Coding Challenge : Healthcare Appointment Booking System

Objective

You are required to build a **Healthcare Appointment Booking System** that allows patients to book appointments with doctors. Your system will consist of:

- A **FastAPI backend** that:
  - Manages doctor availability and appointment bookings.
  - Ensures proper appointment scheduling with slot limitations.
  - Handles **cancellation logic** and frees up slots.
- A **ReactJS frontend** that:
  - Displays available doctors and their slots.
  - Allows users to book and cancel appointments.
- A **Python SDK** (generated using OpenAPI Generator CLI) for programmatic access to the system.
- **Automation scripts** (PowerShell, Shell, or any other script) to simplify setup and execution.

Your solution should demonstrate **best practices** in:

- **FastAPI integration** with database handling.
- **React UI flows** for scheduling and managing appointments.
- **OpenAPI documentation** for SDK generation.
- **Automated scripts** to efficiently deploy the system.

Leverage **LLMs, open-source libraries, or API documentation** where applicable. Creativity in adding extra features is encouraged.

Tasks & Requirements

1. Backend Development (FastAPI + Database)

Add Doctor to the System

- **POST** `/doctors/`
- **Request Body:** Doctor Name, Specialty, Available Slots per Day
- **Behavior:**
  - Validate input and ensure positive slot count.
  - Store doctor details in the database.
- **Response:** Doctor details with ID and available slots.

Retrieve Doctors

- **GET** `/doctors/` → List all doctors and their availability.
- **GET** `/doctors/{doctor_id}` → Retrieve details of a specific doctor.

Book an Appointment

- **POST** `/doctors/{doctor_id}/appointments/`
- **Request Body:** Patient Name, Appointment Date, Time Slot
- **Behavior:**
  - Check if the selected slot is available.
  - Ensure the booking time is within valid hours (e.g., 9 AM - 5 PM).

- Store appointment details and reduce the available slots.

- **Response:** Appointment details with confirmation message.

Cancel Appointment

- **DELETE** `/appointments/{appointment_id}`
- **Behavior:**
  - Mark the appointment as canceled.
  - Increase the doctor's available slots.
- **Response:** Confirmation message.

OpenAPI Docs

- Ensure **FastAPI** exposes an OpenAPI spec (`http://localhost:8000/openapi.json`).
- Document request/response schemas properly.

Unit Tests

- Tests for:
  - Adding doctors, booking, and canceling appointments.
  - Ensuring valid and invalid time slot handling.
  - Preventing overbooking.

---

2. Frontend Client (ReactJS)

Develop a ReactJS Dashboard that Communicates with FastAPI

- **View Doctors & Availability**
  - Fetch and display doctors (`GET /doctors/`).
  - Show specialties and remaining slots.
- **Book an Appointment**
  - Form to enter patient name, select doctor, and choose a time slot.
  - On submit, call **POST /doctors/{doctor_id}/appointments/**.
- **Cancel Appointment**
  - Input field for appointment ID.
  - On submit, call **DELETE /appointments/{appointment_id}**`.
- **UI/UX Considerations**
  - Display **booking success/failure** messages.
  - If slots are unavailable, disable booking option.

---

3. Python SDK (OpenAPI Generator CLI)

Generate the SDK

- Use the OpenAPI spec (`http://localhost:8000/openapi.json`).
- Example command:

```
openapi-generator-cli generate -i http://localhost:8000/openapi.json -g python -o healthcare_sdk

```

Validate & Use the SDK

- After generation, ensure it supports:

- **add_doctor()** → Add a new doctor.
- **book_appointment()** → Schedule an appointment.
- **cancel_appointment()** → Cancel an appointment.
- **list_doctors() / get_doctor_by_id()** → Retrieve doctor information.

Sample Script for SDK Usage

```
from healthcare_sdk.api.doctors_api import DoctorsApi
from healthcare_sdk import ApiClient

client = ApiClient()
doctors_api = DoctorsApi(client)

# Retrieve all available doctors
doctors = doctors_api.get_doctors()
print(doctors)

```

---

4. Automation Scripts

Setup Script (PowerShell, Bash, etc.)

- **Python Virtual Environment**
  - Create & activate a virtual environment.
- **Install Python Dependencies**

```
pip install -r requirements.txt

```

- **Configure Database**
  - Apply migrations and seed data.
- **Install React Dependencies**

```
npm install

```

Execution Script

- **Start FastAPI Backend**

```
uvicorn main:app --host 0.0.0.0 --port 8000

```

- **Start React Frontend**

```
npm start

```

Completion Criteria

- **Functional System:**
  - Allows users to **add doctors, book, and cancel appointments**.
  - Manages **doctor availability dynamically**.
  - Provides a **ReactJS dashboard** for management.

- **Appointment Scheduling Logic:**
  - Prevents overbooking.
  - Handles **valid working hours** properly.
- **Python SDK:**
  - **Generated via OpenAPI.**
  - **Demonstrated with a sample script.**
- **Automation:**
  - One script to set everything up.
  - One script (or set of commands) to run the system.
- **Testing:**
  - Backend tests covering scheduling flow and validation handling.

Bonus Features (Optional)

- **Real-Time Slot Updates**
  - Implement WebSockets or polling to **update slot availability in real-time**.
- **Email Notifications**
  - Send **confirmation emails** when an appointment is booked.
  - Send reminders before an appointment.
- **User Authentication**
  - Restrict appointment booking to **logged-in users**.
- **Detailed Error Messages**
  - Provide clear feedback for **invalid time slots, overbooking, or missing doctor details**.

Deliverables

- **Backend (FastAPI) source code**
- **ReactJS frontend code**
- **Python SDK (OpenAPI generated)**
- **Setup & Execution Scripts**
- **Unit tests for backend**
- **README with setup instructions**

Good luck!