Hospital Management Usecase – Spring Boot Application

1. **Hospital Management Usecase**

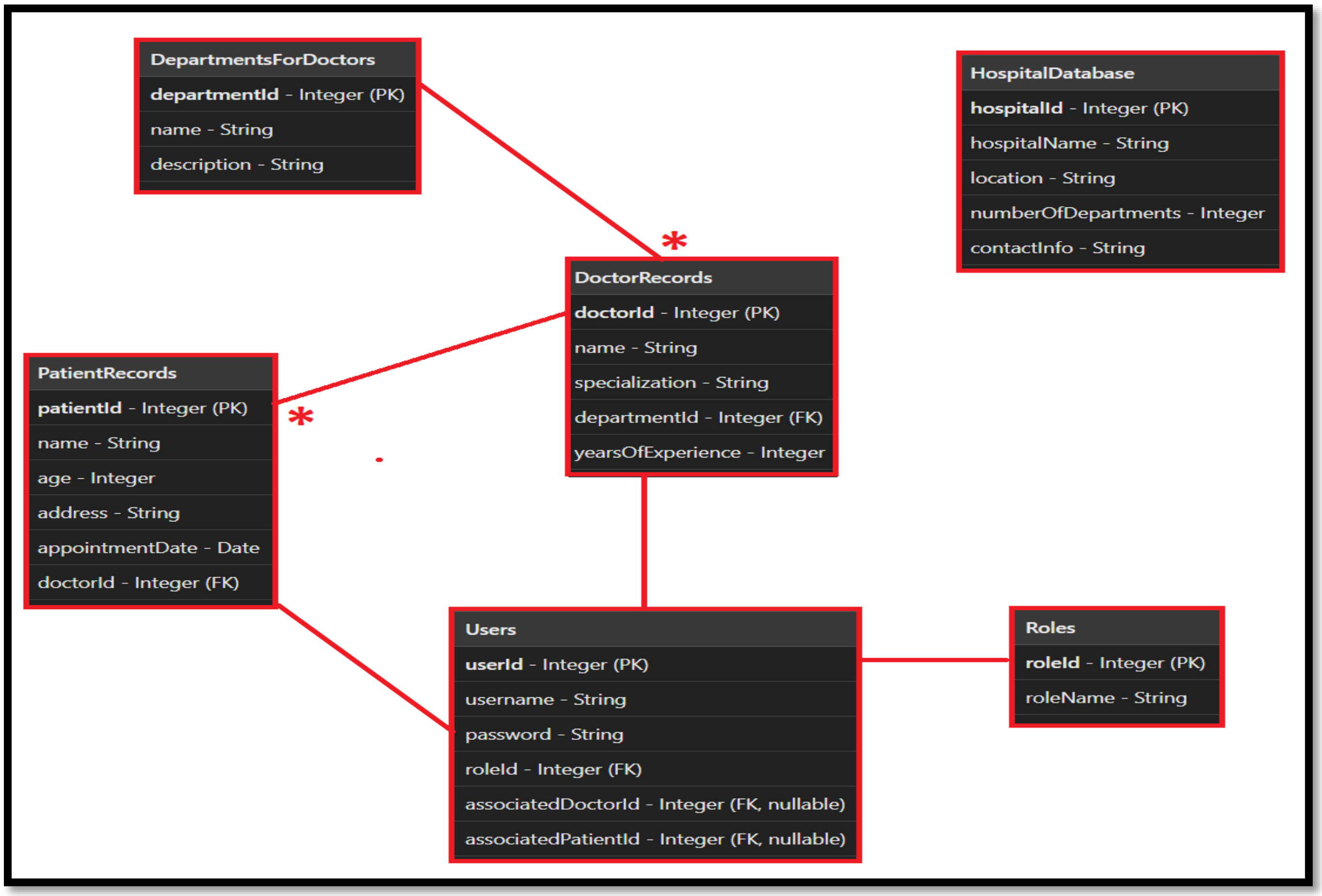
   Question: The hospital management system will allow patients to book appointments with doctors, doctors to update patient diagnoses, and hospital admins to manage department and doctor records. The relationships between tables are designed to capture real-world associations, like doctors working in departments, patients consulting multiple doctors, and so on. As an initial MVP you are required to develop a restful API backend application in springboot.

