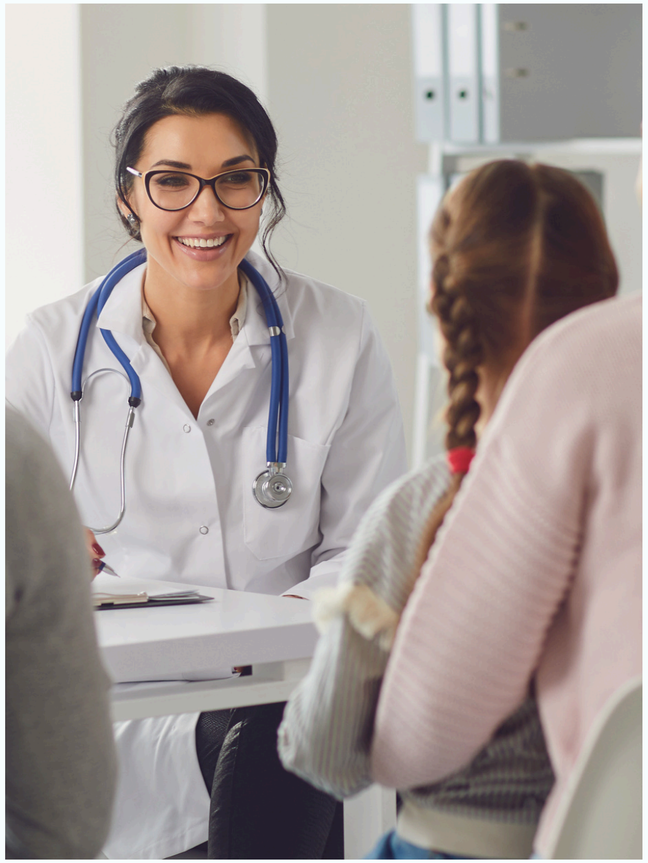
CA-3

HOSPITAL MANAGEMENT SYSTEM

OUR TEAM

- AJAY
- DINESH
- GAUTAM
- VISHAL
- HARSHIT
- SUDHEER



SPRINGBOOT

ABOUT PROJECT

We, as a group, are tasked with developing a RESTful API for a Hospital Management System using Spring Boot. This API will manage operations related to patients, doctors, appointments, departments, and staff. Together, we will follow specific steps to complete the project effectively.

---

www.reallygreatsite.com

ENTITIES AND ATTRIBUTES

- 1. Patient: Attributes like id, name, age, gender, address, contact.
- 2. Doctor: Attributes like id, name, specialization, experience, contact.
- 3. Appointment: Attributes like id, appointmentDate, status, patientId, doctorId.
- 4. Department: Attributes like id, name, headOfDepartment.
- 5. Staff: Attributes like id, name, role, departmentId.

1. Patient
   - ID: Unique identifier for the patient
   - Name: Full name of the patient
   - Age: Patient's age
   - Gender: Patient's gender
   - Address: Residential address of the patient
   - Contact: Contact details of the patient

ENTITY RELATIONSHIPS

SPRINGBOOT

1. Appointment and Patient
   - Relationship: Many-to-One
   - Description:
   - A single patient can have multiple appointments, but each appointment is associated with only one patient.

. Appointment and Doctor
- Relationship: Many-to-One
- Description:
- A single doctor can have multiple appointments, but each appointment is associated with only one doctor.

Department and Staff
- Relationship: One-to-Many
- Description:
- A single department can have multiple staff members, but each staff member belongs to only one department.

FIELDS OF ENTITIES

1. Patient
   - id
   - name
   - age
   - gender
   - address
   - contact

2.Doctor
   - id
   - name
   - specialization
   - experience
   - contact

3. Appointment
   - id
   - appointmentDate
   - status
   - patientId
   - doctorId

. 4.Department
   - id
   - name
   - headOfDepartment

5. Staff
   - id
   - name
   - role
   - departmentId

RESTful

PATIENT RELATED ENDPOINTS

Patient-related endpoints:

- Retrieve all patients: GET /api/patients
- Get a patient by ID: GET /api/patients/{id}
- Add a new patient: POST /api/patients
- Update a patient: PUT /api/patients/{id}
- Delete a patient: DELETE /api/patients/{id}

