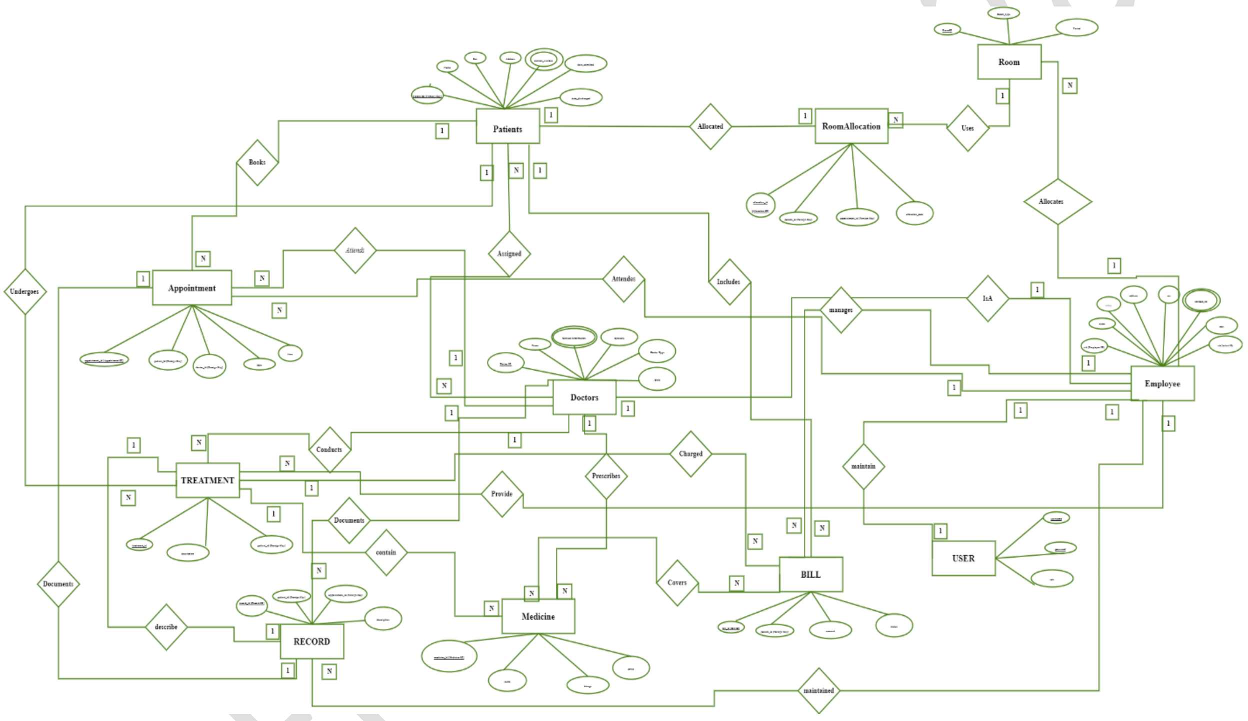
# 21AIE303: DBMS

# Assignment 3

K.V.Vamshidharreddy

CB.EN.U4AIE22028

## ER diagram for Hospital Management System



## Code:

### appointment.py

```python
import sqlite3

class Appointment:
    def __init__(self):
        self.conn = sqlite3.connect('hospital_management.db')
        self.c = self.conn.cursor()
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS Appointment (
            appointment_id INTEGER PRIMARY KEY AUTOINCREMENT,
            patient_id INTEGER NOT NULL,
            doctor_id INTEGER NOT NULL,
            appointment_time TEXT NOT NULL,
            status TEXT NOT NULL,
            FOREIGN KEY (patient_id) REFERENCES Patient(pid),
            FOREIGN KEY (doctor_id) REFERENCES Doctor(did)
        )''')
        self.conn.commit()

    def add_appointment(self, patient_id, doctor_id, appointment_time,
status):
        self.c.execute('INSERT INTO Appointment (patient_id, doctor_id,
appointment_time, status) VALUES (?, ?, ?, ?)',
                       (patient_id, doctor_id, appointment_time, status))
        self.conn.commit()

    def view_appointments(self):
        self.c.execute('SELECT * FROM Appointment')
        return self.c.fetchall()

    def update_appointment(self, appointment_id, patient_id, doctor_id,
appointment_time, status):
        self.c.execute('''
        UPDATE Appointment SET patient_id = ?, doctor_id = ?, appointment_time
= ?, status = ? WHERE appointment_id = ?
        ''', (patient_id, doctor_id, appointment_time, status,
appointment_id))
        self.conn.commit()

    def delete_appointment(self, appointment_id):
        self.c.execute('DELETE FROM Appointment WHERE appointment_id = ?',
(appointment_id,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

## bill.py

```
import sqlite3

class Bill:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.c.execute('''CREATE TABLE IF NOT EXISTS Bill (
            bid INTEGER PRIMARY KEY AUTOINCREMENT,
            pid INTEGER,
            treatment TEXT,
            medicine_code INTEGER,
            medicine_name TEXT,
            quantity INTEGER,
            price REAL,
            FOREIGN KEY (pid) REFERENCES Patient(pid) ON DELETE CASCADE
        )''')
        self.conn.commit()

    def add_bill(self, pid, treatment, medicine_code, medicine_name, quantity,
price):
        self.c.execute('INSERT INTO Bill (pid, treatment, medicine_code,
medicine_name, quantity, price) VALUES (?, ?, ?, ?, ?, ?)',
                       (pid, treatment, medicine_code, medicine_name,
quantity, price))
        self.conn.commit()

    def view_bills(self):
        self.c.execute('SELECT * FROM Bill')
        return self.c.fetchall()

    def update_bill(self, bid, pid, treatment, medicine_code, medicine_name,
quantity, price):
        self.c.execute('''UPDATE Bill SET pid = ?, treatment = ?,
medicine_code = ?, medicine_name = ?, quantity = ?, price = ? WHERE bid =
?''',
                       (pid, treatment, medicine_code, medicine_name,
quantity, price, bid))
        self.conn.commit()

    def delete_bill(self, bid):
        self.c.execute('DELETE FROM Bill WHERE bid = ?', (bid,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

