Hospital Management System

By:
Smriti Singh(T00678188),
Surya Parkash (T00682916)

Hospital Management System

The purpose of this project is to implement a simple Hospital Management System. The system is designed to manage information about patients, doctors, appointments, and departments within a hospital. Users, such as hospital staff, can perform various tasks including adding or updating patient information, viewing patient details, adding doctors, viewing departments and doctors, admitting and discharging patients, booking appointments, and viewing appointments for a specific patient. The program provides a text-based interface for users to interact with the hospital management functionalities.

Purpose

1. Add Patient: The application enables the addition of new patients to the system, capturing relevant details such as name, age, gender, etc.

2. Add Doctor: The application enables the addition of new patients and doctors to the system, capturing relevant details such as name, specialization, and experience, etc.

3. Search Patient by Name: Users can search for patient information by entering a name. This feature facilitates quick retrieval of patient details.

4. View doctors and their departments: Users can see a list of doctor along with their departments. This feature facilitates quick retrieval of doctor details.

5. Book Appointment: The system supports the booking of appointments for patients. Users can select a department, view available doctors in that department, and choose a specific doctor to book an appointment.

6. Display Appointment Details: Users can view appointment details for a specific patient, including the assigned doctor, department, and additional appointment details.

Target Users

The target users for this Hospital Management System project could include hospital administrators, receptionists, and healthcare professionals who need to manage patient records, doctor information, appointments, and other related data within the hospital.

Additionally, it could be used by staff responsible for admitting and discharging patients, scheduling appointments, and maintaining the overall organization of the hospital.

1. Administrators: Those responsible for overseeing the entire hospital management system, monitoring patient admissions, managing doctor information, and ensuring the smooth functioning of the hospital.

2. Receptionists: Staff at the reception desk who might use the system to add or update patient information, schedule appointments, and view patient details.

3. Doctors: Healthcare professionals who could use the system to view patient details, and other relevant information.

4. System Operators: Individuals responsible for the technical aspects of maintaining and operating the Hospital Management System.

Thoughts Behind the Project

The addresses the need for a basic Hospital Management System, with the intention of providing a simple yet functional tool for managing patient and doctor information, appointments, and other related data within a hospital setting. Here are some thoughts behind the project:

1. Record Management: The project focuses on managing patient records, doctor information, and appointments. This is crucial for maintaining an organized and efficient healthcare environment.

2. User Interaction: The console-based interface provides a straightforward way for users to interact with the system. While it might lack the graphical sophistication of some modern systems, it remains accessible and can be easily navigated.

3. Functionality: The system covers essential functionalities such as adding/updating patient and doctor information, admitting and discharging patients, scheduling appointments, and viewing logs. These functionalities align with typical requirements in a hospital setting.

4. Departmental Organization: The inclusion of departments and doctors within departments suggests an awareness of the hierarchical structure within a hospital, where doctors specialize in specific areas.

5. Random Assignment of Doctors: The project incorporates a random assignment of doctors from a specified department when admitting a patient. This adds an element of realism, simulating the practical challenges of assigning patients to available doctors.

6. Simplicity: The design appears to prioritize simplicity, making it easy for users to understand and use the system without unnecessary complexity.

7. Continuous Interaction: The program is designed to run in a loop, allowing users to perform multiple tasks without restarting the application.

Overall, this project is a foundational implementation that could be extended and enhanced based on specific requirements or used as a starting point for further development.

Design Considerations

The project exhibits several design considerations, some intentional and others that might be enhanced in future iterations. Some of the design considerations in the project are:

1. Object-Oriented Design: The project employs object-oriented principles by defining classes such as `Patient`, `Doctor`, `Appointment`, and `Hospital`. This design allows for encapsulation, abstraction, and better organization of code.

2. Class Responsibilities: Each class (`Patient`, `Doctor`, and `Hospital`) has well-defined responsibilities, contributing to a modular and maintainable codebase.

