully").
- Login form with proper field validation (e.g., required fields, password length).
- Redirects after successful login based on role (e.g., admins go to an admin dashboard, job seekers go to job listings).
- Informative UI updates during authentication (e.g., loading indicators while tokens are being validated).

Tools:

Node.js, Express.js, bcrypt, JWT, Postman (for testing API endpoints)

Module 3: Job Posting and Application System

Objectives:

- Allow recruiters to post jobs.
- Enable seekers to apply for jobs.
- Track and manage applications.

Implementation:

This module allows recruiters to create detailed job postings, including fields such as **title**, **description**, **salary**, **location**, and **application deadline**. Recruiters can manage these postings through a simple and intuitive interface, with all job data stored in the database.

Job seekers can browse job listings, view detailed descriptions, and submit applications. Each job application is associated with a **job ID** and **user ID**, ensuring that the application can be linked to both the job posting and the job seeker.

The system tracks application statuses, allowing recruiters to see whether an applicant has been reviewed, accepted, or rejected. This system also supports dynamic filtering based on job details such as location, salary, and industry.

Additionally, recruiters have the ability to update the status of an application, which is then reflected in real-time on the job seeker's dashboard.

UI/UX Features:

- **Job Posting UI**: A form for recruiters with clear field labels for easy entry of job details.

- **Job Listings UI**: A searchable and sortable list of job postings.
- **Job Application UI**: A simple, one-click application process for job seekers, displaying a "Apply" button for each listing.
- **Application Status**: Job seekers are notified when their application status changes.

Tools:

MongoDB, Express.js, Node.js, Mongoose, Axios, React.js

Module 4: Frontend Development

Objectives:

- Build responsive user interfaces for all roles.
- Provide intuitive navigation and clear visual responses to user actions.

Implementation:

The frontend of the Smart Recruitment System was built using **React.js** to create a highly interactive and responsive user interface. Each page was carefully crafted to cater to the needs of different user roles (job seekers, recruiters, and admins). The application was structured into reusable and modular components to ensure easy maintainability.

Pages developed included:

- **Home**: A welcoming page with basic information and job search functionality.
- **Login/Register**: Forms for users to register and log in to the system.
- **Job Listings**: A page that dynamically displays job posts with filters to narrow down search results based on job type, location, salary, etc.
- **Job Details**: A detailed view of each job post, including descriptions, salary range, and application options.
- **User Dashboard**: A personalized area where job seekers can manage their profiles, view saved jobs, and track application status.
- **Resume Builder**: An integrated feature to help job seekers build and download their resumes in PDF format.

Axios was used for API communication, allowing the frontend to interact with the backend server efficiently. **React Router DOM** was implemented to manage routing and navigation within the application, ensuring smooth transitions between pages. **Protected**

routes and **conditional rendering** ensured that the correct views were shown to users based on their authentication status and role.

UI/UX Features:

- **Responsive Design**: Tailwind CSS was used to create a mobile-first, responsive design for smooth viewing on all screen sizes.
- **Intuitive Navigation**: Easy-to-navigate layout with a clear flow between pages. The use of **React Router DOM** ensured that navigation was seamless.
- **Dynamic Feedback**: Modals, loading indicators, and visual feedback to improve the user experience when interacting with the platform (e.g., submitting an application, saving a job, etc.).
- **User Role-based Views**: Different views for job seekers, recruiters, and admins, with appropriate information and controls based on user roles.

Tools: React.js, Axios, React Router DOM, Tailwind CSS

Module 5: Backend Development

Objectives:

- Develop a RESTful API for all operations.
- Ensure modular and maintainable server code.

Implementation:

The backend of the Smart Recruitment System was built using **Node.js** and **Express.js**, providing a robust foundation for handling API requests. A modular approach was adopted by separating concerns into controllers, models, and routes, ensuring maintainable and scalable server-side code.

The core features implemented included:

- **Authentication**: User registration, login, and session management were handled using **JWT (JSON Web Tokens)** for secure, stateless authentication.
- **Job Posting**: Recruiters can create, update, and delete job posts. Job seekers can view job posts and apply for them.
- **Applications**: Job applications were managed by linking user IDs to job IDs. The backend tracks application statuses and enables users to update or withdraw their applications.

- **Profile Management**: Users can update their profile information, including personal details and uploaded resumes.

Additional features included:

- **Soft Delete**: A "soft delete" system was used for job posts and user profiles, where data was marked as deleted rather than being permanently removed, ensuring that related data remains intact for auditing purposes.
- **Pagination**: Pagination was implemented in job listings and applications to ensure that large datasets could be efficiently handled and displayed.

UI/UX Features:

- **API Responses**: All API responses were designed to be consistent and meaningful, with clear success and error messages to guide users.
- **Performance Optimization**: Pagination, efficient querying, and other performance considerations ensured that the application scaled well with an increasing amount of users and data.

Tools: Node.js, Express.js, Mongoose

Module 6: Profile Management and Resume Handling

Objectives:

- Allow users to manage profile data.
- Enable upload and download of resumes.

