HEXAWARE CODING CHALLENGE

-BY AAYUSHI GUPTA

TOPIC-HOSPITAL MANAGEMENT SYSTEM

Problem Statement:

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

```
CREATE DATABASE HospitalManagementSystem;
USE HospitalManagementSystem;

CREATE TABLE Patients (
    patientId INT PRIMARY KEY,
    firstName VARCHAR(50),
    lastName VARCHAR(50),
    dateOfBirth DATE,
    gender CHAR(1),
    contactNumber VARCHAR(15),
    address VARCHAR(255)
);
CREATE TABLE Doctors (
    doctorId VARCHAR(10) PRIMARY KEY,
    firstName VARCHAR(50),
    lastName VARCHAR(50),
    specialization VARCHAR(100),
    contactNumber VARCHAR(15),
    address VARCHAR(255)
);
CREATE TABLE Appointments (
    appointmentId INT PRIMARY KEY,
    patientId INT FOREIGN KEY REFERENCES Patients(patientId),
    doctorId VARCHAR(10) FOREIGN KEY REFERENCES Doctors(doctorId),
    appointmentDate DATETIME,
    description VARCHAR(255)
);

INSERT INTO Patients (patientId, firstName, lastName, dateOfBirth, gender,
contactNumber, address)
VALUES
```

(1, 'Aman', 'Singh', '1990-05-12', 'M', '9876543213', '123 Street A, City'),
(2, 'Nisha', 'Khan', '1985-09-21', 'F', '9876543214', '456 Street B, City'),
(3, 'Suman', 'Patel', '2000-02-15', 'F', '9876543215', '789 Street C, City');

INSERT INTO Doctors (doctorId, firstName, lastName, specialization, contactNumber, address)VALUES
('D001', 'Shrey', 'Gupta', 'Dentist', '9876543210', 'Block A, Clinic Center, City'),
('D002', 'Priya', 'Mehta', 'Cardiologist', '9876543211', 'Block B, Hospital Complex, City'),
('D003', 'Raj', 'Sharma', 'Orthopedic', '9876543212', 'Block C, Health Plaza, City');

INSERT INTO Appointments (appointmentId, patientId, doctorId, appointmentDate, description)VALUES
(1, 1, 'D001', '2024-10-09 10:30', 'Dental checkup'),
(2, 2, 'D002', '2024-10-09 14:00', 'Heart checkup'),
(3, 3, 'D003', '2024-10-10 09:00', 'Back pain consultation');

1. Create the following **model/entity classes** within package **entity** with variables declared private,
constructors(default and parametrized,getters,setters and toString())
1. Define `**Patient**` class with the following confidential attributes:
a. patientId
b. firstName
c. lastName;
d. dateOfBirth
e. gender
f. contactNumber
g. address;

Code :

```
class Patient:
    def __init__(self, patient_id=None, first_name=None, last_name=None, date_of_birth=None, gender=None, contact_number=None, address=None):
        self.__patient_id = patient_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__gender = gender
        self.__contact_number = contact_number
```

```
        self.__address = address

    # Getters and Setters
    def get_patient_id(self):
        return self.__patient_id

    def set_patient_id(self, patient_id):
        self.__patient_id = patient_id

    # Repeat for other attributes

    def __str__(self):
        return f"Patient[ID: {self.__patient_id}, Name: {self.__first_name}
{self.__last_name}, DOB: {self.__date_of_birth}, Gender: {self.__gender},
Contact: {self.__contact_number}, Address: {self.__address}]"
```

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

Code:

```
class Doctor:
    def __init__(self, doctor_id=None, first_name=None, last_name=None,
specialization=None, contact_number=None):
        self.__doctor_id = doctor_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__specialization = specialization
        self.__contact_number = contact_number

    # Getters and Setters
    def get_doctor_id(self):
        return self.__doctor_id

    # Repeat for other attributes

    def __str__(self):
```

```
        return f"Doctor[ID: {self.__doctor_id}, Name: {self.__first_name}
{self.__last_name}, Specialization: {self.__specialization}, Contact:
{self.__contact_number}]"
```

3. Appointment Class:

a. appointmentId
b. patientId
c. doctorId
d. appointmentDate
e. description

Code:

```
class Appointment:
    def __init__(self, appointment_id=None, patient_id=None, doctor_id=None,
appointment_date=None, description=None):
        self.__appointment_id = appointment_id
        self.__patient_id = patient_id
        self.__doctor_id = doctor_id
        self.__appointment_date = appointment_date
        self.__description = description

    # Getters and Setters
    def get_appointment_id(self):
        return self.__appointment_id

    # Repeat for other attributes

    def __str__(self):
        return f"Appointment[ID: {self.__appointment_id}, Patient ID:
{self.__patient_id}, Doctor ID: {self.__doctor_id}, Date:
{self.__appointment_date}, Description: {self.__description}]"
```

2. 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.