3. User Interaction: The system utilizes a simple console-based interface for user interaction. While this approach might lack the visual appeal of a graphical user interface (GUI), it ensures simplicity and ease of use.

4. Data Storage: Patient and doctor information is stored in vectors, and appointments are stored within the respective doctor objects. This straightforward approach works well for the current scope, but for a larger system, considerations like database integration might be necessary.

5. Random Doctor Assignment: The random assignment of a doctor from a specified department during patient admission adds a degree of realism. However, future improvements might include a more sophisticated assignment algorithm or the ability to choose a specific doctor.

6. Departmental Structure: The inclusion of departments for doctors shows an understanding of the hierarchical organization within a hospital. This design

consideration supports the practicality of managing doctors based on their specializations.

7. Operator Overloading: The `==` and `!=` operators are overloaded for the `Patient` class, allowing for more natural comparisons between `Patient` objects.

8. Dynamic Data: The program uses dynamic data structures, such as vectors for storing patients and doctors. This allows for flexible management of data and easy expansion of the system.

9. Menu-Driven Interface: The system utilizes a menu-driven approach, making it easy for users to navigate through different functionalities. Each option corresponds to a specific task, contributing to user-friendly interactions.

10. Code Reusability: The design allows for code reusability, as common functionalities like viewing logs are encapsulated within the `Patient` class and can be easily called.

<u>Scope of the First Version</u>

The first version of this Hospital Management System project covers essential functionalities for managing patient and doctor information within a hospital setting. This includes:

1. Patient Management:
- Addition and updating of patient information (name, age, diagnosis).
- Changing the status of a patient (e.g., Registered, Admitted, Discharged).
- Viewing logs for a specific patient, including appointments, symptoms, and medication.

2. Doctor Management:
- Addition of doctor information (name, department, specialization, experience).
- Viewing the list of doctors, organized by departments.

3. Appointment Management:
- Booking appointments for patients with specific doctors.
- Viewing appointments for a specific patient.

4. Hospital Operations:
- Admitting a patient, which involves assigning a doctor from a specified department.
- Discharging a patient, changing their status to "Discharged."

5. User Interaction:
- Menu-driven console interface for users to interact with the system.

- Continuous loop allowing users to perform multiple tasks without restarting the application.

6. Object-Oriented Design:
- Implementation of classes (`Patient`, `Doctor`, `Appointment`, `Hospital`) using object-oriented principles, providing a modular and organized codebase.

7. Random Doctor Assignment:
- Randomly assigning a doctor from a specified department when admitting a patient, simulating real-world scenarios.

8. Simplicity:
- Emphasis on simplicity and ease of use, making it suitable for basic hospital management needs.

Testing

Testing the functionality of the Hospital Management System involves systematically checking various aspects of the program to ensure that it behaves as expected. Below are the steps to test the different functionalities of this project:

1. Add or Update Patient:
    - Choose option 1 from the menu.
    - Enter the details for a new patient.
    - Verify that the patient is added successfully.
    - Update the details for an existing patient and ensure the information is updated.

2. View Patients:
    - Choose option 2 from the menu.
    - Enter the name of a patient to view their logs.
    - Verify that the patient's details, diagnosis, status, and appointments are displayed correctly.

3. Add Doctor:
    - Choose option 3 from the menu.
    - Enter the details for a new doctor.
    - Verify that the doctor is added successfully.

4. View Doctors and Departments:
    - Choose option 4 from the menu.
    - Verify that the departments and doctors are displayed correctly.

5. Admit Patient:
    - Choose option 5 from the menu.
    - Enter the name of a patient to admit.
    - Verify that the patient is assigned to a doctor in the specified department.
    - Check that the patient's status is updated to "Admitted."

6. Discharge Patient:
    - Choose option 6 from the menu.
    - Enter the name of a patient to discharge.
    - Verify that the patient's status is updated to "Discharged."