Implementation:

In this module, users were given the ability to manage and update their profile information. The following features were implemented to enhance the user experience:

- **Profile Information Update**: Users could update personal details such as name, contact information, skills, and experiences.
- **Profile Picture Upload**: A system for uploading and updating profile pictures was integrated using **Multer**. This functionality allowed users to select an image file from their device, which was then uploaded to the server and associated with their profile.
- **Resume Builder**: A custom resume builder was developed using **jsPDF**, allowing users to create professional resumes directly within the platform. The resume

template was designed to closely resemble traditional CV formats, with sections for education, work experience, skills, and other relevant information. Users could easily fill in these fields and generate downloadable PDF versions of their resumes.

The system provided a seamless process for both updating profile information and handling resumes. It was designed to be user-friendly, with clear instructions and feedback during the resume creation process.

UI/UX Features:

- **Editable Profile Page**: Users could update their profile information in a clean, easy-to-use interface, which dynamically updated the page as changes were made.
- **Profile Picture Upload**: The profile picture upload feature provided a straightforward way for users to select and upload their image. The interface allowed users to preview their picture before finalizing the upload.
- **Resume Download**: After generating the resume, users could easily download their resume in PDF format. The download button was clearly visible and provided an immediate action after completing the resume.

Tools: React, jsPDF, Multer (for image uploads)

Module 7: Job Filters and Search Optimization

Objectives:

- Improve job discovery through filters.
- Implement fuzzy search and relevance sorting.

Implementation:

This module aimed to optimize the job search process by allowing users to filter jobs based on specific criteria and enhancing the search experience with fuzzy search techniques. Key features included:

- **Job Filtering**: Users could filter jobs based on criteria such as **location**, **salary range**, **industry**, and **role**. These filters provided dynamic search options, allowing users to narrow down their results effectively.
- **Relevance Sorting**: Search results were sorted by relevance, ensuring the most suitable jobs were displayed first. The relevance sorting algorithm took into account factors like keyword matching, job posting date, and user preferences.

- **Saved Jobs and Applied Jobs**: Jobs that users had saved or already applied for were visually marked, making it easier for them to track their job search and avoid redundant applications.

The search optimization ensured that users found relevant job listings faster and more efficiently, improving the overall experience of job discovery on the platform.

UI/UX Features:

- **Dynamic Search Bar**: The search bar was integrated with real-time suggestions as users typed, making it easier to find relevant jobs quickly.
- **Filter Sidebar**: A sidebar for filtering job search results was implemented with intuitive sliders for salary range, checkboxes for job roles, and location dropdowns. This ensured a smooth and customizable search process.
- **Job Result Highlighting**: Jobs that had been saved or already applied for were visually marked with clear labels or icons, providing immediate feedback to users.

Tools: MongoDB query filters, Regex search, useEffect optimization in React

Module 8: Mock Interview System

Objectives:

- Simulate job interview scenarios.
- Provide feedback to users.

Implementation:

The mock interview module was designed to simulate a real interview experience, helping users practice answering common interview questions and receive constructive feedback. Key features included:

- **Randomized Questions**: Users were presented with a set of randomly chosen interview questions, ensuring they received diverse practice and exposure to different types of questions.
- **Answer Tracking**: As users answered each question, their responses were stored and linked to their user profile. This allowed users to review their past answers and track their progress over time.
- **Interview Scenarios**: The module could simulate various types of interviews, such as technical interviews, HR interviews, or situational interviews, based on the user's career goals and job roles.

The mock interview system was aimed at helping users enhance their interview skills and increase their chances of success in real-world job interviews.

UI/UX Features:

- **Question Display**: The system displayed each interview question clearly, with a countdown timer to simulate real-time pressure during the interview.
- **Answer Submission**: Users could type their answers into a text box and submit them. A rich text editor was included for formatting answers where needed.
- **Feedback Modal**: Once the interview was complete, users received feedback in a modal window, providing suggestions on areas for improvement.
- **Progress Tracking**: Users had access to a dashboard displaying their past mock interview performance, allowing them to track improvements over time.

Tools: Custom API, MongoDB, React forms

Module 9: UI/UX Enhancements and Notifications

Objectives:

- Improve visual appeal and interactivity.
- Provide notifications and alerts for key actions.

Implementation:

The UI/UX enhancements aimed to create a visually pleasing, interactive, and user-friendly interface. Key features of the UI improvements included:

- **Animations**: Smooth transitions and animations were implemented using Framer Motion to create a dynamic experience for users. These animations helped improve the overall user engagement and make interactions feel more intuitive.
- **Responsive Layouts**: The design was fully responsive, ensuring that the platform provided an optimal experience across various screen sizes, including desktops, tablets, and smartphones. Tailwind CSS was leveraged to implement a fluid grid system, allowing elements to adapt and reflow based on the screen size.
- **Modals**: Modals were used for critical actions such as confirming job applications, displaying alerts, or updating profile information. The modal dialogs were designed to ensure the user could easily confirm or cancel their actions.
- **Toast Notifications**: React Toastify was used to implement toast notifications that notify users about key actions, such as successful job applications, profile updates, or system errors. These notifications were non-intrusive, appearing briefly at the top of the screen without interrupting the user's workflow.

These enhancements ensured that users had a smooth, engaging experience while navigating through the system. The combination of animations, responsive design, and notifications allowed for a more polished and professional user interface.