Code :

class Patient:

```
    def __init__(self, patientId=None, firstName=None, lastName=None,
dateOfBirth=None, gender=None, contactNumber=None, address=None):
        self.__patientId = patientId
        self.__firstName = firstName
        self.__lastName = lastName
        self.__dateOfBirth = dateOfBirth
        self.__gender = gender
        self.__contactNumber = contactNumber
        self.__address = address

    # Getters and Setters
    def get_patientId(self):
        return self.__patientId

    def set_patientId(self, patientId):
        self.__patientId = patientId

    def get_firstName(self):
        return self.__firstName

    def set_firstName(self, firstName):
        self.__firstName = firstName

    def get_lastName(self):
        return self.__lastName

    def set_lastName(self, lastName):
        self.__lastName = lastName

    def get_dateOfBirth(self):
        return self.__dateOfBirth

    def set_dateOfBirth(self, dateOfBirth):
        self.__dateOfBirth = dateOfBirth

    def get_gender(self):
        return self.__gender

    def set_gender(self, gender):
        self.__gender = gender
```

```
    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_address(self):
        return self.__address

    def set_address(self, address):
        self.__address = address

    def __str__(self):
        return f"Patient(ID: {self.__patientId}, Name: {self.__firstName}
{self.__lastName})"

class Doctor:
    def __init__(self, doctorId=None, firstName=None, lastName=None,
    specialization=None, contactNumber=None):
        self.__doctorId = doctorId
        self.__firstName = firstName
        self.__lastName = lastName
        self.__specialization = specialization
        self.__contactNumber = contactNumber

    # Getters and Setters
    def get_doctorId(self):
        return self.__doctorId

    def set_doctorId(self, doctorId):
        self.__doctorId = doctorId

    def get_firstName(self):
        return self.__firstName

    def set_firstName(self, firstName):
        self.__firstName = firstName

    def get_lastName(self):
        return self.__lastName
```

```
    def set_lastName(self, lastName):
        self.__lastName = lastName

    def get_specialization(self):
        return self.__specialization

    def set_specialization(self, specialization):
        self.__specialization = specialization

    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def __str__(self):
        return f"Doctor(ID: {self.__doctorId}, Name: {self.__firstName}
        {self.__lastName}, Specialization: {self.__specialization})"

class Appointment:
    def __init__(self, appointmentId, patientId, doctorId, appointmentDate,
description):
        self.appointmentId = appointmentId
        self.patientId = patientId
        self.doctorId = doctorId
        self.appointmentDate = appointmentDate
        self.description = description

    def get_patientId(self):
        return self.patientId

    def get_doctorId(self):
        return self.doctorId

    def get_appointmentDate(self):
        return self.appointmentDate

    def get_description(self):
        return self.description

    def __str__(self):
```

```
        return (f"Appointment ID: {self.appointmentId}, "
                f"Patient ID: {self.patientId}, "
                f"Doctor ID: {self.doctorId}, "
                f"Date: {self.appointmentDate}, "
                f"Description: {self.description}")
```

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

Code:

 # dao/i_hospital_service.py

```
from abc import ABC, abstractmethod
from entity.appointment import Appointment

class IHospitalService(ABC):
    @abstractmethod
    def getAppointmentById(self, appointmentId):
        pass
```

```
    @abstractmethod
    def getAppointmentsForPatient(self, patientId):
        pass

    @abstractmethod
    def getAppointmentsForDoctor(self, doctorId):
        pass

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

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

    @abstractmethod
    def cancelAppointment(self, appointmentId):
        pass
```

