HOSPITAL MANAGEMENT

By Sara Pricilla A

TABLE OF CONTENTS

1.INTRODUCTION

In the rapidly evolving landscape of healthcare, efficient digital solutions are essential. The Hospital Management System is a simplified and modular application designed to manage patient appointments, doctors, and scheduling in a clinical environment. Built using Python and PyMySQL, this system demonstrates core software development practices such as object-oriented programming, layeredarchitecture, and exception handling.

2.OBJECTIVE OF THE PROJECT

The main objectives of this project are:

- To automate appointment booking and management between doctors and patients.
- To implement robust backend interaction using MySQL.
- To showcase clean architecture using DAO, entity, util, and exception layers.To enforce security and validation through admin and user-level access.

3.OVERVIEW OF THE PROJECT

This system manages the following:

- Patients: Personal details like name, contact, DOB, etc.
- Doctors: Basic professional and personal details.
- Appointments: Scheduled meetings between patients and doctors.
- The system is menu-driven and performs operations like scheduling, viewing, updating, and canceling appointments.

4.SCOPE OF THE PROJECT

- Used in clinics, diagnostic centers, and small hospitals.
- Helps in record-keeping and minimizes manual effort.
- Easily extendable to include billing, prescriptions, and test reports in the future.

5.PURPOSE OF THE PROJECT

The purpose is to provide a simple yet functional interface to manage hospital data effectively while demonstrating:

- Real-time database connectivity.
- Role-based access (Admin, Doctor, Patient).
- Clean separation of concerns via packages.

6.SOFTWARE REQUIREMENTS

Component	Details
Programming Language	Python 3.x
Database	MySQL
Connectivity	PyMySQL
IDE	VS Code / PyCharm
Property Config	`.properties` file for DB
Operating System	Windows/Linux

7.IMPLEMENTATION

Directory Structure:

- entity: Classes for Patient, Doctor, Appointment.
- dao: Interface HospitalService and its implementation HospitalServiceImpl.
- exception: Custom exceptions like PatientNumberNotFoundException, AppointmentNotFoundException.
- util: Database connectivity using DBConnection and PropertyUtil.
- main.py: Menu-driven execution with login validation and role-based access.

Functional Features:

- Schedule, View, Update, and Cancel Appointments.
- Admin login required for sensitive operations.
- Uses MySQL to persist and retrieve data.
- Custom exceptions enhance error reporting and debugging.

8.DATABASE SCHEMA:

```
create database hospital_management;
use hospital_management;
create table patient (
    patientid int primary key auto_increment,
    firstname varchar(50) not null,
    lastname varchar(50) not null,
    dateofbirth date not null,
    gender enum('male', 'female', 'other') not null,
    contactnumber varchar(15),
    address varchar(255)
);
create table doctor (
    doctorid int primary key auto_increment,
    firstname varchar(50) not null,
    lastname varchar(50) not null,
```

```
    specialization varchar(100),
    contactnumber varchar(15)
);
create table appointment (
    appointmentid int primary key auto_increment,
    patientid int,
    doctorid int,
    appointmentdate date,
    description text,
    foreign key (patientid) references patient(patientid),
    foreign key (doctorid) references doctor(doctorid)
);
```

```
insert into patient (firstname, lastname, dateofbirth, gender, contactnumber,
address) values
('arun', 'kumar', '1994-05-12', 'male', '9876543210', 'chennai, tamil nadu'),
('divya', 'raj', '1990-11-23', 'female', '9871234567', 'madurai, tamil nadu'),
('karthik', 'murugan', '1996-08-03', 'male', '9865432109', 'coimbatore, tamil nadu'),
('meena', 'sundar', '1992-04-18', 'female', '9856123478', 'trichy, tamil nadu'),
('vignesh', 'ram', '1998-01-30', 'male', '9845012345', 'salem, tamil nadu'),
('revathi', 'bala', '1995-07-14', 'female', '9834567890', 'vellore, tamil nadu'),
('sathish', 'vel', '1993-09-27', 'male', '9823012345', 'erode, tamil nadu');
```

```
mysql> select * from patient;
+-----------+-----------+----------+-------------+--------+---------------+------------------------+
| patientid | firstname | lastname | dateofbirth | gender | contactnumber | address                |
+-----------+-----------+----------+-------------+--------+---------------+------------------------+
|         1 | arun      | kumar    | 1994-05-12  | male   | 9876543210    | chennai, tamil nadu    |
|         2 | divya     | raj      | 1990-11-23  | female | 9871234567    | madurai, tamil nadu    |
|         3 | karthik   | murugan  | 1996-08-03  | male   | 9865432109    | coimbatore, tamil nadu |
|         4 | meena     | sundar   | 1992-04-18  | female | 9856123478    | trichy, tamil nadu     |
|         5 | vignesh   | ram      | 1998-01-30  | male   | 9845012345    | salem, tamil nadu      |
|         6 | revathi   | bala     | 1995-07-14  | female | 9834567890    | vellore, tamil nadu    |
|         7 | sathish   | vel      | 1993-09-27  | male   | 9823012345    | erode, tamil nadu      |
+-----------+-----------+----------+-------------+--------+---------------+------------------------+
7 rows in set (0.19 sec)
```

```
insert into doctor (firstname, lastname, specialization, contactnumber) values
('suresh', 'raman', 'cardiology', '9111223344'),
('lakshmi', 'venkat', 'pediatrics', '9123456780'),
('ganesh', 'mani', 'dermatology', '9234567890'),
('usha', 'selvam', 'neurology', '9345678901'),
('mohan', 'krishnan', 'orthopedics', '9456789012'),
('deepa', 'shankar', 'gynecology', '9567890123'),
```

