BERZIET UNIVERSITY

Faculty of Engineering & Technology – Electrical & Computer Engineering Department

Summer Semester 2022

LINUX LABORATORY ENCS3130

Project2 Report (Python Project)

Prepared by

Mahmoud Samara ID: 1191602

Mohammad AbuJaber ID: 1190298

Instructors

Dr. Aziz Qaroush

Dr. Mohammad Jubran

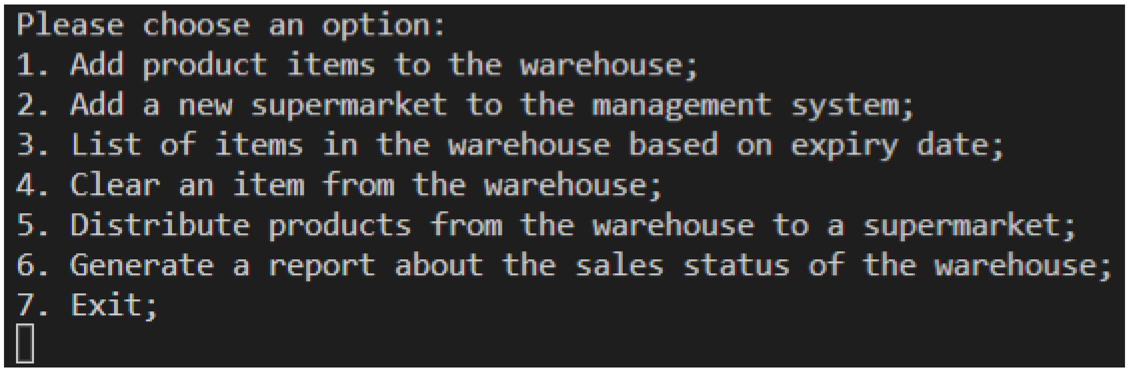
Date: 30th August 2022

Table of Contents

Table of Figures

1. Discussion The Project Code

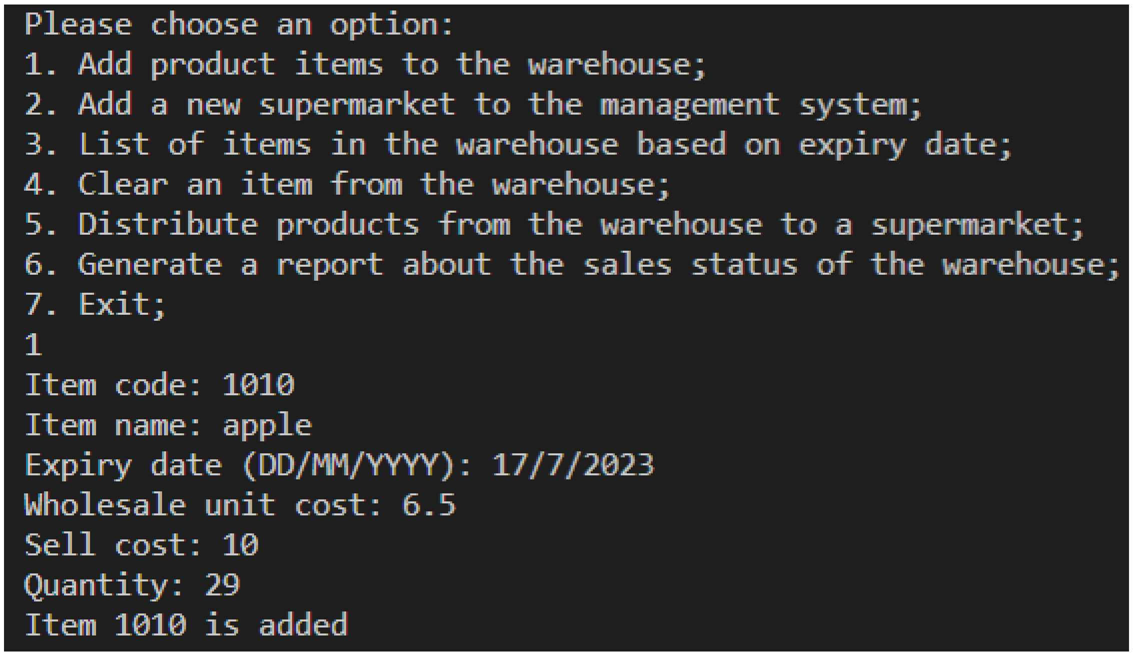
1.1. The Main Menu



```
Please choose an option:
1. Add product items to the warehouse;
2. Add a new supermarket to the management system;
3. List of items in the warehouse based on expiry date;
4. Clear an item from the warehouse;
5. Distribute products from the warehouse to a supermarket;
6. Generate a report about the sales status of the warehouse;
7. Exit;
```

