

Coding Challenge

Hospital Mangement

V. Chandana Priya Reddy

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

```
create DATABASE Hospital_Management;
```

```
use Hospital_Management;
```

```
CREATE TABLE Patient(  
    patientId varchar(5) PRIMARY KEY,  
    firstName VARCHAR(50) NOT NULL,  
    lastName VARCHAR(50) NOT NULL,  
    dateOfBirth DATE NOT NULL,  
    gender VARCHAR(10) NOT NULL,  
    contactNumber VARCHAR(15) NOT NULL,  
    address VARCHAR(255) NOT NULL);
```

```
CREATE TABLE Doctor (  
    doctorId varchar(5) PRIMARY KEY,  
    firstName VARCHAR(50) NOT NULL,  
    lastName VARCHAR(50) NOT NULL,  
    specialization VARCHAR(50) NOT NULL,  
    contactNumber VARCHAR(15) NOT NULL);
```

```
create table Appointment(  
    appointmentId int primary key,  
    patientId varchar(5) not null,  
    doctorId varchar(5) not null,  
    appointmentDate DATETIME not null,  
    description varchar(50),  
    FOREIGN KEY(patientId) References Patient(patientId),  
    FOREIGN KEY (doctorId) References Doctor(DoctorId)  
);
```

Insert into Patient(patientId, firstName,lastName,dateOfBirth,gender,contactNumber,address)

VALUES

('P1', 'Charitha', 'Curie', '1972-09-27', 'Female', '9897969576', 'Kadapa'),
('P2', 'Nandu', 'Dev', '2002-01-12', 'Male', '9876543210', 'Chennai'),
('P3', 'Setha', 'Ramu', '1985-04-15', 'Female', '9444555667', 'Allahabad'),
('P4', 'Tony', 'Stark', '2015-05-29', 'Male', '7778889999', 'Hyderabad'),
('P5', 'Natasha', 'Romanoff', '1984-11-22', 'Female', '6667778888', 'Mumbai'),
('P6', 'Steve', 'Rogers', '1999-07-04', 'Male', '5556667797', 'Kurnool'),
('P7', 'Bruce', 'Wayne', '2019-02-19', 'Male', '4678654390', 'Indore'),
('P8', 'Selena', 'Gomez', '2020-07-22', 'Female', '3334445567', 'Chennai'),
('P9', 'John', 'Doe', '2001-03-17', 'Male', '2223334484', 'Kurnool'),
('P10', 'Emily', 'Chen', '2005-10-01', 'Female', '1112223333', 'Kadapa');

INSERT into Doctor(doctorId,firstName,lastName,specialization,contactNumber)

Values

('D01', 'Chandu', 'Stark', 'Cardiologist', '9123456780'),
('D02', 'Priya', 'Singh', 'Neurologist', '4329087654'),
('D03', 'Teju', 'Rathode', 'Surgeon', '98766546543'),
('D04', 'Aayushi', 'Gupta', 'Pediatrician', '8769806547'),
('D05', 'Bruce', 'Banner', 'Dermatologist', '9876543210'),
('D06', 'Sathak', 'Kulkarni', 'Oncologist', '8907654762'),
('D07', 'Vikas', 'Reddy', 'Dermatologist', '5764789432'),
('D08', 'Kushi', 'Joshi', 'Rhumetologist', '9876867869'),
('D09', 'Harthik', 'Pandey', 'Gastroenterologist', '8769806598'),
('D10', 'Harsha', 'Gupta', 'Endocrinologist', '7869087651');

Insert into appointment(appointmentId,patientId,doctorId,appointmentDate,description)

Values

(1,'P10','D07','2024-10-28 10:30','Hair loss'),
(10,'P7','D08','2024-10-13 11:00','Arthritis'),
(2,'P8','D09','2024-11-02 10:30','Stomach Ache'),
(3,'P1','D10','2024-10-17 9:30','diabetes'),
(4,'P3','D03','2024-10-11 12:00','Surgery'),

```
(5,'P4','D07','2024-10-29 11:30','Hair loss'),  
(6,'P5','D02','2024-11-03 10:30','Migrane'),  
(7,'P6','D01','2024-11-01 11:00','Hyper Tension'),  
(8,'P9','D07','2024-10-30 12:00','Hair loss'),  
(9,'P2','D04','2024-10-12 12:00','Allergy');
```