6. Define **HospitalServiceImpl** class and implement all the methods I**HospitalServiceImpl .**

Code:

```
from dao.i_hospital_service import IHospitalService
from entity.appointment import Appointment
from exception.patient_number_not_found_exception import
PatientNumberNotFoundException
from util.db_conn_util import DBConnUtil
from tabulate import tabulate
from datetime import datetime

class HospitalServiceImpl(IHospitalService):

    def getAppointmentById(self, appointmentId):
        connection = DBConnUtil.get_connection()
        cursor = connection.cursor()
        try:
            cursor.execute("SELECT * FROM Appointments WHERE
appointmentId = ?", (appointmentId,))
```

```
        appointment_data = cursor.fetchone()
        if appointment_data:
            # Create a list of appointment details for tabulate
            appointment_details = [
                ["Appointment ID", appointment_data[0]],
                ["Patient ID", appointment_data[1]],
                ["Doctor ID", appointment_data[2]],
                ["Appointment Date", appointment_data[3]],
                ["Description", appointment_data[4]],
            ]
            print("\n********* Appointment Details *********")
            print(tabulate(appointment_details, tablefmt="grid"))
            print("***************************************\n")
        else:
            print("\n***** Appointment not found!!!!! *****\n")
            return None
    except Exception as e:
        print(f"An error occurred while fetching appointment: {e}")
        return None
    finally:
        cursor.close()

def getAppointmentsForPatient(self, patientId):
    connection = DBConnUtil.get_connection()
    cursor = connection.cursor()
    try:
        cursor.execute("SELECT * FROM Appointments WHERE patientId = ?",
        (patientId,))
        appointments = []
        for row in cursor.fetchall():
            appointments.append(Appointment(
                appointmentId=row[0],
                patientId=row[1],
                doctorId=row[2],
                appointmentDate=row[3],
                description=row[4]
            ))
        return appointments
    except Exception as e:
        print(f"An error occurred while fetching appointments for patient: {e}")
        return []
```

```
        finally:
            cursor.close()

    def getAppointmentsForDoctor(self, doctorId):
        connection = DBConnUtil.get_connection()
        cursor = connection.cursor()
        try:
            cursor.execute("SELECT * FROM Appointments WHERE doctorId = ?",
            (doctorId,))
            appointments = []
            for row in cursor.fetchall():
                appointments.append(Appointment(
                    appointmentId=row[0],
                    patientId=row[1],
                    doctorId=row[2],
                    appointmentDate=row[3],
                    description=row[4]
                ))
            return appointments
        except Exception as e:
            print(f"An error occurred while fetching appointments for doctor: {e}")
            return []
        finally:
            cursor.close()

    def doctor_exists(self, doctor_id):
        connection = DBConnUtil.get_connection()
        cursor = connection.cursor()
        try:
            cursor.execute("SELECT COUNT(*) FROM Doctors WHERE doctorId
            = ?", (doctor_id,))
            count = cursor.fetchone()[0]
            return count > 0  # Return True if doctor exists
        except Exception as e:
            print(f"An error occurred while checking doctor existence: {e}")
            return False
        finally:
            cursor.close()


    def scheduleAppointment(self, appointment: Appointment):
```

```
connection = DBConnUtil.get_connection()
cursor = connection.cursor()

try:
# Check if the patient ID exists
    cursor.execute("SELECT COUNT(*) FROM Patients WHERE patientId
    = ?", (appointment.get_patientId(),))
    patient_exists = cursor.fetchone()[0]
    if not patient_exists:
        print("\n**********************************************")
        print(f" Patient ID {appointment.get_patientId()} does not exist.")
        print("\n**********************************************")
        return False

# Check if the doctor ID exists
    cursor.execute("SELECT COUNT(*) FROM Doctors WHERE
      doctorId= ?", (appointment.get_doctorId(),))
    doctor_exists = cursor.fetchone()[0]
    if not doctor_exists:
        print("\n**********************************************")
        print(f"Error: Doctor ID {appointment.get_doctorId()} does not exist.")
        print("\n**********************************************")
        return False

# If both IDs exist, proceed with scheduling the appointment
    cursor.execute(
        "INSERT INTO Appointments (appointmentId, patientId, doctorId,
        appointmentDate, description) VALUES (?, ?, ?, ?, ?)",
        (appointment.appointmentId, appointment.get_patientId(),
        appointment.get_doctorId(), appointment.get_appointmentDate(),
        appointment.get_description())
    )
    connection.commit()
    print("\n**************************************")
    print("Appointment scheduled successfully!")
    print("\n**************************************")
    return True

except Exception as e:
    print(f"An error occurred while scheduling appointment: {e}")
    return False
```

```
    finally:
        cursor.close()

def cancelAppointment(self, appointmentId):
    connection = DBConnUtil.get_connection()
    cursor = connection.cursor()

    try:
    # First, check if the appointment ID exists
        cursor.execute("SELECT COUNT(*) FROM Appointments WHERE
        appointmentId = ?", (appointmentId,))
        appointment_exists = cursor.fetchone()[0]

        if appointment_exists == 0:
            print(f"Appointment ID {appointmentId} does not exist.")
            return False

       # If the appointment exists, proceed to delete it
        cursor.execute("DELETE FROM Appointments WHERE
            appointmentId= ?", (appointmentId,))
        connection.commit()
        return True

    except Exception as e:
        print(f"An error occurred while canceling appointment: {e}")
        return False

    finally:
        cursor.close()

def patient_exists(self, patientId):
    connection = DBConnUtil.get_connection()
    cursor = connection.cursor()
    try:
        cursor.execute("SELECT COUNT(*) FROM Patients WHERE patientId
        = ?", (patientId,))
        count = cursor.fetchone()[0]
        return count > 0  # Return True if patient exists
    except Exception as e:
        print(f"An error occurred while checking patient existence: {e}")
```

```
            return False
        finally:
            cursor.close()

    def updateAppointment(self, appointment: Appointment):
        connection = DBConnUtil.get_connection()
        cursor = connection.cursor()
        try:
        # Check if the appointment ID exists
            cursor.execute("SELECT COUNT(*) FROM Appointments WHERE
            appointmentId = ?", (appointment.appointmentId,))
            count = cursor.fetchone()[0]
                if count == 0:
                print("\n**********************************************")
                print("Appointment ID does not exist.")
                return False

        # Proceed with the update if the ID exists
            cursor.execute(
                "UPDATE Appointments SET patientId = ?, doctorId = ?,
appointmentDate = ?, description = ? WHERE appointmentId = ?",
                (appointment.get_patientId(), appointment.get_doctorId(),
                  appointment.get_appointmentDate(), appointment.get_description(),
                  appointment.appointmentId)
            )
            connection.commit()
            return True
        except Exception as e:
            print(f"An error occurred while updating appointment: {e}")
            return False
        finally:
            cursor.close()
```