## doctor.py

```
import sqlite3

class Doctor:
    def __init__(self, db_name='hospital_management.db'):
        # Establish connection to the database
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()

        # Create the 'Doctor' table if it doesn't exist
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS Doctor (
            did INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            contact_no TEXT NOT NULL,
            doctor_type TEXT NOT NULL,
            speciality TEXT NOT NULL,
            shift TEXT NOT NULL
        )
        ''')
        self.conn.commit()

    def add_doctor(self, eid, name, contact_no, doctor_type, speciality,
shift):
        # Insert new doctor record into the table
        self.c.execute('INSERT INTO Doctor (did, name, contact_no,
doctor_type, speciality, shift) VALUES (?, ?, ?, ?, ?, ?)',
                       (eid, name, contact_no, doctor_type, speciality,
shift))
        self.conn.commit()

    def view_doctors(self):
        # Retrieve all doctor records from the table
        self.c.execute('SELECT * FROM Doctor')
        return self.c.fetchall()

    def update_doctor(self, did, name, contact_no, doctor_type, speciality,
shift):
        # Update a doctor record based on the given id (did)
        self.c.execute('''UPDATE Doctor SET name = ?, contact_no = ?,
doctor_type = ?, speciality = ?, shift = ? WHERE did = ?''',
                       (name, contact_no, doctor_type, speciality, shift,
did))
        self.conn.commit()

    def delete_doctor(self, did):
        # Delete a doctor record based on the given id (did)
        self.c.execute('DELETE FROM Doctor WHERE did = ?', (did,))
        self.conn.commit()

    def close_connection(self):
        # Close the database connection
        self.conn.close()
```

## employee.py

```python
import sqlite3

class Employee:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()

        # Create the 'Employee' table if it doesn't exist
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS Employee (
            eid INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            salary REAL,
            address TEXT,
            sex TEXT,
            contact_no TEXT,
            nid INTEGER,
            role TEXT
        )
        ''')
        self.conn.commit()

    def add_employee(self, name, salary, address, sex, contact_no, nid, role,
doctor_type=None, speciality=None, shift=None):
        # Insert new employee record into the table
        self.c.execute('INSERT INTO Employee (name, salary, address, sex,
contact_no, nid, role) VALUES (?, ?, ?, ?, ?, ?, ?)',
                       (name, salary, address, sex, contact_no, nid, role))
        self.conn.commit()

        # If the role is Doctor, also add to the Doctor table
        if role.lower() == 'doctor':
            eid = self.c.lastrowid  # Get the last inserted employee ID
            self.c.execute('INSERT INTO Doctor (did, name, contact_no,
doctor_type, speciality, shift) VALUES (?, ?, ?, ?, ?, ?)',
                           (eid, name, contact_no, doctor_type, speciality,
shift))
            self.conn.commit()

    def view_employees(self):
        # Retrieve all employee records from the table
        self.c.execute('SELECT * FROM Employee')
        return self.c.fetchall()

    def update_employee(self, eid, name, salary, address, sex, contact_no,
nid, role):
        # Update an employee record based on the given id (eid)
        self.c.execute('''UPDATE Employee SET name = ?, salary = ?, address =
?, sex = ?, contact_no = ?, nid = ?, role = ? WHERE eid = ?''',
                       (name, salary, address, sex, contact_no, nid, role,
eid))
        self.conn.commit()

    def delete_employee(self, eid):
        # Delete an employee record based on the given id (eid)
        self.c.execute('DELETE FROM Employee WHERE eid = ?', (eid,))
        self.conn.commit()

    def close_connection(self):
        # Close the database connection
        self.conn.close()
```

medicine.py

```
import sqlite3

class Medicine:
    def __init__(self):
        self.conn = sqlite3.connect('hospital_management.db')
        self.c = self.conn.cursor()
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS Medicine (
            medicine_code INTEGER PRIMARY KEY AUTOINCREMENT,
            medicine_name TEXT NOT NULL,
            description TEXT,
            price REAL NOT NULL
        )''')
        self.conn.commit()

    def add_medicine(self, medicine_name, description, price):
        self.c.execute('INSERT INTO Medicine (medicine_name, description,
price) VALUES (?, ?, ?)',
                       (medicine_name, description, price))
        self.conn.commit()

    def view_medicines(self):
        self.c.execute('SELECT * FROM Medicine')
        return self.c.fetchall()

    def update_medicine(self, medicine_code, medicine_name, description,
price):
        self.c.execute('''
        UPDATE Medicine SET medicine_name = ?, description = ?, price = ?
WHERE medicine_code = ?
```

```
        ''', (medicine_name, description, price, medicine_code))
        self.conn.commit()

    def delete_medicine(self, medicine_code):
        self.c.execute('DELETE FROM Medicine WHERE medicine_code = ?',
(medicine_code,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

## patient.py

```
import sqlite3

class Patient:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.c.execute('''CREATE TABLE IF NOT EXISTS Patient (
            pid INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            sex TEXT,
            address TEXT,
            contact_no TEXT,
            date_admitted TEXT,
            date_discharged TEXT
        )''')
        self.conn.commit()

    def add_patient(self, name, sex, address, contact_no, date_admitted,
date_discharged):
        self.c.execute('INSERT INTO Patient (name, sex, address, contact_no,
date_admitted, date_discharged) VALUES (?, ?, ?, ?, ?, ?)',
                       (name, sex, address, contact_no, date_admitted,
date_discharged))
        self.conn.commit()

    def view_patients(self):
        self.c.execute('SELECT * FROM Patient')
        return self.c.fetchall()

    def update_patient(self, pid, name, sex, address, contact_no,
date_admitted, date_discharged):
        self.c.execute('''UPDATE Patient SET name = ?, sex = ?, address = ?,
contact_no = ?, date_admitted = ?, date_discharged = ? WHERE pid = ?''',
                       (name, sex, address, contact_no, date_admitted,
date_discharged, pid))
        self.conn.commit()

    def delete_patient(self, pid):
        self.c.execute('DELETE FROM Patient WHERE pid = ?', (pid,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

