<u>Use Case: Healthcare Appointment System</u>

Objective: Develop a comprehensive Healthcare Appointment System where patients can book appointments and manage their schedules.

Instructions

1. **Understand the Requirements:** Review the features, acceptance criteria, and technology stack to understand the project's scope.
2. **Coding Platform:**
    - **Backend:** Java, Spring Boot and MySQL.
    - **Frontend:** Angular/React/Vue.js.
3. **Design the Database Schema:** Create tables for Users, Patients, Appointments, Medical_Records, and Bills in MySQL.
4. **Implement User Management:**
    - Create endpoints for registering patients, logging in, and managing profiles.
5. **Implement Appointment Booking:**
    - Create endpoints for booking, viewing, and canceling appointments.
    - Mock the notification service for appointment confirmations and cancellations.
    - Doctor data or module can be mocked for implementation and testing purposes.
6. **(Optional) Implement Additional Features:**
    - Create endpoints for viewing medical records and billing history.
7. **Test the Application:**
    - Write unit tests and integration tests to ensure all functionalities work as expected(**Optional at last write this**).
8. **Deploy the Application:**
    - Locally deploy the backend and frontend to a server and ensure the application is accessible.
9. **Demo the Application:**
    - Demonstrate the registration, login, appointment booking, and cancellation features.

User Role:

- Patient

Features:

1. **User Management:**
    - **Mandatory:**
        i. **Register:** Patients can create an account with personal details and medical history.
        ii. **Login:** Patients can log in using their email and password.
        iii. **Profile Management:** Patients can view and update their profile information.

        iv. **Book Appointment:** Patients can search for doctors and book appointments.
        v. **View Appointments:** Patients can view upcoming and past appointments.
        vi. **Cancel Appointment:** Patients can cancel appointments.
   - **Optional**
        i. **View Medical Records:** Patients can view their medical history and past appointments.
        ii. **View Billing History:** Patients can view the history of their bills.

<u>**Acceptance Criteria:**</u>

Mandatory:Patient Registration,Patient Login,Appointment Booking and Cancel Appointment

1. **Patient Registration:**
   - **Scenario: Patient registers with personal details.**
     - **Given:** A patient provides valid personal details and medical history.
     - **When:** The patient submits the registration form.
     - **Then:** An account is created, and the patient receives a confirmation email.
   - **Scenario: Patient registration with missing details.**
     - **Given:** A patient provides incomplete personal details.
     - **When:** The patient submits the registration form.
     - **Then:** An error message is displayed, indicating the missing fields.
2. **Patient Login:**
   - **Scenario: Patient logs in with correct credentials.**
     - **Given:** A registered patient with valid email and password.
     - **When:** The patient submits the login form.
     - **Then:** The patient is redirected to their dashboard.
   - **Scenario: Patient login with incorrect credentials.**
     - **Given:** A registered patient with an invalid email or password.
     - **When:** The patient submits the login form.
     - **Then:** An error message is displayed, indicating invalid credentials.
3. **Appointment Booking:**
   - **Scenario: Patient books an appointment with a doctor.**
     - **Given:** A logged-in patient and a doctor with available slots.
     - **When:** The patient selects a slot and confirms the booking.
     - **Then:** The appointment is created, and both patient and doctor receive a confirmation notification**(Mock the notification service).**
   - **Scenario: Patient books an appointment with no available slots.**
     - **Given:** A logged-in patient and a doctor with no available slots.
     - **When:** The patient tries to book an appointment.
     - **Then:** An error message is displayed, indicating no available slots.
4. **Cancel Appointment:**
   - **Scenario: Patient cancels an upcoming appointment.**

        - **Given:** A logged-in patient with a scheduled appointment.
        - **When:** The patient selects to cancel the appointment.
        - **Then:** The appointment is canceled, and both patient and doctor receive a cancellation notification**(Mock the notification service).**

Technology Stack:

Backend:

- Java: Core programming language.
- Spring Boot: Framework for building the RESTful API.
- Spring Security: For authentication and authorization.
- Spring Data JPA: For database access.
- MySQL: Relational database.

Frontend:

- Angular/React/Vue.js: Frameworks for building the application.

Final Output you should be able to deploy the model you have have develop and demo that

Database Schema:

Tables:

1. **Users:** This table stores the basic information for all users, regardless of their role (either Patient or Doctor). Stores information about system users, including their name, email (unique identifier), password, role (either 'Patient' or 'Doctor'), contact number, address, and date of birth.

2. **Patients :** This table stores information specific to patients. It links to the Users table via a foreign key. Linked to the Users table via a foreign key (user_id), storing patient-specific details such as medical history.

3. **Appointments:** Manages scheduled appointments between patients and doctors, with fields for patient and doctor IDs (foreign keys referencing Patients and Doctors tables respectively), appointment date, and status ('Scheduled', 'Completed', or 'Cancelled').

4. **Medical_Records:** Tracks detailed medical records, associated with specific patients and doctors through foreign keys (patient_id and doctor_id), and linked to appointments (appointment_id). It includes a field for recording detailed medical notes (record_details) and a timestamp (updated_at) for the latest update.

5. **Bills:** Manages billing information related to appointments, with fields for patient and appointment IDs (foreign keys referencing Patients and Appointments tables respectively), amount due, billing status ('Pending' or 'Paid'), and the timestamp (generated_at) when the bill was generated.

API Endpoints:

1. **User Management(Mandatory):**
   - POST /api/register: Register a new patient.
   - POST /api/login: Authenticate user and return token.
   - GET /api/profile: Retrieve patient profile.
   - PUT /api/profile: Update patient profile.
2. **Appointment Booking(Mandatory):**
   - GET /api/doctors: Retrieve a list of doctors.
   - GET /api/doctors/{id}/availability: Retrieve availability of a specific doctor.
   - POST /api/appointments: Book an appointment.
   - GET /api/appointments: Retrieve a list of appointments for the logged-in patient.
   - DELETE /api/appointments/{id}: Cancel an appointment.
3. **Medical Records(Optional):**
   - GET /api/medical-records: Retrieve medical records for the logged-in patient.
4. **Billing(Optional):**
   - GET /api/bills: Retrieve billing history for the logged-in patient.