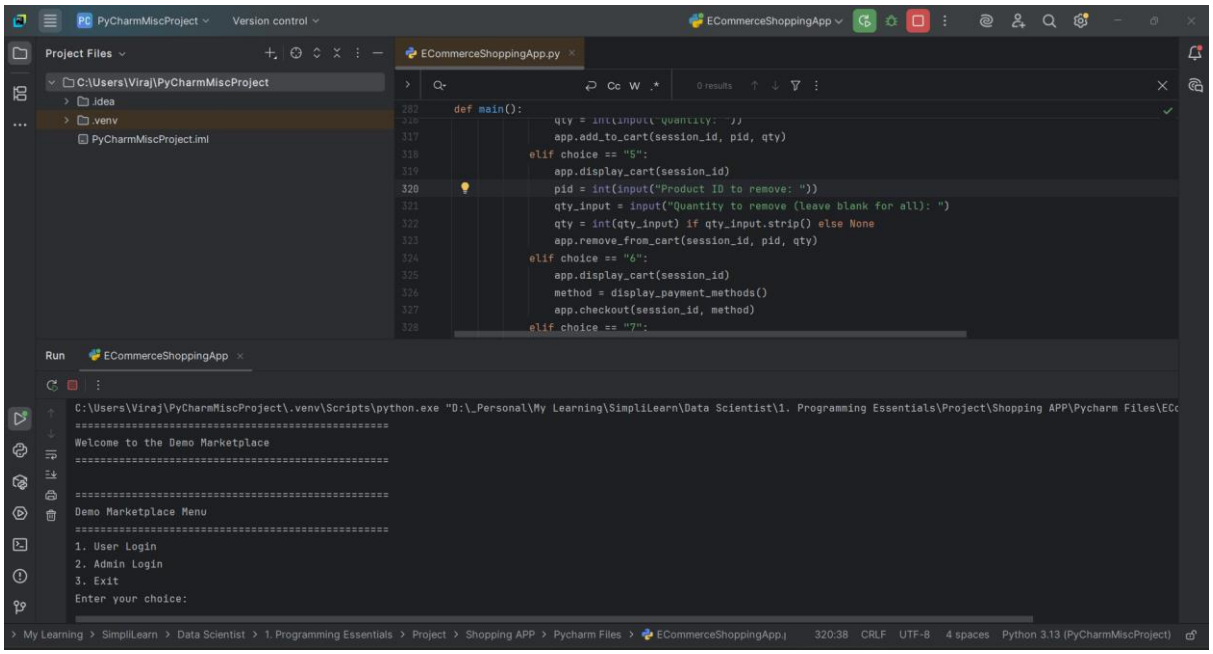
Online Shopping App – E-Commerce Project

1. Introduction

The **Online Shopping App – E-Commerce Project** is a Python-based console application that simulates the core functionality of an online shopping system. This project demonstrates backend logic such as user and admin authentication, product catalog management, shopping cart operations, and payment simulation, all implemented using object-oriented programming concepts.

The primary goal of this project is to develop a structured and modular Python program that showcases how a real-world e-commerce platform operates internally, focusing on backend logic without involving databases or user interface design.



---

2. Objective of the Project

The objective of this project is to design and implement a simple, interactive, and role-based shopping application that allows:

- **Users** to log in, browse products, manage their shopping carts, and simulate payments.
- **Admins** to log in securely, manage products and categories, and control the catalog.

This project emphasizes the concepts of class design, data management using Python dictionaries, and session-based functionality—all crucial for developing real-world software systems.

---

3. Tools and Technologies Used

- **Programming Language:** Python 3.x
- **IDE Used:** PyCharm (recommended)
- **Libraries:** Standard Python library
- **Paradigm:** Object-Oriented Programming (OOP)
- **Environment:** Console-based application

---

4. System Design Overview

The project follows a **modular and role-based design**. It uses a single class to encapsulate all system features while maintaining a clear distinction between **user operations** and **admin operations**.

Core Design Concepts:

- **Encapsulation:** All functionality related to shopping and management is encapsulated within a single class ECommerceBackend.
- **Session Management:** Each login creates a unique session ID to track the current user or admin.
- **Role Separation:** Access control ensures that only users can manage carts and only admins can modify the product catalog.
- **Dictionary-Based Databases:** Products, categories, users, and carts are stored in Python dictionaries, simulating a lightweight in-memory database.

