C:\Users\janani\Desktop\RGU.PNG

**BSc (Hons) Artificial Intelligence and Data Science**

**Module: CM1601 Programming Fundamentals**

**Coursework 1 Report**

**Module Leader: Ms. Sachinthani Perera**

**RGU Student ID : 2313082**

**IIT Student ID : 20220578**

**Student Name : B.G.C.Gomes**

**Abstract**

This project is about an internet cafe, the things that happen there, and the things that are in the cafe. This system is specifically targeted at students who use internet cafes to do projects, assignments, and for enjoyment but do not have access to personal computers.

In this project, we must create an inventory system that allows users to add, delete, update, and view items. Also, as part of our project, we must define six dealers and interact with them as the user directs.

Six wholesale dealers, each of which sells high-quality items, have already provided the creator of this system with their contact information. Owing to the supplier's similarities, the creator has chosen to pick four dealers at random to buy the remaining items. John needs a way to control inventories because the cafe needs a lot of equipment.

Contents

[**Problem** 4](#_Toc132144657)

[**Problem Understanding** 4](#_Toc132144658)

[**Python Code (Source Code)** 5](#_Toc132144659)

[**Main.py** 5](#_Toc132144660)

[**Inventory\_dictionary.py** 10](#_Toc132144661)

[**Functions** 12](#_Toc132144662)

[**Introduction()** 12](#_Toc132144663)

[**Main()** 13](#_Toc132144664)

[**Load\_inventory() and save\_inventory()** 13](#_Toc132144665)

[**Add item()** 14](#_Toc132144666)

[**Delete\_item()** 16](#_Toc132144667)

[**Update\_item()** 16](#_Toc132144668)

[**View\_item()** 17](#_Toc132144669)

[**Test Cases** 18](#_Toc132144670)

[**Test Case 01** 22](#_Toc132144671)

[**Test case 02** 22](#_Toc132144672)

[**Test Case 03** 23](#_Toc132144673)

[**Test Case 04** 23](#_Toc132144674)

[**Test case 05** 24](#_Toc132144675)

[**Test case 06** 24](#_Toc132144676)

[**Test Case 07** 25](#_Toc132144677)

[**Test case 08** 25](#_Toc132144678)

[**Test Case 09** 26](#_Toc132144679)

[**Test Case 10** 27](#_Toc132144680)

[**Test case 11** 27](#_Toc132144681)

[**Text File** 28](#_Toc132144682)

[**Text Case 12** 28](#_Toc132144683)

[**Test case 13** 29](#_Toc132144684)

[**Test Case 14** 29](#_Toc132144685)

[**Test case 15** 30](#_Toc132144686)

[**Test case 16** 30](#_Toc132144687)

[**Test case 17** 31](#_Toc132144688)

[**Conclusion** 32](#_Toc132144689)

# **Problem**

Python command line applications should be used to develop this program. As described in the abstract, the project requires us to develop an inventory system that will benefit students without access to computers or the internet. For that, a system to manage the system's components, such as the number of computers, displays, keyboards, and so on, must be created. creator has received details of 6 dealers for buying items to the café. Creator is going to select four of those six dealers randomly. So we have to create a system to adding items, deleting items, updating items, viewing items and saving item details to a text file. Another task of this project is selecting four random dealers from the dealers. After selecting four dealers randomly we will be able to display the details of those four random dealers and system should display the particular item details when particular dealer name given.

# **Problem Understanding**

You must construct these functions as you develop this project.

You must initially display the inventory system's menu. For the process to continue, the system needs input.

When a user entered the "AID" option, the system instructs them to add a new item to their inventory. The system needs to obtain the following data before adding an item. such includes the item's code, name, brand, price, quantity, category, and date of purchase. The system can successfully add an item to the inventory by receiving this information.

A "DID" input should be received by the system when deleting an item. The owner should be able to erase item details using the system's item code search function.

System must receive "VID" as input in order to view an item. System should output the current total of the items that have previously been purchased and show item details in descending order while taking item code into account. All the information in this table needs to be correctly formatted.

The user must select "SID" as the input option if they wish to save these details to a text file. The owner should always have the option to save the item's details to the same text file using the system.

When "SDD" is selected, the system should simulate a drawing at random and choose four dealers from a text file in accordance. When the selection is complete, the message "4 Dealers are Picked Randomly" is displayed. The following information would be known to a dealer. Name, phone number, address, and products of the dealers (3 items for each dealer). Name, Brand, Price, and Quantity are necessary item data.

The input must be "VRL" if the user wishes to see all the information about the chosen dealers. This will cause the system to display all of the dealer information for the randomly chosen dealer. sorted according to the location.

When "LDI" is entered into the system, it ought to be able to display the products when one of the randomly chosen dealer names is entered.

This is how the project to control inventories works.

# **Python Code (Source Code)**

## **Main.py**

from tabulate import tabulate  
import json  
import random  
import inventory\_dictionary  
items\_dictionary = {}  
  
def inventory\_introduction():  
 from tabulate import tabulate  
  
 head=[["WELCOME TO ONE NET CAFE - INVENTORY SYSTEM" ] ]  
  
 print(tabulate(head**,**tablefmt="double\_grid"))  
  
 #table = tabulate(data, tablefmt="grid")  
  
def main():  
 #from colorama import Fore  
 print( """\033[3m  
 • Type AID for adding item details.  
 • Type DID for deleting item details.  
 • Type UID for updating item details.  
 • Type VID for viewing the items table.   
 • Type SID for saving the item details to the text file at any time.  
 • Type SDD for selecting four dealers randomly from a file.  
 • Type VRL for displaying all the details of the randomly selected dealers.   
 • Type LDI for display the items of the given dealer.  
 • Type ESC to exit the program.  
 \033[0m""")  
  
def inventory\_load():  
 global items\_dictionary  
 try:  
 with open("inventory.txt"**,** "r") as file:  
 items\_dictionary = json.load(file)  
 except FileNotFoundError:  
 pass  
  
def inventory\_save():  
 with open("inventory.txt"**,** "w") as f:  
 for i\_code**,** i\_details in items\_dictionary.items():  
 item\_record = f" 'Item code'= {i\_code}, 'Item name'={i\_details['Item Name']},'Item brand'={i\_details['Item Brand']}, 'Item price'={i\_details['Item Price']},'Quantity'={i\_details['Item Quantity']}, 'category'= {i\_details['Item Category']} 'Purchased Date'= {i\_details['Purchased Date']}\n"  
 f.write(item\_record)  
  
def adding\_item\_details():  
 i\_code = ""  
 while not i\_code:  
 i\_code = input("Enter Item Code: ")  
 i\_code=i\_code.strip()  
 if i\_code in items\_dictionary:  
 print(f"\033[1m\033[94m{i\_code} item already in inventory.\033[0m")  
 i\_code = ""  
 continue  
  
 i\_name = ""  
 while not i\_name:  
 i\_name = input("Enter Item Name: ")  
 i\_name=i\_name.strip()  
  