('rajesh', 'kumar', 'general medicine', '9678901234');

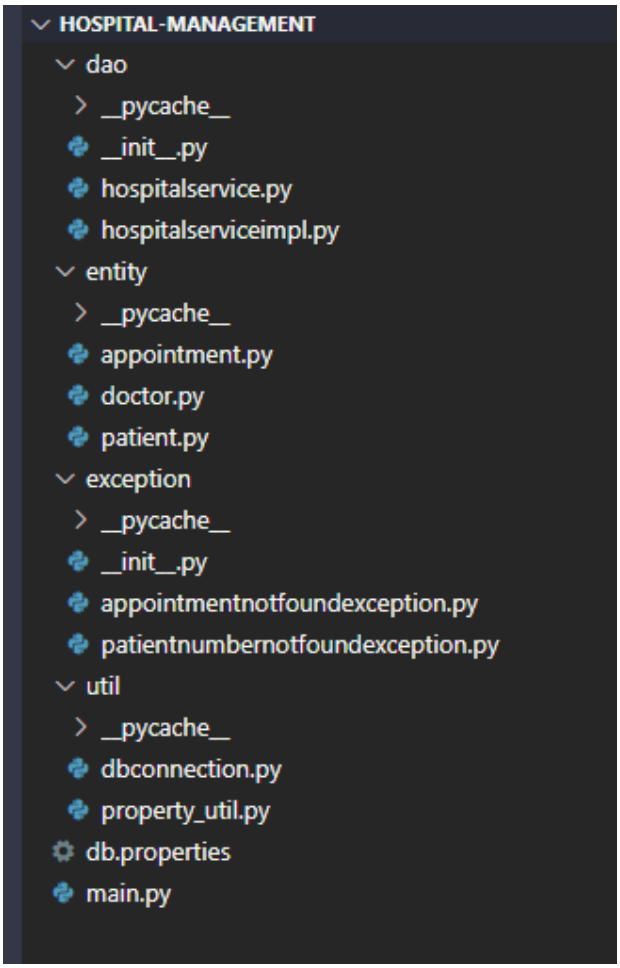
```
mysql> select * from doctor;
+----------+-----------+----------+------------------+---------------+
| doctorid | firstname | lastname | specialization   | contactnumber |
+----------+-----------+----------+------------------+---------------+
|        1 | suresh    | raman    | cardiology       | 9111223344    |
|        2 | lakshmi   | venkat   | pediatrics       | 9123456780    |
|        3 | ganesh    | mani     | dermatology      | 9234567890    |
|        4 | usha      | selvam   | neurology        | 9345678901    |
|        5 | mohan     | krishnan | orthopedics      | 9456789012    |
|        6 | deepa     | shankar  | gynecology       | 9567890123    |
|        7 | rajesh    | kumar    | general medicine | 9678901234    |
+----------+-----------+----------+------------------+---------------+
7 rows in set (0.00 sec)
```

insert into appointment (patientid, doctorid, appointmentdate, description) values
(1, 1, '2025-04-10', 'routine heart check-up'),
(2, 2, '2025-04-12', 'child vaccination'),
(3, 3, '2025-04-13', 'skin rash treatment'),
(4, 4, '2025-04-14', 'headache consultation'),
(5, 5, '2025-04-15', 'joint pain follow-up'),
(6, 6, '2025-04-16', 'pregnancy scan'),
(7, 7, '2025-04-17', 'fever and cough');

