**Case Study on E-Commerce System**

An E-Commerce System is a software platform that enables businesses and consumers to buy and sell products or services online. It includes various components such as product management, customer interactions, payment processing, and order fulfillment.

Example of How an E-Commerce System Works

1️.Customer searches for a product → System fetches product details from the database.  
2️.Customer adds products to the cart → Stock is checked before adding.  
3️.Customer proceeds to checkout → Payment gateway is called for transaction.  
4️.Order is confirmed & processed → Stock is updated, and order is sent for shipping.  
5️.Customer receives tracking details → Delivery is completed.

* A object oriented programming system for managing online shopping with products, customers, shopping carts, and order processing. Focuses on product hierarchies, order workflows, and payment processing.
* **Software Requirements for the E-Commerce System**

To develop and run this Python-based E-Commerce System efficiently, you need the following **software requirements** :

**1. Programming Language**

* **Python 3.8+** (Recommended: Python 3.10 or later)
  + Ensure you have Python installed.

**2. Required Python Libraries**

Although the core system does not require external libraries, the following can enhance functionality:

| **Library** | **Purpose** | **Installation** |
| --- | --- | --- |
| datetime | Built-in Python module for managing dates and times | Pre-installed |
| abc | Provides abstract base classes for enforcing OOP principles | Pre-installed |
| pytest | For unit testing the system | pip install pytest |
| sqlite3 | Lightweight database for storing product, customer, and order data | Pre-installed |
| flask (optional) | If building a web API for the system | pip install flask |
| tkinter (optional) | GUI-based shopping cart interface | Pre-installed in most Python versions |

**3. Development Environment**

* **IDEs / Code Editors**:
  + **PyCharm** (Recommended for structured Python projects)
  + **VS Code** (Lightweight and supports Python extensions)
  + **Jupyter Notebook** (For prototyping)
  + **IDLE** (Basic Python editor)
* **Version Control**:
  + **Git** for managing and tracking changes.

**System Requirements:**

**OS**: Windows, macOS, Linux

**RAM**: Minimum **4GB** (8GB+ recommended for large datasets)

**Processor**: Any modern **x86/x64** or ARM-based processor

**Key Object-Oriented Programming (OOP) Components in an E-Commerce System**

An **E-Commerce System** follows **OOP principles** to make the system modular, scalable, and reusable. Below are the key OOP concepts used in the design:

**1️.Encapsulation (Data Hiding)**

* Encapsulation restricts direct access to object data and allows controlled modifications via methods.
* **Example:**
  + The Customer class has private attributes like \_email, \_password that are accessed via getter/setter methods.

**2️.Inheritance (Code Reusability)**

* Inheritance allows one class to inherit properties and behaviors from another class.
* **Example:**
  + ElectronicsProduct and ClothingProduct **inherit** from Product.

**3️.Polymorphism (Method Overriding & Overloading) Method Overriding:** A child class provides its own implementation of a method.

* **Example:** The Payment class has a process\_payment() method that is overridden by subclasses (CreditCardPayment, PayPalPayment).
* # Example Usage

payment1 = CreditCardPayment()

payment1.process\_payment() # Output: Processing credit card payment.

payment2 = PayPalPayment()

payment2.process\_payment() # Output: Processing PayPal payment

**4️. Abstraction (Hiding Implementation Details)**

* Abstraction hides the complex details and only shows the necessary parts.
* **Example:**

The Order class interacts with a payment method but doesn’t need to know the details of how the payment is processed.

# Example Usage

order\_payment = StripePayment()

order\_payment.process\_payment(100) # Output: Processing $100 via Stripe

**5️.Composition (Has-a Relationship)**

* Instead of using inheritance, **composition** allows objects to contain other objects.
* **Example:** An Order **has a** ShoppingCart object.