E-Commerce Management System Report

Step 1: OOA Analysis

Idea:
A simple e-commerce system that can manage products, inventory, shopping cart, discounts, orders, and extended features (loyalty points, product reviews, order tracking).

Objects (Nouns)

- **Product**
- **Electronics (subclass of Product)**
- **InventoryList**
- **ShoppingCart**
- **Discount (interface)**
- **Order**

Attributes

- **Product:** id, name, price, stock, reviews
- **Electronics:** adds warranty fee
- **InventoryList:** list of products
- **ShoppingCart:** list of items, total price, loyalty points
- **Order:** id, status (pending, shipped, delivered)

Methods (Verbs)

- **Product:** update stock, add review, show reviews, display info
- **Electronics:** override displayInfo, override updateStock (with warranty fee)
- **InventoryList:** add, remove, showAll
- **ShoppingCart:** add product (operator+=), display cart, apply discount
- **Order:** update status, track order

Inheritance

- `Discount` as an **interface** → implemented by `Product` and `ShoppingCart`
- `Electronics` inherits `Product`
- Virtual methods (`displayInfo`, `updateStock`) allow polymorphism

Step 2: Class Design

- **Encapsulation:** attributes are protected/private, accessed by getters
- **Inheritance:** Electronics extends Product, override methods
- **Interface (Discount):** allows applying discounts to both products and shopping carts
- **Polymorphism:** virtual functions make the system flexible

---

Step 3: Code Walkthrough

- **Product:** base class for all products. Supports stock management, reviews, and discount application.
- **Electronics:** specialized product with warranty fee, overrides stock update & display info.
- **InventoryList (template):** manages generic product collections.
- **ShoppingCart:** operator overloading (+=) to add products, manages total cost and loyalty points, applies discount.
- **Order:** tracks order status, allows updates.

Step 4: Test Results

Note:

👉 All **test cases** below were **100% generated by AI (ChatGPT)**.

👉 Only the **code** was written by me.

Example Output

```
===== TEST CASES START =====
===== Inventory List =====
Product Info
Name: Book
Price (VND): 100000
Stock: 5
...
p1 == p2 ? NO
p1 == p3 ? YES
Book added to cart
Laptop added to cart
LIST SHOPPING
...
Discount applied to cart: -2000000 VND
Reviews for Book:
- Good quality!
- Affordable price.
Update stock for Book:
Sold 2 units of Book
Remaining: 3
Not enough stock for product: Book Available: 3 Requested: 10
Update stock for Laptop (Electronics override):
Sold 1 units of Laptop
Remaining: 1
Fee applied : 500000 VND
Order#1
Status: Pending
Order#1
Status: Shipped
Order#1
Status: Delivered
===== TEST CASES END =====
```

Observations:

- Inventory list shows all products correctly.
- Operator== compares products by ID.
- ShoppingCart accumulates products, loyalty points increase by 10 each time.
- Discount applied successfully reduces total.
- Product reviews stored & displayed.
- Stock update works, Electronics adds warranty fee.
- Order tracking updates states (Pending → Shipped → Delivered).

Step 5: LLM Usage

I used ChatGPT in these ways:

- **Brainstorming ideas**:
  Suggested possible additional features like Loyalty Points, Order Tracking, and Product Reviews.
- **Debugging guidance**:
  Asked how to override virtual functions (`updatestock`, `displayInfo`).
- **Syntax clarifications**:
  Asked about operator overloading (`operator+=`) for adding products to the cart.
- **Report drafting help**:
  Used ChatGPT to format this report in Markdown for easier PDF export.
- **Test Cases:**
  All test cases in this report were generated entirely by ChatGPT.

Important:
All **code** was written by me.
The **LLM only provided brainstorming support, structural guidance, and generated test cases**.

Example prompt I asked:

> "Give me some additional features to add into a simple e-commerce system besides the required ones."

It suggested **Loyalty Points, Order Tracking, and Product Reviews** → I adapted these directly into my project.