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

entity/patient.py

class Patient:

```
def __init__(self, patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address):  
    self.patientId = patientId  
    self.firstName = firstName  
    self.lastName = lastName  
    self.dateOfBirth = dateOfBirth  
    self.gender = gender  
    self.contactNumber = contactNumber  
    self.address = address
```

#setter methods

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

```
def set_firstName(self,firstName):
    self.firstName = firstName
def set_lastName(self,lastName):
    self.lastName = lastName
def set_dateOfBirth(self,dateOfBirth):
    self.dateOfBirth = dateOfBirth
def set_gender(self,gender):
    self.gender = gender
def set_contactNumber(self,contactNumber):
    self.contactNumber = contactNumber
def set_address(self,address):
    self.address = address
```

#getter methods

```
def get_patientId(self):
    return self.patientId
def get_firstName(self):
    return self.firstName
def get_lastName(self):
    return self.lastName
def get_dateOfBirth(self):
    return self.dateOfBirth
def get_gender(self):
    return self.gender
def get_contactNumber(self):
    return self.contactNumber
def get_address(self):
    return self.address

def __str__(self):
    return f"Patient ID: {self.patientId()}, Name: {self.firstName} {self.lastName}, " \
           f"DOB: {self.dateOfBirth}, Gender: {self.gender}, Contact: {self.contactNumber}, " \
           f"Address: {self.address}"
```

2. Define **Doctor** class with the following confidential attributes:

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

entity/doctor.py

class Doctor:

```
def __init__(self,doctorId,firstName,lastName,specialization,contactNumber):
```

```
    self.doctorId = doctorId
```

```
    self.firstName = firstName
```

```
    self.lastName = lastName
```

```
    self.specialization = specialization
```

```
    self.contactNumber = contactNumber
```

#Setter methods

```
def set_doctorId(self,doctorId):
```

```
    self.doctorId = doctorId
```

```
def set_firstName(self,firstName):
```

```
    self.firstName = firstName
```

```
def set_lastName(self,lastName):
```

```
    self.lastName = lastName
```

```
def set_specialization(self,specialization):
```

```
    self.specialization = specialization
```

```
def set_contactNumber(self,contactNumber):
```

```
    self.contactNumber = contactNumber
```

#Getter methods

```
def get_doctorId(self):
```

```
    return self.doctorId
```

```
def get_firstName(self):
```

```

        return self.firstName
def get_lastName(self):
    return self.lastName
def get_specialization(self):
    return self.specialization
def get_contactNumber(self):
    return self.contactNumber

def __str__(self):
    return f"Doctor ID: {self.doctorId}, Name: {self.firstName} {self.lastName}, "\
        f"Specialization: {self.specialization}, Contact: {self.contactNumber}"

```

3. Appointment Class:

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

entity/appointment.py

class Appointment:

```

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

```

#Setter methods

```

def set_appointmentId(self,appointmentId):

```

```

        self.appointmentId = appointmentId
def set_patientId(self,patientId):
    self.patientId = patientId
def set_doctorId(self,doctorId):
    self.doctorId = doctorId
def set_appointmentDate(self,appointmentDate):
    self.appointmentDate = appointmentDate
def set_description(self,description):
    self.description = description

#Getter methods

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

```

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

dao/IHospitalService.py

```
from abc import ABC, abstractmethod
```

```
from entity.appointment import Appointment
```

```
from typing import List
```

```
class IHospitalService(ABC):
```

```
    @abstractmethod
```

```
    def getAppointmentById(self, appointmentId) -> Appointment:
```

```
        pass
```

```
    @abstractmethod
```

```
    def getAppointmentsForPatient(self, patientId) -> List[Appointment]:
```

```
        pass
```

```
    @abstractmethod
```

```
    def getAppointmentsForDoctor(self, doctorId) -> List[Appointment]:
```

```
        pass
```



```
@abstractmethod
```

```
def scheduleAppointment(self, appointment: Appointment) -> bool:
```

```
    pass
```

```
@abstractmethod
```

```
def updateAppointment(self, appointment: Appointment) -> bool:
```

```
    pass
```

```
@abstractmethod
```

```
def cancelAppointment(self, appointmentId) -> bool:
```

```
    pass
```