 i\_brand = ""  
 while not i\_brand:  
 i\_brand = input("Enter Item Brand: ")  
 i\_brand=i\_brand.strip()  
  
 i\_price = ""  
 while not i\_price:  
 try:  
 i\_price = float(input("Enter Item Price: "))  
 #i\_price=i\_price.strip()  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 continue  
  
 i\_quantity = ""  
 while not i\_quantity:  
 try:  
 i\_quantity = int(input("Enter Item Quantity: "))  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid integer value.\033[0m")  
 continue  
  
 i\_category = ""  
 while not i\_category:  
 i\_category = input("Enter Item Category: ")  
 i\_category=i\_category.strip()  
  
 item\_purchased\_date = ""  
 while not item\_purchased\_date:  
 item\_purchased\_date = input("Enter Purchased Date (DD/MM/YYYY): ")  
 item\_purchased\_date=item\_purchased\_date.strip()  
  
 i\_details = {"I Name": i\_name**,** "I Brand": i\_brand**,** "I Price": i\_price**,** "I Quantity": i\_quantity**,** "I Category": i\_category**,** "I Purchased Date": item\_purchased\_date}  
  
 items\_dictionary[i\_code] = i\_details  
 print(f"\033[1m\033[92m{i\_name} ({i\_code}) added to inventory.\033[0m")  
 inventory\_save()  
  
def deleting\_item():  
 i\_code = input("Enter Item Code: ")  
 if i\_code in items\_dictionary:  
 del items\_dictionary[i\_code]  
 print(f"\033[1m\033[93m{i\_code} deleted from inventory.\033[0m")  
 inventory\_save()  
 else:  
 print(f"\033[1m\033[31m{i\_code} not found in inventory.\033[0m")  
  
def updating\_item():  
 i\_code = input("Enter Item Code: ")  
 if i\_code in items\_dictionary:  
 print(f"\033[1m\033[36mNew details for the item (Dont Enter Anything If you need to keep the recent record.):\033[0m")  
 i\_name = input(f"Previous Item Name: {items\_dictionary[i\_code]['I Name']}\nNew Item Name: ")  
 i\_name=i\_name.strip()  
 i\_brand = input(f"Previous Item Brand: {items\_dictionary[i\_code]['I Brand']}\nNew Item Brand: ")  
 i\_brand=i\_brand.strip()  
 while True:  
 try:  
 i\_price = float(input(f"Previous Item Price: {items\_dictionary[i\_code]['I Price']}\nNew Item Price: "))  
 break  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 while True:  
 try:  
 i\_quantity = int(input(f"Previous Item Quantity: {items\_dictionary[i\_code]['I Quantity']}\nNew Item Quantity: "))  
 break  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 i\_category = input(f"Previous Item Category: {items\_dictionary[i\_code]['I Category']}\nNew Item Category: ")  
 i\_category=i\_category.strip()  
 item\_purchased\_date = input(f"Previous Purchased Date: {items\_dictionary[i\_code]['I Purchased Date']}\nNew Purchased Date (DD/MM/YYYY): ")  
 item\_purchased\_date=item\_purchased\_date.strip()  
  
 if i\_name:  
 items\_dictionary[i\_code]['I Name'] = i\_name  
 if i\_brand:  
 items\_dictionary[i\_code]['I Brand'] = i\_brand  
 if i\_price:  
 items\_dictionary[i\_code]['I Price'] = float(i\_price)  
 if i\_quantity:  
 items\_dictionary[i\_code]['I Quantity'] = int(i\_quantity)  
 if i\_category:  
 items\_dictionary[i\_code]['I Category'] = i\_category  
 if item\_purchased\_date:  
 items\_dictionary[i\_code]['I Purchased Date'] = item\_purchased\_date  
 inventory\_save()  
 print(f"\033[1m\033[92m {i\_code} ({i\_name}) details updated.\033[0m")  
 #print(f"{i\_code}{i\_name} details updated.")  
 else:  
 print(f"\033[1m\033[31m{i\_code} not found in inventory.\033[0m") #red  
  
def viewing\_items():  
 if not items\_dictionary:  
 # print("No items found!")  
 print("\033[1m\033[31mNo items found!\033[0m")  
 else:  
 items = []  
 total\_purchased\_items = **0** for i\_code**,** item\_details in items\_dictionary.items():  
 items.append([i\_code**,** item\_details["I Name"]**,** item\_details["I Brand"]**,** item\_details["I Price"]**,** item\_details["I Quantity"]**,** item\_details["I Category"]**,** item\_details["I Purchased Date"]])  
 total\_purchased\_items += item\_details["Item Quantity"]  
  
 items = sorted(items**,** key=lambda x: x[**0**]**,** reverse=True)  
  
 headers = ["\033[1mItem code\033[0m"**,** "\033[1mItem Name\033[0m"**,** "\033[1mItem Brand\033[0m"**,**"\033[1mItem Price\033[0m"**,** "\033[1mItem Quantity\033[0m"**,** "\033[1mItem Category\033[0m"**,**"\033[1mPurchased Date\033[0m"]  
 print(tabulate(items**,** headers=headers**,** tablefmt="heavy\_grid"))  
  
 print("\033[1m\033[93mTotal Purchased Items: {}\033[0m".format(total\_purchased\_items))  
  
while True:  
  
 inventory\_introduction()  
 main()  
  
 choice = input("Enter your choice: ")  
 choice = choice.upper()  
 if choice == 'AID':  
 adding\_item\_details()  
 print()  
  
 elif choice == 'DID':  
 deleting\_item()  
 elif choice == 'VID':  
 viewing\_items()  
 print()  
 elif choice == 'UID':  
 updating\_item()  
 print()  
  
  
 elif choice == 'SDD':  
 inventory\_dictionary.ddd()  
 try:  
  
 with open('dealers.txt'**,** 'r') as file:  
 dealers\_json = file.read()  
 dealers = json.loads(dealers\_json)  
  
 random\_dealers = random.sample(list(dealers.keys())**, 4**)  
 print("\033[1m\033[34m4 Dealers are selected Randomly\033[0m")  
 print()  
 except:  
 print("\033[1m\033[31mFile Not Found!\033[0m")  
  
  
 elif choice == 'VRL':  
  
 try:  
 for i in range(len(random\_dealers)):  
 for j in range(len(random\_dealers) - i - **1**):  
 if dealers[random\_dealers[j]]['Dealer\_Location'] > dealers[random\_dealers[j + **1**]]['Dealer\_Location']:  
 random\_dealers[j]**,** random\_dealers[j + **1**] = random\_dealers[j + **1**]**,** random\_dealers[j]  
  
 rows = []  
 for dealer in random\_dealers:  
 d\_row = [dealer**,** dealers[dealer]['Telephone\_Number']**,** dealers[dealer]['Dealer\_Location']]  
 rows.append(d\_row)  
 for item in dealers[dealer]['items']:  
 i\_row = [None**,** None**,** None**,** item['name']**,** item['brand']**,** item['price']**,** item['quantity']]  
 rows.append(i\_row)  
  