RESTful

DOCTOR RELATED ENDPOINTS

Doctor-related endpoints:

- Retrieve all doctors: GET /api/doctors
- Get a doctor by ID: GET /api/doctors/{id}
- Add a new doctor: POST /api/doctors
- Update a doctor: PUT /api/doctors/{id}
- Delete a doctor: DELETE /api/doctors/{id}

RESTful

APPOINTMENT RELATED ENDPOINTS

Appointment-related endpoints:

- Retrieve all appointments: GET /api/appointments
- Get an appointment by ID: GET /api/appointments/{id}
- Add a new appointment: POST /api/appointments
- Update an appointment: PUT /api/appointments/{id}
- Cancel an appointment: DELETE /api/appointments/{id}

RESTful

DEPARTMENT RELATED ENDPOINTS

Department-related endpoints:

- Retrieve all departments: GET /api/departments
- Get a department by ID: GET /api/departments/{id}
- Add a new department: POST /api/departments
- Update a department: PUT /api/departments/{id}
- Delete a department: DELETE /api/departments/{id}

RESTful

STAFF RELATED ENDPOINTS

Staff-related endpoints:

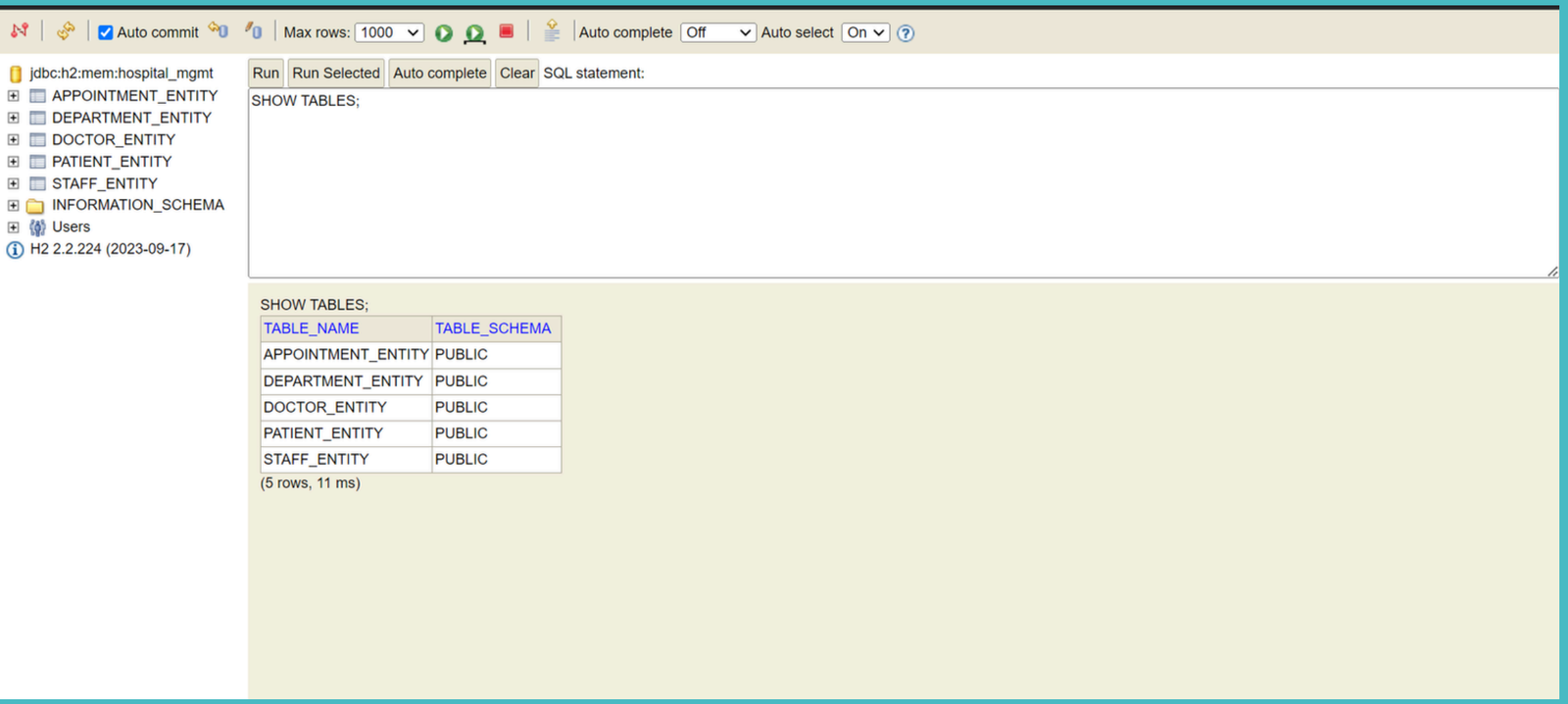
- Retrieve all staff members: GET /api/staff
- Get a staff member by ID: GET /api/staff/{id}
- Add a new staff member: POST /api/staff
- Update staff details: PUT /api/staff/{id}
- Delete a staff member: DELETE /api/staff/{id}