---

5. Class and Function Descriptions

ECommerceBackend Class

This class forms the backbone of the project. It maintains user and admin records, manages the product catalog, and implements all core functionalities.

Constructor (__init__)

Initializes demo databases for users, admins, categories, and products. It also sets up session and cart management structures.

Key Attributes:

- users and admins: Store demo credentials.
- categories: Contains category IDs and names.
- products: Holds product data including ID, name, category, and price.
- user_carts: Stores user-specific cart items.
- active_sessions: Tracks logged-in users and their session types.

Main Methods:

1. **display_welcome_message()** – Prints the welcome banner when the program starts.
2. **login()** – Authenticates users or admins and creates session IDs.
3. **logout()** – Ends the current session and clears data if required.
4. **is_valid_session()** – Checks if the session is valid and ensures the correct access level.
5. **display_catalog()** – Displays all available products in a structured format.
6. **display_categories()** – Lists all available product categories.
7. **display_cart()** – Shows the contents of a user's cart with individual and total prices.
8. **add_to_cart()** – Adds a product to the user’s shopping cart.
9. **remove_from_cart()** – Removes an item or reduces its quantity in the cart.
10. **checkout()** – Simulates the payment and order placement process.
11. **add_product()** – Allows admin to add new products to the catalog.
12. **update_product()** – Enables admin to update existing product details.
13. **delete_product()** – Deletes products from the catalog and clears them from user carts.
14. **add_category()** – Adds new product categories.
15. **delete_category()** – Deletes existing categories (only if no products belong to them).

```python
    # ------------------- Static Welcome Message -------------------
    @staticmethod
    def display_welcome_message():
        print("=" * 50)
        print("Welcome to the Demo Marketplace")
        print("=" * 50)

    # ------------------- Login / Logout -------------------
    def login(self, username, password, user_type="user"):
        database = self.users if user_type == "user" else self.admins
        if username in database and database[username]["password"] == password:
            session_id = str(self.next_session_id)
            self.next_session_id += 1
            self.active_sessions[session_id] = {"username": username, "type": user_type}
            if user_type == "user":
                self.user_carts[session_id] = {}
            print(f"Login successful. Welcome {database[username]['name']}!")
            return session_id
        else:
            print("Invalid credentials. Please try again.")
            return None
```

```python
    def is_valid_session(self, session_id, required_type=None):
        if session_id not in self.active_sessions:
            print("Invalid session. Please login again.")
            return False
        if required_type and self.active_sessions[session_id]["type"] != required_type:
            print(f"Access denied. Requires {required_type} privileges.")
            return False
        return True
```

```python
    # ------------------- Display Catalog / Categories -------------------
    def display_catalog(self, session_id):
        if not self.is_valid_session(session_id):
            return False
        print("\nProduct Catalog")
        print("=" * 50)
        print("{:<10} {:<20} {:<15} {:<10}".format( *args: "ID", "Name", "Category", "Price"))
        print("-" * 55)
        for pid, product in self.products.items():
            category = self.categories.get(product["category_id"], "Unknown")
            print("{:<10} {:<20} {:<15} ₹{:<10}".format( *args: pid, product["name"], category, product["price"]))
        return True

    def display_categories(self, session_id):
        if not self.is_valid_session(session_id):
            return False
        print("\nProduct Categories")
        print("=" * 50)
        print("{:<10} {:<20}".format( *args: "ID", "Category Name"))
        print("-" * 30)
        for cid, cname in self.categories.items():
            print("{:<10} {:<20}".format( *args: cid, cname))
        return True
```

```python
    # ------------------- User Cart Operations -------------------
    def display_cart(self, session_id):
        if not self.is_valid_session(session_id, required_type: "user"):
            return False
        cart = self.user_carts.get(session_id, {})
        if not cart:
            print("\nYour cart is empty.")
            return True
        total_amount = 0
        print("\nYour Shopping Cart")
        print("=" * 50)
        print("{:<10} {:<20} {:<10} {:<10} {:<10}".format( *args: "ID", "Product", "Price", "Quantity", "Total"))
        print("-" * 60)
        for pid, qty in cart.items():
            if pid in self.products:
                price = self.products[pid]["price"]
                total = price * qty
                total_amount += total
                print("{:<10} {:<20} ₹{:<10} {:<10} ₹{:<10}".format( *args: pid, self.products[pid]["name"], price, q
        print("-" * 60)
        print(f"Total Amount: ₹{total_amount}")
        return True
```

```python
    def add_to_cart(self, session_id, product_id, quantity=1):
        if not self.is_valid_session(session_id, required_type: "user"):
            return False