UI/UX Features:

- **Interactive Feedback**: Animations were incorporated to show progress, such as during job applications or profile updates, making the interface feel more interactive.
- **Toast Notifications**: Immediate feedback was provided to users through non-blocking toast messages, such as "Job Application Successful," "Profile Updated," and "Error: Something Went Wrong."
- **Modal Alerts**: For critical actions like confirming job applications, users were presented with modals to ensure they were making intentional decisions.

Tools: React Toastify, Tailwind CSS, Framer Motion

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

The Job Portal System was developed to streamline the process of job searching, application, and recruitment for students and employers. Built using the MERN (MongoDB, Express.js, React.js, Node.js) stack, the system integrates a user-friendly interface, secure authentication, role-based access control, and dynamic data handling. This chapter outlines the key results of the portal's development, focusing on functionality, performance, user interactions, and feedback. It also discusses the platform's current impact, areas for refinement, and future scope.

5.2 System Performance and Functionality

The system was tested extensively for its core modules including job posting, job application, resume management, job filtering, and mock interviews. The platform demonstrated stable performance under multiple user roles (Job Seeker and Recruiter), achieving its intended outcomes.

Authentication and Access Control

The authentication system, based on JWT and bcrypt hashing, provides secure login and signup features. Role-based access ensures that job seekers and recruiters only access relevant functionalities. Job seekers can browse and apply for jobs, while recruiters can post jobs and manage applications. The system preserves login state and offers redirection based on roles after authentication.

Job Posting and Application System

Recruiters are able to post jobs with details like title, description, salary, role, location, number of openings, and start/end dates. Job seekers can view job cards, filter jobs, apply to open positions, and track application status. Application deadlines are dynamically calculated and shown as "X days left to apply," while filled positions are auto-marked as unavailable.

Resume Upload and Profile Management

Job seekers can build a profile with personal, educational, and professional details. The resume handling module allows PDF uploads and downloads, with real-time rendering and formatting. A profile completion indicator encourages users to fill out all fields, enhancing their visibility to recruiters.

Job Filtering and Search

The filtering system allows dynamic search based on location, industry (role), and salary. It uses structured queries and updates in real time, significantly improving the job search experience. Job seekers can also save jobs for later, with visual indicators to reflect saved state across navigation.

Mock Interview System

A mock interview module was implemented to simulate interview questions for job seekers. Questions are categorized by topic, and user answers can be stored for review. This module aims to increase user preparedness and confidence before real interviews.

Notifications and UI Feedback

The system uses toast notifications and modals to guide users through actions such as successful applications, login/logout, or profile updates. Though user-submitted feedback forms were not implemented, the UI gives responsive visual cues for loading, errors, and success states.

5.3 Usability Testing and User Insights

Positive Feedback

Usability testing was conducted with students and mock recruiters. Most users found the portal clean, responsive, and easy to use. Features like “days left to apply,” real-time search filtering, resume preview/download, and saved jobs were frequently appreciated. The role-based redirection and clean dashboard views for both roles were also praised.

Areas for Improvement

Some users suggested adding:

- Sorting options (by most recent, highest salary, etc.) in job listings
- Social media login for quicker onboarding
- More industry-specific mock interview questions
- In-app messaging between job seekers and recruiters
- Status notifications when a recruiter accepts or rejects an application

5.4 Technical Challenges and Solutions

- **Role Management:**

  Maintaining separate functionalities and UIs for job seekers and recruiters required conditional rendering and robust access middleware. This was solved using role tags in JWT and protected routes in both frontend and backend.

• **Resume Handling and PDF Export**

Generating clean PDF resumes while preserving user formatting posed a challenge. This was addressed using jsPDF with custom styles to ensure consistent output.

• **State Management of Saved Jobs:**

Keeping the saved state of jobs consistent across pages was a challenge due to routing and cache issues. This was resolved using React state management along with backend checks to fetch user-saved job IDs and update the UI on component load.

5.5 Future Enhancements and Improvements

1. **Advanced Analytics for Recruiters**
   Enable recruiters to view statistics on how many users viewed, saved, or applied to a job posting.

2. **In-App Messaging System**
   Allow job seekers and recruiters to communicate directly within the portal, improving response time and engagement.

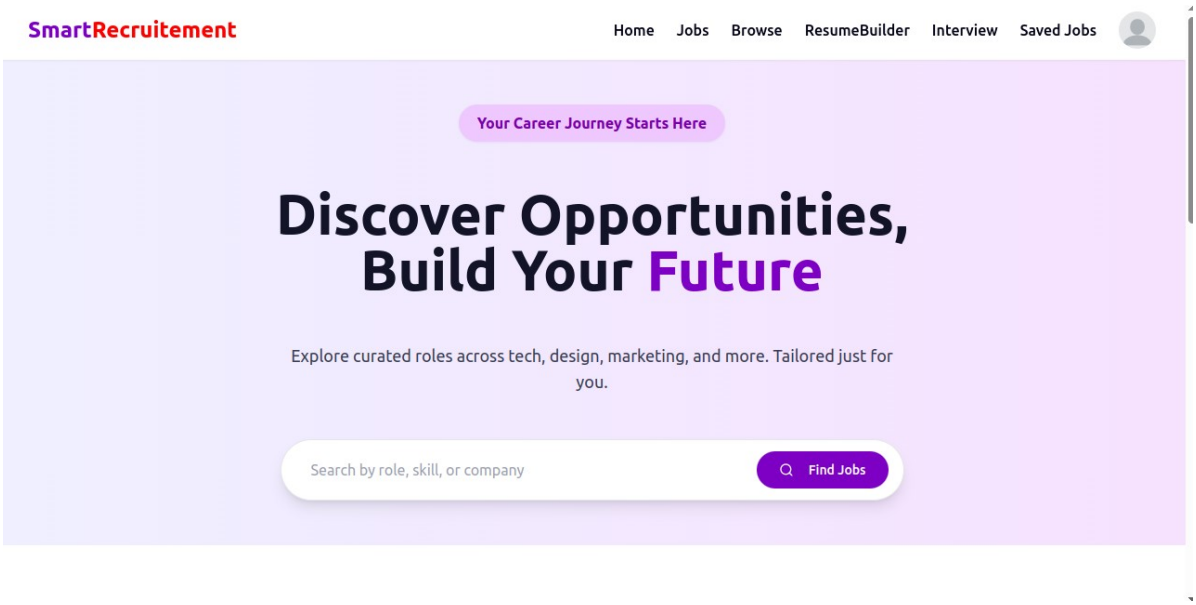
3. **Interview Scheduling and Calendar Integration**
   A calendar tool to schedule interviews and notify users via email or in-app alerts.

