JOB PORTAL

**A PROJECT REPORT**

**for**

**Major Project (KCA451) Session (2024-25)**

**Submitted by**

**NEHA SHARMA (2300290140106)**

**POOJA**

**(2300290140117)**

**Submitted in partial fulfilment of the Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of Dr. Shashank Bhardwaj**

**Associate Professor**

****

# Submitted to

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

**(APRIL 2025)**

**CERTIFICATE**

Certified that Neha Sharma (2300290140106) and Pooja (2300290140117) have carried out the project work having “Job Portal” (Project-KCA451) for Master of Computer Application from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student herself and the contents of the project report do not form the basis for the award of any other degree to the employer or to anybody else from this or any other University/Institution. This is to certify that the above statement made by the employer is correct to the best of my knowledge.

Date:

**Dr. Shashank Bhardwaj Dr. Akash Rajak**

**Associate Professor Dean**

**Department of Computer Applications Department of Computer Applications KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**ABSTRACT**

The Job Portal is a digital solution designed to simplify and secure the electoral process by leveraging modern web technologies. Developed using Node.js, ReactJS, TailwindCSS, and MongoDB, this system aims to eliminate the challenges of traditional paper-based voting by providing a user-friendly, transparent, and tamper-resistant platform for managing job recruitments.

This project facilitates online voting for registered users (job seekers) and provides admin (recruiter)istrative control to the job recruitment authorities. The platform supports secure login for both job seekers and admin (recruiter)istrators, with role-based access ensuring appropriate control and data segregation. Job Seekers can cast their job applications electronically after successful authentication, while the admin (recruiter) has full control over managing job seeker registrations, employer profiles, and monitoring job recruitment progress in real time.

The system ensures one person, one job application through strict backend validation, and the database is structured to store and manage all job applications in a secure and organized manner. After the voting period ends, the system automatically tallies the results and displays them with complete transparency, ensuring accuracy and minimizing human error.

With a focus on data integrity, security, and usability, the Job Portal offers a reliable alternative to manual voting procedures, making it suitable for college-level job recruitments, institutional polls, and small-scale organizational voting. By moving the job application process online, this project reduces the chances of manipulation, saves time, and provides an efficient solution for modern-day electoral needs.

**ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Shashank Bhardwaj** for his guidance, help, and encouragement throughout my project work. His enlightening ideas, comments and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and admin (recruiter)istrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other forms of help. Without their support, completion of this work would not have been possible on time. They keep my life filled with enjoyment and happiness.

**Neha Sharma**

**Pooja**

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Certificate | | |  | Ii |
|  | Abstract | |  | Iii |
|  | Acknowledgements | | | Iv |
|  | Table of Contents | | | V |
|  | List of Tables | |  | Viii |
|  | List of Figures | | | Ix |
| 1. | Introduction | |  | 10-12 |
|  | 1.1 | Overview | | 10 |
|  | 1.2 | Features of the Platform | | 10 |
|  |  | 1.2.1 | User Registration and Login | 10 |
|  |  | 1.2.2 | Job Seeker Dashboard | 10 |
|  |  | 1.2.3 | Admin (Recruiter) Dashboard | 11 |
|  |  | 1.2.4 | Voting Process | 11 |
|  |  | 1.2.5 | Results Display | 11 |
|  | 1.3 | Technologies Used | | 12 |
| 2. | Feasibility Study | | | 13-18 |
|  | 2.1 | Technical Feasibility | | 13 |
|  |  | 2.1.1 Technologies Used | | 13 |
|  |  | 2.1.2 Deployment on Setup | | 14 |
|  |  | 2.1.3 Scalability | | 15 |
|  |  | 2.1.4 Security Considerations | | 15 |
|  | 2.2 | Operational Feasibility | | 15 |
|  |  | 2.2.1 User Interaction Flow | | 15 |
|  |  | 2.2.2 Job Recruitment Management | | 16 |
|  |  | 2.2.3 User and Role Management | | 16 |
|  |  | 2.2.4 Content Delivery and Accessibility | | 16 |
|  | 2.3 | Financial Feasibility | | 17 |
|  |  | 2.3.1 Development Cost | | 17 |
|  |  | 2.3.2 Operational Cost | | 17 |
|  |  | 2.3.3 Revenue Generation | | 18 |
|  |  | 2.3.4 Long Term Sustainability | | 18 |
|  | 2.4 | Conclusion | | 18 |
| 3. | Design | |  | 19-26 |
|  | 3.1 Introduction | | | 19 |
|  | 3.2 Front-End Design | | | 19 |
|  |  | 3.2.1 Layout and Navigation | | 19 |

|  |  |  |
| --- | --- | --- |
|  | 3.2.2 Home Page Design | 20 |
|  | 3.2.3 Job Recruitment Page Design | 20 |
|  | 3.2.4 Job Recruitment Detail Page | 20 |
|  | 3.2.5 Account and Profile Page | 20 |
|  | 3.2.6 Admin (Recruiter) Dashboard | 21 |
|  | 3.3 Back-End Design | 21 |
|  | 3.3.1 System Architecture | 21 |
|  | 3.2.2 Database Design | 22-24 |
|  | 3.4 Security Considerations | 24 |
|  | 3.5 Database Schema (ER Diagram) | 25-26 |
| 4 | Testing | 27-30 |
|  | 4.1 Introduction | 27 |
|  | 4.2 Types of Testing | 27 |
|  | 4.2.1 Manual Testing | 27 |
|  | 4.2.2 Unit Testing | 27 |
|  | 4.2.3 Integration Testing | 28 |
|  | 4.2.4 User Acceptance Testing | 28 |
|  | 4.3 Testing Tools | 29 |
|  | 4.4 Bug Reporting and Fixes | 29 |
|  | 4.5 Conclusion | 30 |
| 5 | Project Screenshot | 31-34 |
|  | 5.1 Registration | 31 |
|  | 5.1.1 Registration Page | 32 |
|  | 5.1.2 Admin (Recruiter) Panel (Login Section)  5.1.3 Job Seeker Login Page | 32  33 |
|  | 5.2 Employer List Page | 32 |
|  | 5.3 Add Employer Form | 33 |
|  | 5.4 Job Seekers List Page | 34 |
|  | 5.5 Database Structure in MongoDB Compass (Recruiter) | 34 |
| 6 | Future Scope | 35-38 |
|  | 6.1 Enhanced User Authentication and Authorization | 35 |
|  | 6.2 Admin (Recruiter) Capabilities and Monitoring | 35 |
|  | 6.3 Job Seeker Experience Enhancements | 36 |
|  | 6.4 Security Improvements | 36 |
|  | 6.5 Integration Possibilities | 36 |
|  | 6.6 Mobile Application | 36 |
|  | 6.7 Collaborations and Certifications | 37 |

|  |  |  |
| --- | --- | --- |
|  | 6.8 AI-Powered Assistance and Automation | 37 |
|  | 6.9 Integration with External Systems | 37 |
|  | 6.10 Scalability and Performance | 37-38 |
| 7 | Conclusion | 39-40 |
| 8 | References | 41 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of Table** | **Page** |
| 3.1 | Users Table | 22 |
| 3.2 | Job Seekers Table | 23 |
| 3.3 | Job Applications Table | 24 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name of Figure** | **Page No.** |
| 3.1 | Database Schema (ER Diagram) | 26 |
| 5.1.1 | Dashboard | 31 |
| 5.1.2 | Job Openings | 32 |
| 5.1.3 | Job Positions | 33 |
| 5.1.4 | Login Page | 34 |

# CHAPTER 1 INTRODUCTION

### OVERVIEW

The Job Portal is a web-based application developed using Node.js, MongoDB, ReactJS, and TailwindCSS to provide a secure, transparent, and efficient way of conducting job recruitments. It enables users to register, log in, and cast their job applications digitally. Authentication mechanisms are used to prevent duplicate or unauthorized voting attempts.