        product_id = int(product_id)
        quantity = int(quantity)
        if product_id not in self.products:
            print("Product not found.")
            return False
        if quantity <= 0:
            print("Quantity must be > 0.")
            return False
        cart = self.user_carts.get(session_id, {})
        cart[product_id] = cart.get(product_id, 0) + quantity
        self.user_carts[session_id] = cart
        print(f"Added {quantity} {self.products[product_id]['name']}(s) to cart.")
        return True
```

```python
    def remove_from_cart(self, session_id, product_id, quantity=None):
        if not self.is_valid_session(session_id, required_type: "user"):
            return False
        product_id = int(product_id)
        cart = self.user_carts.get(session_id, {})
        if product_id not in cart:
            print("Product not in cart.")
            return False
        if quantity is None or quantity >= cart[product_id]:
            del cart[product_id]
            print(f"Removed product {product_id} from cart.")
        else:
            quantity = int(quantity)
            cart[product_id] -= quantity
            print(f"Reduced quantity of product {product_id} by {quantity}.")
        self.user_carts[session_id] = cart
        return True
```

```python
    # ------------------- Checkout -------------------
    def checkout(self, session_id, payment_method):
        if not self.is_valid_session(session_id, required_type: "user"):
            return False
        cart = self.user_carts.get(session_id, {})
        if not cart:
            print("Cart empty. Nothing to checkout.")
            return False
        total = sum(self.products[pid]["price"] * qty for pid, qty in cart.items() if pid in self.products)
        methods = {"1": "Net Banking", "2": "PayPal", "3": "UPI", "4": "Debit Card", "5": "Credit Card"}
        method_name = methods.get(payment_method, payment_method)
        print("\nProcessing Payment...")
        if method_name.lower() == "upi":
            print(f"Redirecting to UPI for payment of ₹{total}")
        else:
            print(f"Redirecting to {method_name} for payment of ₹{total}")
        print("Order successfully placed!")
        self.user_carts[session_id] = {}  # clear cart
        return True
```

```python

    # ------------------- Admin Operations -------------------
    def add_product(self, session_id, name, category_id, price):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        category_id = int(category_id)
        price = float(price)
        if category_id not in self.categories:
            print("Invalid category.")
            return False
        if price <= 0:
            print("Price must be > 0.")
            return False
        pid = self.next_product_id
        self.next_product_id += 1
        self.products[pid] = {"name": name, "category_id": category_id, "price": price}
        print(f"Product '{name}' added with ID {pid}.")
        return True
```

```python

    def update_product(self, session_id, product_id, name=None, category_id=None, price=None):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        product_id = int(product_id)
        if product_id not in self.products:
            print("Product not found.")
            return False
        if name:
            self.products[product_id]["name"] = name
        if category_id:
            category_id = int(category_id)
            if category_id not in self.categories:
                print("Category invalid.")
                return False
            self.products[product_id]["category_id"] = category_id
        if price:
            price = float(price)
            if price <= 0:
                print("Price must be > 0.")
                return False
            self.products[product_id]["price"] = float(price)  # type: ignore
        print(f"Product {product_id} updated.")
        return True
```

```python
    def delete_product(self, session_id, product_id):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        product_id = int(product_id)
        if product_id not in self.products:
            print("Product not found.")
            return False
        del self.products[product_id]
        for cart in self.user_carts.values():
            cart.pop(product_id, None)
        print(f"Product {product_id} deleted.")
        return True
```

```python
    def add_category(self, session_id, name):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        cid = self.next_category_id
        self.next_category_id += 1
        self.categories[cid] = name
        print(f"Category '{name}' added with ID {cid}.")
        return True

    def delete_category(self, session_id, category_id):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        category_id = int(category_id)
        if category_id not in self.categories:
            print("Category not found.")
            return False
        if any(p["category_id"] == category_id for p in self.products.values()):
            print("Cannot delete category with products.")
            return False
        del self.categories[category_id]
        print(f"Category {category_id} deleted.")
        return True
```

---

6. Feature Implementation Summary

User Features:

- Secure login with demo credentials.
- View product catalog and categories.
- Add, remove, and view items in the cart.
- Proceed to checkout with simulated payment options (UPI, Debit Card, etc.).
- Session-based cart management.

