Fullstack Practical test

Objective:

Evaluate hands-on skills in API development, ReactJS integration, MongoDB operations, file handling, scheduling, and real-time communication.

Duration:

30- 40 hrs (one week)

Evaluation Topics:

1. **NodeJS:** Devlop API using NodeJS
2. **ReactJS**: Develop Frontend using ReactJS
3. **MongoDB** – Database schema design and data operations
4. **File Upload Handling** – Support for uploading files with validation and structured storage
5. **Email Functionality** – Trigger email notifications using SMTP or email service providers
6. **Image Processing** – Crop and resize images using a suitable library (e.g., Sharp)
7. **JWT Authentication** – Secure login and route protection using JSON Web Tokens
8. **Redis Integration** – Use for session management or caching
9. **Scheduled Jobs (Cron)** – Automate routine tasks such as email reminders
10. **WebSockets (Socket.io)** – Real-time communication for updates and notifications
11. **CSV Export** – Generate and download data reports in CSV format
12. **API Documentation (Swagger)** – Document all endpoints with an OpenAPI specification
13. **Docker Support** – Containerize the application using Docker for consistent deployment

Requirements:

1. Authentication & Authorization (API and Frontend)

- Implement **user registration** and **login** using **JWT**.
- Passwords must be securely hashed.
- Protect certain routes using JWT middleware.

2. User Profile (API and Frontend)

- Upload profile image.
- Image should be **cropped and resized** (e.g., 200x200 px) using a library like `sharp`.
- Store image URL in MongoDB.

3. Task Module (API and Frontend)

- Users can **create**, **update**, and **delete** tasks.
- Each task should include:
    - Title, Description, Status (Pending, Completed)
    - Due Date
    - File attachments (PDF, DOCX, JPG — max 10MB)
- Files should be stored in a folder structure `/uploads/userId/taskId/`.

4. Email Notifications (API)

- Send an email on task creation and task completion.
- Use nodemailer (or any SMTP provider like SendGrid).

5. Download CSV Report (API and Frontend)

- Endpoint to **download all tasks as CSV**.
- CSV should include: Task ID, Title, Status, Created Date, Due Date.

6. Redis Usage (API)

- Store session or user login token in Redis with a 24-hour TTL.
- Optional: Use Redis to cache user profile data for quick retrieval.

7. Cron Job (API)

- Run a **cron job every day at 8 AM** to:
    - Check for tasks due today
    - Send reminder emails to respective users

8. Socket.io Integration (API and Frontend)

- Real-time update for task status changes:
    - Notify connected clients (browser tab) when a task's status is updated by any user.

9. API documentation should be in swagger

Tech stacks:

1. **Node.js (Express.js)**
2. **ReactJS**
3. **MongoDB** (Mongoose)
4. **JWT** for authentication

5. **Redis**
6. **Socket.io**
7. **Nodemailer** (or any email service)
8. **Multer** for file upload
9. **Sharp** for image resizing
10. **node-cron**
11. **fast-csv** or `json2csv` for CSV generation
12. **Swagger**
13. **Docker**