LONDON METROPOLITAN UNIVERSITY

informatics college • pokhara

Module Code & Module Title
CS4051NP Fuhndamentals Of Computing

Assessment weightage & Type

60% Individual Coursework - 1

Year and Semester
2022-23 Autumn

Student name: Prabal Gurung

Group: C3

London met ID: 22069041

College ID:NP04CP4A220088

Assignment due date: 12 May 2023

Assignment submission date: 12 May 2023

Declaration

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded.

Contents

List of figures:

List of tables:

1. Introduction:

This project is about a laptop building a system that buys and sells from manufacture and customers respectively. This project will be able to place order and change its stock accoroding to the inputs of user and change (.txt) file everytime order is placed. This project consists of five main module after a carefull making and study of flowchart. The module name are: main.py, operation.py, displayOperation.py, read.py, write.py all the modules are explained in detail below.

Goals and Objective:

To develop a program which helps in selling and buying of laptops.

To keep record of transaction and store in text files.

To generate a bill and store whenever transaction is done.

2. Algorithm:

An algorithm is step by step procedure designed to solve a specific problem or accomplish a particular task. It provides a clear and precise description on how to perform a series of operations or computations in a logical and efficient manner.

STEP 1: Start
STEP 2: While True
    STEP 3: Display Main module
    STEP 4: Input userInput
    STEP 5: If userInput equals one, then do the following
        STEP 6: DISPLAY Table
        STEP 7: Input "any button"
    STEP 8 Else (if userInput equals two), do the following
        STEP 9: Clear screen and print suitable message
        STEP 10: Try the following
            STEP 11: Perform the necessary input
            STEP 12: Handle the potential exception
        STEP 13: Except (if errror occurs), do the following
            STEP 14: Display warning, goto 9
        STEP 15: If name, address is greater than zero and number equals ten
            STEP 16: While True
                STEP 17: Display table, list
                STEP 18: Get stockData from read class
                STEP 19: Try the following:
                STEP 20: Perform the necessary input
                STEP 21: Minus choice of user by one
                STEP 22: If stock minus demand greater than zero
                STEP 23: If demand and choice greater than zero
                STEP 24: Perform necessary calculation
                STEP 25: Make bill
                STEP 26: Change stock
                STEP 27: Try following
                STEP 28: Input selection
                STEP 29: Handle the potential exception
                STEP 30: Except(If an exception), do
                STEP 31: Display warning, goto 27
                STEP 32: If selection equals one
                STEP 33: Goto Step 16
                STEP 34: ELSE (if selection equals two), do
                STEP 35: Clear screen
                STEP 36: Add final total in bill
                STEP 37: Show Bill
                STEP 38 Display "ThankYou"
                STEP 39: Goto step 2

STEP 40: ELSE (if none from above), do
STEP 41: Display warning
STEP 42: Goto step 27
STEP 43: ELSE (if condition false), do
STEP 44: Display Warning
STEP 45: ELSE (if condition false), do
STEP 46: Display "Out of Stock"
STEP 47: Except (if error occurs), do
STEP 48: Display warning, goto 19
STEP 49: Else (If any Input equals zero), do the following
STEP 50: Display exit, goto 2
STEP 51: Else (if both condition false), do the following
STEP 52: Display warning, goto 9
STEP 53: Else (if userInput equals 3), do the following
STEP 54: Clear screen and print suitable message
STEP 55: Try the following
STEP 56: Perform the necessary input
STEP 57: Handle the potential exception
STEP 58: Except (if errror occurs), do the following
STEP 59: Display warning, goto 54
STEP 60: If name, address is greater than zero and number equals ten
STEP 61: While True
STEP 62: Display table, list
STEP 63: Get stockData from read class
STEP 64: Try the following:
STEP 65: Perform the necessary input
STEP 66: Minus choice of user by 1
STEP 67: If choice and demand greater thanzero
STEP 68: Calculate Amount
STEP 69: Refill stock
STEP 70: Make bill
STEP 71: Try the following
STEP 72: Input selection
STEP 73: Except(If an exception), do
STEP 74: Display warning, goto 71
STEP 75: If selection equals one
STEP 76: Goto Step 61
STEP 77: ELSE (if selection equals two), do
STEP 78: Clear screen
STEP 79: Add final total in bill
STEP 80: Show Bill
STEP 81: Display "ThankYou"
STEP 82: Goto step 2
STEP 83: ELSE (if none from above), do
STEP 84: Display warning
STEP 85: Goto step 71

STEP 86: Else (If condition flase), do

STEP 87: Display warning
STEP 88: Except (if error occurs), do
STEP 89: Display warning
STEP 90 Else (above condition equals false), do the following
STEP 91: Dislpay suitable message
STEP 92: Else (if userInput equals four) do the following:
STEP 93: Display exit
STEP 94: END
STEP 95: Else (If all condition false), do the following:
STEP 96: Display warning

3. Pseudocode

Pseudocode is a detailed yet readable description of what a computer program is suppose to do, expressed in a formally-styled natural language rather than programming language itself. (S, 2023)

3.1 Main.py

```
IMPORT displayOperation
IMPORT operation

WHILE True
    CALL cos_Main
    INPUT userInput
    IF userInput == 1
        CALL get_stockTable
    ELSE IF userInput == 2
        CALL sellProcess
    ELSE IF userInput == 3
        CALL buyFromVendor
    ELSE IF userInput == 4
        CALL exit
        break
    ELSE
        CALL warningMainDisplay
```

3.2 operation.py

```
IMPORT os
IMPORT displayOperation
IMPORT read
IMPORT write

INITIALIZE grandTotal
INITIALIZE vendorTotal

METHOD butFromVendor
        CALL os
        PRINT suitable message
        TRY
                INPUT vendorName
        EXCEPT
                PRINT warning message
        IF vendorName length is less than zero
                CALL proceedBuy
        ELSE IF vendorName is equal to zero
                CALL exit
        ELSE
                CALL warningMainDIsplay
        END IF
END METHOD

METHOD proceedBuy
        WHILE True:
                CALL getStockTable
                CALL getListForSell
                INITIALIZE stockData
                TRY
                        INPUT userChoice
                        INPUT userDemand

                        CALCULATE userChoice

                        IF userChoice, userDemand greater than zero and userChoice smaller than five
                                INITIALIZE price WITH stockData array
                                CALCULATE totalAmount
                                CALL vendorTotal
                                CALCULATE vendorTotal
                                CALL refillstock
```

```
                        CALL vendorBill
                        CALL buyAgainVendor
                        BREAK
                ELSE
                        INPUT any
                END IF
        EXCEPT
                CALL waningMainDisplay
    END WHILE
END METHOD
METHOD buyAgainVendor
    CALL buyMore
    TRY
        INPUT userSelection
    EXCEPT
        CALL warningMainDisplay
    IF userSelection == 1
        CALL proceedBuy
    ELSE IF userSelection == 2
        CALL os
        CALL addVendorTotal
        CALL displayBill
    ELSE
        CALL warningMainDisplay
        CALL buyAgainVendor
    END IF
END METHOD
METHOD sellProcess
    CALL os
    PRINT suitable message
    TRY
        INPUT customerName
        INPUT customerAddress
        INPUT cusotmerNumber
    EXCEPT
        CALL exit
    IF customerName, customerAddress has length greater than 0 and customerNumber equals 10
        CALL startSell
    ELSE IF customerName, customerAddress or customerNumber equals 0
        CALL exit
    ELSE
        CALL warningMainDisplay
    END IF
END METHOD
METHOD startSell
    WHILE True
        CALL get_stockTable
        CALL getListForSell
```

```
            INITIALIZE stockData
            TRY
                INPUT userChoice
                INPUT userDemand
                CALCULATE userChoice
                IF stock – userDemand is greater than 0
                    IF userChoice and stockData is greater than 0
                        INITIALIZE price
                        CALCULATE totalAmount
                        CALL grandTotal
                        CALCULATE grandTotal
                        CALL bill
                        CALL changeStock
                        CALL buyAgain
                        BREAK
                    ELSE
                        CALL warningMainDIsplay
                    END IF
                ELSE
                    CALL os
                    PRINT suitable message
            EXCEPT
                CALL waningMainDisplay
        END WHILE
END METHOD
METHOD buyAgain
        CALL buyMore
        TRY
            INPUT userSelection
        EXCEPT
            CALL warningMainDisplay
        IF userSelection == "1"
            CALL startSell
        ELSE IF
            CALL os
            CALL addGrandTotal
            CALL readData
        ELSE
            CALL warningMainDisplay
            CALL buyAgain
        END IF
END METHOD
SET grandTotal
SET vendorTotal
```

3.3 read.py

```
METHOD get_stockData
        INTIALIZE data
        OPEN stock.txt AS file
        READ file
        FOR i IN file length
                APPEND data
        END FOR
        CLOSE file
        RETURN data
END METHOD

METHOD displayBill
        OPEN bill.txt AS bill
        PRINT bill
        CLOSE bill
END METHOD

METHOD readData
        OPEN bill.txt AS bill
        PRINT bill
        CLOSE bill
END METHOD
```

3.4 write.py

```
METHOD refillStock
        INITIALIZE stockData
        CALCULATE stockData
        CALL write_stockData_to_file
END METHOD

METHOD changeStock
        INITIALIZE stockData
        CALCULATE stockData
        CALL write_stockData_to_file
END METHOD

METHOD write_stockData_to_file
        INITIALIZE stockData_str
        FOR sublist IN stockData
                INITIALIZE line
                FOR item IN sublist
                        ADD item
                END FOR
                CALCULATE line
                CALCULATE stockData_str
        END FOR
        OPEN stock.txt AS file
        WRITE stockData_str
        CLOSE file
END METHOD
```

3.5 displayOperation.py

```
IMPORT os
IMPORT read

METHOD cosMain
        CALL os
        PRINT suitable ascii art
        PRINT suitable table
END METHOD

METHOD
        CALL os
        INITAILIZE reads
        INITIALIZE j
        PRINT header of table
        FOR row in reads
                PRINT reads array
        END FOR
END METHOD

METHOD getListForSell
        PRINT suitable message
END METHOD
METHOD
        PRINT Thank You
END METHOD
METHOD warningMainDisplay
        PRINT error message
END METHOD
METHOD thankYou
        PRINT suitable message
END METHOD

METHOD buyMore
        PRINT suittable message
END METHOD
METHOD bill
        IF os have path
                OPEN Bill.txt AS file
                WRITE appended item IN file
                CLOSE file
        ELSE
                OPEN Bill.txt AS file
                WRITE make bill
                CLOSE file
```

```
        END IF
END METHOD
METHOD addGrandTotal
        OPEN Bill.txt AS file
        WRITE appended item IN file
        CLOSE file
END METHOD
METHOD vendorBill
        IF os have path
                OPEN Bill.txt AS file
                WRITE appended item IN file
                CLOSE file
        ELSE
                OPEN Bill.txt AS file
                WRITE make bill
                CLOSE file
        END IF
END METHOD

METHOD addVendorTotal
        Calculate withoutVAT
        OPEN Bill.txt AS file
        WRITE IN file
END METHOD
```

4. Flowchart:

Flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can br adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process or a project plan. (ASQ, 2023)