PyCharmMiscProject | Version control | ECommerceShoppingApp

Project Files

C:\Users\Viraj\PyCharmMiscProject
- .idea
- .venv
- PyCharmMiscProject.iml

ECommerceShoppingApp.py

```python
class ECommerceBackend:

    # ------------------ Login / Logout ------------------
    def login(self, username, password, user_type="user"):
        database = self.users if user_type == "user" else self.admins
        if username in database and database[username]["password"] == password:
            session_id = str(self.next_session_id)
            self.next_session_id += 1
            self.active_sessions[session_id] = {"username": username, "type": user_type}
            if user_type == "user":
                self.user_carts[session_id] = {}
            print(f"Login successful. Welcome {database[username]['name']}!")
            return session_id
        else:
            print("Invalid credentials. Please try again.")
            return None
```

Run | ECommerceShoppingApp

```
==================================================
Demo Marketplace Menu
==================================================
1. User Login
2. Admin Login
3. Exit
Enter your choice: 1
Username: user1
Password: pass1
Login successful. Welcome Alice!

==================================================
Demo Marketplace Menu
```

My Learning > SimpliLearn > Data Scientist > 1. Programming Essentials > Project > Shopping APP > Pycharm Files > ECommerceShoppingApp.py 19:30 CRLF UTF-8 4 spaces Python 3.13 (PyCharmMiscProject)

