Hospital Management System

Java Coding Challenge 3

Name: Shardul Satish Kulkarni

SupersetID: 5270707

To view the Code:

https://github.com/shardulkulk297/HospitalManagement

Created Directory Structure

1. Entity: For entity classes
2. Dao: For service interfaces and classes
3. Util: For database Connection class.
4. Main: Main module of the app
5. Exception: For Custom Exceptions

Task 1: Creating SQL Schema

1. Created Database HospitalManagement
2. Created 3 tables as mentioned with the given attributes along with Foreign key and primary key constraints

```
-- Active: 1742545260664@@127.0.0.1@3306@hospitalmanagement
Create Database HospitalManagement;
USE HospitalManagement;

Create table Patient(
    patientId INT PRIMARY KEY AUTO_INCREMENT,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
```

```
    dateOfBirth DATE NOT NULL,
    gender VARCHAR(10) NOT NULL,
    contactNumber VARCHAR(15) NOT NULL,
    address VARCHAR(255) NOT NULL
);

Create Table Doctor(
    doctorId INT PRIMARY KEY AUTO_INCREMENT,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    specialization VARCHAR(50) NOT NULL,
    contactNumber VARCHAR(15) NOT NULL
);

Create Table Appointment(
    appointmentId INT PRIMARY KEY AUTO_INCREMENT,
    patientId INT NOT NULL,
    doctorId INT NOT NULL,
    appointmentDate DATE NOT NULL,
    description VARCHAR(100) NOT NULL,
    FOREIGN KEY (patientId) REFERENCES Patient(patientId),
    FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
);
```

3. Inserted Mock Data with following records:

```
INSERT INTO patient (firstName, lastName, dateOfBirth, gender, contactNumb
VALUES
('John', 'Doe', '1990-05-12', 'Male', '9876543210', '123 Main Street'),
('Jane', 'Smith', '1985-08-23', 'Female', '9123456789', '456 Oak Avenue'),
('Michael', 'Johnson', '1992-01-15', 'Male', '9988776655', '789 Pine Road'),
('Emily', 'Clark', '1998-07-19', 'Female', '9898989898', '321 Maple Lane'),
('David', 'Brown', '1980-03-05', 'Male', '9112233445', '555 Cedar Blvd');

INSERT INTO doctor (firstName, lastName, specialization, contactNumber)
VALUES
('Sarah', 'Miller', 'Cardiology', '9999999999'),
```

```
('Robert', 'Williams', 'Orthopedics', '8888888888'),
('Anna', 'Davis', 'Pediatrics', '7777777777'),
('James', 'Garcia', 'Dermatology', '6666666666'),
('Laura', 'Martinez', 'Dentistry', '5555555555');

INSERT INTO appointment (patientId, doctorId, appointmentDate, description)
VALUES
(1, 1, '2025-04-10', 'Regular checkup'),
(2, 3, '2025-04-12', 'Child vaccination'),
(3, 2, '2025-04-15', 'Knee pain consultation'),
(4, 5, '2025-04-18', 'Tooth filling'),
(5, 4, '2025-04-20', 'Skin rash treatment');
```

Task 2: Creating Entity Classes:

