**ONLINE SHOPPING MANAGEMENT SYSTEM**

**Project Description**

**1. Aim of the Project**

The aim of this project is to develop a simple, text-based clothing shop application in Python. The project focuses on simulating an online shopping experience, including browsing items, adding items to a cart, viewing the cart, and completing a purchase. This project does not use any external packages and emphasizes user interaction, input handling, and basic object-oriented programming principles.

**2. Project Description**

The project is a console-based application where users can log in, browse a list of clothing items, add items to their shopping cart, view their cart, remove items from the cart, and proceed to checkout. The application handles user authentication, manages item stock, and processes payment details. The project is structured using classes to represent items, users, and the shopping cart, demonstrating the use of classes and methods in Python.

**3. Functionalities**

The main functionalities of the project include:

* **User Authentication:** Users can log in using predefined credentials.
* **Browsing Items:** Users can browse available clothing items, which are displayed with details such as name, category, size range, price, and stock.
* **Adding to Cart:** Users can select items and sizes to add to their shopping cart.
* **Viewing Cart:** Users can view the contents of their cart at any time.
* **Removing from Cart:** Users can remove items from their cart if they change their mind.
* **Checkout Process:** Users can proceed to checkout, provide address details, choose a payment method, and confirm their order.

**4. Input Versatility with Error Handling and Exception Handling**

The application includes various input prompts that guide the user through different operations. It incorporates error handling to manage invalid inputs and ensure smooth execution. For example:

* **Size Selection:** The application prompts users to select a size and validates the input against a predefined list of sizes.
* **Payment Method:** Users can select between 'cash' and 'google pay' as payment methods. For 'google pay', the application further validates the mobile number to ensure it is a 10-digit numeric value.
* **Login:** The login process checks for valid usernames and passwords, providing feedback on incorrect credentials.

These measures enhance user experience by preventing invalid inputs and providing clear guidance for correct entries.

**5. Code Implementation**

The project uses classes to represent different components:

* **Item Class:** Represents a clothing item with attributes like name, price, category, size, and stock.
* **Shop Class:** Manages the list of available items and provides methods to display items and retrieve a specific item by index.
* **Cart Class:** Manages the shopping cart, including adding items, removing items, displaying the cart, and calculating the total amount.
* **User Class:** Represents a user with a username and password.

Key functions include:

* **get\_cloth\_size :** Prompts the user to select a size and validates the input.
* **get\_address\_details:** Prompts the user to enter address details.
* **get\_payment\_method:** Prompts the user to select a payment method and validates the input, especially for Google Pay.
* **confirm\_order:** Confirms the user's intention to place an order.

The main function orchestrates the overall flow of the application, handling user login, item browsing, cart management, and checkout.

**Description:-**

In this project, implement various modules using basic Python programming concepts.each module is designed to handle specific functionalities of the online shopping management system.For example, let's consider the implementation of a online shopping management module:

class Item:

def \_\_init\_\_(self, name, price, category, size, stock):

self.name = name

self.price = price

self.category = category

self.size = size

self.stock = stock

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

return f"{self.name} - {self.category} - Size: {self.size} - ${self.price:.2f} (Stock: {self.stock})"

class Shop:

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

self.items = [

Item("T-Shirt", 200.0, "Casual", "S to XXL", 10),

Item("Jeans", 500.0, "Denim", "S to XXL", 5),

Item("Jacket", 100.0, "Outerwear", "S to XXL", 3),

Item("Dress", 700.0, "Formal", "S to XXL", 7),

Item("Shirt", 700.0, "Formal", "S to XXL", 7),

Item("Chudi", 600.0, "Traditional", "S to XXL", 19),

Item("Chudi", 500.0, "Western", "S to XXL", 7),

Item("Maxi", 700.0, "Western", "S to XXL", 7),

Item("Aliya cut kurti", 750.0, "Casual", "S to XXL", 5),

Item("Kurti", 300.0, "Formal", "S to XXL", 7),

Item("Leggings", 100.0, "Casual", "S to XXL", 20)

]

def display\_items(self):

print("\nAvailable Items:")

for i, item in enumerate(self.items, start=1):

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

def get\_item(self, index):

if 0 <= index < len(self.items):

return self.items[index]

else:

return None

class Cart:

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

self.cart\_items = []

def add\_to\_cart(self, item, size):

if item.stock > 0:

self.cart\_items.append((item, size))

item.stock -= 1

print(f"{item.name} (Size: {size}) added to cart.")

else:

print(f"{item.name} is out of stock.")

def remove\_from\_cart(self, item\_name):

for item, size in self.cart\_items:

if item.name == item\_name:

self.cart\_items.remove((item, size))

item.stock += 1

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

return

print(f"{item\_name} is not in your cart.")