This system delivers an end-to-end digital job recruitment process, covering employer listing, secure job application casting, and real-time result generation. It minimizes manual intervention, ensures data integrity, and enhances the overall efficiency of job recruitment management. The system supports two main user roles: Job Seekers and Admin (Recruiter)s.

### Features of the Platform

The platform is built for clarity, accessibility, and tamper-proof voting. Key features include:

### User Registration and Login

* + - * **Registration:**
        + Job Seekers can sign up using their name, email, and password.
        + Email verification via OTP ensures authenticity.
        + Admin (Recruiter) accounts are added directly into the database.

### Login:

* + - * + Users authenticate via email and password.
        + Server-side session handling ensures secure login.
        + Invalid login attempts are handled gracefully.

### Job Seeker Dashboard

### Job Seekers can

* + - * + View ongoing job recruitments and related information.
        + Check employer profiles.
        + Cast one secure job application per job recruitment.

### Once a job application is cast, the system prevents further participation in that job recruitment.

### Admin (Recruiter) Dashboard

* + - * Admin (Recruiter)s can:
        + Create/manage job recruitments (titles, deadlines, etc.).
        + Add, update, or remove employers.
        + Monitor system statistics (total job applications, job seeker status).
        + Access a list of job seekers and their voting status.
        + View and manage job recruitment results in real-time.

### Voting Process

* + - * + One job application per user per job recruitment is enforced.
        + Job Applications are recorded securely in the database.
        + Duplicate voting is blocked through backend validation.
        + Voting is only possible during the active job recruitment period.

### Results Display

* + - * + Results are automatically calculated once voting ends.
        + Admin (Recruiter)s can choose to make results public.
        + The winner is determined based on job application count.
        + Job Applications are immutable post-submission, ensuring full integrity.
        + All access is protected by role-based session controls.
        + Results are displayed in a table format with:
      * Job Application totals
      * Job Recruitment Title
      * Status (e.g., Winner / Participant)
      * Only authorized admin (recruiter)s can access and manipulate result data.

### Technologies Used

* **Frontend:**
  + ReactJS & TailwindCSS: For UI structure and styling.
  + Express.js: For interactivity and form handling.

### Backend:

* + Node.js: Integrated for specific server-side operations such as OTP email handling, admin (recruiter) form processing, or result export functionalities, making it a hybrid full-stack platform.

### Database:

* + MongoDB: Stores user details, job recruitment data, and job application records.

### Authentication:

* + Node.js Sessions: Used to maintain secure user sessions across pages.
  + **Passwords:** Stored using hashing functions for security.

# CHAPTER 2 FEASIBILITY STUDY

The Job Portal has been developed using Node.js, MongoDB, ReactJS, and TailwindCSS, all of which are stable, widely supported, and well-suited for web application development. These technologies are lightweight and require minimal resources, making them ideal for institutional use.

A feasibility study is essential to determine whether the Job Portal project is viable in terms of its technical, operational, and financial aspects. Below is a detailed feasibility analysis of the project, updated to reflect deployment and real-world usage scenarios:

### Technical Feasibility

Technical feasibility assesses the ability of the proposed technologies and tools to support the development and operational requirements of the Job Portal. Here, we evaluate the core components of the project:

### Technologies Used

The Job Portal is built using the MERN Stack along with some additional technologies and tools to enhance functionality, security, and user experience. The chosen tech stack ensures smooth real-time operations, scalability, and secure data management across all modules of the system:

* + - * Node.js (Hypertext Preprocessor): Node.js is the core server-side scripting language used in the development of the voting platform. It is responsible for handling the business logic of the system, including:
* User Authentication: Node.js scripts process login credentials, verify job seeker identities, and initiate secure sessions.
* Job Application Casting: Once authenticated, users can submit their job application, which is validated and securely stored in the database using Node.js.
  + - * MongoDB: MongoDB is used as the relational database management system (RDBMS) for storing and managing all data within the system. It is structured, secure, and optimized for high-performance querying. The following types of data are stored in MongoDB:
* User Records: Details of job seekers and admin (recruiter) users, including login credentials, roles, and status.
* Employer Information: Names, positions, and profile details of job recruitment employers.
* Job Application Records: Each job application cast is stored with job seeker and employer references to ensure one-person-one-job application logic.
  + - * ReactJS and TailwindCSS: ReactJS (Hypertext Markup Language) structures the content of web pages, while TailwindCSS (Cascading Style Sheets) is used for styling and layout. These technologies collectively handle the presentation layer of the system.
* ReactJS elements define forms, navigation links, tables, and interactive components.
* TailwindCSS ensures responsive design, clean layout, and accessibility across desktop and mobile devices.
  + - * Express.js: Express.js is used on the client side to add interactivity and validate inputs before they are submitted to the server. In this project, it serves to:
* Alert users on incorrect or missing input fields.
* Toggle form visibility and control dynamic elements.
  + - * MERN Stack Environment: Used during development as a local server environment. It includes Apache, Node.js, MongoDB, and MongoDB Compass (Recruiter) for integrated testing and deployment simulation.

### Deployment Setup

The system is designed to be deployed on a standard Node.js-enabled web server, either through shared hosting or local deployment via tools like MERN Stack Environment/WAMP.

* + - * Frontend and Backend: Node.js scripts embedded with ReactJS/TailwindCSS are hosted via Apache or similar web servers.
      * Database: MongoDB stores all data and is managed using MongoDB Compass (Recruiter).
      * File Structure: Organized with modular Node.js files for login, voting, result display, and session control, simplifying deployment and maintenance.

### Scalability

The system is scalable for use in institutional or organizational voting with features like:

* + - * Modular codebase in Node.js, separating voting, authentication, and admin (recruiter) modules.
      * Normalized database schema to reduce redundancy and allow efficient scaling.
      * Support for a growing number of users, job recruitments, and employers without performance degradation.

This design allows easy upgrades or future feature additions without architectural overhaul.

### Security Considerations

Security is a core aspect of this system, with the following protections in place:

* + - * Session Management: Node.js sessions secure user identity throughout interactions.
      * **Input Validation**: All form inputs are sanitized to avoid SQL injection or scripting attacks.
      * Role-Based Access Control (RBAC): Only admin (recruiter)s can access sensitive sections such as job application counts and job recruitment management.
      * **Password Hashing**: User passwords are encrypted using hashing functions before storage.
      * **Secure Data Transactions**: If deployed on HTTPS-enabled servers, all user actions and data transfers are encrypted end-to-end.