 headers = ['Dealer Name'**,** 'Contact Number'**,** 'Location'**,** 'Item Name'**,** 'Brand'**,**'price'**,** 'Quantity']  
 print(tabulate(rows**,** headers=headers))  
 except:  
 print("\033[1m\033[91mYou have not selected any dealers yet!!!\033[0m")  
  
 elif choice == 'LDI':  
 try:  
 dealer\_name = input("Enter Dealer Name ( Please select from the randomly selected dealer table ) : ")  
 dealer\_name=dealer\_name.strip()  
  
 if dealer\_name in random\_dealers:  
 print(f"\033[1m\033[36mDealer name: {dealer\_name}\033[0m ")  
  
  
 get\_dealer=[]  
  
 for item in dealers[dealer\_name]['items']:  
 i\_row = [item['name']**,** item['brand']**,** item['quantity']**,** item['price']]  
 get\_dealer.append(i\_row)  
 headers = ['\033[1mItem Name\033[0m'**,** '\033[1mBrand\033[0m'**,** '\033[1mQuantity\033[0m'**,**'\033[1mPrice\033[0m']  
 print(tabulate(get\_dealer**,** headers=headers**,** tablefmt='heavy\_grid'))  
  
 else:  
 #print("Dealer not found!")  
 print("\033[1m\033[31mDealer is not in the system (Please use randomly selected dealers...) \033[0m")  
 except:  
 print("\033[1m\033[31mThere are no any randomly selected dealers. Please try again the 'SDD'...\033[0m")  
  
 elif choice=='SID':  
 inventory\_save()  
 print(f"\033[1m\033[92mInventory saved.\033[0m")  
  
 elif choice == 'ESC':  
 print("Thankyou")  
 break  
  
 else:  
 print("\033[1m\033[31mInvalid choice.Try again.\033[0m")

## **Inventory\_dictionary.py**

* import json  
  dealers = {  
   'Gawesh Gomes': {  
   'Telephone\_Number': '+94761328236'**,** 'Dealer\_Location':'Colombo'**,** 'items': [  
   {  
   'name': 'dell XPS Laptop'**,** 'brand': 'DELL'**,** 'price': 'RS. 590000.00'**,** 'quantity': '50'  
   }**,** {  
   'name': 'Gaming keyboard'**,** 'brand': 'asus'**,** 'price': 'RS. 4900.00'**,** 'quantity': '30'  
   }**,** {  
   'name': 'CPU'**,** 'brand': 'DELL'**,** 'price': 'RS. 220000.00'**,** 'quantity': '20'  
   }  
   ]  
   }**,** 'Radil Damsara': {  
   'Telephone\_Number': '+94714325437'**,** 'Dealer\_Location':'Kaluthara'**,** 'items': [  
   {  
   'name': 'flash drive'**,** 'brand': 'HP'**,** 'price': 'Rs. 12000.00'**,** 'quantity': '40'  
   }**,** {  
   'name': 'Dell XPS 13'**,** 'brand': 'Dell'**,** 'price': 'Rs. 550000.00'**,** 'quantity': '15'  
   }**,** {  
   'name': 'heat fan pro'**,** 'brand': 'Dell'**,** 'price': 'Rs. 45000.00'**,** 'quantity': '5 '  
   }  
   ]  
   }**,** 'Devindi\_Perera': {  
   'Telephone\_Number': '+94754567242'**,** 'Dealer\_Location':'Kiribathgoda'**,** 'items': [  
   {  
   'name': 'Redmi note 9 pro'**,** 'brand': 'Redmi'**,** 'price': 'Rs. 70000.00'**,** 'quantity': '8'  
   }**,** {  
   'name': 'Gaming mouse'**,** 'brand': 'Asus'**,** 'price': 'Rs. 3900'**,** 'quantity': '16'  
   }**,** {  
   'name': 'monitor'**,** 'brand': 'HP'**,** 'price': 'Rs. 80000.00'**,** 'quantity': '12'  
   }  
   ]  
   }**,** 'Diyathma\_wijewardhana': {  
   'Telephone\_Number': '+94769765434'**,** 'Dealer\_Location':'Moratuwa'**,** 'items': [  
   {  
   'name': 'Iphone 14 pro max'**,** 'brand': 'Apple'**,** 'price': 'Rs. 649000.00'**,** 'quantity': '10'  
   }**,** {  
   'name': 'computer lamp'**,** 'brand': 'orange'**,** 'price': 'Rs. 4800.00'**,** 'quantity': '50'  
   }**,** {  
   'name': 'Iphone charger'**,** 'brand': 'apple'**,** 'price': 'Rs. 12000.00'**,** 'quantity': '40'  
   }  
   ]  
   }**,** 'Sehandu\_Siriwardhana': {  
   'Telephone\_Number': '+94786756453'**,** 'Dealer\_Location':'kurunagala'**,** 'items': [  
   {  
   'name': 'laptop bags'**,** 'brand': 'asus'**,** 'price': 'Rs. 2500.00'**,** 'quantity': '50'  
   }**,** {  
   'name': 'Headphone'**,** 'brand': 'samsung'**,** 'price': 'Rs. 3000.00'**,** 'quantity': '24'  
   }**,** {  
   'name': 'SSD'**,** 'brand': 'Sandisk'**,** 'price': 'Rs. 10000.00'**,** 'quantity': '35'  
   }  
   ]  
   }**,** 'Malindu\_Dilshan': {  
   'Telephone\_Number': '+94772534657'**,** 'Dealer\_Location':'Panadura'**,** 'items': [  
   {  
   'name': 'Memory card'**,** 'brand': 'Sandisk'**,** 'price': 'Rs. 4000.00'**,** 'quantity': '20'  
   }**,** {  
   'name': 'Memory card readers'**,** 'brand': 'Ugreen'**,** 'price': 'Rs. 1000.00'**,** 'quantity': '45'  
   }**,** {  
   'name': 'projector'**,** 'brand': 'ViewSonic'**,** 'price': 'Rs. 120000.00'**,** 'quantity': '18'  
   }  
   ]  
   }  
  }  
    
    
    
  # Convert dictionary to JSON string  
  def ddd():  
   dealers\_json = json.dumps(dealers)  
    
   # Write JSON string to file  
   with open('dealers.txt'**,** 'w') as file:  
   file.write(dealers\_json)

# **Functions**

## **Introduction()**

def inventory\_introduction():  
 from tabulate import tabulate  
  
 head=[["WELCOME TO ONE NET CAFE - INVENTORY SYSTEM" ] ]  
  
 print(tabulate(head**,**tablefmt="double\_grid"))  
  
 #table = tabulate(data, tablefmt="grid")

After running this function system will print the welcome message inside a table cell.