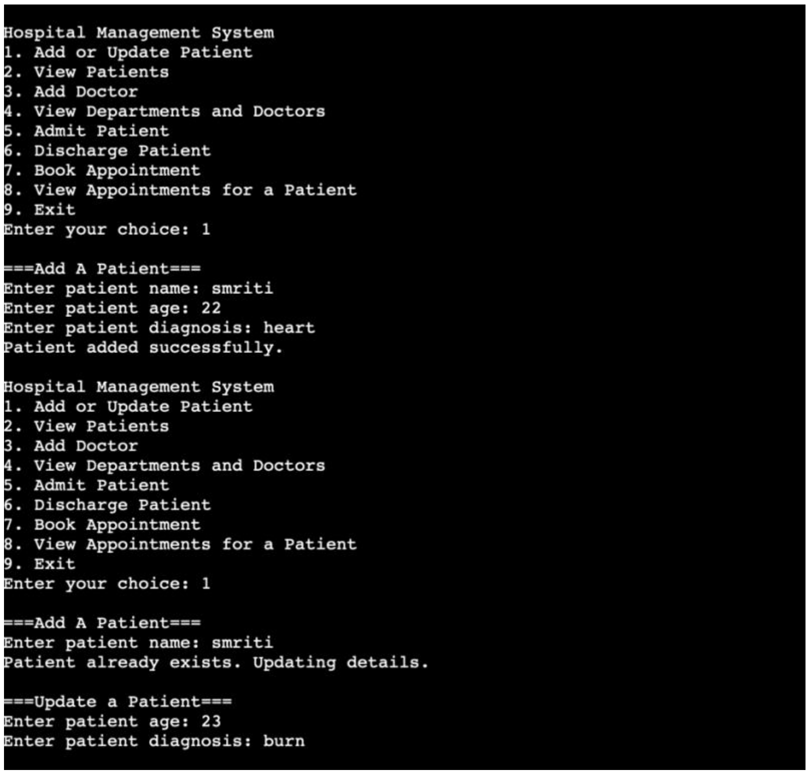
7. Book Appointment:
    - Choose option 7 from the menu.
    - Enter the name of a patient.
    - Choose a department and a doctor for the appointment.
    - Verify that the appointment is booked successfully.

8. View Appointments for a Patient:
    - Choose option 8 from the menu.
    - Enter the name of a patient to view their appointments.
    - Verify that the patient's appointments are displayed correctly.

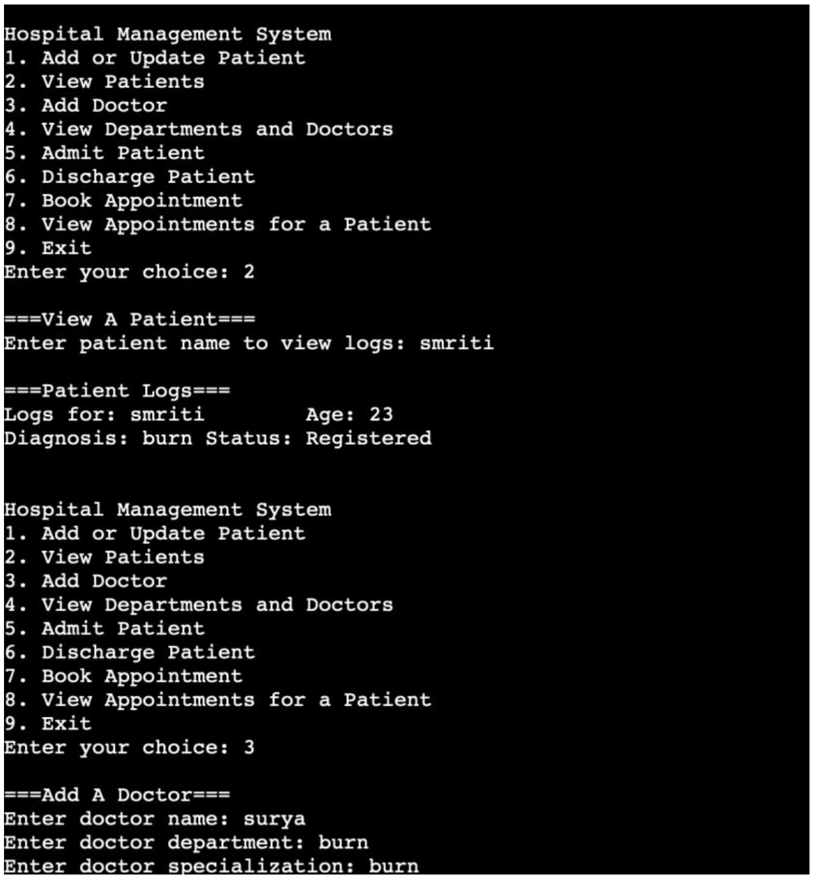
9. Exit Program:
    - Choose option 9 from the menu.
    - Verify that the program exits gracefully.

