**Source Code**

import uuid

# --- Demo Data (In-memory "databases") ---

# Users: username -> password

users\_db = {

"user1": "password1",

"user2": "password2"

}

# Admins: username -> password

admins\_db = {

"admin": "adminpass"

}

# Categories: category\_id -> category\_name

categories\_db = {

1: "Boots",

2: "Coats",

3: "Jackets",

4: "Caps"

}

# Products: product\_id -> dict with product info

products\_db = {

1: {"name": "Leather Boots", "category\_id": 1, "price": 2500},

2: {"name": "Winter Coat", "category\_id": 2, "price": 3500},

3: {"name": "Denim Jacket", "category\_id": 3, "price": 2000},

4: {"name": "Baseball Cap", "category\_id": 4, "price": 500},

}

# Sessions: session\_id -> dict with user info and cart (for users) or admin info

sessions = {}

# Helper function to generate session ids

def generate\_session\_id():

return str(uuid.uuid4())

# --- Utility Functions ---

def print\_welcome():

print("Welcome to the Demo Marketplace\n")

def verify\_user(username, password):

return users\_db.get(username) == password

def verify\_admin(username, password):

return admins\_db.get(username) == password

def display\_categories():

print("\nProduct Categories:")

for cid, cname in categories\_db.items():

print(f" {cid}. {cname}")

def display\_products():

print("\nProduct Catalog:")

print(f"{'ID':<5} {'Name':<20} {'Category':<10} {'Price (Rs.)':<10}")

for pid, pinfo in products\_db.items():

cname = categories\_db.get(pinfo["category\_id"], "Unknown")

print(f"{pid:<5} {pinfo['name']:<20} {cname:<10} {pinfo['price']:<10}")

def get\_category\_by\_id(cid):

return categories\_db.get(cid)

def get\_product\_by\_id(pid):

return products\_db.get(pid)

# --- User Functions ---

def user\_menu(session\_id):

while True:

print("\nUser Menu:")

print("1. View Catalog")

print("2. View Cart")

print("3. Add Item to Cart")

print("4. Remove Item from Cart")

print("5. Checkout")

print("6. Logout")

choice = input("Choose an option: ").strip()

if choice == "1":

display\_products()

elif choice == "2":

view\_cart(session\_id)

elif choice == "3":

add\_item\_to\_cart(session\_id)

elif choice == "4":

remove\_item\_from\_cart(session\_id)

elif choice == "5":

checkout(session\_id)

elif choice == "6":

print("Logging out...")

sessions.pop(session\_id, None)

break

else:

print("Invalid choice, please try again.")

def view\_cart(session\_id):

cart = sessions[session\_id]["cart"]

if not cart:

print("Your cart is empty.")

return

print("\nYour Cart:")

total = 0

print(f"{'Product ID':<12} {'Name':<20} {'Quantity':<10} {'Price per unit':<15} {'Subtotal':<10}")

for pid, qty in cart.items():

product = get\_product\_by\_id(pid)

if product:

subtotal = product["price"] \* qty

total += subtotal

print(f"{pid:<12} {product['name']:<20} {qty:<10} {product['price']:<15} {subtotal:<10}")

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

def add\_item\_to\_cart(session\_id):

try:

pid = int(input("Enter Product ID to add: "))

product = get\_product\_by\_id(pid)

if not product:

print("Product not found.")

return

qty = int(input("Enter quantity: "))

if qty <= 0:

print("Quantity must be positive.")

return

cart = sessions[session\_id]["cart"]

cart[pid] = cart.get(pid, 0) + qty

print(f"Added {qty} x {product['name']} to cart.")

except ValueError:

print("Invalid input. Please enter numeric values.")

def remove\_item\_from\_cart(session\_id):

try:

pid = int(input("Enter Product ID to remove: "))

cart = sessions[session\_id]["cart"]

if pid not in cart:

print("Product not in cart.")

return

qty = int(input("Enter quantity to remove: "))

if qty <= 0:

print("Quantity must be positive.")

return

if qty >= cart[pid]:

del cart[pid]

print("Product removed from cart.")

else:

cart[pid] -= qty

print(f"Removed {qty} units of product from cart.")

except ValueError:

print("Invalid input. Please enter numeric values.")

def checkout(session\_id):

cart = sessions[session\_id]["cart"]

if not cart:

print("Your cart is empty. Cannot checkout.")

return

total = sum(products\_db[pid]["price"] \* qty for pid, qty in cart.items())

print(f"Total payable amount: Rs. {total}")

print("Payment Options:")

print("1. UPI")

print("2. Debit Card")

print("3. Net Banking")

print("4. PayPal")

choice = input("Select payment method (1-4): ").strip()

payment\_methods = {

"1": "Unified Payment Interface (UPI)",

"2": "Debit Card",

"3": "Net Banking",

"4": "PayPal"

}

if choice not in payment\_methods:

print("Invalid payment option.")