1. Entity Class: Patient
    a. Created Class Patient with Default constructor, parameterized constructor and getter, setter methods:
    b. Overridden toString() to print the details of Patient class.
    c. Also defined display_patient_details method.

```
package app.entity;

import java.util.Date;

public class Patient {
    private int patientId;
    private String firstName;
    private String lastName;
    private Date dateOfBirth;
    private String gender;
    private String contactNumber;
    private String address;
```

```
public Patient(){

}

public Patient(int patientId, String firstName, String lastName, Date dateOfB
    this.patientId = patientId;
    this.firstName = firstName;
    this.lastName = lastName;
    this.dateOfBirth = dateOfBirth;
    this.gender = gender;
    this.contactNumber = contactNumber;
    this.address = address;
}

//Getter methods
public int getPatientId() {
    return patientId;
}

public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}

public Date getDateOfBirth() {
    return dateOfBirth;
}


public String getContactNumber() {
    return contactNumber;
}

public String getAddress() {
    return address;
```

```
}

public String getGender() {
    return gender;
}

//setter methods

public void setPatientId(int patientId) {
    this.patientId = patientId;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public void setAddress(String address) {
    this.address = address;
}

public void setContactNumber(String contactNumber) {
    this.contactNumber = contactNumber;
}

public void setDateOfBirth(Date dateOfBirth) {
    this.dateOfBirth = dateOfBirth;
}

public void setGender(String gender) {
    this.gender = gender;
}

//Overriding toString() method
@Override
```

```
    public String toString() {
        return "Patient { " +
                "Patient ID: " + patientId +
                ", First Name: '" + firstName + '\'' +
                ", Last Name: '" + lastName + '\'' +
                ", Date of Birth: " + dateOfBirth +
                ", Gender: '" + gender + '\'' +
                ", Contact Number: '" + contactNumber + '\'' +
                ", Address: ' " + address + '\'' +
                " }";
    }

    public void display_patient_details(){
        System.out.println("PatientId: " + patientId);
        System.out.println("FirstName: " + firstName);
        System.out.println("LastName: " + lastName);
        System.out.println("DateOfBirth: " + dateOfBirth);
        System.out.println("Gender: " + gender);
        System.out.println("ContactNumber: " + contactNumber);
        System.out.println("Address: " + address);
    }
}
```

2. Entity Class: Doctor
    a. Created Doctor class with appropriate constructors and methods.
    b. Overridden the toString method and defined display method

```
package app.entity;

public class Doctor {
    private int doctorId;
    private String firstName;
    private String lastName;
    private String specialization;
    private String contactNumber;
```

```
public Doctor(){

}

public Doctor(int doctorId, String firstName, String lastName,
            String specialization, String contactNumber)
{
   this.doctorId = doctorId;
   this.firstName = firstName;
   this.lastName = lastName;
   this.specialization = specialization;
   this.contactNumber = contactNumber;
}

//setter methods


public void setDoctorId(int doctorId) {
   this.doctorId = doctorId;
}

public void setFirstName(String firstName) {
   this.firstName = firstName;
}

public void setLastName(String lastName) {
   this.lastName = lastName;
}

public void setSpecialization(String specialization) {
   this.specialization = specialization;
}

public void setContactNumber(String contactNumber) {
   this.contactNumber = contactNumber;
}

//Getter methods
```

```
public int getDoctorId() {
    return doctorId;
}

public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}

public String getSpecialization() {
    return specialization;
}

public String getContactNumber() {
    return contactNumber;
}

@Override
public String toString() {
    return "Doctor { " +
            "Doctor ID: " + doctorId +
            ", First Name: '" + firstName + '\'' +
            ", Last Name: '" + lastName + '\'' +
            ", Specialization: '" + specialization + '\'' +
            ", Contact Number: '" + contactNumber + '\'' +
            " }";
}

public void display_doctor_details(){
    System.out.println("DoctorID: " + doctorId);
    System.out.println("FirstName: " + firstName);
    System.out.println("LastName: " + lastName);
    System.out.println("Specialization: " + specialization);
    System.out.println("ContactNumber: "  +contactNumber);
```

```
    }
}
```

3. Entity Class Appointment:
   a. Created Appointment class with appropriate constructors and methods
   b. Defined toString and display methods

```
package app.entity;

import java.util.Date;

public class Appointment {
    private int appointmentId;
    private int patientId;
    private int doctorId;
    private Date appointmentDate;
    private String description;

    public Appointment(){

    }

    public Appointment(int appointmentId, int patientId, int doctorId, Date ap
        this.appointmentId = appointmentId;
        this.patientId = patientId;
        this.doctorId = doctorId;
        this.appointmentDate = appointmentDate;
        this.description = description;
    }

    public Appointment(int patientId, int doctorId, Date appointmentDate, St
        this.patientId = patientId;
        this.doctorId = doctorId;
        this.appointmentDate = appointmentDate;
        this.description = description;
    }
```

```
//setter methods


public void setAppointmentId(int appointmentId) {
   this.appointmentId = appointmentId;
}

public void setPatientId(int patientId) {
   this.patientId = patientId;
}

public void setDoctorId(int doctorId) {
   this.doctorId = doctorId;
}

public void setAppointmentDate(Date appointmentDate) {
   this.appointmentDate = appointmentDate;
}

public void setDescription(String description) {
   this.description = description;
}

//Getter methods


public int getAppointmentId() {
   return appointmentId;
}

public int getPatientId() {
   return patientId;
}

public int getDoctorId() {
   return doctorId;
}
```

```
    public Date getAppointmentDate() {
        return appointmentDate;
    }

    public String getDescription() {
        return description;
    }

    @Override
    public String toString() {
        return "Appointment { " +
                "Appointment ID: " + appointmentId +
                ", Patient ID: " + patientId +
                ", Doctor ID: " + doctorId +
                ", Appointment Date: " + appointmentDate +
                ", Description: '" + description + '\'' +
                " }";
    }

    public void display_appointment_details(){
        System.out.println("AppointmentId: " + appointmentId);
        System.out.println("PatientId: "+patientId);
        System.out.println("DoctorId: " + doctorId);
        System.out.println("AppointmentDate: " + appointmentDate);
        System.out.println("Description: " + description);
    }
}
```

