**JOB PORTAL**

**Team Member’s Name:**

1. Akash Das [BWU/BTA/21/002]
2. Subhamay Ganguly [BWU/BTA/21/011]
3. Raktim Bhattacharya [BWU/BTA/21/018]
4. Shubham Swarnakar [BWU/BTA/21/034]

### **1. Overview**

The Job Hiring Portal is designed to streamline the process of job postings and applicant management. Its primary functionalities include:

* **Job Listings:** Employers can post job opportunities with relevant details such as job title, description, salary, and location.
* **Applicant Management:** Employers can view, sort, and manage applications submitted by candidates.
* **User Roles:**
  + Employers: Create and manage job listings, view applications, and communicate with candidates.
  + Applicants: Search for jobs, submit applications, and track application statuses.

This application leverages modern web technologies for an efficient and responsive user experience.

### **2. Project Structure**

The application follows a modular structure for easy maintenance and scalability.

#### **Backend:**

* **models/**: Contains database schemas for entities like users, jobs, and applications. Using an Object-Relational Mapping (ORM) tool like Mongoose simplifies database interactions.
* **node\_modules/**: Houses third-party libraries automatically generated when dependencies are installed via npm install.
* **package.json**: Defines the project's metadata, scripts, and dependencies. For example, dependencies may include Express.js for routing and Mongoose for database management.
* **routes/**: Contains API definitions for endpoints like:
  + /jobs: For managing job listings.
  + /applications: For handling applicant data.
  + /users: For authentication and user roles.
* **server.js**: Acts as the entry point for the backend, initializing the server and connecting to the database.

#### **Frontend:**

* **jobportal/**: Includes all frontend code organized into components, services, and assets.
  + **Components:** Individual UI elements, such as job cards, application forms, and dashboards.
  + **Services:** Handle API interactions, such as fetching job listings or submitting applications.

### **3. Setup Instructions**

#### **Backend Setup:**

1. Navigate to the backend directory.
2. Install dependencies:
3. npm install

This installs essential libraries like Express.js, Mongoose, and possibly authentication packages like Passport.js or JWT.

1. Start the server:
2. node server.js

Ensure MongoDB is running locally or configure the connection string for a cloud database.

#### **Frontend Setup:**

1. Navigate to the jobportal directory.
2. Install dependencies:
3. npm install

This includes libraries like React, Axios for API calls, and React Router for navigation.

1. Start the development server:
2. npm start

Open the browser at the provided URL to view the application.

### **4. API Details**

The API enables smooth communication between the frontend and backend. Some examples include:

* **Job API:**
  + GET /jobs: Fetch all job listings.
  + POST /jobs: Add a new job (requires authentication).
  + PUT /jobs/:id: Update job details.
  + DELETE /jobs/:id: Remove a job.
* **Application API:**
  + POST /applications: Submit a new application.
  + GET /applications/:jobId: View applications for a specific job.
* **User API:**
  + POST /users/register: Register a new user.
  + POST /users/login: Authenticate a user and generate a token.

### **5. Dependencies**

#### **Backend Dependencies:**

* **Node.js:** Runtime for executing JavaScript server-side.
* **Express.js:** Lightweight framework for building robust APIs.
* **MongoDB:** NoSQL database for flexible and scalable data storage.
* **Mongoose:** ODM library for MongoDB.

#### **Frontend Dependencies:**

* **React:** Component-based library for building interactive UIs.
* **Axios:** Promise-based HTTP client for API requests.
* **React Router:** Manages navigation between different views in the application.
* **Bootstrap/Tailwind CSS (optional):** For styling and layout.

**THANK YOU**