**Building a Simple E-Commerce System with Python**

Sevde Zümrüt İspay

***Abstract***

E-commerce systems are widely used today. In this article, we will explain the basic steps to build a simple e-commerce system using Python. Our project consists of a modular structure that includes product management, customer operations, cart management, and order processing.

**Code Explanation**

**1. Product Module**

This module defines a class that contains the name and price of products. The \_\_str\_\_ method ensures that products can be printed in a readable format.

class Product:

def \_\_init\_\_(self, name, price):

self.name = name

self.price = price

def \_\_str\_\_(self):

return f"{self.name} - {self.price} TL"

**2. Customer Module**

A class is defined to store customer information. The \_\_str\_\_ method makes it easy to display customer details.

class Customer:

def \_\_init\_\_(self, name):

self.name = name

def \_\_str\_\_(self):

return f"Customer: {self.name}"

**3. Cart Module**

This module defines a class that allows customers to add and remove products from the cart. The add\_product, remove\_product, and show\_cart methods manage the cart operations.

class Cart:

def \_\_init\_\_(self):

self.items = []

def add\_product(self, product):

self.items.append(product)

print(f"{product.name} added to cart.")

def remove\_product(self, product\_name):

for product in self.items:

if product.name == product\_name:

self.items.remove(product)

print(f"{product.name} removed from cart.")

return

print("Product not found in cart.")

def show\_cart(self):

if not self.items:

print("Your cart is empty.")

else:

print("Your cart contains:")

for product in self.items:

print(product)

**4. Order Module**

This module manages order creation and calculates the total price.

class Order:

def \_\_init\_\_(self, customer, cart):

self.customer = customer

self.cart = cart

self.total = cart.total\_price()

def checkout(self):

if not self.cart.items:

print("Your cart is empty, order cannot be placed.")

else:

print(f"Order placed for {self.customer}. Total Amount: {self.total} TL")

**5. Main Program Module**

This part provides user interaction. The user can list products, add them to the cart, and place an order.

from product import Product

from customer import Customer

from cart import Cart

from order import Order

def main():

customer\_name = input("Enter your name: ")

customer = Customer(customer\_name)

cart = Cart()

products = [

Product("Laptop", 15000),

Product("Phone", 10000),

Product("Headphones", 500),

Product("Keyboard", 800)

]

while True:

print("\n1. List Products")

print("2. Add Product to Cart")

print("3. Remove Product from Cart")

print("4. View Cart")

print("5. Place Order")

print("6. Exit")

choice = input("Make a selection: ")

if choice == "1":

print("Available Products:")

for i, product in enumerate(products, 1):

print(f"{i}. {product}")

elif choice == "2":

product\_index = int(input("Enter the product number to add: ")) - 1

if 0 <= product\_index < len(products):

cart.add\_product(products[product\_index])

else:

print("Invalid product number!")

elif choice == "3":

product\_name = input("Enter the product name to remove: ")

cart.remove\_product(product\_name)

elif choice == "4":

cart.show\_cart()

elif choice == "5":

order = Order(customer, cart)

order.checkout()

elif choice == "6":

print("Exiting...")

break

else:

print("Invalid choice, try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

***Conclusion***

This project provides a great starting point for understanding the fundamental components of an e-commerce system. Its modular structure allows for easy expansion. In the future, features like database integration, user authentication, and payment systems can be added to make the system more functional.