Coding Challenge: Hospital Management System **BALAKUMARAN P**

• Project submissions should be done through the partcipants' Github repository and the link should be shared with trainers and Hexavarsity.

• Follow object-oriented principles throughout the project. Use classes and objects to model realworld entities,encapsulate data and behavior, and ensure code reusability.

• Throw user defined exceptions from corresponding methods and handled.

• The following Directory structure is to be followed in the application.

o entity

▪ Create entity classes in this package. All entity class should not have any business logic.

o dao

▪ Create Service Provider interface to showcase functionalities.

▪ Create the implementation class for the above interface with db interaction.

o exception

▪ Create user defined exceptions in this package and handle exceptions whenever needed.

o util

▪ Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.

▪ Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).

o main

▪ Create a class MainModule and demonstrate the functionalities in a menu driven application.

Problem Statement:

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

Creating Patient Table:

```
mysql> CREATE TABLE Patient (
    ->     patientId INT PRIMARY KEY,
    ->     firstName VARCHAR(255),
    ->     lastName VARCHAR(255),
    ->     dateOfBirth DATE,
    ->     gender VARCHAR(10),
    ->     contactNumber VARCHAR(15),
    ->     address TEXT
    -> );
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> DESC Patient;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| patientId     | int          | NO   | PRI | NULL    |       |
| firstName     | varchar(255) | YES  |     | NULL    |       |
| lastName      | varchar(255) | YES  |     | NULL    |       |
| dateOfBirth   | date         | YES  |     | NULL    |       |
| gender        | varchar(10)  | YES  |     | NULL    |       |
| contactNumber | varchar(15)  | YES  |     | NULL    |       |
| address       | text         | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
7 rows in set (0.02 sec)
```

Creating Doctor Table:

```
mysql> CREATE TABLE Doctor (
    ->     doctorId INT PRIMARY KEY,
    ->     firstName VARCHAR(255),
    ->     lastName VARCHAR(255),
    ->     specialization VARCHAR(255),
    ->     contactNumber VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> DESC Doctor;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| doctorId       | int          | NO   | PRI | NULL    |       |
| firstName      | varchar(255) | YES  |     | NULL    |       |
| lastName       | varchar(255) | YES  |     | NULL    |       |
| specialization | varchar(255) | YES  |     | NULL    |       |
| contactNumber  | varchar(15)  | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Creating Appointment Table:

```
mysql> CREATE TABLE Appointment (
    ->     appointmentId INT PRIMARY KEY,
    ->     patientId INT,
    ->     doctorId INT,
    ->     appointmentDate DATE,
    ->     description TEXT,
    ->     FOREIGN KEY (patientId) REFERENCES Patient(patientId),
    ->     FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> DESC Appointment;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| appointmentId   | int  | NO   | PRI | NULL    |       |
| patientId       | int  | YES  | MUL | NULL    |       |
| doctorId        | int  | YES  | MUL | NULL    |       |
| appointmentDate | date | YES  |     | NULL    |       |
| description     | text | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

2. 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;

```
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

    def print_details(self):
        print(f"Patient ID: {self.patientId}")
        print(f"First Name: {self.firstName}")
        print(f"Last Name: {self.lastName}")
        print(f"Date of Birth: {self.dateOfBirth}")
        print(f"Gender: {self.gender}")
        print(f"Contact Number: {self.contactNumber}")
        print(f"Address: {self.address}")
```

2. Define 'Doctor` class with the following confidential attributes:

a. doctorId b. firstName c. lastName d. specialization e. contactNumber;

```
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

    def print_details(self):
        print(f"Doctor ID: {self.doctorId}")
        print(f"First Name: {self.firstName}")
```

```
        print(f"Last Name: {self.lastName}")
        print(f"Specialization: {self.specialization}")
        print(f"Contact Number: {self.contactNumber}")
```

3. Appointment Class:

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

```
class Appointment:
    def __init__(self, appointmentId=None, patientId=None, doctorId=None,
appointmentDate=None,
                 description=None):
        self.appointmentId = appointmentId
        self.patientId = patientId
        self.doctorId = doctorId
        self.appointmentDate = appointmentDate
        self.description = description

    def print_details(self):
        print(f"Appointment ID: {self.appointmentId}")
        print(f"Patient ID: {self.patientId}")
        print(f"Doctor ID: {self.doctorId}")
        print(f"Appointment Date: {self.appointmentDate}")
        print(f"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.

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
class IHospitalService(ABC):

    @abstractmethod
    def get_appointment_by_id(self, appointment_id):
        pass

    @abstractmethod
    def generate_appointment_id(self):
        pass

    @abstractmethod
    def get_appointments_for_patient(self, patient_id):
        pass

    @abstractmethod
    def get_appointments_for_doctor(self, doctor_id):
        pass

    @abstractmethod
    def schedule_appointment(self, appointment_id):
        pass

    @abstractmethod
    def update_appointment(self, appointment_id):
        pass

    @abstractmethod
    def cancel_appointment(self, appointment_id):
        pass
```

