Test Task: Smart Meeting Scheduler

Objective:
Develop a **Meeting Scheduler** application that enables users to manage meetings by avoiding overlaps and suggesting free time slots for scheduling.

Task Description

Scenario:

Users can register, log in, and schedule meetings with start and end times. The system must ensure that meetings for the same user do not overlap. If an overlap is detected, the system should suggest the next available slot.

Requirements

Backend (Node.js + Express + MongoDB):

1. **Authentication**:
   - Use **JWT-based authentication**.
   - Endpoints for:
     - User registration.
     - User login.
2. **Meeting Management**:
   - Endpoint to **add a meeting**:
     - Fields: `title`, `start_time`, `end_time`, `description`.
     - Validate that no two meetings for the same user overlap.
     - If there's a conflict, return an error with a **suggested free slot** for the meeting duration.
   - Endpoint to **fetch all meetings** for a logged-in user.
   - Endpoint to **fetch free time slots** for a specific day.
3. **Database Design**:
   - Use collections for `users` and `meetings`:
     - `users`: Stores user information.
     - `meetings`: Stores meeting details (`user_id`, `start_time`, `end_time`, etc.).
   - Optimize queries for overlap detection and free slot calculation.

Frontend (React + TypeScript + Redux):

1. **Login/Registration Page**:
   - Implement user authentication using JWT.

    - Store user state in **Redux**.
2. **Meeting Scheduler Dashboard**:
    - **Form**:
        - Add a meeting with fields: `title`, `start_time`, `end_time`, `description`.
        - Display validation errors for overlapping meetings.
    - **List View**:
        - Display all scheduled meetings with pagination and sorting.
    - **Free Slot Viewer**:
        - Display available time slots for a selected day.
3. **Bonus Points**:
    - Implement a **calendar view** (e.g., using `react-calendar`) for meetings.
    - Allow users to **edit or delete meetings**.
    - Add **local storage/session persistence** for user login state.
4. **Styling**:
    - Use **Material-UI** or **Ant Design** for a professional and responsive design.