

PYTHON CODING CHALLENGE

Coding Challenge-3: Hospital Management System

Submitted By : Astha Raj

Problem Statement:

1Create SQL Schema from the following classes class, use the class attributes for table column names.

```
mysql 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 338
Server version: 8.0.35 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database hospitalmanagementsystem
-> ;
Query OK, 1 row affected (0.04 sec)

mysql> use hospitalmanagementsystem;
Database changed
mysql> CREATE TABLE Patient (
->     patientId INT AUTO_INCREMENT PRIMARY KEY,
->     firstName VARCHAR(255),
->     lastName VARCHAR(255),
->     dateOfBirth DATE,
->     gender VARCHAR(10),
->     contactNumber VARCHAR(20),
->     address TEXT
-> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE Doctor (
->     doctorId INT AUTO_INCREMENT PRIMARY KEY,
->     firstName VARCHAR(255),
->     lastName VARCHAR(255),
->     specialization VARCHAR(255),
->     contactNumber VARCHAR(20)
-> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Appointment (
->     appointmentId INT AUTO_INCREMENT PRIMARY KEY,
->     patientId INT,
->     doctorId INT,
->     appointmentDate DATETIME,
->     description TEXT,
->     FOREIGN KEY (patientId) REFERENCES Patient(patientId),
->     FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
-> );
Query OK, 0 rows affected (0.05 sec)
```

```

MySQL 8.0 Command Line Client
->     lastName VARCHAR(255),
->     specialization VARCHAR(255),
->     contactNumber VARCHAR(20)
-> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Appointment (
->     appointmentId INT AUTO_INCREMENT PRIMARY KEY,
->     patientId INT,
->     doctorId INT,
->     appointmentDate DATETIME,
->     description TEXT,
->     FOREIGN KEY (patientId) REFERENCES Patient(patientId),
->     FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
-> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO Patient (firstName, lastName, dateOfBirth, gender, contactNumber, address)
-> VALUES
->     ('Rahul', 'Gupta', '1990-05-15', 'Male', '9876543210', '123, ABC Street, Mumbai'),
->     ('Priya', 'Sharma', '1985-09-25', 'Female', '8765432109', '456, XYZ Street, Delhi'),
->     ('Amit', 'Patel', '1978-12-03', 'Male', '7654321098', '789, PQR Street, Kolkata'),
->     ('Anita', 'Singh', '1993-07-19', 'Female', '6543210987', '101, LMN Street, Chennai'),
->     ('Sachin', 'Kumar', '1980-03-08', 'Male', '5432109876', '202, UVW Street, Bangalore'),
->     ('Aarav', 'Patel', '1995-04-10', 'Male', '9876543210', '123, ABC Street, Mumbai'),
->     ('Sneha', 'Reddy', '1988-08-20', 'Female', '8765432109', '456, XYZ Street, Delhi'),
->     ('Arjun', 'Gupta', '1979-11-15', 'Male', '7654321098', '789, PQR Street, Kolkata'),
->     ('Isha', 'Kumar', '1992-06-19', 'Female', '6543210987', '101, LMN Street, Chennai'),
->     ('Vivan', 'Sharma', '1983-03-25', 'Male', '5432109876', '202, UVW Street, Bangalore')
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO Doctor (firstName, lastName, specialization, contactNumber)
-> VALUES
->     ('Dr. Rajesh', 'Sharma', 'Cardiologist', '9876543211'),
->     ('Dr. Nisha', 'Patel', 'Pediatrician', '8765432101'),
->     ('Dr. Manoj', 'Singh', 'Dermatologist', '7654321091'),
->     ('Dr. Pooja', 'Gupta', 'Gynecologist', '6543210981'),
->     ('Dr. Amit', 'Kumar', 'Orthopedician', '5432109871'),
->     ('Dr. Neha', 'Singh', 'Oncologist', '9876543211'),
->     ('Dr. Rohan', 'Mehta', 'Neurologist', '8765432101'),
->     ('Dr. Ritu', 'Shah', 'Psychiatrist', '7654321091'),
->     ('Dr. Karan', 'Verma', 'Endocrinologist', '6543210981'),
->     ('Dr. Anika', 'Das', 'ENT Specialist', '5432109871')
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Appointment (patientId, doctorId, appointmentDate, description)
-> VALUES
->     (1, 2, '2024-02-10 09:00:00', 'Regular checkup'),
->     (2, 3, '2024-02-12 10:30:00', 'Vaccination'),
->     (3, 4, '2024-02-14 14:00:00', 'Skin treatment'),
->     (4, 5, '2024-02-16 11:15:00', 'Prenatal consultation'),
->     (5, 1, '2024-02-18 15:30:00', 'Knee pain consultation'),
->     (6, 6, '2024-02-20 09:45:00', 'Eye checkup'),
->     (7, 7, '2024-02-22 11:30:00', 'Dental cleaning'),
->     (8, 8, '2024-02-24 13:15:00', 'Allergy testing'),
->     (9, 9, '2024-02-26 15:45:00', 'Ultrasound'),
->     (10, 10, '2024-02-28 17:00:00', 'Sports injury evaluation')
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from appointment;
+-----+-----+-----+-----+-----+
| appointmentId | patientId | doctorId | appointmentDate | description |
+-----+-----+-----+-----+-----+
| 1 | 1 | 2 | 2024-02-10 09:00:00 | Regular checkup |
| 2 | 2 | 3 | 2024-02-12 10:30:00 | Vaccination |
| 3 | 3 | 4 | 2024-02-14 14:00:00 | Skin treatment |
| 4 | 4 | 5 | 2024-02-16 11:15:00 | Prenatal consultation |
| 5 | 5 | 1 | 2024-02-18 15:30:00 | Knee pain consultation |
| 6 | 6 | 6 | 2024-02-20 09:45:00 | Eye checkup |
| 7 | 7 | 7 | 2024-02-22 11:30:00 | Dental cleaning |
| 8 | 8 | 8 | 2024-02-24 13:15:00 | Allergy testing |
| 9 | 9 | 9 | 2024-02-26 15:45:00 | Ultrasound |
| 10 | 10 | 10 | 2024-02-28 17:00:00 | Sports injury evaluation |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

1. Create the following model/entity classes within package entity with variables declared private, constructors(default and parametrized),getters, setters and `toString()`

- Define `Patient` class with the following confidential attributes: a. patientId b. firstName c. lastName; d. dateOfBirth e. gender f. contactNumber g.address;