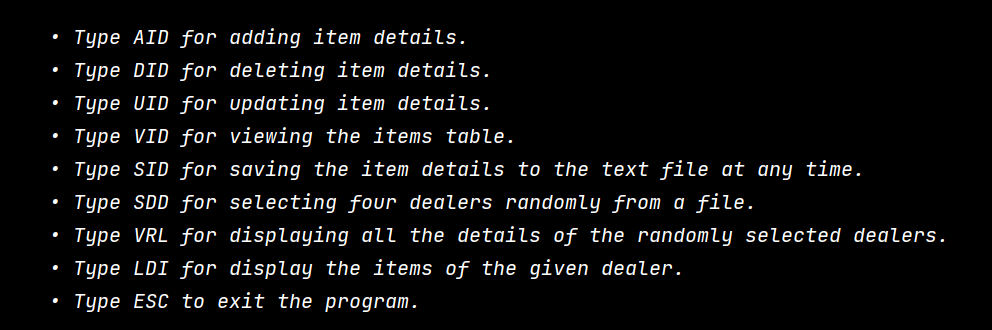
**Graphical user interface, text

Description automatically generated**

## **Main()**

def main():  
 #from colorama import Fore  
 print( """\033[3m  
 • Type AID for adding item details.  
 • Type DID for deleting item details.  
 • Type UID for updating item details.  
 • Type VID for viewing the items table.   
 • Type SID for saving the item details to the text file at any time.  
 • Type SDD for selecting four dealers randomly from a file.  
 • Type VRL for displaying all the details of the randomly selected dealers.   
 • Type LDI for display the items of the given dealer.  
 • Type ESC to exit the program.  
 \033[0m""")

After running this main function system display the main menu of the system.

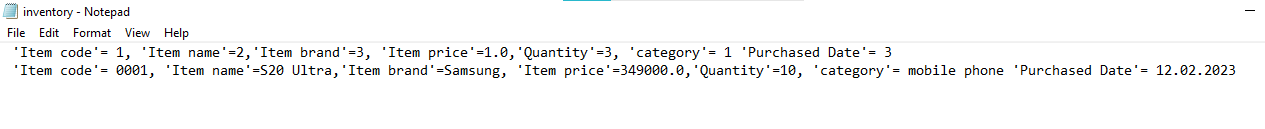


## **Load\_inventory() and save\_inventory()**

def inventory\_load():  
 global items\_dictionary  
 try:  
 with open("inventory.txt"**,** "r") as file:  
 items\_dictionary = json.load(file)  
 except FileNotFoundError:  
 pass  
  
def inventory\_save():  
 with open("inventory.txt"**,** "w") as f:  
 for i\_code**,** i\_details in items\_dictionary.items():  
 item\_record = f" 'Item code'= {i\_code}, 'Item name'={i\_details['Item Name']},'Item brand'={i\_details['Item Brand']}, 'Item price'={i\_details['Item Price']},'Quantity'={i\_details['Item Quantity']}, 'category'= {i\_details['Item Category']} 'Purchased Date'= {i\_details['Purchased Date']}\n"  
 f.write(item\_record)

After running this function system can store their item details and save them in a text file.

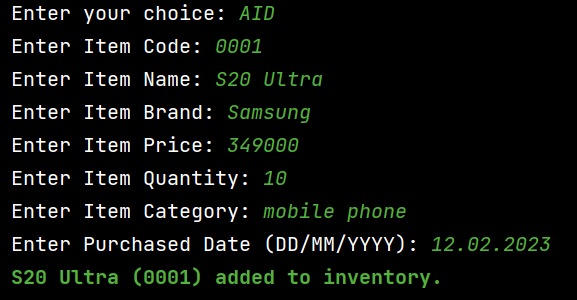




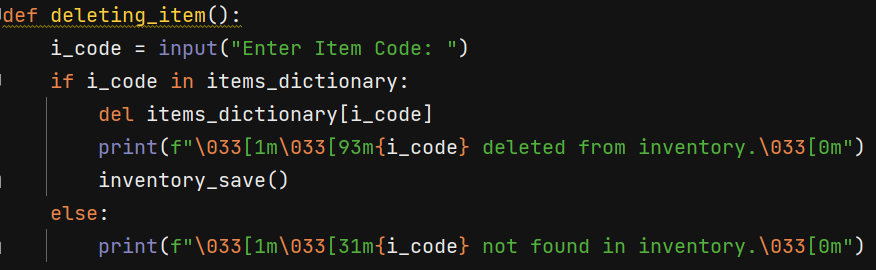
## **Add item()**

def adding\_item\_details():  
 i\_code = ""  
 while not i\_code:  
 i\_code = input("Enter Item Code: ")  
 i\_code=i\_code.strip()  
 if i\_code in items\_dictionary:  
 print(f"\033[1m\033[94m{i\_code} item already in inventory.\033[0m")  
 i\_code = ""  
 continue  
  
 i\_name = ""  
 while not i\_name:  
 i\_name = input("Enter Item Name: ")  
 i\_name=i\_name.strip()  
  
 i\_brand = ""  
 while not i\_brand:  
 i\_brand = input("Enter Item Brand: ")  
 i\_brand=i\_brand.strip()  
  
 i\_price = ""  
 while not i\_price:  
 try:  
 i\_price = float(input("Enter Item Price: "))  
 #i\_price=i\_price.strip()  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 continue  
  
 i\_quantity = ""  
 while not i\_quantity:  
 try:  
 i\_quantity = int(input("Enter Item Quantity: "))  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid integer value.\033[0m")  
 continue  
  
 i\_category = ""  
 while not i\_category:  
 i\_category = input("Enter Item Category: ")  
 i\_category=i\_category.strip()  
  
 item\_purchased\_date = ""  
 while not item\_purchased\_date:  
 item\_purchased\_date = input("Enter Purchased Date (DD/MM/YYYY): ")  
 item\_purchased\_date=item\_purchased\_date.strip()  
  
 i\_details = {"I Name": i\_name**,** "I Brand": i\_brand**,** "I Price": i\_price**,** "I Quantity": i\_quantity**,** "I Category": i\_category**,** "I Purchased Date": item\_purchased\_date}  
  
 items\_dictionary[i\_code] = i\_details  
 print(f"\033[1m\033[92m{i\_name} ({i\_code}) added to inventory.\033[0m")  
 inventory\_save()

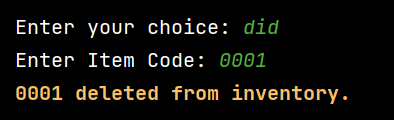
After running this function the system will be able to get details of an item.