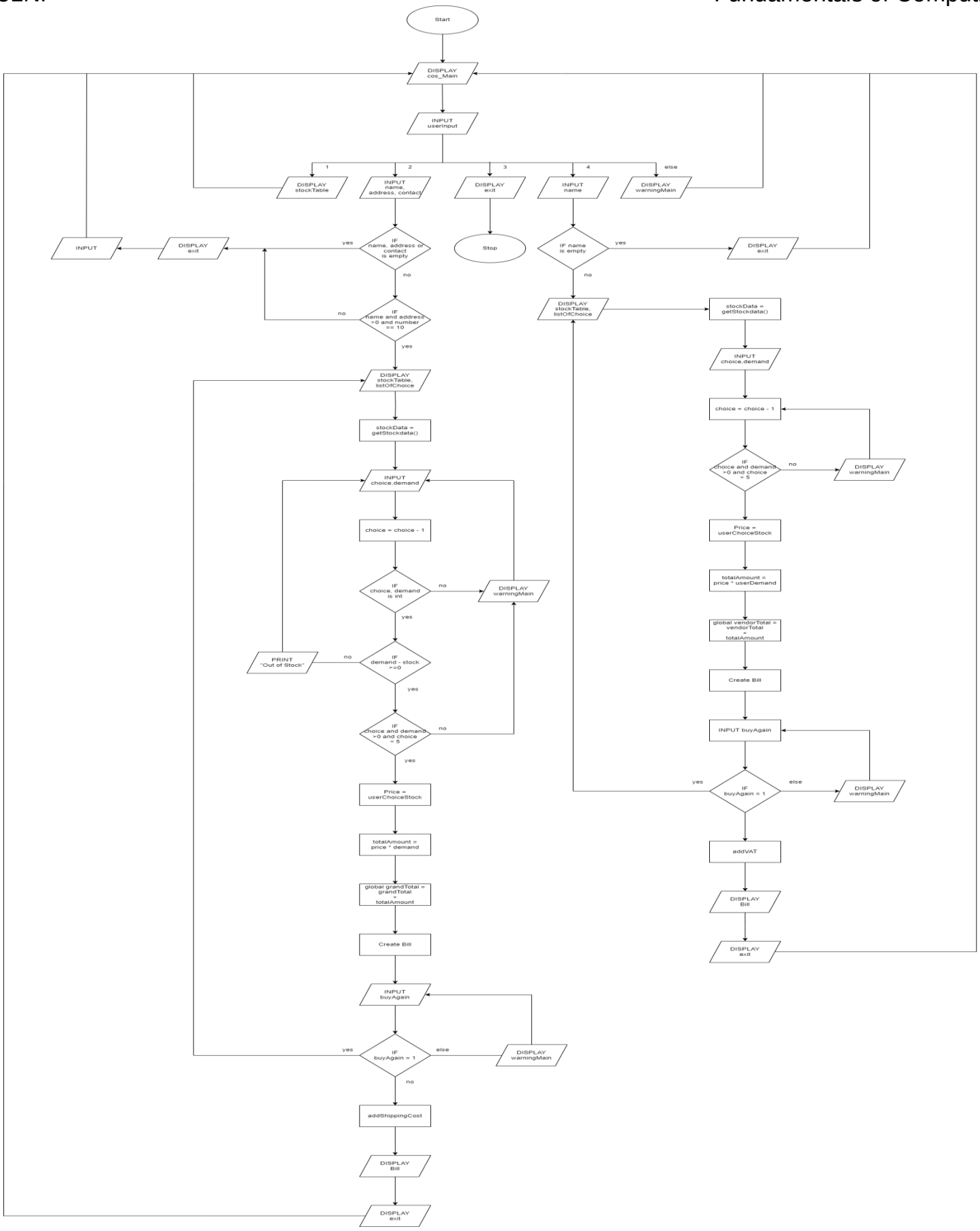


Figure 1 Flowchart of the program

5. Data structure

A data structure is a way of organinzing and storing data in a computer system or program. It provides a systematic and efficient way to access and manipulate data. It defines the relationship and interactions between elements, allowing for efficient storage, retrieval and modification.

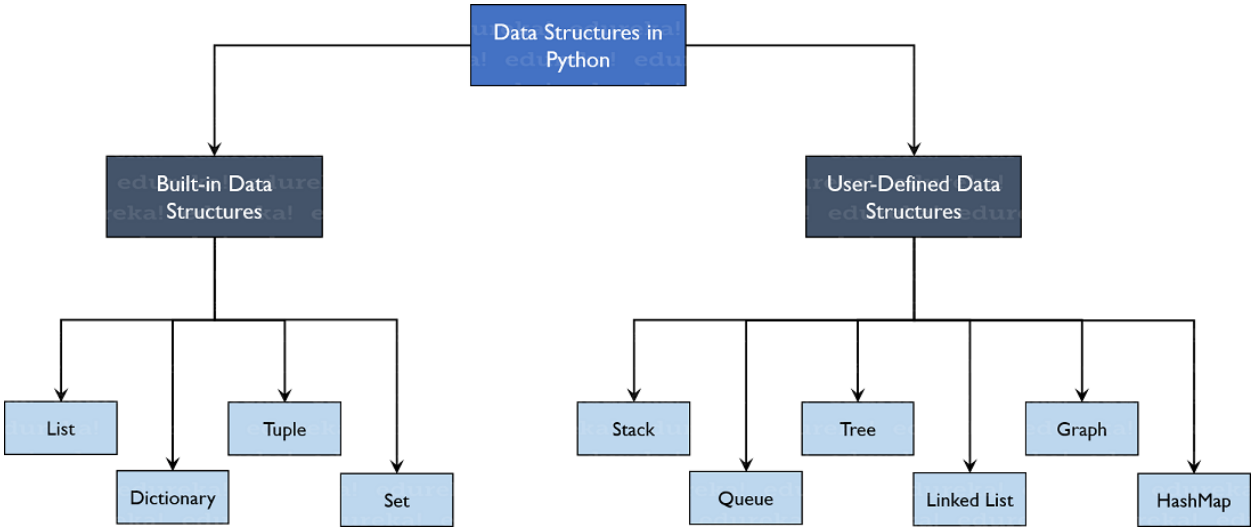


Figure 2 Data Structure in Pyhton

In Python programming language, there are two types of Data Structures:

Built-in Data Structure
User-Defined Data Structure

Here the list of data structure that has been used during the project:
String:

```
userInput = input("Choose from (1-4): ")
```

Figure 3 String

String is a primitive data type and mainly used to store text files. String is almost used in most of the places in project like for making bill or accepting name input from users. And text file are also initially stored in string.

Integer:

```
userChoice = int(input("Enter Your Choice: "))
userDemand = int(input("Enter Number Of Laptop: "))
```

Figure 4 integer

Integer is a primitive data type that are used to store numerical datas. Integer also called int, has been used in various topics like in calculation of stock changing stock or adding VAT and shipping cost. This interger helps to perform various simple math.

Boolean

```
while True:
```

Figure 5 Boolean

Boolean is a primitice data types and it has been used to loop the program until user breaks or sets it to false. Boolean has only two mode true or false and is helpful in various loops or choosing between and two direction

List

```
data = read.get_stockData()
```

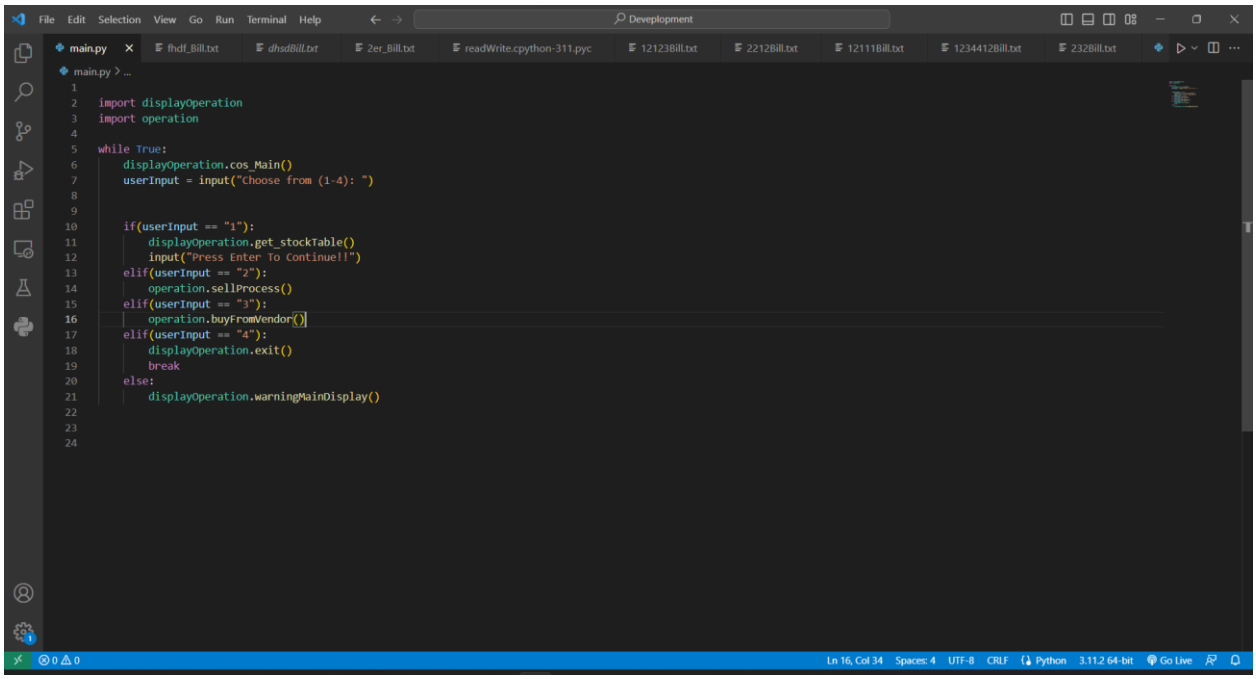
Figure 6 List

List data structure are used to store the values in array. It helps in easier access. List is an array which will help to store multiple data at once. Usally the extraction of file from outside is handled in list.

6. Program

The project is designed to control all the laptops stock exports and imports encompasses a comprehensive system the efficiently manages the movement of laptops in and out if the inventory. The project aims to efficiently control the exports and imports of laptop stocks.

6.1 Implementation of Program:



```python
import displayOperation
import operation

while True:
    displayOperation.cos_Main()
    userInput = input("Choose from (1-4): ")


    if(userInput == "1"):
        displayOperation.get_stockTable()
        input("Press Enter To Continue!!")
    elif(userInput == "2"):
        operation.sellProcess()
    elif(userInput == "3"):
        operation.buyFromVendor()
    elif(userInput == "4"):
        displayOperation.exit()
        break
    else:
        displayOperation.warningMainDisplay()
```

Figure 7 main module

This is the main code from where the project is implemented and all the other frames are added along. This code input is accepted in String so any exception other than 1, 2, 3, 4 will be terminated to final else. which will determine it as wrong value and loops the program with while.

