The ESSEX Project

In this module 2 (Object-Oriented Information Systems) of my master studies at the University of Essex, I did a UML class diagram for a supermarket. Based on my class design I have implemented the fully function Python OOP console-based program to mimic the eCommerce shopping cart software.

After discussing with Professor Beran Necat, I have determined to use a Python dictionary data structure to hold the data and using inheritance to manipulate the data across the system. This program allows users such as site admin, sellers and customers to interact with the system simultaneously while preserving data (session) integrity.

I have spent close to 200 hrs on building this program and achieve a deeper understanding of this Object-oriented Information system.

The system comprises 14 different classes:

datastorage class

This class provides an initial dataset to facilitate all the CRUD operations throughout the entire program. All the Python classes in this system software use this data set to store and retrieve data during the runtime. According to this data structure, there are three different users: site owner, seller and customer. This class does not contain any methods. It incorporates a single class variable to hold the entire dataset.

admin class

This class provides utility methods for admin-related tasks, such as displaying all the shoppers, and all the sellers to the website owner. It also generates the admin user interface.

customer class

This class provides utility methods for customer-related tasks, such as product search, adding a product to the shopping cart, displaying the shopping cart for the logged-in user, product checkout and log out functionality. It also generates the customer user interface.

seller class

This class provides utility methods to generate the seller user interface. It also contains seller related functionality, such as product search, which shows all the products that belong to the currently logged-in sellers. The seller can also add new products, update and delete existing products, and check pending orders from different customers.

order class

This class provides utility methods order related tasks. The primary responsibility of this class is to obtain all the required user information from the application UI and transfer it to the data access layer.

This class is also liable for monitoring the following bases-cases:

- The purchaser might try to check out an empty cart.
- The purchaser adds one or more products to the shopping cart and proceeded to the order page, but then add another product. The system must maintain the state by accessing the data layer.
- Several buyers might endeavour to buy the same product, if so data access layer must update the presentation layer.
- The user successfully executed all the required steps to complete the order. At this stage, the data access layer must have the latest data from the local storage and pass it down to the business layer. If the user visits the checkout page, it should be empty.
- Simultaneously, when the order is successfully handled by the system, the order class is also responsible for generating a six digit ASCII code which considered being the order number.
- Once the unique order number is created, the business logic will update the sellers account to show the pending orders.

```
...
        order_menu_selection = prompt(order_menu)
        # if the user decided to check out the current order
        if order_menu_selection['order_menu-selection'] == 'Yes':
            if len(check_out_sku) == 0:
                # if the user decided to check out an empty shopping cart, then handle this base-case
                print("Your shopping basket is empty")
                # update the state, so the shopping cart class is aware of it
                Order.status = "Empty"
            else:
                # the shopping cart is not empty, look up seller by ids
                for i in range(len(seller_id)):
                    # retrieve the seller record
                    record = Order.__ecommerce_data.get(seller_id[i])
                    # for each sku in the customer cart check if it's a match with the seller sku data list
                    for r in range(len(record["sku"])):
                        # check the added skus are available in the data storage
                        if check_out_sku[i] == record["sku"][r]:
                            # for debug purpose only
                            # print("Before ", Order.__ecommerce_data.get(seller_id[i])["stock"][r])
                            # update the new stock qty for each seller
                            Order.__ecommerce_data.get(seller_id[i])["stock"][r] = \
                                Order.__ecommerce_data.get(seller_id[i])["stock"][r] - check_out_qty[i]
                            # add order details to the seller account
                            import os
                            import base64
                            order_id = base64.b64encode(os.urandom(6)).decode('ascii')
                            while order_id not in Order.__ecommerce_data.get(seller_id[i])["order_id"]:
                                Order.__ecommerce_data.get(seller_id[i])["order_id"].append(order_id)
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_full_name"].append(
                                customer_record["first_name"] + " " + customer_record["last_name"])
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_id"].append(account_number)
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_product_name"].append(
                                check_out_product_name)
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_sku"].append(str(check_out_sku)[1:-1])
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_qty"].append(
                                str(check_out_qty)[1:-1])
                            Order.__ecommerce_data.get(seller_id[i])["order_shipping_status"].append(
                                "Order is awaiting picking")
                            # Order.__ecommerce_data.get(seller_id[i])["order_customer_total"].append(
                            #     check_out_qty * check_out_price)
                            Order.__ecommerce_data.get(seller_id[i])["order_customer_price_per_unit"].append(
                                check_out_price)

                print("Your order has been successfully processed")
                # update the state, so the shopping cart class is aware of it
                Order.status = "Success"
        if order_menu_selection['order_menu-selection'] == 'No':
            # update the state, so the shopping cart class is aware of it
            Order.status = "No"
        # return the order status for later use
        return Order.status
```

orderstatus class

This class provides the site admin or the 3rd party seller to revise the shipping status. Such that, Order is awaiting picking, Order is shipped, Order is delayed, or Order is delivered by using the six digits unique order number which was generated by the order class.