### Operational Feasibility

Operational feasibility assesses how practical and user-friendly the system is for real-world use by non-technical users. The system has been designed with simplicity and clarity to ensure successful adoption.

### User Interaction Flow

There are two main user types: Job Seekers and Admin (Recruiter)s.

* + - * Job Seekers: Log in, view job recruitments, select employers, and cast one secure job application per job recruitment.
      * Admin (Recruiter)s: Manage users, create/edit job recruitments, manage employers, and view results.

### Job Recruitment Management

Admin (Recruiter)s can handle all aspects of the job recruitment process from the admin (recruiter) panel:

* + - * Add/edit/delete employer entries.
      * Control job recruitment timelines (start and end dates).
      * View overall job application status in real-time.

Future updates can introduce scheduling and advanced filtering features.

### User and Role Management

The system distinguishes between two user roles: Admin (Recruiter)s and Job Seekers. Admin (Recruiter)s have full control over system operations, including managing users, configuring job recruitments, and viewing results. Job Seekers, on the other hand, can only participate in job recruitments by casting their job applications. Admin (Recruiter)s can create, edit, or remove user accounts and ensure each user can only job application once per job recruitment, maintaining the integrity and organization of the job application process.

### Content Delivery and Accessibility

The platform prioritizes accessibility and ease of use across all user devices:

* + - * The system supports responsive layouts for both desktop and mobile users.
      * A minimal UI helps even non-technical users perform tasks without confusion.
      * Text, buttons, and navigation elements are clearly labeled.
      * This ensures accessibility and ease of use during live job recruitments.
      * The interface adjusts automatically to different screen sizes and devices, including desktops, laptops, tablets, and smartphones.
      * The layout avoids unnecessary graphics or clutter, allowing users to focus on core actions like logging in, voting, or managing data.
      * Menus and buttons are prominently placed and consistently styled across all pages for intuitive navigation.
      * All interface text, labels, and instructions are written in simple, action-oriented language to avoid confusion.
      * User input fields provide clear error messages if data is incomplete or invalid, helping prevent mistakes during registration or voting.
      * Color contrasts between text and background meet accessibility standards, ensuring readability for users with visual impairments.
      * All clickable elements (buttons, forms) can be accessed using keyboard shortcuts, improving usability for differently-abled users.
      * Lightweight design and minimal external dependencies ensure fast page loads, even on slower internet connections.

### Financial Feasibility

Financial feasibility evaluates the projected costs for developing, hosting, and maintaining the Job Portal. The use of open-source technologies ensures the system is affordable and sustainable.

### Development Costs

* + - * Personnel: The project can be built and maintained by developers with knowledge of Node.js, MongoDB, and basic front-end technologies (ReactJS/TailwindCSS/Express.js). Since these skills are widely available, personnel costs can be kept low, especially for institutions using internal teams or students.
      * Development Tools: All tools used in the system—such as the code editor, local server environments (MERN Stack Environment/WAMP), and database tools—are open-source or free, minimizing initial investment and licensing costs.

### Operational Costs

* + - * Hosting: The system can be hosted on any standard shared hosting provider that supports Node.js and MongoDB, making it extremely cost-effective. No specialized cloud infrastructure is required.
      * Database Management: MongoDB, commonly available on most hosting platforms, are sufficient for handling user accounts, job applications, and results, reducing the need for expensive cloud database solutions.

### Third-Party Services: No third-party or premium tools are necessary for regular use.

### Revenue Generation

* + - * Institutional Licensing Model: If adopted on a larger scale, the system could be monetized through licensing to educational institutions, organizations, or government bodies interested in conducting secure internal job recruitments.
      * Customization Services: Revenue could also be generated by offering customized versions of the job portal with additional features, branding, or multilingual support for clients with specific needs.

### Long-Term Sustainability

The long-term sustainability of the job portal relies on its ability to remain lightweight, secure, and adaptable. Since it is built using open-source Node.js and MongoDB, operational costs are minimal, and the system can be hosted on low-cost servers. Its simplicity makes it maintainable even with limited technical staff, while future modular enhancements can expand its functionality. Sustainability will also depend on keeping the platform secure, user-friendly, and aligned with the evolving needs of organizations and institutions.

### 2.4 Conclusion

The Job Portal is a technically feasible, operationally efficient, and financially sustainable solution for secure digital job recruitments. It uses widely adopted tools that ensure stability and ease of deployment across varied environments. Its scalability, security, and low cost make it suitable for educational institutions, clubs, and organizations seeking to conduct reliable, tamper-free job application processes.

The feasibility study concludes that the Job Portal is technically sound, operationally efficient, and financially sustainable. Built with widely available technologies such as Node.js and MongoDB, the system offers a stable and cost-effective solution for conducting secure job recruitments.

The platform ensures smooth interactions between job seekers and admin (recruiter)s, with clear user roles and secure job application processes. Its low infrastructure requirements make it easy to deploy across different environments, from educational institutes to corporate settings.

With potential for future revenue through licensing or customization, and the flexibility to scale, the system demonstrates strong potential as a reliable digital voting solution that balances accessibility, security, and affordability.

# CHAPTER 3 DESIGN

This section covers the overall design considerations, user interface (UI) design, system architecture, database schema, and API structure of the Job Portal.

### 3.1 Introduction

The design of the Job Portal aims to create a secure, intuitive, and user-friendly environment for both job seekers and admin (recruiter)istrators. The application is developed using Node.js, ReactJS, TailwindCSS, Express.js, and MongoDB, ensuring compatibility with most hosting platforms and supporting efficient data processing. The system consists of both frontend and backend components, designed to work in harmony for smooth user interaction, secure authentication, and accurate job application management. The modular structure allows for easy maintenance, testing, and future scalability.

### Front-End Design (User Interface)

The system uses a consistent layout across all pages with clearly labeled buttons and navigation options. The top bar or sidebar contains links to key pages including Home, Register, Login, Job Application, and Logout. Once a user logs in, the navigation updates based on their role (admin (recruiter) or job seeker). For example, job seekers can see job recruitments and job application, while admin (recruiter)s access management tools and voting statistics.

### Layout and Navigation

The system uses a consistent layout across all pages with clearly labeled buttons and navigation options. The top bar or sidebar contains links to key pages including Home, Register, Login, Job Application, and Logout. Once a user logs in, the navigation updates based on their role (admin (recruiter) or job seeker). For example, job seekers can see job recruitments and job application, while admin (recruiter)s access management tools and voting statistics.

* + - * Home: Directs users to the homepage with basic information and an overview of the job portal.
      * **Register/Login**: Allows users to register or log in securely using their credentials.
      * Job Application: Enables authenticated users to view the list of available job recruitments and cast their job applications.

### Home Page Design

The home page serves as an introductory screen for the platform. It includes a welcome message, a brief explanation of the system, and buttons that guide users to the login or registration page. A summary of how the job application process works is presented in a step-wise format. At the bottom, a footer displays links to terms, privacy policy, and contact information.