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.

Code:

```
import pyodbc
```

```
from util.db_property_util import DBPropertyUtil
class DBConnUtil:
    connection = None

    @staticmethod
    def get_connection():
        if DBConnUtil.connection is None:
            connection_string = DBPropertyUtil.get_property_string()
            DBConnUtil.connection = pyodbc.connect(connection_string)
        return DBConnUtil.connection


    def get_next_appointment_id():
        connection = DBConnUtil.get_connection()
        cursor = connection.cursor()
        try:
            cursor.execute("SELECT MAX(appointmentId) FROM Appointments")
            max_id = cursor.fetchone()[0]
            return (max_id + 1) if max_id is not None else 1
        except Exception as e:
            print(f"An error occurred while fetching the next appointment ID: {e}")
            return 1  # Default to 1 if an error occurs
        finally:
            cursor.close()
```

Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

Code:

```
# util/db_property_util.py
class DBPropertyUtil:
    @staticmethod
    def get_property_string():
        # Define your connection details here
        hostname = "LAPTOP-N5SA57O6\MSSQLSERVER01"
        database = "HospitalManagementSystem"
```

```
    connection_string = f"DRIVER={{SQL Server}};" \
                        f"SERVER={hostname};" \
                        f"DATABASE={database};" \

    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

Code:

```
# exception/patient_number_not_found_exception.py
class PatientNumberNotFoundException(Exception):
    def __init__(self, message="Patient number not found"):
        self.message = message
        super().__init__(self.message)
```

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

Code:

```
# main/main_module.py
import os
from dao.hospital_service_impl import HospitalServiceImpl
from entity.appointment import Appointment
from exception.patient_number_not_found_exception import
PatientNumberNotFoundException
from util.db_conn_util import DBConnUtil
from tabulate import tabulate
from datetime import datetime

