Project: Personal Task Manager

Description:
Develop a web application that allows users to manage their daily tasks. Users should be able to create an account, log in, and manage a list of tasks. Each task should have a title, description, due date, and status (e.g., pending, in-progress, completed).

Features:

1. User Authentication:
    - Sign up and log in functionality.
    - Password encryption (e.g., using bcrypt).
2. Task Management:
    - CRUD operations (Create, Read, Update, Delete) for tasks.
    - Mark tasks as complete/incomplete.
    - Filter tasks by status.
3. UI Components:
    - Task list with sorting and filtering options.
    - Form for adding and editing tasks.
    - User profile page to update user details.
4. Backend:
    - RESTful API using Node.js and Express.
    - Database integration (e.g., MySql) to store users and tasks.
5. Frontend:
    - Responsive design using React.
    - State management (e.g., using Redux or Context API).

Detailed Breakdown:

Backend (Node.js & Express):

- User Routes:
    - `POST /signup`: Create a new user.
    - `POST /login`: Authenticate a user and generate a token.
    - `GET /profile`: Get user details.
    - `PUT /profile`: Update user details.
- Task Routes:
    - `GET /tasks`: Get all tasks for the authenticated user.
    - `POST /tasks`: Create a new task.

        - `GET /tasks/:id`: Get a specific task.
        - `PUT /tasks/:id`: Update a task.
        - `DELETE /tasks/:id`: Delete a task.

Frontend (React):

- Pages:
    - Home Page: Welcome message and navigation.
    - Sign Up and Log In: Forms for user authentication.
    - Dashboard: Display user's tasks with filtering options.
    - Task Form: Form to add or edit tasks.
    - Profile: User profile page to update details.
- Components:
    - TaskList: Display a list of tasks.
    - TaskItem: Display individual task details.
    - TaskForm: Form to handle task creation and updates.
    - NavBar: Navigation bar with links to different pages.

Additional Instructions:

1. Setup Instructions:
    - Provide a README file with detailed setup instructions for both frontend and backend.
    - Include instructions for running the development servers and connecting to the database.
2. Code Quality:
    - Encourage the use of ESLint for code linting.
    - Ensure proper commenting and documentation of code.
    - Implement unit tests for critical functions and components.
3. Version Control:
    - Use Git for version control.
    - Create a GitHub repository for the project.
    - Use branches for new features and ensure proper pull requests and code reviews.