Report: E-commerce Product Management System

1. Object-Oriented Analysis (OOA)

Step 1: Identify Objects
+Product (product)
Electronics (electronic product, inherits from Product)
ShoppingCart (shopping cart)
Order (order – can be extended later)
InventoryList (inventory list – using template)
Discountable (discount interface)

Step 2: Identify Attributes
Product: id, name, price, stock
Electronics: inherits from Product, adds warrantyMonths
ShoppingCart: list of products + quantity
InventoryList<T>: vector<T> items
Order: list of products, total amount (extended)

Step 3: Identify Methods
Product: display(), applyDiscount(), get/set for attributes
Electronics: display() override
ShoppingCart: addToCart(), showCart(), operator+=
InventoryList<T>: addItem(), showAll(), getItems()
Discountable: applyDiscount() (pure virtual)

Step 4: Identify Inheritance Relationships
Electronics generalization from Product
Product realization of Discountable interface
ShoppingCart uses (composition) Product
InventoryList is a template for managing Product

2. Class Design and Explanation

Inheritance: Class Electronics is built based on class Product, extending an additional attribute for warranty time (warrantyMonths). This is an "is-a" relationship, meaning Electronics is also a type of Product.

Operator Overloading: The program defines the equality comparison operator (==) in class Product to compare two products based on ID. Additionally, class ShoppingCart overloads the plus equals operator (+=) to add a product to the cart. This makes the operation of adding products more natural and closer to mathematical notation.

Interface (Abstract Class): Class Discountable defines a pure virtual method applyDiscount(rate). All classes inheriting from Product are required to implement this method, ensuring polymorphism when applying discounts to various types of products.

Template Class: Class InventoryList<T> is implemented as a template to store different types of objects, such as Product, Electronics, or others in the future. Using templates makes the source code more flexible and highly reusable.

3. Code Walkthrough

Base Class Product: Stores basic product information including product code (id), name (name), price (price), and stock quantity (stock). At the same time, this class implements the applyDiscount method to adjust the selling price when applying a discount.

Class Electronics: Inherits from Product and overrides the display method to show additional information about the warranty period.

Operator Overloading:
In class Product, the equality comparison operator (==) is used to compare two products based on ID.
In class ShoppingCart, the plus equals operator (+=) is used to add a product to the cart if it is in stock.

Template InventoryList<T>: Implemented with a vector to manage the list of products. The main methods include addItem, showAll, and getItems.

Class ShoppingCart: Instead of just storing a list of Product pointers, the program implements an additional CartItem structure with two components: Product pointer and quantity (quantity). This allows the cart to manage product quantities and automatically deduct from stock when adding products.

4. Test Results

Case 1: Add Product to Inventory
Input: Product id = P01, name = Book, price = 50, stock = 10
Output: Inventory displays the newly added product.
Case 2: Add Electronic Product
Input: Electronics id = E01, name = Laptop, price = 1500, stock = 5, warranty = 24
Output: Inventory displays Laptop with 24-month warranty.
Case 3: Add to Cart
Input: Select P01, quantity = 3
Output: Cart displays 3 Books, stock reduced to 7.
Case 4: View Cart Multiple Times
Output: Still displays the correct quantity, no crash.

5. UML Diagrams

Class Diagram (description):
Product: id, name, price, stock, +display(), +applyDiscount(), +operator==
Electronics: +warrantyMonths, override display()
Discountable: +applyDiscount() (abstract)
InventoryList<T>: +addItem(), +showAll(), +getItems()
ShoppingCart: +addToCart(), +showCart(), operator+=
Sequence Diagram (operation "Add to cart"):
User enters productID + quantity

System searches for the product in InventoryList
If found, call ShoppingCart.addToCart()
ShoppingCart checks stock, deducts quantity, adds to cart
Displays notification “Added successfully”

6. Using LLM (AI Tool)

During the process of completing the assignment, I used ChatGPT to:
Brainstorm ideas on how to implement the template InventoryList.
Reference how to write operator overloading for the cart.
I asked “Suggest a template class for inventory in C++” and then adjusted it to fit the project requirements.