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

```
def menu(self):
    while True:
        menu = [
            ["1.", "Get Appointment Details by ID"],
            ["2.", "Schedule Appointment"],
            ["3.", "Update Appointment"],
            ["4.", "Cancel Appointment"],
            ["5.", "Get Appointments for Patient"],
            ["6.", "Get Appointments for Doctor"],
            ["7.", "Exit"]
        ]

        # Print the menu using tabulate
        print(tabulate(menu, headers=["Option", "Description"], tablefmt="grid"))

        choice = input("\nEnter your choice: ")
        if choice == '1':
            appointment_id = input("\nEnter appointment ID: ")
            try:
                # Convert appointment_id to an integer
                appointment_id = int(appointment_id)
                appointment = self.service.getAppointmentById(appointment_id)
                if appointment is None:
                    print("")
            except ValueError:
                print("Invalid input. Please enter a numeric appointment ID.")
            except PatientNumberNotFoundException as e:
                print(e)

        elif choice == '2':
            patient_id = input("Enter patient ID: ")
            doctor_id = input("Enter doctor ID: ")
            appointment_date = input("Enter appointment date (YYYY-MM-DD
                                     HH:MM): ")
            description = input("Enter appointment description: ")

            # Get the next available appointment ID
            appointment_id = DBConnUtil.get_next_appointment_id()

            if appointment_id is None:
```

```
            print("Failed to fetch next appointment ID. Please try again.")
        else:
            # Create Appointment object
            appointment = Appointment(
                appointmentId=appointment_id,  # Use the generated ID
                patientId=int(patient_id),  # Assuming patient ID is numeric
                doctorId=doctor_id,
                appointmentDate=appointment_date,
                description=description
            )

        # Schedule the appointment using the service
        if self.service.scheduleAppointment(appointment):
            print("Appointment scheduled successfully.")
        else:
            print("Failed to schedule appointment. Try Again\n")

    elif choice == '3':
        appointment_id = input("\nEnter appointment ID to update: ")
        try:
            appointment_id = int(appointment_id)
        except ValueError:
            print("Invalid input. Please enter a numeric appointment ID.")
            continue  # Go back to the menu

        new_patient_id = input("Enter new patient ID: ")
        if not self.service.patient_exists(new_patient_id):
            print("The specified patient ID does not exist. Please check and try
                  again.")
            continue

        new_doctor_id = input("Enter new doctor ID: ")
        new_appointment_date_str = input("Enter new appointment date
                                         (YYYY-MM-DD HH:MM): ")

        try:
            new_appointment_date =
datetime.strptime(new_appointment_date_str, '%Y-%m-%d %H:%M')
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD HH:MM.")
            continue
```

```
        new_description = input("Enter new appointment description: ")
            appointment = Appointment(
            appointmentId=appointment_id,
            patientId=new_patient_id,
            doctorId=new_doctor_id,
            appointmentDate=new_appointment_date,
            description=new_description
        )
            if self.service.updateAppointment(appointment):
            print("\n**************************************")
            print("\tAppointment updated successfully.")
            print("\n**************************************")

        else:
            print("Failed to update appointment.")
            print("\n**************************************")

    elif choice == '4':
        appointment_id = input("Enter appointment ID to cancel: ")

        if self.service.cancelAppointment(appointment_id):
            print("Appointment canceled successfully.")
        else:

            print("\n**********************************************
            ******")
            print(f"\tAppointment ID {appointment_id} does not exist.\n \tfailed
                to cancel
                appointment.\n**********************************************
                ***********")
    elif choice == '5':
        patient_id = input("Enter patient ID to fetch appointments: ")
        appointments = self.service.getAppointmentsForPatient(patient_id)

        if appointments:
            print(f"\nAppointments for patient ID: {patient_id}\n")
                table_data = [[appointment.appointmentId,
                appointment.doctorId, appointment.appointmentDate,
                appointment.description]
             for appointment in appointments]
```

```
                headers = ["Appointment ID", "Doctor ID", "Appointment Date",
                           "Description"]

                print(tabulate(table_data, headers=headers, tablefmt="grid"))
                print("\n")
            else:
                print(f"\n------No appointments found for patient ID: {patient_id}----
                      --")
        elif choice == '6':
            doctor_id = input("Enter doctor ID to fetch appointments: ")
                if not self.service.doctor_exists(doctor_id):
                print("The doctor ID does not exist. Please check and try again.")
                continue  # Go back to the menu
               appointments = self.service.getAppointmentsForDoctor(doctor_id)
              if appointments:
                print(f"\nAppointments for doctor ID: {doctor_id}\n")
                table_data = [[appointment.appointmentId, appointment.patientId,
                               appointment.appointmentDate, appointment.description]
                              for appointment in appointments]

                headers = ["Appointment ID", "Patient ID", "Appointment Date",
                     "Description"]
                print(tabulate(table_data, headers=headers, tablefmt="grid"))
                print("\n")
            else:
                print(f"No appointments found for doctor ID: {doctor_id}")

        elif choice == '7':
            print("Exiting the system.")
            break
        else:
            print("Invalid choice. Please try again.")

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

OUTPUT

	patientId	firstName	lastName	dateOfBirth	gender	contactNumber	address
1	1	Aman	Singh	1990-05-12	M	9876543213	123 Street A, City
2	2	Nisha	Khan	1985-09-21	F	9876543214	456 Street B, City
3	3	Suman	Patel	2000-02-15	F	9876543215	789 Street C, City

	doctorId	firstName	lastName	specialization	contactNumber	address
1	D001	Shrey	Gupta	Dentist	9876543210	Block A, Clinic Center, City
2	D002	Priya	Mehta	Cardiologist	9876543211	Block B, Hospital Complex, City
3	D003	Raj	Sharma	Orthopedic	9876543212	Block C, Health Plaza, City

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	1	D001	2024-10-09 10:30:00.000	Dental checkup
2	2	2	D002	2024-10-09 14:00:00.000	Heart checkup
3	3	3	D003	2024-10-10 09:00:00.000	Back pain consultation

```
+----------+------------------------------+
|   Option | Description                  |
+==========+==============================+
|        1 | Get Appointment Details by ID |
+----------+------------------------------+
|        2 | Schedule Appointment         |
+----------+------------------------------+
|        3 | Update Appointment           |
+----------+------------------------------+
|        4 | Cancel Appointment           |
+----------+------------------------------+
|        5 | Get Appointments for Patient  |
+----------+------------------------------+
|        6 | Get Appointments for Doctor   |
+----------+------------------------------+
|        7 | Exit                         |
+----------+------------------------------+