Task 2: Created IHospitalService Interface:

```
package app.dao;

import app.entity.Appointment;
import app.exception.PatientNumberNotFoundException;
```

```
import java.util.List;

public interface IHospitalService {
    public Appointment getAppointmentById(int appointmentId);
    public List<Appointment> getAppointmentsForPatient(int patientId) throws I
    public List<Appointment> getAppointmentsForDoctors(int doctorId);
    public boolean scheduleAppointment(Appointment appointment);
    public boolean updateAppointment(Appointment appointment);
    public boolean cancelAppointment(int appointmentId);
}
```

Task 3: Created Database Connection:

1. Before implementing all the classes, in the Util package creating DBConnection class which returns getConnection() method of Connection interface.
2. To pass in the Connection String created db.properties and defined following attributes:

```
hostname = localhost
dbname = Hospitalmanagement
username = root
password = Shardul@297
port = 3306
```

3. Created a PropertyUtil class which takes in these properties and has a method named getPropertyString() which returns Connection String

```
package app.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
```

```
public class PropertyUtil {

   public static String getPropertyString(){

      Properties prop = new Properties();

      try{
         FileInputStream fs = new FileInputStream("db.properties");
         prop.load(fs);

      }
      catch(IOException e)
      {
         e.printStackTrace();
      }

      String hostname = prop.getProperty("hostname");
      String dbname     = prop.getProperty("dbname");
      String username   = prop.getProperty("username");
      String password   = prop.getProperty("password");
      String port       = prop.getProperty("port");

      String connectionString = "jdbc:mysql://" + hostname + ":" + port + "/" +
            "?user=" + username + "&password=" + password;

      return connectionString;

   }
}
```

4. Following is the DBConnection class:

```
package app.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```
public class DBConnection {

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(PropertyUtil.getPropertyString());

    }
}
```

Task 4: Creating custom Exception

1. Created custom Exception `PatientNumberNotFound` under exception package and used it whenever there is invalid input for Patient Id

```
package app.exception;

public class PatientNumberNotFoundException extends Exception {
    public PatientNumberNotFoundException(String message){
        super(message);
    }
}
```

Task 5 : Created HospitalServiceimpl class

1. Implemented all the methods from IHospitalService interface in this class.
2. Defined Conenction object in Constructor so that we don’t have to redefine it again and again.
3. Also defined Appointment reference variable as we will use that throught this class.
4. For each method here’s what I have done in steps
    a. First I checked for edge cases, I checked whether the given patientId or doctorId or the appointment object passed in as a parameter is 0 or null, if it is 0 or null then I have written a error statement and returned from that part itself.

b. Then I have written a try catch block as statements used can throw SQLException.

c. In try block I have first written the SQL Query then created a PreparedStatement to execute that query.

d. If the results are found then it sets the values and returns what is necessary else it shows errors.