PyCharmMiscProject | Version control | ECommerceShoppingApp

Project Files

C:\Users\Viraj\PyCharmMiscProject
- .idea
- .venv
- PyCharmMiscProject.iml

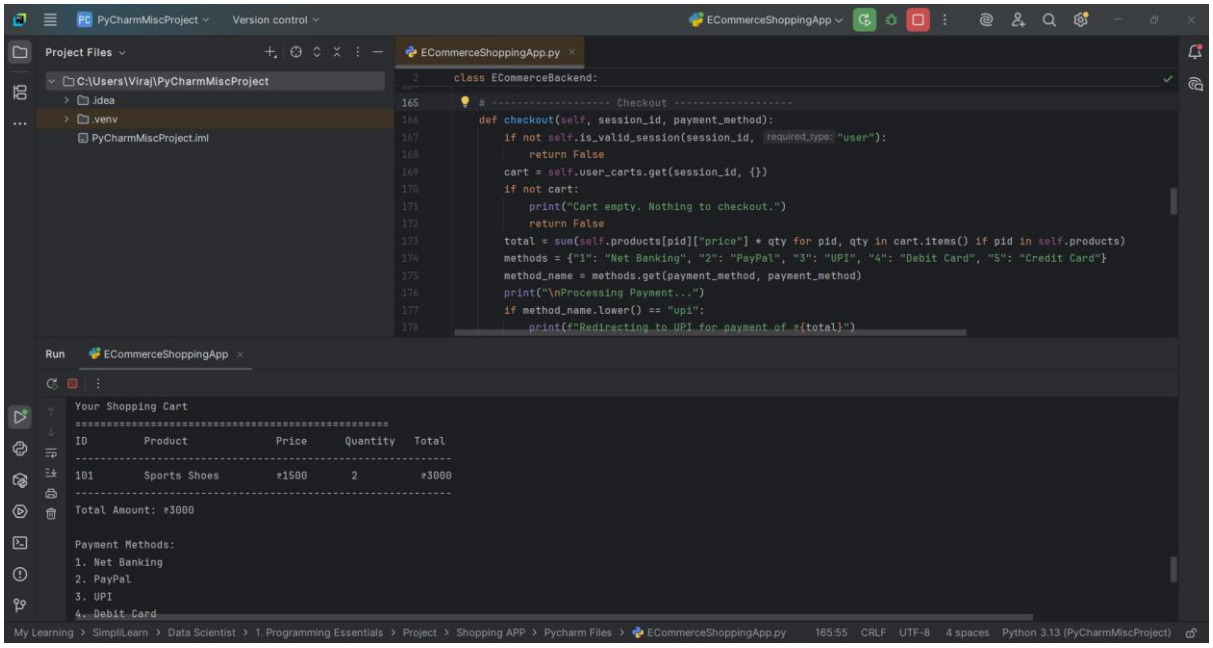
ECommerceShoppingApp.py

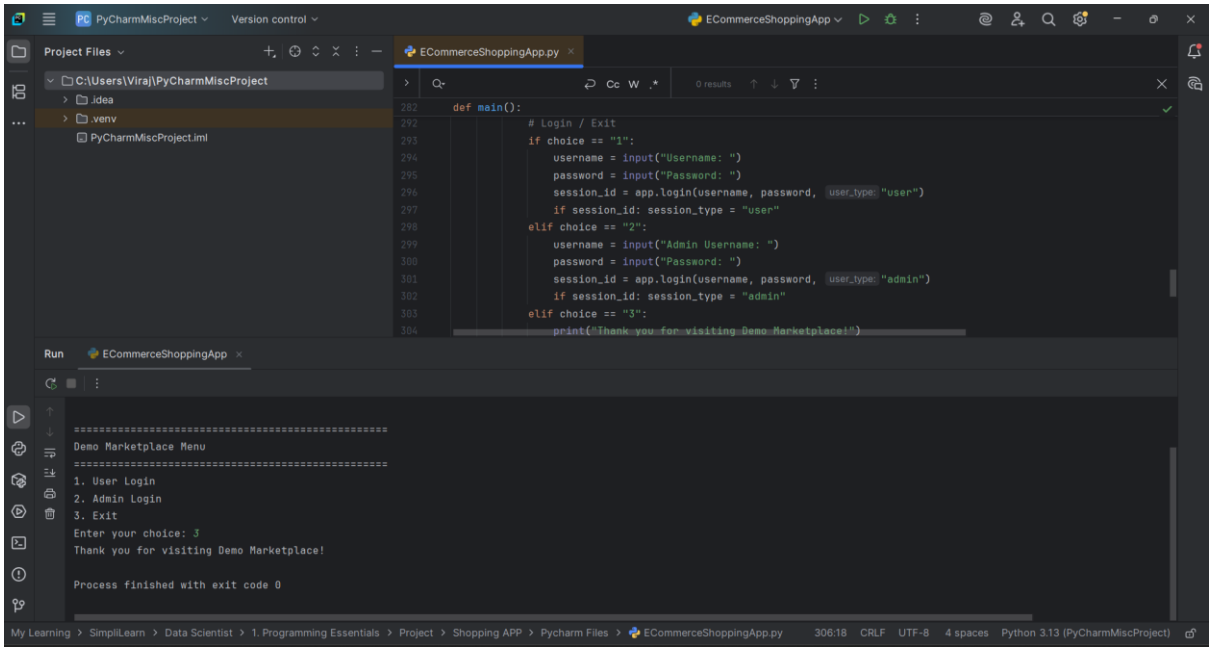
```python
class ECommerceBackend:
    def __init__(self):
        }

        # Products
        self.products = {
            101: {"name": "Sports Shoes", "category_id": 1, "price": 1500},
            102: {"name": "Leather Boots", "category_id": 1, "price": 2000},
            103: {"name": "T-Shirt", "category_id": 2, "price": 500},
            104: {"name": "Jeans", "category_id": 2, "price": 1200},
            105: {"name": "Smartphone", "category_id": 3, "price": 15000},
            106: {"name": "Laptop", "category_id": 3, "price": 45000},
            107: {"name": "Cap", "category_id": 4, "price": 300},
            108: {"name": "Watch", "category_id": 4, "price": 2500}
        }
```