   Here is the requirement for the application

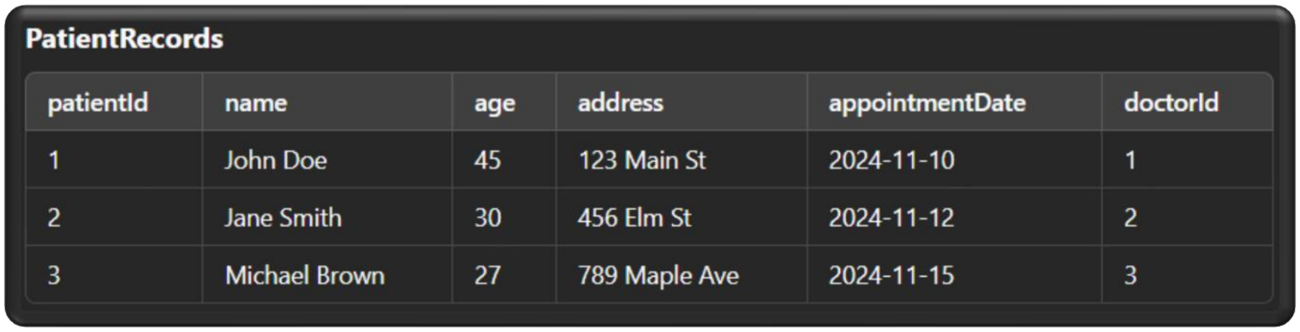
   The hospital management system has the following roles:

   - **Patient**: Can book appointments, view their records.
   - **Doctor**: Can view and update patient records.
   - **Admin**: Manages departments and doctor records.

   A. Database models are created and initialized as below:

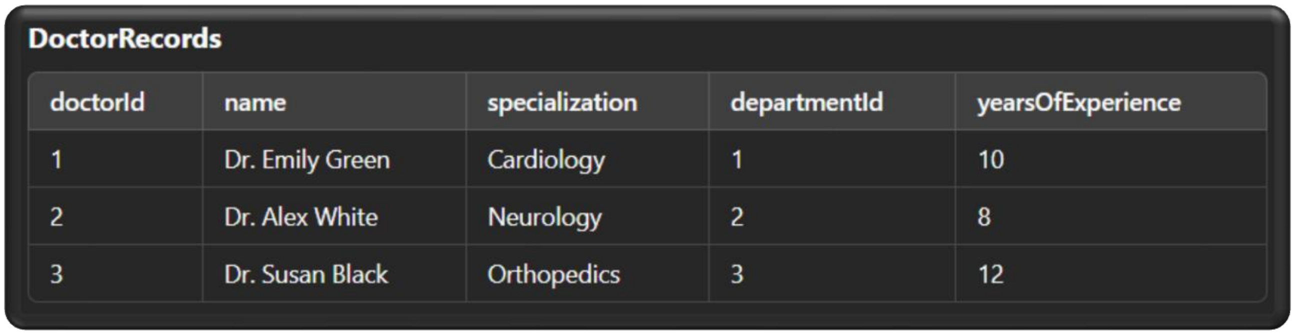


B. Populate the table with below records initially as test data



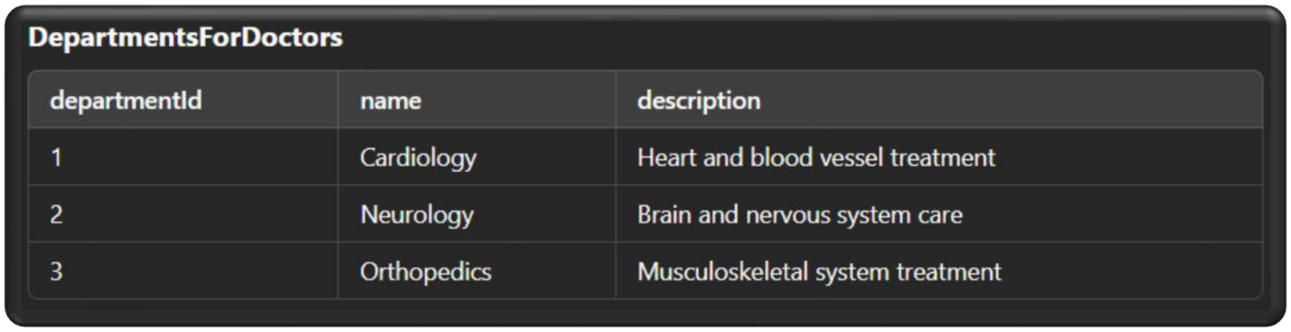
PatientRecords

patientId	name	age	address	appointmentDate	doctorId
1	John Doe	45	123 Main St	2024-11-10	1
2	Jane Smith	30	456 Elm St	2024-11-12	2
3	Michael Brown	27	789 Maple Ave	2024-11-15	3