### Job Recruitment Page Design

This page displays a list of all ongoing and upcoming job recruitments. Each job recruitment card or table row contains the job recruitment title, a brief description, and its current status (active or closed). Only logged-in job seekers are allowed to participate. Clicking on an active job recruitment opens the Job Recruitment Detail page. The interface is minimal and responsive, allowing job seekers to easily navigate on both mobile and desktop devices.

### Job Recruitment Detail Page

Once a user selects an job recruitment, they are redirected to a detailed page that includes:

* + - * Job Recruitment Overview: Includes job recruitment description, start and end time, and instructions for voting.
      * Employer Profiles: Each employer has a brief profile outlining their name, agenda, and optional photo.
      * Job Application Submission: Secure voting interface where the user selects a employer and submits their job application.

### Account and Profile Page

The account functionality is handled through session management and the index.php page. Once a user logs in, their session is created and used across different pages to track their identity.

* + - * **Profile Information**: The session contains user data such as username and user ID, which can be used to personalize pages like the voting screen or sidebar.
      * **Dashboard Button**: The index.php acts as the landing page for authenticated users. It likely serves as a dashboard, offering options such as voting or viewing results.
      * **Logout Button**: The logout functionality is implemented in sess.php, where session variables are unset and the user is redirected to the login screen.

For users with admin (recruiter) roles (identified via session or user-level check), additional navigation and control panels are available.

### Admin (Recruiter) Dashboard

The Admin (Recruiter) Dashboard is designed for users with elevated permissions. The side\_bar.php and job application\_result.php pages suggest admin (recruiter) functionalities.

* + - * User Management: Admin (Recruiter)s can manage employers, job seekers, and view job application results through dedicated panels.
      * Sidebar Navigation: side\_bar.php provides navigation links. These are dynamically customized for admin (recruiter)s to include links for managing users, results, or voting sessions.
      * Overview Section: job application\_result.php provides an overview of voting statistics, which functions as the admin (recruiter)'s Results Dashboard, showing job application counts and summaries.

### Back-End Design (Server, API, and Database)

* + 1. **System Architecture**

The Job Portal follows a client-server architecture, where the client-side interface is built using ReactJS, TailwindCSS, and Express.js within Node.js files to provide a responsive and interactive experience for users. The server-side is powered by Node.js, which handles core functionalities such as user authentication, job application submission, session management, and result generation. MongoDB serves as the backend database, storing structured data related to users, job seekers, employers, and job applications. This architecture ensures a clear separation between presentation and logic layers, enabling secure communication between the user interface and the database. It also provides scalability, maintainability, and support for role-based access control across different modules of the system.

The job portal follows a client-server architecture, where:

* + - * The client side is built using ReactJS, TailwindCSS, and Express.js within Node.js pages, providing the front-end for users to interact with (e.g., voting interface, login forms).
      * The server side is implemented using Node.js, handling core application logic such as user authentication (login\_query.php), job application submission (submit\_job application.php), session management (sess.php), and displaying results (job application\_result.php).
      * MongoDB is used as the database, storing user data, employer information, and voting records. This setup ensures data integrity, consistency, and supports structured querying.

### Database Design

The database schema is designed to handle the core functionalities of the Job Portal. The following tables are used:

### Users Collection

This collection stores information about the users, including both students and admin (recruiter)s.

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| Id | INT | Unique identifier for each user |
| name | VARCHAR | Full name of the user |
| email | VARCHAR | Email address (used for login) |
| password | VARCHAR | Hashed password for security |
| Role | ENUM | Defines whether the user is an admin (recruiter) or a job seeker |
| created\_at | DATETIME | Timestamp when the account was created |
| updated\_at | DATETIME | Timestamp of the last profile update   |  | | --- | |  |  |  | | --- | |  | |

Table 3.1: Users Table

### Job Seekers Collection

This collection stores the job seeker information for the Job Portal.

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| id | INT   |  | | --- | |  |  |  | | --- | |  | | Unique ID for each job seeker |
| name | VARCHAR | Full name of the job seeker |
| email | VARCHAR | Used for login |
| password | VARCHAR | Hashed password |
| aadhaar | VARCHAR | Aadhaar number for identity |
| job seekerId | VARCHAR | Unique job seeker ID (used during the job application process) |
| role | VARCHAR | Role of the user |
| hasJob Applicationd | BOOLEAN | Flag to check if the user has already cast their job application |
| createdAt | DATETIME | Timestamp when the job seeker account was created |

Table 3.2 Job Seekers Table

### Job Applications Collection

This collection stores the individual job applications cast by users during an job recruitment.

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| \_id | INT | Unique job application record ID |
| job seeker\_id | INT | Reference the Job Seeker’s ID |
| employer\_id | INT | References the selected employers   |  | | --- | |  | |
| job recruitment\_id | INT | References the job recruitment where the job application was cast |
| casted\_at | DATETIME | Date and time when the job application was recorded |
| status | ENUM | Status of the job application (valid/invalid) |

Table 3.3: Job Applications Table

### 3.4 Security Considerations

The system includes several built-in security features to ensure data confidentiality, user privacy, and integrity of the job recruitment process.

* **Session Authentication**: Only logged-in users can access protected pages.
* Password Encryption: Passwords are stored securely using Node.js’s password\_hash() function.
* One Person, One Job Application: The backend restricts users from voting more than once per job recruitment.
* **Input Validation**: Form inputs are validated to prevent SQL injection and improper data entries.
* **Secure Queries**: All SQL operations use prepared statements.
* Role Separation: Admin (Recruiter) and job seeker access is enforced using session-based roles.

### 3.5 Database Schema (ER Diagram):

### Entities and Attributes:

### User

### Represents admin (recruiter)-level users who manage job recruitments.

### Fields: id (PK), name, email, password, role, created\_at

### Job Seekers

### Registered individuals who are eligible to job application.

### Fields: id (PK), name, email, password, has\_job applicationd, job seeker\_id

### Employers

### Individuals participating in job recruitments.

### Fields: id (PK), name, position, image, job recruitment\_id

### Job Applications

### A record of each job application cast in the system.

### Fields: id (PK), job seeker\_id (FK), employer\_id (FK), job recruitment\_id (FK), casted\_at

### Relationships:

### Each job application links to a job seeker and a employer through foreign keys.

### Each job seeker can job application only once in an job recruitment, tracked using has\_job applicationd..

### Employers are associated with job recruitments via the job recruitment ID.

### 

### 

### Figure 3.1 Database Schema (ER Diagram)

# CHAPTER 4

# TESTING

Testing is a critical phase in the development lifecycle of any system. It ensures that the software operates according to the specified requirements and is free from major defects. For the Job Portal, various testing methods were applied to verify the accuracy.

### Introduction to Testing

Testing is a vital phase in software development to ensure that the system performs as expected and meets the specified requirements. For the Job Portal, testing focused on validating the core functionalities including user registration, login, job application casting, employer management, and result calculation. The system was tested to ensure secure operation, correctness of logic, and smooth user interactions. Both manual testing and basic functional validations were used throughout development to guarantee stability and reliability.

