



**Faculty of Engineering & Technology Electrical & Computer
Engineering Department**

Linux Laboratory – ENCS3130

**Python Project
E-commerce System**

Prepared by:

| | |
|-----------------|---------|
| Lama Nasser | 120090 |
| Maha Maher Mali | 1200746 |

Instructor: Dr. Aziz Qaroush

Teacher Assistant: Eng. Tareq Zidan

Section: 2

Date: 30-8-2023

Abstract

This is a python project using PyCharm to make an E-commerce Online Shopping System which is a web-based platform designed for efficient online business operations and customer interactions. It features two distinct user roles: the E-commerce Administrator, responsible for managing the online store, and the Shopper, who uses the platform to browse and make purchases.

The E-commerce Online Shopping System has special features just for administrators. Admins can do things like adding, changing, and giving discounts on products. They can also manage user accounts, see lists of users, and handle customer orders. They can save important product and user information to files for safekeeping. On the other hand, shoppers have their own unique functions. They can put items in their shopping cart, check and change what's in the cart, and buy the things they want.

When the system starts, it reads information from text files to keep products and user details safe and organized. Users are checked by their user_id to make sure they're allowed to use the system. If they're not allowed, they get a message saying "Access Denied." The system also checks the data very carefully to make sure it's correct, which makes the platform work well and user-friendly. This project shows how important strong e-commerce systems are in today's digital world.

```
120019;orange juice;juice;3;400;JuiceComp;0;0
120074;orange juice;juice;3;500;JuiceComp;1;1;2;2023-01-23
18945;shirt;clothes;5;100;clothes market;1;1;3;2024-06-15
110212;Apple;fruits;2;50;fruits market;1;1;7;2022-03-20
14444;Orange;fruits;2;50;fruits market;1;1;7;2022-03-20
111112;apple;fruits;2;50;fruits market;1;1;7;2022-03-20
```

Figure 1:Products File

```
012901;Lama Nasser;19/1/2020;shopper;1;{018945:10,110212:3};1
033333;Maha Mali;18/1/2022;admin;0
```

Figure 2:Users File

Table of Contents

| | |
|--|----|
| Abstract..... | 2 |
| Code Description..... | 4 |
| The Products Class..... | 4 |
| Place An Item on Sale..... | 4 |
| Update Product..... | 4 |
| Execute Order | 5 |
| List products (str Method in product class) | 5 |
| The Users Class..... | 5 |
| Add Item to Basket | 5 |
| Update information for the User | 6 |
| Clear the Basket | 6 |
| Remove Product..... | 6 |
| Update Basket | 6 |
| Place An Order..... | 7 |
| Display All Users (str Method in user class) | 7 |
| The Main Class | 8 |
| Is Product Exist..... | 9 |
| Is User Exist..... | 9 |
| Add Product | 10 |
| Place Item on Sale..... | 11 |
| Update Product..... | 11 |
| Update User..... | 12 |
| Display All Users..... | 12 |
| Display Products | 13 |
| Add Product to The Basket..... | 13 |
| Display Basket | 14 |
| Update Basket | 15 |
| Place An Order..... | 15 |
| Execute Order | 15 |
| Save Products Information..... | 16 |
| Save Users Information..... | 16 |
| List Shoppers | 16 |

| | |
|---------------------------------|----|
| Test Cassese..... | 17 |
| File and Login..... | 17 |
| Add Product | 18 |
| Place an item on sale..... | 20 |
| Update product..... | 21 |
| Add a new user..... | 22 |
| Update user | 22 |
| Display all users..... | 23 |
| List products..... | 24 |
| List shoppers | 28 |
| Add product to the basket | 29 |
| Display basket..... | 30 |
| Update basket..... | 31 |
| Place order..... | 33 |
| Execute order | 34 |
| Save products to a file..... | 34 |
| Save users to a text file | 34 |
| Exit | 35 |
| Appendix..... | 36 |
| Products class..... | 36 |
| User Class | 37 |
| Main Class | 38 |

Table of Figures

| | |
|---|----|
| Figure 1:Products File..... | 2 |
| Figure 2:Users File | 2 |
| Figure 3:User File..... | 17 |
| Figure 4: Product File | 17 |
| Figure 5:Login with ID does not exist | 17 |
| Figure 6:Login as admin | 17 |
| Figure 7:Login As shopper..... | 18 |
| Figure 8:Shopper cannot add product..... | 18 |
| Figure 9: Admin can add product..... | 19 |
| Figure 10:Product is on sale | 20 |
| Figure 11:Add product to sale..... | 20 |
| Figure 12:Product does not exist..... | 20 |
| Figure 13:Menu to update the product..... | 21 |
| Figure 14:product Before Update Name | 21 |
| Figure 15: product After Update Name..... | 21 |
| Figure 16:Add new user..... | 22 |
| Figure 17:Update the date of birthday for user | 22 |
| Figure 18:Before and after update the date of birthday for user | 23 |
| Figure 19:Display All user | 23 |
| Figure 20:List All products..... | 24 |
| Figure 21: List Offers Products..... | 25 |
| Figure 22:List Category Products..... | 26 |
| Figure 23: List Name Of product | 27 |
| Figure 24: shoppers who have added items for their basket..... | 28 |
| Figure 25: all shoppers who have unprocessed order | 29 |
| Figure 26:Add product to the basket..... | 29 |
| Figure 27:Display basket..... | 30 |
| Figure 28:Clear the basket..... | 31 |
| Figure 29: Remove a specific product from the basket..... | 31 |
| Figure 30: Update the number of items for a specific product | 32 |
| Figure 31:Admin cannot place an order..... | 33 |
| Figure 32:Place an Order..... | 33 |
| Figure 33:Excute order..... | 34 |
| Figure 34:Save product to file..... | 34 |
| Figure 35:Save user to file..... | 34 |
| Figure 36:Exit | 35 |

Code Description

The Products Class

This class, called "Product," is like a digital description of a product that can be found in an online store or database. It includes important information about the product, such as its name, category (like electronics, clothing, or food), price, how many of them are available in the store, and who supplies it.

Additionally, it can tell you if the product is currently on sale with a special offer price and when that offer expires. It also helps keep track of how many of these products are sold, reducing the inventory when someone buys one. So, in a way, it's like an online product card that stores all the details you need when shopping online.

Place An Item on Sale

This function, does something like putting a product on sale in an online store. When a product is on sale, it means you can buy it at a special, discounted price for a limited time.

Update Product

This function, is like editing the information of a product in an online store. It allows you to change details about the product, such as its name, category, regular price, how many are available in stock, who supplies it, and whether it's on sale or not

Execute Order

This function, is like subtracting items from a store's inventory when someone buys them. When a customer places an order and buys a certain number of items, this function reduces the quantity of those items available in the store.

List products (str Method in product class)

This function, is like making a product card. It collects information about a product, like its name, price, and how many are available. If the product is on sale, it also shows the sale price and when the sale ends.

The Users Class

This class, called "User," is like a digital profile for people who use an E-commerce system. It keeps important information about them, such as their name, a unique ID, their birthdate, and what role they have in the system, like a regular user or an admin with special privileges. It also knows if they are currently using the system. Users can use this class to do things like adding items to a virtual shopping cart called a "basket," updating their information, or placing an order when they're done shopping. So, in a nutshell, the User class helps the computer system remember who its users are and what they're doing.

Add Item to Basket

This function is like adding items to your E-commerce system's (online shopping cart). When you use it, you tell the computer which product you want to buy (using a special ID), and how many of that product you want. The computer keeps track of what you've chosen and how many you want to purchase. So, it's like putting things in your digital shopping cart so you can buy them later.

Update information for the User

The function use to update the information of the user such as: name, date of birthday ...etc, when you use it, you can change your name, date of birth, role (like whether you're a regular user or an admin), and whether you're active or not (like if you're currently logged in). You can also update your digital shopping cart (basket) and your order status. It's a way to make sure your profile and shopping information are all up to date.

Clear the Basket

When you use this function, all the items you added to your cart will be removed, and your cart will be empty. It's a way to start fresh if you change your mind about what you want to buy or if you want to add different items to your cart.

Remove Product

This function enables you to delete a specific item from your E-commerce system's. If the product you wish to delete is already in your basket, you may use this method to remove it so you don't wind up buying it.

Update Basket

This function enables you to modify the quantity of a certain item in your E-commerce system's. If the product is already in your basket, this method will change the quantity to the new amount once you supply the product ID and the new quantity you want.

Place An Order

The E-commerce system determines if you've already placed an order when you use this function. This method changes the order status from 0 to 1 and confirms that your order has been successfully placed if you haven't yet made an order (the order status is 0). To prevent you from unknowingly ordering the same items twice, it notifies you that your order has already been placed if it has already been placed (the status is 1).

Display All Users (str Method in user class)

This function creates a detailed profile of an E-commerce system user. It includes their ID, real name, birthdate, and role (like a regular user or administrator). It also indicates if the user is currently active in the system.