Define HospitalServiceImpl class and implement all the methods IHospitalServiceImpl

dao/HospitalServiceImpl.py

```
from dao.IHospitalService import IHospitalService
```

```
from entity.appointment import Appointment
```

```
from exception.PatientNumberNotFound import PatientNumberNotFoundException
```

```
from util.DBConnection import DBConnection
```

```
from tabulate import tabulate
```

```
class HospitalServiceImpl(IHospitalService):
```

```
    def getAppointmentById(self, appointmentId):
```

```
        conn = DBConnection.getConnection()
```

```
        cursor=conn.cursor()
```

```
        try:
```

```
            query = "SELECT * FROM Appointment WHERE appointmentId = ?"
```

```
            cursor.execute(query, (appointmentId,))
```

```
            appointment = cursor.fetchone()
```

```
        if appointment:
```

```
            appointment_details=[
```

```

        ['Appointment ID',appointment[0]],
        ["Patient ID",appointment[1]],
        ["Doctor ID",appointment[2]],
        ["Appointment Date",appointment[3]],
        ["Description",appointment[4]],
    ]
    print("-----Appointment Details----- ")
    print(tabulate(appointment_details,tablefmt="grid"))

else:
    print("-----Appointment Not Found-----")
except Exception as e:
    print(f"Error in fetching appointment: {e}")
    return None
finally:
    cursor.close()

def getAppointmentsForPatient(self, patientId):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        query = "SELECT * FROM Appointment WHERE patientId = ?"
        cursor.execute(query,(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 PatientNumberNotFoundException as e:
        print(e)
        return []
    finally:
        cursor.close()

def getAppointmentsForDoctor(self, doctorId):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        query = "SELECT * FROM Appointment WHERE doctorId = ?"
        cursor.execute(query, (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"Error in fetching appointments for doctor: {e}")
        return []

def doctor_exists(self, doctorId):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        query = "SELECT count(*) FROM Doctor WHERE doctorId = ?"
        cursor.execute(query, (doctorId,))
        count = cursor.fetchone()[0]
        return count > 0 #If doctor exists
    except Exception as e:

```

```
    print(f"Error in doctor exists: {e}")
    return False
finally:
    cursor.close()
```

```
def patient_exists(self, patientId):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        query = "SELECT count(*) FROM Patient WHERE patientId = ?"
        cursor.execute(query, (patientId,))
        count = cursor.fetchone()[0]
        return count > 0
    except Exception as e:
        print(f"Error in patient exists: {e}")
        return False
    finally:
        cursor.close()
```

```
def get_next_appointmentId(self):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        cursor.execute("SELECT Max(appointmentId) FROM Appointment")
        max_id = cursor.fetchone()[0]
        return (max_id+1) if max_id is not None else 1
    except Exception as e:
        print(f"Error in get_next_appointment: {e}")
        return 1
    finally:
        cursor.close()
```

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

```

conn = DBConnection.getConnection()
cursor=conn.cursor()
try:
    cursor.execute("SELECT COUNT(*) FROM Patient where
patientId=?", (appointment.get_patientId(),))
    patient_exists = cursor.fetchone()[0]
    if not patient_exists:
        print(f"Patient ID {appointment.get_patientId()} does not exist")
        return False

    cursor.execute("Select Count(*) From Doctor where
doctorId=?", (appointment.get_doctorId(),))
    doctor_exists = cursor.fetchone()[0]
    if not doctor_exists:
        print(f"Doctor ID {appointment.get_doctorId()} does not exist")
        return False

    cursor.execute("INSERT INTO Appointment (appointmentId, patientId, doctorId,
appointmentDate, description) VALUES (?, ?, ?, ?, ?)",
    (appointment.get_appointmentId(), appointment.get_patientId(),
appointment.get_doctorId(), appointment.get_appointmentDate(), appointment.get_description()))
    conn.commit()
    print("Appointment Scheduled")
    return True
except Exception as e:
    print(f"Error in scheduleAppointment: {e}")
def updateAppointment(self, appointment):
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
    try:
        cursor.execute("Select count(*) from Appointment where
appointmentId=?", (appointment.appointmentId,))
        count = cursor.fetchone()[0]
        if not count:

```

```
print("-----Appointment Not Found-----")
return False
```

```
cursor.execute("UPDATE Appointment SET patientId=?,doctorId=?, appointmentDate = ?,
description = ? WHERE appointmentId =
?",(appointment.get_patientId(),appointment.get_doctorId(),appointment.get_appointmentDate(),a
ppointment.get_description(),appointment.appointmentId))
conn.commit()
return True
except Exception as e:
    print(f"Error in updateAppointment: {e}")
    return False
finally:
    cursor.close()
```

```
def cancelAppointment(self, appointmentId):
```

```
    conn = DBConnection.getConnection()
    cursor=conn.cursor()
```

```
    try:
```

```
        cursor.execute("Select count(*) from Appointment where
appointmentId=?", (appointmentId,))
```

```
        count = cursor.fetchone()[0]
```

```
        if count==0:
```

```
            print("-----Appointment Not Found-----")
            return False
```

```
        cursor.execute("DELETE FROM Appointment WHERE appointmentId = ?",(appointmentId,))
```

```
        conn.commit()
```

```
        return True
```

```
    except Exception as e:
```

```
        print(f"Error in cancelAppointment: {e}")
        return False
```

```
    finally:
```

```
        cursor.close()
```

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

util/DBConnection.py

```
import pyodbc
from util.PropertyUtil import PropertyUtil

class DBConnection:

    @staticmethod
    def getConnection():
        try:
            properties=PropertyUtil.getPropertyString()
            connection=pyodbc.connect(**properties)
            cursor=connection.cursor()
            return connection
        except Exception as e:
            print(str(e) + '--Database is not connected--')
            return None
```

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

util/propertyFile.txt

```
Driver={SQL Server}
host=Chandana\SQLEXPRESS
database=Hospital_Management
user=Chandana
password=Chandana03
port=1433
```

util/PropertyUtil.py

```
class PropertyUtil:
    @staticmethod
    def
getPropertyString(property_file_path='C://Users//chand//PycharmProjects//HospitalManagement//
util//propertyFile.txt'):
    try:
        with open(property_file_path, 'r') as file:
            properties = {}
            for line in file:
                key, value = line.strip().split('=')
                properties[key.strip()] = value.strip()
            return properties
    except Exception as e:
        print(f"Error reading property file: {e}")
        return None
```

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

exception/PatientNumberNotFoundException.py

```
class PatientNumberNotFoundException(Exception):
    def __init__(self,patientId):
        super().__init__(f"Patient with Id {patientId} not found.")
```