```

1  class Patient:
2      def __init__(self, patientId=None, firstName=None, lastName=None, dateOfBirth=None, gender=None, contactNumber=None, address=None):
3          self.patientId = patientId
4          self.firstName = firstName
5          self.lastName = lastName
6          self.dateOfBirth = dateOfBirth
7          self.gender = gender
8          self.contactNumber = contactNumber
9          self.address = address
10
11     def __str__(self):
12         return f"Patient ID: {self.patientId}, Name: {self.firstName} {self.lastName}, DOB: {self.dateOfBirth}, Gender: {self.gender}, Contact Number: {self.contactNumber}, Address: {self.address}"
13

```

- Define `Doctor` class with the following confidential attributes: a. doctorId b. firstName c. lastName d. specialization e. contactNumber;

```

1  class Doctor:
2      def __init__(self, doctorId=None, firstName=None, lastName=None, specialization=None, contactNumber=None):
3          self.doctorId = doctorId
4          self.firstName = firstName
5          self.lastName = lastName
6          self.specialization = specialization
7          self.contactNumber = contactNumber
8
9      def __str__(self):
10         return f"Doctor ID: {self.doctorId}, Name: {self.firstName} {self.lastName}, Specialization: {self.specialization}, Contact Number: {self.contactNumber}"
11

```

- Appointment Class: a. appointmentId b. patientId c. doctorId d. appointmentDate e. Description