Run | ECommerceShoppingApp

```
==================================================
ID        Name                Category        Price
----------------------------------------------------
101       Sports Shoes        Footwear        ₹1500
102       Leather Boots       Footwear        ₹2000
103       T-Shirt             Clothing        ₹500
104       Jeans               Clothing        ₹1200
105       Smartphone          Electronics     ₹15000
106       Laptop              Electronics     ₹45000
107       Cap                 Accessories     ₹300
108       Watch               Accessories     ₹2500
Product ID to add: 101
Quantity: 2
```

My Learning > SimpliLearn > Data Scientist > 1. Programming Essentials > Project > Shopping APP > Pycharm Files > ECommerceShoppingApp.py 22:1 CRLF UTF-8 4 spaces Python 3.13 (PyCharmMiscProject)





Admin Features:

- Secure admin login with credential validation.
- Add, update, and delete products.
- Add or delete categories dynamically.
- Access control prevents unauthorized actions (e.g., admin cannot use user cart functions).

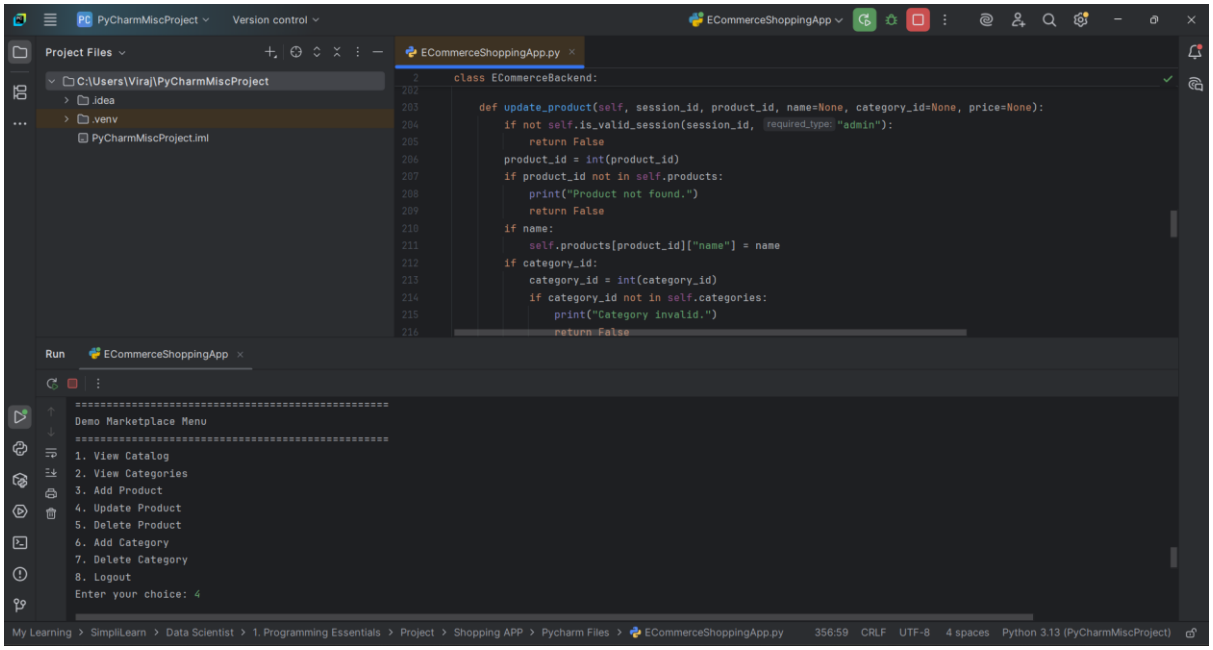
```python
class ECommerceBackend:
    def __init__(self):
        # Demo admin
        self.admins = {
            "admin": {"password": "admin123", "name": "Super Admin"}
        }

        # Categories
        self.categories = {
            1: "Footwear",
            2: "Clothing",
            3: "Electronics",
            4: "Accessories"
        }
```