e. For method getAppointmentsForPatient(int patientId) I have thrown the custom Exception which is mentioned in the assignment as per the conditions as shown in the assignment.

```
package app.dao;

import app.entity.Appointment;
import app.exception.PatientNumberNotFoundException;
import app.util.DBConnection;
import com.mysql.cj.x.protobuf.MysqlxPrepare;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class HospitalServiceImpl implements IHospitalService {
    private Connection con;
    private Appointment apo = null;

    public HospitalServiceImpl(){
        try{
            con = DBConnection.getConnection();
        }
        catch(SQLException e){
            e.printStackTrace();
        }

    }
```

```
@Override
public Appointment getAppointmentById(int appointmentId){



    if(appointmentId == 0 || appointmentId < 0)
    {
        System.out.println("Appointment ID cannot be 0 or less than 0");
    }

    try{

        String sql = "Select * from appointment WHERE appointmentId = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, appointmentId);

        ResultSet rs = stmt.executeQuery();

        if(rs.next()){
            apo = new Appointment();
            apo.setAppointmentId(appointmentId);
            apo.setAppointmentDate(rs.getDate("appointmentDate"));
            apo.setPatientId(rs.getInt("patientId"));
            apo.setDescription(rs.getString("description"));
            apo.setDoctorId(rs.getInt("doctorId"));
        }

        else{
            System.out.println("NO Appointments FOUND");
        }



    }
    catch(SQLException e)
    {
        e.printStackTrace();
    }
```

```
    finally {
        try{
            con.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }

    }

    return apo;

}

@Override
public List<Appointment> getAppointmentsForPatient(int patientId) throws I


    List<Appointment> appointments = new ArrayList<>();

    if(patientId == 0 || patientId < 0){
        System.out.println("Patient ID can't be 0");
        throw new PatientNumberNotFoundException("Patient ID can't be 0");

    }

    try{
        String sql = "Select * from appointment WHERE patientId = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, patientId);

        ResultSet rs = stmt.executeQuery();
        boolean flag = false;
        while(rs.next()){
            flag = true;
            apo = new Appointment();
```

```
            apo.setAppointmentId(rs.getInt("appointmentId"));
            apo.setPatientId(patientId);
            apo.setDoctorId(rs.getInt("doctorId"));
            apo.setAppointmentDate(rs.getDate("appointmentDate"));
            apo.setDescription(rs.getString("description"));

            appointments.add(apo);
        }

        if(!flag){
            throw new PatientNumberNotFoundException("PATIENT NOT FOUNI
        }




    }

    catch(SQLException e)
    {
        e.printStackTrace();
    }

    finally {
        try{
            con.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }

    }

    return appointments;

}
```

```
@Override
public List<Appointment> getAppointmentsForDoctors(int doctorId){
    if(doctorId == 0 || doctorId < 0)
    {
        System.out.println("DOCTOR ID MUST BE A NUMBER WHICH IS NOT 0
    }


    List<Appointment> appointments = new ArrayList<>();


    try{

        String sql = "Select * from appointment WHERE doctorId = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, doctorId);

        ResultSet rs = stmt.executeQuery();
        boolean flag = false;
        while(rs.next()){
            flag = true;
            apo = new Appointment();
            apo.setAppointmentId(rs.getInt("appointmentId"));
            apo.setDoctorId(doctorId);
            apo.setPatientId(rs.getInt("patientId"));
            apo.setDescription(rs.getString("description"));
            apo.setAppointmentDate(rs.getDate("appointmentDate"));