```

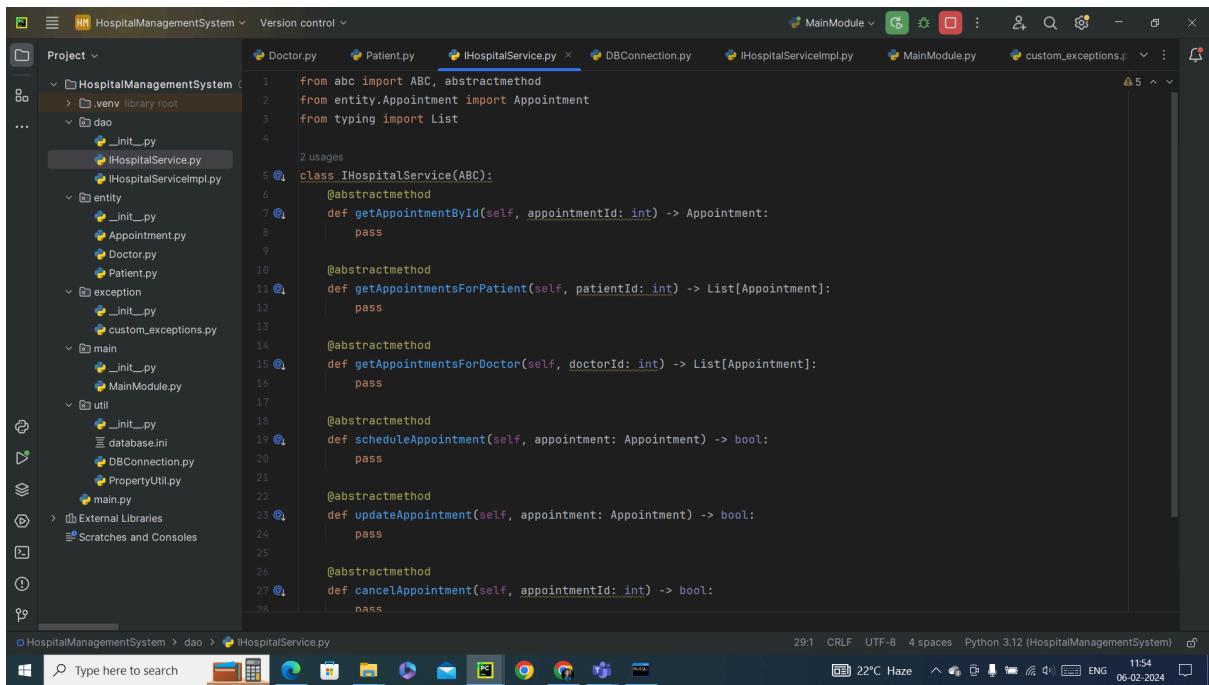
1  class Appointment:
2      def __init__(self, appointmentId=None, patientId=None, doctorId=None, appointmentDate=None, description=None):
3          self.appointmentId = appointmentId
4          self.patientId = patientId
5          self.doctorId = doctorId
6          self.appointmentDate = appointmentDate
7          self.description = description
8
9      def __str__(self):
10         return f"Appointment ID: {self.appointmentId}, Patient ID: {self.patientId}, Doctor ID: {self.doctorId}, Appointment Date: {self.appointmentDate}, Description: {self.description}"
11

```

- Implement the following for all model classes. Write default constructors and overload the constructor with parameters, getters and setters, method to print all the member variables and values.

3. Define IHospitalService interface/abstract class with following methods to interact with database Keep the interfaces and implementation classes in package dao

- a. getAppointmentById()
 - i. Parameters: appointmentId
 - ii. ReturnType: Appointment object
- b. getAppointmentsForPatient()
 - i. Parameters: patientId
 - ii. ReturnType: List of Appointment objects
- c. getAppointmentsForDoctor()
 - i. Parameters: doctorId
 - ii. ReturnType: List of Appointment objects
- d. scheduleAppointment()
 - i. Parameters: Appointment Object
 - ii. ReturnType: Boolean
- e. updateAppointment()
 - i. Parameters: Appointment Object
 - ii. ReturnType: Boolean
- f. cancelAppointment()
 - i. Parameters: AppointmentId
 - ii. ReturnType: Boolean