Figure 1: Main Menu

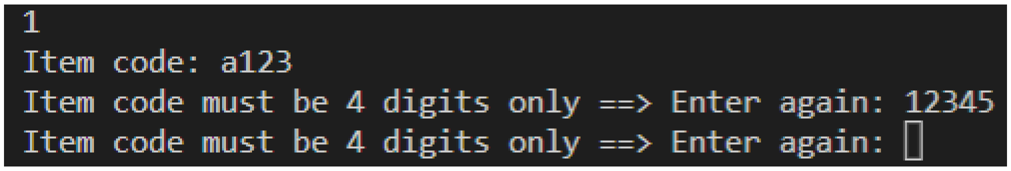
1.2. Add Items (option 1)



```
Please choose an option:
1. Add product items to the warehouse;
2. Add a new supermarket to the management system;
3. List of items in the warehouse based on expiry date;
4. Clear an item from the warehouse;
5. Distribute products from the warehouse to a supermarket;
6. Generate a report about the sales status of the warehouse;
7. Exit;
1
Item code: 1010
Item name: apple
Expiry date (DD/MM/YYYY): 17/7/2023
Wholesale unit cost: 6.5
Sell cost: 10
Quantity: 29
Item 1010 is added
```

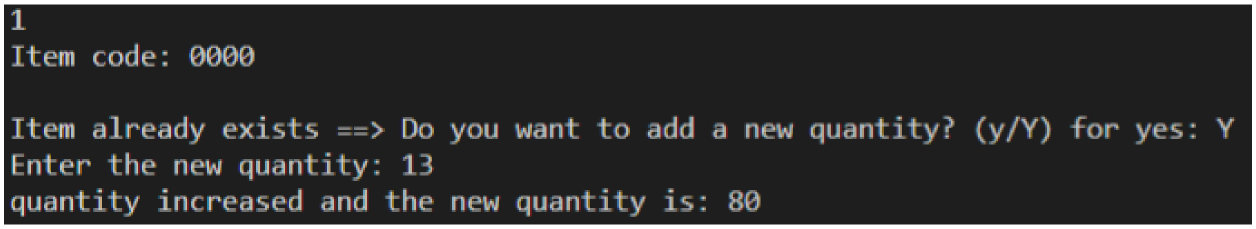
Figure 2: Add Item

In this option, the program will ask the user to add an item and enter specific information about that item. Since the user will enter the information, then there are some exceptions. First of all, no two items will have the same code, so if a user enters two items with the same code, an error message will appear. Moreover, the user can not add an exp date in the past. Finally, if the user adds a new item with the specific information for the previous item, the quantity value will only change, and other things will remain the same. Also, the whole cost, selling cost, and quantity must be only numbers.