4. **Admin Panel and Reporting**
   Add an admin role to monitor flagged content, view usage metrics, and manage users for platform integrity.

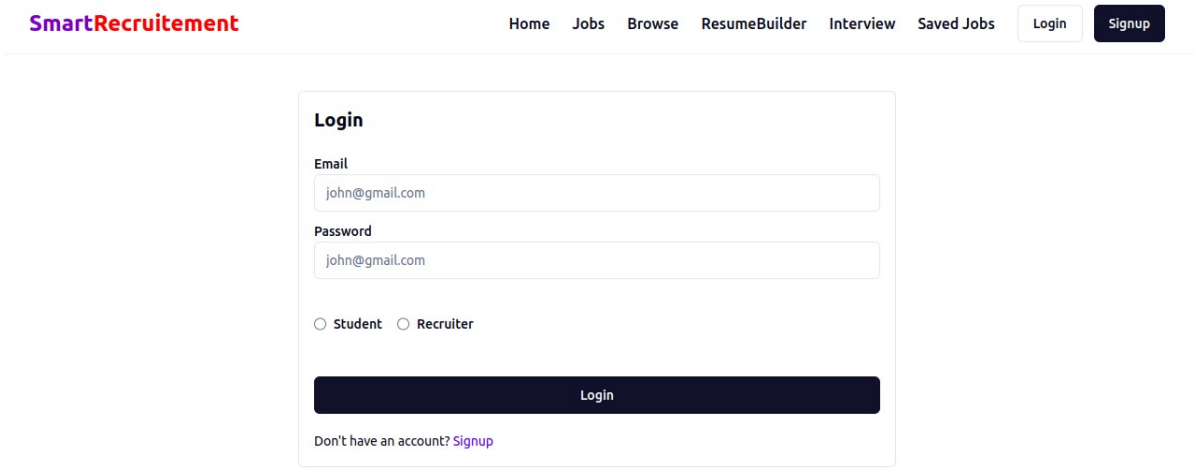
5. **Mobile App Version**
   A mobile-friendly app with push notifications for job updates, interview calls, and application status.