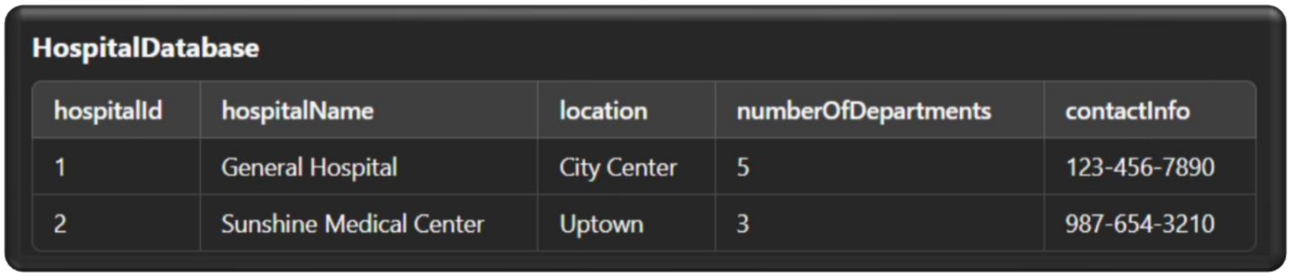
DoctorRecords

doctorId	name	specialization	departmentId	yearsOfExperience
1	Dr. Emily Green	Cardiology	1	10
2	Dr. Alex White	Neurology	2	8
3	Dr. Susan Black	Orthopedics	3	12



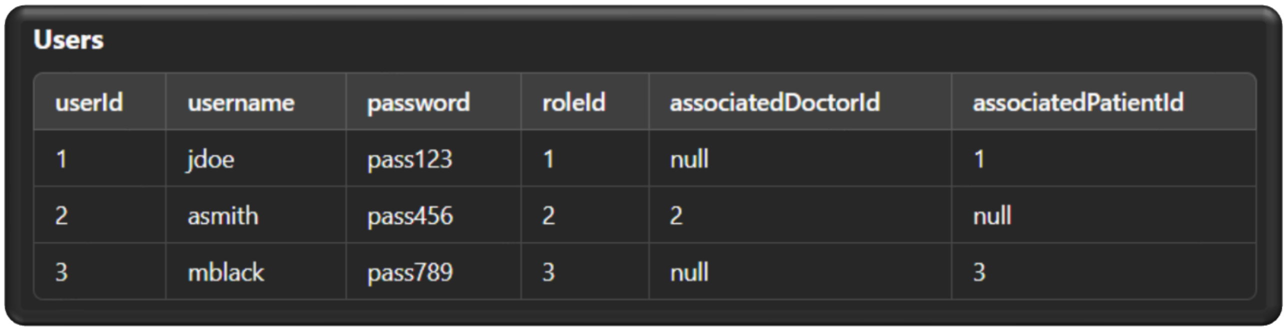
DepartmentsForDoctors

departmentId	name	description
1	Cardiology	Heart and blood vessel treatment
2	Neurology	Brain and nervous system care
3	Orthopedics	Musculoskeletal system treatment



