Hospital Management System

OBJECT ORIENTED PROGRAMMING

GROUP MEMBERS

Muhammad Ali Siddiqui BSCS-10
Muhammad Sami BSCS-16

SUBMITTED TO MS. WAJEEHA

Overview

The Hospital Management System is a console-based Python project that simulates core hospital operations using Object-Oriented Programming (OOP). This project includes functionalities for managing patients and doctors, and generating bills. It is built using fundamental OOP principles such as inheritance, encapsulation, and composition.

OOP concept used

- **Class & Objects** : Used to model entities like Person, Patient, Doctor, and Billing.
- **Inheritance** : Patient and Doctor classes inherit from the Person class.
- **Encapsulation** : Data members are encapsulated within classes and accessed through methods.
- **Constructor** : Each class uses constructors to initialize attributes.
- **Method Overriding** : show_details() method is overridden in derived classes.
- **Composition** : Billing class uses Patient objects to generate bills.

Objectives

- Manage patient records (add/delete)
- Manage doctor records
- Assign doctors to patients
- Display all stored information
- Generate billing for patients

Tools & Technologies

- Language: Python 3
- IDE: Visual Studio Code
- Modules: Custom OOP modules – person, patient, doctor, billing
- Libraries: No external libraries used
- Execution: Terminal/Command Line based

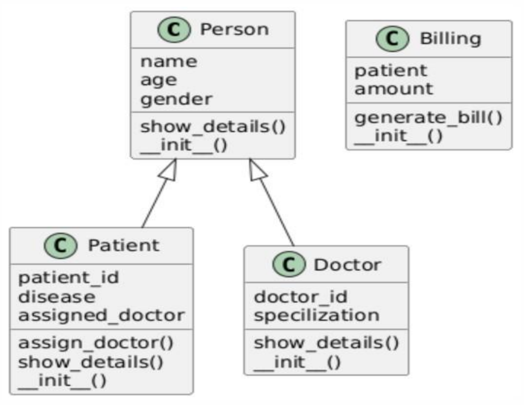
Modules Description

main.py	The main entry point; handles the user menu
person.py	Base class for both patients and doctors
patient.py	Subclass of Person, holds patient-specific data
doctor.py	Subclass of Person, stores doctor information
billing.py	Generates bills for a specific patient
main_gui.py	The main entry point; handles the user menu

System Design

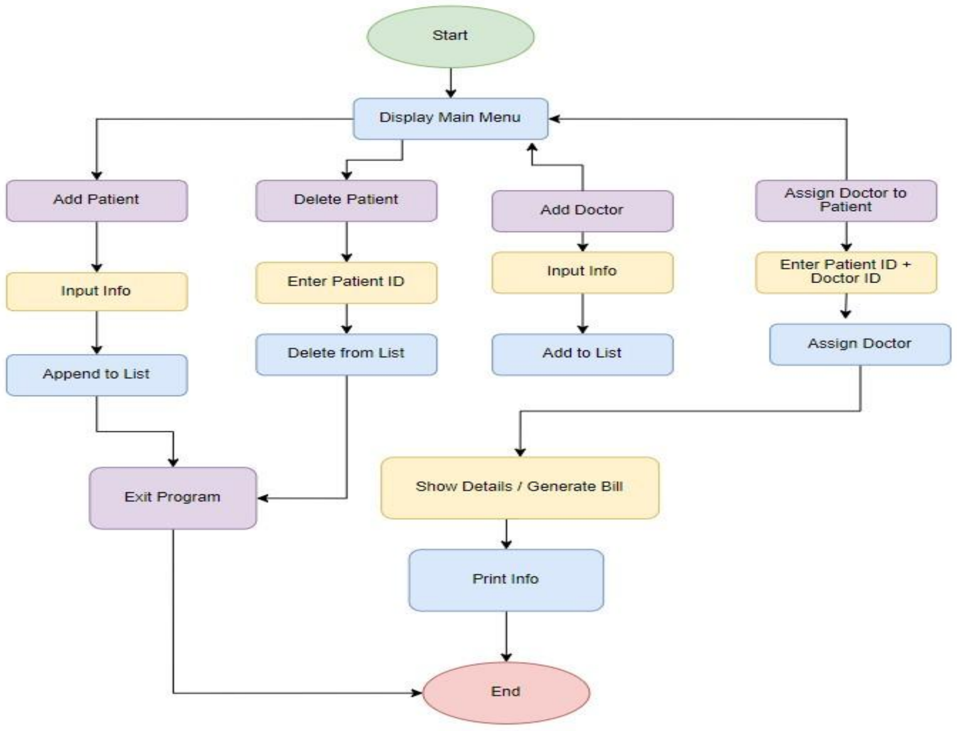
The system follows a linear flow and object-oriented class hierarchy. Below are the two design elements used:

Class Diagram



This diagram represents the OOP structure of the system. Person is the base class for Patient and Doctor. The Billing class uses a Patient instance to calculate the total bill, demonstrating **composition**.

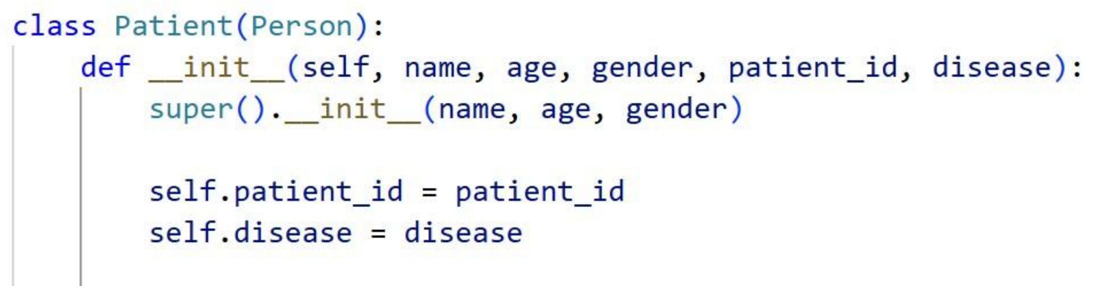
Flowchart



The above flowchart outlines the linear interaction model of the Hospital Management System. Upon launching the application, the user is presented with a menu offering several options: adding patients or doctors, assigning doctors to patients, displaying existing records, and generating patient bills. Each menu option leads to a specific functionality implemented via object oriented classes and methods.

Implementation

Below is a simplified version of class usage from the project:



```
class Patient(Person):
    def __init__(self, name, age, gender, patient_id, disease):
        super().__init__(name, age, gender)

        self.patient_id = patient_id
        self.disease = disease
```

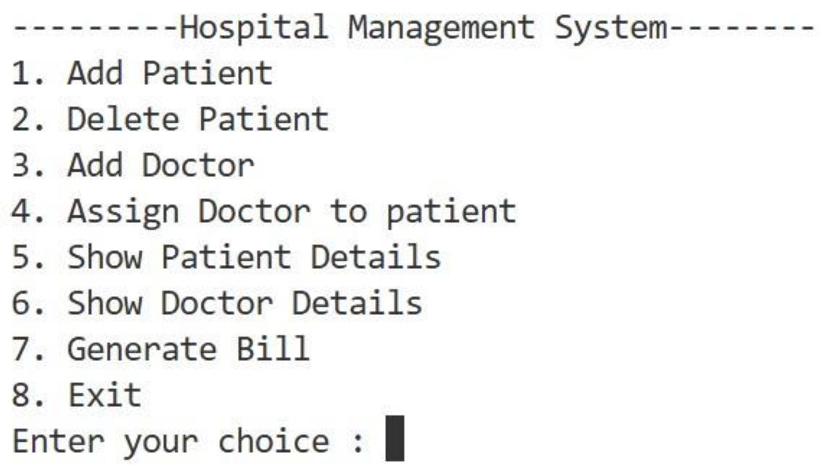
The user interacts through a terminal-based menu:
print ("1. Add Patient")
...
choice = input ("Enter your choice: ")

All records are stored in memory using lists; persistent storage (files or database) is not used in this version.