5.6 UI/UX Designs

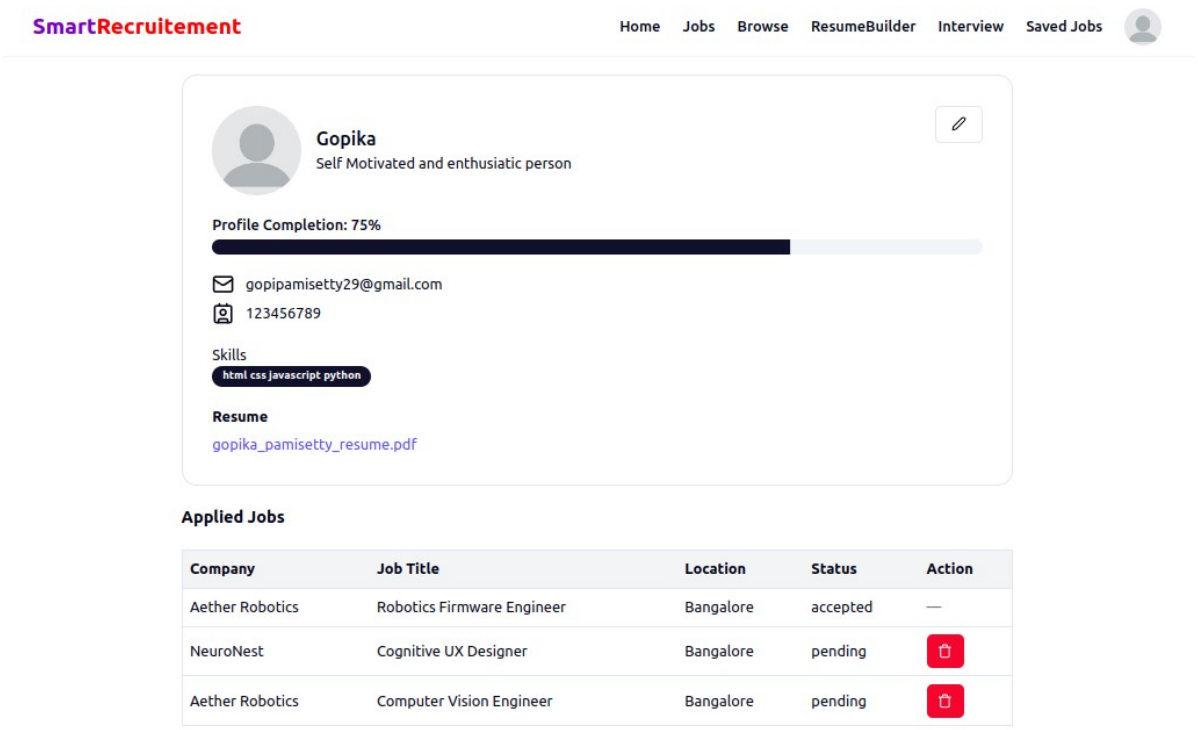
HOME PAGE



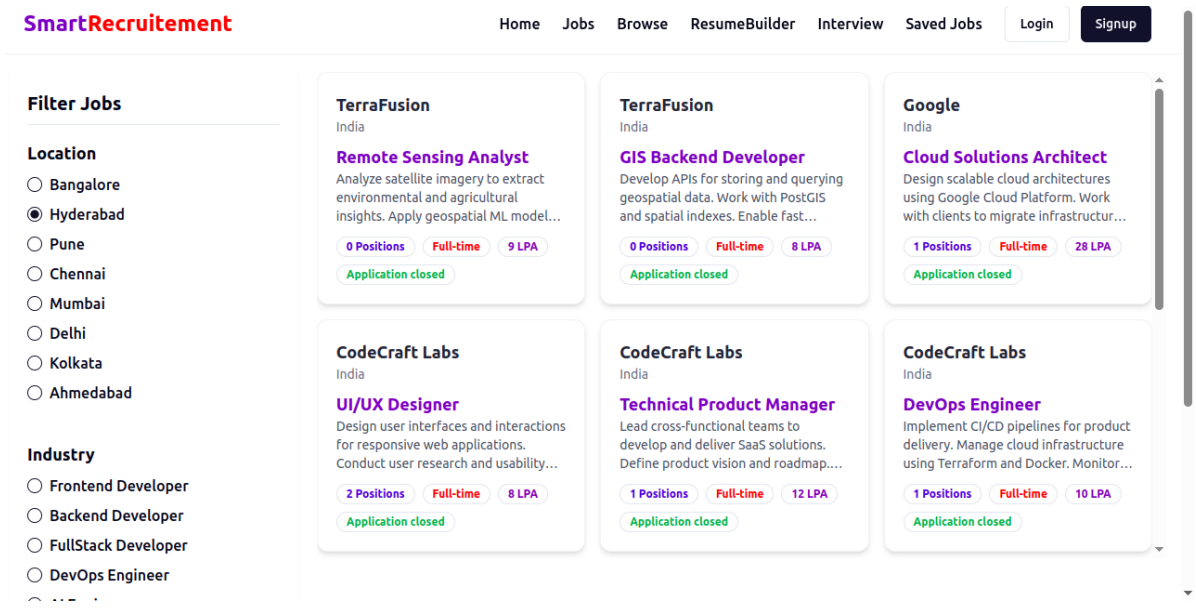
LOGIN PAGE



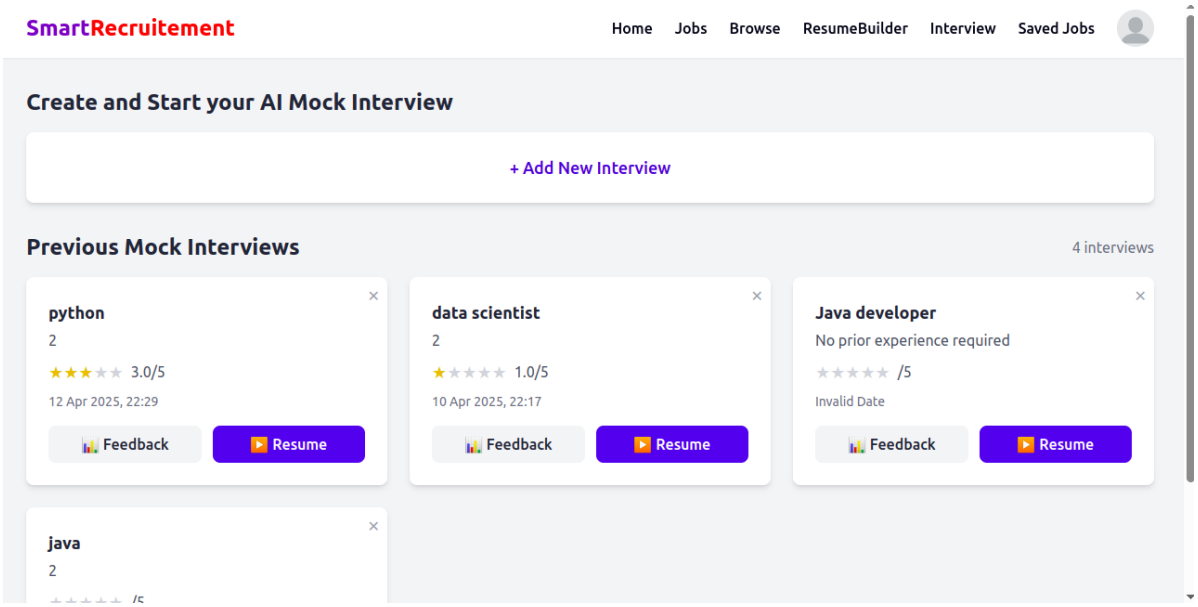
PROFILE PAGE



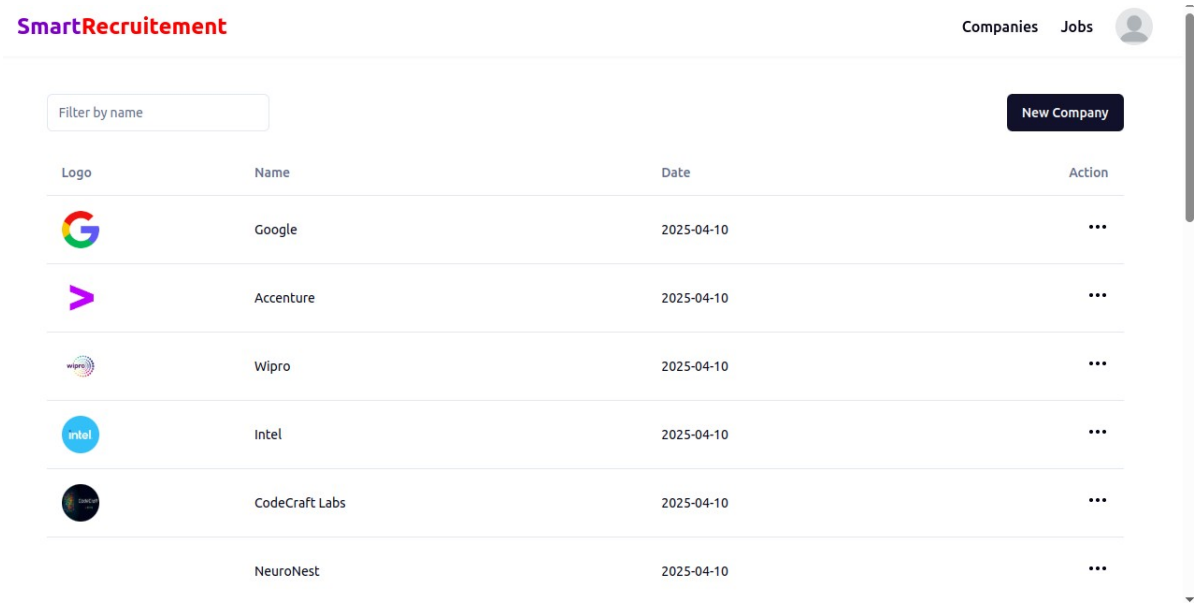
BROWSE JOBS



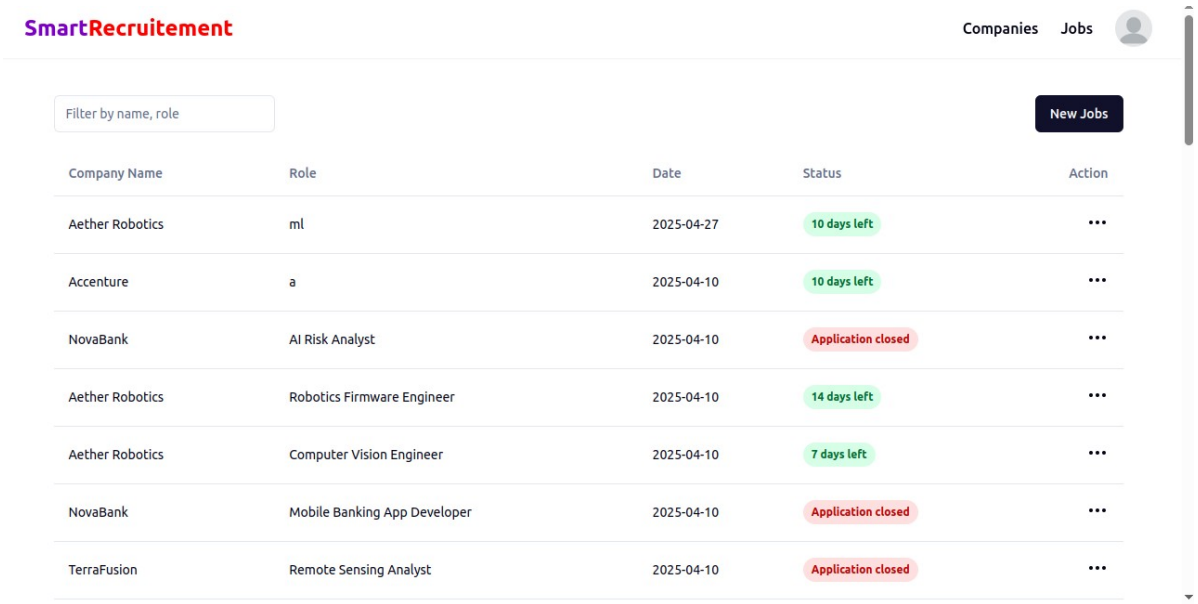
INTERVIEW



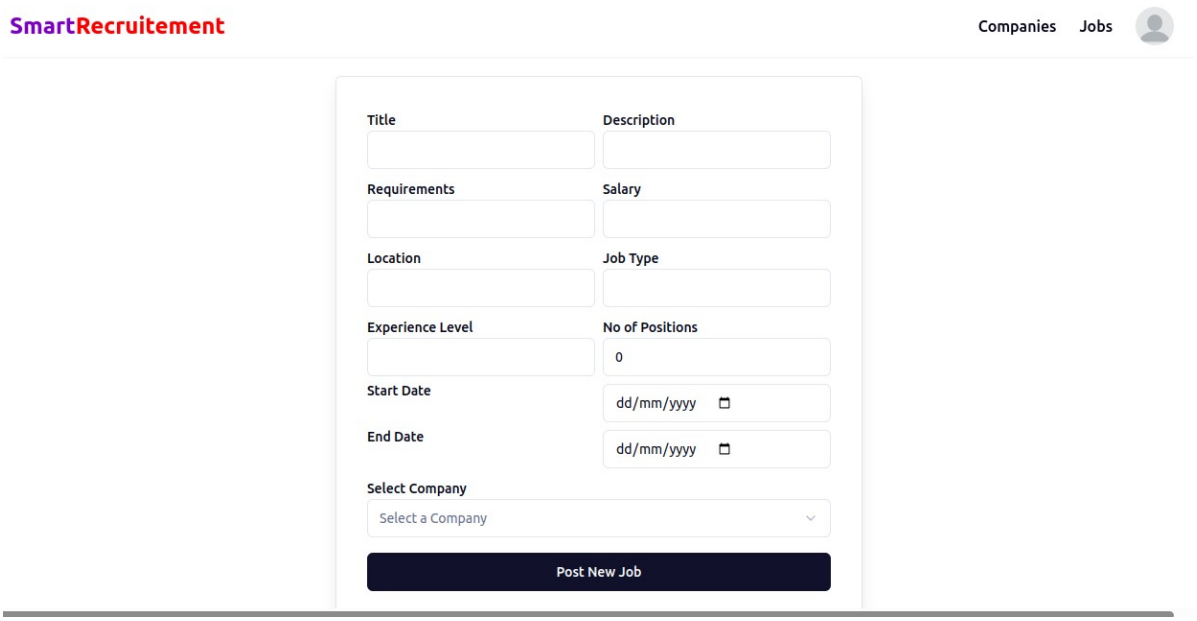
COMPANY DASHBOARD



JOB DASHBOARD



POST NEW JOB



BACKEND : USERS DATA

```
jobportal> db.users.find()
[
  {
    _id: ObjectId('67f78bbdc6d5101e073dbbb6'),
    fullname: 'Gopika Pamisetty',
    email: 'gopikapamisetty@gmail.com',
    phoneNumber: 6300346433,
    password: '$2b$10$gJiRDyXQBhaALixw0suxAuU9UE2QFQprXOUvKcyxPWHC3BwBQlWQC',
    role: 'recruiter',
    profile: { profilePhoto: '', skills: [] },
    createdAt: ISODate('2025-04-10T09:13:33.983Z'),
    updatedAt: ISODate('2025-04-18T16:54:16.435Z'),
    __v: 1,
    savedJobs: []
  },
  {
    _id: ObjectId('67f79417c6d5101e073dbbd4'),
    fullname: 'Gopika',