## record.py

```
import sqlite3

class Record:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.c.execute('''CREATE TABLE IF NOT EXISTS Record (
            record_no INTEGER PRIMARY KEY AUTOINCREMENT,
            appointment TEXT NOT NULL,
            patient_ID INTEGER NOT NULL,
            description TEXT NOT NULL,
            FOREIGN KEY (patient_ID) REFERENCES Patient(pid) ON DELETE CASCADE
        )''')
        self.conn.commit()

    def add_record(self, appointment, patient_ID, description):
        self.c.execute('INSERT INTO Record (appointment, patient_ID,
description) VALUES (?, ?, ?)',
                       (appointment, patient_ID, description))
        self.conn.commit()

    def view_records(self):
        self.c.execute('SELECT * FROM Record')
        return self.c.fetchall()

    def update_record(self, record_no, appointment, patient_ID, description):
        self.c.execute('''UPDATE Record SET appointment = ?, patient_ID = ?,
description = ? WHERE record_no = ?''',
                       (appointment, patient_ID, description, record_no))
        self.conn.commit()

    def delete_record(self, record_no):
        self.c.execute('DELETE FROM Record WHERE record_no = ?', (record_no,))
        self.conn.commit()
    def close_connection(self):
        self.conn.close()
```

room_allocation.py

```
import sqlite3

class RoomAllocation:
    def __init__(self):
        self.conn = sqlite3.connect('hospital_management.db')
        self.c = self.conn.cursor()
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS RoomAllocation (
            allocation_id INTEGER PRIMARY KEY AUTOINCREMENT,
            patient_id INTEGER NOT NULL,
            room_id INTEGER NOT NULL,
            start_date TEXT NOT NULL,
            end_date TEXT NOT NULL,
            FOREIGN KEY (patient_id) REFERENCES Patient(pid),
            FOREIGN KEY (room_id) REFERENCES Room(rid)
        )''')
        self.conn.commit()

    def add_allocation(self, patient_id, room_id, start_date, end_date):
        self.c.execute('INSERT INTO RoomAllocation (patient_id, room_id,
start_date, end_date) VALUES (?, ?, ?, ?)',
                       (patient_id, room_id, start_date, end_date))
        self.conn.commit()

    def view_allocations(self):
        self.c.execute('SELECT * FROM RoomAllocation')
        return self.c.fetchall()

    def update_allocation(self, allocation_id, patient_id, room_id,
start_date, end_date):
        self.c.execute('''
        UPDATE RoomAllocation SET patient_id = ?, room_id = ?, start_date = ?,
end_date = ? WHERE allocation_id = ?
        ''', (patient_id, room_id, start_date, end_date, allocation_id))
        self.conn.commit()

    def delete_allocation(self, allocation_id):
        self.c.execute('DELETE FROM RoomAllocation WHERE allocation_id = ?',
(allocation_id,))
        self.conn.commit()
```

```python
    def close_connection(self):
        self.conn.close()
```

## room.py

```python
import sqlite3

class Room:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.c.execute('''CREATE TABLE IF NOT EXISTS Room (
            rid INTEGER PRIMARY KEY AUTOINCREMENT,
            room_type TEXT NOT NULL,
            period TEXT NOT NULL
        )''')
        self.conn.commit()

    def add_room(self, room_type, period):
        self.c.execute('INSERT INTO Room (room_type, period) VALUES (?, ?)',
                       (room_type, period))
        self.conn.commit()

    def view_rooms(self):
        self.c.execute('SELECT * FROM Room')
        return self.c.fetchall()

    def update_room(self, rid, room_type, period):
        self.c.execute('''UPDATE Room SET room_type = ?, period = ? WHERE rid
= ?''',
                       (room_type, period, rid))
        self.conn.commit()

    def delete_room(self, rid):
        self.c.execute('DELETE FROM Room WHERE rid = ?', (rid,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

## treatment.py

```python
import sqlite3

class Treatment:
    def __init__(self):
        self.conn = sqlite3.connect('hospital_management.db')
        self.c = self.conn.cursor()
        self.c.execute('''
        CREATE TABLE IF NOT EXISTS Treatment (
            treatment_id INTEGER PRIMARY KEY AUTOINCREMENT,
            treatment_name TEXT NOT NULL,
            description TEXT,
            cost REAL NOT NULL
        )''')
        self.conn.commit()

    def add_treatment(self, treatment_name, description, cost):
        self.c.execute('INSERT INTO Treatment (treatment_name, description,
cost) VALUES (?, ?, ?)',
                       (treatment_name, description, cost))
        self.conn.commit()

    def view_treatments(self):
        self.c.execute('SELECT * FROM Treatment')
        return self.c.fetchall()

    def update_treatment(self, treatment_id, treatment_name, description,
cost):
        self.c.execute('''
        UPDATE Treatment SET treatment_name = ?, description = ?, cost = ?
WHERE treatment_id = ?
        ''', (treatment_name, description, cost, treatment_id))
        self.conn.commit()

    def delete_treatment(self, treatment_id):
        self.c.execute('DELETE FROM Treatment WHERE treatment_id = ?',
(treatment_id,))
        self.conn.commit()

    def close_connection(self):
        self.conn.close()
```

user.py

```
import sqlite3
import hashlib

class User:
    def __init__(self, db_name='hospital_management.db'):
        self.conn = sqlite3.connect(db_name)
        self.c = self.conn.cursor()
        self.c.execute('''CREATE TABLE IF NOT EXISTS User (
            uid INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE,
            password TEXT,
            role TEXT
        )''')
        self.conn.commit()

    def hash_password(self, password):
        return hashlib.sha256(password.encode()).hexdigest()

    def register(self, username, password, role):
        if len(password) <= 6:
            print("Password must be greater than 6 characters.")
            return

        hashed_password = self.hash_password(password)
        try:
            self.c.execute('INSERT INTO User (username, password, role) VALUES
(?, ?, ?)',
                           (username, hashed_password, role))
            self.conn.commit()
            print("User registered successfully!")
        except sqlite3.IntegrityError:
            print("Username already exists. Please try a different one.")