            appointments.add(apo);
        }

        if(!flag){
            System.out.println("No appointments Today SIR");
        }

    }

    catch(SQLException e)
```

```
    {
        e.printStackTrace();
    }

    finally {
        try{
            con.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }

    }

    return appointments;

}

@Override
public boolean scheduleAppointment(Appointment appointment){

    if(appointment == null){
        System.out.println("NO APPOINTMENTS!");
        return false;
    }

    boolean scheduled = false;

    try{

        String sql = "Insert into appointment (patientId, doctorId, appointmentD
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, appointment.getPatientId());
        stmt.setInt(2, appointment.getDoctorId());
        stmt.setDate(3, new Date(appointment.getAppointmentDate().getTime(
        stmt.setString(4, appointment.getDescription());
```

```
            int rowsAdded = stmt.executeUpdate();

            if(rowsAdded > 0){
                scheduled = true;
                System.out.println("Appointment Scheduled Successfully");
            }
            else{
                System.out.println("SOMETHING WENT WRONG WHILE SCHEDULIN
            }


        }

         catch(SQLException e)
        {
            e.printStackTrace();
        }

        finally {
            try{
                con.close();
            }
            catch(SQLException e)
            {
                e.printStackTrace();
            }

        }
        return scheduled;

    }

    @Override
    public boolean updateAppointment(Appointment appointment){

        if(appointment == null){
            System.out.println("NO APPOINTMENTS!");
            return false;
```

```
        }
        boolean update = false;

        try{

            String sql = "Update appointment SET patientId = ?, doctorId = ?, appo
            PreparedStatement stmt = con.prepareStatement(sql);

            stmt.setInt(1, appointment.getPatientId());
            stmt.setInt(2, appointment.getDoctorId());
            stmt.setDate(3, new Date(appointment.getAppointmentDate().getTime(
            stmt.setString(4, appointment.getDescription());
            stmt.setInt(5, appointment.getAppointmentId());

            int rowsUpdated = stmt.executeUpdate();

            if(rowsUpdated > 0){
                update = true;
                System.out.println("UPDATED Successfully");
            }

            else{
                System.out.println("Something WENT WRONG");
            }


        }

        catch(SQLException e)
        {
            e.printStackTrace();
        }

        finally {
            try{
                con.close();
            }
            catch(SQLException e)
```

```
        {
            e.printStackTrace();
        }

    }

    return update;


}

@Override
public boolean cancelAppointment(int appointmentId){

    if(appointmentId ==0 || appointmentId < 0){
        System.out.println("APPOINTMENT ID CANNOT BE 0 OR NEGATIVE");
        return false;
    }

    boolean cancel = false;

    try{
        String sql = "Delete from appointment WHERE appointmentId = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, appointmentId);

        int rowsDeleted = stmt.executeUpdate();

        if(rowsDeleted > 0)
        {
            cancel = true;
            System.out.println("DELETED SUCCESSFULLY");
        }
        else{
            System.out.println("SOMETHING WENT WRONG WHILW DELETEINC
        }

    }
```

```
        catch(SQLException e)
        {
            e.printStackTrace();
        }

        finally {
            try{
                con.close();
            }
            catch(SQLException e)
            {
                e.printStackTrace();
            }

        }
        return cancel;

    }



}
```