    email: 'gopipamisetty29@gmail.com',
    phoneNumber: 123456789,
    password: '$2b$10$TLgU10tIhzV7/cF8I8ebVulmhCnud67Ps.nvhdwiDVFoK2FU2QNTu',
    role: 'student',
    profile: {
      profilePhoto: '',
      skills: [ 'html css javascript python' ],
      bio: 'Self Motivated and enthusiatic person ',
      resume: 'https://res.cloudinary.com/dhu1wjrq2/raw/upload/v1745733346/oiryc525xtin9ennf6j6',
      resumeOriginalName: 'gopika_pamisetty_resume.pdf'
    },
    createdAt: ISODate('2025-04-10T09:49:11.056Z'),
    updatedAt: ISODate('2025-05-03T05:49:09.271Z'),
    __v: 15,
    savedJobs: [
      ObjectId('67fa93cd690fd90cc20b8196'),
      ObjectId('67f7a52818e9dead146b146a'),
      ObjectId('67f7b81e27d052fc3033e26c')
    ]
  }
]
jobportal>
```

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

The **Smart Recruitement System** developed as part of this project has effectively achieved its objective of creating a robust, user-centric platform that connects job seekers with recruiters in a streamlined manner. By leveraging the **MERN stack** (MongoDB, Express.js, React.js, and Node.js), the system ensures smooth performance, intuitive navigation, and seamless data management across modules.

Core features such as user authentication, profile creation, resume upload and download, job posting, application management, and role-based access control have been successfully implemented. The platform supports dynamic job filtering, "Save for Later" functionality, and mock interview scheduling, making it practical and interactive for both job seekers and recruiters.

User feedback and testing have demonstrated that the system is responsive, secure, and easy to navigate, with high engagement due to features like resume PDF export and saved job tracking. Integration of JWT-based role management, conditional rendering, and protected API routes ensures that both security and role-specific functionality are maintained throughout the application.