6.2 Purchase Phase of Program:

```python
def buyFromVendor():
    os.system("cls")
    print("Leave name empty to go to Main Screen")
    print("")
    print("")
    try:
        vendorName = input("Enter Company Name: ")
    except:
        displayOperation.warningMainDisplay()

    if (len(vendorName) > 0):
        proceedBuy(vendorName)
    elif (len(vendorName) == 0):
        displayOperation.exit()
        input()
    else:
        displayOperation.warningMainDisplay()

def proceedBuy(vendorName):
    while True:
        displayOperation.get_stockTable()
        displayOperation.getListForSell()
        stockData = read.get_stockData()

        try:
            userChoice = int(input("Enter Your Choice: "))
            userDemand = int(input("Enter Number Of Laptop: "))

            userChoice = userChoice - 1

            if(userChoice >= 0 and userDemand > 0 and userChoice < 5):
                price = stockData[int(userChoice)][2].replace("$","")
                totalAmount = int(price) * int(userDemand)
                global vendorTotal
                vendorTotal += totalAmount
```

Figure 8 operation module 1

```python
                write.refillStock(userDemand, userChoice)
                displayOperation.vendorBill(vendorName, userChoice, userDemand, totalAmount)
                buyAgainVendor(vendorName)
                break
            else:
                input("Please Enter Valid Choice")
        except:
            displayOperation.warningMainDisplay()

def buyAgainVendor(vendorName):
    displayOperation.buyMore()
    try:
        userSelection = input("Choose 1 or 2: ")
    except:
        displayOperation.warningMainDisplay()
    if (userSelection == "1"):
        proceedBuy(vendorName)
    elif(userSelection == "2"):
        os.system("cls")
        displayOperation.addVendorTotal(vendorName, vendorTotal)
        read.displayBill(vendorName)
        input("Press Enter To Continue")
    else:
        displayOperation.warningMainDisplay()
        input("Press any key to Continue")
        buyAgainVendor(vendorName)
```

Figure 9 operation module 2

This is the Buy phase of the program where all the necessary function are called when needed. Firstly user input validation is checked and after the verification project changes the stock and prints the desired bill by calling function. And in case of exception or error warning message is called. The program runned in shell is shown in test 3

All the invalid input user put will throw an exception that will call warning message display.

6.3 Sell Phase of Program:

```python
def sellProcess():
    os.system("cls")
    print("Leave a field empty to go to Main Screen")
    print("")
    print("")
    try:
        customerName = input("Enter Customer Name: ")
        customerAddress = input("Enter Customer Address: ")
        customerNumber = input("Enter Customer Number: ")
    except:
        displayOperation.exit()

    if(len(customerName) > 0 and len(customerAddress) > 0 and len(customerNumber) == 10):
        startSell( customerName, customerAddress, customerNumber)
    elif(len(customerName) == 0 or len(customerAddress) == 0 or len(customerNumber) == 0):
        displayOperation.exit()
        input()
    else:
        displayOperation.warningMainDisplay()

def startSell(customerName, customerAddress, customerNumber):
    while True:
        displayOperation.get_stockTable()
        displayOperation.getListForSell()
        stockData = read.get_stockData()

        try:
            userChoice = input("Enter Your Choice: ")
            userDemand = input("Enter Number Of Laptop: ")

            userChoice = str(int(userChoice) - 1)

            if(int(stockData[int(userChoice)][3]) - int(userDemand) >= 0:
                if((int(stockData[int(userChoice)][3]))>= 0 and int(userChoice) >= 0):
                    price = stockData[int(userChoice)][2].replace("$","")
                    totalAmount = int(price) * int(userDemand)
```

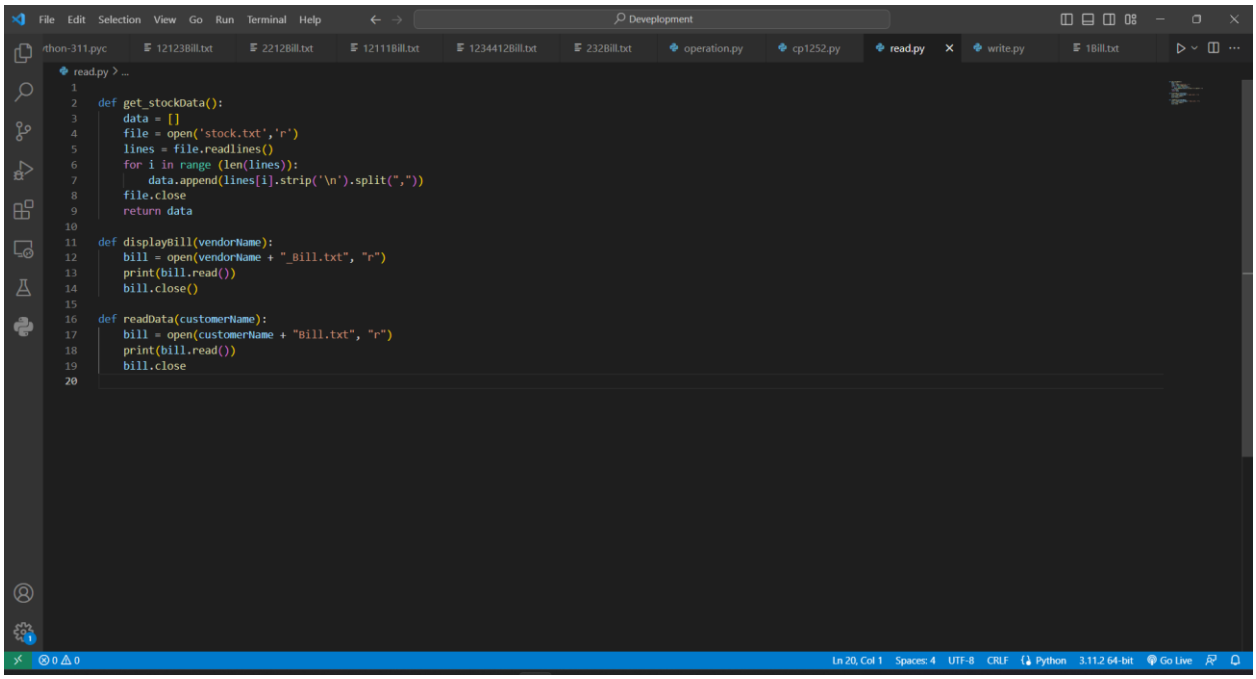
Figure 10 Operation module 3

```python
                    grandTotal += totalAmount

                    displayOperation.bill(customerName, customerAddress, customerNumber, userDemand, userChoice, stockData, totalAmount)
                    write.changeStock(userDemand, userChoice)
                    buyAgain(customerName, customerAddress, customerNumber)
                    break
                else:
                    displayOperation.warningMainDisplay()
            else:
                os.system("cls")
                print("Out Of Stock!! Please Lower Your Demand.")
                input("")
        except:
            displayOperation.warningMainDisplay()

def buyAgain(customerName, customerAddress, customerNumber):
    displayOperation.buyMore()

    try:
        userSelection = input("Choose 1 or 2: ")
    except:
        displayOperation.warningMainDisplay()
    if(userSelection == "1"):
        startSell(customerName, customerAddress, customerNumber)
    elif(userSelection == "2"):
        os.system("cls")
        displayOperation.addGrandTotal(customerName, grandTotal)
        read.readData(customerName)
        input()
        displayOperation.thankYou()
        input("Press Enter To Continue")
    else:
        displayOperation.warningMainDisplay()
        input("Press any key to Continue")
        buyAgain(customerName, customerAddress, customerNumber)
```

Figure 11 Operation module 4

This is the sell phase where all the function are called which are necessary for selling and same thing is done as the buyFromVendor but stock is decreased instead of increasing the value.
All the invalid input user put will throw an exception that will call warning message display.

6.4 Read



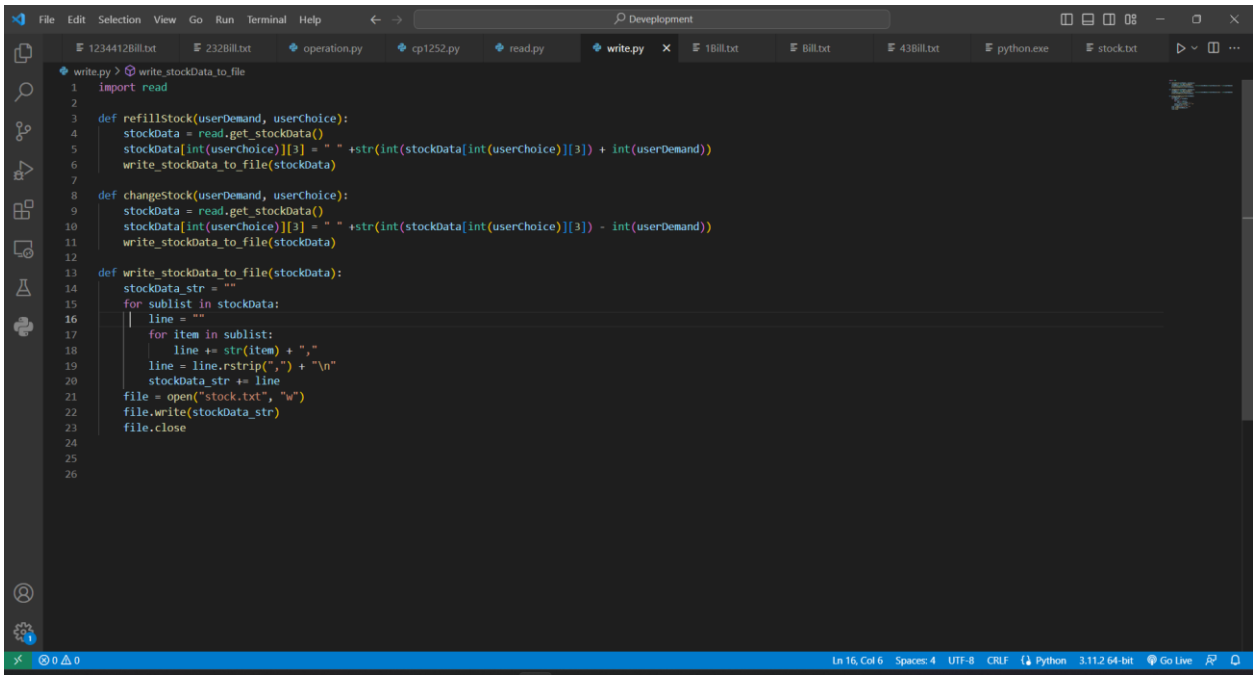
```python
def get_stockData():
    data = []
    file = open('stock.txt','r')
    lines = file.readlines()
    for i in range (len(lines)):
        data.append(lines[i].strip('\n').split(","))
    file.close
    return data

def displayBill(vendorName):
    bill = open(vendorName + "_Bill.txt", "r")
    print(bill.read())
    bill.close()

def readData(customerName):
    bill = open(customerName + "Bill.txt", "r")
    print(bill.read())
    bill.close
```

Figure 12 Read module

This program helps the project to read alll the data outside of program and lets the code use it as its own and helps to build table and read bill to shell it in terminal.

All the invalid input user put will throw an exception that will call warning message display.

6.5 Write



```python
import read

def refillStock(userDemand, userChoice):
    stockData = read.get_stockData()
    stockData[int(userChoice)][3] = " " +str(int(stockData[int(userChoice)][3]) + int(userDemand))
    write_stockData_to_file(stockData)

def changeStock(userDemand, userChoice):
    stockData = read.get_stockData()
    stockData[int(userChoice)][3] = " " +str(int(stockData[int(userChoice)][3]) - int(userDemand))
    write_stockData_to_file(stockData)

def write_stockData_to_file(stockData):
    stockData_str = ""
    for sublist in stockData:
        line = ""
        for item in sublist:
            line += str(item) + ","
        line = line.rstrip(",") + "\n"
        stockData_str += line
    file = open("stock.txt", "w")
    file.write(stockData_str)
    file.close
```

Figure 13 Write module

This code has multiple function where stock changes are done. For when the project owner buys it refills the stock or when the project sells the stock of laptops it will decrease. This code can help make various change in out files, make file and change file when necessary or append.