Task 6: Created MainModule class in mainmod package.

1. In this class I created main method and in that defined switch case statements.
2. In switch cases there are options for executing the methods in HospitalManagementImpl class.
3. I have also printed the results from the methods in the Console with proper error statements.

```
package app.mainMod;

import app.dao.HospitalServiceImpl;
```

```
import app.entity.Appointment;
import app.exception.PatientNumberNotFoundException;

import java.sql.SQLException;
import java.time.LocalDate;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

public class MainModule {

    private static Date ConvertDate(LocalDate date){

        Date utilDate = Date.from(date.atStartOfDay(ZoneId.systemDefault()).toIr

        return utilDate;

    }


    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);



        while(true){
            System.out.println("--");
            System.out.println("WELCOME TO HOSPITAL MANAGEMENT SYSTEM
            System.out.println("--");
            System.out.println("Select Option");
            System.out.println("1. Get specific Appointments");
            System.out.println("2. Get Appointments for Patients");
            System.out.println("3. Get Appointments For Doctors");
            System.out.println("4. Schedule Appointment");
            System.out.println("5. Update Appointment");
```

```
System.out.println("6. Cancel Appointment");
System.out.println("0. EXIT");
System.out.println("Enter Your Option");
int option = sc.nextInt();

switch(option){
   case 1->{

      System.out.println("Enter Appointment ID");
      int apoId = sc.nextInt();

      HospitalServiceImpl impl = new HospitalServiceImpl();
      Appointment apo = impl.getAppointmentById(apoId);

      System.out.println("Your Appointment: ");

      if(apo == null){
         System.out.println("NO APPOINTMENTS");
      }
      else{
         System.out.println(apo);
      }

   }

   case 2->{
      System.out.println("Enter patient Id");
      int patientId = sc.nextInt();



      HospitalServiceImpl impl = new HospitalServiceImpl();
      List<Appointment> appointments = new ArrayList<>();
      try{
         appointments = impl.getAppointmentsForPatient(patientId);

      } catch (PatientNumberNotFoundException e) {
         e.printStackTrace();
```

```
            }

            System.out.println("Your Appointments");

            if(appointments.isEmpty())
            {
                System.out.println("NO APPOINMENTS SCHEDULED");
            }
            else{
                for(Appointment a: appointments){
                    System.out.println(a);
                }

            }

        }

        case 3->{
            System.out.println("Enter DoctorId");
            int doctorId = sc.nextInt();

            HospitalServiceImpl impl = new HospitalServiceImpl();
            List<Appointment> appointments = impl.getAppointmentsForDoct
            System.out.println("Doctor's Appointments:");

            if(appointments.isEmpty()){
                System.out.println("NO APPOINTMENTS");
            }
            else{

                for(Appointment a: appointments){
                    System.out.println(a);
                }


            }
```

```
            }

            case 4->{
                System.out.println("Schedule Appointment");

                LocalDate apDate = LocalDate.of(2025, 05, 05);

                Appointment appointment = new Appointment(
                        1, 2, ConvertDate(apDate), "Dentist tooth checkup"
                );

                HospitalServiceImpl impl = new HospitalServiceImpl();

                boolean scheduled =  impl.scheduleAppointment(appointment);

                if(scheduled){
                    System.out.println("Appointment Scheduled");
                }
                else{
                    System.out.println("SOMETHING WENT WRONG TRY AGAIN");
                }



            }

            case 5->{
                System.out.println("Enter Appointment ID of the appointment you
                int apoId = sc.nextInt();

                HospitalServiceImpl impl = new HospitalServiceImpl();
                LocalDate apDate = LocalDate.of(2025, 05, 10); //updating Date
                Appointment appointment = new Appointment(apoId, 1, 3, Convert

                boolean updateStatus = impl.updateAppointment(appointment);
                if(updateStatus){
                    System.out.println("Updated Successfully");
                }
                else{
```

```
                System.out.println("SOMETHING WENT WRONG WHILE Updati
            }

        }

        case 6->{
            System.out.println("Enter appointment ID to cancel: ");
            int apoId = sc.nextInt();
            HospitalServiceImpl impl = new HospitalServiceImpl();

            boolean deleteStatus = impl.cancelAppointment(apoId);

            if(deleteStatus){
                System.out.println("CANCELED SUCCESSFULLY");
            }
            else{
                System.out.println("SOMETHING WENT WRONG WHILE DELETI
            }

        }
        case 0->{
            System.out.println("--");
            System.out.println("Thanks for Visiting!! GoodByeeee😊");
            System.out.println("--");
            sc.close();
            System.exit(0);
        }

        default -> {
            System.out.println("WRONG OPTION");
        }
    }


}
```

```
    }

}
```

OUTPUT:

```
--
WELCOME TO HOSPITAL MANAGEMENT SYSTEM
--
Select Option
1. Get specific Appointments
2. Get Appointments for Patients
3. Get Appointments For Doctors
4. Schedule Appointment
5. Update Appointment
6. Cancel Appointment
0. EXIT
```

```
Enter Your Option
5
Enter Appointment ID of the appointment you want to update:
7
Enter Updated Patient ID: 1
Enter Updated Doctor ID: 5
Enter Updated Appointment Date (yyyy-MM-dd): 2025-05-07
Enter Updated Description: Dentist Chekcup
UPDATED Successfully
Updated Successfully
--
```

```
Enter Your Option
1
Enter Appointment ID
1
Your Appointment:
Appointment { Appointment ID: 1, Patient ID: 1, Doctor ID: 1, Appointment Date: 2025-04-10, Description: 'Regular checkup' }
--
```

```
Enter Your Option
2
Enter patient Id
1
Your Appointments
Appointment { Appointment ID: 1, Patient ID: 1, Doctor ID: 1, Appointment Date: 2025-04-10, Description: 'Regular checkup' }
Appointment { Appointment ID: 7, Patient ID: 1, Doctor ID: 5, Appointment Date: 2025-05-07, Description: 'Dentist Chekcup' }
Appointment { Appointment ID: 8, Patient ID: 1, Doctor ID: 2, Appointment Date: 2025-05-07, Description: 'Knee Injury' }
--
```

```
Enter Your Option
3
Enter DoctorId
3
Doctor's Appointments:
Appointment { Appointment ID: 2, Patient ID: 2, Doctor ID: 3, Appointment Date: 2025-04-12, Description: 'Child vaccination' }
--
```