    def login(self, username, password):
        hashed_password = self.hash_password(password)
        self.c.execute('SELECT * FROM User WHERE username = ? AND password =
?', (username, hashed_password))
        user = self.c.fetchone()
        if user:
            print("Login successful!")
            return user
        else:
            print("Invalid credentials.")
            return None

    def close_connection(self):
        self.conn.close()
```

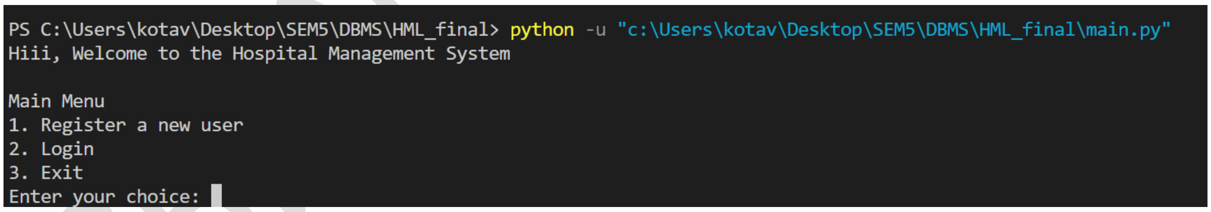
## Main.py

```python
import sys
from patient import Patient
from doctor import Doctor
from room import Room
from employee import Employee
from bill import Bill
from record import Record
from user import User
from treatment import Treatment
from appointment import Appointment
from room_allocation import RoomAllocation
from medicine import Medicine  # Assuming you have a medicine module

def main():
    print("Hiii, Welcome to the Hospital Management System")
    user = User()

    while True:
        print("\nMain Menu")
        print("1. Register a new user")
        print("2. Login")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            username = input("Enter new username: ")
            password = input("Enter new password: ")
            role = input("Enter role
(Manager/Doctor/Patient/Receptionist/Nurse): ")
            user.register(username, password, role)
        elif choice == '2':
            username = input("Enter username: ")
            password = input("Enter password: ")
            user_details = user.login(username, password)
            if user_details:
                role = user_details[3]  # Extract the role from user details
                logged_in_menu(role)
            else:
                print("Login failed. Please try again.")
        elif choice == '3':
            print("Exiting...")
            user.close_connection()
            sys.exit()
        else:
            print("Invalid choice. Please try again.")

def logged_in_menu(role):
    patient = Patient()
    doctor = Doctor()
    room = Room()
    employee = Employee()
    bill = Bill()
    record = Record()
    treatment = Treatment()
    appointment = Appointment()
    room_allocation = RoomAllocation()
    medicine = Medicine()  # Initialize medicine management

    while True:
        print("\nLogged-in Menu")
        if role == "Manager":
            print("1. Patient Management")
            print("2. Doctor Management")
            print("3. Room Management")
            print("4. Employee Management")
            print("5. Billing Management")
            print("6. Record Management")
            print("7. Treatment Management")
            print("8. Appointment Management")
            print("9. Room Allocation Management")
            print("10. Medicine Management")
        elif role == "Doctor":
            print("1. Patient Management")
            print("2. Room Management")
            print("3. Record Management")
            print("4. Treatment Management")
            print("5. Appointment Management")
            print("6. Medicine Management")
        elif role == "Patient":
            print("1. Personal Records")
            print("2. Billing Management")
            print("3. Appointment Management")
            print("4. Medicine Information")
        elif role == "Receptionist":
            print("1. Patient Management")
            print("2. Room Management")
            print("3. Appointment Management")
            print("4. Medicine Management")
        elif role == "Nurse":
            print("1. Patient Care Records")
            print("2. Room Management")
            print("3. Treatment Management")
            print("4. Medicine Management")

        print("0. Logout")
        choice = input("Enter your choice: ")

        if choice == '1':
            if role in ["Manager","Patient","Nurse", "Doctor",
"Receptionist"]:
                patient_menu(patient)
        elif choice == '2':
            if role == "Manager":
                doctor_menu(doctor)
            elif role in ["Doctor", "Receptionist", "Nurse"]:
                room_menu(room)
            elif role == "Patient":
                billing_menu(bill)
        elif choice == '3':
            if role == "Manager":
                room_menu(room)
            elif role == "Doctor":
                record_menu(record)
            elif role == "Nurse":
                treatment_menu(treatment)
            elif role in ["Receptionist", "Patient"] :
                appointment_menu(appointment)
        elif choice == '4':
            if role == "Manager":
                employee_menu(employee)
            elif role in ["Receptionist", "Nurse","Patient"]:
                medicine_menu(medicine)
            elif role == "Doctor":
                treatment_menu(treatment)
        elif choice == '5':
            if role == "Manager":
                billing_menu(bill)
            elif role == "Doctor":
                appointment_menu(appointment)
        elif choice == '6':
            if role == "Manager":
                record_menu(record)
            elif role == "Doctor":
                medicine_menu(medicine)
        elif choice == '7':
            if role in ["Manager"]:
                treatment_menu(treatment)
        elif choice == '8':
            if role in ["Manager", "Doctor", "Receptionist", "Patient"]:
                appointment_menu(appointment)
        elif choice == '9' and role == "Manager":
            room_allocation_menu(room_allocation)
        elif choice == '10':
            if role in ["Manager", "Doctor", "Receptionist", "Nurse"]:
                medicine_menu(medicine)
        elif choice == '0':
            print("Logging out...")
            break
        else:
            print("Invalid choice or Access Denied.")