```
from abc import ABC, abstractmethod
from entity.Appointment import Appointment
from typing import List

class IHospitalService(ABC):
    @abstractmethod
    def getAppointmentById(self, appointmentId: int) -> Appointment:
        pass

    @abstractmethod
    def getAppointmentsForPatient(self, patientId: int) -> List[Appointment]:
        pass

    @abstractmethod
    def getAppointmentsForDoctor(self, doctorId: int) -> List[Appointment]:
        pass

    @abstractmethod
    def scheduleAppointment(self, appointment: Appointment) -> bool:
        pass

    @abstractmethod
    def updateAppointment(self, appointment: Appointment) -> bool:
        pass

    @abstractmethod
    def cancelAppointment(self, appointmentId: int) -> bool:
        pass
```

6. Define HospitalServiceImpl class and implement all the methods IHospitalServiceImpl .

The screenshot shows the PyCharm IDE interface with the project structure on the left and the code editor on the right. The code editor displays the `HospitalServiceImpl.py` file under the `HospitalManagementSystem` package. The code implements the `IHospitalServiceImpl` interface with two methods: `getAppointmentById` and `getAppointmentsForPatient`. Both methods use MySQL queries to fetch appointment data from the database.

```
import mysql.connector
from typing import List
from dao.IHospitalService import IHospitalService
from entity.Appointment import Appointment
from exception.custom_exceptions import PatientNumberNotFoundException

class HospitalServiceImpl(IHospitalService):
    def __init__(self, db_config):
        self.connection = mysql.connector.connect(**db_config)
        self.cursor = self.connection.cursor()

    def getAppointmentById(self, appointmentId: int) -> Appointment:
        query = "SELECT * FROM appointment WHERE appointmentId = %s"
        self.cursor.execute(query, params=(appointmentId,))
        result = self.cursor.fetchone()
        if result:
            return Appointment(*result)
        else:
            raise PatientNumberNotFoundException(f"Appointment with ID {appointmentId} not found")

    def getAppointmentsForPatient(self, patientId: int) -> List[Appointment]:
        query = "SELECT * FROM appointment WHERE patientId = %s"
        self.cursor.execute(query, params=(patientId,))
        results = self.cursor.fetchall()
        if results:
```

The screenshot shows the continuation of the `HospitalServiceImpl.py` file. It includes implementations for `getAppointmentsForDoctor`, `scheduleAppointment`, and `updateAppointment`. The `getAppointmentsForDoctor` method returns a list of appointments for a specific doctor. The `scheduleAppointment` method inserts a new appointment into the database. The `updateAppointment` method updates an existing appointment in the database.

```
def getAppointmentsForDoctor(self, doctorId: int) -> List[Appointment]:
    query = "SELECT * FROM appointment WHERE doctorId = %s"
    self.cursor.execute(query, params=(doctorId,))
    results = self.cursor.fetchall()
    return [Appointment(*result) for result in results]

def scheduleAppointment(self, appointment: Appointment) -> bool:
    query = "INSERT INTO appointment (patientId, doctorId, appointmentDate, description) VALUES (%s, %s, %s, %s)"
    data = (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description)
    self.cursor.execute(query, data)
    self.connection.commit()
    return True

def updateAppointment(self, appointment: Appointment) -> bool:
    query = "UPDATE appointment SET patientId = %s, doctorId = %s, appointmentDate = %s, description = %s WHERE appointmentId = %s"
    data = (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description, appointment.appointmentId)
    self.cursor.execute(query, data)
    self.connection.commit()
    return True
```

```

1 usage
def scheduleAppointment(self, appointment: Appointment) -> bool:
    query = "INSERT INTO appointment (patientId, doctorId, appointmentDate, description) VALUES (%s, %s, %s, %s)"
    data = (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description)
    self.cursor.execute(query, data)
    self.connection.commit()
    return True

1 usage
def updateAppointment(self, appointment: Appointment) -> bool:
    query = "UPDATE appointment SET patientId = %s, doctorId = %s, appointmentDate = %s, description = %s WHERE appointmentId = %s"
    data = (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description, appointment.appointmentId)
    self.cursor.execute(query, data)
    self.connection.commit()
    return True

1 usage
def cancelAppointment(self, appointmentId: int) -> bool:
    query = "DELETE FROM appointment WHERE appointmentId = %s"
    self.cursor.execute(query, params=(appointmentId,))
    self.connection.commit()
    return True

```

7. Create a utility class DBConnection in a package util with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file. Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property file containing connection details like hostname, dbname, username, password, port number and returns a connection string.

