SE 540

Final Project

Project Overview:

In this project, we will be applying the fundamental principles and design patterns learned in this course. This project is designed to provide you with a hands-on experience in building a robust and maintainable software application while incorporating SOLID principles, Singleton, Factory, and Builder design patterns. You will also be using github, and unit testing which we will cover in the upcoming weeks.

With this project, you will have the opportunity to create a simplified yet comprehensive **shopping system**. This system will allow customers to browse products, add items to their shopping cart, place orders, and even simulate the payment process.

Project Components:

- User Authentication:
    - Create a user authentication system where customers can log in, and their cart data will be associated with their accounts.

- Product Catalog:
    - Develop a catalog of products with essential information.
    - This catalog can be stored in memory or loaded from a file or database.

- Order Processing:
    - Implement the process of placing orders, utilizing the Singleton and Factory patterns for cart management and product creation.

- Payment Processing:
    - Simulate a payment processing system (e.g., using a mock payment gateway).
    - Keep this module separate from the core shopping functionality.

- Logging:
    - Create a logging system to record important events and transactions.

Implementing SOLID Principles

- Single Responsibility Principle (SRP):
    - Ensure that each class has a single responsibility.
    - For instance, create separate classes for cart management, user authentication, product catalog, order processing, and logging.

- Open/Closed Principle (OCP):
    - Design classes and modules that are open for extension but closed for modification.
    - For example, create abstract classes or interfaces to represent products and implement concrete product classes that can be extended without modifying existing code.

- Liskov Substitution Principle (LSP):
    - Make sure that subclasses can be substituted for their base classes without affecting the correctness of the program.
    - For instance, if you have a base Product class, ensure that derived classes (e.g., Electronics, Clothing) can be used interchangeably.

- Interface Segregation Principle (ISP):
    - Create interfaces that are specific to the needs of the client.
    - Avoid fat interfaces with methods that are not relevant to all implementing classes.

- Dependency Inversion Principle (DIP):
    - Utilize dependency injection to decouple high-level modules from low-level modules.
    - For instance, inject dependencies like the product catalog and the shopping cart into the order processing module.

Implementing Design Patterns

- Singleton Design Pattern:
    - Implement the Singleton pattern for the shopping cart.
    - Ensure that there is only one instance of the cart throughout the application.

- Factory Design Pattern:
    - Create a ProductFactory that encapsulates the creation of different types of products.
    - Use this factory to create product objects for the catalog.

- Builder Design Pattern:
    - Implement a CartBuilder class to construct shopping carts.
    - Provide a fluent interface to add and remove items from the cart, set customer information, and finalize the order.

Testing

Develop unit tests for critical components, such as the shopping cart, product catalog, and order processing.

Submission

1. UML diagrams. (%10)
2. Your full code zipped. (%50)
3. A link to the github of the project. (%5)
4. README file explaining how to run the application. (%3)
5. Powerpoint presentation that includes an overview of the system. (%2)
6. A video presentation (%30)
    a. This will start with you going over the system design and why did you make the design decisions for each part.
    b. A walk through of your running application.

General Notes

- The project could be console based, GUI based (%5 bonus), or you can make it a web application(%5 bonus) if you prefer.
- You can use files or a database for storage. If you are using a database, the full database code must be uploaded with your submission.
- There are a lot of design decisions you need to make; all these decisions need to be explained in your video.