### Types of Testing

Different types of testing were performed throughout development to ensure that the system was reliable, secure, and user-friendly.

### Manual Testing

Manual testing was carried out by interacting with the user interface directly through a browser. Testers performed key actions such as:

* + - * Registering as a new job seeker and logging in using the provided credentials.
      * Casting job applications in active job recruitments to verify the one-job application-per-user functionality.
      * Navigating across pages to test session handling and access control.
      * Observing UI behavior across different devices and screen sizes.
      * Checking admin (recruiter) functionalities including employer and job seeker management.

### Unit Testing

Unit testing was done manually for individual Node.js components and functions:

* + - * Validating that user registration scripts correctly hash and store passwords.
      * Ensuring login logic verifies credentials accurately using session variables.
      * Checking that job application submissions only occur once per job seeker per job recruitment.
      * Confirming the correctness of admin (recruiter)-side job application tallying and result display.

### Integration Testing

Integration testing ensured smooth interaction between the database, server scripts, and front-end components:

* + - * Testing the connection between form submissions and backend processing (e.g., registration, voting, result display).
      * Verifying data flow from the UI (forms) to MongoDB database using Node.js queries.
      * Ensuring the sidebar and admin (recruiter) dashboard loaded relevant dynamic data from the database.
      * Checking that role-based access allowed users and admin (recruiter)s to access only their respective pages.

### User Acceptance Testing (UAT)

User acceptance testing involved letting actual users (students or faculty) try the system and provide feedback:

* + - * Gathering user impressions of the clarity and ease of use of the UI.
      * Ensuring that error messages and button labels were understandable.
      * Ensuring that error messages and button labels were understandable.
      * Testing the overall flow from registration to job application casting.

This phase was essential for fine-tuning the system before deployment and building user trust in a secure, intuitive voting experience.

### Testing Tools

The following tools and techniques were used during the testing phase:

### MERN Stack Environment (Apache + MongoDB + Node.jsMyAdmin (Recruiter)):

* + Used to run the local server and database for development and testing.
  + MongoDB Compass (Recruiter) was used to inspect and verify changes in the database.

### Browser Developer Tools (Chrome DevTools):

* + Used for debugging front-end ReactJS, TailwindCSS, and Express.js.
  + Console and Network tabs helped trace requests and errors during form submission

### Manual Browsing & Inspection:

* + Each Node.js module (registration, login, voting, admin (recruiter) dashboard) was tested manually for expected behavior.

### Bug Reporting and Fixes

During the testing phase of the Job Portal, several issues were detected and successfully resolved to ensure a stable and secure platform. Key bugs and their fixes included:

1. **Session Timeout Issues:** Users were occasionally logged out too quickly. Session logic was adjusted in sess.php to improve timing.
2. Job Application Submission Errors: In some cases, job applications were not registering correctly due to missing job recruitment ID. The job application handling logic was updated for validation.
3. Admin (Recruiter) Page Loading Delay: When large numbers of job seekers or employers were present, the admin (recruiter) panel took longer to load. Pagination and query optimization were introduced.
4. Input Validation Bugs: Some fields accepted blank or invalid input. Client-side Express.js and server-side Node.js validation were both improved.

### Conclusion

The Job Portal underwent thorough testing to validate its functionality, usability, and data accuracy. Manual and unit testing ensured core modules like registration, login, voting, and result calculation worked correctly. Integration testing confirmed smooth data flow across components. The platform has proven to be stable and ready for real-world deployment in institutional settings. Future improvements can include automated test cases and performance benchmarking tools for large-scale usage.

# CHAPTER 5 PROJECT SCREENSHOT

### Dashboard:

### The dashboard of the Job Portal serves as the central hub for job seekers after successful login. It provides easy access to all major functionalities of the platform, such as browsing job listings, managing profiles, and tracking job applications. The interface displays personalized information including the user's name, application history, and recommended jobs based on skills and preferences.

### Key features of the dashboard include:

### A navigation sidebar with links to Home, Job Listings, Applications, Profile, and Logout.

### A real-time status panel showing active applications and interview schedules.

### Notifications for new job openings and recruiter messages.

### A responsive design ensuring smooth usability across desktop and mobile devices.

### All data displayed on the dashboard is dynamically retrieved from the database, ensuring that users always see the most recent updates relevant to their job search.

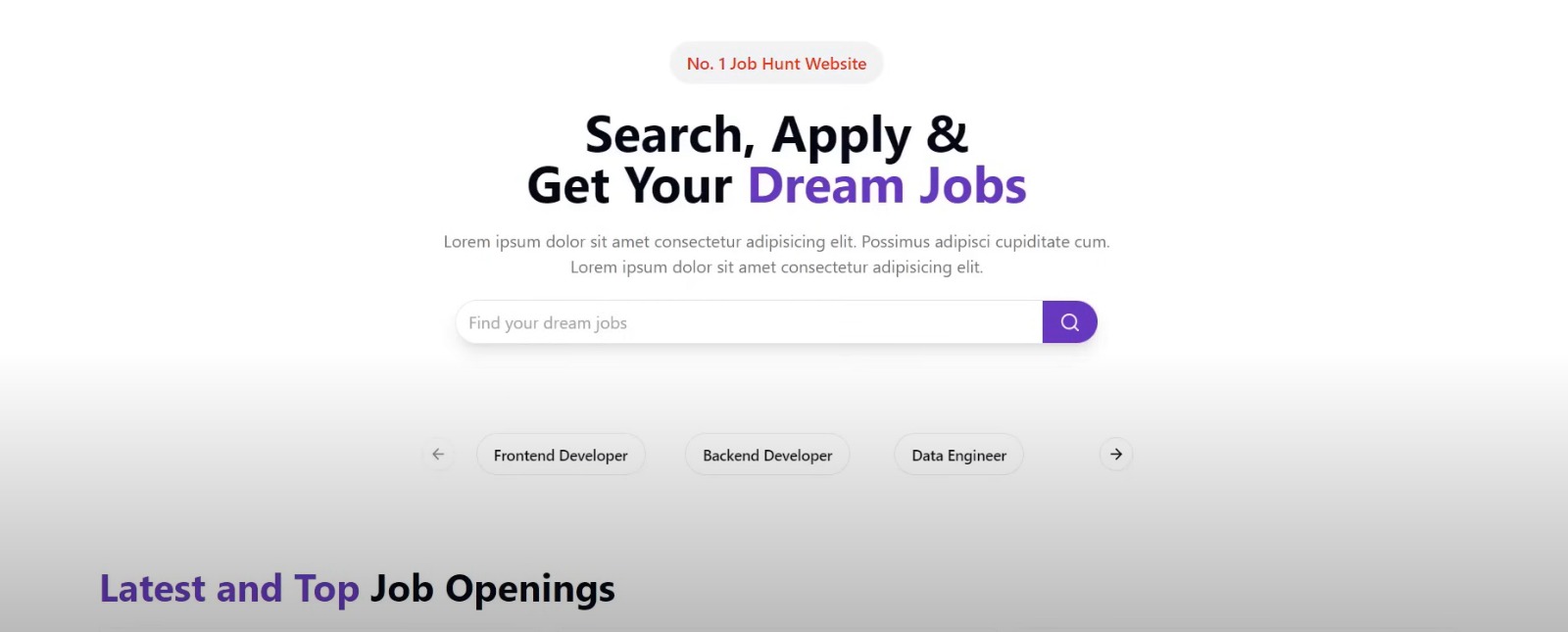


Figure 5.1.1: Dashboard

### Job Openings:

The **Job Openings** section of the Job Portal is the core feature where job seekers can explore available employment opportunities. It provides a clean and intuitive layout showcasing all active job listings posted by registered employers. Each job card or row includes essential details such as job title, company name, location, salary range, and application deadline. Users can click on individual listings to view the full job description, qualifications required, and the option to apply directly from the interface. The design ensures quick navigation, real-time updates, and filtering options to help job seekers find relevant opportunities with ease. This section plays a vital role in bridging the gap between potential candidates and employers by making job discovery straightforward and efficient.

