## Task: Medical Appointment Booking System with CRUD Operations

**Timeline:** 4 days for completion. Tasks not submitted within this timeframe will not be accepted.

**Overview:**

Build a **Medical Appointment Booking System** where users can:

1. Register and log in.
2. View available doctors and their schedules.
3. Book appointments.
4. View their appointment details.
5. Perform CRUD operations (Create, Read, Update, Delete) on appointments.

**Requirements:**

**1. Frontend**

- Use **React** with **Material-UI** (MUI) or **Tailwind CSS** for styling and component design.
- **Pages/Components**:
    - **Registration Page**:
        - Create a form to register a new user (username, email, password).
        - Use **bcrypt.js** to hash the password before sending it to the backend.
    - **Login Page**:
        - A login form to authenticate users using username/email and password.
        - Store the **JWT** token in **localStorage** for authenticated requests.
    - **Doctor List Page**:

        - Display a list of doctors with their specialties, availability, and profile information.
        - Add a filter/search option by specialty.
    - **Appointment Booking Page**:
        - Provide a form to select a doctor, date, and time (based on availability).
        - Show available timeslots dynamically from the backend.
        - Allow users to book an appointment and show a confirmation upon success.
    - **User Dashboard**:
        - Show all upcoming and past appointments for the logged-in user.
        - Allow users to **Edit** (Update) and **Delete** their appointments.
        - Show the option to view, edit, and delete their booked appointments.

**2. Backend**

- Use **Node.js** with **Express.js**.
- **Authentication**:
    - **POST /register**: Register a new user (store username, email, password).
    - **POST /login**: Authenticate the user and return a JWT token.
- **CRUD Operations on Appointments**:
    - **GET /appointments**: Fetch all appointments for the logged-in user (protected route).
    - **POST /appointments**: Book a new appointment (protected route with JWT token).
    - **PUT /appointments/:appointmentId**: Edit (Update) an existing appointment (protected route).

    - **DELETE /appointments/:appointmentId**: Delete an appointment (protected route).
- **Doctor Information**:
    - **GET /doctors**: Fetch a list of doctors along with their specialties and availability.
- Use **MongoDB** for storing:
    - **Users**: Store user details (username, email, hashed password).
    - **Doctors**: Store doctor details (name, specialty, schedule).
    - **Appointments**: Store user bookings (doctor, user, date/time, status).

**3. Frontend Integration for CRUD Operations**

- Use **React** to handle:
    - **Create**: Form to book an appointment and send a POST request to the backend.
    - **Read**: Fetch and display the list of available doctors and booked appointments.
    - **Update**: Allow users to edit the date/time of a booked appointment.
    - **Delete**: Provide a **Delete** button next to appointments for cancellation.
- Each user's appointments should be dynamically rendered with options to **Edit** or **Delete**.

**4. Deployment**

- Host the project on **GitHub** for code review.
- Deploy the application:
    - **Backend**: Host on **Any free hosting service**.
    - **Frontend**: Host on **Vercel/Netlify/ Any free hosting service**
- Provide the live demo links for both frontend and backend.

## 5. Optional Bonus Points

- Implement **JWT authentication middleware** for protecting the **GET**, **POST**, **PUT**, and **DELETE** appointment routes.
- Use **Redux Toolkit** for state management to handle user sessions and appointment data.
- Add **Pagination** for the doctor list and appointment display.
- Display **error/success** messages using Material-UI **Snackbars**.
- Implement **password reset** functionality (optional).

## Deliverables:

1. GitHub repository link with clear instructions in a README.md file on how to run the project locally.
2. Live demo links for both frontend and backend.
3. A brief document explaining:
    - Project structure.
    - Design decisions.
    - Any additional features implemented.

**Note**: This task must be completed within **4 days**. Tasks not submitted within this timeframe will not be accepted.

If any point tasks listed above are taking extra time, consider them **optional**. Focus on the core requirements first.

Let me know if you need anything else!

**Contact Details (for any doubts):**
Afroz Pasha
Lead Developer at Shuraa Group
Email: afroz@shuraa.com
Phone: +971 50 125 8161