product class

This class provides utility methods for all the product-related actions such as add products, edit existing products, deleting a product, and show all products. The customer, admin and seller classes are composite of the product class in order to perform the basic functionality.

productsearch class

This class provides utility methods to search products by keywords and return the results from the storage.

shoppingcart class

This class provides utility methods for shopping cart-related tasks. Such as, adding new products to the shopping cart, current cart lookup, and checkout. When the user selects the checkout method, it created a new instance of the order class. In the OOP paradigm, this is a relationship known as has a relationship.

payment_interface class

The payment interface is an abstract class, which means there is no actual implementation that takes place inside this file. The payment class must employ inheritance and implement all the abstract methods specified in this class structure.

payment class

This class provides utility methods for payment-related tasks. This class inherits its basic characteristics from the payment_interface class by utilizing the OOP concept known as `is an association`. This class must implement all the abstract methods or else it will cause an error.

user class

This class provides utility methods to log in users and create new users in the local data storage. The new users can be customers and 3rd party sellers. The login method implemented inside this class check the user credentials and create an instance of that specific user type.

```
...

    def login(self, account_number, password):
        """
        This method validated the user credentials based on user id, and password, and then it will route the user
        to a specific user dashboard.
        :param account_number:
        :param password:
        :return:
        """
        # if the account number exist then retrieve the record
        if User.__ecommerce_data.get(account_number):
            # set the current record for validation
            self.record = User.__ecommerce_data.get(account_number)
            if self.record['password'] == password and self.record['type'] == "owner":
                # call admin main menu upon successful login
                admin = Admin()
                admin.site_admin(account_number)
            elif self.record['password'] == password and self.record['type'] == "seller":
                # call seller menu upon successful login
                seller = Seller()
                seller.seller_admin(account_number)
            elif self.record['password'] == password and self.record['type'] == "customer":
                # call customer menu upon successful login
                customer = Customer()
                customer.customer_admin(account_number)
            else:
                # incorrect password, set to record None for security reasons
                print("User name or password is incorrect")
                # site_menu = Menu()
                # site_menu.user_login_menu()
        else:
            print("Account not found")

...
```

menu class

This class provides the main utility methods for the driver code. This is the main execution point of the program. Regardless of what tasks they have performed once they end the session, it will bring them to this main menu.

Sample work flows

The admin user

```
 User Login  (Use arrow keys)
 ❯ Login
   Register
   Exit
```

The admin login to the system

```
 User Login  Login
Please enter the user id: 10000
Please enter the password: UDwh&AWD72g21
```

```
 Admin Menu  (Use arrow keys)
 ❯ Product search
   Show all products
   Add product
   Edit product
   Delete product
   Show all sellers
   Show customers
   Current orders
   Logout
```

The admin list all the 3rd party sellers

```
  Admin Menu  Show all sellers
+-------------------------------------------+
|                All sellers                |
+------------------+------------------------+
| Seller full name | Email address          |
+------------------+------------------------+
| chris herzog     | chriher22@aol.com      |
| joseph smith     | JosephSSmith@rhyta.com |
+------------------+------------------------+
```

The admin list all the products that belongs to his/her store

```
 +-----------------------------------------------------------------------------------------------+
|                                       Your product list                                       |
+--------+----------------------------------------------------+------------------+--------------+
| SKU    | Product name                                       | Price per unit $ | In stock QTY |
+--------+----------------------------------------------------+------------------+--------------+
| su1001 | Python Crash Course, 2nd Edition                   | 29.99            | 10           |
| su1002 | Automate The Boring Stuff With Python, 2nd Edition | 33.99            | 12           |
| su1003 | Learning Python, 5th Edition                       | 33.86            | 12           |
+--------+----------------------------------------------------+------------------+--------------+
```

The admin list current pending orders

```
 +------------------------------------------------------------------------------------------------------------------------------------
|                                                                                Orders
+----------+-------------+-------------------+--------+----------------------------------------------------+------------------+---
| Order ID | Customer ID | Customer full name |  SKU   |                    Product Name                    | Price per unit $ | Ord
+----------+-------------+-------------------+--------+----------------------------------------------------+------------------+---
| PZVwELy+ |    10003    |   Malcolm Smith    | su1002 | Automate The Boring Stuff With Python, 2nd Edition |      33.99       |
+----------+-------------+-------------------+--------+----------------------------------------------------+------------------+---
 Update order status  (Use arrow keys)
 ❯ Yes
   No
```

The seller user

```
 User Login  Login
Please enter the user id: 10002
Please enter the password: UDwh&AWD72g23
```

```
 Seller Menu  (Use arrow keys)
 ❯ Product search
   Show all products
   Add product
   Edit product
   Delete product
   Current orders
   Logout
 ```