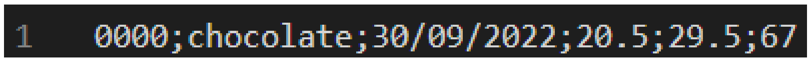
```
1
Item code: a123
Item code must be 4 digits only ==> Enter again: 12345
Item code must be 4 digits only ==> Enter again: 
```

Figure 3: Checking Four Digits



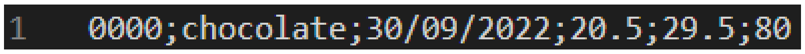
```
1
Item code: 0000

Item already exists ==> Do you want to add a new quantity? (y/Y) for yes: Y
Enter the new quantity: 13
quantity increased and the new quantity is: 80
```

Figure 4: Updating the Quantity



```
0000;chocolate;30/09/2022;20.5;29.5;67
```

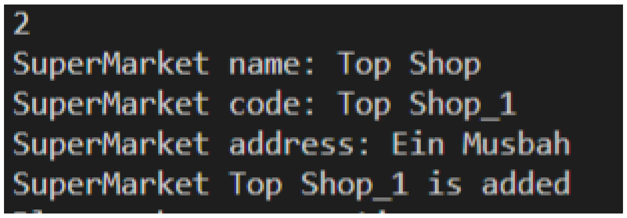
Figure 5: Data Before Updating the Quantity



```
0000;chocolate;30/09/2022;20.5;29.5;80
```

Figure 6: Data After Updating the Quantity

1.3. Add Supermarket (option 2)



```
2
SuperMarket name: Top Shop
SuperMarket code: Top Shop_1
SuperMarket address: Ein Musbah
SuperMarket Top Shop_1 is added
```

Figure 7: Add Supermarket

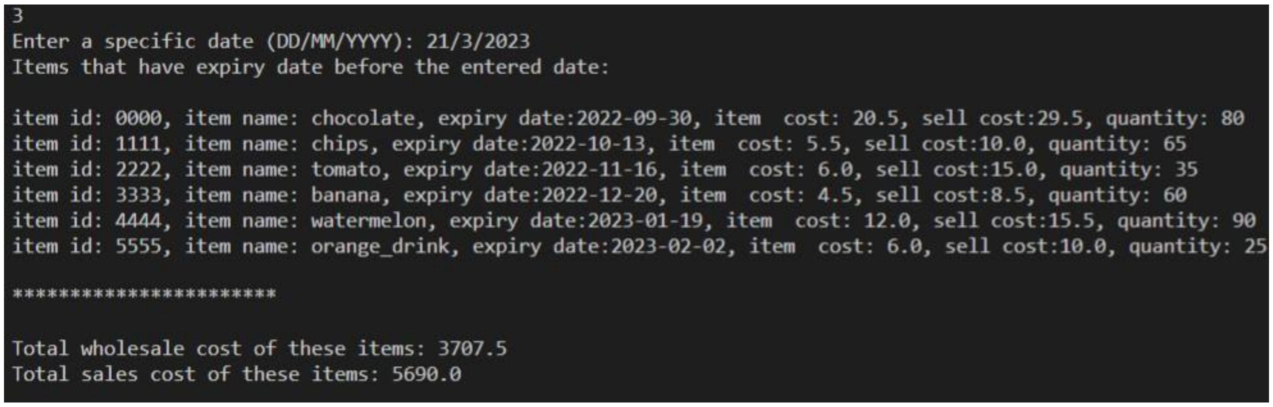
1.4. List of items in the warehouse based on expiry date

In the third option, the program will ask the user to enter a specific date. This date will act as the required exp date, so the program must print all items with information that have an expiry date before the input date. In addition, for the printed items that we find before a specific date, we will print the total wholesale cost of these items and the total sales cost of these items.