Moreover, it displays the items in the user's digital shopping basket, specifying the products and quantities. If the user has not placed an order, it mentions that they are still adding items. When they finish shopping, it shows that they're ready to make a purchase. In essence, it's like a digital user ID card that tells you everything about the user and their shopping preferences in the computer system.

The Main Class

When you run the main class, you initiate an E-commerce System with a login process and a menu-driven interface. Here's a summary of what happens:

Login Process

- You start by logging into the system, where you input your user ID.
- The system checks if the ID exists and identifies your role as either a "shopper" or an "admin."
- Once logged in, you receive a welcome message based on your role.

Main Menu:

- After logging in, you are presented with a main menu.
- The menu offers various options for performing actions within the E-commerce System.

Menu Options:

- Depending on your role (shopper or admin), you have different permissions for menu options.
- Shopper (role 0) has limited permissions compared to the admin (role 1).

Available Actions:

- The available actions in the E-commerce System encompass a range of functionalities, including product and user management, basket operations, order processing, data storage, and system exit.

Permission Control:

- The system enforces role-based permissions, allowing admins to perform administrative tasks and shoppers to manage their shopping experience.

User Interaction:

- You interact with the system by choosing options from the menu, and the system responds accordingly.

Logging Out:

- You can log out at any time by selecting the "Exit" option.

Invalid Input Handling:

- The system provides feedback for invalid inputs and guides you to make valid selections.
-

Is Product Exist

This function checks if a product with a specific ID exists in the system. It goes through the list of all products available and compares each product's ID with the one you provided. If it finds a matching ID, it returns "1," indicating that the product exists; otherwise, it returns "0," meaning the product is not found.

Is User Exist

This function checks if a user with a specific ID exists in the system. It goes through the list of all users and compares each user's ID with the one you provided. If it finds a matching ID, it returns "1," indicating that the user exists in the system; otherwise, it returns "0," meaning the user is not found.

Add Product

This function, allows users to add new items for sale on an online shopping, according these steps:

- Users enter essential details like the product's ID (a unique code), name, category, price, available quantity (inventory), and the supplier's name.
- The system checks if the entered product ID is six digits long and not used for any other product. If it's incorrect, users must provide a valid ID.
- Users specify the product's name, category (e.g., Electronics or Clothing), price, inventory level, and supplier.
- Users decide whether the product is on sale. If it is, they provide the sale price and the date until which the sale is valid. If not, they indicate that there's no sale.
- The system then creates the new product listing and adds it to the online store's inventory.
- Users receive confirmation that the product has been successfully added to the online store.

Place Item on Sale

This function, is a tool for online shop owners to put a product on sale or change its sale details it's like changing the price tag on a product in an online store to show that it's on sale, and specifying the new discounted price and the sale's duration, according these steps:

- The shop owner chooses a product by entering its unique ID.
- The system checks if this product exists in the store's inventory.
- If the product exists, the system checks if it's already on sale and shows the current sale details if applicable.
- If the product is not on sale, the shop owner can set a new sale price and specify how long the sale will last.
- Once the new sale details are provided, the system updates the product's information, and the product becomes available at the discounted price for the specified duration.

Update Product

This function allows the shop owner to modify information about a product in the store. It first asks for the product's unique ID to identify which product needs updating. Then, it provides a menu of options for what aspects of the product can be changed, such as the product's name, category, price, inventory, supplier, sale status, sale price, and sale expiration date.

The shop owner selects the specific information they want to update and provides the new values. Once the updates are made, the system stores the changes, and the product's information is updated successfully.

Update User

This function is used to create and add a new user to the system. It prompts the administrator to input specific information about the user:

- User ID: A unique 6-digit identifier for the user.
- User Name: The name of the user.
- Date of Birth: The user's date of birth.
- Role: Whether the user is a shopper or an admin.
- Active Status: Whether the user is active (1) or not (0).

The function ensures that the user ID is exactly 6 digits and checks if the ID already exists in the system. It also validates the role and active status inputs to ensure they are correct. Once all the required information is provided, the function creates a new user object with this data and adds it to the list of users. This way, a new user can be registered in the system successfully.

Display All Users

This function shows a list of all the users in the system and provides their details, making it easy for the administrator to view and manage user information.

Display Products

This function allows you to list products in different ways based on your choice:

- List All Products: It shows a list of all available products in the store.
- List Offers Products: It displays products that are currently on sale with special offers.
- List Category Products: You can input a specific category, and the function will show all products belonging to that category.
- List Name Products: You can input the name of a product, and the function will display all products with a matching name.

For each of these options, the function goes through the product list and prints out the details of the selected products. The product details are obtained using the `__str__` function of the product class.

Add Product to The Basket

This function allows a user to add a product to their shopping basket. According these steps:

- Input Product ID: The user enters the unique ID of the product they want to add to their basket. The function checks if the product with that ID exists in the store.
- Input Quantity: The user specifies the quantity (number of items) of the selected product they want to add to their basket.
- Adding to Basket: The function then finds the user with the given user ID and adds the specified product with the chosen quantity to their shopping basket.
- Confirmation: After adding the product to the basket, the function displays the updated information about the user, including the contents of their basket.

Display Basket

This function serves to display the contents of a user's shopping basket and calculate the total cost of the items. According these steps:

- **Input User ID:** This function takes a user's unique ID to identify whose shopping basket to show.
- **Display Basket Contents:** It finds the user with the matching ID and lists the items in their basket.
- **Display Product Details:** For each item in the basket, it shows the product's ID, name, category, price, supplier, and whether it's on sale. If on sale, it displays the offer price and validity date.
- **Calculate Item Cost:** The function computes the cost of each item by multiplying its quantity by the price and displays this cost.
- **Total Basket Cost:** It adds up the costs of all items in the basket to provide the overall cost of the entire shopping basket.

Update Basket

This function allows a user to make changes to their shopping basket. According these steps:

- **Clear the Basket:** Users can choose to empty their shopping basket, removing all items, effectively starting with an empty basket.
- **Remove a Specific Product:** If users want to remove a particular product from their basket, they provide the product's ID, and the function removes it if it's present in the basket.
- **Update the Number of Items:** If users wish to adjust the quantity of a specific product in their basket, they provide the product's ID and the new quantity, allowing for easy modifications.

Place An Order

This function is like the "checkout" button. When you press it, you're telling the system, "I'm ready to buy these things." It takes your ID, finds your cart, and helps you complete your purchase.

Execute Order

Think of this like the moment when you're shopping online and you've already chosen what you want to buy. When you click "Place Order" on a website, this function does something similar. It looks at your list of items, figures out how many of each you want, and then actually completes the purchase for you.

Save Products Information

This function helps you store information about different products, like their names, prices, and more, in a computer file. Think of it like creating a digital catalog for all the products you have in a store. This way, you can access and use this information later whenever you need it.

Save Users Information

This function allows you to save information about users, including their names, roles, and more, into a computer file. It's like creating a digital record of all the users in your system. This data can be retrieved and used later when needed.

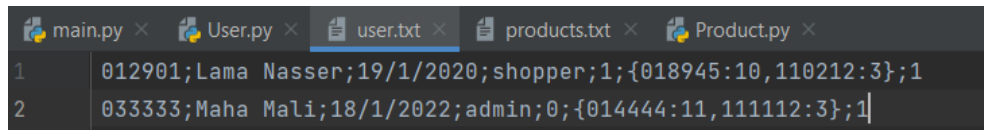
List Shoppers

This function provides options to list and display information about shoppers in the system. According these steps:

- **List All Shoppers:** This option provides a list of all shoppers, including their names and unique IDs, regardless of their shopping activity.
- **List Shoppers with Items in Their Basket:** This option identifies and displays shoppers who have actively added items to their shopping baskets.
- **List Shoppers with Unprocessed Orders:** Shoppers who have placed orders but haven't had them processed or completed yet are listed under this choice.
- **List Shoppers with Requested Orders:** This option shows users who have initiated an order but haven't completed the purchase process.

