

MERN Stack Based Smart Job Portal: A Comprehensive Web Application for Efficient Recruitment Management

Nikhil, Robin & Arjun

¹²³ Department of Computer Science and Engineering

Dr. APJ Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India

ABSTRACT

In today's competitive world, the process of finding a job or hiring a suitable candidate has become very challenging and time-consuming. Traditional methods of recruitment such as newspaper advertisements and manual applications are slow, inefficient, and geographically limited. This paper presents the design and development of a Smart Job Portal Web Application built using the MERN Stack (MongoDB, Express.js, React.js, and Node.js). The proposed system provides a unified platform where job seekers can search and apply for jobs easily, and employers can post job openings and manage applications in an organized manner. The system includes features such as user registration and authentication using JSON Web Tokens (JWT), role-based access control for job seekers and employers, real-time job search with filters, resume upload using Multer, and an admin dashboard for complete system management. The application follows a RESTful API architecture, ensuring scalability and ease of integration. The results demonstrate that the MERN-based job portal significantly improves the recruitment experience by reducing manual effort, saving time, and providing a better user experience as compared to traditional methods.

Keywords:

MERN Stack, MongoDB, Express.js, React.js, Node.js, RESTful API, Web Application, Recruitment System, JWT Authentication, Full Stack Development.

1. INTRODUCTION

The rapid growth of the internet and technology has changed the way people look for jobs and how companies hire employees. In the past, job seekers used to read newspapers, visit company offices, or rely on references to find job opportunities. Employers also had to spend a lot of time reviewing physical resumes and conducting manual screenings. These traditional processes were slow, costly, and limited to specific geographic areas.

Today, online job portals have become the most popular way to connect job seekers with employers. Websites like Naukri.com, LinkedIn, Monster, and Indeed are used by millions of people every day. These platforms allow users to post and find jobs from anywhere in the world at any time. However, many existing

systems are either too complex for beginners or lack features that are needed by both small companies and individual job seekers.

This paper presents a Smart Job Portal developed using the MERN Stack, which stands for MongoDB, Express.js, React.js, and Node.js. The MERN Stack is a popular modern technology combination used to build fast, scalable, and full-featured web applications using only JavaScript language. Using the MERN Stack means both the front-end (what the user sees) and the back-end (the server and database) are built with one language, making development simpler and faster.

The proposed job portal allows two types of users: Job Seekers who can create profiles, search jobs, and apply online; and Employers who can register their company, post job vacancies, and review applications. An Admin panel is also included to manage the entire system. The system is designed to be simple, secure, and user-friendly.

1.1 Problem Statement

Despite the availability of many job portals, students and fresh graduates from tier-2 and tier-3 cities in India still face difficulties in finding relevant job opportunities. Many existing platforms are not mobile-friendly, require paid subscriptions, or do not cater to local job markets. Similarly, small and medium enterprises (SMEs) struggle to find an affordable and easy-to-use hiring platform. There is a clear need for a simple, free, and efficient job portal that is easy to use for all types of users.

1.2 Objectives of the Study

- To design and develop a full-stack job portal web application using the MERN Stack.
- To provide a secure login system using JWT (JSON Web Token) authentication.
- To implement role-based access for job seekers, employers, and admin.
- To enable job seekers to search, filter, and apply for jobs online.
- To allow employers to post jobs and manage applications efficiently.
- To build a responsive and user-friendly interface using React.js.
- To evaluate the system based on performance, usability, and user satisfaction.

2. LITERATURE REVIEW

Several researchers and developers have worked on job portal systems and recruitment management systems. A brief review of related work is presented below:

Singh and Gupta (2021) developed an online recruitment system using PHP and MySQL. Their system provided basic features such as job posting and application submission. However, the system lacked real-time updates and was not mobile responsive, which are very important features in today's mobile-first world.

Kumar et al. (2022) proposed a job recommendation system using machine learning algorithms. Their system used collaborative filtering to suggest relevant jobs to users based on their profile and past search history. While their approach improved job matching accuracy, the implementation required heavy computational resources and was difficult to scale.