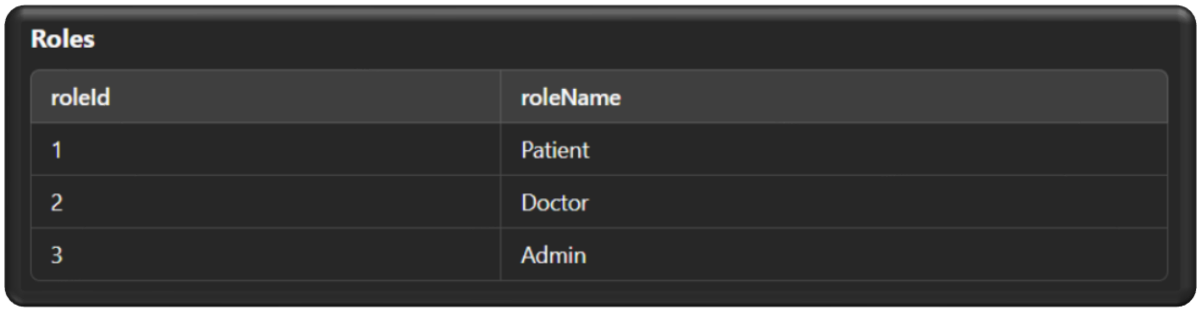
HospitalDatabase

hospitalId	hospitalName	location	numberOfDepartments	contactInfo
1	General Hospital	City Center	5	123-456-7890
2	Sunshine Medical Center	Uptown	3	987-654-3210



Users

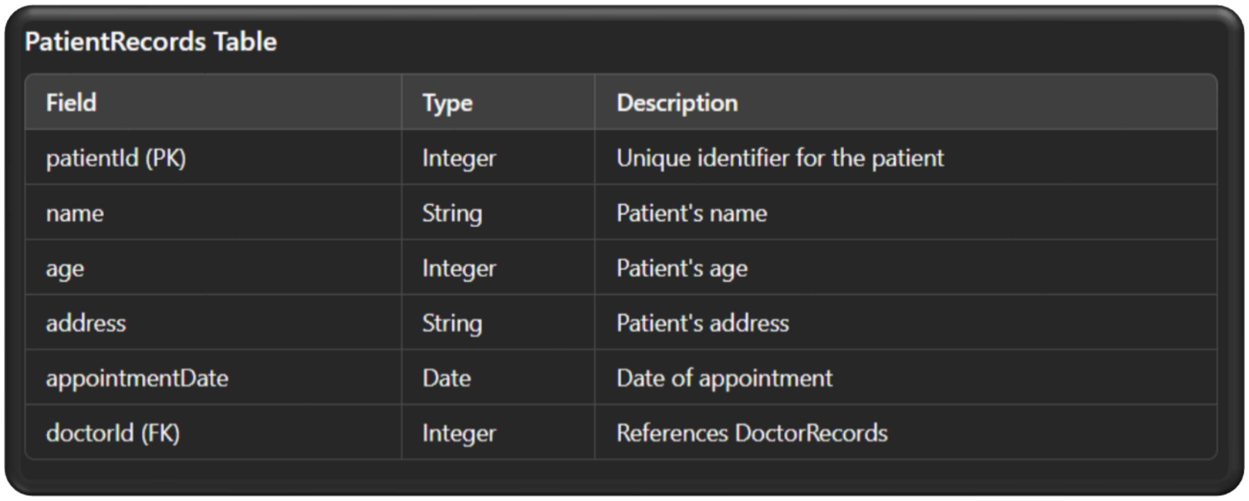
userId	username	password	roleId	associatedDoctorId	associatedPatientId
1	jdoe	pass123	1	null	1
2	asmith	pass456	2	2	null
3	mblack	pass789	3	null	3



Roles

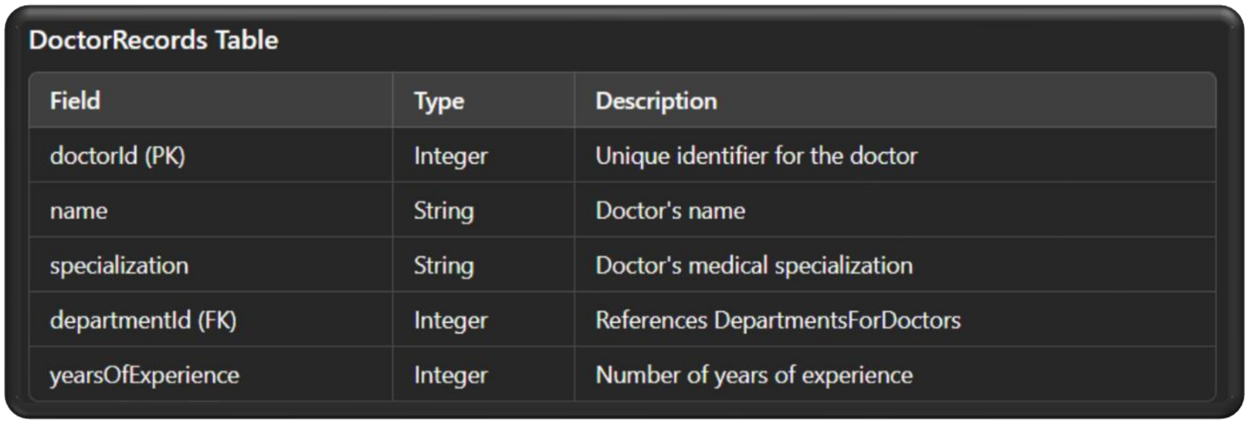
roleId	roleName
1	Patient
2	Doctor
3	Admin