Test Cases File and Login

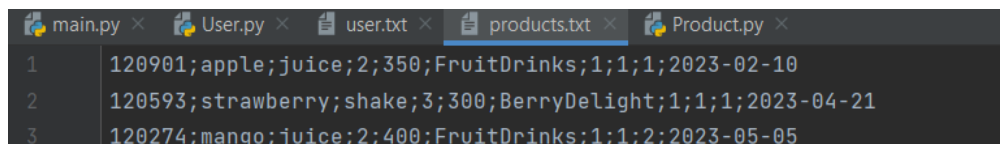
➤ User File.



```
main.py × User.py × user.txt × products.txt × Product.py ×  
1 012901;Lama Nasser;19/1/2020;shopper;1;{018945:10,110212:3};1  
2 033333;Maha Mali;18/1/2022;admin;0;{014444:11,111112:3};1
```

Figure 3: User File

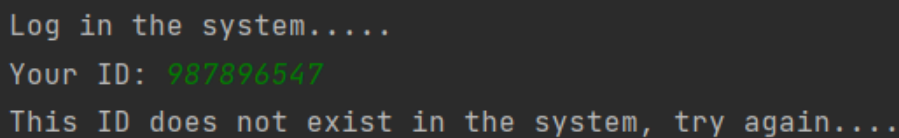
➤ Product File.



```
main.py × User.py × user.txt × products.txt × Product.py ×  
1 120901;apple;juice;2;350;FruitDrinks;1;1;1;2023-02-10  
2 120593;strawberry;shake;3;300;BerryDelight;1;1;1;2023-04-21  
3 120274;mango;juice;2;400;FruitDrinks;1;1;2;2023-05-05
```

Figure 4: Product File

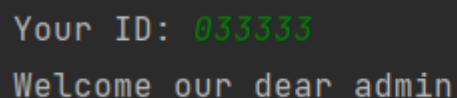
➤ user with ID does not exist



```
Log in the system.....  
Your ID: 987896547  
This ID does not exist in the system, try again....
```

Figure 5: Login with ID does not exist

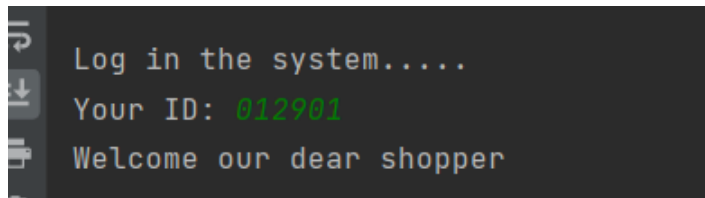
➤ Login As Admin



```
Your ID: 033333  
Welcome our dear admin
```

Figure 6: Login as admin

- Login As shopper

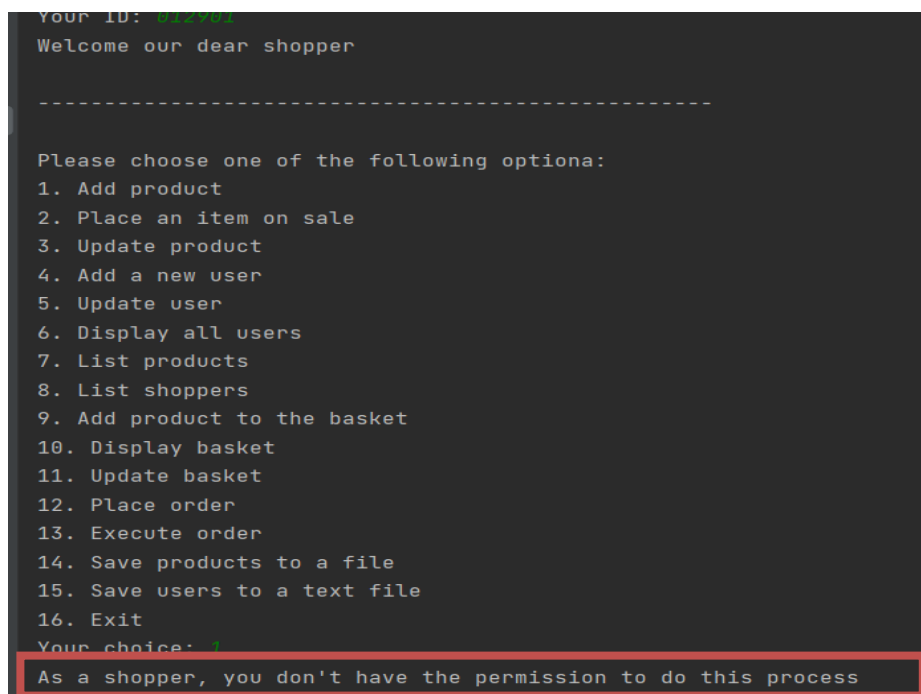


```
Log in the system.....  
Your ID: 012901  
Welcome our dear shopper
```

Figure 7:Login As shopper

Add Product

- When the shopper attempts to add product error message will appear.



```
YOUR ID: 012901  
Welcome our dear shopper  
  
-----  
  
Please choose one of the following options:  
1. Add product  
2. Place an item on sale  
3. Update product  
4. Add a new user  
5. Update user  
6. Display all users  
7. List products  
8. List shoppers  
9. Add product to the basket  
10. Display basket  
11. Update basket  
12. Place order  
13. Execute order  
14. Save products to a file  
15. Save users to a text file  
16. Exit  
Your choice: 1  
As a shopper, you don't have the permission to do this process
```

Figure 8:Shopper cannot add product

- When the Admin attempts to add product

```
Your choice: 1
Input the product ID: 887799
Input the product name: milkh
Input the category of the product: drink
Input the price of the product: 5
Input the inventory of the product: 100
Input the supplier of the product: milkcom
Does this product have an offer? (input 1 if yes, 0 if no): 1
Input the offer price of the product: 1
Input the valid until date for the product (d/m/y): 22/4/2024
The product has been added successfully....
```

Figure 9: Admin can add product

Place an item on sale

- Product is already on sale.

```
Your choice: 2
Input the ID of the product: 014444
This product is already on sale
the offer price is: 7
Thr valid until date is: 2022-03-20
```

Figure 10:Product is on sale

- Product is not on sale.

```
Your choice: 2
Input the ID of the product: 120019
Input the offer price of this product: 3
Input the valid until date of this product: 5/1/2024
The product has been placed on sale successfully...
```

Figure 11:Add product to sale

- Product does not exist in product.txt

```
Your choice: 2
Input the ID of the product: 8888888
No product with this ID,try again...
```

Figure 12:Product does not exist

Update product

- When user choice 3, the menu will appear to choose what exactly he needs to update.

Note: I need to update just the name in this case, but when you are trying the code, you can update anything's according to the menu.

```
Please Choose What you want to Update For The Product:
1. Product Name
2. Product Category
3. Product Price
4. Product Inventory
5. Product Supplier
6. Product Has an Offer
7. Product Offer Price
8. Product Valid Until
```

Figure 13: Menu to update the product

```
111112;Banana;fruits;2;50;fruits market;1;7;20/3/2022
```

Figure 14: product Before Update Name

```
Your choice: 1
Input the new name of the product: apple

The product has been updated successfully....
```

```
111112;apple;fruits;2;50;fruits market;1;1;7;2022-03-20
```

Figure 15: product After Update Name

Add a new user

- When user choice 4, he can add new user.

```
Your choice: 4
Input the user ID: 887766
Input the user name: lamaa
Input the user's date of birth: 17/4/2002
Input the role of the user (shopper/admin): admin
Is the user active (1) or not (0): 1
The user has been added successfully
```

Figure 16:Add new user

Update user

- When user choice 5, the menu will appear to choose what exactly he needs to update.
- **Note: I need to update just the date of birthday in this case, but when you are trying the code, you can update anything's according to the menu**

```
Your choice: 5
Input the ID of the user: 033333
Please Choose What you want to Update For The User:
1. User Name
2. User Date Of Birthday
3. User Role
4. User Active
Your choice: 2
Input the user's date of birth (d/m/y): 3/10/2002

The user has been updated successfully...
```

Figure 17:Update the date of birthday for user


```
033333;Maha Mali;18/1/2022;admin;0;{014444:11,111112:3};1
```

```
033333;Maha Mali;2002-10-03;admin;0;{'014444': '11', '111112': '3'};1
```

Figure 18: Before and after update the date of birthday for user

Display all users

```
Your choice: 6

Users in the system.....

Shopper ID: 12901
Name: Lama Nasser
Date of birth: 2020-01-19
Role: shopper
Lama Nasser is active
products in the basket:
Product ID: 018945, Number of Items: 10
Product ID: 110212, Number of Items: 3

Lama Nasser finished adding items

-----

Shopper ID: 33333
Name: Maha Mali
Date of birth: 2022-01-18
Role: admin
Maha Mali is not active
```