return

method = payment\_methods[choice]

if method == "UPI":

print(f"You will be shortly redirected to the portal for Unified Payment Interface to make a payment of Rs. {total}")

else:

print(f"Your order is successfully placed using {method}. Amount paid: Rs. {total}")

# Clear cart after payment

sessions[session\_id]["cart"].clear()

# --- Admin Functions ---

def admin\_menu(session\_id):

while True:

print("\nAdmin Menu:")

print("1. View Catalog")

print("2. Add New Product")

print("3. Update Existing Product")

print("4. Remove Product")

print("5. Add New Category")

print("6. Remove Category")

print("7. Logout")

choice = input("Choose an option: ").strip()

if choice == "1":

display\_products()

elif choice == "2":

add\_new\_product(session\_id)

elif choice == "3":

update\_existing\_product(session\_id)

elif choice == "4":

remove\_product(session\_id)

elif choice == "5":

add\_new\_category(session\_id)

elif choice == "6":

remove\_category(session\_id)

elif choice == "7":

print("Logging out...")

sessions.pop(session\_id, None)

break

else:

print("Invalid choice, please try again.")

def add\_new\_product(session\_id):

try:

pname = input("Enter product name: ").strip()

display\_categories()

cid = int(input("Enter category ID for product: "))

if cid not in categories\_db:

print("Category does not exist.")

return

price = float(input("Enter price: "))

if price <= 0:

print("Price must be positive.")

return

new\_pid = max(products\_db.keys()) + 1 if products\_db else 1

products\_db[new\_pid] = {"name": pname, "category\_id": cid, "price": price}

print(f"Product '{pname}' added successfully with ID {new\_pid}.")

except ValueError:

print("Invalid input. Please enter numeric values for category and price.")

def update\_existing\_product(session\_id):

try:

pid = int(input("Enter product ID to update: "))

product = get\_product\_by\_id(pid)

if not product:

print("Product not found.")

return

print(f"Current details: Name: {product['name']}, Category ID: {product['category\_id']}, Price: {product['price']}")

pname = input("Enter new product name (leave blank to keep current): ").strip()

display\_categories()

cid\_input = input("Enter new category ID (leave blank to keep current): ").strip()

price\_input = input("Enter new price (leave blank to keep current): ").strip()

if pname:

product['name'] = pname

if cid\_input:

cid = int(cid\_input)

if cid in categories\_db:

product['category\_id'] = cid

else:

print("Category does not exist. Keeping previous category.")

if price\_input:

price = float(price\_input)

if price > 0:

product['price'] = price

else:

print("Invalid price. Keeping previous price.")

print("Product updated successfully.")

except ValueError:

print("Invalid input. Please enter numeric values where required.")

def remove\_product(session\_id):

try:

pid = int(input("Enter product ID to remove: "))

if pid not in products\_db:

print("Product not found.")

return

removed\_product = products\_db.pop(pid)

print(f"Product '{removed\_product['name']}' removed from catalog.")

except ValueError:

print("Invalid input. Please enter a valid product ID.")

def add\_new\_category(session\_id):

cname = input("Enter new category name: ").strip()

if not cname:

print("Category name cannot be empty.")

return

if cname in categories\_db.values():

print("Category already exists.")

return

new\_cid = max(categories\_db.keys()) + 1 if categories\_db else 1

categories\_db[new\_cid] = cname

print(f"Category '{cname}' added successfully with ID {new\_cid}.")

def remove\_category(session\_id):

try:

display\_categories()

cid = int(input("Enter category ID to remove: "))

if cid not in categories\_db:

print("Category not found.")

return

# Check if any product belongs to this category

in\_use = any(p["category\_id"] == cid for p in products\_db.values())

if in\_use:

print("Cannot remove category because some products belong to it. Remove or reassign those products first.")

return

removed\_cat = categories\_db.pop(cid)

print(f"Category '{removed\_cat}' removed successfully.")

except ValueError:

print("Invalid input. Please enter a valid category ID.")

# --- Main Login and Routing Logic ---

def main():

print\_welcome()

while True:

print("\nLogin Menu:")

print("1. User Login")

print("2. Admin Login")

print("3. Exit")

choice = input("Select option: ").strip()

if choice == "1":

username = input("Enter username: ").strip()

password = input("Enter password: ").strip()

if verify\_user(username, password):

session\_id = generate\_session\_id()

sessions[session\_id] = {"role": "user", "username": username, "cart": {}}

print(f"User '{username}' logged in successfully. Session ID: {session\_id}")

user\_menu(session\_id)

else:

print("Invalid user credentials.")

elif choice == "2":

username = input("Enter admin username: ").strip()

password = input("Enter admin password: ").strip()

if verify\_admin(username, password):

session\_id = generate\_session\_id()

sessions[session\_id] = {"role": "admin", "username": username}

print(f"Admin '{username}' logged in successfully. Session ID: {session\_id}")

admin\_menu(session\_id)

else:

print("Invalid admin credentials. Access denied.")

elif choice == "3":

print("Exiting Demo Marketplace. Goodbye!")

break

else:

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

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

main()