DEPENDENCIES

- MYSQL CONNECTOR: DATABASE
- SPRING BOOT STARTER DATA JPA: ORM
- SPRING BOOT STARTER WEB: WEB
- SPRING BOOT DEVTOOLS: DEVELOPMENT
- H2 DATABASE: LIGHTWEIGHT DATABASE
- SPRING BOOT STARTER TEST: TESTING

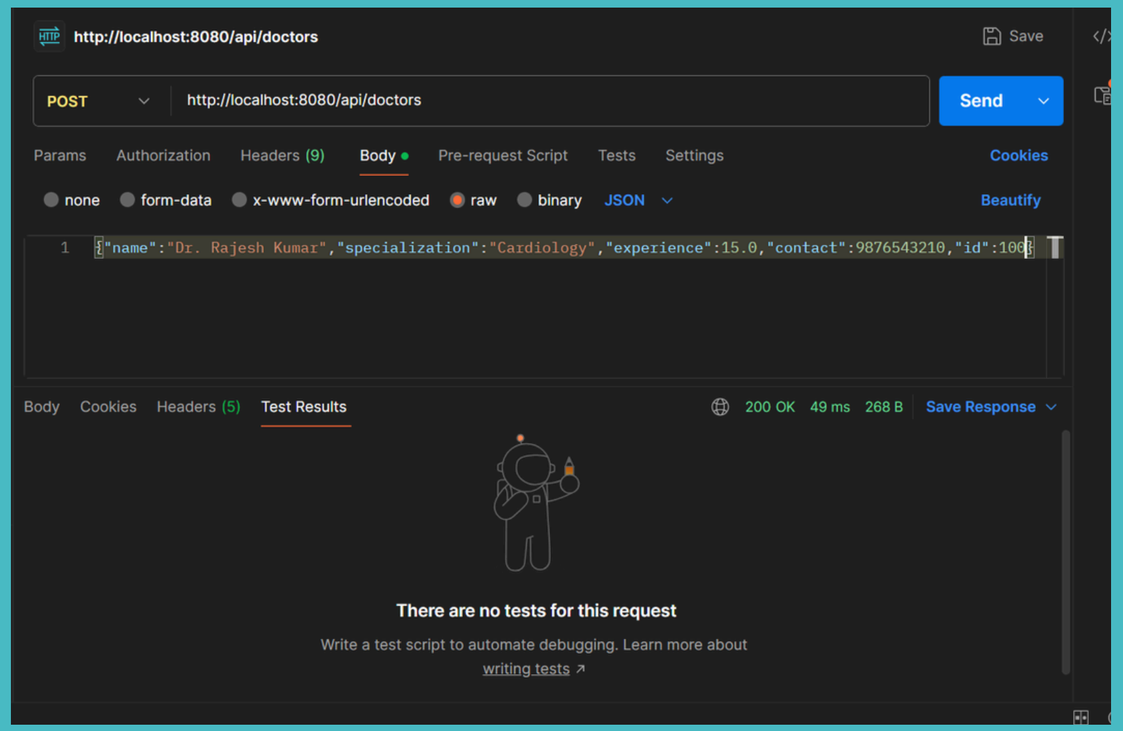
- MYSQL CONNECTOR (MYSQL-CONNECTOR-JAVA)
- SPRING BOOT STARTER DATA JPA (SPRING-BOOT-STARTER-DATA-JPA)
- SPRING BOOT STARTER WEB (SPRING-BOOT-STARTER-WEB)
- SPRING BOOT DEVTOOLS (SPRING-BOOT-DEVTOOLS)
- H2 DATABASE (H2)
- SPRING BOOT STARTER TEST (SPRING-BOOT-STARTER-TEST)