Overall, the Job Portal system provides a comprehensive and scalable solution for employment-related interactions, laying a strong foundation for further expansion and enhancement based on real-world needs.

6.2. Future Enhancements

While the current version of the Job Portal System is stable and functional, several enhancements can be introduced in future versions to improve user experience, performance, and overall platform utility.

6.2.1 Real-Time Chat Between Job Seekers and Recruiters

Adding a real-time messaging feature using WebSockets or Firebase can facilitate direct communication between applicants and recruiters. This will enable quick clarifications, interview scheduling, and engagement without relying on external tools.

6.2.2 Admin Dashboard

Introducing an admin role with a centralized dashboard will allow for better monitoring and moderation. Features can include user management, job approval, analytics dashboards, and soft-deleted data recovery.

6.2.3 Resume Parsing and AI-Based Job Matching

Using NLP techniques or integrating with AI services can allow for automatic parsing of resumes and intelligent job suggestions based on user profiles. This would greatly enhance the relevance of job listings shown to users.

6.2.4 Interview Scheduling and Calendar Integration

Allowing recruiters to send interview invites and sync them with external calendars (like Google Calendar) would streamline the interview process. Applicants could receive reminders and status updates.

6.2.5 Multi-Role Support and Organization Accounts

Expanding role support to include multiple recruiters under one organization account would be useful for companies with large HR teams. Permissions can be assigned to sub-users for managing job postings and reviewing applicants.

6.2.6 Push Notifications and Email Alerts

Adding a system for push or email notifications can keep users updated on new job postings, application status changes, interview calls, or saved job reminders, improving user engagement and retention.

6.2.7 Enhanced Search with NLP and Filters

Improving the job search experience with natural language processing, typo-tolerant search (like Fuse.js), and more refined filters (e.g., by skill, experience, or education) will help users find relevant jobs more quickly.

6.2.8 Mobile Application

Developing a native mobile app (React Native or Flutter) would improve accessibility and convenience. Features like resume uploads from mobile storage, camera access for profile pictures, and biometric login can be introduced.

6.2.9 Real-Time Analytics for Recruiters

Recruiters could benefit from insights like the number of views per job post, application conversion rates, and engagement metrics. This can be shown in the recruiter dashboard with charts and tables.

6.2.10 Accessibility Improvements

Improving accessibility by supporting screen readers, keyboard navigation, and ARIA labels would make the platform more inclusive for differently-abled users

CHAPTER 7

REFERENCES

Below is a list of references and resources that were used throughout the development of the **SMART RECRUITEMENT SYSTEM**

1. **Node.js Documentation**
   ◦ Node.js Foundation (2025). Node.js®: A JavaScript runtime built on Chrome's V8 JavaScript engine. Retrieved from https://nodejs.org
2. **React Documentation**
   ◦ React (2025). React – A JavaScript library for building user interfaces. Retrieved from https://reactjs.org
3. **Express.js Documentation**
   ◦ Express (2025). Fast, unopinionated, minimalist web framework for Node.js. Retrieved from https://expressjs.com
4. **MongoDB Documentation**
   ◦ MongoDB Inc. (2025). MongoDB – The application data platform. Retrieved from https://www.mongodb.com
5. **MongoDB Atlas Documentation**
   ◦ MongoDB, Inc. (2025). MongoDB Atlas – Cloud database service. Retrieved from https://www.mongodb.com/cloud/atlas
6. **Mongoose Documentation**
   ◦ Mongoose (2025). Elegant MongoDB object modeling for Node.js. Retrieved from https://mongoosejs.com
7. **JWT (JSON Web Token) Documentation**
   ◦ Auth0 (2025). JWT.IO – JSON Web Tokens Introduction. Retrieved from https://jwt.io
8. **jsPDF Documentation**
   ◦ jsPDF (2025). Client-side JavaScript PDF generation. Retrieved from https://github.com/parallax/jsPDF
9. **Material UI Documentation**
   ◦ Material-UI (2025). Material UI – React components for faster and easier web development. Retrieved from https://mui.com
10. **Bcrypt.js Documentation**
    ◦ Bcrypt (2025). bcrypt.js – Password hashing library for Node.js. Retrieved from https://github.com/dcodeIO/bcrypt.js

11. **React Router Documentation**
◦ React Router (2025). Declarative routing for React apps. Retrieved from https://reactrouter.com

12. **Render.com Documentation**
◦ Render (2025). Cloud platform for deploying full-stack applications. Retrieved from https://render.com

13. **Postman API Platform**
◦ Postman (2025). Collaboration platform for API development. Retrieved from https://www.postman.com

14. **W3C Web Accessibility Guidelines**
◦ W3C (2025). Web Content Accessibility Guidelines (WCAG) 2.1. Retrieved from https://www.w3.org/WAI/WCAG21/

15. **GitHub - Job Portal System Repository**
◦ GitHub (2025). Project Repository. *(Replace with your repo link if available)*

16. **MDN Web Docs**
◦ Mozilla Developer Network (2025). HTML, CSS, and JavaScript documentation. Retrieved from https://developer.mozilla.org

17. **W3Schools**
◦ W3Schools (2025). Web development tutorials and references. Retrieved from https://www.w3schools.com