#### The seller search by the keyword, here we can see there are three results for `python`

 ```
 Please enter any keyword: python
+--------+----------------------------------------------------+-----+------------------+
|  SKU   |                    Product name                    | QTY | Price per unit $ |
+--------+----------------------------------------------------+-----+------------------+
| su1001 |          Python Crash Course, 2nd Edition          |  10 |      29.99       |
| su1002 | Automate The Boring Stuff With Python, 2nd Edition |  12 |      33.99       |
| su1003 |            Learning Python, 5th Edition            |  12 |      33.86       |
+--------+----------------------------------------------------+-----+------------------+
 Seller Menu  (Use arrow keys)
 ❯ Product search
   Show all products
   Add product
   Edit product
   Delete product
   Current orders
   Logout
```

The customer user

```
 User Login  Login
Please enter the user id: 10004
Please enter the password: UDwh&AWD72g24
```

The customer is looking at all the products from the third-party sellers and the site owner

```
 Customer Menu  Add to shopping cart
+------------------------------------------------------------------------------------------------------------------------------------------
|                                                            Products from all stores
+--------------+-------------+--------+----------------------------------------------------------------------+------------------+----
| Selection id | Seller name | SKU    | Product name                                                         | Price per unit $ | In
+--------------+-------------+--------+----------------------------------------------------------------------+------------------+----
| 1            | 10000       | su1001 | Python Crash Course, 2nd Edition                                     | 29.99            | 10
| 2            | 10000       | su1002 | Automate The Boring Stuff With Python, 2nd Edition                   | 33.99            | 12
| 3            | 10000       | su1003 | Learning Python, 5th Edition                                         | 33.86            | 12
| 4            | 10001       | ch1004 | Calphalon Classic Oil-Infused Ceramic PTFE and PFOA Free Cookware    | 299.99           | 100
| 5            | 10001       | ch1005 | Silicone Cooking Utensil Set                                         | 21.99            | 112
| 6            | 10001       | ch1006 | Flour Water Salt Yeast: The Fundamentals of Artisan Bread and Pizza  | 17.99            | 23
| 7            | 10002       | jo1007 | Ball Complete Book of Home Preserving                                | 19.99            | 10
| 8            | 10002       | jo1008 | Stainless Steel Mixing Bowl Set                                      | 24.99            | 100
+--------------+-------------+--------+----------------------------------------------------------------------+------------------+----
```

The customer has added two items to the shopping cart by using the unique SKU generated by the system

```
 Please enter the sku: jo1008
Please enter the qty: 2
```

The customer is viewing the shopping cart

```
 +--------------+---------------------------------+--------+-------------------+-----+
| Seller name  | Product name                    | SKU    | Price per unit $  | QTY |
+--------------+---------------------------------+--------+-------------------+-----+
| joseph smith | Stainless Steel Mixing Bowl Set | jo1008 | 24.99 * 2 = 49.98 | 2   |
+--------------+---------------------------------+--------+-------------------+-----+
+--------------+
| Cart total $ |
+--------------+
|    49.98     |
+--------------+
```

The customer is ordering out the product

```
 Would you like to proceed ?  (Use arrow keys)
 ❯ Yes
   No
Your order has been successfully processed
```

The customer trying to check out again but the shopping cart is empty

```
 +-------------+--------------+-----+------------------+-----+
| Seller name | Product name | SKU | Price per unit $ | QTY |
+-------------+--------------+-----+------------------+-----+
+-------------+--------------+-----+------------------+-----+
+--------------+
| Cart total $ |
+--------------+
|      0       |
+--------------+
```

The seller receives the order

```
 Please enter the user id: 10002
Please enter the password: UDwh&AWD72g23
Seller Menu  Current orders
+-----------------------------------------------------------------------------------------------------------------------------------
|                                                                           Orders
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
| Order ID | Customer ID | Customer full name |   SKU    |             Product Name            | Price per unit $ | Ordered QTY |
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
| JLZakRvV |    10004    |   Pascal Brogdon   | 'jo1008' | ['Stainless Steel Mixing Bowl Set'] |     [24.99]      |      2      | Or
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
Update order status  (Use arrow keys)
 ❯ Yes
   No
```

The seller update the shipping status

```
 Update order status  Yes
Current shipping status  Order is delayed
Please enter the order id: JLZakRvV
```

```
 +---------------------------------------------------------------------------------------------------------------------------------
|                                                                      Orders
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
| Order ID | Customer ID | Customer full name |   SKU    |             Product Name            | Price per unit $ | Ordered QTY | Sh
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
| JLZakRvV |    10004    |   Pascal Brogdon   | 'jo1008' | ['Stainless Steel Mixing Bowl Set'] |     [24.99]      |      2      | Or
+----------+-------------+--------------------+----------+-------------------------------------+------------------+-------------+---
```