DATABASE



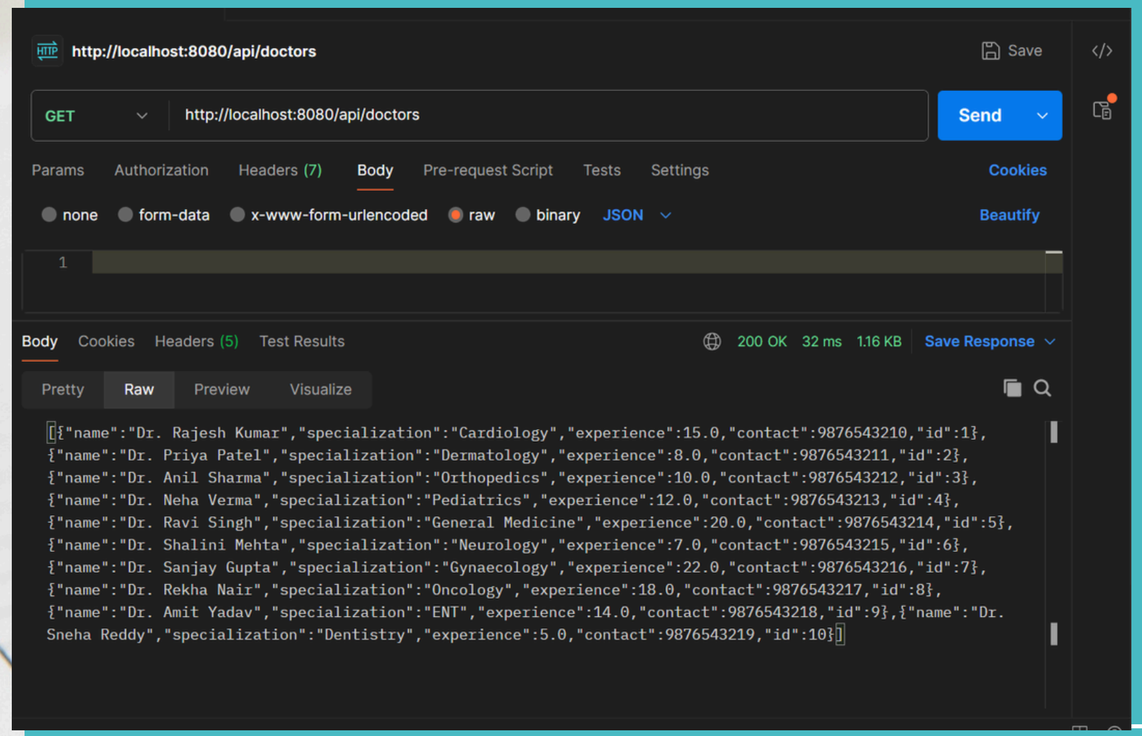
The H2 database contains tables such as APPOINTMENT_ENTITY, DOCTOR_ENTITY, PATIENT_ENTITY, STAFF_ENTITY, and DEPARTMENT_ENTITY, each with specific fields like IDs, names, and relational attributes. These tables are interconnected using foreign key constraints, defining relationships such as appointments between doctors and patients, and staff assigned to departments.

