Project Title: Task Manager App

Goal:

Participants will create a task management application using their knowledge of Node.js, React, and PostgreSQL. This project will reinforce REST API design, database integration, React state management, and CRUD functionality.

Features & Functionality

Core Features:

1. **User Management:**
   - Users can register, log in, and log out.
   - Authentication using JWT.
   - Passwords should be hashed for security (e.g., using bcrypt).
2. **Task Management:**
   - Users can create, update, delete, and view their tasks.
   - Each task should include:
     - Title
     - Description
     - Status (`To-Do`, `In Progress`, `Done`)
     - Priority (`Low`, `Medium`, `High`)
     - Due Date
   - Tasks should be visible only to the user who created them.
3. **Search and Filter:**
   - Search tasks by title or description.
   - Filter tasks by status or priority.
4. **Task Sorting:**
   - Sort tasks by due date, priority, or creation date.
5. **Dashboard:**
   - A dashboard that shows an overview of tasks:
     - Total tasks
     - Tasks by status (e.g., number of `To-Do`, `In Progress`, `Done`).

API Endpoints

Auth Routes

- `POST /api/register`
  Create a new user.
- `POST /api/login`
  Authenticate a user and return a JWT.
- `POST /api/logout`
  Log out the user (optional, e.g., to blacklist the token).

User Routes

- `GET /api/user`
  Fetch user profile (requires authentication).

Task Routes

- `GET /api/tasks`
  Fetch all tasks for the authenticated user (support query params for search, filter, and sort).
- `POST /api/tasks`
  Create a new task.
- `GET /api/tasks/:id`
  Fetch a specific task by its ID.

- `PUT /api/tasks/:id`
  Update a task.
- `DELETE /api/tasks/:id`
  Delete a task.

UI Requirements

Login & Registration Pages:

- Form for login and registration with client-side validation.

Dashboard:

- Overview of tasks grouped by status (e.g., To-Do, In Progress, Done).
- Summary widgets for task counts.

Task List Page:

- Display tasks in a list or card view.
- Options to search, filter, and sort tasks.
- Buttons to edit or delete tasks.

Task Form Page:

- Form for creating or editing a task.
- Fields: Title, Description, Status, Priority, Due Date.

Responsive Design:

- Make the UI mobile-friendly.

Additional Requirements

1. **Backend:**
   - Use Node.js with Express.js.
   - Use Prisma as ORM.
   - Comply to starter architecture which will be provided.
2. **Frontend:**
   - Use React with functional components.
   - Use a state management library like Context API or Redux.
3. **Error Handling:**
   - Display appropriate error messages for failed operations (e.g., unauthorized, validation errors).
4. **Validation:**
   - Validate data on both frontend and backend.
5. **Bonus Features:**
   - Task reminders (e.g., via email or notifications).
   - Drag-and-drop for task status updates.
   - Collaborative tasks (assign tasks to other users).