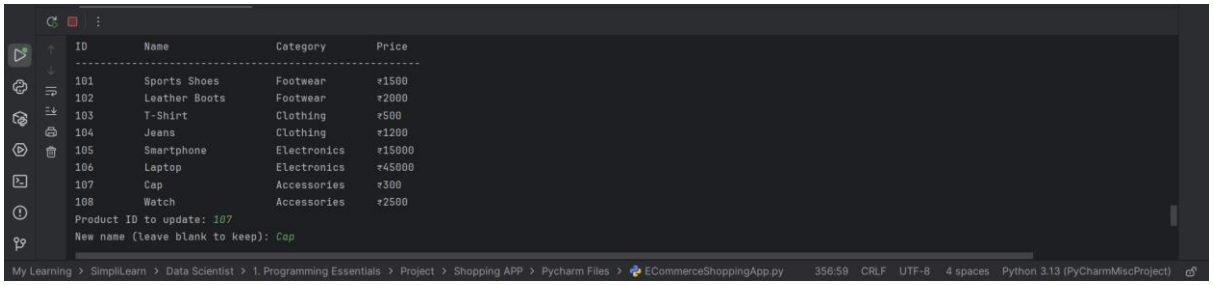
```
==================================================
1. User Login
2. Admin Login
3. Exit
Enter your choice: 2
Admin Username: admin
Password: admin123
Login successful. Welcome Super Admin!

==================================================
Demo Marketplace Menu
==================================================
```

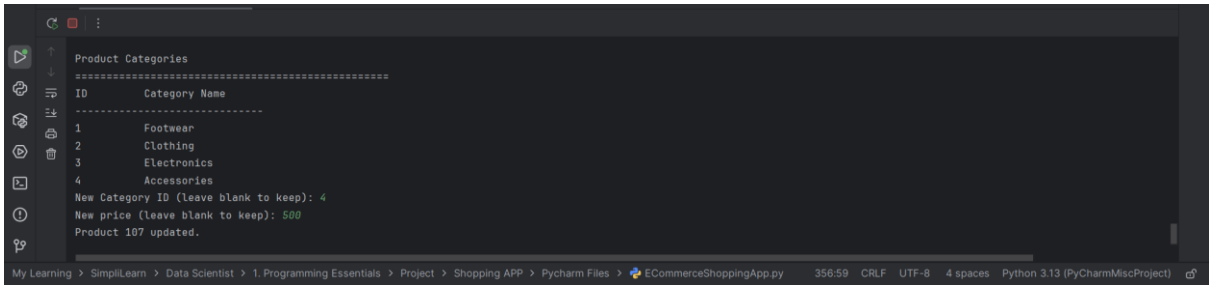
```python
class ECommerceBackend:

    # ------------------- Admin Operations -------------------
    def add_product(self, session_id, name, category_id, price):
        if not self.is_valid_session(session_id, required_type: "admin"):
            return False
        category_id = int(category_id)
        price = float(price)
        if category_id not in self.categories:
            print("Invalid category.")
            return False
        if price <= 0:
            print("Price must be > 0.")
            return False
        pid = self.next_product_id
```

```
==================================================
1. View Catalog
2. View Categories
3. Add Product
4. Update Product
5. Delete Product
6. Add Category
7. Delete Category
8. Logout
Enter your choice: 3
Product name: Watch

Product Categories
```

```
Product Categories
==================================================
ID          Category Name
------------------------------
1           Footwear
2           Clothing
3           Electronics
4           Accessories
Category ID: 4
Price: 700
Product 'Watch' added with ID 109.
```







Session Handling:

- Each successful login generates a new session ID.
- When a user logs out, their session and cart data are cleared.
- The system prevents invalid or unauthorized access attempts.

---

7. Key Learnings and Concepts Applied

During the development of this project, the following programming concepts and skills were reinforced:

- **Object-Oriented Programming** – Use of classes, objects, and methods.
- **Data Structures** – Effective use of dictionaries for in-memory databases.
- **Role-Based Access Control** – Restricting functionalities to users or admins.
- **Session Handling** – Tracking user actions with session IDs.
- **Error Handling and Validation** – Managing incorrect inputs and preventing invalid operations.
- **Code Reusability** – Modular and reusable function definitions.

---

8. Conclusion

The **Online Shopping App – E-Commerce Project** successfully demonstrates how an e-commerce backend operates using Python. It showcases a clean separation between user and admin functionalities, session-based data management, and object-oriented architecture.

Through this project, I learned how to structure Python programs efficiently, handle user interactions dynamically, and simulate real-world system behavior without relying on external databases or GUIs. This project serves as a strong foundation for future enhancements, such as integrating with a web interface or connecting to a database, to create a fully functional online shopping system.