10. Random Doctor Assignment:
    - Admit multiple patients to a department and observe the random assignment of doctors.
    - Verify that the assignment is fair and realistic.

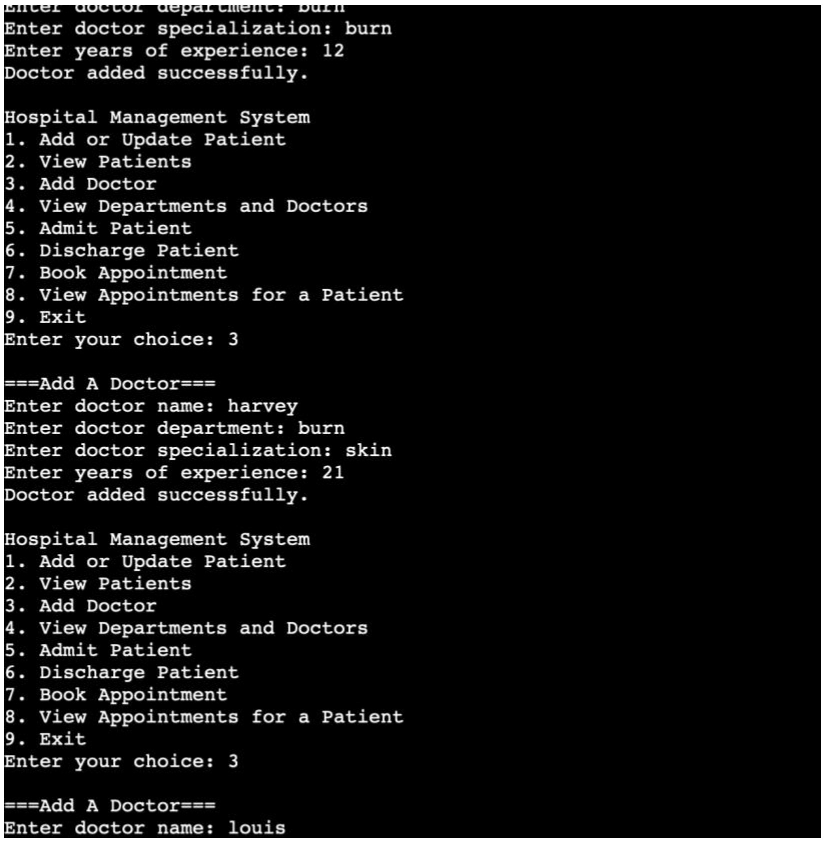
Here is a sample of the run:



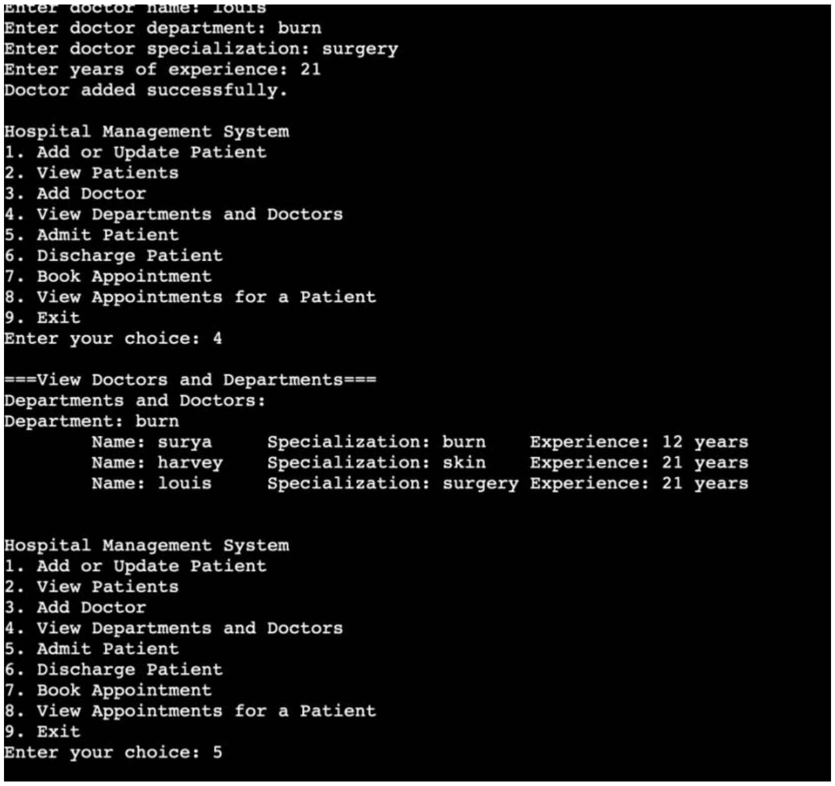
```
Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 1

===Add A Patient===
Enter patient name: smriti
Enter patient age: 22
Enter patient diagnosis: heart
Patient added successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 1

===Add A Patient===
Enter patient name: smriti
Patient already exists. Updating details.

===Update a Patient===
Enter patient age: 23
Enter patient diagnosis: burn
```



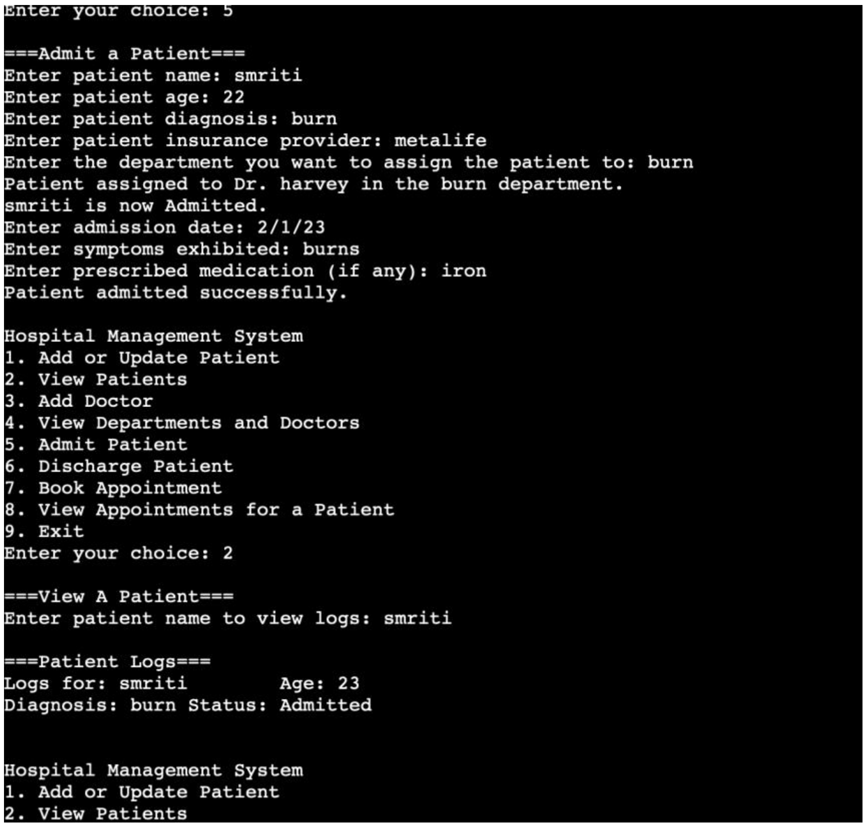
```
Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 2

===View A Patient===
Enter patient name to view logs: smriti

===Patient Logs===
Logs for: smriti        Age: 23
Diagnosis: burn Status: Registered


Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 3

===Add A Doctor===
Enter doctor name: surya
Enter doctor department: burn
Enter doctor specialization: burn
```



