Project Documentation: Flower shop

Student: Asset Aruzhan

Group: SE-2210

**1. Introduction**

**1.1 Project Overview**

A JavaScript-based e-commerce platform called ShopMaster is used to handle products in an online store. The purpose of this project is to demonstrate how different design patterns may be used in software development. ShopMaster shows how to use design patterns to improve code maintainability, flexibility, and scalability by applying the Strategy, Singleton, Decorator, Adapter, Factory, and Observer patterns.

**1.2 Importance of Design Patterns**

Software developers need to use design patterns because they provide well-tested answers to common design issues. They give developers a common language, enhance the quality of the code, and encourage the division of responsibilities in software architecture. ShopMaster wants to demonstrate the importance of design patterns in producing software that is expandable, reliable, and maintainable.

**1.3 Objectives**

The main objectives of ShopMaster are as follows:

Implement and demonstrate the use of key design patterns in a real-world application.

Develop a functional e-commerce system for product management.

Showcase the importance of design patterns in solving software development challenges.

Provide a foundation for future enhancements and improvements.

**2. Project Requirements**

**2.1 Implemented Design Patterns**

ShopMaster uses the subsequent design patterns:

Strategy pattern: product pricing strategies are chosen dynamically.

Singleton pattern: guarantees that the inventory management of the shop has a single instance.

Decorator Pattern: Adapts features to products dynamically.

Adapter Pattern: Retrieves product details by interacting with external systems.

Factory Pattern: Produces a variety of goods, such as apparel and electronics.

Observer Pattern: Tracks and reports variations in the cost of goods.

**2.2 Features**

Shop Inventory (Singleton Pattern)

Manages the shop's inventory with a single instance.

Provides methods to add, remove, and retrieve products.

Product Factory (Factory Pattern)

Creates different types of products: electronics and clothing.

Products

Base class for products, with id, name, and price attributes.

Electronics and Clothing subclasses with a type attribute.

ProductDecorator (Decorator Pattern)

Dynamically adds features to products.

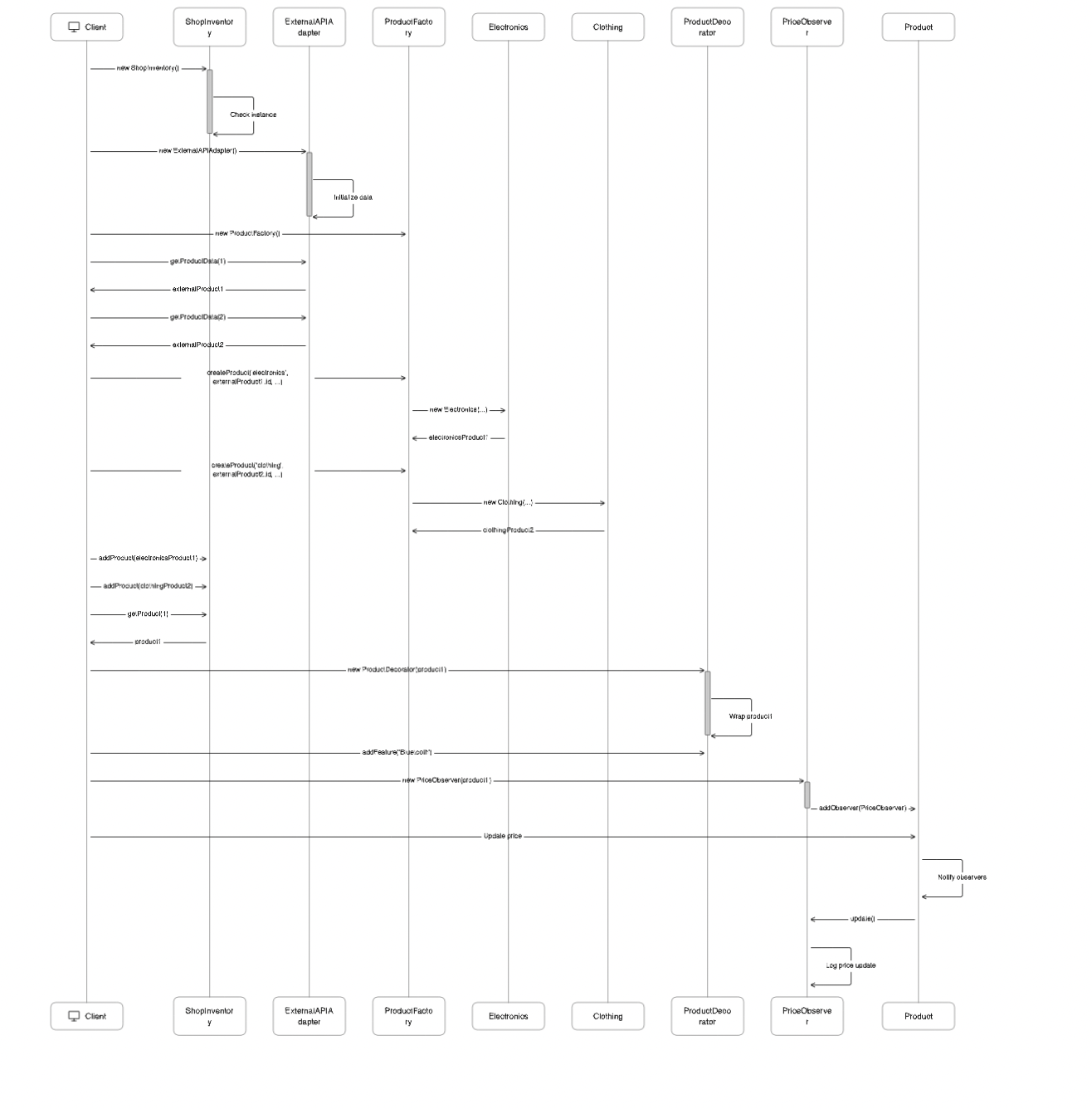
ExternalAPIAdapter (Adapter Pattern)

Adapts external API data to fetch product details.

PriceObserver (Observer Pattern)

Observes and notifies changes in product prices.

**3. UML Diagrams**



**4. Conclusion**

**4.1 Summary**

ShopMaster is an effective example of how several design patterns may be implemented in an e-commerce platform. It demonstrates how well these patterns work to enhance code organization, reuse, and maintainability.

**4.2 Challenges Faced**

ShopMaster is an effective example of how several design patterns may be implemented in an e-commerce platform. It demonstrates how well these patterns work to enhance code organization, reuse, and maintainability.

**4.3 Future Improvements**

Future enhancements may include a user interface, a database for persistent data storage, and additional design pattern integrations for more complex scenarios.