## **Delete\_item()**



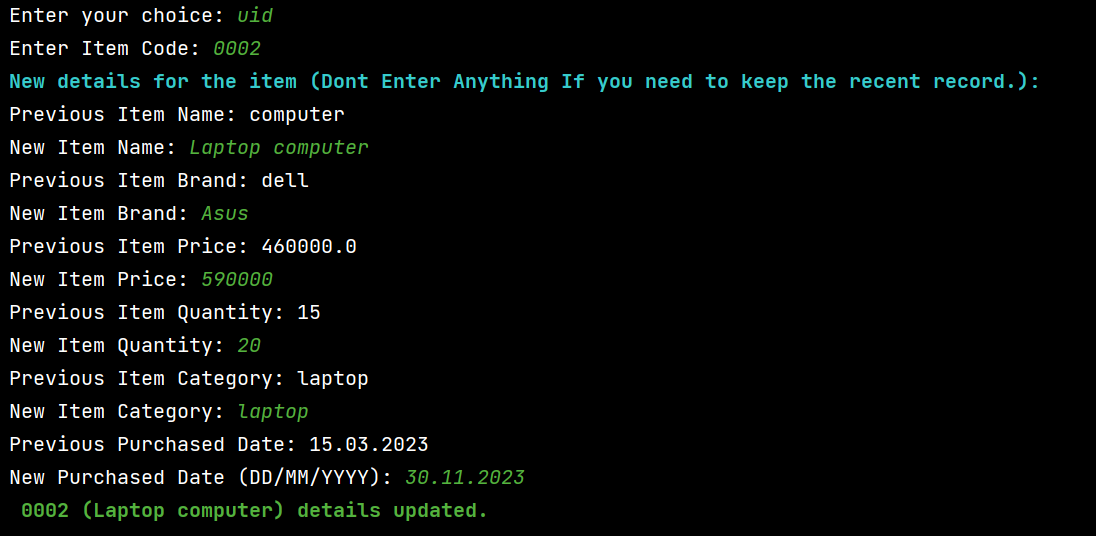
Items can be removed from the inventory system using the delete function. The system ought to remove all the information about that specific item from the inventory system once a user requests a specific item code.



## **Update\_item()**

def updating\_item():  
 i\_code = input("Enter Item Code: ")  
 if i\_code in items\_dictionary:  
 print(f"\033[1m\033[36mNew details for the item (Dont Enter Anything If you need to keep the recent record.):\033[0m")  
 i\_name = input(f"Previous Item Name: {items\_dictionary[i\_code]['I Name']}\nNew Item Name: ")  
 i\_name=i\_name.strip()  
 i\_brand = input(f"Previous Item Brand: {items\_dictionary[i\_code]['I Brand']}\nNew Item Brand: ")  
 i\_brand=i\_brand.strip()  
 while True:  
 try:  
 i\_price = float(input(f"Previous Item Price: {items\_dictionary[i\_code]['I Price']}\nNew Item Price: "))  
 break  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 while True:  
 try:  
 i\_quantity = int(input(f"Previous Item Quantity: {items\_dictionary[i\_code]['I Quantity']}\nNew Item Quantity: "))  
 break  
 except ValueError:  
 print(f"\033[1m\033[31mInvalid input. Please enter a valid number value.\033[0m")  
 i\_category = input(f"Previous Item Category: {items\_dictionary[i\_code]['I Category']}\nNew Item Category: ")  
 i\_category=i\_category.strip()  
 item\_purchased\_date = input(f"Previous Purchased Date: {items\_dictionary[i\_code]['I Purchased Date']}\nNew Purchased Date (DD/MM/YYYY): ")  
 item\_purchased\_date=item\_purchased\_date.strip()  
  
 if i\_name:  
 items\_dictionary[i\_code]['I Name'] = i\_name  
 if i\_brand:  
 items\_dictionary[i\_code]['I Brand'] = i\_brand  
 if i\_price:  
 items\_dictionary[i\_code]['I Price'] = float(i\_price)  
 if i\_quantity:  
 items\_dictionary[i\_code]['I Quantity'] = int(i\_quantity)  
 if i\_category:  
 items\_dictionary[i\_code]['I Category'] = i\_category  
 if item\_purchased\_date:  
 items\_dictionary[i\_code]['I Purchased Date'] = item\_purchased\_date  
 inventory\_save()  
 print(f"\033[1m\033[92m {i\_code} ({i\_name}) details updated.\033[0m")  
 #print(f"{i\_code}{i\_name} details updated.")  
 else:  
 print(f"\033[1m\033[31m{i\_code} not found in inventory.\033[0m") #red

After running this function we can update every information of an item.



## **View\_item()**

def viewing\_items():  
 if not items\_dictionary:  
 # print("No items found!")  
 print("\033[1m\033[31mNo items found!\033[0m")  
 else:  
 items = []  
 total\_purchased\_items = **0** for i\_code**,** item\_details in items\_dictionary.items():  
 items.append([i\_code**,** item\_details["I Name"]**,** item\_details["I Brand"]**,** item\_details["I Price"]**,** item\_details["I Quantity"]**,** item\_details["I Category"]**,** item\_details["I Purchased Date"]])  
 total\_purchased\_items += item\_details["Item Quantity"]  
 #headers = ["Item ID", "Item Name", "Item Brand", "Item Price", "Item Quantity", "Item Category", "Purchased Date"]  
 #print(tabulate(items, headers=headers, tablefmt="heavy\_grid"))  
 items = sorted(items**,** key=lambda x: x[**0**]**,** reverse=True)  
  
 headers = ["\033[1mItem code\033[0m"**,** "\033[1mItem Name\033[0m"**,** "\033[1mItem Brand\033[0m"**,**"\033[1mItem Price\033[0m"**,** "\033[1mItem Quantity\033[0m"**,** "\033[1mItem Category\033[0m"**,**"\033[1mPurchased Date\033[0m"]  
 print(tabulate(items**,** headers=headers**,** tablefmt="heavy\_grid"))  
 #print("Total Purchased Items: {}".format(total\_purchased\_items))  
 print("\033[1m\033[93mTotal Purchased Items: {}\033[0m".format(total\_purchased\_items))

By entering "VID" and the total number of the item, the user can display the inventory's contents in decreasing order.



