Event Scheduling Web Application - PERN Stack Assignment

Overview

The goal of this task is to assess your understanding of full-stack development using **React**, **Node.js**, **Express**, and **PostgreSQL**.
You are required to design and develop a simple **Event Scheduling Application** that allows users to create, view, and join events (with best coding practices and code commenting).

Objectives

- Demonstrate proficiency in backend API design, database schema creation, and frontend development.
- Practice state management, API integration, and basic error handling.
- Develop a responsive React frontend.
- Follow clean code principles, including modular design and commenting.
- Avoid using AI tools; focus on writing your own logic.

Technology Requirements

- **Frontend:** React (you may use Tailwind CSS or basic CSS for styling)
- **Backend:** Node.js with Express.js
- **Database:** PostgreSQL
- **Optional:** Any ORM or query builder (like Sequelize, Prisma, or Drizzle)
- **Authentication:** JSON Web Token (JWT) based user login/signup

Core Features

1. User Management

- Users should be able to sign up and log in.
- Each user should have a profile containing basic information such as name, email, and registration date.
- Only logged-in users can create or join events.

2. Event Management

- Logged-in users can create, edit, and delete their own events.
- Each event should include details like title, description, date, time, and location.
- All users should be able to view the list of available events.

3. Join / Leave Events (RSVP System)

- A logged-in user can join or leave any event.
- An event should show the list of attendees who have joined.
- A user can only join an event once and can leave it later if they want.

4. Frontend Functionality

- Pages should include:
    - **Login / Signup**
    - **All Events** (listing page)
    - **Event Details** (showing information and attendees)
    - **Create / Edit Event**
- Smooth navigation between pages using React Router.
- Display appropriate success or error messages for actions.

Best Practices

- Use functional components and hooks.
- Maintain a clean component structure.
- Handle API errors with user feedback.
- Comment complex logic or state management parts.

Coding Standards

- Follow consistent indentation (2 spaces or tabs).
- Use descriptive variable and function names.
- Add comments where logical or complex code appears.

- Write modular code: separate API calls, components, and utility functions.
- Avoid using external AI-assisted code suggestions or auto-writing.

Submission

- Upload your code to a GitHub repo with commits showing your development process.
- Include a README with setup instructions.
- Optionally, prepare a brief demo video or screenshots.

Timeline

- Assignment time: 3-5 days.
- Provide regular updates or queries as needed.

Final Notes

This assignment is designed to test your ability to build a full-featured web app following best practices without AI assistance. Focus on writing clear, maintainable code, and document your work thoroughly.