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+-----------------------+
| appointmentid | patientid | doctorid | appointmentdate | description           |
+---------------+-----------+----------+-----------------+-----------------------+
|             1 |         1 |        1 | 2025-04-10      | routine heart check-up |
|             2 |         2 |        3 | 2025-04-12      | child vaccination     |
|             3 |         3 |        3 | 2025-04-13      | skin rash treatment   |
|             4 |         4 |        2 | 2027-09-05      | headache consultation |
|             5 |         5 |        6 | 2025-04-15      | joint pain follow-up  |
|             6 |         6 |        6 | 2023-09-08      | pregnancy scan        |
|             7 |         7 |        7 | 2025-04-17      | fever and cough       |
|             8 |         1 |        3 | 2026-07-07      | pulmonary problem     |
|            10 |         4 |        5 | 2025-05-04      | follow-up-visit       |
+---------------+-----------+----------+-----------------+-----------------------+
9 rows in set (0.00 sec)
```

9.CODING PART

Project Directory:



Appointment.py:

```
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

    def set_appointment_id(self, appointment_id):
        self.__appointment_id = appointment_id

    def get_patient_id(self):
        return self.__patient_id

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

    def get_doctor_id(self):
        return self.__doctor_id

    def set_doctor_id(self, doctor_id):
        self.__doctor_id = doctor_id

    def get_appointment_date(self):
        return self.__appointment_date

    def set_appointment_date(self, appointment_date):
        self.__appointment_date = appointment_date

    def get_description(self):
        return self.__description

    def set_description(self, description):
        self.__description = description

    def __str__(self):
        return f"Appointment(appointment_id={self.__appointment_id},
patient_id={self.__patient_id}, " \
            f"doctor_id={self.__doctor_id},
appointment_date='{self.__appointment_date}', " \
            f"description='{self.__description}')"
```

Doctor.py:

```
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

    def set_doctor_id(self, doctor_id):
        self.__doctor_id = doctor_id

    def get_first_name(self):
        return self.__first_name

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def get_last_name(self):
        return self.__last_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def get_specialization(self):
        return self.__specialization

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

    def get_contact_number(self):
```

```
        return self.__contact_number

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number

    def __str__(self):
        return f"Doctor(doctor_id={self.__doctor_id},
first_name='{self.__first_name}', last_name='{self.__last_name}', " \
                f"specialization='{self.__specialization}',
contact_number='{self.__contact_number}')"
```

Patient.py

```
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

    def get_first_name(self):
        return self.__first_name

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def get_last_name(self):
        return self.__last_name
```

```
    def set_last_name(self, last_name):
        self.__last_name = last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

    def get_gender(self):
        return self.__gender

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

    def get_contact_number(self):
        return self.__contact_number

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number

    def get_address(self):
        return self.__address

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

    def __str__(self):
        return f"Patient(patient_id={self.__patient_id},
first_name='{self.__first_name}', last_name='{self.__last_name}', " \
                f"date_of_birth='{self.__date_of_birth}', gender='{self.__gender}', " \
                f"contact_number='{self.__contact_number}',
address='{self.__address}')"
```

patientnumbernotfoundexception.py:

```
class PatientNumberNotFoundException(Exception):
    def __init__(self, message="Patient ID not found in database"):
```

```
        super().__init__(message)
```

appointmentnotfoundexception.py:

```
class AppointmentNotFoundException(Exception):
    def __init__(self, message="Appointment not found."):
        super().__init__(message)
```

dbconnection.py:

```
import pymysql
import sys
from util.property_util import PropertyUtil

class DBConnection:
    connection = None

    @staticmethod
    def get_connection():
        if DBConnection.connection is None:
            try:
                props = PropertyUtil.get_property_string('db.properties')
                DBConnection.connection = pymysql.connect(
                    host=props['host'],
                    user=props['user'],
                    password=props['password'],
                    database=props['database'],
                    port=props['port']
                )
            except pymysql.MySQLError as e:
                print(f"Database Connection Error: {e}")
                sys.exit(1)
        return DBConnection.connection
```

property_util.py:

```
import configparser

class PropertyUtil:
    @staticmethod
```

```
def get_property_string(filename):
    config = configparser.ConfigParser()
    config.read(filename)

    host = config.get('database', 'host')
    dbname = config.get('database', 'dbname')
    user = config.get('database', 'user')
    password = config.get('database', 'password')
    port = config.get('database', 'port')

    return {
        'host': host,
        'user': user,
        'password': password,
        'database': dbname,
        'port': int(port)
    }
```

Db.properties:

```
[database]
host = 127.0.0.1
dbname = hospitalmanagement
user = root
password = root
port = 3306
```

hospitalservice.py:

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

class IHospitalService(ABC):

    @abstractmethod
    def get_appointment_by_id(self, appointment_id: int) -> Appointment:
        pass

    @abstractmethod
```

```
    def get_appointments_for_patient(self, patient_id: int):
        pass

    @abstractmethod
    def get_appointments_for_doctor(self, doctor_id: int):
        pass

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

    @abstractmethod
    def update_appointment_doctor(self, appointment_id: int, new_doctor_id: int) -
> bool:
        pass

    @abstractmethod
    def update_appointment_date(self, appointment_id: int, new_date: str) -> bool:
        pass

    @abstractmethod
    def cancel_appointment(self, appointment_id: int) -> bool:
        pass
```