Screenshots

Below are screenshots showcasing the key functionalities of the Hospital Management System For both GUI and CLI :

Picture showing the menu options like:

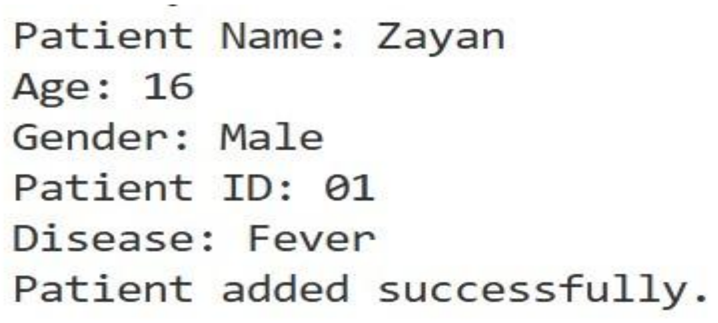


```
---------Hospital Management System--------
1. Add Patient
2. Delete Patient
3. Add Doctor
4. Assign Doctor to patient
5. Show Patient Details
6. Show Doctor Details
7. Generate Bill
8. Exit
Enter your choice :
```

This demonstrates the main menu of the Hospital Management System.

Add Patient

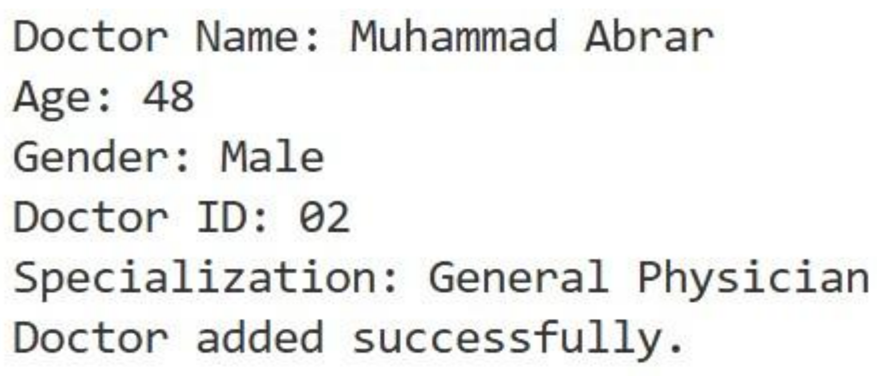
Picture of user input for adding a patient*:*



```
Patient Name: Zayan
Age: 16
Gender: Male
Patient ID: 01
Disease: Fever
Patient added successfully.
```

Add Doctor

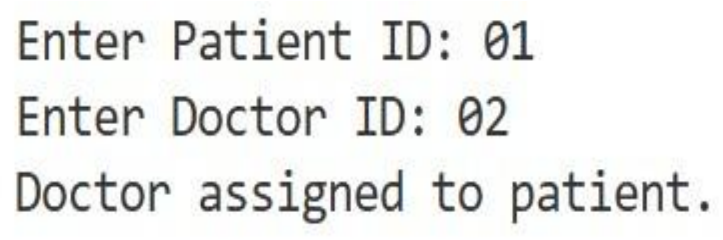
Picture shows of user input for adding a doctor:



```
Doctor Name: Muhammad Abrar
Age: 48
Gender: Male
Doctor ID: 02
Specialization: General Physician
Doctor added successfully.
```

Assign Doctor to Patient

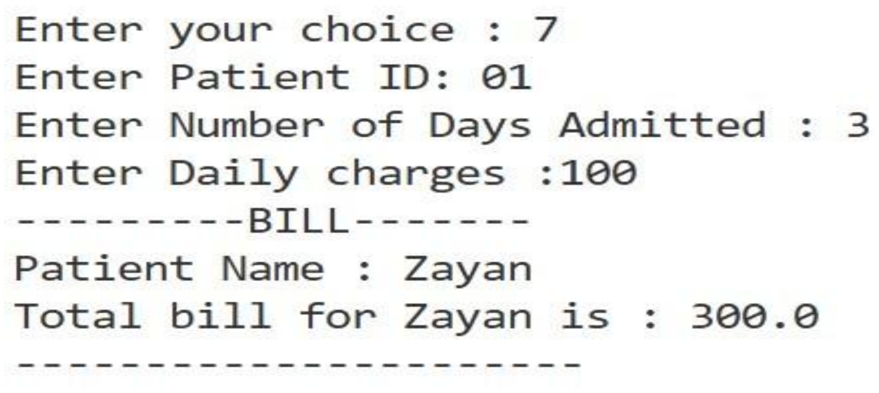
Picture showing doctor assignment:



```
Enter Patient ID: 01
Enter Doctor ID: 02
Doctor assigned to patient.
```