CREATE OPERATION



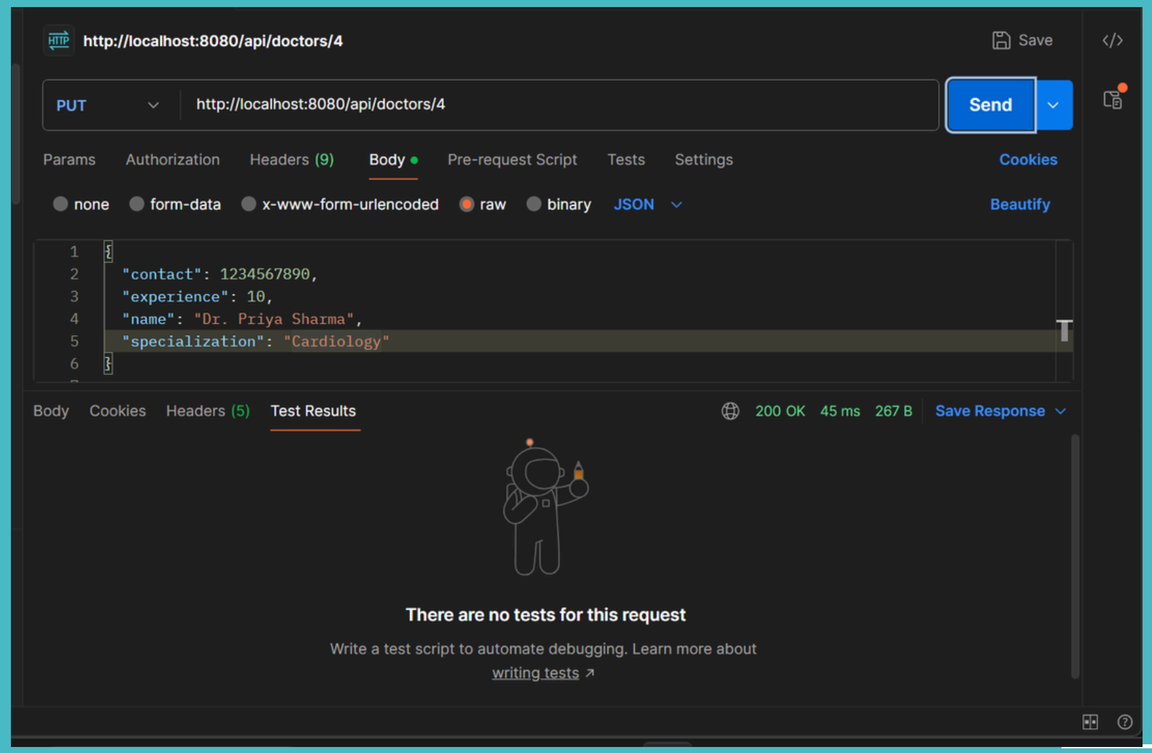
The Create Doctor API (POST /api/doctors) adds a new doctor by accepting a JSON payload with the doctor's details and returns the created doctor's information.

READ OPERATION



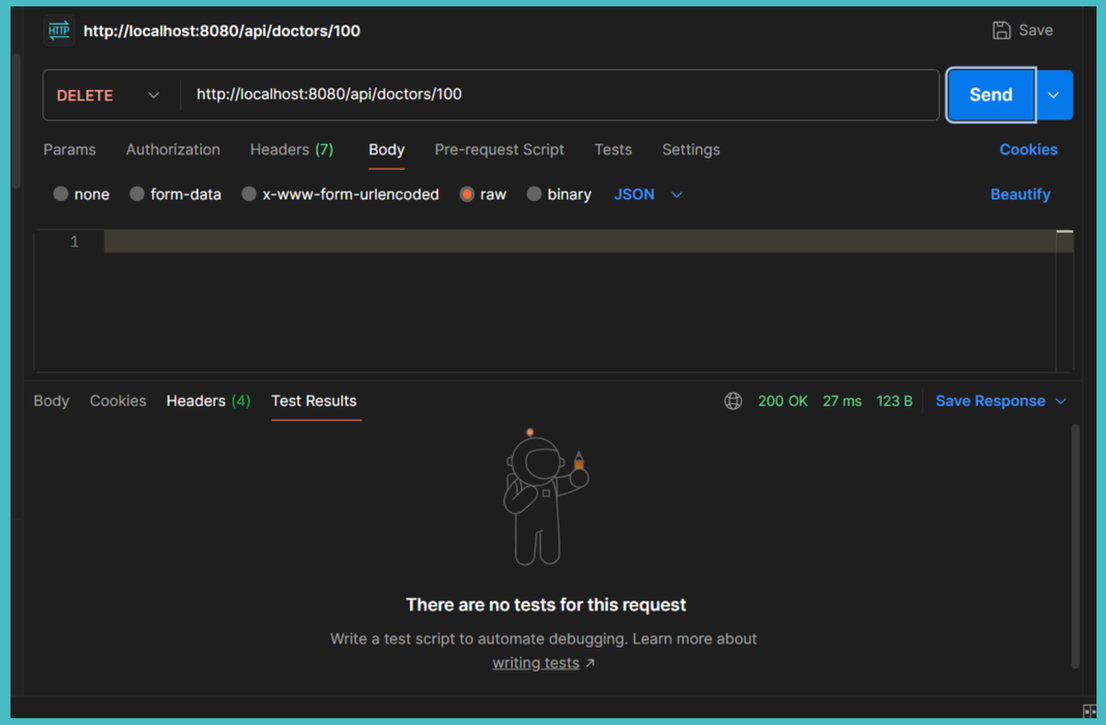
We as a group tested the GET method of the /api/doctors API in Postman by sending a request to http://localhost:8080/api/doctors and verified the response data.