    # Close connections to the databases
    close_all_connections(patient, doctor, room, employee, bill, record,
treatment, appointment, room_allocation, medicine)

def close_all_connections(*args):
    for component in args:
        component.close_connection()


def patient_menu(patient):
    while True:
        print("\nPatient Management Menu")
        print("1. Add Patient")
        print("2. View Patients")
        print("3. Update Patient")
        print("4. Delete Patient")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            name = input("Enter name: ")
            sex = input("Enter sex: ")
            address = input("Enter address: ")
            contact_no = input("Enter contact number: ")
            date_admitted = input("Enter date admitted (YYYY-MM-DD): ")
            date_discharged = input("Enter date discharged (YYYY-MM-DD) or
'N/A': ")
            patient.add_patient(name, sex, address, contact_no, date_admitted,
date_discharged)
        elif choice == '2':
            patients = patient.view_patients()
            for p in patients:
                print(p)
        elif choice == '3':
            pid = int(input("Enter Patient ID to update: "))
            name = input("Enter new name: ")
            sex = input("Enter new sex: ")
            address = input("Enter new address: ")
            contact_no = input("Enter new contact number: ")
            date_admitted = input("Enter new date admitted (YYYY-MM-DD): ")
            date_discharged = input("Enter new date discharged (YYYY-MM-DD) or 
'N/A': ")
            patient.update_patient(pid, name, sex, address, contact_no, 
date_admitted, date_discharged)
        elif choice == '4':
            pid = int(input("Enter Patient ID to delete: "))
            patient.delete_patient(pid)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def doctor_menu(doctor):
    while True:
        print("\nDoctor Management Menu")
        print("1. Add Doctor")
        print("2. View Doctors")
        print("3. Update Doctor")
        print("4. Delete Doctor")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            name = input("Enter name: ")
            contact_no = input("Enter contact number: ")
            doctor_type = input("Enter doctor type 
(Permanent/Visiting/Trainee): ")  # Updated doctor_type
            speciality = input("Enter speciality: ")
            shift = input("Enter shift (Morning/Evening/Night): ")  # Added 
shift input
            doctor.add_doctor(name, contact_no, doctor_type, speciality, 
shift)
            print("Doctor added successfully.")
        elif choice == '2':
            doctors = doctor.view_doctors()
            if doctors:
                for d in doctors:
                    print(d)
            else:
                print("No doctors found.")
        elif choice == '3':
            did = int(input("Enter Doctor ID to update: "))
            name = input("Enter new name: ")
            contact_no = input("Enter new contact number: ")
            doctor_type = input("Enter new doctor type 
(Permanent/Visiting/Trainee): ")  # Updated doctor_type input
            speciality = input("Enter new speciality: ")
            shift = input("Enter new shift (Morning/Evening/Night): ")  #
Added shift input for update
            doctor.update_doctor(did, name, contact_no, doctor_type,
speciality, shift)
            print("Doctor updated successfully.")
        elif choice == '4':
            did = int(input("Enter Doctor ID to delete: "))
            doctor.delete_doctor(did)
            print("Doctor deleted successfully.")
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")



def room_menu(room):
    while True:
        print("\nRoom Management Menu")
        print("1. Add Room")
        print("2. View Rooms")
        print("3. Update Room")
        print("4. Delete Room")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            room_type = input("Enter room type: ")
            period = input("Enter period: ")
            room.add_room(room_type, period)
        elif choice == '2':
            rooms = room.view_rooms()
            for r in rooms:
                print(r)
        elif choice == '3':
            rid = int(input("Enter Room ID to update: "))
            room_type = input("Enter new room type: ")
            period = input("Enter new period: ")
            room.update_room(rid, room_type, period)
        elif choice == '4':
            rid = int(input("Enter Room ID to delete: "))
            room.delete_room(rid)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")
def employee_menu(employee):
    while True:
        print("\nEmployee Management Menu")
        print("1. Add Employee")
        print("2. View Employees")
        print("3. Update Employee")
        print("4. Delete Employee")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            name = input("Enter name: ")
            salary = float(input("Enter salary: "))
            address = input("Enter address: ")
            sex = input("Enter sex: ")
            contact_no = input("Enter contact number: ")
            nid = int(input("Enter National ID: "))
            role = input("Enter role (Doctor/Receptionist/Nurse): ")

            # If role is Doctor, ask for doctor-specific details
            if role.lower() == 'doctor':
                doctor_type = input("Enter doctor type 
(Permanent/Visiting/Trainee): ")
                speciality = input("Enter speciality: ")
                shift = input("Enter shift (Morning/Evening/Night): ")
                employee.add_employee(name, salary, address, sex, contact_no, 
nid, role, doctor_type, speciality, shift)
            else:
                employee.add_employee(name, salary, address, sex, contact_no, 
nid, role)

