**Report: Ecommerce Store Management System**

**Name:** Nguyễn Huỳnh Minh Tuyết  
**Student ID:** 24110144

**1. OOA Analysis (4-step model)**

**Step 1: Identify objects (nouns)**

* Product
* CartItem
* Cart
* Customer
* Operation (Management System)

**Step 2: Identify attributes (descriptive nouns)**

* Product: productID, productName, price, stock
* CartItem: product, quantity, subtotal
* Cart: cartID, list of CartItems, totalAmount
* Customer: name, customerID, cart
* Operation: list of products, list of customers

**Step 3: Identify methods (verbs)**

* Product: reduceStock(), getters/setters
* CartItem: setQuantity(), operator+= (increase quantity), getters
* Cart: operator+= (add item), operator== (compare carts), calculateTotal(), displayCart()
* Customer: getters, getCart()
* Operation: addProduct(), addCustomer(), displayProducts(), addToCart(), displayCustomerCart()

**Step 4: Inheritance relationships**

* The current system keeps inheritance simple.
* Relationships:
  + Cart contains multiple CartItems (composition).
  + Customer owns a Cart (aggregation).
  + Operation manages Products and Customers (association).
* Inheritance can be easily extended (e.g., ClothingProduct inheriting Product).

**2. Class Design & Why Inheritance is Used**

**Design idea:**  
The system models a small online shopping platform in C++.

* Product: stores product data.
* CartItem: represents a line item (product + quantity).
* Cart: manages multiple CartItems and calculates the total price.
* Customer: owns a single Cart.
* Operation: central controller that manages products, customers, and shopping actions.

**Why inheritance is used (extendable design):**

* Products could have specialized subtypes (e.g., ElectronicsProduct, ClothingProduct).
* This allows reuse of attributes/methods and specialization of behavior.

**Operator overloading:**

* Cart::operator+= to add items to a cart.
* Cart::operator== to compare two carts by total amount.

**3. Code Walkthrough**

* **Product:** includes attributes id, name, price, stock; reduceStock() decreases stock when purchased.
* **CartItem:** holds product, quantity, subtotal; operator+= increases quantity.
* **Cart:** stores a vector of CartItems; operator+= adds new items; calculateTotal() recalculates; operator== compares carts.
* **Customer:** contains name, ID, Cart.
* **Operation:** manages all products and customers; provides methods to add products, add customers, and process add-to-cart operations.
* **main():** demonstrates the system by creating products and customers, showing product list, adding items to carts, and displaying final results.

**4. Test Results**

**Sample console output:**

Available Products:

- P001 - T-Shirt (15$, stock: 50)

- P002 - Jeans (40$, stock: 30)

- P003 - Sneakers (60$, stock: 20)

Added to cart: T-Shirt x2

Added to cart: Sneakers x1

Added to cart: Jeans x3

Customer: Nguyen Van A

Cart ID: CART-C001

- T-Shirt x 2 = 30$

- Sneakers x 1 = 60$

Total: 90$

Customer: Le Thi B

Cart ID: CART-C002

- Jeans x 3 = 120$

Total: 120$

**Explanation:**

* Products are added successfully into the system.
* Customers add products to their carts using the overloaded += operator.
* Totals are calculated correctly and displayed.
* Product stock decreases after being purchased.

**5. LLM Usage (ChatGPT)**

During development, I used ChatGPT to assist with:

* Brainstorming the OOA analysis and class relationships.
* Suggesting ideas for operator overloading in Cart.
* Refining the structure of the report to match the required format.

**Example prompts (paraphrased):**

* “Write a simple Ecommerce system in C++ with Product, Cart, Customer, and operator overloading.”
* “Help me create a report in the same format as my Banking/Transportation System assignment.”

The final C++ code and report content were written and tested by me. ChatGPT provided structural suggestions and helped polish explanations.