UPDATE OPERATION



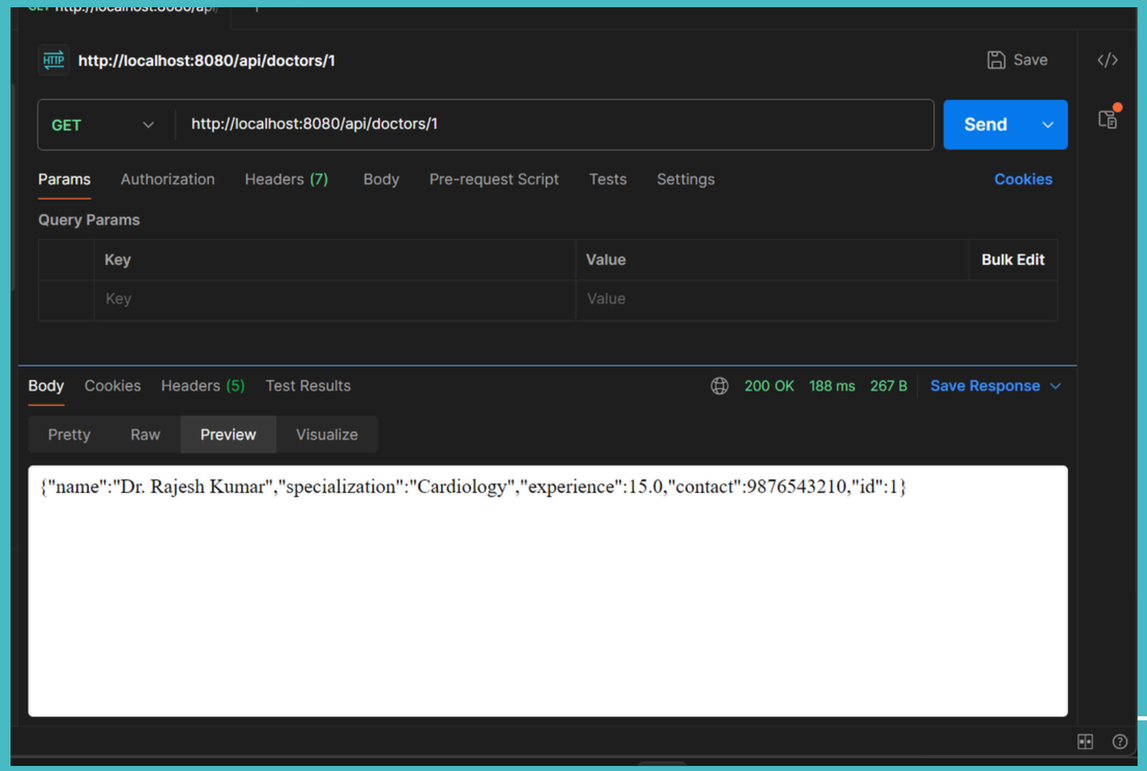
The Update Doctor API (PUT /api/doctors/{id}) updates an existing doctor's details by their ID using the provided data in the request body.

DELETE OPERATION



The Delete Doctor API (DELETE /api/doctors/{id}) removes a doctor's record from the database based on their unique ID.