hospitalserviceimpl.py:

```
from dao.hospitalservice import IHospitalService
from entity.appointment import Appointment
from entity.patient import Patient
from entity.doctor import Doctor
from exception.patientnumbernotfoundexception import
PatientNumberNotFoundException
from exception.appointmentnotfoundexception import
AppointmentNotFoundException
from util.dbconnection import DBConnection

class HospitalServiceImpl(IHospitalService):

    def __init__(self):
```

```
        self.conn = DBConnection.get_connection()
        self.cursor = self.conn.cursor()
        self._setup_table()

    def _setup_table(self):
        self.cursor.execute('''CREATE TABLE IF NOT EXISTS appointment (
                                appointment_id INTEGER PRIMARY KEY,
                                patient_id INTEGER,
                                doctor_id INTEGER,
                                appointment_date TEXT,
                                description TEXT)''')
        self.conn.commit()

    def get_appointment_by_id(self, appointment_id: int) -> Appointment:
        query = "SELECT appointmentid, patientid, doctorid, appointmentdate,
description FROM appointment WHERE appointmentid = %s"
        self.cursor.execute(query, (appointment_id,))
        row = self.cursor.fetchone()

        if row:
            appointment = Appointment(*row)
            print("\n□  Appointment Details")

print("────────────────────────────────────────────────────────────────────────")
            print("□ Appointment ID   : ", appointment.get_appointment_id())
            print("□□ Patient ID       : ", appointment.get_patient_id())
            print("□□ Doctor ID        : ", appointment.get_doctor_id())
            print("□ Date             : ", appointment.get_appointment_date())
            print("□ Description      : ", appointment.get_description())

print("────────────────────────────────────────────────────────────────────────")

            return appointment
        else:
            print("No appointment found with the given ID.")
            return None
```

```
C:\Users\admin\Downloads\hospital-management>python main.py

---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): 1
Enter admin username: admin
Enter admin password: admin123

---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 2
Enter Appointment ID: 4

📋  Appointment Details
────────────────────────────────────────
🆔 Appointment ID   :  4
👤 Patient ID       :  4
👨‍⚕️ Doctor ID        :  4
📅 Date             :  2025-04-14
📝 Description      :  headache consultation
────────────────────────────────────────
Appointment(appointment_id=4, patient_id=4, doctor_id=4, appointment_date='2025-04-14', description='headache consultation')
```

```
def get_appointments_for_patient(self, patient_id: int):
    try:
        query = """
            SELECT
                a.appointmentid, a.patientid, a.doctorid, a.appointmentdate,
a.description,
                p.firstname, p.lastname, p.address
            FROM
                appointment a
            JOIN
                patient p ON a.patientid = p.patientid
            WHERE
                a.patientid = %s
        """
        self.cursor.execute(query, (patient_id,))
        rows = self.cursor.fetchall()

        if not rows:
            self.cursor.execute("SELECT firstname, lastname, address FROM
patient WHERE patientid = %s", (patient_id,))
            patient_info = self.cursor.fetchone()
            if patient_info:
                patient = Patient(patient_id, patient_info[0], patient_info[1], None,
None, None, patient_info[2])
```

```
                    print(f"\nHello {patient.get_first_name()} {patient.get_last_name()}
from {patient.get_address()}, you have no appointments.")
                else:
                    raise PatientNumberNotFoundException(f"No patient found with ID:
{patient_id}")
                return []

            # Build patient object from first row
            patient = Patient(patient_id, rows[0][5], rows[0][6], None, None, None,
rows[0][7])
            print(f"\nHello {patient.get_first_name()} {patient.get_last_name()} from
{patient.get_address()}, you have {len(rows)} appointment(s).")

            appointments = []

            for row in rows:
                appointment = Appointment(row[0], row[1], row[2], row[3], row[4])
                print("\n□  Appointment Details")

print("────────────────────────────────────────────────────────────
──")
                print("□ Appointment ID   : ", appointment.get_appointment_id())
                print("□□ Patient ID      : ", appointment.get_patient_id())
                print("□♂□ Patient Name    : ", patient.get_first_name(),
patient.get_last_name())
                print("□□ Doctor ID       : ", appointment.get_doctor_id())
                print("□ Appointment Date : ", appointment.get_appointment_date())
                print("□ Description      : ", appointment.get_description())

print("────────────────────────────────────────────────────────────
──")

                appointments.append(appointment)

            return appointments

        except PatientNumberNotFoundException as e:
            print("Error:", e)
            return []
```

```
    except Exception as e:
        print("An unexpected error occurred:", e)
        return []
```

```
---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 3
Enter Patient ID: 1

Hello arun kumar from chennai, tamil nadu, you have 2 appointment(s).

📋  Appointment Details
──────────────────────────────────────
🆔 Appointment ID   :  1
👤 Patient ID       :  1
👤 Patient Name     :  arun kumar
👨‍⚕️ Doctor ID        :  1
📅 Appointment Date :  2025-04-10
📝 Description      :  routine heart check-up
──────────────────────────────────────

📋  Appointment Details
──────────────────────────────────────
🆔 Appointment ID   :  8
👤 Patient ID       :  1
👤 Patient Name     :  arun kumar
👨‍⚕️ Doctor ID        :  3
📅 Appointment Date :  2026-07-07
📝 Description      :  pulmonary problem
──────────────────────────────────────
```

```
def get_appointments_for_doctor(self, doctor_id: int):
    try:
        query = """
            SELECT
                a.appointmentid, a.patientid, a.doctorid, a.appointmentdate,
a.description,
                p.firstname, p.lastname,
                d.firstname, d.lastname
            FROM
                appointment a
            JOIN
                patient p ON a.patientid = p.patientid
```

```
            JOIN
                doctor d ON a.doctorid = d.doctorid
            WHERE
                a.doctorid = %s
        """
        self.cursor.execute(query, (doctor_id,))
        rows = self.cursor.fetchall()

        if not rows:
            # Check if doctor exists
            self.cursor.execute("SELECT firstname, lastname FROM doctor
WHERE doctorid = %s", (doctor_id,))
            doctor = self.cursor.fetchone()
            if doctor:
                print(f"\nHello Dr. {doctor[0]} {doctor[1]}, you have no appointments
scheduled.")
            else:
                raise PatientNumberNotFoundException(f"No doctor found with ID:
{doctor_id}")
            return []

        # Extract doctor name
        doctor_fname = rows[0][7]
        doctor_lname = rows[0][8]
        print(f"\nHello Dr. {doctor_fname} {doctor_lname}, you have {len(rows)}
appointment(s) to undertake:")

        appointments = []

        for row in rows:
            appointment = Appointment(row[0], row[1], row[2], row[3], row[4])
            patient_fname = row[5]
            patient_lname = row[6]

            print("\n□  Appointment Details")

print("────────────────────────────────────────────────────────────────────────────────")
            print("□ Appointment ID   : ", appointment.get_appointment_id())
            print("□♂□ Patient Name     : ", patient_fname, patient_lname)
```

```
            print("□□ Doctor Name      : ", doctor_fname, doctor_lname)
            print("□ Appointment Date : ", appointment.get_appointment_date())
            print("□ Description      : ", appointment.get_description())

print("────────────────────────────────────────────────────────────────────────────────")

            appointments.append(appointment)

        return appointments

    except PatientNumberNotFoundException as e:
        print("Error:", e)
        return []