Figure 19: Display All user

List products

- List All Products.

```
main x
11. Update basket
12. Place order
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 7

Please Choose What you want to List From Products:
1. List All Products
2. List Offers Products
3. List Category Products
4. List Name Products
Your choice: 1

Product ID: 120019
Name: orange juice
Category: juice
Price: 3
Inventory: 400
Supplier: JuiceCamp
Offer price: 1
The product is valid until: 2023-01-20

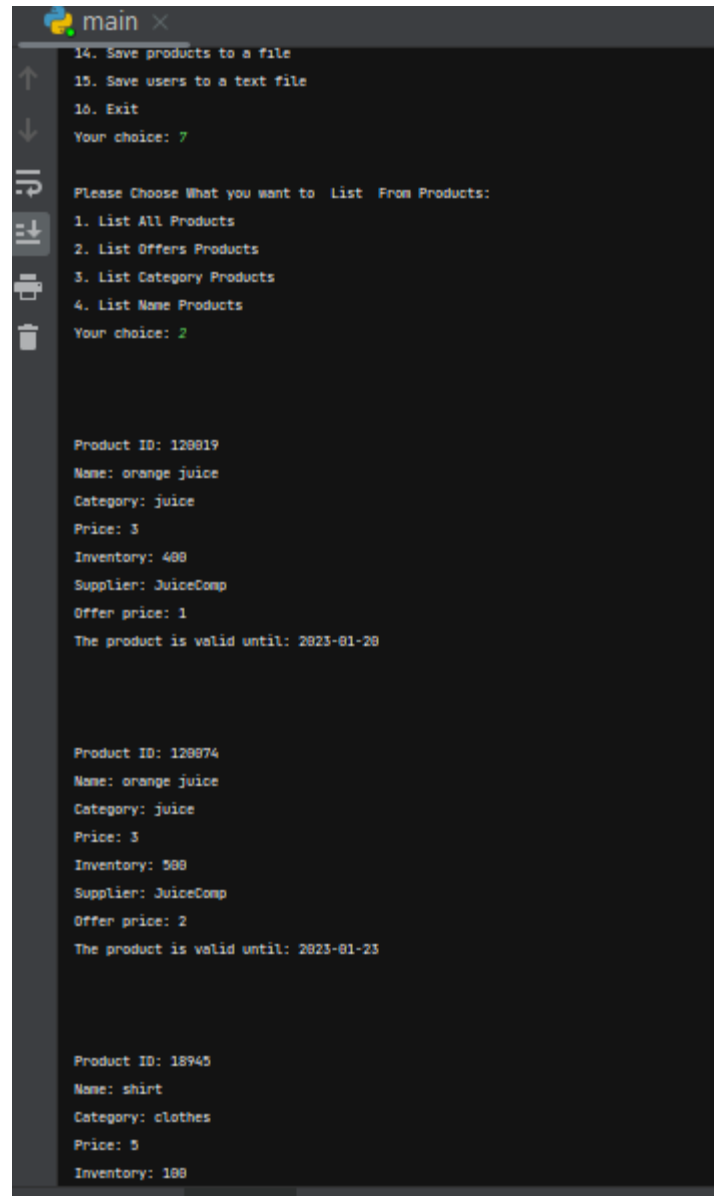
Product ID: 120074
Name: orange juice
Category: juice
Price: 3
Inventory: 500
Supplier: JuiceCamp
Offer price: 2
The product is valid until: 2023-01-23

Product ID: 18943
Name: shirt
Category: clothes
Price: 5
Inventory: 100
Supplier: clothes market
Offer price: 3
The product is valid until: 2024-06-15

Product ID: 110212
Name: Apple
```

Figure 20:List All products

➤ List Offers Products



```
main x
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 7

Please Choose What you want to List From Products:
1. List All Products
2. List Offers Products
3. List Category Products
4. List Name Products
Your choice: 2

Product ID: 129819
Name: orange juice
Category: juice
Price: 3
Inventory: 498
Supplier: JuiceComp
Offer price: 1
The product is valid until: 2023-01-20

Product ID: 129874
Name: orange juice
Category: juice
Price: 3
Inventory: 500
Supplier: JuiceComp
Offer price: 2
The product is valid until: 2023-01-23

Product ID: 18945
Name: shirt
Category: clothes
Price: 5
Inventory: 100
```

Figure 21: List Offers Products

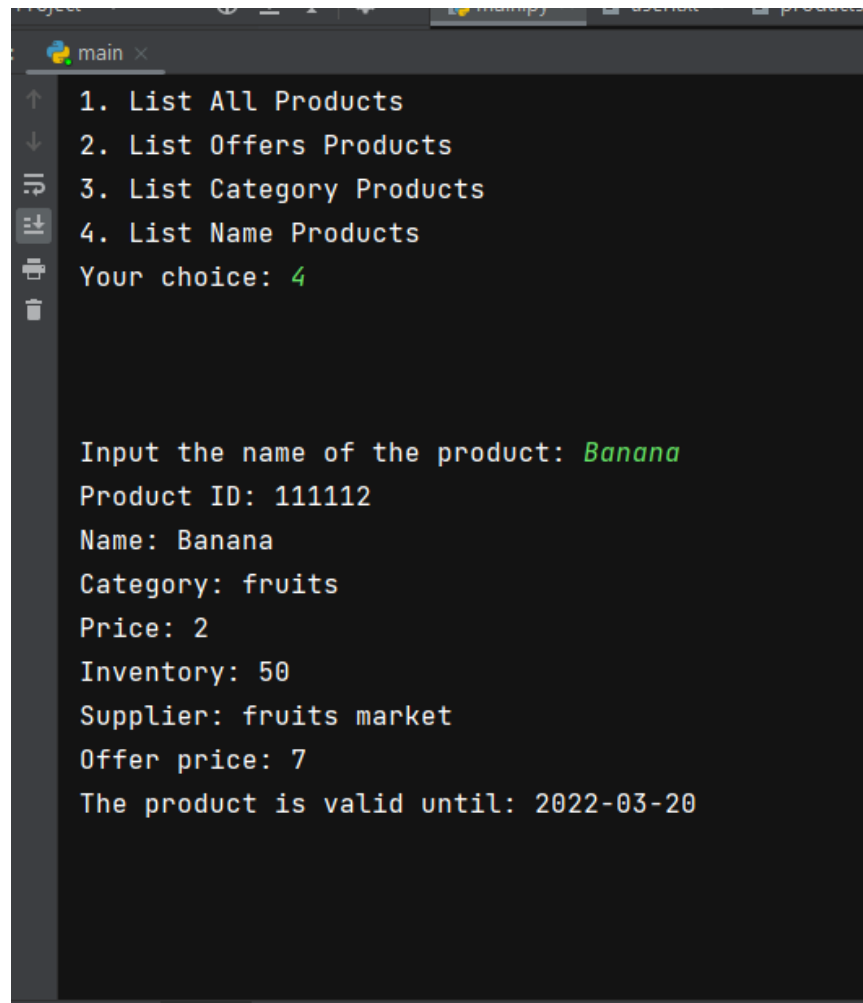
➤ List Category Products

```
1. List All Products
2. List Offers Products
3. List Category Products
4. List Name Products
Your choice: 3

Input the category: clothes
Product ID: 18945
Name: shirt
Category: clothes
Price: 5
Inventory: 100
Supplier: clothes market
Offer price: 3
The product is valid until: 2024-06-15
```

Figure 22:List Category Products

➤ List Name Products



```
main x
1. List All Products
2. List Offers Products
3. List Category Products
4. List Name Products
Your choice: 4

Input the name of the product: Banana
Product ID: 111112
Name: Banana
Category: fruits
Price: 2
Inventory: 50
Supplier: fruits market
Offer price: 7
The product is valid until: 2022-03-20
```

Figure 23: List Name Of product

List shoppers

- When the user choice option 8 which is list shoppers then the menu will appear, to choose what exactly he wants to list.
- List all shoppers

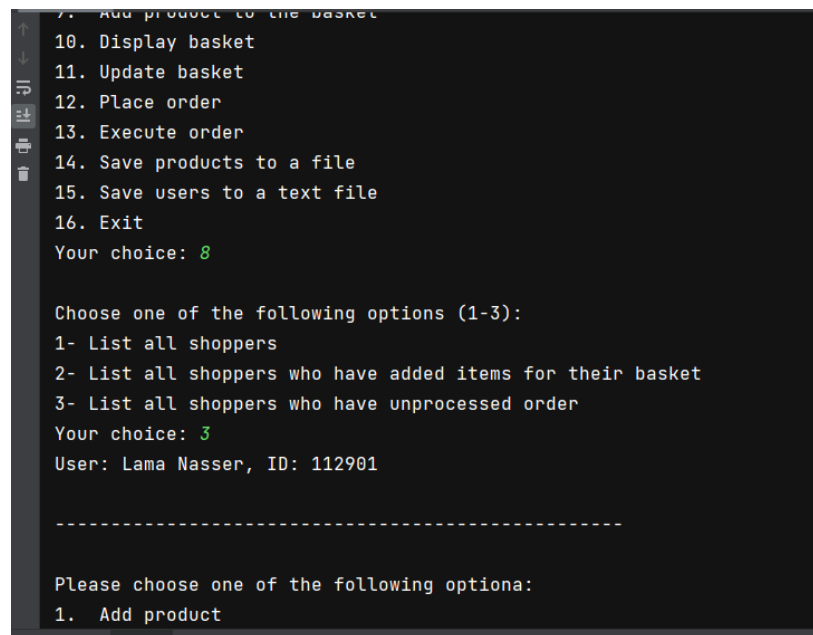
```
Choose one of the following options (1-3):  
1- List all shoppers  
2- List all shoppers who have added items for their basket  
3- List all shoppers who have unprocessed order  
Your choice: 1  
User: Lama Nasser, ID: 12901
```