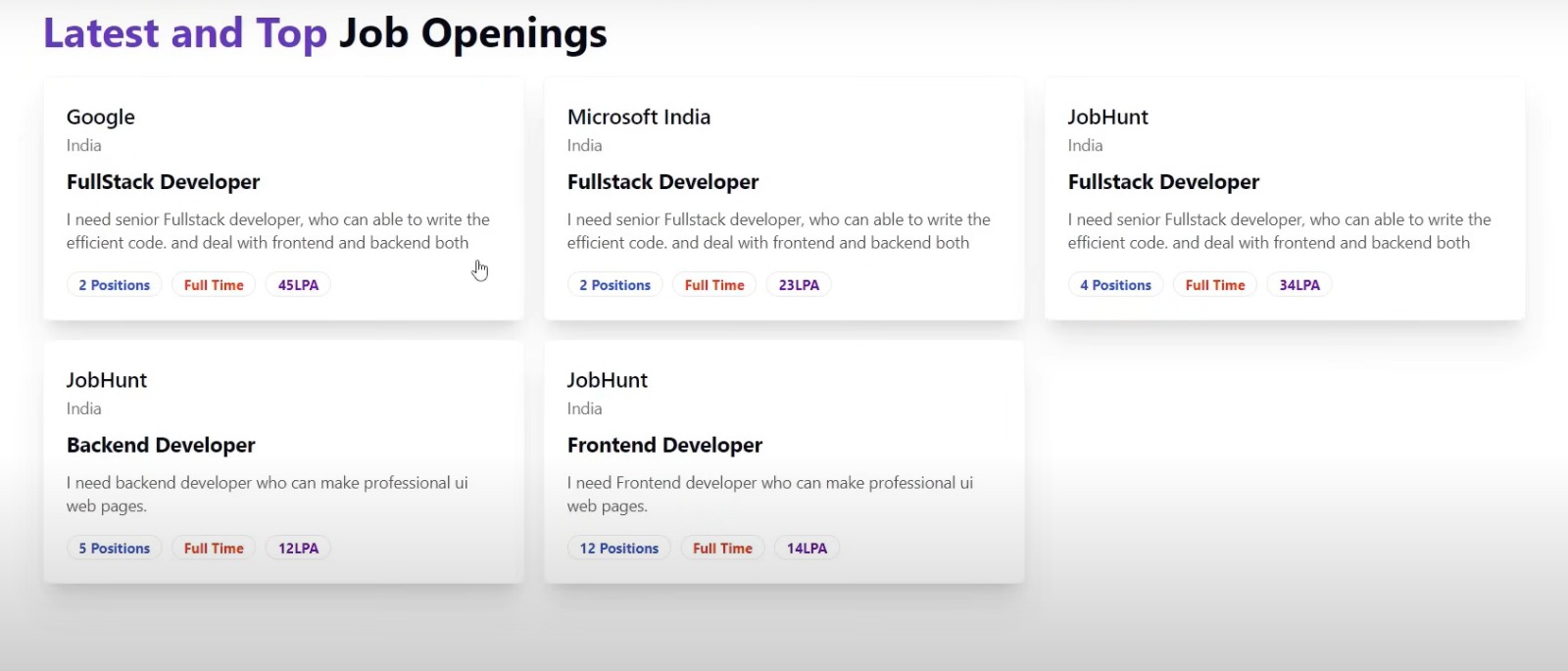


Figure 5.1.2: Job Openings

### Job Position:

The job position involves managing the authentication process for job seekers to ensure secure access to the job application system. Candidates must provide their ID Number and Password to verify their identity before proceeding.To maintain fairness and integrity in the recruitment process, the system limits each job seeker to a single login session and one active job application at a time. This policy is clearly communicated to users to prevent duplicate or fraudulent applications.The role requires designing and maintaining a simple, clear, and user-friendly interface that ensures a smooth and efficient experience for all job seekers.