    except Exception as e:
        print("\nAn unexpected error occurred while retrieving doctor's appointments:", str(e))
        return []
```

```
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 4
Enter Doctor ID: 2

Hello Dr. lakshmi venkat, you have no appointments scheduled.

---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 4
Enter Doctor ID: 5

Hello Dr. mohan krishnan, you have 1 appointment(s) to undertake:

📋  Appointment Details
──────────────────────────────────────────
🆔 Appointment ID   :  10
👤 Patient Name     :  meena sundar
👨‍⚕️ Doctor Name      :  mohan krishnan
📅 Appointment Date :  2025-05-04
📝 Description      :  follow-up-visit
──────────────────────────────────────────
Appointment(appointment_id=10, patient_id=4, doctor_id=5, appointment_date=
```

```
def schedule_appointment(self, appointment: Appointment) -> bool:
    try:
        # □ Validate patient
        self.cursor.execute("SELECT 1 FROM patient WHERE patientid = %s",
(appointment.get_patient_id(),))
        if not self.cursor.fetchone():
            raise PatientNumberNotFoundException(f"□ Patient with ID
{appointment.get_patient_id()} not found.")

        # □ Validate doctor
        self.cursor.execute("SELECT 1 FROM doctor WHERE doctorid = %s",
(appointment.get_doctor_id(),))
        if not self.cursor.fetchone():
```

```
            print(f"□ Warning: Doctor with ID {appointment.get_doctor_id()} not
found. Proceeding anyway.")

        # □ Insert (excluding appointmentid since it's AUTO_INCREMENT)
        self.cursor.execute(
            """
            INSERT INTO appointment (patientid, doctorid, appointmentdate,
description)
            VALUES (%s, %s, %s, %s)
            """,
            (
                appointment.get_patient_id(),
                appointment.get_doctor_id(),
                appointment.get_appointment_date(),
                appointment.get_description()
            )
        )
        self.conn.commit()

        # □ Retrieve generated appointmentid
        generated_id = self.cursor.lastrowid

        # □ Display confirmation
        print("\n□ Appointment Scheduled Successfully!")
        print("□  Added Appointment Details")

print("────────────────────────────────────────────────────────────────────────────────────")
        print("□ Appointment ID   : ", generated_id)
        print("□□ Patient ID       : ", appointment.get_patient_id())
        print("□□ Doctor ID        : ", appointment.get_doctor_id())
        print("□ Appointment Date : ", appointment.get_appointment_date())
        print("□ Description      : ", appointment.get_description())

print("────────────────────────────────────────────────────────────────────────────────────")

        return True

    except PatientNumberNotFoundException as e:
```

```
        print(e)
        return False

    except Exception as e:
        print("□ An unexpected error occurred while scheduling the appointment:",
str(e))
        return False
```

```
---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 1
Patient ID: 6
Doctor ID: 4
Date (YYYY-MM-DD): 2026-05-05
Description: general checkup

✅ Appointment Scheduled Successfully!
📌  Added Appointment Details
──────────────────────────────────────────
🆔 Appointment ID   :  12
👤 Patient ID       :  6
👨‍⚕️ Doctor ID        :  4
📅 Appointment Date :  2026-05-05
📝 Description      :  general checkup
──────────────────────────────────────────
```

```
def update_appointment_doctor(self, appointment_id: int, new_doctor_id: int) ->
bool:
    try:
        self.cursor.execute("SELECT 1 FROM appointment WHERE
appointmentid = %s", (appointment_id,))
        if not self.cursor.fetchone():
            raise PatientNumberNotFoundException(f"Appointment ID
{appointment_id} not found.")

        self.cursor.execute(
            "UPDATE appointment SET doctorid = %s WHERE appointmentid =
%s",
            (new_doctor_id, appointment_id)
        )
```

```
        self.conn.commit()

        updated = self.get_appointment_by_id(appointment_id)
        print("\n□ Doctor update attempted. Current Appointment Details:")
        print("Appointment ID     :", updated.get_appointment_id())
        print("Patient ID         :", updated.get_patient_id())
        print("Doctor ID          :", updated.get_doctor_id())
        print("Appointment Date   :", updated.get_appointment_date())
        print("Description        :", updated.get_description())

        return True

    except PatientNumberNotFoundException as e:
        print("□ Error:", e)
        return False

    except Exception as e:
        print("□ Error updating doctor:", str(e))
        return False
```

```
---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 5

-- Update Appointment --
1. Change Doctor
2. Change Appointment Date
3. Exit
Enter your choice (1 or 2 or 3): 1
Enter Appointment ID: 4
Enter New Doctor ID: 2

📋  Appointment Details
──────────────────────────────────────
🆔 Appointment ID   :  4
👤 Patient ID       :  4
👨‍⚕️ Doctor ID        :  2
📅 Date             :  2025-04-14
📝 Description      :  headache consultation
──────────────────────────────────────

✅ Doctor update attempted. Current Appointment Details:
Appointment ID     : 4
Patient ID         : 4
Doctor ID          : 2
Appointment Date   : 2025-04-14
Description        : headache consultation
```

```
def update_appointment_date(self, appointment_id: int, new_date: str) -> bool:
    try:
        # Check if appointment exists
        self.cursor.execute("SELECT 1 FROM appointment WHERE
appointmentid = %s", (appointment_id,))
        if not self.cursor.fetchone():
            raise PatientNumberNotFoundException(f"Appointment ID
{appointment_id} not found.")

        # Perform update
        self.cursor.execute(
            "UPDATE appointment SET appointmentdate = %s WHERE
appointmentid = %s",
            (new_date, appointment_id)
        )
        self.conn.commit()

        if self.cursor.rowcount > 0:
            print("\n□ Appointment date updated successfully. Updated
Appointment:")
            updated = self.get_appointment_by_id(appointment_id)
            print("Appointment ID     :", updated.get_appointment_id())
            print("Patient ID         :", updated.get_patient_id())
            print("Doctor ID          :", updated.get_doctor_id())
            print("Appointment Date   :", updated.get_appointment_date())
            print("Description        :", updated.get_description())
            return True
        return False

    except PatientNumberNotFoundException as e:
        print("□ Error:", e)
        return False

    except Exception as e:
        print("□ Error updating appointment date:", str(e))
        return False
```

```
---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 5

-- Update Appointment --
1. Change Doctor
2. Change Appointment Date
3. Exit
Enter your choice (1 or 2 or 3): 2
Enter Appointment ID: 4
Enter New Appointment Date (YYYY-MM-DD): 2027-09-05

✅ Appointment date updated successfully. Updated Appointment:

📋  Appointment Details
──────────────────────────────────────
🆔 Appointment ID   :  4
👤 Patient ID       :  4
👨‍⚕️ Doctor ID        :  2
📅 Date             :  2027-09-05
📝 Description      :  headache consultation
──────────────────────────────────────
```

```
def cancel_appointment(self, appointment_id: int) -> bool:
    try:
        self.cursor.execute("DELETE FROM appointment WHERE appointmentid
= %s", (appointment_id,))
        self.conn.commit()

        if self.cursor.rowcount == 0:
            raise AppointmentNotFoundException(f"Appointment ID
{appointment_id} not found.")

        print("\n □Appointment cancelled successfully.")
        print("□ Please collect your refund amount: ₹500")
        return True

    except AppointmentNotFoundException as e:
        print("□", e)
        return False
```

```
    except Exception as e:
        print("□ Error cancelling appointment:", str(e))
        return False
```

```
---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 6
Appointment ID to cancel: 12

 ✅Appointment cancelled successfully.
💸 Please collect your refund amount: ₹500
```

```
from entity.appointment import Appointment
from dao.hospitalserviceimpl import HospitalServiceImpl
from exception.patientnumbernotfoundexception import
PatientNumberNotFoundException
```