# **Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input | Expected Output | Actual Output | Remark |
| 01 | Enter ‘AID’ for adding items  Ex: Enter your choice : AID | Output 🡪  Enter Item Code:  Enter Item Name:  Enter Item Brand:  Enter Item Price:  Enter Item Quantity:  Enter Item Category:  Enter Purchased Date (DD/MM/YYYY): | Output 🡪  Enter Item Code:  Enter Item Name:  Enter Item Brand:  Enter Item Price:  Enter Item Quantity:  Enter Item Category:  Enter Purchased Date (DD/MM/YYYY): | Test case pass |
| 02 | Try to skip the details without giving it. | Until the necessary information is provided, the system will repeatedly make the same message. | Until the necessary information is provided, the system will repeatedly make the same message. | Test case pass |
| 03 | Give the same item code again and again. | Instead of displaying an error message, ask for a different item code to print a message to the user. | Instead of displaying an error message, ask for a different item code to print a message to the user. | Test case pass |
| 04 | Giving a string value for the item price | Print a message to the user to enter a valid integer number | Print a message to the user to enter a valid integer number | Test case pass |
| 05 | Giving a string value for item quantity | Print a message to the user to enter a valid integer number | Print a message to the user to enter a valid integer number | Test case pass |
| 06 | Enter ‘DID’ for deleting item | Print a message to the user the item is deleted.  Display item deleted from inventory. | Print a massage to the user the item is deleted.  Display item deleted from inventory. | Test case pass |
| 07 | Enter a wrong item code for ‘DID’ | Print a message to user “item not found in inventory” | Print a message to user “item not found in inventory” | Test case pass |
| 08 | Enter ‘UID’ for update items | Print a input statements such as,  New details for the item (Don’t Enter Anything If you need to keep the recent record.):  Previous Item Name: Viva book  New Item Name:  Previous Item Brand: Asus  New Item Brand:  Previous Item Price: 420000.0  New Item Price:  Previous Item Quantity: 18  New Item Quantity:  Previous Item Category: laptop  New Item Category:  Previous Purchased Date: 18.12.2023  New Purchased Date (DD/MM/YYYY):  0002 (notebook) details updated. | Print a input statements such as,  New details for the item (Don’t Enter Anything If you need to keep the recent record.):  Previous Item Name: Viva book  New Item Name:  Previous Item Brand: Asus  New Item Brand:  Previous Item Price: 420000.0  New Item Price:  Previous Item Quantity: 18  New Item Quantity:  Previous Item Category: laptop  New Item Category:  Previous Purchased Date: 18.12.2023  New Purchased Date (DD/MM/YYYY):  0002 (notebook) details updated. | Test case  pass |
| 09 | Enter a wrong item code for ‘UID’ | Print a message to user “item not found in inventory” | Print a message to user “item not found in inventory” | Test case pass |
| 10 | Enter ‘VID’ for view item | Print a table contains all items and total purchased items in the inventory system | Print a table contains all items and total purchased items in the inventory system | Test case pass |
| 11 | Enter ‘SID’ for saving the item details to the text file at any time. | Print a message to user and save the inventory details in a text file.  Ex: “Inventory saved.” | Print a message to user and save the inventory details in a text file.  Ex: “Inventory saved.” | Test case pass |
| 12 | Enter ‘SDD’ for selecting four dealers randomly from a file. | Print a message to user  Ex: “ 4 Dealers are selected Randomly” | Print a message to user  Ex: “ 4 Dealers are selected Randomly” | Test case pass |
| 13 | Enter “VRL” for displaying all the details of the randomly selected dealers. | Print “ ‘dealer name’ , ‘Contact number’ , ‘Dealer location’ , ‘Item name’ , ‘Brand’ , ‘price’ , ‘quantity’” in a table | Print “ ‘dealer name’ , ‘Contact number’ , ‘Dealer location’ , ‘Item name’ , ‘Brand’ , ‘price’ , ‘quantity’” in a table | Test case pass |
| 14 | If User enters ‘VRL’ option without entering ‘SDD’ option | Print a message to user  Ex: “You have not selected any dealers yet” | Print a message to user  Ex: “You have not selected any dealers yet” | Test case pass |
| 15 | Enter ‘LDI’ for display the items of the given dealer. | Ask user to enter a dealer name for display the details about the dealer.  Ex: “Enter Dealer Name ( Please select from the randomly selected dealer table ) :” | Ask user to enter a dealer name for display the details about the dealer.  Ex: “Enter Dealer Name ( Please select from the randomly selected dealer table ) :” | Test case pass |
| 16 | Enter a wrong name what not in the system | Print a message to user  Ex: “Dealer is not in the system (Please use randomly selected dealers...)” | Print a message to user  Ex: “Dealer is not in the system (Please use randomly selected dealers...)” | Test case pass |
| 17 | Enter ‘ESC’ to exit the program. | Print a massage to user the program was end.  Ex: “Thankyou” | Print a massage to user the program was end.  Ex: “Thankyou” | Test case pass |

## **Test Case 01**

Enter ‘AID’ for adding items

Ex: Enter your choice : AID

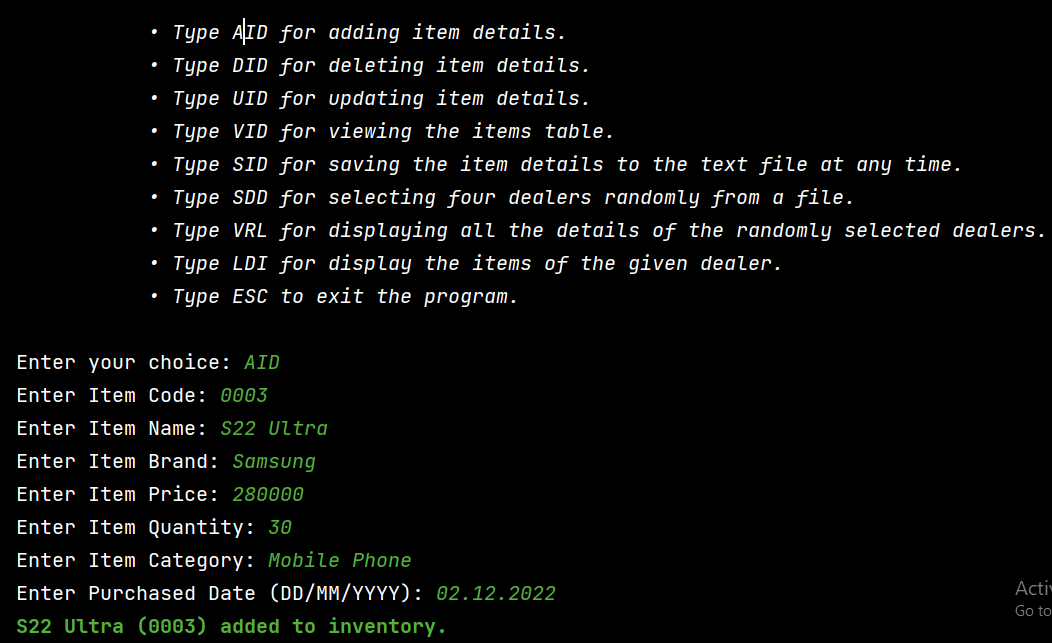


Figure 1

## **Test case 02**

Try to skip the details without giving it. ( exception handling)

**Text

Description automatically generated**

Figure 2

## **Test Case 03**

Give the same item code again and again (exception handling)