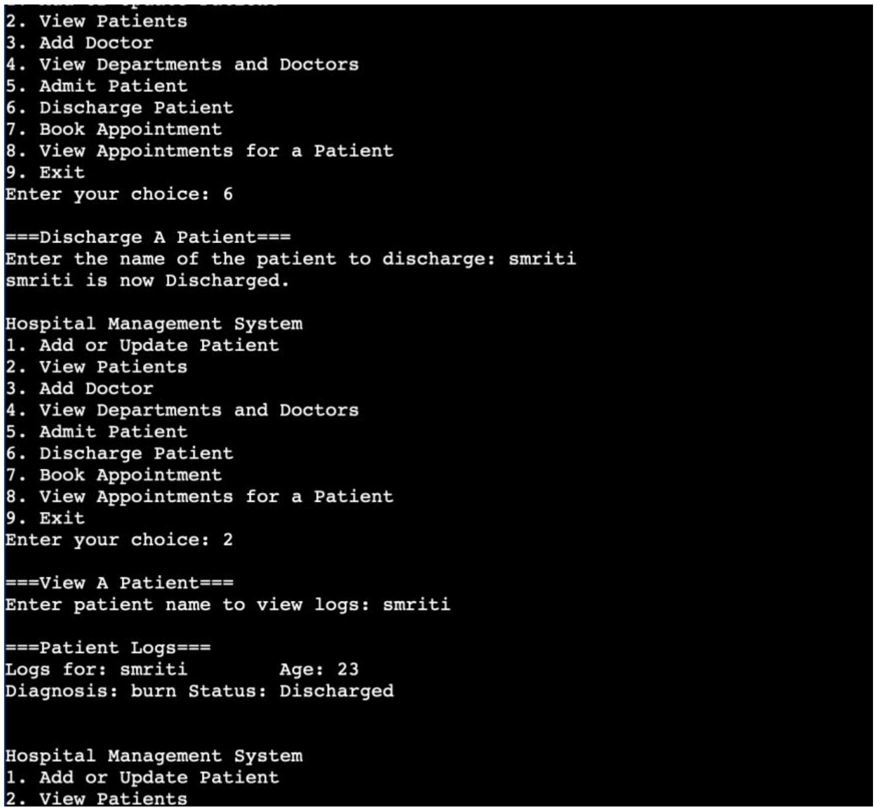
```
Enter doctor department: burn
Enter doctor specialization: burn
Enter years of experience: 12
Doctor added successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 3

===Add A Doctor===
Enter doctor name: harvey
Enter doctor department: burn
Enter doctor specialization: skin
Enter years of experience: 21
Doctor added successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 3

===Add A Doctor===
Enter doctor name: louis
```