The code we used for the calculation was shown below:

```
    def list_items(self, date):
        """list all the items that have an expiry date before an input date
'date'"""
        print("Items that have expiry date before the entered date:\n")
        totalcost = 0
        totalsell = 0
        for item in self.items:
            """check if the item is expired"""
            if item.item_exp_date < date:
                """print the item details"""
                print("item id: " + item.item_code + ", item name: " +
item.item_name + ", expiry date:" + str(item.item_exp_date)
                      + ", item  cost: " + str(item.purchasing_cost) + ", sell
cost:" + str(item.sell_cost) + ", quantity: " + str(item.quantity))
                """calculate the total cost and total sell"""
                totalcost = totalcost + (item.purchasing_cost * item.quantity)
                totalsell = totalsell + (item.sell_cost * item.quantity)

        if totalcost == 0 and totalsell == 0:
            print("No items found!\n")
        else:
            print("\n***********************")
            print("\nTotal wholesale cost of these items: " + str(totalcost))
            print("Total sales cost of these items: "+str(totalsell) + "\n")
```

The output:

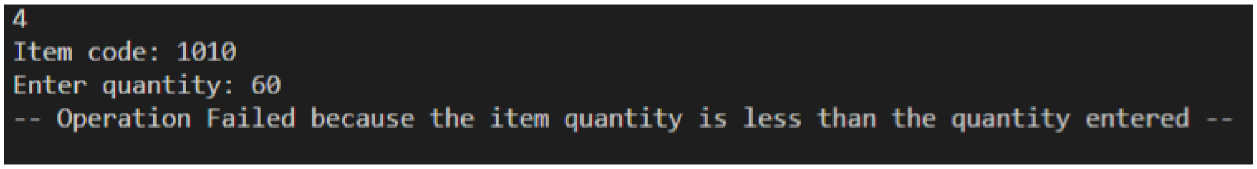


```
3
Enter a specific date (DD/MM/YYYY): 21/3/2023
Items that have expiry date before the entered date:

item id: 0000, item name: chocolate, expiry date:2022-09-30, item  cost: 20.5, sell cost:29.5, quantity: 80
item id: 1111, item name: chips, expiry date:2022-10-13, item  cost: 5.5, sell cost:10.0, quantity: 65
item id: 2222, item name: tomato, expiry date:2022-11-16, item  cost: 6.0, sell cost:15.0, quantity: 35
item id: 3333, item name: banana, expiry date:2022-12-20, item  cost: 4.5, sell cost:8.5, quantity: 60
item id: 4444, item name: watermelon, expiry date:2023-01-19, item  cost: 12.0, sell cost:15.5, quantity: 90
item id: 5555, item name: orange_drink, expiry date:2023-02-02, item  cost: 6.0, sell cost:10.0, quantity: 25

***********************

Total wholesale cost of these items: 3707.5
Total sales cost of these items: 5690.0
```

Figure 8: List of items in the warehouse based on expiry date

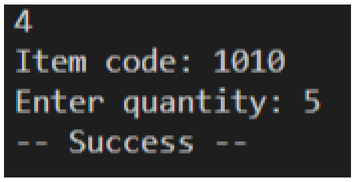
1.5. Clear an item from the warehouse

In the fourth part, the user will enter a specific code for a chosen item, then the program will ask the user to input the quantity that needs to be cleared (which should be at most the available quantity). Then the software will clear the item and print a confirmation message. Note that, if the item does not exist, the software should print an error message on the screen.



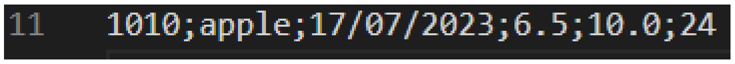
```
4
Item code: 1010
Enter quantity: 60
-- Operation Failed because the item quantity is less than the quantity entered --
```

Figure 9: Clear More Than the Exist Quantity



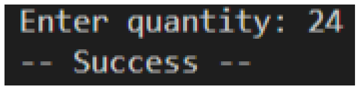
```
4
Item code: 1010
Enter quantity: 5
-- Success --
```