Enter your choice: 1

Enter appointment ID: 3

********* Appointment Details *********
+------------------+-------------------------+
| Appointment ID   | 3                       |
+------------------+-------------------------+
| Patient ID       | 3                       |
+------------------+-------------------------+
| Doctor ID        | D003                    |
+------------------+-------------------------+
| Appointment Date | 2024-10-10 09:00:00     |
+------------------+-------------------------+
| Description      | Back pain consultation  |
+------------------+-------------------------+
***************************************
```

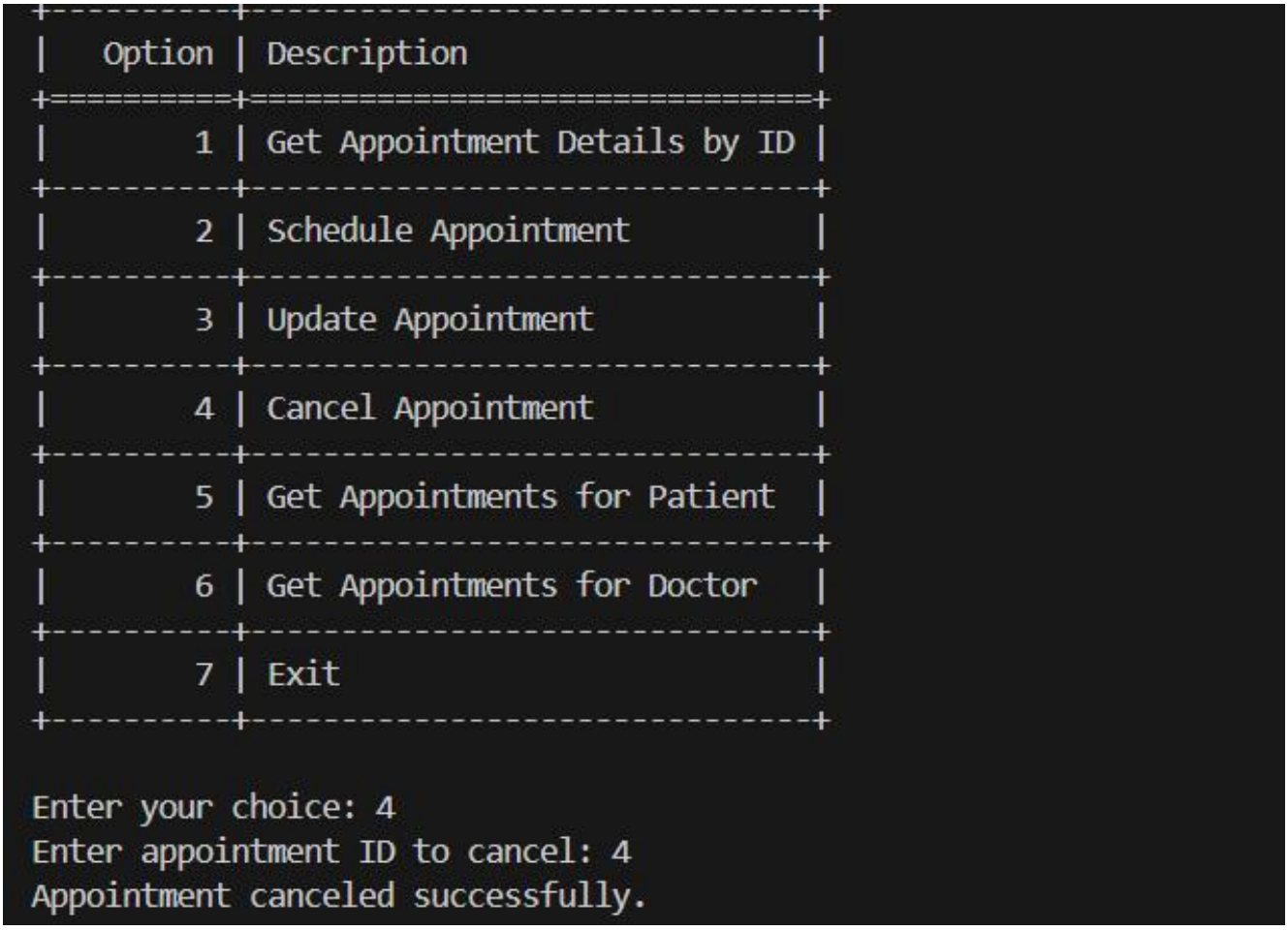
```
+----------+------------------------------+
|   Option | Description                  |
+==========+==============================+
|        1 | Get Appointment Details by ID |
+----------+------------------------------+
|        2 | Schedule Appointment         |
+----------+------------------------------+
|        3 | Update Appointment           |
+----------+------------------------------+
|        4 | Cancel Appointment           |
+----------+------------------------------+
|        5 | Get Appointments for Patient  |
+----------+------------------------------+
|        6 | Get Appointments for Doctor   |
+----------+------------------------------+
|        7 | Exit                         |
+----------+------------------------------+