```

host=localhost
user=root
password=Abcd300#$
database=hospitalmanagementsystem
port=3306

```

```

Doctor.py Patient.py IHospitalService.py DBConnection.py x PropertyUtil.py IHospitalServiceImpl.py database.ini
1 import mysql.connector
2
3 from util.PropertyUtil import PropertyUtil
4
5
6 usages
7 class DBConnection:
8     connection = None
9
10     @staticmethod
11     def getConnection():
12         if DBConnection.connection is None:
13             connection_string = PropertyUtil.getPropertyString()
14             DBConnection.connection = mysql.connector.connect(**connection_string)
15
16     return DBConnection.connection

```



```

Doctor.py Patient.py IHospitalService.py DBConnection.py x PropertyUtil.py IHospitalServiceImpl.py database.ini
1 from configparser import ConfigParser
2
3
4 usages
5 class PropertyUtil:
6     1 usage
7     @staticmethod
8     def getPropertyString():
9         config = ConfigParser()
10        config.read('database.ini')
11
12        hostname = config['DATABASE']['hostname']
13        dbname = config['DATABASE']['dbname']
14        username = config['DATABASE']['username']
15        password = config['DATABASE']['password']
16        port = config['DATABASE']['port']
17
18        connection_string = {
19            'host': hostname,
20            'database': dbname,
21            'user': username,
22            'password': password,
23            'port': port
24        }
25
26        return connection_string

```

8. Create the exceptions in package myexceptions Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method, 1.

PatientNumberNotFoundException :throw this exception when user enters an invalid patient number which doesn't exist in db

```

Service.py DBConnection.py PropertyUtil.py IHospitalServiceImpl.py database.ini MainModule.py custom_exceptions.py
1 usages
2 class PatientNumberNotFoundException(Exception):
3     pass

```

9. Create class named MainModule with main method in package mainmod. Trigger all the methods in service implementation class.

The screenshot shows the PyCharm IDE interface with the project 'HospitalManagementSystem' open. The file 'MainModule.py' is selected in the left sidebar under the 'main' package. The code in the editor is as follows:

```

from dao.IHospitalServiceImpl import HospitalServiceImpl
from entity.Appointment import Appointment
from exception.custom_exceptions import PatientNumberNotFoundException

class MainModule:
    def __init__(self):
        self.service = HospitalServiceImpl()

    def main():
        db_config = {
            'host': 'localhost',
            'database': 'hospitalmanagementsystem',
            'user': 'root',
            'password': 'Bcd300#',
            'port': '3306'
        }

        service = HospitalServiceImpl(db_config)

        while True:
            print("\nHospital Management System Menu:")
            print("1. Get appointment by ID")
            print("2. Get appointments for patient")
            print("3. Get appointments for doctor")
            print("4. Schedule appointment")
            print("5. Update appointment")
            print("6. Cancel appointment")
            choice = input("Enter your choice: ")

            if choice == "1":
                appointment_id = int(input("Enter appointment ID: "))
                try:
                    appointment = service.getAppointmentById(appointment_id)
                    print(appointment)
                except PatientNumberNotFoundException as e:
                    print(e)

            elif choice == "2":
                patient_id = int(input("Enter patient ID: "))
                try:
                    appointments = service.getAppointmentsForPatient(patient_id)
                    for appointment in appointments:
                        print(appointment)
                except PatientNumberNotFoundException as e:
                    print(e)

            elif choice == "3":
                doctor_id = int(input("Enter doctor ID: "))
                appointments = service.getAppointmentsForDoctor(doctor_id)
                for appointment in appointments:
                    print(appointment)