All the invalid input user put will throw an exception that will call warning message display.

7. Testing

In this topic, there are various screenshot that shows various things that have been tried during project.
All the data change shown in the data below is changed to initial by the end of the project and all test proves that the project code won't shut or crash by itself no matter the input.

7.1 Test1

OBJECTIVE	Implementation of Try, Except
ACTION	Provide invalid input and show the message.
EXPECTED RESULT	Exception message is displayed
ACTUAL RESULT	Exception message is shown
RESULT	Pass

Table 1 Show invalid message

Figure 14 Inserting invalid number

Figure 15 Invalid message shown

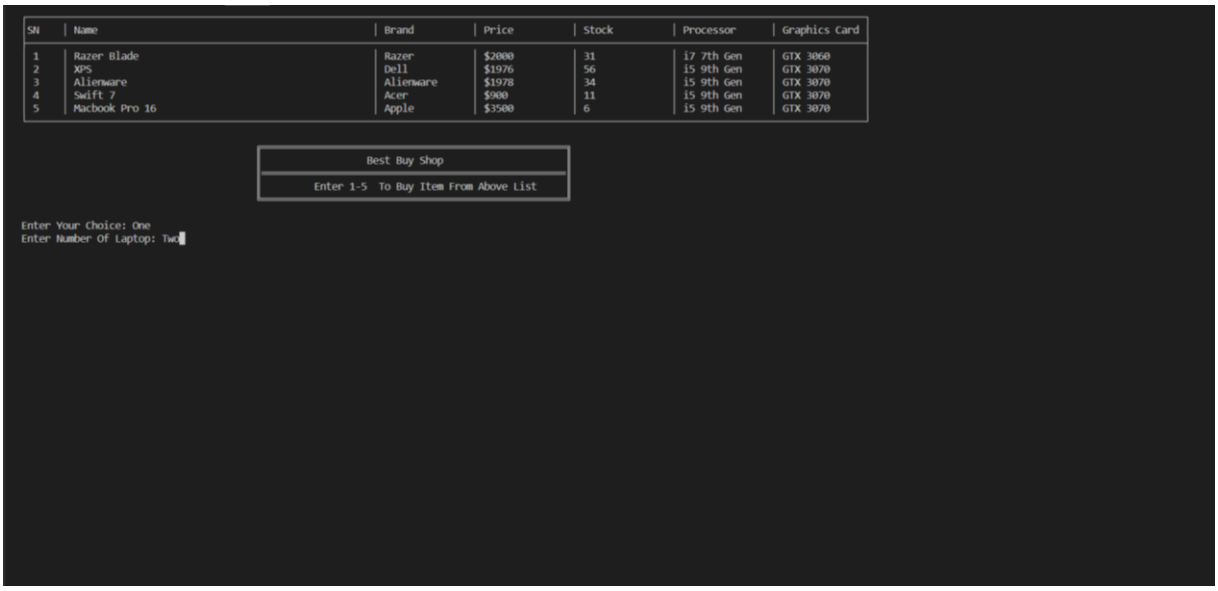
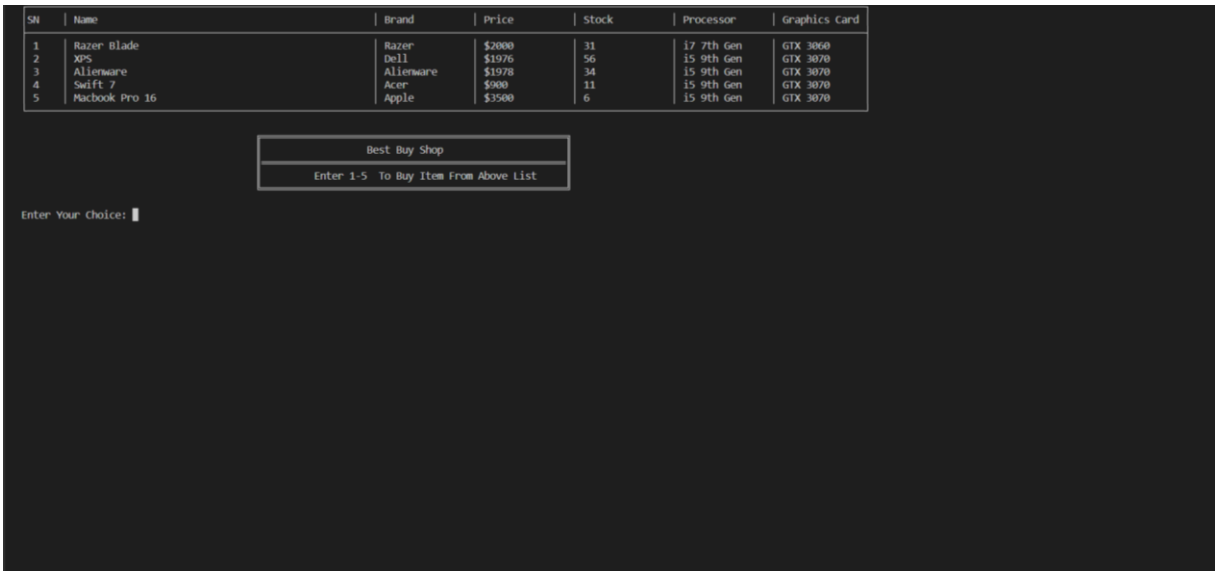


Figure 16 inserting invalid input during selling

Figure 17 Invalid message shown



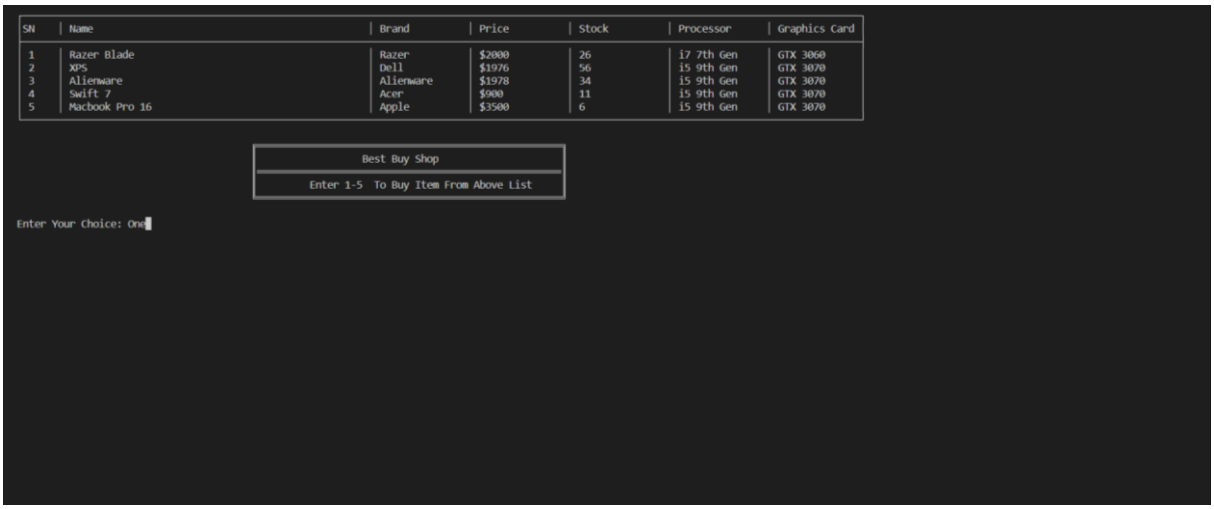
SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	31	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

Best Buy Shop

Enter 1-5 To Buy Item From Above List

Enter Your Choice:

Figure 18 sell program doesn't crash



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	26	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

Best Buy Shop

Enter 1-5 To Buy Item From Above List

Enter Your Choice: One

Figure 19 inserting string in buy phase

Figure 20 Invalid message shown

7.2 Test 2

OBJECTIVE	To Put negative value in demand and show error message
ACTION	Enter negative information in buy from vendor and sell to customer
EXPECTED RESULT	Error message is displayed
ACTUAL RESULT	Error message is shown
RESULT	Pass

Table 2 Put negative value in demand

Sell to customer with negative value

Figure 21 Inserting value in sell

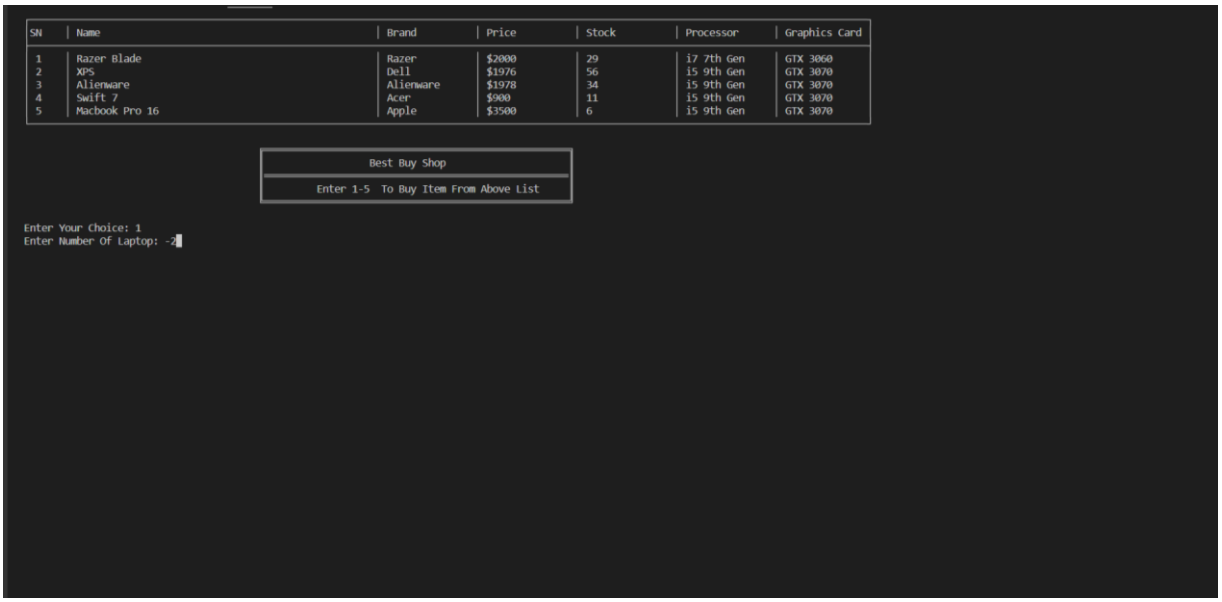
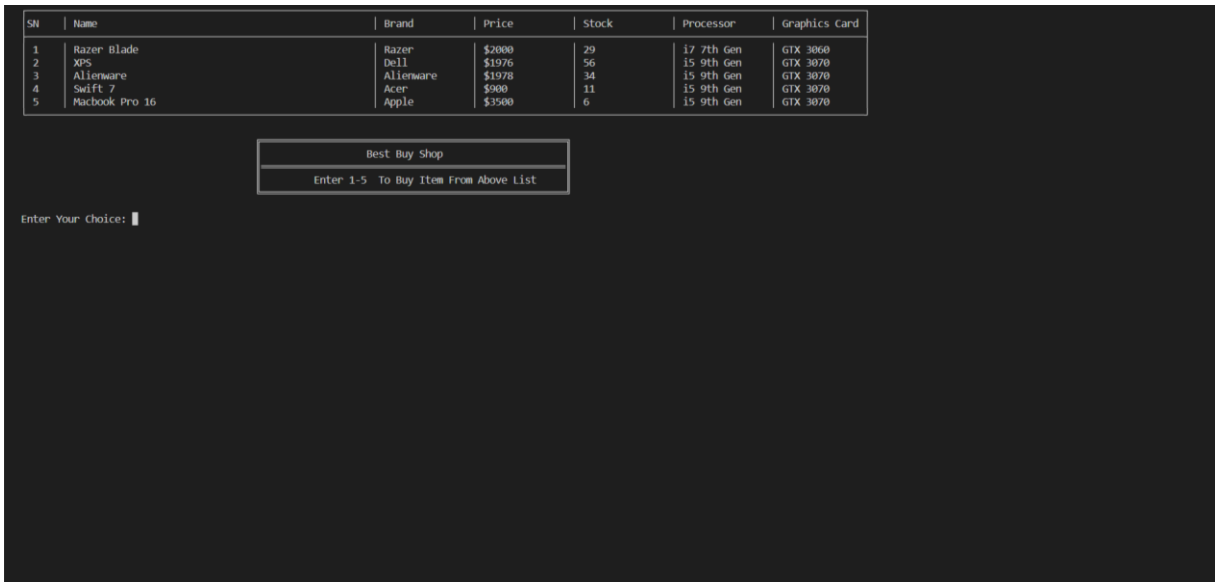