Enter your choice: 1

Enter appointment ID: 11

***** Appointment not found!!!!! *****
```

```
+----------+------------------------------+
|   Option | Description                  |
+==========+==============================+
|        1 | Get Appointment Details by ID |
+----------+------------------------------+
|        2 | Schedule Appointment         |
+----------+------------------------------+
|        3 | Update Appointment           |
+----------+------------------------------+
|        4 | Cancel Appointment           |
+----------+------------------------------+
|        5 | Get Appointments for Patient  |
+----------+------------------------------+
|        6 | Get Appointments for Doctor   |
+----------+------------------------------+
|        7 | Exit                         |
+----------+------------------------------+

Enter your choice: 2
Enter patient ID: 1
Enter doctor ID: D002
Enter appointment date (YYYY-MM-DD HH:MM): 2024-11-11 11:11
Enter appointment description: Cardio Treatment

************************************
Appointment scheduled successfully!

************************************
```

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	1	D001	2024-10-09 10:30:00.000	Dental checkup
2	2	2	D002	2024-10-09 14:00:00.000	Heart checkup
3	3	3	D003	2024-10-10 09:00:00.000	Back pain consultation
4	4	1	D002	2024-11-11 11:11:00.000	Cardio Treatment

```
+----------+------------------------------+
|   Option | Description                  |
+==========+==============================+
|        1 | Get Appointment Details by ID |
+----------+------------------------------+
|        2 | Schedule Appointment         |
+----------+------------------------------+
|        3 | Update Appointment           |
+----------+------------------------------+
|        4 | Cancel Appointment           |
+----------+------------------------------+
|        5 | Get Appointments for Patient  |
+----------+------------------------------+
|        6 | Get Appointments for Doctor   |
+----------+------------------------------+
|        7 | Exit                         |
+----------+------------------------------+

Enter your choice: 3

Enter appointment ID to update: 5
Enter new patient ID: 3
Enter new doctor ID: D002
Enter new appointment date (YYYY-MM-DD HH:MM): 2024-11-11 11:11
Enter new appointment description: Heart Transplant Surgery

****************************************
        Appointment updated successfully.

****************************************
```

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	1	D001	2024-10-09 10:30:00.000	Dental checkup
2	2	2	D002	2024-10-09 14:00:00.000	Heart checkup
3	3	3	D003	2024-10-10 09:00:00.000	Back pain consultation
4	4	2	D001	2024-10-29 01:01:00.000	Stomach Operation
5	5	3	D002	2024-11-11 11:11:00.000	Heart Transplant Sur...

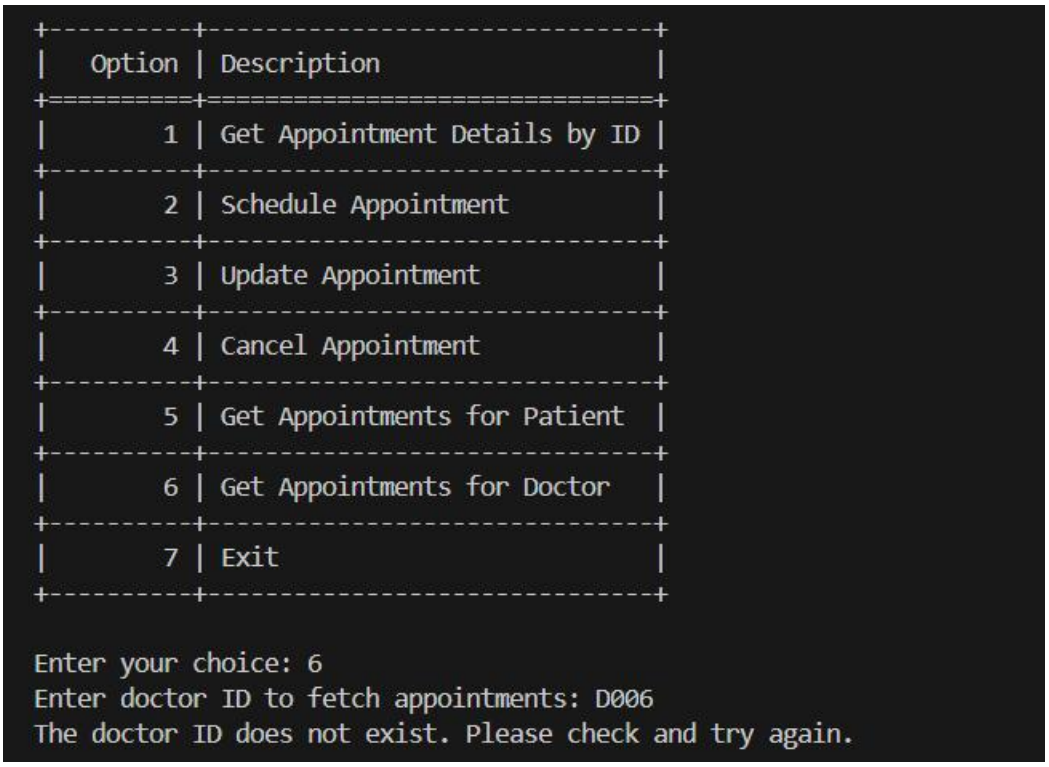


	appointmentId	patientId	doctorId	appointmentDate	description
1	1	1	D001	2024-10-09 10:30:00.000	Dental checkup
2	2	2	D002	2024-10-09 14:00:00.000	Heart checkup
3	3	3	D003	2024-10-10 09:00:00.000	Back pain consultation
4	5	3	D002	2024-11-11 11:11:00.000	Heart Transplant Surgery