Figure 10: Clearing Less Than Exist Quantity



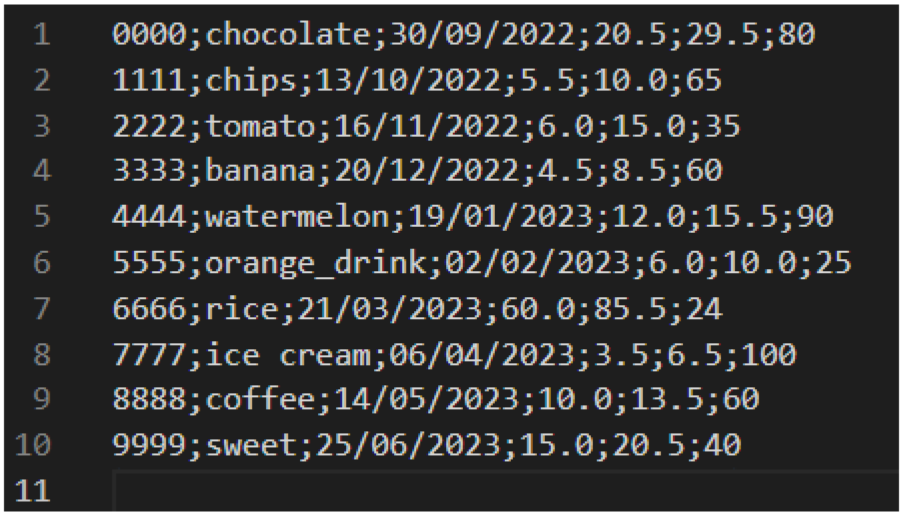
```
1010;apple;17/07/2023;6.5;10.0;24
```

Figure 11: File Data



```
Enter quantity: 24
-- Success --
```

Figure 12: Clearing All the Quantity



```
0000;chocolate;30/09/2022;20.5;29.5;80
1111;chips;13/10/2022;5.5;10.0;65
2222;tomato;16/11/2022;6.0;15.0;35
3333;banana;20/12/2022;4.5;8.5;60
4444;watermelon;19/01/2023;12.0;15.5;90
5555;orange_drink;02/02/2023;6.0;10.0;25
6666;rice;21/03/2023;60.0;85.5;24
7777;ice cream;06/04/2023;3.5;6.5;100
8888;coffee;14/05/2023;10.0;13.5;60
9999;sweet;25/06/2023;15.0;20.5;40

```

Figure 13: Data File

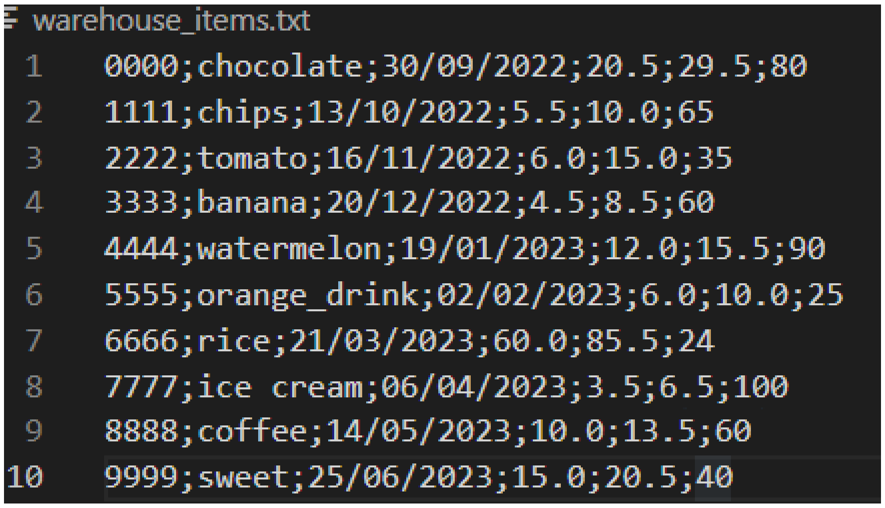
The used code:

```python
def clear_item(managementSystem: ManagementSystem):
    """reading the item code and quantity to remove it by the system"""
    code = str(input("Item code: "))
    while True:
        if not fourDigits(code):
            code = str(
                input("Item code must be 4 digits only ==> Enter again: "))
        else:
            break

    if managementSystem.search_for_item(code):
        quantity = int(input("Enter quantity: "))
        managementSystem.remove_from_items(code, quantity)
    else:
        print("Item does not exist\n")
```

```python
def remove_from_items(self, code, quantity):
        """to remove items from the warehouse"""
        for item in self.items:
            if item.item_code == code:
                if item.quantity == quantity:
                    self.items.remove(item)
                    print("-- Success --\n")
                elif item.quantity > quantity:
                    item.quantity = item.quantity - quantity
                    print("-- Success --\n")
                elif item.quantity < quantity:
                    print(
                        "-- Operation Failed because the item quantity is less
than the quantity entered -- \n")
```