Generate Bill

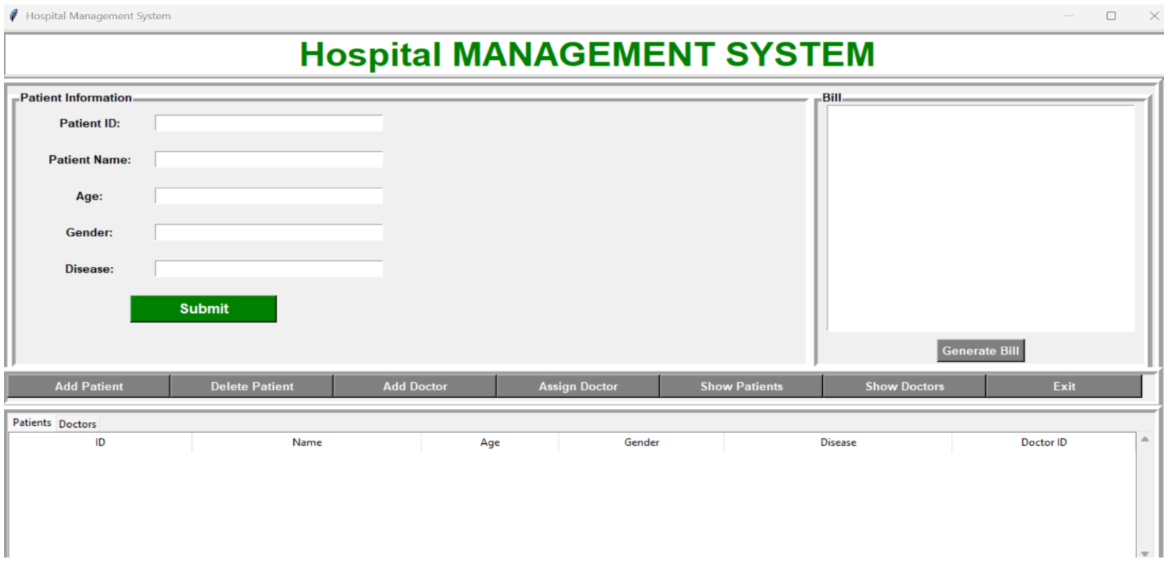
Screenshot showing bill calculation:



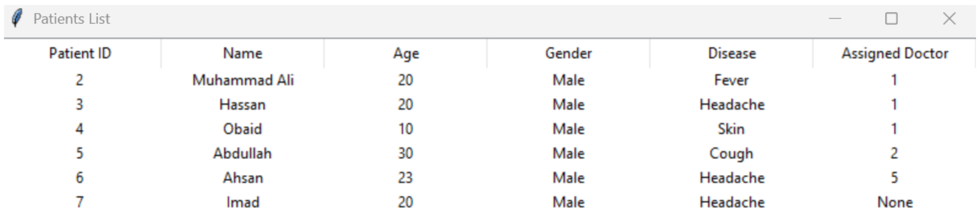
```
Enter your choice : 7
Enter Patient ID: 01
Enter Number of Days Admitted : 3
Enter Daily charges :100
---------BILL-------
Patient Name : Zayan
Total bill for Zayan is : 300.0
----------------------
```

GUI Screenshot:

This is the GUI main interface of Hospital management system :



This shows the different options add/delete patient, add doctor, assign doctor to patient and view details of patient and assigned doctor. This is the comprehensive gui along side with database connectivity.


Patients List

Patient ID	Name	Age	Gender	Disease	Assigned Doctor
2	Muhammad Ali	20	Male	Fever	1
3	Hassan	20	Male	Headache	1
4	Obaid	10	Male	Skin	1
5	Abdullah	30	Male	Cough	2
6	Ahsan	23	Male	Headache	5
7	Imad	20	Male	Headache	None

Conclusion

This project effectively demonstrates the real-world application of Object Oriented Programming principles in Python. It helped solidify understanding of core concepts such as inheritance, encapsulation, and modular design. The system structure is highly extendable, making it a strong base for future enhancements like GUI development, database integration, and user authentication.

References

- Python Official Documentation: https://docs.python.org/3/
- Class materials and course lectures
- Stack Overflow for syntax and debugging support