Assessment

Project: Task Management Application

Overview: Create a Task Management Application where users can add, view, edit, and delete tasks. The application should have a frontend built with **React/Next.js** and a backend built with Node.js and Express. Tasks will be stored in a JSON file or an in-memory array for simplicity. Additionally, the application should demonstrate proficiency in CSS, Bootstrap, Material UI, deployment, and version control with Git.

Requirements :

Backend with Node.js and Express

1. **Set Up Express Server:**
    - Set up a Node.js server with Express to handle HTTP requests.
    - Create endpoints to handle the CRUD (Create, Read, Update, Delete) operations for the tasks.
2. **Implement CRUD Operations:**
    - Create a tasks array or read from a tasks.json file.
    - Implement the API endpoints to handle CRUD operations:
        - `GET /tasks` - Get all tasks.
        - `POST /tasks` - Add a new task.
        - `PUT /tasks/:id` - Edit a task.
        - `DELETE /tasks/:id` - Delete a task.
3. **Test API Endpoints:**
    - Use Postman or a similar tool to test the API endpoints.

Frontend with React/Next.js

1. **Set Up React/Next.js Project:**
    - Set up a **React/Next.js** project.
    - Ensure React Router or Next.js routing is set up to navigate between pages.
2. **Create Components and Services:**
    - Create components for:
        - Listing tasks
        - Adding/editing tasks
        - A task form component
    - Create a service to handle HTTP requests to the backend API.
3. **Implement the Task Components and Service:**
    - Use React forms for adding/editing tasks.
    - Use **React fetch** or **Axios** to make API calls to the Node.js backend.
4. **Create Task Model:**
    - Define a Task model to represent the task structure with fields: `id`, `title`, `description`, and `completed`.

CSS, Bootstrap, and Material UI

1. **Responsive Design:**
    - Ensure the application is fully responsive using **Bootstrap** or CSS Flexbox/Grid.
    - Demonstrate how the application looks on different screen sizes (desktop, tablet, mobile).
2. **Bootstrap Integration:**
    - Integrate **Bootstrap** to style the task list, forms, and buttons.
    - Use Bootstrap components such as modals, alerts, and navbars.
3. **Material UI:**
    - Replace basic HTML components with **Material UI** components.
    - Use **Material UI** table for the task list, Material UI form controls for the task form, and Material UI buttons.
4. **Enhanced UI/UX:**
    - Implement a modal dialog for adding/editing tasks instead of a separate page.
    - Add user feedback with Material UI snackbars for successful operations like task creation, updates, and deletions.
5. **Task Status Indicator:**
    - Use CSS to visually distinguish completed tasks from incomplete ones, such as strikethrough or different background colors.
    - Implement progress bars or badges for task priority using Bootstrap or Material UI.

Deployment and Version Control

1. **Git Repository:**
    - Initialize a Git repository for the project.
    - Make regular commits to the repository to show progress and document changes.
2. **Deploy the Application:**
    - Deploy the Node.js backend on a service like Render.
    - Deploy the **React/Next.js** frontend on a service like **Netlify** or **Vercel**.
    - Ensure the deployed application works seamlessly with the backend.

THANK YOU !