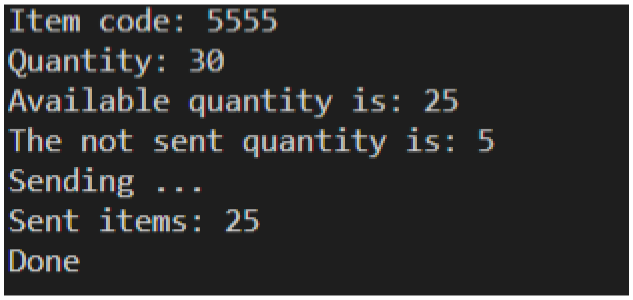
1.6. Distribute products from the warehouse to a supermarket

In the fifth part we will Distribute products from the warehouse to a supermarket, so in this part at first our code will make input files for all supermarkets in the system without adding any thing, then The software, will then check the warehouse and distribute the requested quantities of each item. It will also add these items to the list of items available at the supermarket and remove them from the warehouse. We must note that if the requested quantity of any item is not enough, the software will distribute only the available quantity. It will also print on the screen a message about the item and the number of requested but not distributed quantities of this item. For example, if in the warehouse we have a specific item 10 and a supermarket request 15 , then the software writes that the item sent is 10 and the item not sent is 5. If the item is not available from the supermarket, an error message will appear, and the requested code and the requested amount will be printed for the unavailable item.



```
warehouse_items.txt
0000;chocolate;30/09/2022;20.5;29.5;80
1111;chips;13/10/2022;5.5;10.0;65
2222;tomato;16/11/2022;6.0;15.0;35
3333;banana;20/12/2022;4.5;8.5;60
4444;watermelon;19/01/2023;12.0;15.5;90
5555;orange_drink;02/02/2023;6.0;10.0;25
6666;rice;21/03/2023;60.0;85.5;24
7777;ice cream;06/04/2023;3.5;6.5;100
8888;coffee;14/05/2023;10.0;13.5;60
9999;sweet;25/06/2023;15.0;20.5;40
```

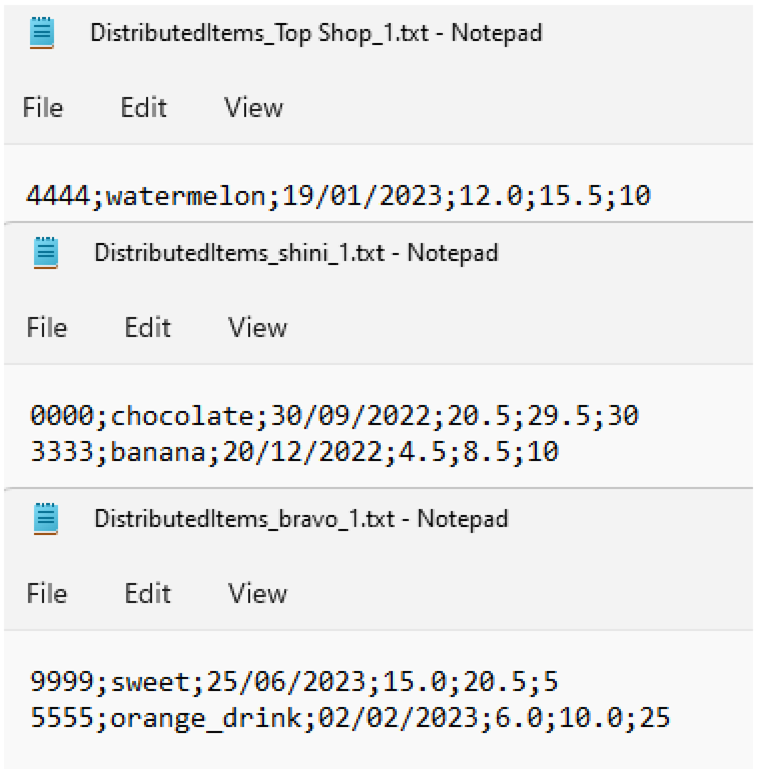
Figure 14: Data Before Distribution



```
Item code: 5555
Quantity: 30
Available quantity is: 25
The not sent quantity is: 5
Sending ...
Sent items: 25
Done
```