- List all shoppers who have added items for their basket

```
10. Display basket  
11. Update basket  
12. Place order  
13. Execute order  
14. Save products to a file  
15. Save users to a text file  
16. Exit  
Your choice: 8  
  
Choose one of the following options (1-3):  
1- List all shoppers  
2- List all shoppers who have added items for their basket  
3- List all shoppers who have unprocessed order  
Your choice: 2  
User: Lama Nasser, ID: 112901  
  
-----  
  
Please choose one of the following options:  
1. Add product  
2. Place an item on sale
```

Figure 24: shoppers who have added items for their basket

- List all shoppers who have unprocessed order



```
7. Add product to the basket
10. Display basket
11. Update basket
12. Place order
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 8

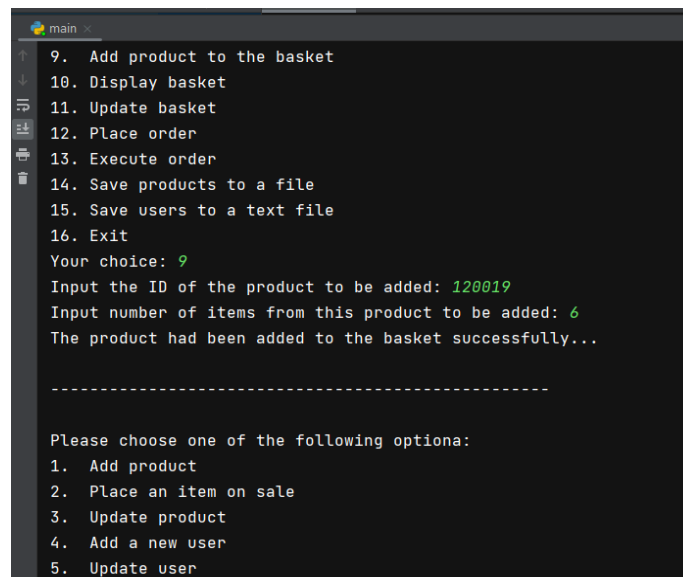
Choose one of the following options (1-3):
1- List all shoppers
2- List all shoppers who have added items for their basket
3- List all shoppers who have unprocessed order
Your choice: 3
User: Lama Nasser, ID: 112901

-----

Please choose one of the following options:
1. Add product
```

Figure 25: all shoppers who have unprocessed order

Add product to the basket



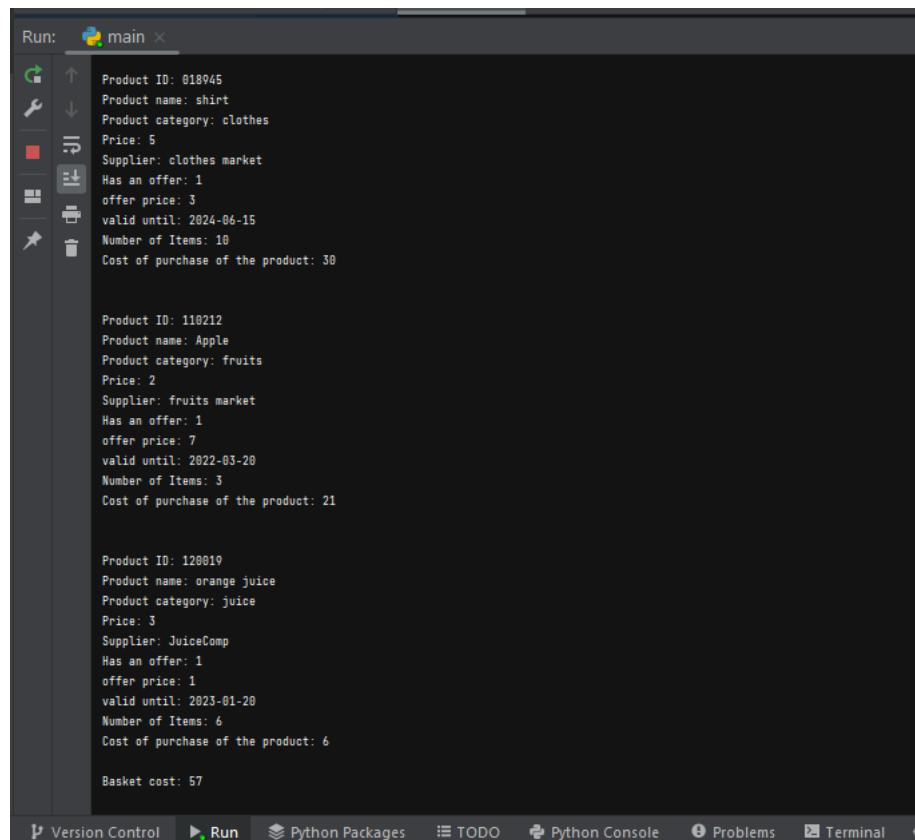
```
main x
9. Add product to the basket
10. Display basket
11. Update basket
12. Place order
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 9
Input the ID of the product to be added: 120019
Input number of items from this product to be added: 6
The product had been added to the basket successfully...

-----

Please choose one of the following options:
1. Add product
2. Place an item on sale
3. Update product
4. Add a new user
5. Update user
```

Figure 26: Add product to the basket

Display basket



```
Run: main x
Product ID: 018045
Product name: shirt
Product category: clothes
Price: 5
Supplier: clothes market
Has an offer: 1
offer price: 3
valid until: 2024-06-15
Number of Items: 10
Cost of purchase of the product: 30

Product ID: 110212
Product name: Apple
Product category: fruits
Price: 2
Supplier: fruits market
Has an offer: 1
offer price: 7
valid until: 2022-03-20
Number of Items: 3
Cost of purchase of the product: 21

Product ID: 120019
Product name: orange juice
Product category: juice
Price: 3
Supplier: JuiceComp
Has an offer: 1
offer price: 1
valid until: 2023-01-20
Number of Items: 6
Cost of purchase of the product: 6

Basket cost: 57
```

Figure 27: Display basket

Update basket

- Clear the basket

```
Please choose one of the following options (1-3):|
1- Clear the basket
2- Remove a specific product from the basket
3- Update the number of items for a specific product
Your choice: 1

The basket has been cleared successfully...
```

Figure 28: Clear the basket

- Remove a specific product from the basket

```
main
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 11

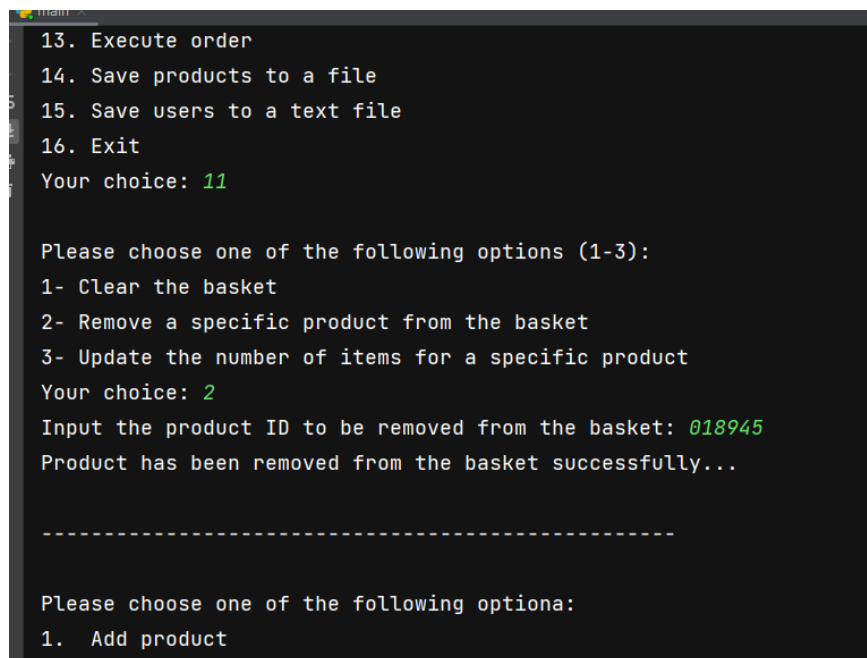
Please choose one of the following options (1-3):
1- Clear the basket
2- Remove a specific product from the basket
3- Update the number of items for a specific product
Your choice: 2
Input the product ID to be removed from the basket: 018945
Product has been removed from the basket successfully...

-----

Please choose one of the following options:
1. Add product
```

Figure 29: Remove a specific product from the basket

- Update the number of items for a specific product



```
train
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 11

Please choose one of the following options (1-3):
1- Clear the basket
2- Remove a specific product from the basket
3- Update the number of items for a specific product
Your choice: 2
Input the product ID to be removed from the basket: 018945
Product has been removed from the basket successfully...

-----

Please choose one of the following options:
1. Add product
```

