Assignment: Job Posting Board with Email Automation

DEADLINE: 27-01-2025

Tech Stack: MERN (MongoDB, Express.js, React.js, Node.js)

Objective

Design and implement a full-stack job posting board where companies can register, verify their accounts via email, post jobs, and send automated emails to candidates. This project will allow you to apply your knowledge of the MERN stack, authentication, and email automation.

[BUILD A BASIC UI TO ACHIEVE THE FEATURES/FUNCTIONALITY]

Assignment Details

Functional Requirements

1. **User Registration (Company):**
   - Companies should be able to **register** by providing basic details.
   - **Email and mobile verification** must be implemented to activate the account. Unverified users should not be allowed to post jobs.
2. **Company Login:**
   - Implement an **auto login system** using JWT cookie based authentication.
3. **Job Posting:**
   - An authenticated company can **post jobs** with the following details:
     - Job title
     - Job description
     - Experience Level eg: BEGINNER, INTERMEDIATE, EXPERT
     - Add candidate eg: add student email id to send email regarding job post
     - End date
4. **Candidate Email Automation:**
   - Add functionality for companies to **send job alerts** or updates to multiple candidates via email.
   - Use a service like **Nodemailer** to automate emails.

    - Ensure emails contain:
        - Job details
        - Sender information
5. **Logout:**
    - Provide a logout option to clear tokens or sessions.

Technical Requirements

- **Frontend (React.js):**
    - Use **React/Next.js**
    - Include forms for company registration, login, and job posting.
    - Make sure to all kind of validations and error handling
- **Backend (Node.js & Express.js):**
    - Set up RESTful APIs for user registration, login, job posting, and email automation.
    - Use **Nodemailer** (or any other email service) to handle email sending.
    - Implement validation for inputs to ensure data consistency.
    - Make sure to handle error gracefully
- **Database (MongoDB):**
    - Store company details, job postings, and email logs in MongoDB.
- **Authentication:**
    - Use **JWT** or session-based authentication to protect routes.

Evaluation Criteria

- Code is clean, readable, and well-structured.
- Proper modularization with clear separation of concerns (e.g., routes, controllers, services).
- Adherence to JavaScript/Node.js best practices (e.g., consistent use of async/await).
- APIs work as per requirements, handling success, errors, and edge cases correctly.
- Appropriate use of HTTP methods (GET, POST, PUT, DELETE) and status codes (200, 400, 404, 500).
- Input validation is implemented, and error responses are meaningful.
- MongoDB schema design aligns with problem requirements.
- CRUD operations (Create, Read, Update, Delete) are implemented correctly.
- Database errors are handled gracefully without crashing the application.
- Inputs are sanitized to prevent vulnerabilities like injection attacks.
- Sensitive information (e.g., database URIs, API keys) is stored in .env files.
- Assignment has been tested using tools like Postman.
- Edge cases (e.g., invalid IDs, missing fields, non-existent resources) are handled properly.
- Pagination is implemented for endpoints returning large datasets where applicable.

- Logs are implemented for debugging and monitoring using a logging library (e.g., Winston, Morgan).
- Proper error handling middleware is used to catch and respond to unexpected errors.
    - A global error handler is implemented to standardize and centralize error responses across the application.
- Middleware is implemented for request validation, authentication, error handling, and logging.
- A health check API is implemented to verify the application's status and readiness (e.g., `/health` returning status `200 OK`).

Submission Guidelines

- Upload your code to **GitHub** and submit the public repository link.
- Include a **README** file explaining how to run the project locally.
- Ensure your project is deployed (e.g., on **Vercel/Netlify** for frontend and **Render/Heroku** for backend) and submit the live demo link.
- Submit a 3-4 mins video explaining the working of the app. Your face should be visible while explaining the project. Use loom to make video (https://www.loom.com/)

Submission link - https://forms.gle/ckV2rBUtdrRp4tiD8