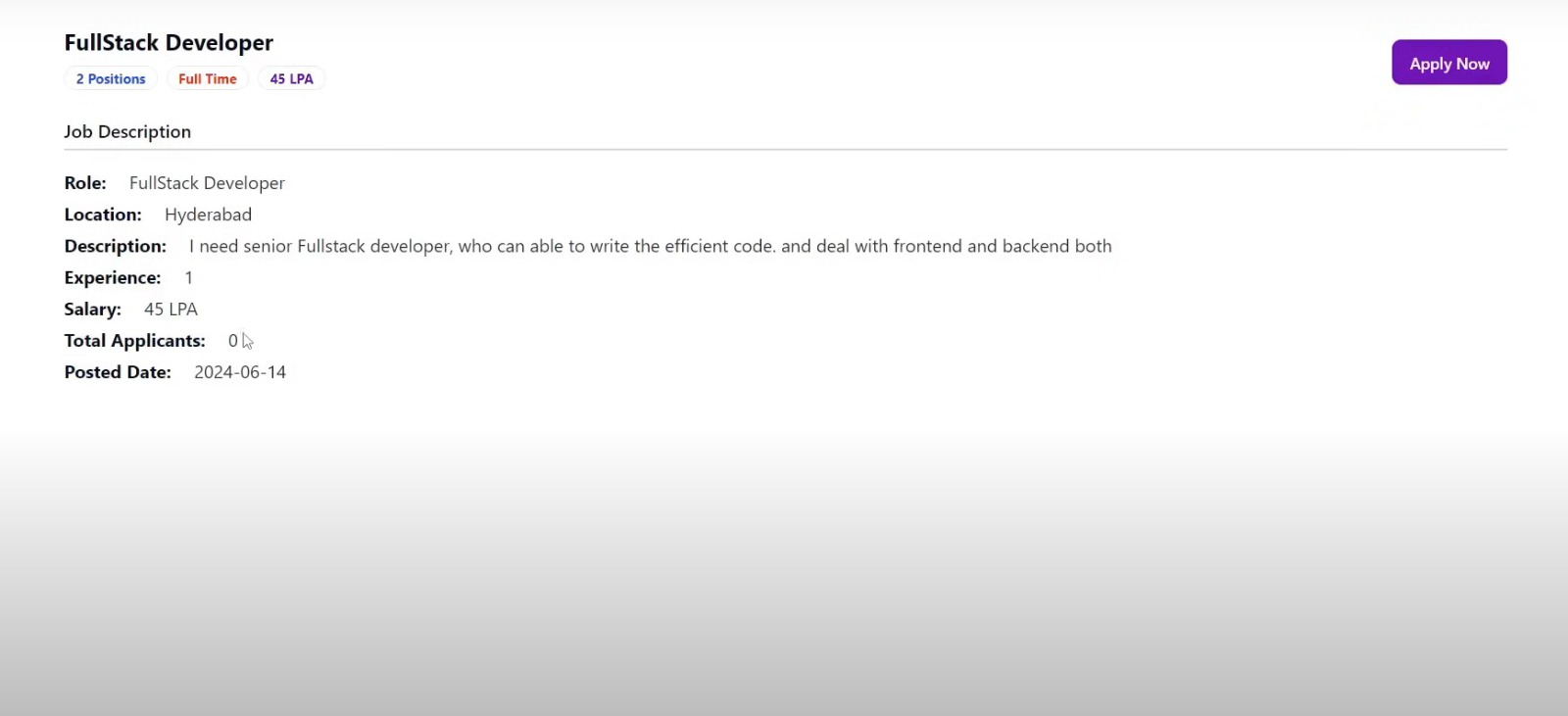


Figure 5.1.3: Job Position

### 5.1.4 Login Page:

The login page serves as the gateway for users to access the job recruitment system. It allows Admin (Recruiter) users to securely sign in with their credentials to manage employer entries and other recruitment tasks.Users must enter their Username and Password to proceed. The interface is designed to be straightforward and user-friendly, ensuring a smooth login experience.Security measures are in place to protect user data and prevent unauthorized access. Additionally, helpful prompts and error messages guide users during the login process, making it easy to recover or reset forgotten credentials.

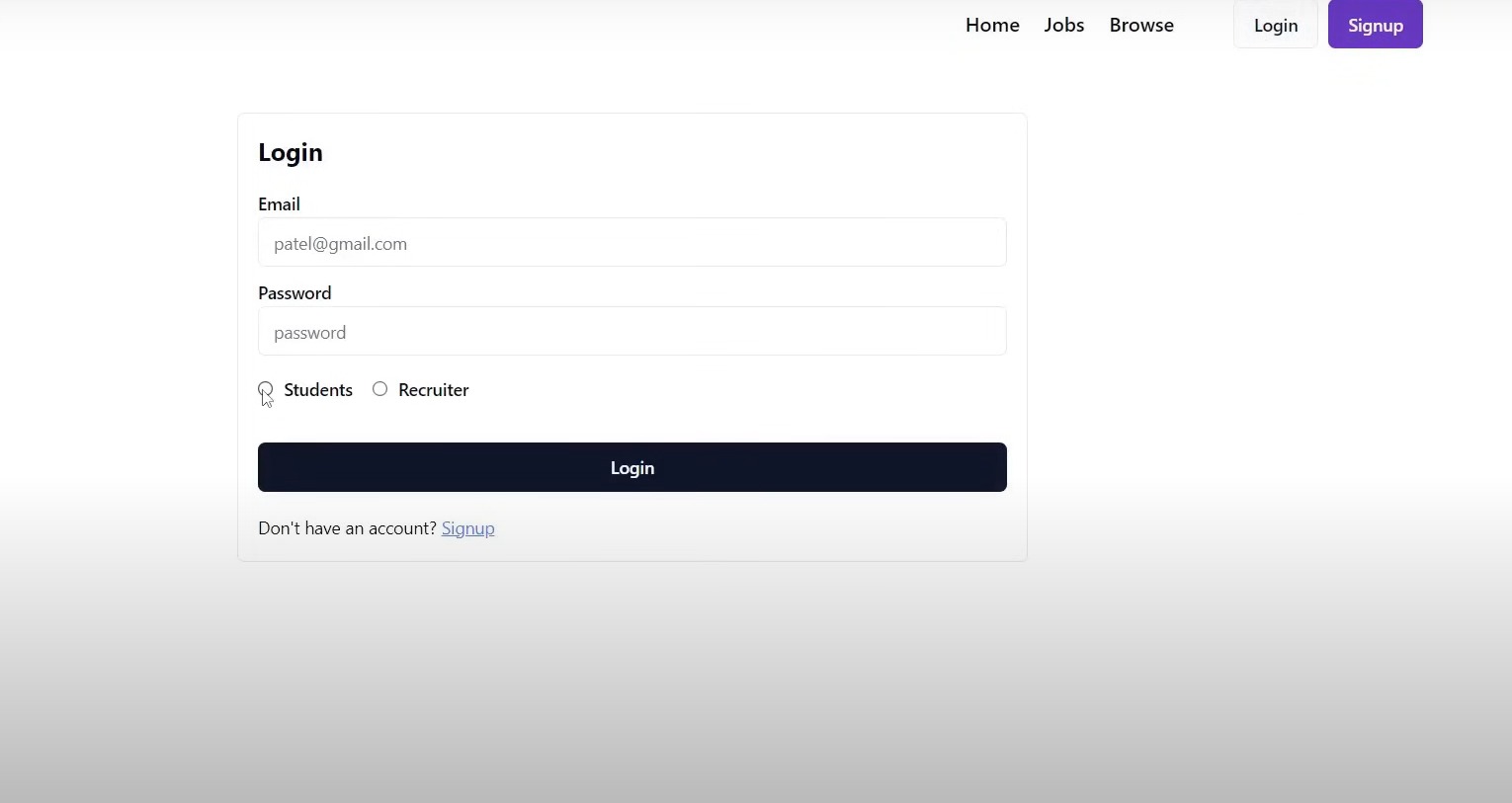
****

Figure 5.1.4: Login Page

# CHAPTER 6 FUTURE SCOPE

The Job Portal, as a modern digital platform, holds strong potential for future upgrades that can enhance both the security and efficiency of the job application process. Below are several areas where the platform can be improved and expanded:

### Enhanced User Authentication and Authorization

* + - Biometric Integration: Future versions could incorporate fingerprint or facial recognition to verify job seeker identity for added security.
    - Multi-Factor Authentication (MFA): Add an extra layer of security using OTP or email verification during login and job application casting.
    - Blockchain-based Job Application Ledger: Implementing blockchain can provide tamper-proof records of job applications, enhancing transparency and trust.
    - **Session Timeout and Auto Logout**: Auto-logout inactive users to prevent unauthorized access.

### Admin (Recruiter) Capabilities and Monitoring

* + - Real-time Voting Analytics: Allow admin (recruiter)s to monitor live job application counts, turnout rates, and demographic statistics with visual dashboards.
    - Custom Job Recruitment Configuration: Enable dynamic job recruitment setup options, allowing admin (recruiter)istrators to define specific rules such as job application limits per user, set restrictions on the number of employers, and apply eligibility filters based on criteria like user roles, departments, or verified status.
    - Logs & Audit Trails: Maintain secure logs of all admin (recruiter) and user actions to ensure accountability and traceability.