C. Table Structure for the question is as follows:



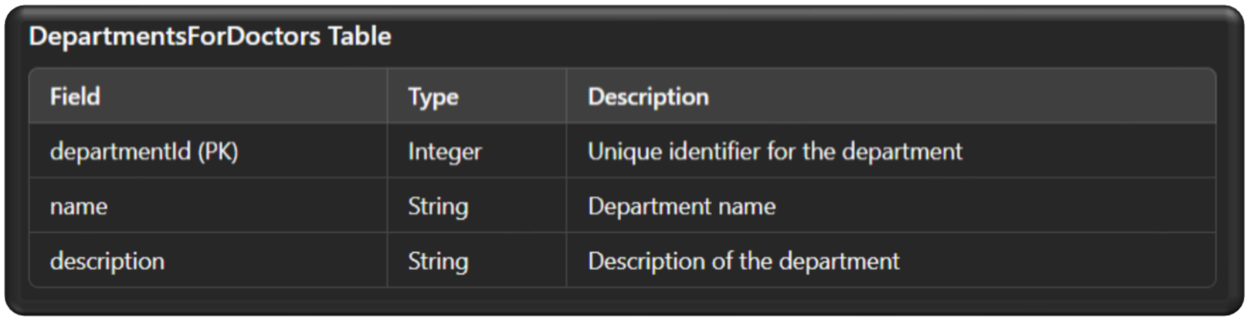
PatientRecords Table

Field	Type	Description
patientId (PK)	Integer	Unique identifier for the patient
name	String	Patient's name
age	Integer	Patient's age
address	String	Patient's address
appointmentDate	Date	Date of appointment
doctorId (FK)	Integer	References DoctorRecords



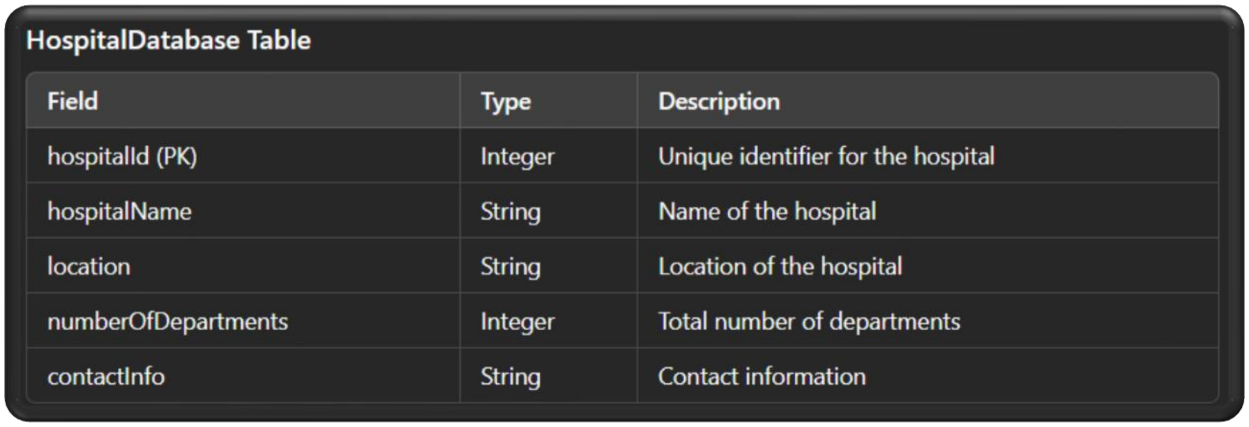
DoctorRecords Table

Field	Type	Description
doctorId (PK)	Integer	Unique identifier for the doctor
name	String	Doctor's name
specialization	String	Doctor's medical specialization
departmentId (FK)	Integer	References DepartmentsForDoctors
yearsOfExperience	Integer	Number of years of experience