Patel and Shah (2022) built a React.js and Firebase-based job portal. Their system was fast and real-time due to Firebase's Realtime Database. However, Firebase has limitations in terms of complex queries and pricing for large-scale applications.

Sharma et al. (2023) implemented a full-stack job portal using Node.js and Angular. Their system supported JWT authentication and file upload for resumes. However, Angular has a steep learning curve compared to React.js and the system lacked an admin panel.

Based on the above review, it is observed that while many job portal systems exist, most of them lack one or more of the following: mobile responsiveness, scalable architecture, role-based access control, or a complete admin management system. Our proposed MERN Stack job portal addresses all these limitations in a single, unified application.

3. SYSTEM ARCHITECTURE AND DESIGN

The proposed job portal follows a three-tier architecture consisting of the Presentation Layer, Application Logic Layer, and Data Layer. This architecture separates the user interface, business logic, and database operations, making the system easy to maintain and scale.

3.1 Technology Stack (MERN)

The MERN Stack consists of four key technologies:

Technology	Full Name	Type	Purpose in Project
M	MongoDB	NoSQL Database	Store users, jobs, applications
E	Express.js	Backend Framework	Build REST APIs
R	React.js	Frontend Library	Build User Interface (UI)
N	Node.js	Runtime Environment	Run JavaScript on Server

Table 1: MERN Stack Components and Their Roles

3.2 System Architecture Layers

- **Presentation Layer (Frontend):** Built with React.js. This is what the user sees and interacts with. It includes pages like Home, Login, Register, Job Search, Job Details, Dashboard for Employer, and Application Status for Job Seeker.
- **Application Logic Layer (Backend):** Built with Node.js and Express.js. This layer handles all the business logic, processes requests from the frontend, performs validation, and communicates with the database. RESTful APIs are used for communication.
- **Data Layer (Database):** MongoDB is used as the database. It stores all data in a flexible JSON-like format called BSON (Binary JSON). Collections include Users, Jobs, Applications, and Companies.

3.3 Database Design

MongoDB, being a NoSQL database, stores data in collections. The main collections used in our system are:

Collection	Key Fields	Description
Users	_id, name, email, password (hashed), role, profilePic, resume	Stores all user accounts (job seeker & employer)
Jobs	_id, title, company, location, salary, description, createdBy, createdAt	Stores all job postings by employers
Applications	_id, jobId, applicantId, status, appliedAt, coverLetter	Tracks job applications by seekers
Companies	_id, name, website, logo, description, userId	Stores company profiles for employers

Table 2: MongoDB Collections and Schema Description

4. SYSTEM MODULES AND FEATURES

The job portal is divided into multiple modules. Each module handles a specific part of the application. The main modules are described below:

4.1 User Authentication Module

This module handles all login and registration functionalities. Users can register as either a Job Seeker or an Employer. The password is encrypted using bcrypt.js before storing in the database so it is never saved as plain text. After login, a JWT (JSON Web Token) is generated and sent to the user. This token is stored in the browser and sent with every subsequent request to verify the user's identity. The token expires after a set time for security purposes.

- Registration: User fills name, email, password, and selects role (Job Seeker / Employer).
- Login: User enters email and password. System verifies and returns JWT token.
- Protected Routes: Certain pages (like Apply for Job, Post Job) are only accessible to logged-in users with the correct role.

4.2 Job Seeker Module

This module is designed for users looking for jobs. The key features include:

- Profile Management: Job seekers can add their personal details, skills, education, and work experience to create a complete profile.
- Resume Upload: Users can upload their resume in PDF format using Multer middleware on the backend.
- Job Search and Filter: Users can search jobs by title, location, or company name. Filters include job type (full-time, part-time, remote), salary range, and experience level.
- Job Application: Users can apply to any job with a single click and write a short cover letter.
- Application Tracking: Users can see the status of all their applications (Applied, Shortlisted, Rejected, Selected).

4.3 Employer Module

This module is designed for companies and recruiters who want to hire candidates. The key features include:

- Company Profile: Employers can create a company profile with name, logo, website, and description.
- Post a Job: Employers can post new job openings with details like title, required skills, salary, location, and job type.
- Manage Job Postings: Employers can edit, delete, or close their job postings.
- View Applications: Employers can see a list of all applicants who have applied for their jobs.
- Update Application Status: Employers can change the status of each application to Shortlisted, Rejected, or Selected.

