**PROJECT SYNOPSIS REPORT**

**ON**

**Job-Portal Application**

**SUBMITTED**

**TO**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FOR**

**Backend Engineering**



**Submitted To:- Submitted By:-**

Mr. Rahul. Asmita Sharma (2310990307).

Balreet Singh (2310990310).

Bhavisha Ahuja (2310990312).

Dhruv Kumar (2310990325)

# **Index**

|  |  |  |
| --- | --- | --- |
| **SrNo.** | **Topic** | **Page No.** |
| 1 | Problem Statement | 3 |
| 2 | Title of project | 3 |
| 3 | Objective & Key Learning’s | 3 |
| 4 | Options available to execute the project | 4 |
| 5 | Tech Stack | 4 |
| 6 | Advantages/ Disadvantages | 5 |
| 7 | Implementation Strategy | 5-6 |
| 8 | Conclusion | 6 |
| 9 | References | 6-7 |

### **Problem Statement:**

In today’s competitive job market, both employers and job seekers face challenges in efficiently managing recruitment and application processes. Employers struggle to organize job postings, screen candidates, and streamline hiring, while job seekers find it difficult to access structured resources like resume building, interview preparation, and job applications within one platform. Existing solutions are often fragmented, leading to inefficiencies, missed opportunities, and poor user experience.

The **Job-Portal** project addresses these issues by providing a unified platform that simplifies the hiring process for employers and enhances the job-seeking experience for candidates. It integrates role-based access, job posting and application workflows, resume building, and interview preparation tools in one centralized system powered by React, Node.js, Express, and MongoDB.

**Title of Project:**

**Job-Portal**

**Objective & Key Learnings:**

* Develop a role-based authentication system for users and employers.
* Enable employers to post jobs, manage applicants, and streamline hiring.
* Provide job seekers with tools like resume builder, interview prep, and job search.
* Build a real-time notification system for application acceptance/rejection.
* Learn integration of React frontend with Node.js backend and MongoDB database.
* Enhance productivity and user experience through intuitive UI and efficient workflows.

**Options Available to Execute the Project:**

### **Monolithic Architecture:** Simplified initial development with single codebase for backend**.**

### **Microservices Architecture:** Modular and scalable, separating user, employer, and job modules.

### **Frameworks and Tools (React, CSS):** Faster frontend development with polished UI.

### **Database Choices:** MongoDB for flexible schema storage of job postings and applications.

### **Real-Time Communication:** WebSockets or polling for instant notifications.

### **CI/CD Integration:** GitHub Actions or Jenkins for automated deployment and testing.

### **Tech Stack:**

**We will use React, Node.js, Express, and MongoDB due to the following reasons:**

* **React:**
  + Component-based architecture ensures reusable UI.
  + Virtual DOM enables high performance and fast rendering.
  + Strong ecosystem and community support.
* **Node.js & Express:**
  + Handles asynchronous requests efficiently.
  + Provides REST APIs to connect frontend and database.
  + Scalable for large numbers of users and jobs.
* **MongoDB:**
  + NoSQL database allows flexible schema for dynamic job postings.
  + High scalability with built-in sharding and replication.
  + JSON-like document structure makes integration seamless with Node.js.

### **Advantages and Disadvantages:**

**Advantages:**

1. **Centralized System:** One platform for job posting, applications, resume building, and interview prep.
2. **Role-Based Access:** Separate dashboards for employers and users.
3. **Scalability:** Built with Node.js and MongoDB, can handle increasing users and jobs.
4. **Real-Time Notifications**: Instant updates for application status**.**

**Disadvantages:**

1. **Complex Implementation:** Role-based features and multiple dashboards increase development time.
2. **Data Dependency:** Performance relies on accurate job postings and user data.
3. **High Maintenance:** Requires regular updates for resume templates, interview questions, and security patches.

**Implementation Strategy:**

**1. Backend Development:**

* Build REST APIs with Node.js + Express for authentication, job posting, and application handling.
* Use MongoDB to store user profiles, job postings, applications, and notifications.
* JWT-based authentication for secure login.

**2. Employer Module:**

* Role-based login to Employer Dashboard.
* Features: Post Job (title, description, company, salary, location, job type).
* Manage Applicants: View applicant list, accept/reject requests.
* Store job postings and applicant details in database.

**3. User Module:**

* Role-based login to User Dashboard.
* **Features:**
  + **Resume Builder:** Generate resumes in standard format.
  + **Interview Prep:** Practice DSA, aptitude, and subject-specific assessments.
  + **Job Search & Apply:** Browse jobs posted by employers, submit applications.
* Application status (accepted/rejected) synced with employer actions.

**4. Notifications:**

* Real-time notifications for application updates using WebSockets or server-side events.

**5. Frontend Development:**

* Develop responsive UI using React.js.
* Use React Router for navigation (Landing → Login → Dashboard).
* Interactive forms for job posting, resume creation, and interview prep.

**6. Security Measures:**

* JWT for authentication.
* HTTPS for secure communication.
* Validation and sanitization of input data to prevent injections.

**7. Testing and Deployment:**

* Perform unit, integration, and UAT (User Acceptance Testing).
* Use Docker for containerization and Kubernetes for orchestration.
* Deploy on cloud (AWS/Heroku/Render) for scalability and availability

### **Conclusion:**

The **Job-Portal Project** offers an integrated and efficient solution for both employers and job seekers. By combining features like job posting, resume building, interview preparation, and real-time application management, it bridges the gap between candidates and recruiters. Built with **React, Node.js, Express, and MongoDB**, the system is scalable, secure, and user-friendly. With role-based dashboards, workflow automation, and centralized data, the portal simplifies hiring while empowering candidates to enhance their career prospects.

**References:**

* [**https://nodejs.org/en**](https://nodejs.org/en)
* **https://react.dev/**
* [**https://expressjs.com/**](https://expressjs.com/)
* [**https://www.mongodb.com/docs/**](https://www.mongodb.com/docs/)
* **https://www.geeksforgeeks.org/mern-stack/**
* **https://developer.mozilla.org/en-US/docs/Learn**