Figure 22 Inserting negative value during stock

Figure 23 Invalid message shown



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	29	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

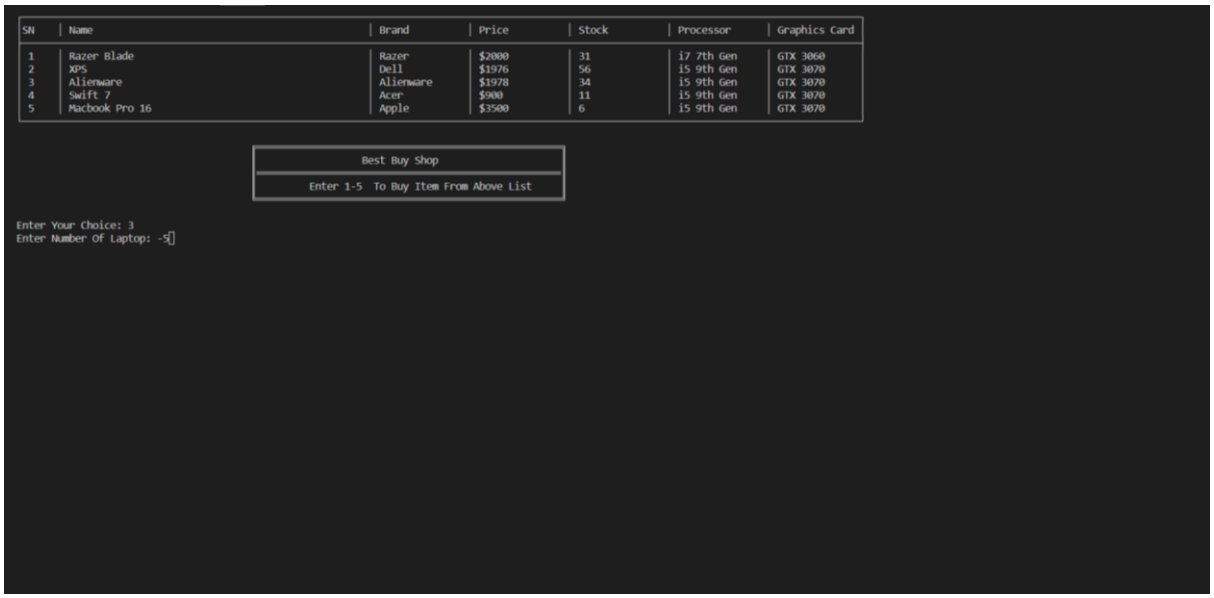
Best Buy Shop

Enter 1-5 To Buy Item From Above List

Enter Your Choice:

Figure 24 Code doesn't crash

Buy from vendor with negative value



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	31	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

Best Buy Shop

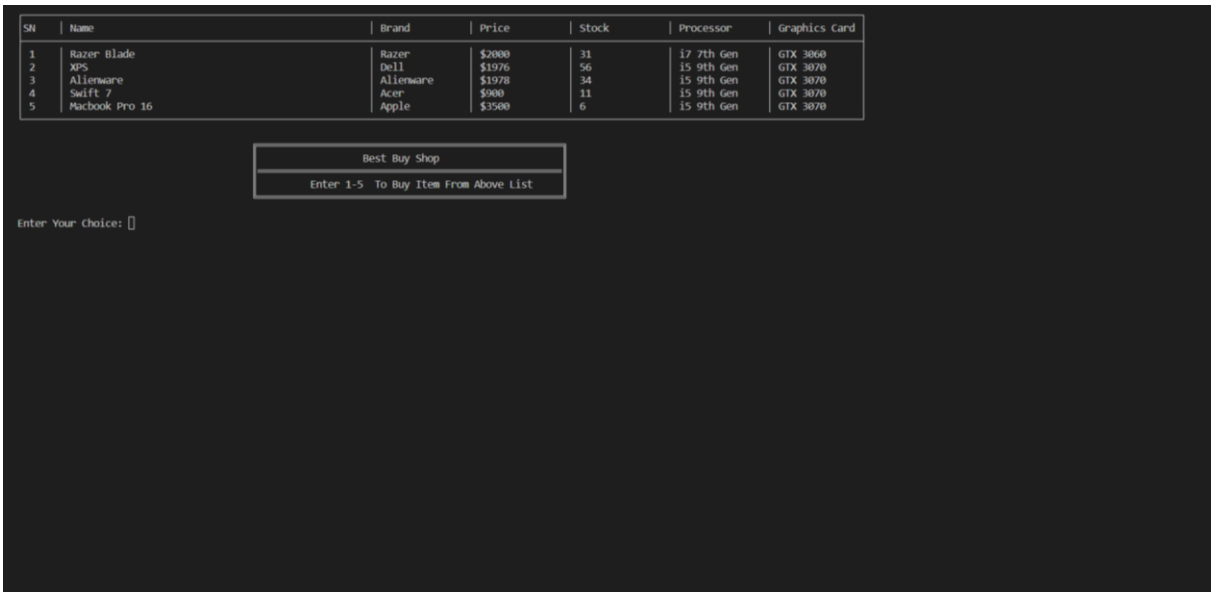
Enter 1-5 To Buy Item From Above List

Enter Your Choice: 3
Enter Number Of Laptop: -5

Figure 25 Inserting negative value in buy phase

Please Enter Valid Choice

Figure 26 Invalid choice shown



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	31	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

Best Buy Shop

Enter 1-5 To Buy Item From Above List

Enter Your Choice:

Figure 27 code not crashing

7.3 Test 3

OBJECTIVE	Purchase multiple laptop from the project
ACTION	Show all the steps required to buy one
EXPECTED RESULT	Bill is made successfully
ACTUAL RESULT	Bill is made and shown succesfully
RESULT	Pass

Table 3 Purchase Table

Figure 28 Main menu

Figure 29 Buy phase

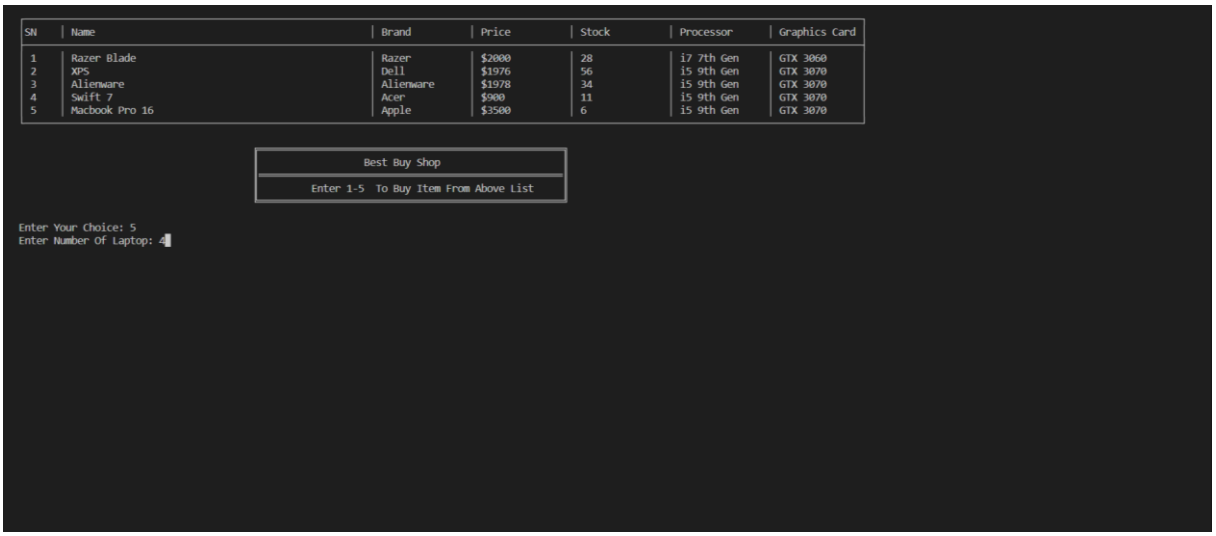


Figure 30 Inserting true value

Figure 31 choosing to buy more

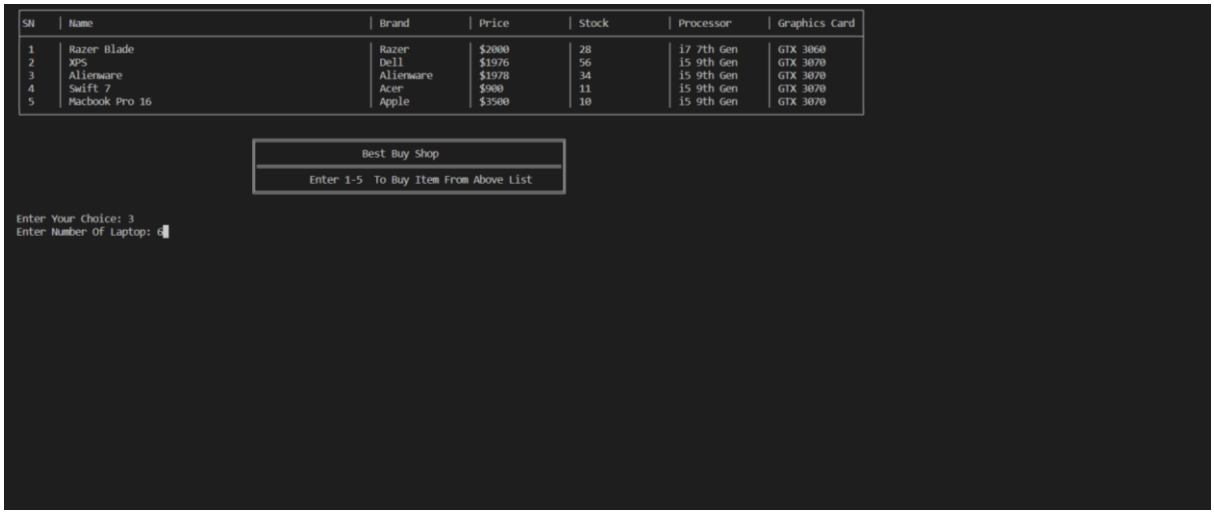
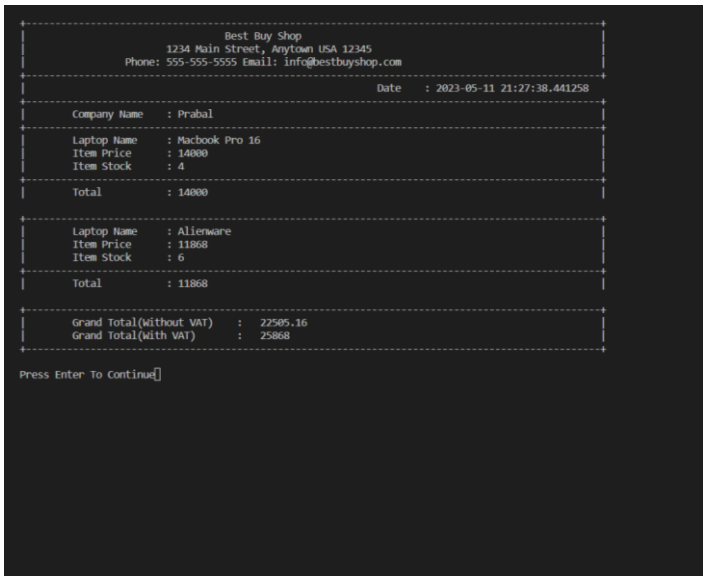