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

```
from dao.service.IHospitalService import IHospitalService
import mysql.connector

from myexceptions.exception import PatientNumberNotFoundException


class HospitalServiceImpl(IHospitalService):
    def __init__(self, database_con):
        self.database_con = database_con

    def generate_appointment_id(self):
        connection = self.database_con

        cur = connection.cursor()
        cur.execute("SELECT MAX(AppointmentID) FROM Appointment")
        appointment_id = cur.fetchone()[0]
        if appointment_id is None:
            appointment_id = 1
        else:
            appointment_id += 1
        return appointment_id

    def get_appointment_by_id(self, appointment_id):
        connection = self.database_con
        appointment_data = None
        if connection:
            try:
                cursor = connection.cursor(dictionary=True)
                cursor.execute("SELECT * FROM Appointment WHERE appointmentId =
%s", (appointment_id,))
                appointment_data = cursor.fetchone()

            except mysql.connector.Error as err:
                print(f"Error: {err}")
            return appointment_data

    def get_appointments_for_patient(self, patient_id):
        connection = self.database_con
        if connection:
            try:
```

```
                cursor = connection.cursor(dictionary=True)
                cursor.execute("SELECT * FROM Appointment WHERE patientID =
%s", (patient_id,))
                appointment_data = cursor.fetchone()
            except mysql.connector.Error as err:
                print(f"Error: {err}")
                raise

            if not appointment_data:
                raise PatientNumberNotFoundException(f"Patient with ID
{patient_id} not found.")

            return appointment_data

    def get_appointments_for_doctor(self, doctor_id):
        connection = self.database_con
        appointment_data = None
        if connection:
            try:
                cursor = connection.cursor(dictionary=True)
                cursor.execute("SELECT * FROM Appointment WHERE doctorID = %s",
(doctor_id,))
                appointment_data = cursor.fetchone()
            except mysql.connector.Error as err:
                print(f"Error: {err}")
            return appointment_data

    def schedule_appointment(self, appointment_data):
        connection = self.database_con
        appointment_id = self.generate_appointment_id()

        try:
            cursor = connection.cursor()
            cursor.execute("INSERT INTO Appointment VALUES (%s,%s,%s,%s,%s)",
                           (appointment_id, appointment_data['patientId'],
appointment_data['doctorId'],
                            appointment_data['appointmentDate'],
appointment_data['description'],))
            connection.commit()
            print("Appointment scheduled successfully!")
        except Exception as e:
```

```
            print(f"Error scheduling appointment: {e}")

    def update_appointment(self, appointment):
        appointment_id=appointment.get('AppointmentID')
        patientId = appointment.get('patientId')
        doctorId = appointment.get('doctorId')
        appointmentDate = appointment.get('appointmentDate')
        description = appointment.get('description')

        connection=self.database_con
        try:
            cursor=connection.cursor()
            cursor.execute('''
                        UPDATE Appointment
                        SET patientId = %s,
                            doctorId = %s,
                            appointmentDate = %s,
                            description = %s
                        WHERE appointmentID = %s
                        ''',

(patientId,doctorId,appointmentDate,description,appointment_id))
            connection.commit()
        except Exception as e:
            print(e)


    def cancel_appointment(self, appointment_id):
        connection=self.database_con
        try:
            cursor=connection.cursor()
            cursor.execute("DELETE FROM Appointment WHERE appointmentID =
%s",(appointment_id,))
            connection.commit()
        except Exception as e:
            print(e)
```

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 fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

```
class PropertyUtil:
    @staticmethod
    def get_property_string(connection_details):
        connection_string = {
            'host': connection_details['host'],
            'user': connection_details['user'],
            'passwd': connection_details['passwd'],
            'port': connection_details['port'],
            'database': connection_details['dbname']
        }

        return connection_string
```

```
import mysql.connector
from util.PropertyUtil import PropertyUtil

class DBConnection:
    connection = None

    @staticmethod
    def get_connection(connection_details):
        if DBConnection.connection is None:
            connection_string =
PropertyUtil.get_property_string(connection_details)
            DBConnection.connection =
mysql.connector.connect(**connection_string)

        return DBConnection.connection
```