Figure 30: Update the number of items for a specific product

Place order

- Admin cannot place an order

```
12. Place order
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 12
As an admin, you don't have the permission to do this process
```

Figure 31:Admin cannot place an order

- Shopper place order

```
16. Exit
Your choice: 12
Your order is already placed...
```

Figure 32:Place an Order

Execute order

```
9. Add product to the basket
10. Display basket
11. Update basket
12. Place order
13. Execute order
14. Save products to a file
15. Save users to a text file
16. Exit
Your choice: 13
User: Lama Nasser, ID: 12901 order has been executed successfully...
-----
```

Figure 33:Excute order

Save products to a file

- When user choice 14, he can input the name of file to save information of products

```
16. Exit
Your choice: 14
Input the file name to save the products information in: |
```

Figure 34:Save product to file

Save users to a text file

- When user choice 15, he can input the name of file to save information of user.

```
16. Exit
Your choice: 15
Input the file name to save the users information in: |
```

Figure 35:Save user to file

Exit

- When user choice 16, he can log out from the system.

```
16. Exit
Your choice: 16
Are you sure you want to exit without saving? (yes/no)yes

logging out has been done successfully
See you later...
```

Figure 36:Exit

Appendix

Products class

```
from datetime import date
from enum import Enum

class Categories(Enum):
    pass

class Product:
    def
__init__(self, ID, name, category, price, inventory, supplier, Has_an_offer, offer_price, valid_until)
:
    self.ID = ID
    self.name = name
    self.category = category
    self.price = price
    self.inventory = inventory
    self.supplier = supplier
    self.Has_an_offer = Has_an_offer
    self.offer_price = offer_price
    data = valid_until.split('/')
    if len(data) == 3:
        self.valid_until = date(int(data[2]), int(data[1]), int(data[0]))
    else:
        self.valid_until = ""

    def place_an_item_on_sale(self, offer_price, valid_until):
        self.Has_an_offer = 1
        self.offer_price = offer_price
        self.valid_until = valid_until

    def
updateProduct(self, name, category, price, inventory, supplier, Has_an_offer, offer_price, valid_until):
    self.name = name
    self.category = category
    self.price = price
    self.inventory = inventory
    self.supplier = supplier
    self.Has_an_offer = Has_an_offer
    self.offer_price = offer_price
    self.valid_until = valid_until

    def executeOrder(self, NumOfItems):
        self.inventory -= NumOfItems

    def __str__(self):
        info = "Product ID: "+str(self.ID)+"\n"
        info += "Name: "+self.name+"\nCategory: "+str(self.category)+"\nPrice:
"+str(self.price)+"\nInventory: "+str(self.inventory)+"\nSupplier: "+self.supplier
        if(self.Has_an_offer == 0):
            info += "\nThis product does not have an offer"
```

```
else:
    info += "\nOffer price: "+str(self.offer_price)+"\n"
    info += "The product is valid until: "+str(self.valid_until)+"\n"
return info
```

User Class

```
from datetime import date

class User():

    def __init__(self, ID, name, DoB, rule, active, basket, order):
        self.ID = ID
        self.name = name
        data = DoB.split('/')
        if len(data) == 3:
            self.DoB = date(int(data[2]), int(data[1]), int(data[0]))
        else:
            self.DoB = ""
        self.role = rule
        self.active = active
        self.basket = {}
        self.basket.update(basket)
        self.order = order

    def addItemToBasket(self, product_id, numOfItems):
        self.basket[product_id] = numOfItems

    def clearBasket(self):
        self.basket.clear()

    def removeProduct(self, product_id):
        for i in self.basket:
            if (i == product_id):
                self.basket.pop(i)

    def updateBasket(self, product_id, numOfItems):
        for i in self.basket:
            if (i == product_id):
                self.basket[i] = numOfItems

    def placeAnOrder(self):
        if(self.order == 0):
            self.order = 1
            print("The order has been placed successfully...")
        else:
            print("Your order is already placed...")
```

```

def __str__(self):
    info = "\n\nShopper ID: "+str(self.ID)+"\nName: "+self.name+"\nDate of birth: "+str(self.DoB)+"\nRole: "+str(self.role)
    if(self.active == 0):
        info += "\n"+self.name+" is not active"
    else:
        info += "\n"+self.name+" is active"

    if self.role == "shopper":
        if(len(self.basket) == 0):
            info += "\nNo products in the basket for this shopper"
        else:
            info += "\nproducts in the basket: \n"
            for i in self.basket:
                info += "Product ID: "+str(i)+", Number of Items: "+str(self.basket[i])+"\n"
            if(self.order == 0):
                info += "\n"+self.name+" did not finish adding items yet\n\n"
            else:
                info += "\n" + self.name + " finished adding items\n\n"

    return info

```

Main Class

```

from datetime import date
import Product
import User

#reading files here
products = []
users = []

#reading files
#reading and storing the users in the system
f = open("user.txt","r")
lineNum = 0
for i in f:
    lineNum += 1
    basket = {} #empty to put values in it
    data = i.split(';')
    if len(data) == 5:
        try:
            user =
User.User(int(data[0]),str(data[1]),str(data[2]),str(data[3]),int(data[4]),{},0)
            users.append(user)
        except ValueError:
            print("Invalid admin information on line #" + str(lineNum) + " in 'users.txt' file")
    elif len(data) == 7:

```



```

        temp = data[5].strip('{}').split(',')
        for j in temp:
            temp1 = j.split(':')
            key = temp1[0]
            value = temp1[1]
            basket[key] = value
        try:
            user =
User.User(int(data[0]),str(data[1]),str(data[2]),str(data[3]),int(data[4]),basket,int(data[6]
))
            users.append(user)

        except ValueError:
            print("Invalid shopper information on line #"+str(lineNum)+" in 'users.txt'
file")
        else:
            print("Invalid user information on line #" + str(lineNum) + " in 'users.txt' file")

#reading and storing the products in the system
f = open("products.txt","r")
lineNum = 0
for i in f:
    lineNum += 1
    data = i.split(';')
    category_str = str(data[2].strip())
    # Check if the category is in the mapping, if not, create a new enum member
    if category_str not in Product.Categories.__members__:
        Product.Categories.category_str = category_str
    category = Product.Categories.category_str
    if len(data) == 9:
        try:
            product =
Product.Product(int(data[0]),str(data[1]),category,int(data[3]),int(data[4]),str(data[5]),int
(data[6]),int(data[7]),str(data[8]))
            products.append(product)
        except ValueError:
            print('Invalid user information on line #'+str(lineNum)+" in products.txt file")
    elif len(data) == 7:
        try:
            product = Product.Product(int(data[0]), str(data[1]), category, int(data[3]),
int(data[4]), str(data[5]),int(data[6]),0,"")
            products.append(product)
        except ValueError:
            print('Invalid user information on line #'+str(lineNum)+" in products.txt file")

#checking if a specific product is exist in the system using the product ID
def isProductExist(product_id):
    for i in products:
        if i.ID == product_id:
            return 1
    return 0

#checking if a specific user is exist in the system using the user ID
def isUserExist(user_id):
    for i in users:
        if i.ID == user_id:

```

```

        return 1
    return 0

#add a product to the system
def addProduct():
    ID = input("Input the product ID: ")
    while 1:
        try:
            ID = int(ID)
            break
        except ValueError:
            print("Invalid input...")
            ID = input("Input the product ID: ")
    while len(str(ID)) != 6:
        print("product ID must be 6 digits only...")
        ID = input("Input the product ID: ")
    while 1:
        try:
            ID = int(ID)
            break
        except ValueError:
            print("Invalid input...")
            ID = input("Input the product ID: ")
    while isProductExist(ID):
        print("This ID is already exist")
        ID = input("Input the product ID: ")
    while 1:
        try:
            ID = int(ID)
            break
        except ValueError:
            print("Invalid input...")
            ID = input("Input the product ID: ")
    while len(str(ID)) != 6:
        print("product ID must be 6 digits only...")
        ID = input("Input the product ID: ")
    while 1:
        try:
            ID = int(ID)
            break
        except ValueError:
            print("Invalid input...")
            ID = input("Input the product ID: ")
    name = input("Input the product name: ")
    category_str = input("Input the category of the product: ")
    if category_str not in Product.Categories.__members__:
        Product.Categories.category_str = category_str
    category = Product.Categories.category_str
    price = input("Input the price of the product: ")
    price = int(price)
    inventory = input("Input the inventory of the product: ")
    inventory = int(inventory)
    supplier = input("Input the supplier of the product: ")
    while 1:
        has_an_offer = input("Does this product have an offer? (input 1 if yes, 0 if no): ")
        has_an_offer = int(has_an_offer)