4.4 Admin Module

The admin has full control over the application. Admin features include:

- View and manage all registered users (job seekers and employers).
- Delete fake or spam job postings.
- View total statistics: total users, jobs posted, applications submitted.
- Ban or deactivate accounts that violate terms of use.

4.5 Mentorship Module

The mentorship module is designed as a career development extension to the core job portal. Mentors register their expertise across predefined domain categories (Software Engineering, Data Science, Product Management, UX/UI Design, etc.), set availability schedules using a calendar interface, and define session types (one-time, recurring, project-based). Job seekers can search mentors using multi-criteria filters (domain, experience level, company background, availability) and send connection requests.

Upon acceptance, both parties gain access to a dedicated real-time messaging channel powered by Socket.IO, enabling low-latency text communication with typing indicators, read receipts, and message history persistence in MongoDB. Session scheduling is integrated with a calendar widget that sends browser push notifications for upcoming sessions. Mentorship session outcomes and ratings are stored for future recommendation improvements.

4.6 REST API Design

The backend is built as a RESTful API, meaning all communication between the frontend and backend happens through HTTP requests. The main API endpoints are:

Method	Endpoint	Access	Description
POST	/api/auth/register	Public	Register new user
POST	/api/auth/login	Public	Login and get JWT token
GET	/api/jobs	Public	Get all jobs with filters
POST	/api/jobs	Employer	Post a new job
PUT	/api/jobs/:id	Employer	Update a job posting
DELETE	/api/jobs/:id	Employer/Admin	Delete a job posting
POST	/api/applications/:jobId	Job Seeker	Apply for a job
GET	/api/applications/user	Job Seeker	Get my applications
PUT	/api/applications/:id	Employer	Update application status

GET	/api/admin/users	Admin	Get all users
-----	------------------	-------	---------------

Table 3: REST API Endpoints

5. IMPLEMENTATION DETAILS

5.1 Frontend Implementation (React.js)

The frontend is built using React.js with functional components and React Hooks. The application uses React Router for navigation between pages. Redux Toolkit is used for global state management, which helps to manage data like the logged-in user's information across the entire application without passing props manually.

The UI is designed using Tailwind CSS, which provides ready-made CSS utility classes to create responsive and beautiful designs quickly. Some key frontend pages include:

- Home Page: Shows latest job listings and a search bar.
- Login / Register Page: Allows users to create accounts and login.
- Browse Jobs Page: Shows all available jobs with filter options.
- Job Details Page: Shows full details of a selected job and an Apply button.
- Job Seeker Dashboard: Shows profile, applied jobs, and their statuses.
- Employer Dashboard: Shows posted jobs and received applications.
- Admin Dashboard: Shows statistics, user list, and job management.

5.2 Backend Implementation (Node.js + Express.js)

The backend is a Node.js server running Express.js. All API routes are organized into separate route files for clarity. Middleware functions are used for authentication (verifying JWT tokens) and authorization (checking user roles before allowing access).

The bcrypt.js library is used to hash passwords before saving to the database, ensuring user passwords are never stored in plain text. The jsonwebtoken library is used to create and verify JWT tokens. The Multer middleware is used to handle file uploads (resumes) and Cloudinary integration is used to store uploaded files in the cloud.

5.3 Database Implementation (MongoDB with Mongoose)

MongoDB is used as the database and is accessed through Mongoose, an Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose allows us to define schemas (structure) for our data, which provides validation and easy querying.

All queries are optimized using indexing on frequently searched fields like job title, location, and company name. This ensures fast search results even with a large number of job postings.

5.4 Security Features

- Password Hashing: All passwords are hashed using bcrypt (salt rounds = 10) before being stored.
- JWT Authentication: Every API request to a protected route must include a valid JWT token in the request header.
- Role-Based Access Control (RBAC): Different users (Job Seeker, Employer, Admin) can only access their permitted pages and APIs.
- Input Validation: All form inputs are validated both on the frontend and backend to prevent invalid data and SQL/NoSQL injection attacks.
- CORS Configuration: Cross-Origin Resource Sharing (CORS) is configured to allow only trusted origins to access the API.