Figure 15: Requested Quantity of Any Item is not Enough

Also, as we can see item code 5555, is deleted since all the quantity of it is delivered


DistributedItems_Top Shop_1.txt - Notepad
File Edit View

```
4444;watermelon;19/01/2023;12.0;15.5;10
```

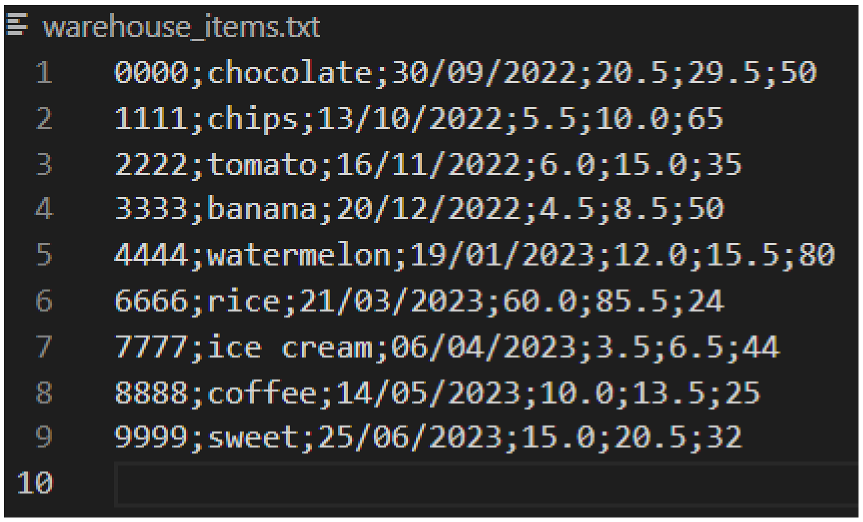
DistributedItems_shini_1.txt - Notepad
File Edit View

```
0000;chocolate;30/09/2022;20.5;29.5;30
3333;banana;20/12/2022;4.5;8.5;10
```

DistributedItems_bravo_1.txt - Notepad
File Edit View

```
9999;sweet;25/06/2023;15.0;20.5;5
5555;orange_drink;02/02/2023;6.0;10.0;25
```