```

```

        if has_an_offer == 1:
            offer_price = input("Input the offer price of the product: ")
            offer_price = int (offer_price)
            valid_until = input("Input the valid until date for the product (d/m/y): ")
            temp = valid_until.split('/')
            while len(temp) != 3:
                print("Invalid input...")
                valid_until = input("Input the valid until date for the product (d/m/y): ")
                temp = valid_until.split('/')
            product =
Product.Product(ID,name,category,price,inventory,supplier,has_an_offer,offer_price,valid_until)
            products.append(product)
            break
        elif has_an_offer == 0:
            product = Product.Product(ID, name, category, price, inventory, supplier,
has_an_offer,0,"")
            products.append(product)
            break
        else:
            print("Invalid input...")
            print("The product has been added successfully....")

def placeItemOnSale():
    ID = input("Input the ID of the product: ")
    ID = int(ID)
    while not isProductExist(ID):
        print("No product with this ID,try again...")
        ID = input("Input the ID of the product: ")
        ID = int(ID)
    for i in products:
        if i.ID == ID:
            if i.Has_an_offer == 1:
                print("This product is already on sale")
                print("the offer price is: "+str(i.offer_price))
                print("Thr valid until date is: "+str(i.valid_until))
            else:
                offer_price = input("Input the offer price of this product: ")
                offer_price = float (offer_price)
                valid_until = input("Input the valid until date of this product: ")
                temp = valid_until.split('/')
                while len(temp) != 3:
                    print("Invalid input...")
                    valid_until = input("Input the valid until date for the product (d/m/y): ")
                temp = valid_until.split('/')
                i.place_an_item_on_sale(offer_price,valid_until)
                print("The product has been placed on sale successfully...")

def updateProduct():
    ID = input("Input the ID of the product: ")
    ID = int(ID)
    while not isProductExist(ID):
        print("No product with this ID,try again...")
        ID = input("Input the ID of the product: ")
        ID = int(ID)

```

```

for i in products:
    if i.ID == ID:
        print("\n\nPlease Choose What you want to Update For The Product: ")
        print("1. Product Name ")
        print("2. Product Category ")
        print("3. Product Price")
        print("4. Product Inventory")
        print("5. Product Supplier ")
        print("6. Product Has an Offer ")
        print("7. Product Offer Price ")
        print("8. Product Valid Until ")
        choice = input("Your choice: ")
        choice = int (choice)
        while choice < 1 or choice > 8:
            print("Invalid input, choose from 1 to 8")
            print("\n\nPlease Choose What you want to Update For The Product: ")
            print("1. Product Name ")
            print("2. Product Category ")
            print("3. Product Price")
            print("4. Product Inventory")
            print("5. Product Supplier ")
            print("6. Product Has an Offer ")
            print("7. Product Offer Price ")
            print("8. Product Valid Until ")
            choice = input("Your choice: ")
            choice = int(choice)
        if choice == 1:
            name = input("Input the new name of the product: ")
            i.name = name
        elif choice == 2:
            category_str = input("Input the category of the product: ")
            if (category_str not in Product.Categories.__members__):
                Product.Categories.category_str = category_str
            category = Product.Categories.category_str
            i.category = category
        elif choice == 3:
            price = input("Input the price of the product: ")
            price = int(price)
            i.price = price
        elif choice == 4:
            inventory = input("Input the inventory of the product: ")
            inventory = int(inventory)
            i.inventory = inventory
        elif choice == 5:
            supplier = input("Input the supplier of the product: ")
            i.supplier = supplier
        elif choice == 6:
            while 1:
                has_an_offer = input("Does this product have an offer? (input 1 if yes, 0
if no): ")
                has_an_offer = int(has_an_offer)
                if has_an_offer == 1 or has_an_offer == 0:
                    i.Has_an_offer = has_an_offer
                else:
                    print("Invalid input...")
        elif choice == 7:

```

```

        if i.Has_an_offer == 1:
            offer_price = input("Input the offer price of this product: ")
            offer_price = int(offer_price)
            i.offer_price = offer_price
        else:
            print("This product is not on sale..")
    elif choice == 8:
        if i.Has_an_offer == 1:
            valid_until = input("Input the valid until date of this product (d/m/y): ")

            data = valid_until.split('/')
            while len(data) != 3:
                print("Invalid input, enter the date as this form (d/m/y)")
                valid_until = input("Input the valid until date of this product (d/m/y): ")

            data = valid_until.split('/')
            i.valid_until = date(int(data[2]), int(data[1]), int(data[0]))
        else:
            print("This product is not on sale..")
    print("\nThe product has been updated successfully....")

def addUser():
    ID = input("Input the user ID: ")
    ID = int(ID)
    while len(str(ID)) != 6:
        print("user ID must be 6 digits only...")
        ID = input("Input the user ID: ")
        ID = int(ID)
    while isUserExist(ID):
        print("This ID is already exist")
        ID = input("Input the user ID: ")
        ID = int(ID)
    name = input("Input the user name: ")
    DoB = input("Input the user's date of birth: ")
    temp = DoB.split('/')
    while len(temp) != 3:
        print("Invalid input...")
        DoB = input("Input the user's date of birth: ")
        temp = DoB.split('/')
    role = input("Input the role of the user (shopper/admin): ")
    role = role.lower()
    while role != "shopper" and role != "admin":
        print("Invalid input")
        role = input("Input the role of the user (shopper/admin): ")
    active = input("Is the user active (1) or not (0): ")
    active = int(active)
    while active != 0 and active != 1:
        print("Invalid input, choose 1 if active or 0 if not active")
        active = input("Is the user active (1) or not (0): ")
        active = int(active)
    user = User.User(ID, name, DoB, role, active, "", 0)
    users.append(user)
    print("The user has been added successfully")

def updateUser():
    ID = input("Input the ID of the user: ")

```

```

ID = int(ID)
while not isUserExist(ID):
    print("No user with this ID, try again...")
    ID = input("Input the ID of the user: ")
    ID = int(ID)
for i in users:
    if i.ID == ID:
        print("Please Choose What you want to Update For The User: ")
        print("1. User Name ")
        print("2. User Date Of Birthday ")
        print("3. User Role ")
        print("4. User Active")
        choice = input("Your choice: ")
        choice = int (choice)
        while(choice < 1 or choice > 4):
            print("Invalid input, choose from 1 tp 4")
            print("Please Choose What you want to Update For The User: ")
            print("1. User Name ")
            print("2. User Date Of Birthday ")
            print("3. User Role ")
            print("4. User Active")
        if choice == 1:
            name = input("Input the user name: ")
            i.name = name
        elif choice == 2:
            DoB = input("Input the user's date of birth (d/m/y): ")
            data = DoB.split('/')
            while len(data) != 3:
                print("Invalid input, enter the date as this formate (d/m/y)")
                DoB = input("Input the user's date of birth (d/m/y): ")
                data = DoB.split('/')
            i.DoB = date(int(data[2]),int(data[1]),int(data[0]))
        elif choice == 3:
            role = input("Input the role of the user (shopper/admin): ")
            role = role.lower()
            while (role != "shopper" and role != "admin"):
                print("Invalid input")
                role = input("Input the role of the user (shopper/admin): ")
            i.role = role
        elif choice == 4:
            active = input("Is the user active (1) or not (0): ")
            active = int(active)
            while active != 0 or active != 1:
                print("Invalid input...")
                active = input("Is the user active (1) or not (0): ")
                active = int(active)
            i.active = active
        print("\nThe user has been updated successfully...\n")

def DissplayAllUsers():
    print("\nUsers in the system.....\n\n")
    for i in users:
        print(str(i.__str__()))
        print("-----")

def DisplayProducts():

```

```

while 1:
    print("\nPlease Choose What you want to List From Products: ")
    print("1. List All Products ")
    print("2. List Offers Products ")
    print("3. List Category Products ")
    print("4. List Name Products")
    choice = input("Your choice: ")
    choice = int(choice)
    if choice >= 1 and choice <= 4:
        break
print("\n\n")
if choice == 1:
    for i in products:
        print(i.__str__())
        print("\n\n")
elif choice == 2:
    for i in products:
        if i.Has_an_offer == 1:
            print(i.__str__())
            print("\n\n")
elif choice == 3:
    category = input("Input the category: ")
    for i in products:
        if i.category == category:
            print(i.__str__())
            print("\n\n")
elif choice == 4:
    name = input("Input the name of the product: ")
    for i in products:
        if i.name == name:
            print(i.__str__())
            print("\n\n")

def addProductToTheBasket(user_id):
    product_id = input("Input the ID of the product to be added: ")
    while 1:
        try:
            product_id = int(product_id)
            break
        except ValueError:
            print("Invalid input...")
            product_id = input("Input the ID of the product to be added:")
    while not isProductExist(product_id):
        print("No product with this ID, try again")
        product_id = input("Input the ID of the product to be added: ")
    while 1:
        try:
            product_id = int(product_id)
            break
        except ValueError:
            print("Invalid input...")
            product_id = input("Input the ID of the product to be added:")
    while 1:
        numOfItems = input("Input number of items from this product to be added: ")
        try:
            numOfItems = int(numOfItems)