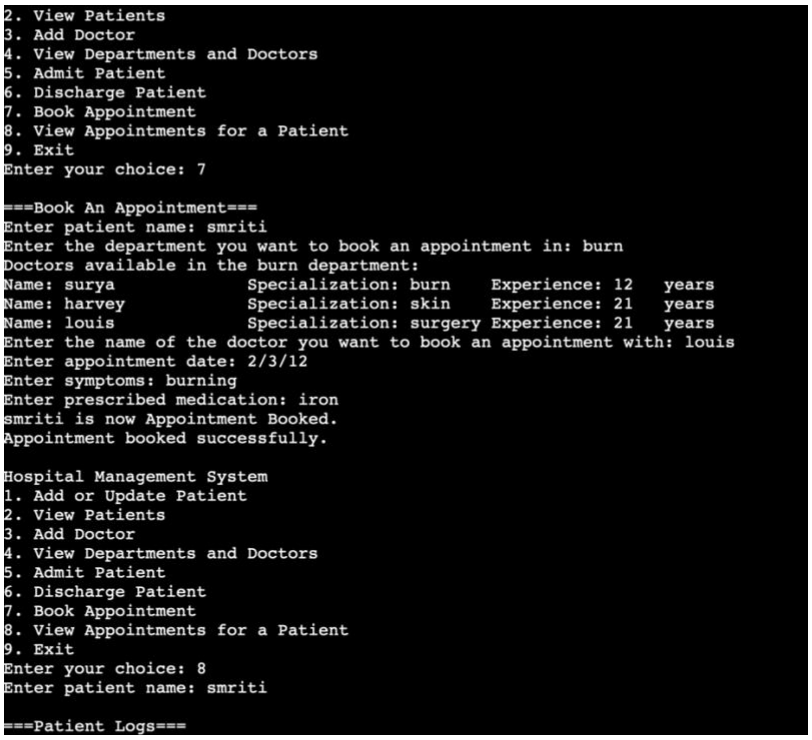
```
Enter doctor name: louis
Enter doctor department: burn
Enter doctor specialization: surgery
Enter years of experience: 21
Doctor added successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 4

===View Doctors and Departments===
Departments and Doctors:
Department: burn
        Name: surya     Specialization: burn    Experience: 12 years
        Name: harvey    Specialization: skin    Experience: 21 years
        Name: louis     Specialization: surgery Experience: 21 years


Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 5
```



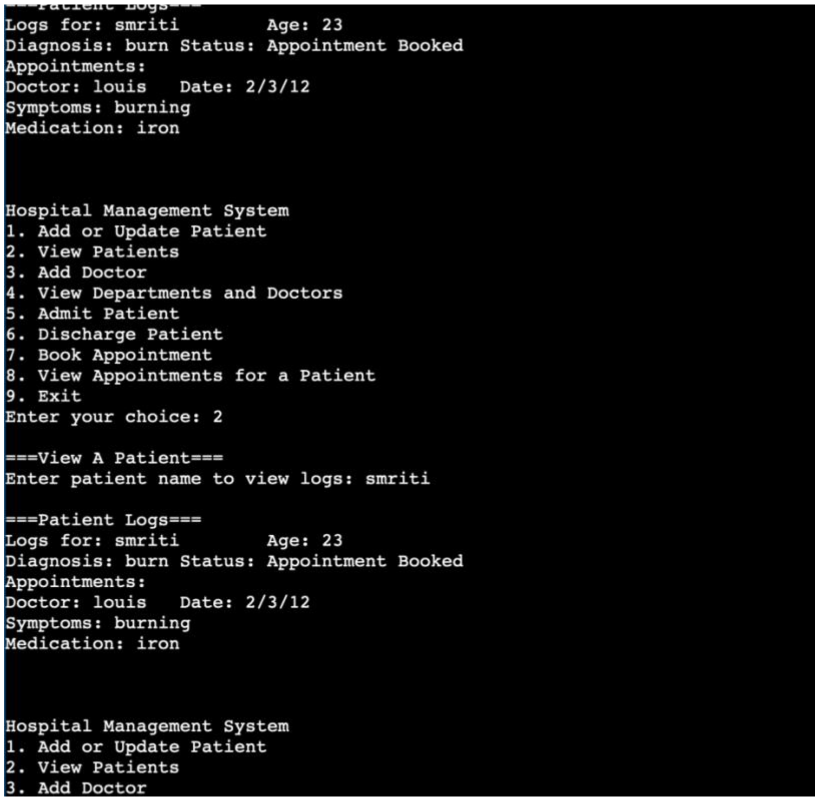
```
Enter your choice: 5

===Admit a Patient===
Enter patient name: smriti
Enter patient age: 22
Enter patient diagnosis: burn
Enter patient insurance provider: metalife
Enter the department you want to assign the patient to: burn
Patient assigned to Dr. harvey in the burn department.
smriti is now Admitted.
Enter admission date: 2/1/23
Enter symptoms exhibited: burns
Enter prescribed medication (if any): iron
Patient admitted successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 2

===View A Patient===
Enter patient name to view logs: smriti

===Patient Logs===
Logs for: smriti        Age: 23
Diagnosis: burn Status: Admitted


Hospital Management System
1. Add or Update Patient
2. View Patients
```