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

main/main.py

```
from dao.HospitalServiceImpl import HospitalServiceImpl
from entity.appointment import Appointment
```



```
from tabulate import tabulate
```

```
class MainModule:
```

```
    def __init__(self):
```

```
        self.hospital_service = HospitalServiceImpl()
```

```
    def menu(self):
```

```
        global appointment
```

```
        print("*" * 40)
```

```
        print("Welcome to Hospital Management System")
```

```
        print("*" * 40)
```

```
        while True:
```

```
            menu = [
```

```
                ["1.", "Get Appointment Details by ID"],
```

```
                ["2.", "Get Appointments for Patient"],
```

```
                ["3.", "Get Appointments for Doctor"],
```

```
                ["4.", "Schedule Appointment"],
```

```
                ["5.", "Update Appointment"],
```

```
                ["6.", "Cancel Appointment"],
```

```
                ["7.", "Exit"]
```

```
            ]
```

```
            # Print the menu using tabulate
```

```
            print(tabulate(menu, headers=["Option", "Description"], tablefmt="grid"))
```

```
            choice = input("Enter your choice: ")
```

```
            if choice == '1':
```

```
                appointment_id=int(input("Enter Appointment ID: "))
```

```
                try:
```

```
                    appointment=self.hospital_service.getAppointmentById(appointment_id)
```

```
                    if appointment is None:
```

```
                        print("")
```

```
                except ValueError :
```

```
                    print("Invalid input. Please enter a valid number.")
```

```

except Exception as e:
    print(f"Error: {e}")
elif choice == '2':
    patient_id=input("Enter Patient ID to fetch appointment: ")
    appointments=self.hospital_service.getAppointmentsForPatient(patient_id)
    if appointments:
        print(f"Appointments for Patient: {patient_id}")
        table_data = [[appointment.appointmentId, appointment.doctorId,
appointment.appointmentDate,
                        appointment.description]
                        for appointment in appointments]
        headers = ["Appointment Id", "Doctor Id", "Appointment Date", "Appointment
Description"]
        print(tabulate(table_data, headers=headers, tablefmt="grid"))
    else :
        print("Patient not found")

elif choice == '3':
    doctor_id=input("Enter Doctor ID to fetch appointments: ")
    if not self.hospital_service.doctor_exists(doctor_id):
        print(f"*****The doctor Id {doctor_id} doesnot exists*****")
        continue
    appointments=self.hospital_service.getAppointmentsForDoctor(doctor_id)
    if appointments:
        print(f"Appointments for Doctor: {doctor_id}")
        table_data = [[appointment.appointmentId, appointment.patientId,
appointment.appointmentDate,
                        appointment.description]
                        for appointment in appointments]
        headers=["Appointment Id", "Patient Id", "Appointment Date", "Appointment
Description"]
        print(tabulate(table_data, headers=headers, tablefmt="grid"))
    else:
        print(f"-----No appointments for Doctor {doctor_id}-----")

```

```

elif choice == '4':
    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: ")
    appointment_id=self.hospital_service.get_next_appointmentId()
    if appointment_id is None:
        print("Failed to get next appointment.")
    else:
        appointment = Appointment(
            appointmentId=appointment_id, # Use the generated ID
            patientId=patient_id,
            doctorId=doctor_id,
            appointmentDate=appointment_date,
            description=description
        )
    if self.hospital_service.scheduleAppointment(appointment):
        print("Appointment scheduled successfully.")
    else:
        print("Unable to schedule appointment.")

elif choice == '5':
    appointment_id=int(input("Enter Appointment ID to update: "))
    new_patient_id=input("Enter New Patient ID: ")
    if not self.hospital_service.patient_exists(new_patient_id):
        print("The specified patient Id does not exist.")
        continue
    new_doctor_id=input("Enter New Doctor ID: ")
    new_appointment_date=input("Enter New Appointment Date(YYYY-MM-DD HH:MM): ")
    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.hospital_service.updateAppointment(appointment):
        print("-----Appointment updated successfully-----")
    else:
        print("Unable to update appointment.")
elif choice == '6':
    appointment_id=int(input("Enter Appointment ID to cancel: "))
    if self.hospital_service.cancelAppointment(appointment_id):
        print("Appointment cancelled successfully.")
    else:
        print("Appointment id does not exist")
elif choice == '7':
    print("Exiting the system")
    break
else:
    print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main_module = MainModule()
    main_module.menu()

```

Outputs:

```
select * from Patient;
```

	patientId	firstName	lastName	dateOfBirth	gender	contactNumber	address
1	P1	Charitha	Curie	1972-09-27	Female	9897969576	Kadapa
2	P10	Emily	Chen	2005-10-01	Female	1112223333	Kadapa
3	P2	Nandu	Dev	2002-01-12	Male	9876543210	Chennai
4	P3	Setha	Ramu	1985-04-15	Female	9444555667	Allahabad
5	P4	Tony	Stark	2015-05-29	Male	7778889999	Hyderabad
6	P5	Natasha	Romanoff	1984-11-22	Female	6667778888	Mumbai
7	P6	Steve	Rogers	1999-07-04	Male	5556667797	Kumool
8	P7	Bruce	Wayne	2019-02-19	Male	4678654390	Indore
9	P8	Selena	Gomez	2020-07-22	Female	3334445567	Chennai
10	P9	John	Doe	2001-03-17	Male	2223334484	Kumool

```
select * from Doctor;
```