```

```

        break
    except ValueError:
        print("Invalid input...")
for i in users:
    if i.ID == user_id:
        i.addItemToBasket(product_id,numOfItems)
        print("The product had been added to the basket successfully...")
        break

def displayBasket(user_id):
    sum = 0
    for i in users:
        if i.ID == user_id:
            if len(i.basket) == 0:
                print('Your basket is empty...')
                return
            for j in i.basket:
                print("\n\nProduct ID: "+str(j))
                for k in products:
                    if k.ID == int(j):
                        print("Product name: "+str(k.name))
                        print("Product category: "+str(k.category))
                        print("Price: "+str(k.price))
                        print("Supplier: "+str(k.supplier))
                        print("Has an offer: "+str(k.Has_an_offer))
                        if k.Has_an_offer == 1:
                            print("offer price: "+str(k.offer_price))
                            print("valid until: "+k.valid_until)
                            print("Number of Items: " + str(i.basket[j]))
                            cost = int(int(i.basket[j]) * k.offer_price)
                            print("Cost of purchase of the product: " + str(cost))
                            sum += cost
                        else:
                            print("Number of Items: "+str(i.basket[j]))
                            print("offer price:valid until = 0/0/0")
                            cost = int (int(i.basket[j]) * k.price)
                            print("Cost of purchase of the product: "+str(cost))
                            sum += cost
                print("\nBasket cost: "+str(sum))

def updateBasket(user_id):
    while 1:
        print("\nPlease choose one of the following options (1-3):")
        print("1- Clear the basket")
        print("2- Remove a specific product from the basket")
        print("3- Update the number of items for a specific product")
        choice = input("Your choice: ")
        try:
            choice = int(choice)
        except ValueError:
            print("Invalid input..")
            continue
        if choice >= 1 and choice <= 3:
            break
        else:
            print("Invalid input..")

```



```

for i in users:
    if i.ID == user_id:
        if choice == 1:
            i.clearBasket()
            print("\nThe basket has been cleared successfully...")
        elif choice == 2:
            product_id = input("Input the product ID to be removed from the basket: ")
            while 1:
                try:
                    product_id = int (product_id)
                    break
                except ValueError:
                    print("Invalid input...")
                    product_id = input("Input the product ID to be removed from the
basket: ")

            found = 0
            for j in i.basket:
                if j == product_id:
                    found = 1
                    i.removeProduct(product_id)
                    print("Product has been removed from the basket successfully...")
            if found == 0:
                print("This product does not exist in your basket...")
                break
        elif choice == 3:
            product_id = input("Input the product ID to be updated from the basket: ")
            while 1:
                try:
                    product_id = int(product_id)
                    break
                except ValueError:
                    print("Invalid input...")
                    product_id = input("Input the product ID to be removed from the
basket: ")

            found = 0
            for j in i.basket:
                if j == product_id:
                    found = 1
                    numOfItems = input("Input the new number of items for this product:
")

                    numOfItems = int(numOfItems)
                    i.updateBasket(product_id, numOfItems)
                    print("The basket has been updated successfully...")
            if found == 0:
                print("This product does not exist in your basket...")
                break

def placeAnOrder(user_id):
    for i in users:
        if i.ID == user_id:
            i.placeAnOrder()

def executeOrder():
    for i in users:

```

```

        if i.order == 1:
            for j in i.basket:
                items = i.basket[j]
                for k in products:
                    if(k.ID == j):
                        k.executeOrder(items)
            i.clearBasket()
            i.order = 0
            print("User: "+i.name+", ID: "+str(i.ID)+" order has been executed successfully...")

def saveProducts():
    filename = input("Input the file name to save the products information in: ")
    f = open(filename, "w")
    for i in products:
        f.write(str(i.ID)+";")
        f.write(str(i.name)+";")
        f.write(str(i.category)+";")
        f.write(str(i.price) + ";")
        f.write(str(i.inventory) + ";")
        f.write(str(i.supplier) + ";")
        f.write(str(i.Has_an_offer) + ";")
        if i.Has_an_offer == 1:
            f.write(str(i.Has_an_offer) + ";")
            f.write(str(i.offer_price) + ";")
            f.write(str(i.valid_until))
        else:
            f.write(str(i.Has_an_offer))
        f.write("\n")

def saveUsers():
    filename = input("Input the file name to save the users information in: ")
    f = open(filename, "w")
    for i in users:
        f.write(str(i.ID) + ";")
        f.write(str(i.name) + ";")
        f.write(str(i.DoB) + ";")
        f.write(str(i.role) + ";")
        f.write(str(i.active) + ";")
        f.write(str(i.basket) + ";")
        f.write(str(i.order))
        f.write("\n")

def listShoppers():
    while 1:
        print("\nChoose one of the following options (1-3): ")
        print("1- List all shoppers")
        print("2- List all shoppers who have added items for their basket")
        print("3- List all shoppers who have unprocessed order")
        choice = input("Your choice: ")
        try:
            choice = int(choice)
        except ValueError:
            print("Invalid input...")
            continue

```

```

        if choice >= 1 and choice <= 3:
            break
        else:
            print("Invalid input...")

for i in users:
    if choice == 1:
        if i.role == "shopper":
            print("User: "+i.name+", ID: "+str(i.ID))
    elif choice == 2:
        if(len(i.basket) != 0):
            print("User: "+i.name+", ID: "+str(i.ID))
    elif choice == 3:
        if i.order == 1:
            print("User: " + i.name + ", ID: " + str(i.ID))

#log in process here
print("Welcom to our E-commerce System\n")
print("Log in the system.....")
role = -1
ID = 0
while 1:
    found = 0
    ID = input("Your ID: ")
    try:
        ID = int(ID)
    except ValueError:
        print("Invalid input...")
        continue
    for i in users:
        if i.ID == ID:
            found = 1
            if i.role == "shopper":
                role = 0
                print("Welcome our dear shopper")
                break
            elif i.role == "admin":
                role = 1
                print("Welcome our dear admin")
                break
    if found == 0:
        print("This ID does not exist in the system, try again....\n")
    if role != -1:
        break
dirty = 0
while 1:

    print("\n-----\n")
    print("Please choose one of the following optiona:")
    print ("1.  Add product ")
    print ("2.  Place an item on sale ")
    print ("3.  Update product ")
    print ("4.  Add a new user")
    print ("5.  Update user")
    print ("6.  Display all users ")

```

```

print ("7. List products ")
print ("8. List shoppers ")
print ("9. Add product to the basket")
print ("10. Display basket")
print ("11. Update basket")
print ("12. Place order ")
print ("13. Execute order ")
print ("14. Save products to a file ")
print ("15. Save users to a text file ")
print ("16. Exit")
choice = input("Your choice: ")
try:
    choice = int(choice)
except ValueError:
    print("Invalid input...")
    continue

if choice == 1:
    if role == 1:
        dirty = 1
        addProduct()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 2:
    if role == 1:
        dirty = 1
        placeItemOnSale()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 3:
    if role == 1:
        dirty = 1
        updateProduct()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 4:
    if role == 1:
        dirty = 1
        addUser()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 5:
    if role == 1:
        dirty = 1
        updateUser()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 6:
    if role == 1:
        DissplayAllUsers()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 7:
    DisplayProducts()
elif choice == 8:
    if role == 1:

```

```

        listShoppers()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 9:
    if role == 0:
        dirty = 1
        addProductToTheBasket(ID)
    else:
        print("As an admin, you don't have the permission to do this process")
elif choice == 10:
    if role == 0:
        displayBasket(ID)
    else:
        print("As an admin, you don't have the permission to do this process")
elif choice == 11:
    if role == 0:
        dirty = 1
        updateBasket(ID)
    else:
        print("As an admin, you don't have the permission to do this process")
elif choice == 12:
    if role == 0:
        dirty = 1
        placeAnOrder(ID)
    else:
        print("As an admin, you don't have the permission to do this process")
elif choice == 13:
    if role == 1:
        dirty = 1
        executeOrder()
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 14:
    if role == 1:
        saveProducts()
        dirty = 0
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 15:
    if role == 1:
        saveUsers()
        dirty = 0
    else:
        print("As a shopper, you don't have the permission to do this process")
elif choice == 16:
    if dirty == 1:
        while 1:
            answer = input("Are you sure you want to exit without saving? (yes/no)")
            answer.lower()
            if answer == "no":
                break
            elif answer == "yes":
                print("\nlogging out has been done successfully")
                print("See you later...")
                exit(1)
        else:

```

```
        print("Invalid input...")
    else:
        print("\nlogging out has been done successfully")
        print("See you later...")
        exit(1)
else:
    print("Invalid input...")
    print("-----")
```