Figure 32 inserting true value again in buy

Figure 33 stop the buy phase



```
+-------------------------------------------------------------------------------------------+
|                                   Best Buy Shop                                           |
|                          1234 Main Street, Anytown USA 12345                              |
|                  Phone: 555-555-5555 Email: info@bestbuyshop.com                          |
+-------------------------------------------------------------------------------------------+
|                                                          Date    : 2023-05-11 21:27:38.441258
+-------------------------------------------------------------------------------------------+
|        Company Name    : Prabal                                                           |
+-------------------------------------------------------------------------------------------+
|        Laptop Name     : Macbook Pro 16                                                   |
|        Item Price      : 14000                                                            |
|        Item Stock      : 4                                                                |
+-------------------------------------------------------------------------------------------+
|        Total           : 14000                                                            |

+-------------------------------------------------------------------------------------------+
|        Laptop Name     : Alienware                                                        |
|        Item Price      : 11868                                                            |
|        Item Stock      : 6                                                                |
+-------------------------------------------------------------------------------------------+
|        Total           : 11868                                                            |

+-------------------------------------------------------------------------------------------+
|        Grand Total(Without VAT)    :   22505.16                                           |
|        Grand Total(with VAT)       :   25868                                              |
+-------------------------------------------------------------------------------------------+

Press Enter To Continue
```

Figure 34 bills shown

7.4 Test 4

OBJECTIVE	Sell multiple laptop from the project
ACTION	Show all the steps required to sell one
EXPECTED RESULT	Bill is made successfully
ACTUAL RESULT	Bill is made and shown succesfully
RESULT	Pass

Table 4 Sell multiple laptop

Figure 35 Main menu

Figure 36 Inserting value in sell phase

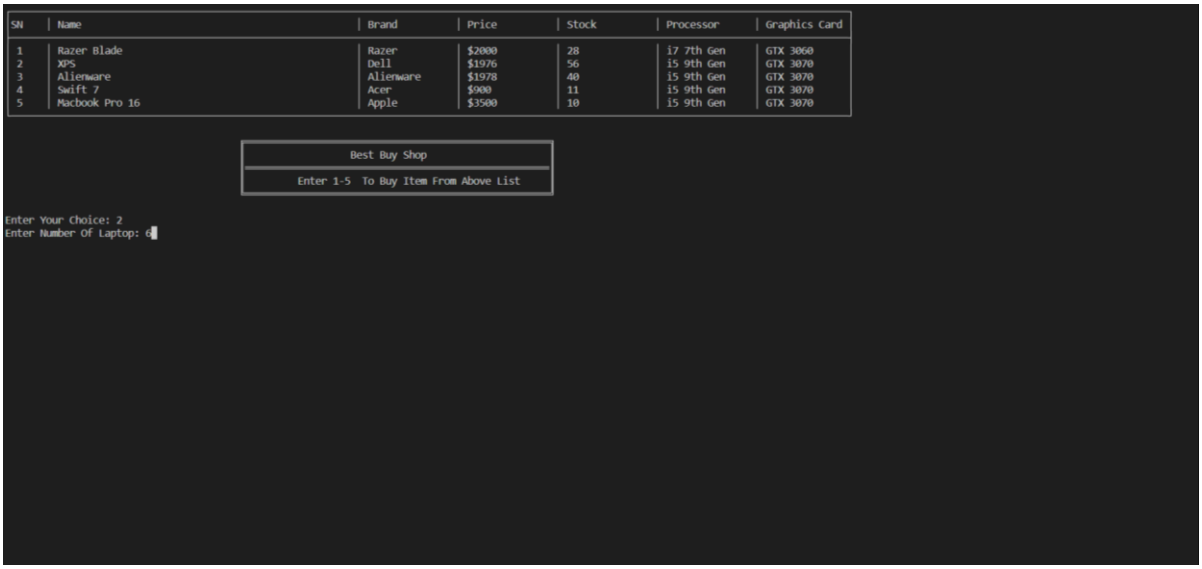
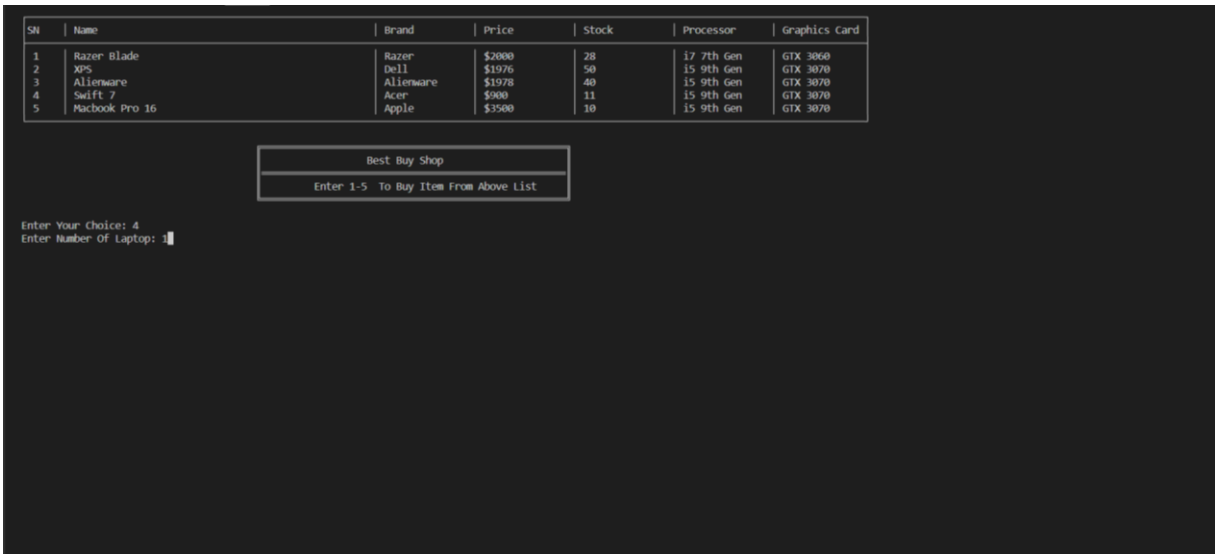


Figure 37 inserting true value in choice of sell phase

Figure 38 choosing to buy more in sell phase



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	28	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	50	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	40	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	10	i5 9th Gen	GTX 3070

```
Best Buy Shop
Enter 1-5  To Buy Item From Above List

Enter Your Choice: 4
Enter Number Of Laptop: 1
```

Figure 39 inserting new sell item in sell

```
1. Buy More
2. Stop

Choose 1 or 2: 2
```

Figure 40 stopping the sell phase

```
+-------------------------------------------------------------------------------------+
|                                   Best Buy Shop                                     |
|                        1234 Main Street, Anytown USA 12345                          |
|                Phone: 555-555-5555 Email: info@bestbuyshop.com                      |
+-------------------------------------------------------------------------------------+
  |                                                   Date    :2023-05-11 21:31:27.669251
|       Customer Name    :   PrabalGurung
|       Customer Address:    Pokhara                                                  |
|       Customer Contact:    9827145355                                               |
+-------------------------------------------------------------------------------------+
|       Laptop Name      :   XPS                                                      |
|       Laptop Brand     :    Dell                                                    |
|       Price            :    $1976                                                   |
|       Quantity         :   6                                                        |
+-------------------------------------------------------------------------------------+
|       Total Amount     :   $11856                                                   |

+-------------------------------------------------------------------------------------+
|       Laptop Name      :   Swift 7                                                  |
|       Laptop Brand     :    Acer                                                    |
|       Price            :    $900                                                    |
|       Quantity         :   1                                                        |
+-------------------------------------------------------------------------------------+
|       Total Amount     :   $900                                                     |

|                                                Shipping Cost   :  1200              |
+-------------------------------------------------------------------------------------+
|       Grand Total      :   13956                                                    |
+-------------------------------------------------------------------------------------+
```

Figure 41 transaction of sell

Figure 42 thank you messsage shown

7.5 Test 5

OBJECTIVE	Show the update in stock of laptop
ACTION	Show the quantity change
EXPECTED RESULT	Stock should change its value
ACTUAL RESULT	Stock value are changed
RESULT	Pass