Main.py:

```
def main():
    service = HospitalServiceImpl()

    while True:  # □ Outer loop to allow re-login
        print("\n---- Hospital Management System Login ----")
        print("1. Admin Login")
        print("2. Doctor Login")
        print("3. Patient Login")
        print("4. Exit")
        login_type = input("Select Login Type (1/2/3/4): ")

        is_admin = False
        is_doctor = False
        is_patient = False

        if login_type == '1':
```

```
    username = input("Enter admin username: ")
    password = input("Enter admin password: ")
    if username == "admin" and password == "admin123":
        is_admin = True
    else:
        print("□ Invalid admin credentials.")
        continue

elif login_type == '2':
    username = input("Enter doctor username: ")
    password = input("Enter doctor password: ")
    if username == "doctor" and password == "doc123":
        is_doctor = True
    else:
        print("□ Invalid doctor credentials.")
        continue

elif login_type == '3':
    username = input("Enter patient username: ")
    password = input("Enter patient password: ")
    if username == "patient" and password == "pat123":
        is_patient = True
    else:
        print("□ Invalid patient credentials.")
        continue

elif login_type == '4':
    print("□ Exiting the system. Goodbye!")
    break

else:
    print("□ Invalid login type.")
    continue

# □ Inner loop: operations after login
while True:
    print("\n---- Hospital Management System ----")
    if is_admin:
        print("1. Schedule Appointment")
    print("2. View Appointment by ID")
```

```
print("3. View Appointments by Patient ID")
print("4. View Appointments by Doctor ID")
if is_admin:
    print("5. Update Appointment")
    print("6. Cancel Appointment")
print("7. Logout")

choice = input("Enter your choice: ")

try:
    if choice == '1' and is_admin:
        pid = int(input("Patient ID: "))
        did = int(input("Doctor ID: "))
        date = input("Date (YYYY-MM-DD): ")
        desc = input("Description: ")
        appt = Appointment(None, pid, did, date, desc)
        service.schedule_appointment(appt)

    elif choice == '2':
        aid = int(input("Enter Appointment ID: "))
        appt = service.get_appointment_by_id(aid)
        print(appt if appt else "No appointment found.")

    elif choice == '3':
        pid = int(input("Enter Patient ID: "))
        appts = service.get_appointments_for_patient(pid)
        for a in appts:
            print(a)

    elif choice == '4':
        did = int(input("Enter Doctor ID: "))
        appts = service.get_appointments_for_doctor(did)
        for a in appts:
            print(a)

    elif choice == '5' and is_admin:
        print("\n-- Update Appointment --")
        print("1. Change Doctor")
        print("2. Change Appointment Date")
        print("3. Exit")
```

```
                sub_choice = input("Enter your choice (1 or 2 or 3): ")

                if sub_choice == '1':
                    aid = int(input("Enter Appointment ID: "))
                    new_did = int(input("Enter New Doctor ID: "))
                    service.update_appointment_doctor(aid, new_did)

                elif sub_choice == '2':
                    aid = int(input("Enter Appointment ID: "))
                    new_date = input("Enter New Appointment Date (YYYY-MM-
DD): ")
                    service.update_appointment_date(aid, new_date)

            elif choice == '6' and is_admin:
                aid = int(input("Appointment ID to cancel: "))
                service.cancel_appointment(aid)

            elif choice == '7':
                print("□ Logging out...")
                break  # exits inner loop, returns to login menu

            else:
                print("Invalid option or access denied.")

        except PatientNumberNotFoundException as e:
            print("Error:", e)
        except ValueError:
            print("Invalid input type.")
        except Exception as e:
            print("An unexpected error occurred:", e)

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

