Hospital Management System

Objective:

The purpose of this assignment is to design and implement a **Hospital Management System (HMS)** using **SOLID** principles. By following these principles, you will ensure that the system's architecture is modular, flexible, maintainable, and easy to extend. This assignment will help you understand how to apply each of the five SOLID principles to real-world software development problems and **create a software architecture** for a hospital management system that can evolve over time.

Context:

SOLID is an acronym for five design principles that promote maintainable and scalable code:

1. **S** - Single Responsibility Principle (SRP)
2. **O** - Open/Closed Principle (OCP)
3. **L** - Liskov Substitution Principle (LSP)
4. **I** - Interface Segregation Principle (ISP)
5. **D** - Dependency Inversion Principle (DIP)

For this assignment, you will be tasked with designing and implementing a **Hospital Management System (HMS)**, focusing on key aspects such as patient management, appointment scheduling, billing, and staff management. The system must be scalable and extensible to accommodate future requirements, such as adding new services (e.g., insurance integration, medical history tracking, or emergency care management).

Scenario:

You are building a **Hospital Management System (HMS)** for a healthcare provider. The system must support the following functionalities:

- **Patient Management**: Register patients, update patient records, manage medical history, and retrieve patient details.
- **Appointment Scheduling**: Allow patients to book appointments with doctors, update appointment status, and manage doctor availability.
- **Billing**: Handle patient billing, payment processing, and generate invoices.
- **Staff Management**: Manage doctor and nurse information, scheduling, and workload tracking.

The system should be designed to be modular, flexible, and extendable. For example, adding new features such as inventory management, prescription management, or medical tests should be easy without requiring significant changes to the existing codebase.

Assignment Tasks:

***Task 1*: Software Architecture Design**

1. **Thoroughly describe the context** of the **Hospital Management System (HMS)** based on the basic information provided. This includes clearly identifying functional and non-functional requirements, outlining the objectives and scope of the project, etc.
2. **Compare and choose a suitable pattern** to design a software architecture for the **HMS.**
3. **Create the software architecture** for the **HMS:** Identify and describe the main components of the system, its functionality, and how it interacts with other components.
4. **Create UML Class Diagrams**: Design class diagrams that show the relationships between the main components (e.g., Patient, Doctor, Appointment, Bill, etc.). Include interfaces and abstract classes where applicable to promote flexibility and extensibility.
5. **Apply SOLID Principles**: explain how the **SOLID** principles have been applied in your design. Specifically, highlight how you have ensured:
    - **Single Responsibility**: Each class and module has one distinct responsibility.
    - **Open/Closed**: The system is open for future extensions but closed for modification.
    - **Liskov Substitution**: Subtypes can be used interchangeably with their base classes without affecting functionality.
    - **Interface Segregation**: Small, focused interfaces are created for each service.
    - **Dependency Inversion**: High-level modules depend on abstractions (interfaces), not concrete implementations.
6. **Future Extensibility**: Discuss how the system is designed to allow for easy extension, such as adding new functionalities like lab tests, insurance claim processing, or patient history management, without violating SOLID principles.

***Task 2*: Code Implementation** (choose **at least one main module** to implement)

1. **Implement Core Functionalities**: Implement key system features based on your design, ensuring the code adheres to the SOLID principles. Some core components to implement include:
    - **Patient Management**: Implement classes for handling patient registration, updating patient records, and retrieving patient details.
    - **Appointment Scheduling**: Implement classes for scheduling, rescheduling, and canceling appointments with doctors. Ensure that you track the availability of doctors.

- **Billing**: Implement classes to handle patient billing, process payments, and generate invoices.
- **Staff Management**: Implement classes for managing hospital staff (e.g., doctors and nurses), including scheduling and assigning tasks.

2. **Apply SOLID in Code**:
   - **Single Responsibility Principle (SRP)**: Each class should handle a single task. For example, the BillingService class should not be responsible for patient management or appointment scheduling.
   - **Open/Closed Principle (OCP)**: Design the system in a way that new features can be added without modifying existing code. For example, you could implement InsuranceBillingService to handle insurance claims without changing the existing BillingService.
   - **Liskov Substitution Principle (LSP)**: Ensure that classes can be substituted with their derived types. For example, if you have an abstract Staff class, the derived classes Doctor and Nurse should behave correctly wherever Staff is used.
   - **Interface Segregation Principle (ISP)**: Define focused, small interfaces. For example, create separate interfaces for PatientManagement, AppointmentManagement, and BillingManagement so that services only implement the methods relevant to their functionality.
   - **Dependency Inversion Principle (DIP)**: Depend on abstractions, not concrete classes.

***Task 3*: Documentation and Reporting**

1. **Design Documentation**:
   - Provide a clear explanation of your design choices and how each SOLID principle was applied.
   - Include UML class diagrams that illustrate the structure and relationships of your system's components.
   - Discuss how the design can be easily extended to accommodate future features without requiring changes to existing code (e.g., adding new staff roles, implementing a notification system, etc.).
2. **Reflection Report**:
   - Write a brief reflection on how applying SOLID principles helped you improve the design of the Hospital Management System. Discuss challenges you faced and how adhering to these principles made your system more modular, maintainable, and extensible.

Submission Instructions:

- Submit a detailed report containing a GitHub URL (source code) on the LMS website.