Table 5 Update in laptop

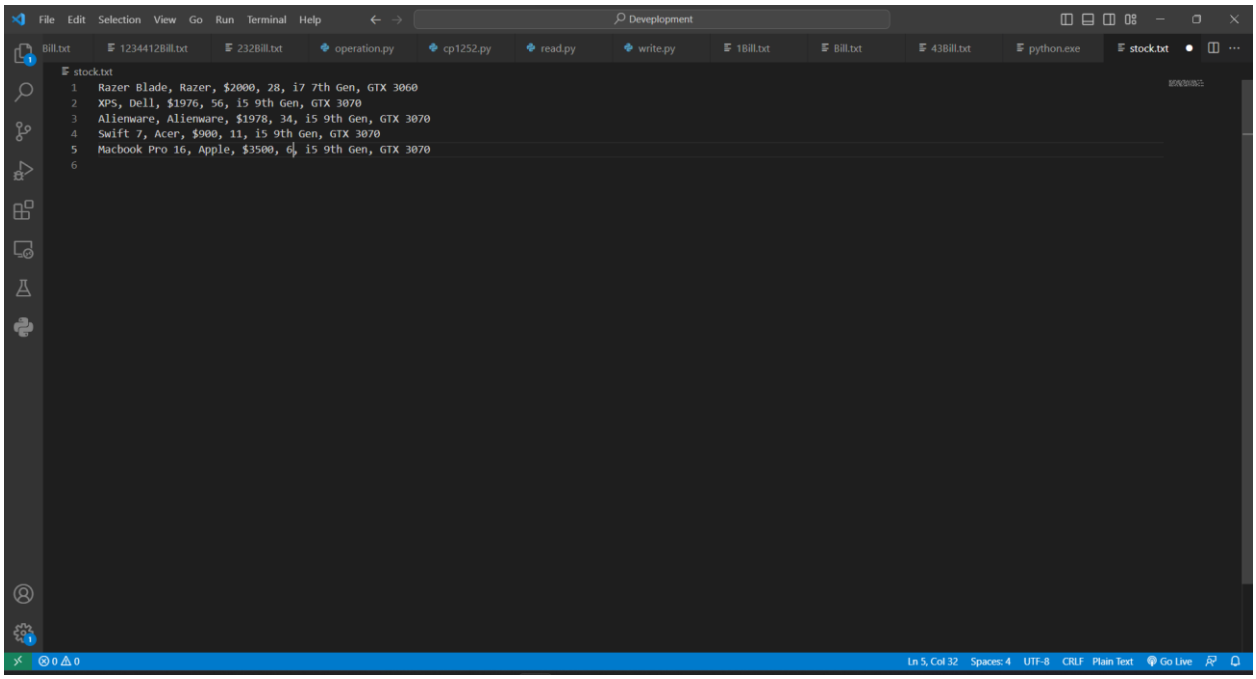
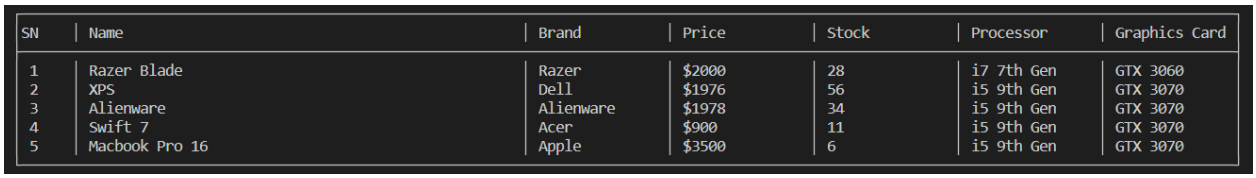


Figure 43 Initial stock in bill



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	28	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	34	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	6	i5 9th Gen	GTX 3070

Figure 44 Initial stock in shell

Initial table before refilling or selling stock in laptop

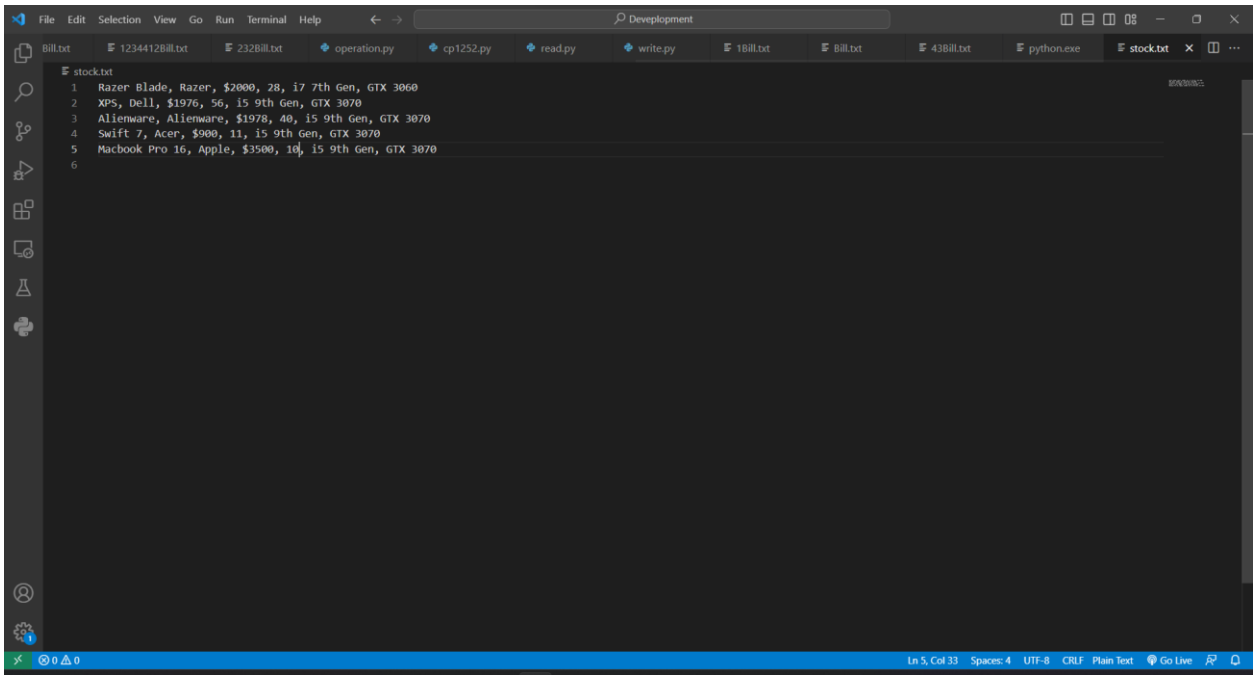


Figure 45 Stock after buy in bill

SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	28	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	40	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	10	i5 9th Gen	GTX 3070

Figure 46 Stock after buy in shell

Stock change after test 3 when table serial number 3 and 5 is added by the amount bought from Vendor

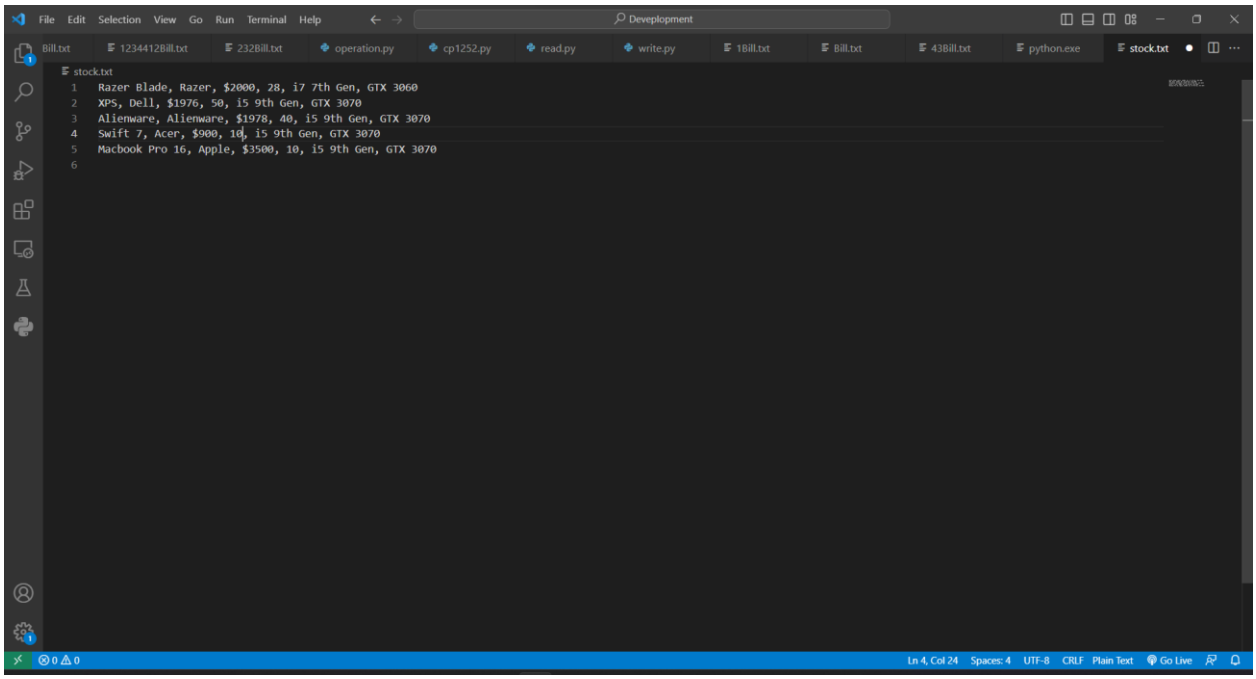
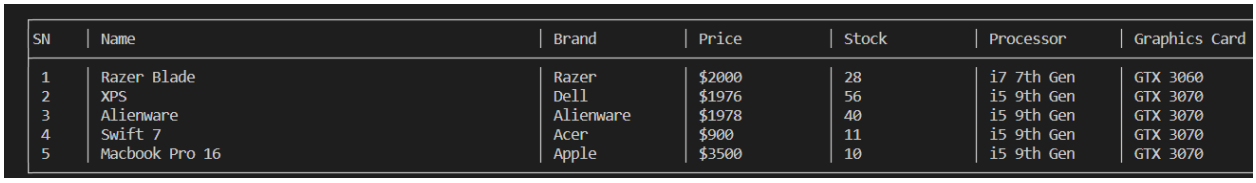


Figure 47 Stock after sell in bill



SN	Name	Brand	Price	Stock	Processor	Graphics Card
1	Razer Blade	Razer	$2000	28	i7 7th Gen	GTX 3060
2	XPS	Dell	$1976	56	i5 9th Gen	GTX 3070
3	Alienware	Alienware	$1978	40	i5 9th Gen	GTX 3070
4	Swift 7	Acer	$900	11	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	$3500	10	i5 9th Gen	GTX 3070

Figure 48 Stock after sell in shell

This is the final table that is shown after test 4 is done and stocks are deducted accoroding to the amount bought.

8. Conclusion

In conclusion, the Python project focused on the operations of a laptop shop, specifically the buying and selling functions. The project successfully implemented modiles for reading and writing data, perfroming necessary validations retreving date and time and clearing the screen. By utilizing various function and loops, the project has provided us unbreakable program which can run no matter the difficulties it is going to face.

During the project there were a lot of difficulties faced but the way project was handeled was a different experience and probably increased skill specially during the logical error faced after try except handling. which made it near impossible to find the where the possible error was made.

Bibilography

Bibliography

ASQ. (2023, 01 12). *asq*. Retrieved from asq.org: https://asq.org/quality-resources/flowchart#:~:text=A%20flowchart%20is%20a%20picture,process%2C%20or%20a%20project%20plan.

Code, V. S. (n.d.). *Documentation for Visual Studio Code*. Retrieved from code.visualstudio.com: https://code.visualstudio.com/docs

Docs, M. W. (n.d.). *JavaScript*. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript

interaction, e. (n.d.). *What is a wireframe?* Retrieved from everyinteraction.com: https://www.everyinteraction.com/definition/wireframes/#:~:text=A%20Wireframe%20is%20a%20visual%20schematic%20that%20conveys,just%20enough%20to%20get%20across%20the%20core%20idea.

Pedamkar, P. (n.d.). *What is CSS3?* Retrieved from educba: https://www.educba.com/what-is-css3/

S, H. S. (2023, February 17). *simplilearn*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/c-tutorial/guide-pseudo-code-in-c#:~:text=DevelopmentExplore%20Program-,What%20Is%20Pseudo%2DCode%20in%20C%3F,cannot%20be%20compiled%20or%20interpreted.