8. Create the exceptions in package myexceptions

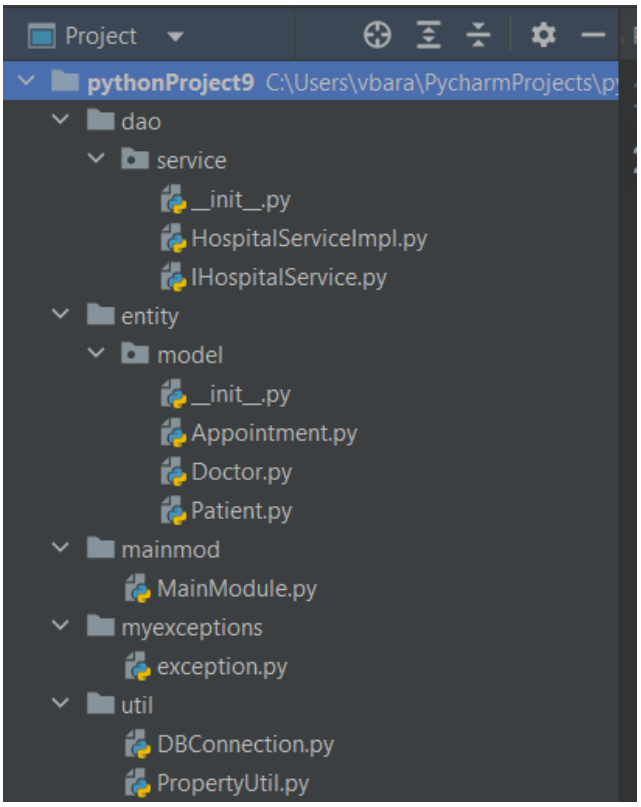
Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. PatientNumberNotFoundException :throw this exception when user enters an invalid patient

number which doesn't exist in db

```
class PatientNumberNotFoundException(Exception):
    pass
```

My Project Structure:



Inserting 10 sample datas into Doctor and Patient tables so as to perform the implementation:

```
mysql> INSERT INTO Doctor (doctorId, firstName, lastName, specialization, contactNumber)
    -> VALUES
    ->     (2, 'Deepa', 'Shankar', 'Orthopedic Surgeon', '8765432109'),
    ->     (3, 'Priya', 'Raj', 'Gynecologist', '7654321098'),
    ->     (4, 'Rajesh', 'Mohan', 'Neurologist', '6543210987'),
    ->     (5, 'Anitha', 'Kumar', 'Pediatrician', '5432109876'),
    ->     (6, 'Senthil', 'Devi', 'Dermatologist', '4321098765'),
    ->     (7, 'Vijay', 'Lakshmi', 'ENT Specialist', '3210987654'),
    ->     (8, 'Malathi', 'Venkatesh', 'Ophthalmologist', '2109876543'),
    ->     (9, 'Ganesh', 'Priya', 'Urologist', '1098765432'),
    ->     (10, 'Meena', 'Suresh', 'Radiologist', '9876543210');
Query OK, 9 rows affected (0.02 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from Doctor;
+----------+-------------+-----------+--------------------+---------------+
| doctorId | firstName   | lastName  | specialization     | contactNumber |
+----------+-------------+-----------+--------------------+---------------+
|        1 | Balakumaran | P         | Neuro Surgeon      | 89246723      |
|        2 | Deepa       | Shankar   | Orthopedic Surgeon | 8765432109    |
|        3 | Priya       | Raj       | Gynecologist       | 7654321098    |
|        4 | Rajesh      | Mohan     | Neurologist        | 6543210987    |
|        5 | Anitha      | Kumar     | Pediatrician       | 5432109876    |
|        6 | Senthil     | Devi      | Dermatologist      | 4321098765    |
|        7 | Vijay       | Lakshmi   | ENT Specialist     | 3210987654    |
|        8 | Malathi     | Venkatesh | Ophthalmologist    | 2109876543    |
|        9 | Ganesh      | Priya     | Urologist          | 1098765432    |
|       10 | Meena       | Suresh    | Radiologist        | 9876543210    |
+----------+-------------+-----------+--------------------+---------------+
10 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Patient (patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address)
    -> VALUES
    ->     (2, 'Priya', 'Kumar', '1985-08-22', 'Female', '8765432109', 'Coimbatore'),
    ->     (3, 'Mohan', 'Devi', '1992-03-10', 'Male', '7654321098', 'Madurai'),
    ->     (4, 'Lakshmi', 'Ganesh', '1988-11-28', 'Female', '6543210987', 'Salem'),
    ->     (5, 'Karthik', 'Anand', '1995-06-18', 'Male', '5432109876', 'Trichy'),
    ->     (6, 'Deepa', 'Rajesh', '1980-09-05', 'Female', '4321098765', 'Vellore'),
    ->     (7, 'Raj', 'Malathi', '1998-01-30', 'Male', '3210987654', 'Tirunelveli'),
    ->     (8, 'Saranya', 'Suresh', '1983-07-12', 'Female', '2109876543', 'Kanyakumari'),
    ->     (9, 'Prakash', 'Meena', '1993-04-25', 'Male', '1098765432', 'Erode'),
    ->     (10, 'Aishwarya', 'Venkatesh', '1987-12-08', 'Female', '9876543210', 'Thanjavur');
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from patient;
+-----------+-----------+-----------+-------------+--------+---------------+-------------+
| patientId | firstName | lastName  | dateOfBirth | gender | contactNumber | address     |
+-----------+-----------+-----------+-------------+--------+---------------+-------------+
|         1 | Bala      | P         | 2002-02-02  | MALE   | 89246723      | Pondicherry |
|         2 | Priya     | Kumar     | 1985-08-22  | Female | 8765432109    | Coimbatore  |
|         3 | Mohan     | Devi      | 1992-03-10  | Male   | 7654321098    | Madurai     |
|         4 | Lakshmi   | Ganesh    | 1988-11-28  | Female | 6543210987    | Salem       |
|         5 | Karthik   | Anand     | 1995-06-18  | Male   | 5432109876    | Trichy      |
|         6 | Deepa     | Rajesh    | 1980-09-05  | Female | 4321098765    | Vellore     |
|         7 | Raj       | Malathi   | 1998-01-30  | Male   | 3210987654    | Tirunelveli |
|         8 | Saranya   | Suresh    | 1983-07-12  | Female | 2109876543    | Kanyakumari |
|         9 | Prakash   | Meena     | 1993-04-25  | Male   | 1098765432    | Erode       |
|        10 | Aishwarya | Venkatesh | 1987-12-08  | Female | 9876543210    | Thanjavur   |
+-----------+-----------+-----------+-------------+--------+---------------+-------------+
10 rows in set (0.00 sec)
```