        elif choice == '2':
            employees = employee.view_employees()
            for emp in employees:
                print(emp)
        elif choice == '3':
            eid = int(input("Enter Employee ID to update: "))
            name = input("Enter new name: ")
            salary = float(input("Enter new salary: "))
            address = input("Enter new address: ")
            sex = input("Enter new sex: ")
            contact_no = input("Enter new contact number: ")
            nid = int(input("Enter new National ID: "))
            role = input("Enter new role: ")
            employee.update_employee(eid, name, salary, address, sex, 
contact_no, nid, role)
        elif choice == '4':
            eid = int(input("Enter Employee ID to delete: "))
            employee.delete_employee(eid)
        elif choice == '5':
            break
```

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


def billing_menu(bill):
    while True:
        print("\nBilling Management Menu")
        print("1. Add Bill")
        print("2. View Bills")
        print("3. Update Bill")
        print("4. Delete Bill")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            patient_id = int(input("Enter Patient ID: "))
            amount = float(input("Enter amount: "))
            status = input("Enter bill status: ")
            bill.add_bill(patient_id, amount, status)
        elif choice == '2':
            bills = bill.view_bills()
            for b in bills:
                print(b)
        elif choice == '3':
            bill_id = int(input("Enter Bill ID to update: "))
            patient_id = int(input("Enter new Patient ID: "))
            amount = float(input("Enter new amount: "))
            status = input("Enter new bill status: ")
            bill.update_bill(bill_id, patient_id, amount, status)
        elif choice == '4':
            bill_id = int(input("Enter Bill ID to delete: "))
            bill.delete_bill(bill_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def record_menu(record):
    while True:
        print("\nRecord Management Menu")
        print("1. Add Record")
        print("2. View Records")
        print("3. Update Record")
        print("4. Delete Record")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            appointment_id = int(input("Enter Appointment ID: "))
            patient_id = int(input("Enter Patient ID: "))
            description = input("Enter description: ")
            record.add_record(appointment_id, patient_id, description)
        elif choice == '2':
            records = record.view_records()
            for r in records:
                print(r)
        elif choice == '3':
            record_id = int(input("Enter Record ID to update: "))
            appointment_id = int(input("Enter new Appointment ID: "))
            patient_id = int(input("Enter new Patient ID: "))
            description = input("Enter new description: ")
            record.update_record(record_id, appointment_id, patient_id,
description)
        elif choice == '4':
            record_id = int(input("Enter Record ID to delete: "))
            record.delete_record(record_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def treatment_menu(treatment):
    while True:
        print("\nTreatment Management Menu")
        print("1. Add Treatment")
        print("2. View Treatments")
        print("3. Update Treatment")
        print("4. Delete Treatment")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            patient_id = int(input("Enter Patient ID: "))
            description = input("Enter treatment description: ")
            treatment.add_treatment(patient_id, description)
        elif choice == '2':
            treatments = treatment.view_treatments()
            for t in treatments:
                print(t)
        elif choice == '3':
            treatment_id = int(input("Enter Treatment ID to update: "))
            patient_id = int(input("Enter new Patient ID: "))
            description = input("Enter new treatment description: ")
            treatment.update_treatment(treatment_id, patient_id, description)
        elif choice == '4':
            treatment_id = int(input("Enter Treatment ID to delete: "))
            treatment.delete_treatment(treatment_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def appointment_menu(appointment):
    while True:
        print("\nAppointment Management Menu")
        print("1. Schedule Appointment")
        print("2. View Appointments")
        print("3. Update Appointment")
        print("4. Cancel Appointment")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            patient_id = int(input("Enter Patient ID: "))
            doctor_id = int(input("Enter Doctor ID: "))
            date = input("Enter appointment date (YYYY-MM-DD): ")
            time = input("Enter appointment time (HH:MM): ")
            appointment.schedule_appointment(patient_id, doctor_id, date,
time)
        elif choice == '2':
            appointments = appointment.view_appointments()
            for a in appointments:
                print(a)
        elif choice == '3':
            appointment_id = int(input("Enter Appointment ID to update: "))
            patient_id = int(input("Enter new Patient ID: "))
            doctor_id = int(input("Enter new Doctor ID: "))
            date = input("Enter new appointment date (YYYY-MM-DD): ")
            time = input("Enter new appointment time (HH:MM): ")
            appointment.update_appointment(appointment_id, patient_id,
doctor_id, date, time)
        elif choice == '4':
            appointment_id = int(input("Enter Appointment ID to cancel: "))
            appointment.cancel_appointment(appointment_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def room_allocation_menu(room_allocation):
    while True:
        print("\nRoom Allocation Management Menu")
        print("1. Allocate Room")
        print("2. View Allocated Rooms")
        print("3. Update Room Allocation")
        print("4. Release Room")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            patient_id = int(input("Enter Patient ID: "))
            room_id = int(input("Enter Room ID: "))
            allocation_date = input("Enter allocation date (YYYY-MM-DD): ")
            room_allocation.allocate_room(patient_id, room_id,
allocation_date)
        elif choice == '2':
            allocations = room_allocation.view_allocations()
            for alloc in allocations:
                print(alloc)
        elif choice == '3':
            allocation_id = int(input("Enter Allocation ID to update: "))
            patient_id = int(input("Enter new Patient ID: "))
            room_id = int(input("Enter new Room ID: "))
            allocation_date = input("Enter new allocation date (YYYY-MM-DD):
")
            room_allocation.update_allocation(allocation_id, patient_id,
room_id, allocation_date)
        elif choice == '4':
            allocation_id = int(input("Enter Allocation ID to release: "))
            room_allocation.release_room(allocation_id)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

def medicine_menu(medicine):
    while True:
        print("\nMedicine Management Menu")
        print("1. Add Medicine")
        print("2. View Medicines")
        print("3. Update Medicine")
        print("4. Delete Medicine")
        print("5. Return to Previous Menu")
        choice = input("Enter your choice: ")

        if choice == '1':
            name = input("Enter medicine name: ")
            dosage = input("Enter dosage: ")
            quantity = int(input("Enter quantity: "))
            price = float(input("Enter price: "))
            medicine.add_medicine(name, dosage, quantity, price)
        elif choice == '2':
            medicines = medicine.view_medicines()
            for med in medicines:
                print(med)
        elif choice == '3':
            mid = int(input("Enter Medicine ID to update: "))
            name = input("Enter new medicine name: ")
            dosage = input("Enter new dosage: ")
            quantity = int(input("Enter new quantity: "))
            price = float(input("Enter new price: "))
            medicine.update_medicine(mid, name, dosage, quantity, price)
        elif choice == '4':
            mid = int(input("Enter Medicine ID to delete: "))
            medicine.delete_medicine(mid)
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

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

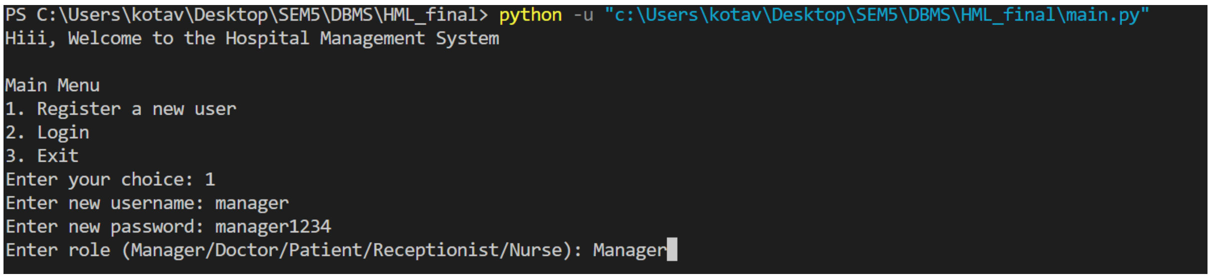
Outputs:

Registration:



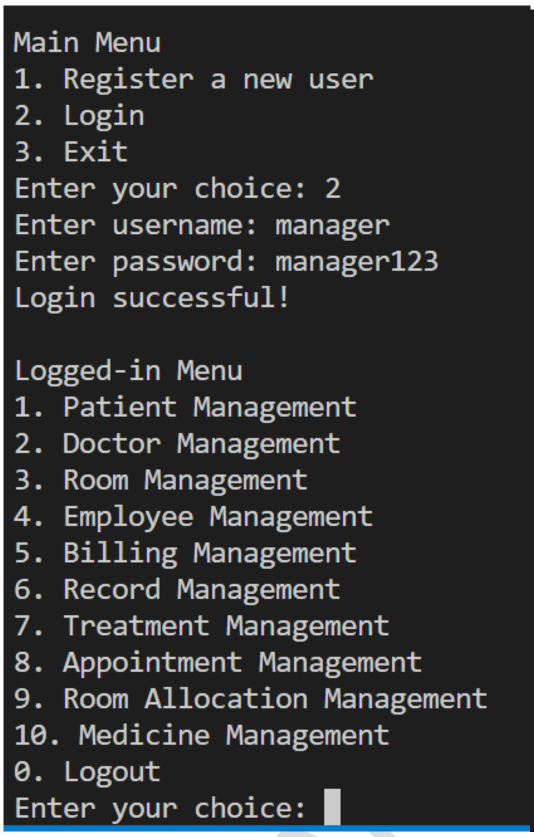
```
PS C:\Users\kotav\Desktop\SEM5\DBMS\HML_final> python -u "c:\Users\kotav\Desktop\SEM5\DBMS\HML_final\main.py"
Hiii, Welcome to the Hospital Management System

Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 
```

Registration with specific Role:

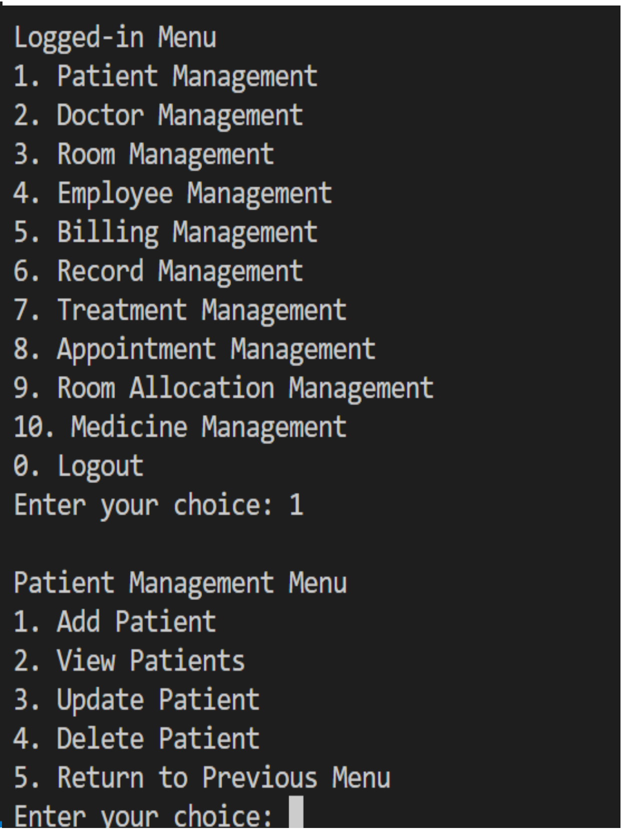


```
PS C:\Users\kotav\Desktop\SEM5\DBMS\HML_final> python -u "c:\Users\kotav\Desktop\SEM5\DBMS\HML_final\main.py"
Hiii, Welcome to the Hospital Management System

Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 1
Enter new username: manager
Enter new password: manager1234
Enter role (Manager/Doctor/Patient/Receptionist/Nurse): Manager
```

Manager logged menu:

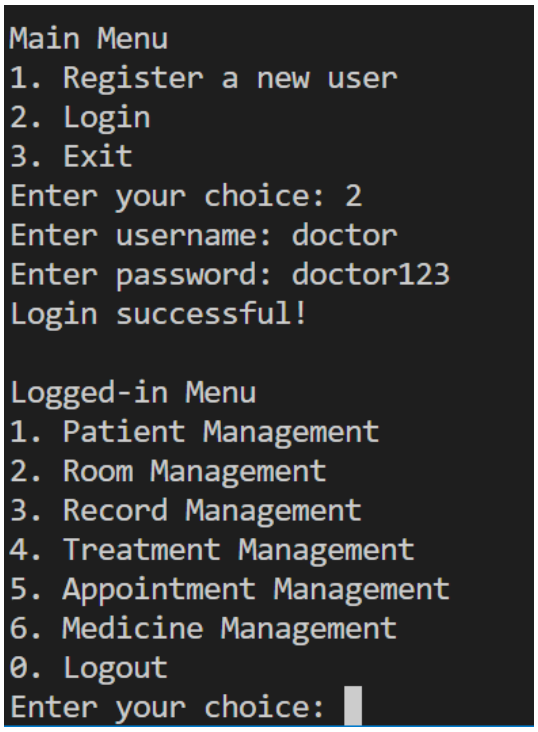


```
Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 2
Enter username: manager
Enter password: manager123
Login successful!