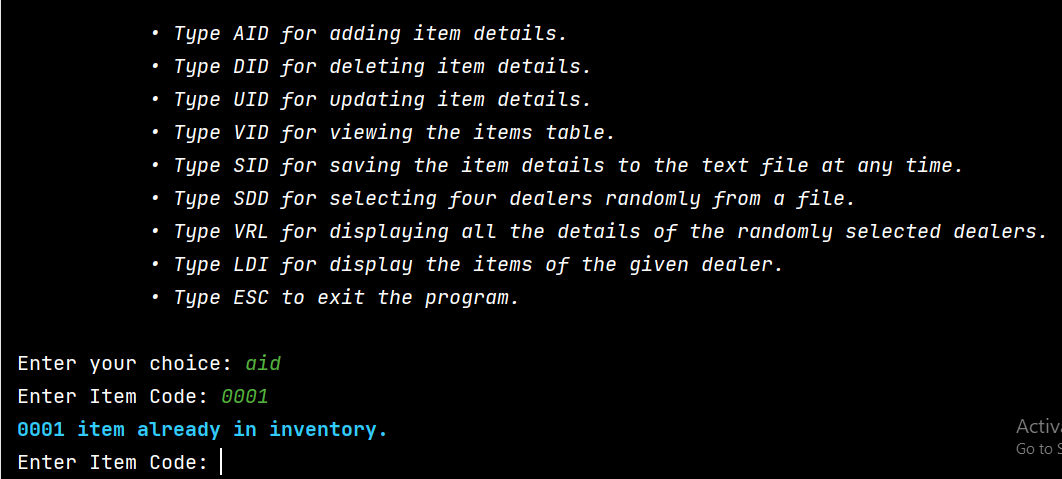
****

Figure 3

## **Test Case 04**

Giving a string value for the item price. (Exception handling)

**Text

Description automatically generated**

## **Test case 05**

Giving a string value for item quantity (Exception handling)

**Text

Description automatically generated**

Figure 4

## **Test case 06**

Enter ‘DID’ for deleting item

**Text

Description automatically generated**

Figure 5

## **Test Case 07**

Enter a wrong item code for ‘DID’ (exception handling)

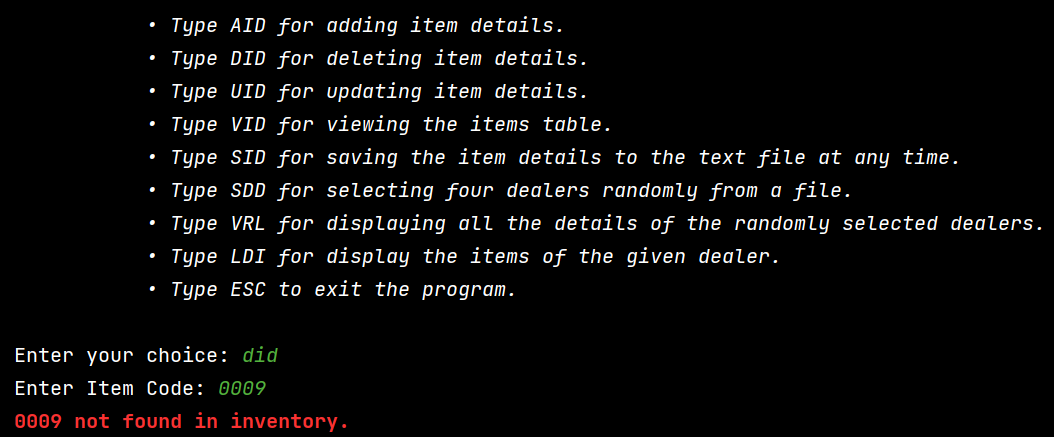


Figure 6

## **Test case 08**

Enter ‘UID’ for update items

****

Figure 7

## **Test Case 09**

Enter a wrong item code for ‘UID’ (exception handling)

**Text

Description automatically generated**

Figure 8

## **Test Case 10**

Enter ‘VID’ for view item

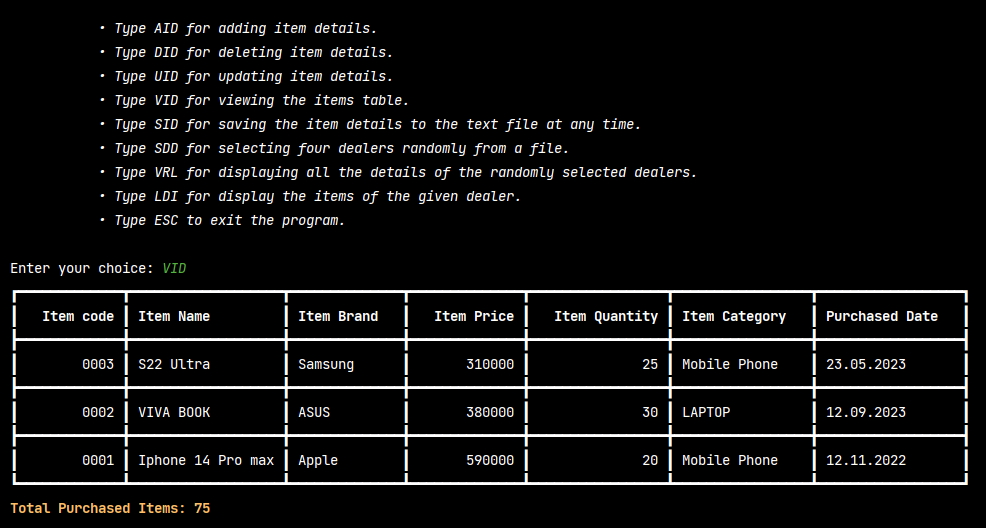
****

Figure 9

## **Test case 11**

Enter ‘SID’ for saving the item details to the text file at any time.

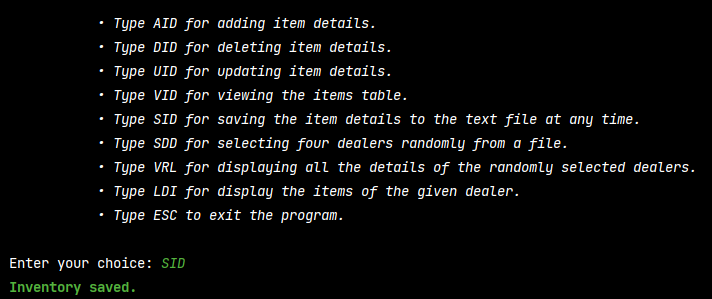
****

Figure 10

### **Text File**

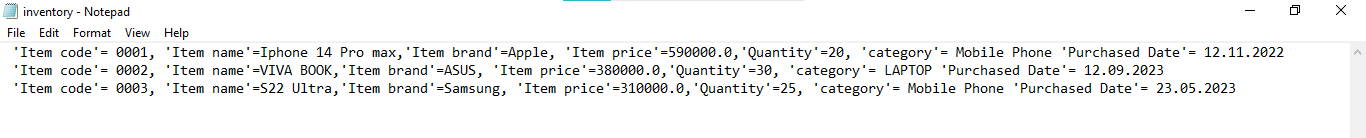


Figure 11



Figure 12

## **Text Case 12**

Enter ‘SDD’ for selecting four dealers randomly from a file.

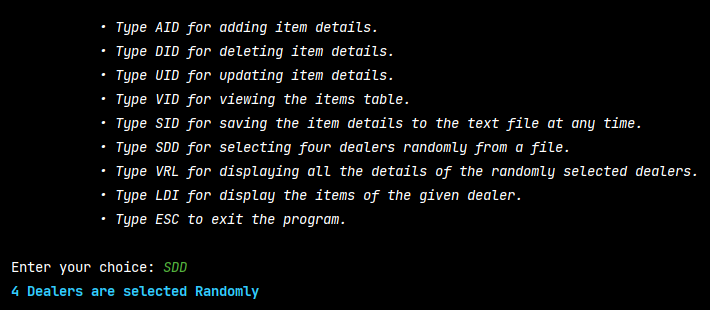
****

Figure 13

## **Test case 13**

Enter “VRL” for displaying all the details of the randomly selected dealers.