            elif choice == "4":
                patient_id = int(input("Enter patient ID: "))
                try:
                    service.scheduleAppointment(patient_id)
                    print("Appointment scheduled successfully!")
                except PatientNumberNotFoundException as e:
                    print(e)

            elif choice == "5":
                appointment_id = int(input("Enter appointment ID: "))
                try:
                    service.updateAppointment(appointment_id)
                    print("Appointment updated successfully!")
                except PatientNumberNotFoundException as e:
                    print(e)

            elif choice == "6":
                appointment_id = int(input("Enter appointment ID: "))
                try:
                    service.cancelAppointment(appointment_id)
                    print("Appointment canceled successfully!")
                except PatientNumberNotFoundException as e:
                    print(e)

            else:
                print("Invalid choice. Please enter a number between 1 and 6.")

        print("Exiting program...")

```

The screenshot continues from the previous one, showing the rest of the 'MainModule.py' code. The code handles user input for scheduling, updating, and canceling appointments.

```

            elif choice == "5":
                appointment_id = int(input("Enter appointment ID: "))
                try:
                    service.updateAppointment(appointment_id)
                    print("Appointment updated successfully!")
                except PatientNumberNotFoundException as e:
                    print(e)

            elif choice == "6":
                appointment_id = int(input("Enter appointment ID: "))
                try:
                    service.cancelAppointment(appointment_id)
                    print("Appointment canceled successfully!")
                except PatientNumberNotFoundException as e:
                    print(e)

            else:
                print("Invalid choice. Please enter a number between 1 and 6.")

        print("Exiting program...")

```

```
elif choice == "4":
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    appointment_date = input("Enter appointment date (YYYY-MM-DD HH:MM:SS): ")
    description = input("Enter appointment description: ")
    new_appointment = Appointment(patientId=patient_id, doctorId=doctor_id, appointmentDate=appointment_date, description=description)
    success = service.scheduleAppointment(new_appointment)
    if success:
        print("Appointment scheduled successfully")
    else:
        print("Failed to schedule appointment")

elif choice == "5":
    appointment_id = int(input("Enter appointment ID: "))
    patient_id = int(input("Enter new patient ID: "))
    doctor_id = int(input("Enter new doctor ID: "))
    appointment_date = input("Enter new appointment date (YYYY-MM-DD HH:MM:SS): ")
    description = input("Enter new appointment description: ")
    updated_appointment = Appointment(appointmentId=appointment_id, patientId=patient_id, doctorId=doctor_id, appointmentDate=appointment_date, description=description)
    success = service.updateAppointment(updated_appointment)
    if success:
        print("Appointment updated successfully")
    else:
        print("Failed to update appointment")

elif choice == "6":
    appointment_id = int(input("Enter appointment ID to cancel: "))
    success = service.cancelAppointment(appointment_id)
    if success:
        print("Appointment canceled successfully")
    else:
        print("Failed to cancel appointment")

elif choice == "7":
    print("Exiting...")
    break

else:
    print("Invalid choice. Please try again.")
```

```
if success:
    print("Appointment updated successfully")
else:
    print("Failed to update appointment")

elif choice == "6":
    appointment_id = int(input("Enter appointment ID to cancel: "))
    success = service.cancelAppointment(appointment_id)
    if success:
        print("Appointment canceled successfully")
    else:
        print("Failed to cancel appointment")

elif choice == "7":
    print("Exiting...")
    break

else:
    print("Invalid choice. Please try again.")

if __name__ == "__main__":
    MainModule.main()
```

```
C:\Users\astha\PycharmProjects\HospitalManagementSystem\venv\Scripts\python.exe C:\Users\astha\PycharmProjects\HospitalManagementSystem\main>MainModule.py

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 1
Enter appointment ID: 2
Appointment ID: 2, Patient ID: 2, Doctor ID: 3, Appointment Date: 2024-02-12 10:30:00, Description: Vaccination