```
+---------+------------------------------+
|  Option | Description                  |
+=========+==============================+
|       1 | Get Appointment Details by ID |
+---------+------------------------------+
|       2 | Schedule Appointment         |
+---------+------------------------------+
|       3 | Update Appointment           |
+---------+------------------------------+
|       4 | Cancel Appointment           |
+---------+------------------------------+
|       5 | Get Appointments for Patient |
+---------+------------------------------+
|       6 | Get Appointments for Doctor  |
+---------+------------------------------+
|       7 | Exit                         |
+---------+------------------------------+

Enter your choice: 5
Enter patient ID to fetch appointments: 3

Appointments for patient ID: 3

+----------------+------------+--------------------+--------------------------+
|  Appointment ID | Doctor ID  | Appointment Date   | Description              |
+================+============+====================+==========================+
|              3 | D003       | 2024-10-10 09:00:00 | Back pain consultation  |
+----------------+------------+--------------------+--------------------------+
|              5 | D002       | 2024-11-11 11:11:00 | Heart Transplant Surgery |
+----------------+------------+--------------------+--------------------------+
```

```
+----------+------------------------------+
|  Option | Description                  |
+==========+==============================+
|       1 | Get Appointment Details by ID |
+----------+------------------------------+
|       2 | Schedule Appointment         |
+----------+------------------------------+
|       3 | Update Appointment           |
+----------+------------------------------+
|       4 | Cancel Appointment           |
+----------+------------------------------+
|       5 | Get Appointments for Patient |
+----------+------------------------------+
|       6 | Get Appointments for Doctor  |
+----------+------------------------------+
|       7 | Exit                         |
+----------+------------------------------+

Enter your choice: 5
Enter patient ID to fetch appointments: 6

------No appointments found for patient ID: 6------
```

```
+----------+------------------------------+
|   Option | Description                  |
+==========+==============================+
|        1 | Get Appointment Details by ID |
+----------+------------------------------+
|        2 | Schedule Appointment         |
+----------+------------------------------+
|        3 | Update Appointment           |
+----------+------------------------------+
|        4 | Cancel Appointment           |
+----------+------------------------------+
|        5 | Get Appointments for Patient  |
+----------+------------------------------+
|        6 | Get Appointments for Doctor   |
+----------+------------------------------+
|        7 | Exit                         |
+----------+------------------------------+

Enter your choice: 6
Enter doctor ID to fetch appointments: D002

Appointments for doctor ID: D002

+------------------+-------------+---------------------+--------------------------+
|   Appointment ID |   Patient ID | Appointment Date    | Description              |
+==================+=============+=====================+==========================+
|                2 |           2 | 2024-10-09 14:00:00 | Heart checkup            |
+------------------+-------------+---------------------+--------------------------+
|                5 |           3 | 2024-11-11 11:11:00 | Heart Transplant Surgery |
+------------------+-------------+---------------------+--------------------------+
```


+----------+------------------------------+
| Option | Description |
+==========+==============================+
| 1 | Get Appointment Details by ID |
+----------+------------------------------+
| 2 | Schedule Appointment |
+----------+------------------------------+
| 3 | Update Appointment |
+----------+------------------------------+
| 4 | Cancel Appointment |
+----------+------------------------------+
| 5 | Get Appointments for Patient |
+----------+------------------------------+
| 6 | Get Appointments for Doctor |
+----------+------------------------------+
| 7 | Exit |
+----------+------------------------------+
Enter your choice: 6
Enter doctor ID to fetch appointments: D006
The doctor ID does not exist. Please check and try again.


+----------+------------------------------+
| Option | Description |
+==========+==============================+
| 1 | Get Appointment Details by ID |
+----------+------------------------------+
| 2 | Schedule Appointment |
+----------+------------------------------+
| 3 | Update Appointment |
+----------+------------------------------+
| 4 | Cancel Appointment |
+----------+------------------------------+
| 5 | Get Appointments for Patient |
+----------+------------------------------+
| 6 | Get Appointments for Doctor |
+----------+------------------------------+
| 7 | Exit |
+----------+------------------------------+
Enter your choice: 7
Exiting the system.