Assignment: E-Commerce System (Case Study)

1. Problem Statement To design an object-oriented E-Commerce system that:

- **Displays available product categories to users**
- **Lists products under selected categories**
- **Allows users to add items to the shopping cart**
- **Processes orders and payments securely**
- **Confirms order placement and maintains order history**
- **Tracks order shipments and deliveries**
- **Provides customer support and reviews**

2. Solution Approach This system follows Object-Oriented Programming (OOP) principles, including:

- **Inheritance: Base class User is inherited by Customer and Admin.**
- **Encapsulation: Payment details and order history are private with controlled access.**
- **Polymorphism: Different payment methods (Credit Card, PayPal, UPI) have different processing mechanisms.**
- **Abstraction: Common functionalities like authentication and order management are defined in base classes.**
- **Composition: An order consists of multiple products, and a shopping cart is associated with a customer.**

Classes and Their Relationships

User :- **Base class for all users in the system.**

Customer :- **Stores customer details, shopping cart, and order history.**

Admin :- **Manages product categories, listings, orders, and inventory.**

Category :- **Represents different product categories.**

Product :- **Represents items available for sale with pricing and stock.**

Cart :- **Manages items selected by a customer before checkout.**

Order :- **Tracks orders placed, payment status, and delivery details.**

Payment :- **Handles transactions using different payment methods.**

Shipment :- **Manages delivery status and tracking information.**

Review :- **Allows customers to review and rate products.**

3. Algorithms and Pseudocode Each algorithm solves a core system function.

3.1. Display Available Categories Input:

Output: List of product categories

Algorithm:

Retrieve available product categories from the database.

Display the list to the user.

Pseudocode:

FUNCTION display_categories():

categories ← get_all_categories()

RETURN categories

Example: display_categories()

Output: ["Electronics", "Clothing", "Home Appliances"]

3.2. Display Products in a Category Input: categoryID

Output: List of products under the selected category

Algorithm:

Retrieve products from the selected category.

Display product details.

Pseudocode:

FUNCTION display_products(categoryID):

products ← get_products_by_category(categoryID)

RETURN products

Example: display_products("Electronics")

Output: [{"id": 101, "name": "Laptop", "price": 60000}, {"id": 102, "name": "Smartphone", "price": 30000}]

3.3. Add Product to Cart Input: customerID, productID, quantity

Output: Product added confirmation

Algorithm:

Check product availability.

Add product to customer's cart.

Update cart and return confirmation.

Pseudocode:

```
FUNCTION add_to_cart(customerID, productID, quantity):
  product ← find_product(productID)
  IF product.stock >= quantity:
    customer ← find_customer(customerID)
    customer.cart.add(product, quantity)
    RETURN "Product added to cart"
  ELSE:
    RETURN "Insufficient stock"
```

Example: add_to_cart(101, "Laptop123", 1)

Output: "Product added to cart"

3.4. Process Order Input: customerID

Output: Order confirmation

Algorithm:

1. Retrieve cart items.
2. Check stock availability.
3. Create an order and update stock.
4. Return confirmation.

Pseudocode:

```
FUNCTION process_order(customerID):
    customer ← find_customer(customerID)
    cart ← customer.get_cart()
    IF cart.is_empty():
        RETURN "Cart is empty"

    order ← create_order(customer, cart.items)
    cart.clear()
    RETURN "Order placed successfully"
```

Example: process_order(101)
Output: "Order placed successfully"

3.5. Process Payment Input: orderID, paymentMethod

Output: Payment confirmation

Algorithm:

Verify order details and amount.

Process payment via the selected method.

Update order status.

Pseudocode:

```
FUNCTION process_payment(orderID, paymentMethod):
    order ← find_order(orderID)
    IF order.status == "Pending":
        success ← paymentMethod.process(order.amount)
        IF success:
            order.update_status("Paid")
```

```
            RETURN "Payment successful"
        ELSE:
            RETURN "Payment failed"
    ELSE:
        RETURN "Order already paid"
```

Example: process_payment(5001, "CreditCard")

Output: "Payment successful"

3.6. Maintain Order and Customer History Input: customerID

Output: List of past orders

Algorithm:

Retrieve all past orders associated with the customer.

Display order details.

Pseudocode:

```
FUNCTION get_order_history(customerID):
    customer ← find_customer(customerID)
    orders ← customer.get_order_history()
    RETURN orders
```

Example: get_order_history(101)

Output: [{"orderID": 5001, "status": "Delivered", "total": 60000}]

4. Conclusion of E-Commerce System:

- Displaying available product categories
- Listing products under selected categories
- Shopping cart management
- Secure order processing and payments
- Order placement and history tracking
- Shipment tracking and customer reviews