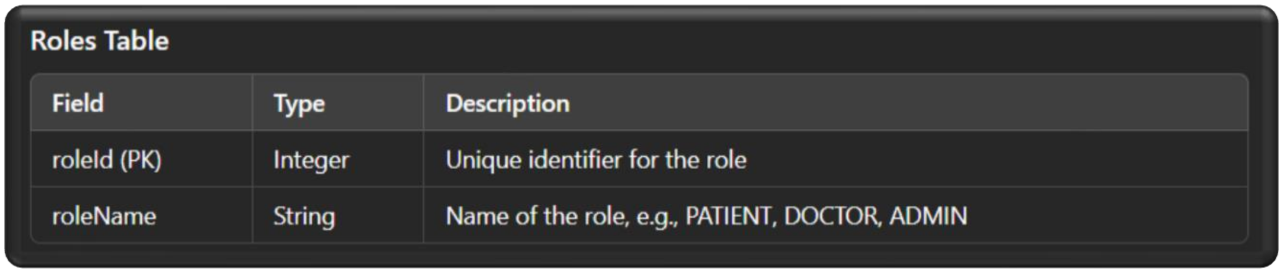


DepartmentsForDoctors Table

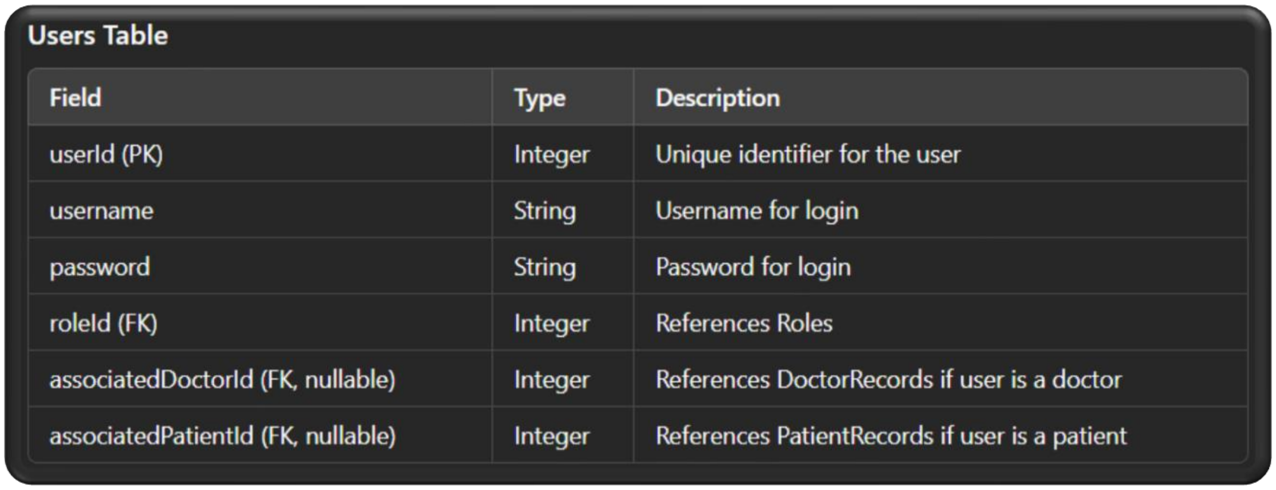
Field	Type	Description
departmentId (PK)	Integer	Unique identifier for the department
name	String	Department name
description	String	Description of the department


HospitalDatabase Table

Field	Type	Description
hospitalId (PK)	Integer	Unique identifier for the hospital
hospitalName	String	Name of the hospital
location	String	Location of the hospital
numberOfDepartments	Integer	Total number of departments
contactInfo	String	Contact information