SERACH WITH ID



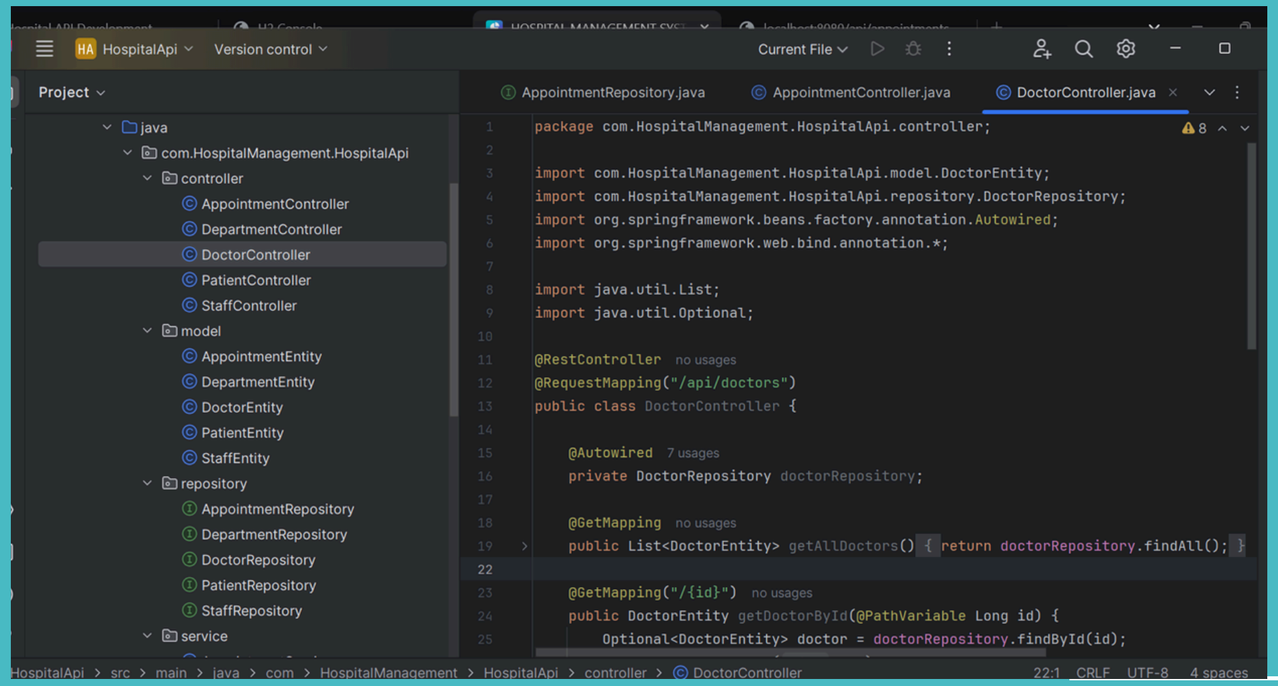
The Get Doctor by ID API (GET /api/doctors/{id}) retrieves the details of a doctor from the database based on their unique ID.

CONTINUATION

We tested the Patient, Appointment, Department, and Staff APIs using Postman, similar to how we tested the Doctor APIs. We performed GET, POST, PUT, and DELETE requests to verify functionality for retrieving, adding, updating, and deleting records. Each entity's endpoints were tested with appropriate data and verified for correct responses. The API behavior was consistent with the expected outcomes for all entities.

SPRINGBOOT

THANK YOU


HospitalApi
Version control
Current File
Project
AppointmentRepository.java
AppointmentController.java
DoctorController.java
java
com.HospitalManagement.HospitalApi
controller
AppointmentController
DepartmentController
DoctorController
PatientController
StaffController
model
AppointmentEntity
DepartmentEntity
DoctorEntity
PatientEntity
StaffEntity
repository
AppointmentRepository
DepartmentRepository
DoctorRepository
PatientRepository
StaffRepository
service
package com.HospitalManagement.HospitalApi.controller;
import com.HospitalManagement.HospitalApi.model.DoctorEntity;
import com.HospitalManagement.HospitalApi.repository.DoctorRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
@RestController no usages
@RequestMapping("/api/doctors")
public class DoctorController {
@Autowired 7 usages
private DoctorRepository doctorRepository;
@GetMapping no usages
public List<DoctorEntity> getAllDoctors() { return doctorRepository.findAll(); }
@GetMapping("/{id}") no usages
public DoctorEntity getDoctorById(@PathVariable Long id) {
Optional<DoctorEntity> doctor = doctorRepository.findById(id);
HospitalApi › src › main › java › com › HospitalManagement › HospitalApi › controller › DoctorController
22:1 CRLF UTF-8 4 spaces