### Job Seeker Experience Enhancements

* + - Job Seeker Notifications: Notify users about voting start/end times, registration deadlines, and results via email/SMS.
    - **Multilingual Interface**: Provide language options to improve accessibility for users from different regions.
    - **Mobile-Friendly Design**: Optimize the UI for seamless usage on smartphones and tablets.

### Security Improvements

* + - End-to-End Encryption: Ensure data transmission is fully secure during all stages, from login to job application submission.
    - **CAPTCHA & Bot Prevention**: Add CAPTCHA challenges during login and registration to avoid automated attacks.

### Integration Possibilities

* + - Government ID Verification: Integrate APIs to verify job seeker identities through national ID databases or student records.
    - Third-Party Hosting & Cloud Scalability: Host the system on scalable platforms (e.g., AWS, Azure) to handle larger job recruitment events.

### Mobile Application

* + - Multi-Platform Support: Develop a dedicated mobile application for both Android and iOS platforms to allow users to participate in job recruitments, view employer profiles, and receive real-time notifications about voting deadlines and results.
    - **Secure Mobile Voting**: Integrate secure mobile authentication methods such as biometric verification (fingerprint or face ID) to ensure safe and user-friendly voting experiences on mobile devices.
    - Push Notifications: Implement push notifications to alert users about upcoming job recruitments, registration deadlines, and important updates regarding the job application process, enhancing engagement and participation.

### Collaborations and Certifications

* + - **Collaborate with Government Bodies**: Partner with electoral commissions, government agencies, and civic organizations to ensure that the system complies with the official regulations and can be adopted for real-word voting scenarios.
    - **Corporate Training Programs**: Expand to offer specialized courses for corporate training, allowing businesses to purchase group licenses for their employees.

### AI-Powered Assistance and Automation

* + - AI Chat Support: Incorporate AI-powered chatbots to guide users through the job application process, provide instant responses to frequently asked questions, and assist with troubleshooting during job recruitments.
    - Intelligent Anomaly Detection: Use AI algorithms to monitor and flag suspicious voting activity in real-time, helping to detect duplicate job applications, bot participation, or irregular patterns that may suggest job seeker fraud.

### Integration with External Systems

* + - Government/Institution Integration: Enable integration with external governmental or institutional databases (such as Aadhaar or student portals) for streamlined job seeker verification and improved authenticity.
    - **Third-Party API Support:** Allow integration with notification services (e.g., Twilio, SendGrid) and analytics platforms (e.g., Google Analytics) to enhance communication and performance tracking within the voting platform.

### Scalability and Performance

* + - Server Optimization: As the number of users and job recruitments increases, the backend is designed to efficiently handle concurrent requests, ensuring that voting, authentication, and result calculations perform smoothly even under high load.
    - Database Management: The system utilizes a well-structured relational database (MongoDB/Node.jsMyAdmin (Recruiter)), optimized through indexing and query optimization techniques to support fast data retrieval and storage, essential for real-time result calculation.
    - **Code Modularity:** The platform's modular architecture allows for seamless upgrades and new feature integrations without disrupting existing functionality.
    - Load Management: Efficient session handling, job application queuing, and minimal data redundancy practices ensure that the system remains stable during peak job recruitment periods.

### Conclusion

The Job Portal is built on a strong foundation, offering essential features such as secure user authentication, dynamic job recruitment configuration, real-time voting, and result display. However, there remains ample room for growth and enhancement. By incorporating advanced functionalities like multi-factor authentication, mobile app integration, admin (recruiter) analytics, and customizable job recruitment settings, the platform can evolve into a comprehensive, scalable solution for diverse voting needs. The future scope of the system focuses on delivering a secure, user-friendly, and adaptable platform that can serve institutions, organizations, and communities on a broader scale, reinforcing trust and transparency in digital job application processes.

# CHAPTER 7 CONCLUSION

The Job Portal is a modern, secure, and lightweight web-based solution for conducting digital job recruitments in a streamlined manner. Developed using Node.js, MongoDB, ReactJS, TailwindCSS, and basic Express.js, the platform delivers a smooth experience for both job seekers and admin (recruiter)istrators while ensuring data security, transparency, and efficiency.

Throughout the development process, essential modules were built — including secure job seeker registration with OTP-based email verification, encrypted login authentication, employer listing, and a backend-controlled job application-casting mechanism. The admin (recruiter) dashboard was designed to facilitate smooth management of job recruitments, employer records, and real-time result tracking.

The platform ensures:

* + - A robust user authentication system using hashed passwords and Node.js session management.
    - A responsive, user-friendly interface optimized for both desktop and mobile devices.
    - An admin (recruiter) panel that simplifies the process of creating job recruitments, managing employers, and tracking job seeker activity.
    - Use of backend validation to maintain job application integrity and prevent duplicate submissions.

The Job Portal also offers strong future potential — with scope for integrating advanced features like blockchain-based job application verification, biometric-based user authentication, and graphical data analytics for result summaries. These enhancements can further strengthen the platform’s credibility and usability for large-scale or institution-level job recruitments.

In conclusion, this project successfully demonstrates how traditional voting challenges can be addressed using well-established web technologies. It promotes secure and transparent digital job recruitment workflows and sets a foundation for more scalable, digital governance solutions.

# CHAPTER 8 REFERENCES

1. Sharma, A., and Gupta, V., “Developing Secure Node.js Applications: Best Practices and Patterns,” International Journal of Web Security, vol. 12, no. 2, pp. 45–52, 2021.
2. Patel, S., “Database Management with MongoDB: A Practical Guide,” Journal of Data Systems, vol. 18, no. 1, pp. 30–37, 2020.
3. Roy, D., and Mehta, P., “User Authentication and Session Management in Node.js,” International Journal of Software Engineering, vol. 16, no. 3, pp. 60–68, 2019.
4. Jain, N., “Designing User-Friendly Interfaces with ReactJS and TailwindCSS,” UI/UX Trends Journal, vol. 14, no. 4, pp. 22–28, 2020.
5. Bhatt, R., “Secure Job Portals: Design Considerations for Digital Platforms,” Journal of E-Governance Applications, vol. 10, no. 1, pp. 15–21, 2022.
6. Thomas, K., “Role-Based Access Control in Web Applications,” *Web Development Journal*, vol. 11, no. 2, pp. 90–97, 2021.
7. Ahmed, M., “Building Scalable Node.js Web Applications,” Computer Engineering Review, vol. 19, no. 3, pp. 39–46, 2022.
8. Sinha, L., “Real-Time Results Display in Job Recruitment Portals,” International Journal of Online Job Portals, vol. 5, no. 2, pp. 55–61, 2023.
9. Tiwari, R., and Sen, A., “Data Integrity and Validation Techniques in Node.js/MongoDB Applications,” Secure Programming Review, vol. 13, no. 1, pp. 18–25, 2020.
10. Bansal, P., “Developing Lightweight Web Apps for Educational Institutions,” *International Journal of IT in Education*, vol. 9, no. 2, pp. 72–79, 2021.