Roles Table

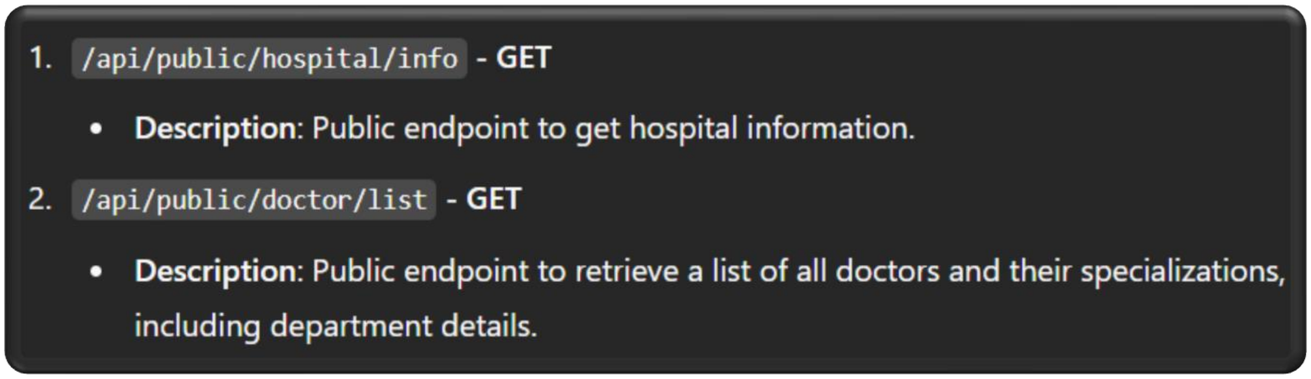
Field	Type	Description
roleId (PK)	Integer	Unique identifier for the role
roleName	String	Name of the role, e.g., PATIENT, DOCTOR, ADMIN


Users Table

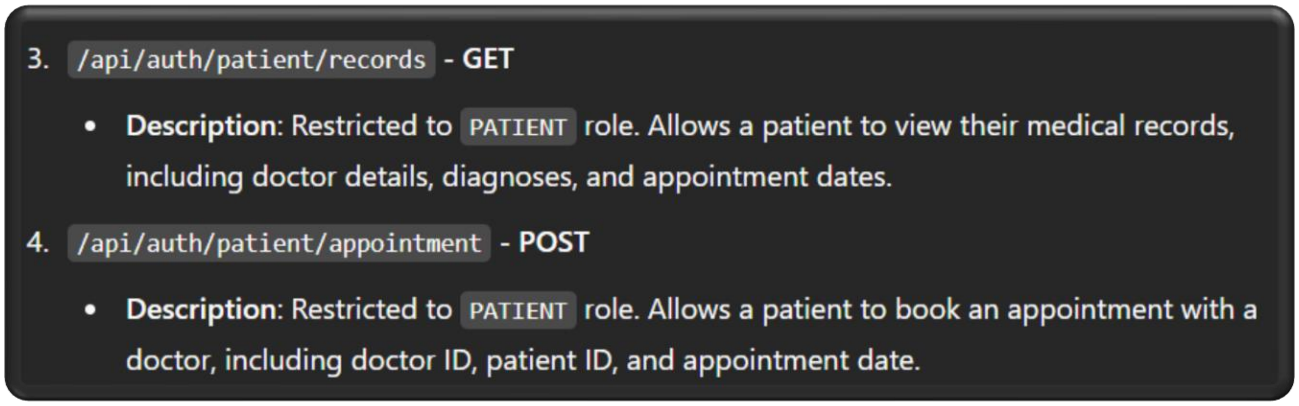
Field	Type	Description
userId (PK)	Integer	Unique identifier for the user
username	String	Username for login
password	String	Password for login
roleId (FK)	Integer	References Roles
associatedDoctorId (FK, nullable)	Integer	References DoctorRecords if user is a doctor
associatedPatientId (FK, nullable)	Integer	References PatientRecords if user is a patient