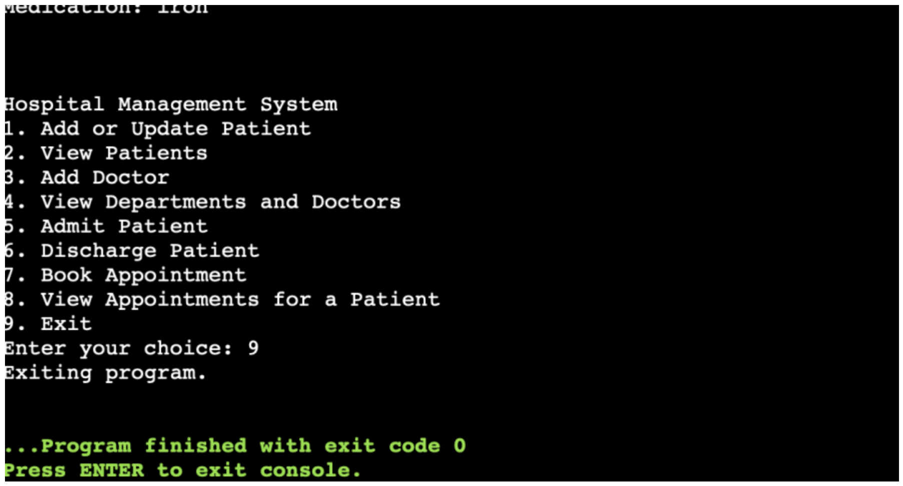
```
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 6

===Discharge A Patient===
Enter the name of the patient to discharge: smriti
smriti is now Discharged.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 2

===View A Patient===
Enter patient name to view logs: smriti

===Patient Logs===
Logs for: smriti        Age: 23
Diagnosis: burn Status: Discharged


Hospital Management System
1. Add or Update Patient
2. View Patients
```



```
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 7

===Book An Appointment===
Enter patient name: smriti
Enter the department you want to book an appointment in: burn
Doctors available in the burn department:
Name: surya            Specialization: burn    Experience: 12   years
Name: harvey           Specialization: skin    Experience: 21   years
Name: louis            Specialization: surgery Experience: 21   years
Enter the name of the doctor you want to book an appointment with: louis
Enter appointment date: 2/3/12
Enter symptoms: burning
Enter prescribed medication: iron
smriti is now Appointment Booked.
Appointment booked successfully.

Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 8
Enter patient name: smriti

===Patient Logs===
```



```
Logs for: smriti        Age: 23
Diagnosis: burn Status: Appointment Booked
Appointments:
Doctor: louis   Date: 2/3/12
Symptoms: burning
Medication: iron



Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 2

===View A Patient===
Enter patient name to view logs: smriti

===Patient Logs===
Logs for: smriti        Age: 23
Diagnosis: burn Status: Appointment Booked
Appointments:
Doctor: louis   Date: 2/3/12
Symptoms: burning
Medication: iron



Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
```



```
Medication: Iron



Hospital Management System
1. Add or Update Patient
2. View Patients
3. Add Doctor
4. View Departments and Doctors
5. Admit Patient
6. Discharge Patient
7. Book Appointment
8. View Appointments for a Patient
9. Exit
Enter your choice: 9
Exiting program.


...Program finished with exit code 0
Press ENTER to exit console.
```