	doctorId	firstName	lastName	specialization	contactNumber
1	D01	Chandu	Stark	Cardiologist	9123456780
2	D02	Priya	Singh	Neurologist	4329087654
3	D03	Teju	rathode	Surgeon	98766546543
4	D04	Aayushi	Gupta	Pediatrician	8769806547
5	D05	Bruce	Banner	Dermatologist	9876543210
6	D06	Sathak	Kulkarni	Oncologist	8907654762
7	D07	Vikas	Reddy	Dermatologist	5764789432
8	D08	Kushi	Joshi	Rhumetologist	9876867869
9	D09	Harthik	Pandey	Gastroenterologist	8769806598
10	D10	Harsha	Gupta	Endocrinologist	7869087651

```
select * from Appointment;
```

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	P10	D07	2024-10-28 10:30:00.000	Hair loss
2	2	P8	D09	2024-11-02 10:30:00.000	Stomach Ache
3	3	P1	D10	2024-10-17 09:30:00.000	diabetes
4	4	P3	D03	2024-10-11 12:00:00.000	Surgery
5	5	P4	D07	2024-10-29 11:30:00.000	Hair loss
6	6	P5	D02	2024-11-03 10:30:00.000	Migrane
7	7	P6	D01	2024-11-01 11:00:00.000	Hyper Tension
8	8	P9	D07	2024-10-30 12:00:00.000	Hair loss
9	9	P2	D04	2024-10-12 12:00:00.000	Allergy
10	10	P7	D08	2024-10-13 11:00:00.000	Arthritis

Welcome to Hospital Management System

+-----+-----+-----+-----+-----+				
	Option		Description	
+=====+=====+=====+=====+=====+				
	1		Get Appointment Details 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: 3

-----Appointment Details-----

+-----+-----+-----+-----+-----+				
	Appointment ID		3	
+-----+-----+-----+-----+-----+				
	Patient ID		P1	
+-----+-----+-----+-----+-----+				
	Doctor ID		D10	
+-----+-----+-----+-----+-----+				
	Appointment Date		2024-10-17 09:30:00	
+-----+-----+-----+-----+-----+				
	Description		diabetes	
+-----+-----+-----+-----+-----+				

```

+-----+-----+
| Option | Description |
+-----+-----+
|      1 | Get Appointment Details 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: 12
-----Appointment Not Found-----

```

```

+-----+-----+
| Option | Description |
+-----+-----+
|      1 | Get Appointment Details 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 to fetch appointment: P3
Appointments for Patient: P3
+-----+-----+-----+-----+
| Appointment Id | Doctor Id | Appointment Date | Appointment Description |
+-----+-----+-----+-----+
|      4 | D03 | 2024-10-11 12:00:00 | Surgery |
+-----+-----+-----+-----+

```

```

+-----+-----+
| Option | Description |
+=====+=====+
|      1 | Get Appointment Details 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 to fetch appointment: P11
Patient not found

```

```

+-----+-----+
| Option | Description |
+=====+=====+
|      1 | Get Appointment Details 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 to fetch appointments: D11
*****The doctor Id D11 doesnot exists*****

```



```
+-----+-----+
| Option | Description |
+-----+-----+
| 1 | Get Appointment Details 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 to fetch appointments: D07
Appointments for Doctor: D07
+-----+-----+-----+-----+
| Appointment Id | Patient Id | Appointment Date | Appointment Description |
+-----+-----+-----+-----+
| 1 | P10 | 2024-10-28 10:30:00 | Hair loss |
+-----+-----+-----+-----+
| 5 | P4 | 2024-10-29 11:30:00 | Hair loss |
+-----+-----+-----+-----+
| 8 | P9 | 2024-10-30 12:00:00 | Hair loss |
+-----+-----+-----+-----+
```

```
+-----+
| Option | Description |
+=====+
|      1 | Get Appointment Details 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: 4
Enter Patient Id: P3
Enter Doctor ID: D07
Enter Appointment Date(YYYY-MM-DD HH:MM): 2024-11-03 12:30
Enter Appointment Description: Hair loss
Appointment Scheduled
Appointment scheduled successfully.
```