Logged-in Menu
1. Patient Management
2. Doctor Management
3. Room Management
4. Employee Management
5. Billing Management
6. Record Management
7. Treatment Management
8. Appointment Management
9. Room Allocation Management
10. Medicine Management
0. Logout
Enter your choice:
```



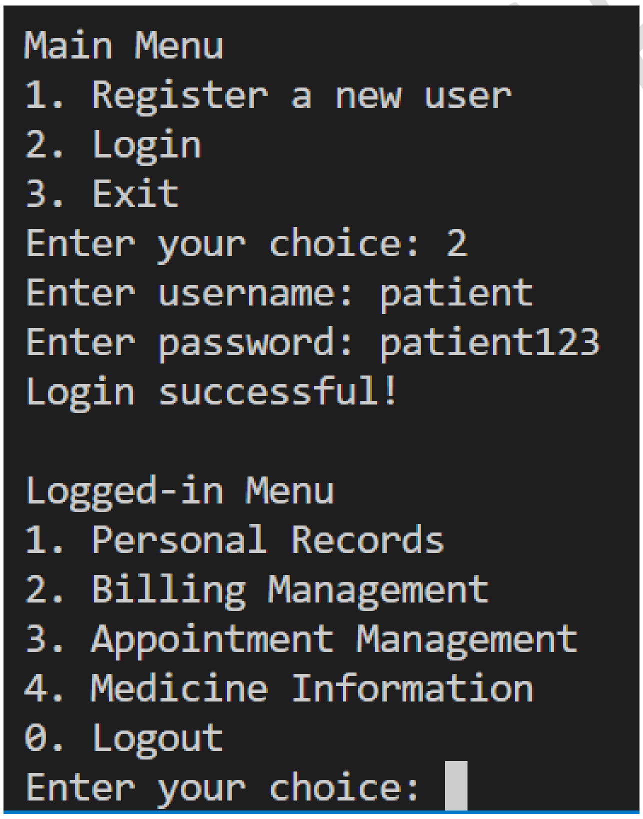
```
Logged-in Menu
1. Patient Management
2. Doctor Management
3. Room Management
4. Employee Management
5. Billing Management
6. Record Management
7. Treatment Management
8. Appointment Management
9. Room Allocation Management
10. Medicine Management
0. Logout
Enter your choice: 1

Patient Management Menu
1. Add Patient
2. View Patients
3. Update Patient
4. Delete Patient
5. Return to Previous Menu
Enter your choice:
```

Doctor logged menu:



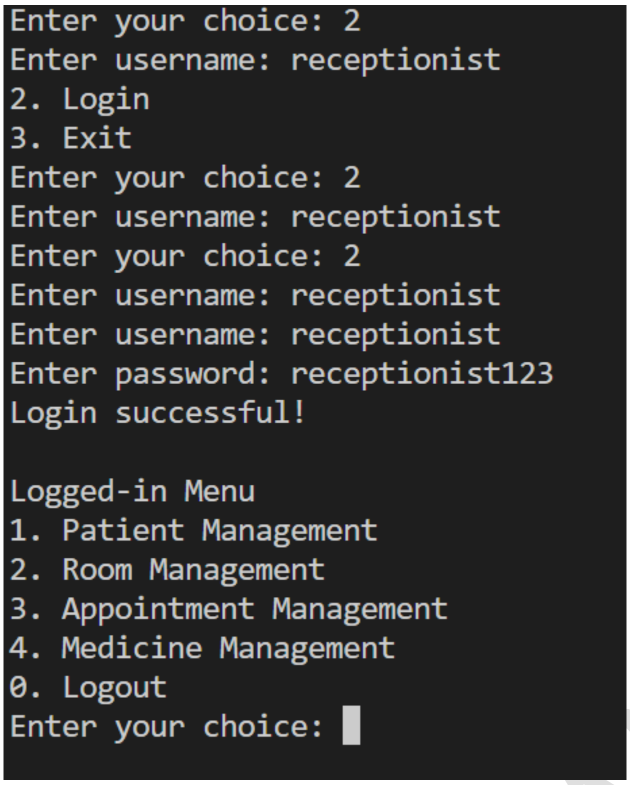
```
Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 2
Enter username: doctor
Enter password: doctor123
Login successful!

Logged-in Menu
1. Patient Management
2. Room Management
3. Record Management
4. Treatment Management
5. Appointment Management
6. Medicine Management
0. Logout
Enter your choice: 
```

Patient logged menu:

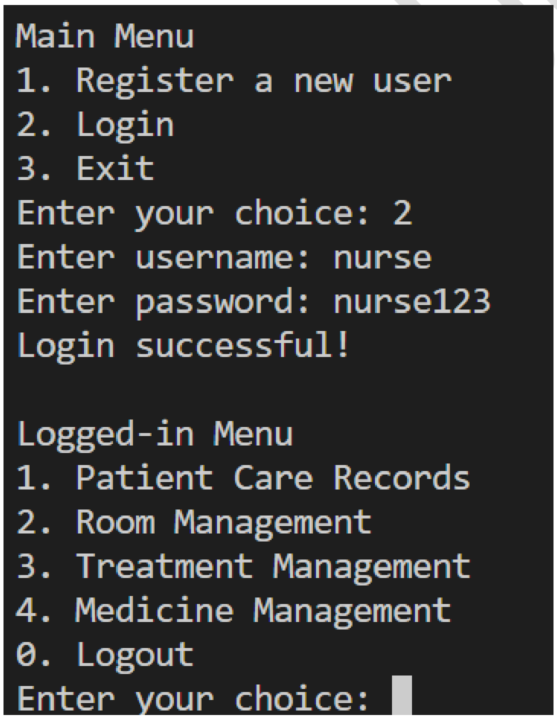


```
Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 2
Enter username: patient
Enter password: patient123
Login successful!

Logged-in Menu
1. Personal Records
2. Billing Management
3. Appointment Management
4. Medicine Information
0. Logout
Enter your choice: 
```

Receptionist logged menu:


```
Enter your choice: 2
Enter username: receptionist
2. Login
3. Exit
Enter your choice: 2
Enter username: receptionist
Enter your choice: 2
Enter username: receptionist
Enter username: receptionist
Enter password: receptionist123
Login successful!

Logged-in Menu
1. Patient Management
2. Room Management
3. Appointment Management
4. Medicine Management
0. Logout
Enter your choice:
```

Nurse logged menu:


```
Main Menu
1. Register a new user
2. Login
3. Exit
Enter your choice: 2
Enter username: nurse
Enter password: nurse123
Login successful!

Logged-in Menu
1. Patient Care Records
2. Room Management
3. Treatment Management
4. Medicine Management
0. Logout
Enter your choice:
```