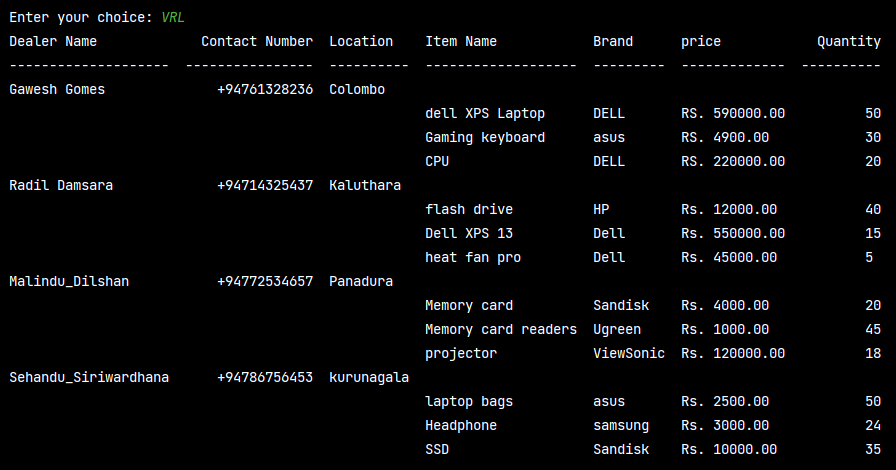
****

Figure 14

## **Test Case 14**

If User enters ‘VRL’ option without entering ‘SDD’ option (exception handling)

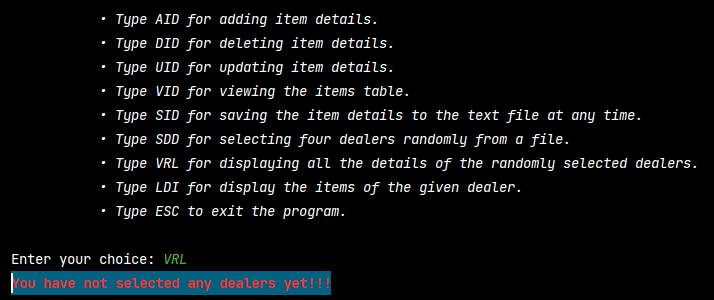
****

Figure 15

## **Test case 15**

Enter ‘LDI’ for display the items of the given dealer.

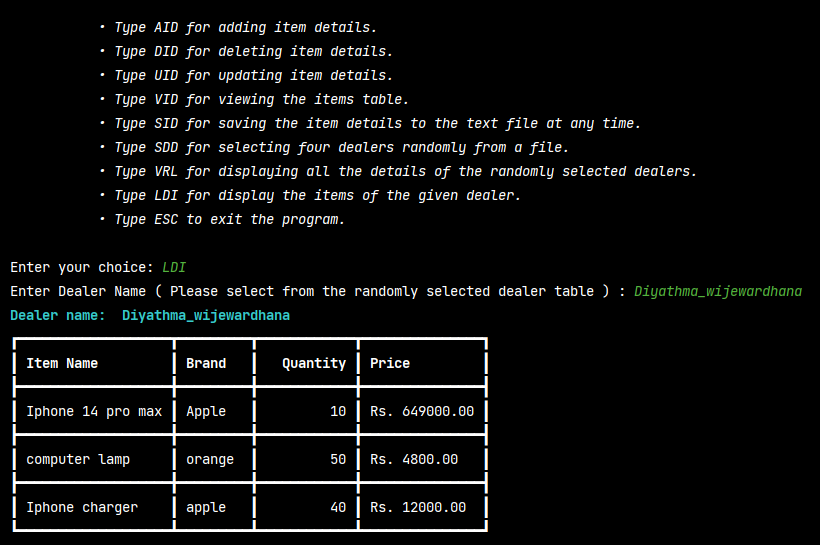
****

Figure 16

## **Test case 16**

Enter a wrong name what not in the system (exception handling)

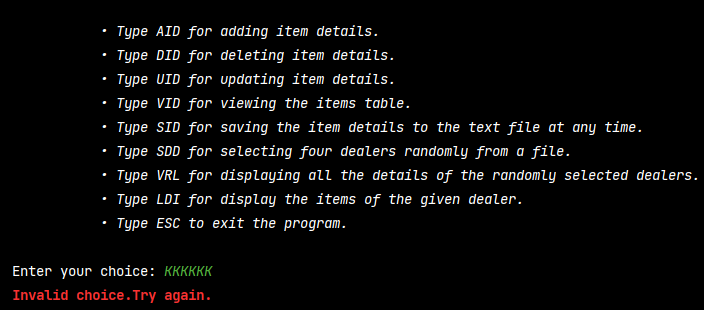
****

Figure 17

## **Test case 17**

Enter ‘ESC’ to exit the program.

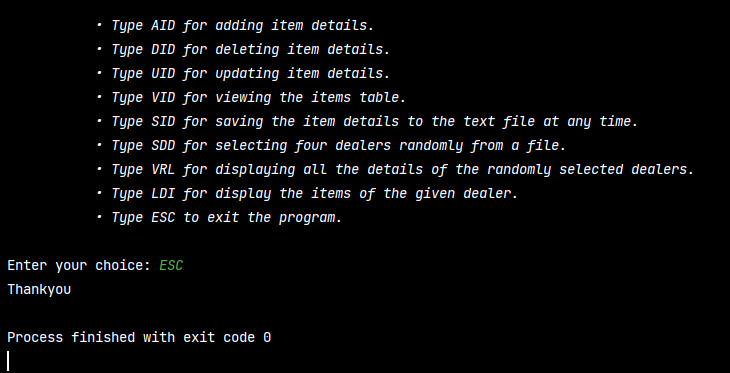
****

Figure 18

# **Conclusion**

A One Net Cafe's inventory system is seen in the code above. The user can see, edit, add, and delete things from the inventory using the system. The system is built on a dictionary structure, in which every item is represented by a special code and a list of properties including name, brand, price, quantity, category, and date of purchase.

A straightforward command-line interface provided by the application helps the user navigate through the many procedures. It is interactive and user-friendly. For the purpose of preventing improper user input, the application also has systems for managing errors and validating data. The program then has a feature that displays the information for four dealers after randomly selecting them from a file. The program gains some entertainment value from this feature by adding an element.

Overall according to my assumption, the inventory system is a well-structured and functional program that provides an efficient and convenient way to manage the inventory of a One Net Cafe. The code can be further improved by adding more features, such as search or sorting functions, or by implementing a graphical user interface to enhance the user experience.