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

```python
from datetime import date


from dao.service.IHospitalService import IHospitalService
from dao.service.HospitalServiceImpl import HospitalServiceImpl
from myexceptions.exception import PatientNumberNotFoundException
from util.DBConnection import DBConnection


from util.PropertyUtil import PropertyUtil


def main():


    connection_details = {
        'host': 'localhost',
        'user': 'root',
        'passwd': 'root',
        'port': '3306',
        'dbname': 'HospitalManagementSystem'
    }


    db_connection = DBConnection.get_connection(connection_details)
    hospital_service = HospitalServiceImpl(db_connection)
    print("Hospital Management System :")
    while True:
```

```
        print('''
                1 ---> Get Appointment Details By appointment ID
                2 ---> Get Appointment for Patient
                3 ---> Get Appointment for Doctor
                4 ---> Schedule Appointment
                5 ---> Update Appointment
                6 ---> Cancel Appointment
                7 ---> Exit

             ''')
        choice = input('Enter Your choice(1-5): ')

        try:
            if choice == '1':
                appointment_id=int(input("Enter your appointment ID: "))

appointments=hospital_service.get_appointment_by_id(appointment_id)
                print("Appointment ID: ",appointments['appointmentId'])
                print("Patient ID: ", appointments['patientId'])
                print("Doctor ID: ", appointments['doctorId'])
                print("Appointment Date: ", appointments['appointmentDate'])
                print("Description: ", appointments['description'])

            if choice == '2':
                patient_id=input("Enter your patient ID: ")

appointments=hospital_service.get_appointments_for_patient(patient_id)
                print("Appointment ID: ", appointments['appointmentId'])
                print("Patient ID: ", appointments['patientId'])
                print("Doctor ID: ", appointments['doctorId'])
                print("Appointment Date: ", appointments['appointmentDate'])
                print("Description: ", appointments['description'])
            if choice == '3':
                doctor_id=input("Enter your Doctor ID: ")

appointments=hospital_service.get_appointments_for_patient(doctor_id)
                print("Appointment ID: ", appointments['appointmentId'])
                print("Patient ID: ", appointments['patientId'])
                print("Doctor ID: ", appointments['doctorId'])
                print("Appointment Date: ", appointments['appointmentDate'])
                print("Description: ", appointments['description'])
```

```
            if choice == '4':
                new_appointment = {
                    'patientId': input("Enter patient ID: "),
                    'doctorId': input("Enter doctor ID: "),
                    'appointmentDate': date.today(),
                    'description': input("Enter Description if any: ")
                }
                hospital_service.schedule_appointment(new_appointment)
                print("Appointment confirmed")

            if choice == '5':
                new_appointment = {
                    'patientId': input("Enter patient ID: "),
                    'doctorId': input("Enter doctor ID: "),
                    'appointmentDate': input("Enter the appointment date: "),
                    'description': input("Enter Description if any to change:
")
                }
                hospital_service.update_appointment(new_appointment)
                print("Appointment Updated Successfully....")

            if choice == '6':
                appointment_id=int(input("Enter your appointment ID: "))
                hospital_service.cancel_appointment(appointment_id)
                print("Your appointment has been cancelled successfully...")
            if choice == '7':
                print("Exiting....")
                break
        except PatientNumberNotFoundException as e:
            print(f"Patient Number Not Found: {e}")
        except Exception as e:
            print(f"An unexpected error occurred: {e}")

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

Implementation:

Scheduling appointment for the patient with patient ID 3 to the doctor with doctor ID 5 on Current Date.