```
C:\Users\admin\Downloads\hospital-management>python main.py

---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): 1
Enter admin username: admin
Enter admin password: admin123

---- Hospital Management System ----
1. Schedule Appointment
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
5. Update Appointment
6. Cancel Appointment
7. Logout
Enter your choice: 7
🔁 Logging out...

---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): |
```

```
---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): 2
Enter doctor username: doctor
Enter doctor password: doc123

---- Hospital Management System ----
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
7. Logout
Enter your choice: 7
🔁 Logging out...

---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): |
```

```
---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): 3
Enter patient username: patient
Enter patient password: pat123

---- Hospital Management System ----
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
7. Logout
Enter your choice: |
```

```
---- Hospital Management System ----
2. View Appointment by ID
3. View Appointments by Patient ID
4. View Appointments by Doctor ID
7. Logout
Enter your choice: 7
🔁 Logging out...

---- Hospital Management System Login ----
1. Admin Login
2. Doctor Login
3. Patient Login
4. Exit
Select Login Type (1/2/3/4): 4
✅ Exiting the system. Goodbye!

C:\Users\admin\Downloads\hospital-management>|
```

10.FUTURE ENHANCEMENT

- Implement GUI using Tkinter or web frontend using Flask/Django.
- Add authentication with hashed passwords for all users.
- Include modules for billing, prescriptions, and report generation.
- Use logging module for tracking application behavior.
- Introduce email/SMS notifications on appointment actions.

11.CONCLUSION

This Hospital Management System project serves as a solid foundational application for managing core hospital functions. It successfully uses Python and PyMySQL to interact with a relational database and employs a clean, scalable structure. While simple and educational in scope, it mirrors the functionality of real-world appointment systems and provides a strong platform for future enhancements.

The project demonstrates the power of Python in backend development and showcases how well it integrates with MySQL for data persistence and operations.