6. RESULTS AND DISCUSSION

The developed MERN Stack Job Portal was tested by a group of 30 users including 20 students (as job seekers) and 10 company representatives (as employers) from Dr. APJ Abdul Kalam Technical University, Lucknow. The system was evaluated based on performance, usability, and user satisfaction.

6.1 Performance Testing

The system was tested for response time, load handling, and reliability. The following results were observed:

Test Parameter	Result	Remarks
Average Page Load Time	1.8 seconds	Excellent
API Response Time (avg)	320 ms	Very Good
Concurrent Users Supported	Up to 200 users	Good for small-medium scale
Resume Upload Time	2-4 seconds (PDF)	Acceptable

Job Search Response Time	< 500 ms	Excellent
System Uptime	99.2%	Very Reliable

Table 4: System Performance Testing Results

6.2 User Satisfaction Survey

After using the system for one week, participants were asked to rate various aspects of the portal on a scale of 1 to 5. The average ratings are shown below:

Evaluation Criteria	Avg Rating (out of 5)	Satisfaction Level
Ease of Registration and Login	4.6	Very High
Job Search and Filter Functionality	4.5	Very High
Application Submission Process	4.7	Very High
User Interface and Design	4.4	High
Mobile Responsiveness	4.3	High
Application Status Tracking	4.5	Very High
Employer Job Posting Experience	4.4	High
Overall System Satisfaction	4.5	Very High

Table 5: User Satisfaction Survey Results

The overall average user satisfaction rating of 4.5 out of 5 indicates that the system is well received by both job seekers and employers. 90% of participants said they would prefer using this portal over traditional job search methods.

7. COMPARISON WITH EXISTING SYSTEMS

The proposed system was compared with some popular existing job portals and previously developed systems based on key features:

Feature	Naukri.com	Singh (2021)	Sharma (2023)	Proposed System
MERN Stack	No	No	Partial	Yes
JWT Authentication	Yes	No	Yes	Yes
Admin Dashboard	Yes	No	No	Yes
Resume Upload	Yes	Yes	Yes	Yes

Role-Based Access	Yes	Partial	Yes	Yes
Free to Use	No	N/A	N/A	Yes
Mobile Responsive	Yes	No	Partial	Yes
Open Source	No	No	No	Yes

Table 6: Feature Comparison with Existing Systems

8. CHALLENGES AND LIMITATIONS

8.1 Challenges Faced

- **State Management Complexity:** Managing the global state of the application across multiple components required careful design using Redux Toolkit. Initial setup was challenging for beginners.
- **JWT Token Expiry Handling:** Handling cases where the JWT token expires while the user is still using the system required implementing token refresh logic.
- **File Upload Management:** Handling large resume files and ensuring they are stored and retrieved securely was initially difficult. Cloudinary integration solved this issue.
- **Cross-Origin Issues (CORS):** During development, CORS errors occurred when the frontend and backend were running on different ports. Proper CORS configuration resolved this.

8.2 Current Limitations

- The system currently does not have a built-in video interview or chat feature between employers and job seekers.
- There is no AI-based job recommendation engine to suggest relevant jobs automatically based on the user's profile.
- Email notification system for application updates has not been implemented yet.
- The system has been tested at a small scale and may require further optimization for handling millions of users.

9. FUTURE SCOPE

The current version of the MERN Stack Job Portal provides a strong foundation for a fully functional recruitment platform. Several enhancements are planned for future versions:

1. **AI-Based Job Recommendation:** Implement a machine learning model that analyses the job seeker's profile, skills, and application history to recommend the most relevant jobs automatically.