```
Hospital Management System :

                1 ---> Get Appointment Details By appointment ID
                2 ---> Get Appointment for Patient
                3 ---> Get Appointment for Doctor
                4 ---> Schedule Appointment
                5 ---> Update Appointment
                6 ---> Cancel Appointment
                7 ---> Exit
```

```
Enter Your choice(1-5): 4
Enter patient ID: 3
Enter doctor ID: 5
Enter Description if any: Carry Water Bottles while coming
Appointment scheduled successfully!
Appointment confirmed
```

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+---------------------------------+
| appointmentId | patientId | doctorId | appointmentDate | description                     |
+---------------+-----------+----------+-----------------+---------------------------------+
|             2 |         1 |        1 | 2024-02-06      | nil                             |
|             3 |         3 |        5 | 2024-02-06      | Carry Water Bottles while coming |
+---------------+-----------+----------+-----------------+---------------------------------+
2 rows in set (0.00 sec)
```

Getting the appointment details by giving appointment ID as input:

```
Enter Your choice(1-5): 1
Enter your appointment ID: 3
Appointment ID:  3
Patient ID:  3
Doctor ID:  5
Appointment Date:  2024-02-06
Description:  Carry Water Bottles while coming
```

Getting appointment details for patient by giving patient ID as input:

```
Enter Your choice(1-5): 2
Enter your patient ID: 2
Appointment ID:  4
Patient ID:  2
Doctor ID:  7
Appointment Date:  2024-02-06
Description:  Eat and come
```

Getting appointment details for doctor by giving Doctor ID as input:

```
Enter Your choice(1-5): 3
Enter your Doctor ID: 7
Appointment ID:  4
Patient ID:  2
Doctor ID:  7
Appointment Date:  2024-02-06
Description:  Eat and come
```

Updating Appointment with appointment ID 2:

```
Enter Your choice(1-5): 5
Enter appointment ID: 2
Enter patient ID: 1
Enter doctor ID: 1
Enter the appointment date: 2023-07-07
Enter Description if any to change: No changes
Appointment Updated Successfully....
```

Before Update:

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+---------------------------------+
| appointmentId | patientId | doctorId | appointmentDate | description                     |
+---------------+-----------+----------+-----------------+---------------------------------+
|             2 |         1 |        1 | 2024-02-06      | nil                             |
|             3 |         3 |        5 | 2024-02-06      | Carry Water Bottles while coming |
|             4 |         2 |        7 | 2024-02-06      | Eat and come                    |
+---------------+-----------+----------+-----------------+---------------------------------+
3 rows in set (0.00 sec)
```

After Update:

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+---------------------------------+
| appointmentId | patientId | doctorId | appointmentDate | description                     |
+---------------+-----------+----------+-----------------+---------------------------------+
|             2 |         1 |        1 | 2023-07-07      | No changes                      |
|             3 |         3 |        5 | 2024-02-06      | Carry Water Bottles while coming |
|             4 |         2 |        7 | 2024-02-06      | Eat and come                    |
+---------------+-----------+----------+-----------------+---------------------------------+
3 rows in set (0.00 sec)
```

Cancelling an particular appointment:

```
Enter Your choice(1-5): 6
Enter your appointment ID: 3
Your appointment has been cancelled successfully...
```

After cancelling it just removes the record from the database:

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+--------------+
| appointmentId | patientId | doctorId | appointmentDate | description  |
+---------------+-----------+----------+-----------------+--------------+
|             2 |         1 |        1 | 2023-07-07      | No changes   |
|             4 |         2 |        7 | 2024-02-06      | Eat and come |
+---------------+-----------+----------+-----------------+--------------+
2 rows in set (0.00 sec)
```