def display\_cart(self):

if not self.cart\_items:

print("Your cart is empty.")

else:

print("\nYour Cart:")

for i, (item, size) in enumerate(self.cart\_items, start=1):

print(f"{i}. {item} (Size: {size})")

def checkout(self):

total = sum(item.price for item, size in self.cart\_items)

print(f"\nTotal amount: ${total:.2f}")

return total

class User:

def \_\_init\_\_(self, username, password):

self.username = username

self.password = password

def get\_cloth\_size():

sizes = ['S', 'M', 'L', 'XL', 'XXL']

while True:

size = input("Please select your size (S, M, L, XL, XXL): ").upper()

if size in sizes:

return size

else:

print("Invalid size. Please select a valid size.")

def get\_address\_details():

address = input("Please enter your address details: ")

return address

def get\_payment\_method():

while True:

payment\_method = input("Please select your payment method (cash/google pay): ").lower()

if payment\_method == 'cash':

return payment\_method, None

elif payment\_method == 'google pay':

while True:

mobile\_number = input("Please enter your mobile number for Google Pay: ")

if mobile\_number.isdigit() and len(mobile\_number) == 10:

return payment\_method, mobile\_number

else:

print("Invalid mobile number. Please enter a 10-digit numeric value.")

else:

print("Invalid payment method. Please select 'cash' or 'google pay'.")

def confirm\_order():

while True:

confirmation = input("Do you want to confirm your order? (yes/no): ").lower()

if confirmation in ['yes', 'no']:

return confirmation

else:

print("Invalid input. Please type 'yes' or 'no'.")

def login(users):

username = input("Enter username: ")

password = input("Enter password: ")

for user in users:

if user.username == username and user.password == password:

print(f"Welcome, {username}!")

return user

print("Invalid username or password.")

return None

def main():

users = [User("user1", "pass1"), User("user2", "pass2")] # Sample users

logged\_in\_user = None

shop = Shop()

cart = Cart()

while True:

if logged\_in\_user:

print("\nWelcome to the Clothing Shop!")

print("1. Browse items")

print("2. View cart")

print("3. Checkout")

print("4. Exit")

choice = input("Enter your choice: ")

if choice == '1':

while True:

shop.display\_items()

try:

item\_index = int(input("Enter the number of the item to add to cart (0 to cancel): ")) - 1

if item\_index == -1:

break

item = shop.get\_item(item\_index)

if item:

size = get\_cloth\_size()

cart.add\_to\_cart(item, size)

repeat = input("Do you want to add any other dresses? (yes/no): ")

if repeat.lower() != 'yes':

break

else:

print("Invalid item number. Please try again.")

except ValueError:

print("Invalid input. Please enter a number.")

elif choice == '2':

cart.display\_cart()

remove\_item = input("Enter the name of the item to remove from cart (or '0' to cancel): ")

if remove\_item != '0':

cart.remove\_from\_cart(remove\_item)

elif choice == '3':

if not cart.cart\_items:

print("Your cart is empty. Please add items before checking out.")

continue

total = cart.checkout()

address = get\_address\_details()

payment\_method, mobile\_number = get\_payment\_method()

confirmation = confirm\_order()

if confirmation == 'yes':

print("Thank you for your shopping.")

print()

print("Order Summary:")

cart.display\_cart()

print(f"Total Amount: ${total:.2f}")

print(f"Address: {address}")

print(f"Payment Method: {payment\_method}")

if mobile\_number:

print(f"Google Pay Mobile Number: {mobile\_number}")

break

else:

print("Order not confirmed.")

elif choice == '4':

print(f"Goodbye, {logged\_in\_user.username}!")

logged\_in\_user = None

else:

print("Invalid choice. Please try again.")

else:

print("\n1. Login")

print("2. Logout")

choice = input("Enter your choice: ")

if choice == '1':

logged\_in\_user = login(users)

elif choice == '2':

print("Thank you for visiting the Clothing Shop!")

break

else:

print("Invalid choice. Please try again.")

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

main()

**6. Results and Outcomes:**

The project successfully simulates a basic online shopping experience with a focus on clothing items. Users can log in, browse items, manage their cart, and complete a purchase. The application handles input validation and provides feedback on user actions, ensuring a smooth and interactive experience.

**7. Conclusion:**

This project demonstrates the use of object-oriented programming in Python to create a simple, console-based shopping application. It highlights the importance of input validation and error handling to ensure robust user interactions. While basic, the project serves as a foundation for understanding more complex e-commerce systems and can be expanded with additional features such as user registration, order history, and inventory management. Overall, it provides a practical example of how to structure and implement a small-scale application using Python.