2. Email and SMS Notifications: Integrate an email service (like Nodemailer or SendGrid) to send automatic notifications to job seekers when their application status changes.
3. Real-Time Chat System: Add a built-in messaging feature using Socket.io so that employers and job seekers can communicate directly within the platform.
4. Video Interview Integration: Integrate a video calling feature (like WebRTC or Zoom API) to allow employers to conduct online interviews directly from the portal.
5. Mobile Application: Develop a mobile app version using React Native, which shares the same backend (Node.js + MongoDB) as the web application.
6. Advanced Analytics Dashboard: Provide employers with detailed analytics such as how many people viewed their job, the most common skills among applicants, and hiring trends.
7. OAuth Integration: Allow users to login using Google or LinkedIn accounts for a faster and more convenient login experience.
8. Government Job Integration: Include a section for government job notifications by fetching data from official government APIs.

10. CONCLUSION

This paper presented the design, development, and testing of a Smart Job Portal Web Application built using the MERN Stack (MongoDB, Express.js, React.js, and Node.js). The system was designed to solve the real-world problems faced by job seekers and employers in the traditional recruitment process.

The portal provides a simple yet feature-rich platform where job seekers can create profiles, upload resumes, search and filter jobs, and track their application status. Employers can post job openings, view candidate profiles, and manage applications efficiently. An admin dashboard ensures proper management and monitoring of the entire system.

The use of JWT authentication, role-based access control, bcrypt password hashing, and proper API design ensures that the system is secure and reliable. The MongoDB database provides flexibility to handle different types of data, while React.js ensures a fast and interactive user experience.

Performance testing results showed excellent response times and reliability, while user satisfaction surveys indicated a high average rating of 4.5 out of 5. The system was well received by both job seekers and employers during testing at Dr. APJ Abdul Kalam Technical University, Lucknow.

In conclusion, the MERN Stack Job Portal is a scalable, secure, and user-friendly solution for modern recruitment needs. It outperforms many existing systems in terms of features, cost (free to use), and ease of development. With the planned future enhancements such as AI job recommendations, real-time chat, and mobile app development, this portal has the potential to become a comprehensive career platform for students and professionals alike.

REFERENCES.

- [1] Singh, R., and Gupta, A. (2021). "Design and Development of Online Recruitment System using PHP and MySQL." *International Journal of Computer Applications*, 175(12), 28-33.
 - [2] Kumar, V., Mishra, S., and Tiwari, R. (2022). "Job Recommendation System Using Machine Learning and Collaborative Filtering." *Journal of Emerging Technologies and Innovative Research (JETIR)*, 9(4), 112-119.
 - [3] Patel, D., and Shah, K. (2022). "Building a Real-Time Job Portal Using React.js and Firebase." *International Journal of Advanced Research in Computer Science (IJARCS)*, 13(3), 45-52.
 - [4] Sharma, P., Verma, N., and Agarwal, S. (2023). "A Full Stack Job Portal Using Node.js, Angular, and MongoDB." *International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)*, 12(2), 88-96.
 - [5] MongoDB Inc. (2024). MongoDB Documentation. Retrieved from <https://www.mongodb.com/docs/>
 - [6] Meta Platforms Inc. (2024). React.js Documentation. Retrieved from <https://react.dev/>
 - [7] OpenJS Foundation. (2024). Node.js Documentation. Retrieved from <https://nodejs.org/en/docs/>
 - [8] OpenJS Foundation. (2024). Express.js Documentation. Retrieved from <https://expressjs.com/>
 - [9] JWT.io. (2024). JSON Web Tokens Introduction. Retrieved from <https://jwt.io/introduction/>
 - [10] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Doctoral Dissertation). University of California, Irvine.
 - [11] Patil, S., and Kulkarni, U. (2023). "Comparative Study of MERN Stack vs MEAN Stack for Web Application Development." *International Journal of Computer Science and Mobile Computing*, 12(6), 22-29.
 - [12] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). "The Semantic Web." *Scientific American*, 284(5), 34-43.
-

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the Department of Computer Science and Engineering, Dr. APJ Abdul Kalam Technical University, Lucknow, for providing the necessary resources, guidance, and support to carry out this research work. We are also thankful to all the students and faculty

members who participated in the testing and evaluation of the proposed system. Special thanks to our project mentor for continuous encouragement and valuable feedback throughout the development process.

© 2026 Nikhil, Robin, Arjun | Dr. APJ Abdul Kalam Technical University, Lucknow