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	P10	D07	2024-10-28 10:30:00.000	Hair loss
2	2	P8	D09	2024-11-02 10:30:00.000	Stomach Ache
3	3	P1	D10	2024-10-17 09:30:00.000	diabetes
4	4	P3	D03	2024-10-11 12:00:00.000	Surgery
5	5	P4	D07	2024-10-29 11:30:00.000	Hair loss
6	6	P5	D02	2024-11-03 10:30:00.000	Migrane
7	7	P6	D01	2024-11-01 11:00:00.000	Hyper Tension
8	8	P9	D07	2024-10-30 12:00:00.000	Hair loss
9	9	P2	D04	2024-10-12 12:00:00.000	Allergy
10	10	P7	D08	2024-10-13 11:00:00.000	Arthritis
11	11	P3	D07	2024-11-03 12:30:00.000	Hair loss

```

+-----+-----+
| Option | Description |
+=====+=====+
|      1 | Get Appointment Details 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: 4
Enter Patient Id: P11
Enter Doctor ID: D09
Enter Appointment Date(YYYY-MM-DD HH:MM): 2024-11-06
Enter Appointment Description: Stomach Ache
Patient ID P11 does not exist
Unable to schedule appointment.

```

```

+-----+-----+
| Option | Description |
+=====+=====+
|      1 | Get Appointment Details 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 to update: 11
Enter New Patient ID: P11
The specified patient Id does not exist.

```

```
+-----+-----+
| Option | Description |
+-----+-----+
|      1 | Get Appointment Details 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 to update: 5
Enter New Patient ID: P10
Enter New Doctor ID: D08
Enter New Appointment Date(YYYY-MM-DD HH:MM): 2024-11-02 12:30
Enter New Appointment Description: Arthritis
-----Appointment updated successfully-----
```

Results		Messages			
	appointmentId	patientId	doctorId	appointmentDate	description
1	1	P10	D07	2024-10-28 10:30:00.000	Hair loss
2	2	P8	D09	2024-11-02 10:30:00.000	Stomach Ache
3	3	P1	D10	2024-10-17 09:30:00.000	diabetes
4	5	P10	D08	2024-11-02 12:30:00.000	Arthritis
5	6	P5	D02	2024-11-03 10:30:00.000	Migrane
6	7	P6	D01	2024-11-01 11:00:00.000	Hyper Tension
7	8	P9	D07	2024-10-30 12:00:00.000	Hair loss
8	9	P2	D04	2024-10-12 12:00:00.000	Allergy
9	10	P7	D08	2024-10-13 11:00:00.000	Arthritis
10	11	P3	D07	2024-11-03 12:30:00.000	Hair loss

```

+-----+-----+
| Option | Description |
+=====+=====+
|      1 | Get Appointment Details 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: 11
Appointment cancelled successfully.

```

	appointmentId	patientId	doctorId	appointmentDate	description
1	1	P10	D07	2024-10-28 10:30:00.000	Hair loss
2	2	P8	D09	2024-11-02 10:30:00.000	Stomach Ache
3	3	P1	D10	2024-10-17 09:30:00.000	diabetes
4	5	P10	D08	2024-11-02 12:30:00.000	Arthritis
5	6	P5	D02	2024-11-03 10:30:00.000	Migrane
6	7	P6	D01	2024-11-01 11:00:00.000	Hyper Tension
7	8	P9	D07	2024-10-30 12:00:00.000	Hair loss
8	9	P2	D04	2024-10-12 12:00:00.000	Allergy
9	10	P7	D08	2024-10-13 11:00:00.000	Arthritis

```

+-----+-----+
| Option | Description |
+-----+-----+
|      1 | Get Appointment Details 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: 12
-----Appointment Not Found-----

```

```

+-----+-----+
| Option | Description |
+-----+-----+
|      1 | Get Appointment Details 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
Existing the system

```