Part IV: Reflect upon and document your experience.

Initial Design and Implementation:
Initial Design and Implementation: The original design and execution concentrated on developing a basic café management system that enabled customers to purchase drinks (coffee) and food items that were pre-packaged. This project provides a fantastic opportunity to explore and utilise design patterns in C++. The first design (Parts I and II) offered a strong basis with evident separation of duties (Drink, Food, CoffeeMaker, FoodMaker classes) and a central coordinator (Manager class adopting the Mediator pattern). The modular structure made it possible to extend the system in Part III.

Extending the Initial Design:
The significant feature in Part III was enabling consumers to pick between "Have Here" and "Takeaway" orders and selecting between three types of milk for WhiteCoffee. The Manager, Order and Host class were changed to support this new order type argument in their takeOrder function. This patch conforms to the Open/Closed Principle, allowing for expansion without altering current code. Whereas WhiteCoffee class was modified to handle different types of milk, CoffeeMaker class was modified to handle selecting milk type during creating WhiteCoffee object and adjustment was made to Manager to facilate the selection of milk during ordering process. This patch conforms to Factory Method Pattern allowing for greater flexibility and maintainability by encapsulating object creating logic which can be used for easy extension to support new food and drink items.

Challenges and Difficulties:
Handling user input and error checking proved problematic, especially with the new order type. Implementing validation logic while preserving a user-friendly interface demands a thorough grasp of C++ control flow and error management.

Rewarding Aspects:
The most fun element was establishing and executing the system architecture. Using the Facade, Mediator, and Factory patterns generated a well-organized, modular application that is easy to understand, maintain, and build. Working with unique pointers and dynamic memory management also helped me enhance my contemporary C++ skills.

Lessons Learned:
This project substantially increased my knowledge of how to apply design patterns to overcome software difficulties. It stressed the requirement of beginning with a flexible, maintainable, and scalable design. The ability to grasp existing code and make informed design choices emphasized the necessity of excellent communication and documentation.

Overall, as an undergraduate student studying design patterns and their implementation in C++, this project was a great experience that enabled me to employ design patterns in a real-world situation.