Figure 16: Some of Distributed Items Files



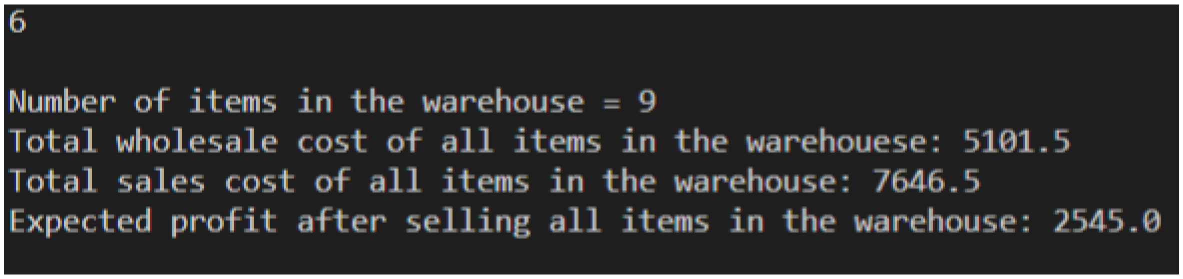
```
0000;chocolate;30/09/2022;20.5;29.5;50
1111;chips;13/10/2022;5.5;10.0;65
2222;tomato;16/11/2022;6.0;15.0;35
3333;banana;20/12/2022;4.5;8.5;50
4444;watermelon;19/01/2023;12.0;15.5;80
6666;rice;21/03/2023;60.0;85.5;24
7777;ice cream;06/04/2023;3.5;6.5;44
8888;coffee;14/05/2023;10.0;13.5;25
9999;sweet;25/06/2023;15.0;20.5;32
```

Figure 17: Data After Distribution

The code:

```
def distribute_product(managementSystem: ManagementSystem):
    """distribute a product to a supermarket"""
    option = 'y'
    market_code = str(input("\nSuperMarket code: "))
    if managementSystem.search_for_supermarket(market_code):
        while option == 'y' or option == 'Y':
            item_code = str(input("Item code: "))
            if managementSystem.search_for_item(item_code):
                quantity = int(input("Quantity: "))
                if quantity < 1:
                    print("Invalid input\n")
                    break
                if managementSystem.available_item(item_code) >= quantity:
                    print("Needed quantity is sent now")
                    managementSystem.send_to_supermarket(
                        item_code, market_code, quantity)
                elif managementSystem.available_item(item_code) > 0:
                    print("Available quantity is: " +
                          str(managementSystem.available_item(item_code)))
                    print("The not sent quantity is: " + str(quantity -
                          managementSystem.available_item(item_code)))
                    print("Sending ...")
                    managementSystem.send_to_supermarket(
                        item_code, market_code,
managementSystem.available_item(item_code))
                else:
                    print("This item is not available right now...")
            else:
                quantity = int(input("Quantity: "))
                if not fourDigits(item_code):
                    print("Item code must be 4 digits only")
                else:
                    print("***********\nItem does not exist")
                print("The code of the item is: " + str(item_code) + " and the
requested amount is: " + str(quantity))
            option = str(
                input("\nDo you want to send another item? y/Y for yes or type
anything to exit: "))
            item_code = ''
            quantity = 0
    else:
        print("this supermarket does not exist")
```

1.7. Generate a report about the sales status of the warehouse



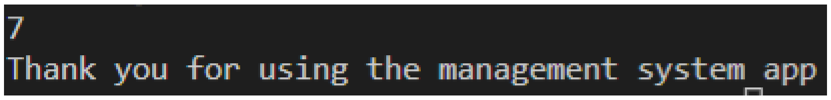
```
6

Number of items in the warehouse = 9
Total wholesale cost of all items in the warehouese: 5101.5
Total sales cost of all items in the warehouse: 7646.5
Expected profit after selling all items in the warehouse: 2545.0
```

Figure 18: Report

The code:

```
def generate_report(self):
        """Generate report"""
        wholesale = 0
        sales = 0
        num_of_items = 0
        for item in self.items:
            if item.quantity > 0:
                """calculate the total wholesale cost of the items"""
                wholesale = wholesale + (item.purchasing_cost * item.quantity)
                sales = sales + (item.sell_cost * item.quantity)
                num_of_items = num_of_items + 1
        print("\nNumber of items in the warehouse = " + str(num_of_items))
        print("Total wholesale cost of all items in the warehouese: " +
str(wholesale))
        print("Total sales cost of all items in the warehouse: " + str(sales))
        print("Expected profit after selling all items in the warehouse: " +
              str(sales - wholesale))
        print()
```

1.8. Exit



```
7
Thank you for using the management system app
```

Figure 19: Exit