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: |
```

```
Appointment ID: 5, Patient ID: 5, Doctor ID: 1, Appointment Date: 2024-02-18 15:30:00, Description: Knee pain consultation
Appointment ID: 10, Patient ID: 5, Doctor ID: 6, Appointment Date: 2020-01-01 00:00:00, Description: ggg
Appointment ID: 12, Patient ID: 5, Doctor ID: 5, Appointment Date: 2020-12-12 10:10:20, Description: Regular Checkup

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 2
Enter patient ID: 5
Appointment ID: 5, Patient ID: 5, Doctor ID: 1, Appointment Date: 2024-02-18 15:30:00, Description: Knee pain consultation
Appointment ID: 10, Patient ID: 5, Doctor ID: 6, Appointment Date: 2020-01-01 00:00:00, Description: ggg
Appointment ID: 12, Patient ID: 5, Doctor ID: 5, Appointment Date: 2020-12-12 10:10:20, Description: Regular Checkup

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 3
```

```
Project ▾ Service.py DBConnection.py PropertyUtil.py IHospitalServiceImpl.py database.ini MainModule.py custom_exceptions.py ... Run MainModule ...
...
enter patient ID: 3
Appointment ID: 5, Patient ID: 5, Doctor ID: 1, Appointment Date: 2024-02-18 15:30:00, Description: Knee pain consultation
Appointment ID: 10, Patient ID: 5, Doctor ID: 6, Appointment Date: 2020-01-01 00:00:00, Description: ggg
Appointment ID: 12, Patient ID: 5, Doctor ID: 5, Appointment Date: 2020-12-12 10:10:20, Description: Regular Checkup
...
Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 3
Enter doctor ID: 6
Appointment ID: 6, Patient ID: 6, Doctor ID: 6, Appointment Date: 2024-02-20 09:45:00, Description: Eye checkup
Appointment ID: 10, Patient ID: 5, Doctor ID: 6, Appointment Date: 2020-01-01 00:00:00, Description: ggg

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: |
```

```
Project ▾ Service.py DBConnection.py PropertyUtil.py IHospitalServiceImpl.py database.ini MainModule.py custom_exceptions.py ... Run MainModule ...
...
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 4
Enter patient ID: 2
Enter doctor ID: 6
Enter appointment date (YYYY-MM-DD HH:MM:SS): 2024-02-15 10:10:50
Enter appointment description: Regular Checkup
Appointment scheduled successfully

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 3
Enter doctor ID: 6
Appointment ID: 6, Patient ID: 6, Doctor ID: 6, Appointment Date: 2024-02-20 09:45:00, Description: Eye checkup
Appointment ID: 10, Patient ID: 5, Doctor ID: 6, Appointment Date: 2020-01-01 00:00:00, Description: ggg
Appointment ID: 13, Patient ID: 2, Doctor ID: 6, Appointment Date: 2024-02-15 10:10:50, Description: Regular Checkup
...
Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: |
```

```
Appointment ID: 13, Patient ID: 2, Doctor ID: 6, Appointment Date: 2024-02-10 10:05:05, Description: Regular Checkup

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit

Enter your choice: 5
Enter appointment ID: 13
Enter new patient ID: 2
Enter new doctor ID: 6
Enter new appointment date (YYYY-MM-DD HH:MM:SS): 2024-02-10 10:05:05
Enter new appointment description: Eye Checkup
Appointment updated successfully

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit

Enter your choice: |
```

```
Enter new patient ID: 2
Enter new doctor ID: 6
Enter new appointment date (YYYY-MM-DD HH:MM:SS): 2024-02-10 10:05:05
Enter new appointment description: Eye Checkup
Appointment updated successfully

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit

Enter your choice: 6
Enter appointment ID to cancel: 13
Appointment canceled successfully

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit

Enter your choice:
```

A screenshot of the PyCharm IDE interface. The terminal window displays the output of a Python application named 'HospitalManagementSystem'. The application's main menu is shown, with option 6 selected to cancel an appointment. The user enters appointment ID 13, and the message 'Appointment canceled successfully' is printed. The application then exits.

```
Appointment updated successfully
...
Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 6
Enter appointment ID to cancel: 13
Appointment canceled successfully

Hospital Management System Menu:
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 7
Exiting...
Process finished with exit code 0
```