D. Endpoint details are as shown below

Public Endpoints – Accessible by All

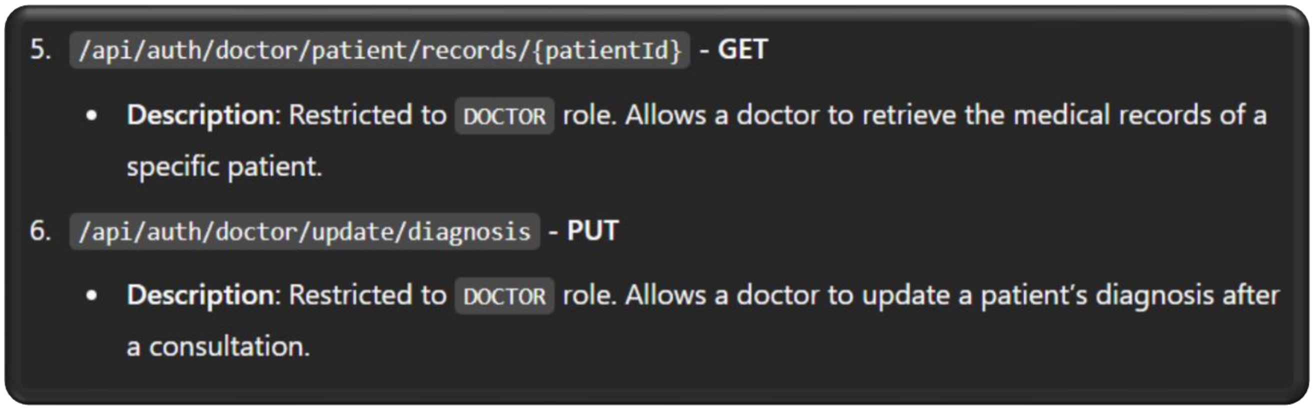

1. `/api/public/hospital/info` - **GET**
   - **Description**: Public endpoint to get hospital information.
2. `/api/public/doctor/list` - **GET**
   - **Description**: Public endpoint to retrieve a list of all doctors and their specializations, including department details.

Patient Endpoints – Accessible only by Patients



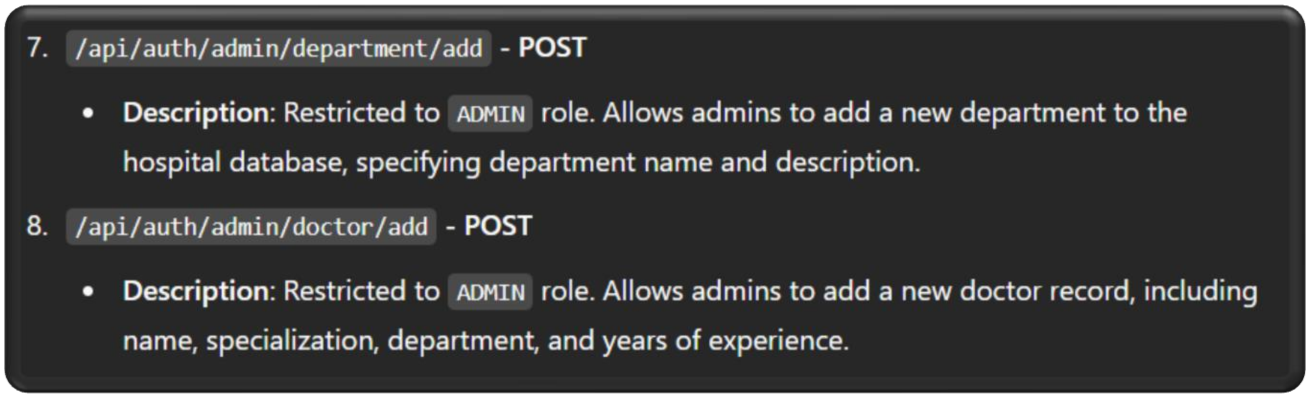
3. `/api/auth/patient/records` - **GET**
   - **Description**: Restricted to `PATIENT` role. Allows a patient to view their medical records, including doctor details, diagnoses, and appointment dates.
4. `/api/auth/patient/appointment` - **POST**
   - **Description**: Restricted to `PATIENT` role. Allows a patient to book an appointment with a doctor, including doctor ID, patient ID, and appointment date.

Doctor Endpoints – Accessible only by Doctors



5. `/api/auth/doctor/patient/records/{patientId}` - **GET**
   - **Description**: Restricted to `DOCTOR` role. Allows a doctor to retrieve the medical records of a specific patient.
6. `/api/auth/doctor/update/diagnosis` - **PUT**
   - **Description**: Restricted to `DOCTOR` role. Allows a doctor to update a patient's diagnosis after a consultation.

Admin Endpoints – Access only by Admins



7. `/api/auth/admin/department/add` - **POST**
   - **Description**: Restricted to `ADMIN` role. Allows admins to add a new department to the hospital database, specifying department name and description.
8. `/api/auth/admin/doctor/add` - **POST**
   - **Description**: Restricted to `ADMIN` role. Allows admins to add a new doctor record, including name, specialization, department, and years of experience.