APPENDIX

main.py

```
# Importing necessary modules

import displayOperation
import operation

# loop until break
while True:
    # call main
    displayOperation.cos_Main()
    # ask input
    userInput = input("Choose from (1-4): ")

    # check if input is right
    if(userInput == "1"):
        # display table
        displayOperation.get_stockTable()
        input("Press Enter To Continue!!")
    elif(userInput == "2"):
        # call sell
        operation.sellProcess()
    elif(userInput == "3"):
        # call buy
        operation.buyFromVendor()
    elif(userInput == "4"):
        # exit
        displayOperation.exit()
        break
    else:
        # display warning
        displayOperation.warningMainDisplay()
```

operation.py

```
# Importing necessary modules
import os
import displayOperation
import read
import write

# instance variable
grandTotal = 0
vendorTotal = 0

def buyFromVendor():
    os.system("cls")
    print("Leave name empty to go to Main Screen")
    print("")
    print("")
    # try input if error throws exception
    try:
        vendorName = input("Enter Company Name: ")
    except:
        displayOperation.warningMainDisplay()

    # if condition true proceed else proceed second or third
    if (len(vendorName) > 0):
        # call proceedbuy
        proceedBuy(vendorName)
    elif (len(vendorName) == 0):
        #exit if null
        displayOperation.exit()
        input()
    else:
        # loop if error
        displayOperation.warningMainDisplay()

def proceedBuy(vendorName):
    while True:
        # display all
        displayOperation.get_stockTable()
        displayOperation.getListForSell()
        stockData = read.get_stockData()

        # try all choice and demand
        try:
            userChoice = int(input("Enter Your Choice: "))
            userDemand = int(input("Enter Number Of Laptop: "))

            userChoice = userChoice - 1
```

```
        # if condition true proceed
        if(userChoice >= 0 and userDemand > 0 and userChoice < 5):
            # calculate
            price = stockData[int(userChoice)][2].replace("$","")
            totalAmount = int(price) * int(userDemand)
            global vendorTotal
            vendorTotal += totalAmount

            # refill stock
            write.refillStock(userDemand, userChoice)
            # make bill
            displayOperation.vendorBill(vendorName, userChoice, userDemand, totalAmount)
            # buy again
            buyAgainVendor(vendorName)
            break
        else:
            # show error message
            input("Please Enter Valid Choice")
    except:
        # show error message
        displayOperation.warningMainDisplay()

def buyAgainVendor(vendorName):
    # display
    displayOperation.buyMore()
    # try if error throw exception
    try:
        userSelection = input("Choose 1 or 2: ")
    except:
        displayOperation.warningMainDisplay()
    # if condition true proceed
    if (userSelection == "1"):
        proceedBuy(vendorName)
    # if first condition false make final bill and read and break program
    elif(userSelection == "2"):
        os.system("cls")
        displayOperation.addVendorTotal(vendorName, vendorTotal)
        read.displayBill(vendorName)
        input("Press Enter To Continue")
    else:
        displayOperation.warningMainDisplay()
        input("Press any key to Continue")
        buyAgainVendor(vendorName)

def sellProcess():
    os.system("cls")
    print("Leave a field empty to go to Main Screen")
    print("")
    print("")
    # try input if error throws exception
    try:
        customerName = input("Enter Customer Name: ")
```

```
        customerAddress = input("Enter Customer Address: ")
        customerNumber = input("Enter Customer Number: ")
    except:
        displayOperation.exit()
    # if condition true proceed to next method
    if(len(customerName) > 0 and len(customerAddress) > 0 and len(customerNumber) == 10):
        startSell( customerName, customerAddress, customerNumber)
    elif(len(customerName) == 0 or len(customerAddress) == 0 or len(customerNumber) == 0):
        displayOperation.exit()
        input()
    else:
        displayOperation.warningMainDisplay()

def startSell(customerName, customerAddress, customerNumber):
    while True:
        # display
        displayOperation.get_stockTable()
        displayOperation.getListForSell()
        stockData = read.get_stockData()
        # try input if error throw exception
        try:
            userChoice = input("Enter Your Choice: ")
            userDemand = input("Enter Number Of Laptop: ")

            userChoice = str(int(userChoice) - 1)

            # if both condition true proceed
            if(int(stockData[int(userChoice)][3])) - int(userDemand) >= 0:
                if((int(stockData[int(userChoice)][3])) >= 0 and int(userChoice) >= 0, int(userDemand) > 0):
                    # calculate
                    price = stockData[int(userChoice)][2].replace("$","")
                    totalAmount = int(price) * int(userDemand)
                    global grandTotal
                    grandTotal += totalAmount

                    # make bill, change stock and proceed to ask buy again
                    displayOperation.bill(customerName, customerAddress, customerNumber, userDemand,
userChoice, stockData, totalAmount)
                    write.changeStock(userDemand, userChoice)
                    buyAgain(customerName, customerAddress, customerNumber)
                    break
                else:
                    # display error of condition doesnt match
                    displayOperation.warningMainDisplay()
            else:
                # say out of stock if first condition false
                os.system("cls")
                print("Out Of Stock!! Please Lower Your Demand.")
                input("")
        except:
            # display warning in case of exception
            displayOperation.warningMainDisplay()
```

```
def buyAgain(customerName, customerAddress, customerNumber):
    displayOperation.buyMore()
    # ask for input if error throw exception
    try:
        userSelection = input("Choose 1 or 2: ")
    except:
        displayOperation.warningMainDisplay()
    # if user selection is one repeat process
    if(userSelection == "1"):
        startSell(customerName, customerAddress, customerNumber)
    # else add grand in bill and read bill
    elif(userSelection == "2"):
        os.system("cls")
        displayOperation.addGrandTotal(customerName, grandTotal)
        read.readData(customerName)
        input()
        displayOperation.thankYou()
        input("Press Enter To Continue")
    else:
        # error display
        displayOperation.warningMainDisplay()
        input("Press any key to Continue")
        buyAgain(customerName, customerAddress, customerNumber)

grandTotal = 0
vendorTotal = 0
```

displayOperation.py

```
# Importing necessary modules

import os
import read
import datetime

# This method is main display from where option display is shown.
def cos_Main():
    os.system("cls")
    print("""

    $$$$$$$\                    $$\          $$$$$$$\                      $$$$$$\  $$\
    $$  __$$\                  $$ |         $$  __$$\                    $$  __$$\ $$ |
    $$ |  $$ | $$$$$$\   $$$$$$$\ $$$$$$\        $$ |  $$ |$$\   $$\ $$\   $$\      $$ /  \__|$$$$$$$\   $$$$$$\
$$$$$$\
    $$$$$$$\ |$$  __$$\ $$  _____|\_$$  _|       $$$$$$$\ |$$ |  $$ |$$ |  $$ |     \$$$$$$\  $$  __$$\ $$  __$$\ $$
__$$\
    $$  __$$\ $$$$$$$$ |\$$$$$$\    $$ |         $$  __$$\ $$ |  $$ |$$ |  $$ |      \____$$\ $$ |  $$ |$$ /  $$ |$$ /
$$ |
    $$ |  $$ |$$   ____| \____$$\   $$ |$$\      $$ |  $$ |$$ |  $$ |$$ |  $$ |     $$\   $$ |$$ |  $$ |$$ |  $$ |$$ |
    $$$$$$$  |\$$$$$$$\ $$$$$$$  |  \$$$$  |     $$$$$$$  |\$$$$$$  |\$$$$$$$ |     \$$$$$$  |$$ |  $$ |\$$$$$$
|$$$$$$$  |
    \_______/  \_______|\_______/    \____/      \_______/  \_______/  \____$$ |      \_______/ \__|  \__| \_____/
$$  ____/
                                                 $$\   $$ |                        $$ |
                                                 \$$$$$$  |                         $$ |
                                                  \_____/                           \__|


    """)

    print("""
                        ┌──────────────────────────────────────────────────────────────────────────────┐
                        │      🛍 Welcome to Best Buy Shop 🛍            │
                        ├──────────────────────────────────────────────────────────────────────────────┤
                        │        Select an option to proceed:         │
                        │                                             │
                        │ 🔍  Enter 1 to Show the Table.              │
                        │ 💻  Enter 2 to Sell Laptop.                 │
                        │ 🛒  Enter 3 to Buy From Vendor.             │
                        │ 📱  Enter 4 to Exit the Program.            │
                        └──────────────────────────────────────────────────────────────────────────────┘

    """)

# This method defines table and shows table when it is called.
def get_stockTable():
    os.system("cls")
```

```
    reads = read.get_stockData()
    j = 1

    print('╭'+ "─" * (143) +'╮')
    print('│'+'SN'.ljust(6)+'│','Name'.ljust(50), '│'+' Brand'.ljust(15), '│'+' Price'.ljust(15), '│'+' Stock'.ljust(15), '│'+'
Processor'.ljust(15), '│'+' Graphics Card'.ljust(15)+'│')
    print('├'+ "─" * (143) +'┤')

    for row in reads:
        print('│', str(j).ljust(4) ,'│', row[0].ljust(50), '│'+row[1].ljust(15), '│'+row[2].ljust(15), '│'+row[3].ljust(15),
'│'+row[4].ljust(15), '│'+row[5].ljust(15)+'│')
        j = j + 1

    print('└'+ "─" * (143) +'┘')

# This method defines a choice when people tries to buy
def getListForSell():
    print("""

╔═══════════════════════════════════════════════════════════════════════════╗
                              ‖                Best Buy Shop                ‖

‖═══════════════════════════════════════════════════════════════════════════‖
                              ‖       Enter 1-5  To Buy Item From Above List    ‖

╚═══════════════════════════════════════════════════════════════════════════╝
    """)

# displays exit when called
def exit():
    print("""
```

```
THANK YOU
```

```
    """)

def warningMainDisplay():
    os.system("cls")
    print("""
```

```
                _____ _                _______     _       __     __   _ _     _   _____                _
                | __ \| |              |  ____|    | |        \ \   / / | (_)   | | |_   _|             | |
                | |__) | | ___  __ _ ___  ___  | |__   _ __ | |_ ___ _ __   \ \ / /_ _| |_  __| | |  | | _ __  _ __  _   _| |_
                |  ___/| |/ _ \/ _` / __|/ _ \ |  __| | '_ \| __/ _ \ '__|   \ V / _` | | |/ _` |  | | | '_ \| '_ \| | | | __|
                | |    | |  __/ (_| \__ \  __/ | |____| | | | ||  __/ |      \  / (_| | | | (_| |  _| |_| | | | |_) | |_| | |_
                |_|    |_|\___|\__,_|___/\___| |_______|_| |_|\__\___|_|       \/ \__,_|_|_|\__,_| |_____|_| |_| .__/
\__,_|\__|
                                                                                                                | |
                                                                                                                |_|

        """)
    input("                                    Press Enter To Continue!!")
```

```
# display thank you when called
def thankYou():
    print("""
                                _______ _          _   __    __       ______        ____          _
                               |__   __| |        | |   \ \   / /      |  ____|      | _ \        (_)
                                 | | | |__   __ _ _ __ | | __  \ \_/ /__  _   _  | |__ ___   _ __  | |_) |_   _ _   _ _ _ __   __ _
                                 | | | '_ \ / _` | '_ \| |/ /   \   / _ \| | | |  __/ _ \| '__| | _ <| | | | | | | '_ \ / _` |
                                 | | | | | | (_| | | | |   <    | | (_) | |_| | | | | (_) | |    | |_) | |_| | |_| | | | | | (_| |
                                 |_| |_| |_|\__,_|_| |_|_|\_\   |_|\___/ \__,_| |_|  \___/|_|    |____/ \__,_|\__, |_|_| |_|\__, |
                                                                                                              __/ |       __/ |
                                                                                                             |___/        |___/

        """)

# display buy more when called
def buyMore():
    print(f"""
                                1. Buy More
                                2. Stop
""")

# display bill when called
def bill(customerName, customerAddress, customerNumber, userDemand, userChoice, stockData,
totalAmount):
    # checks if file exist or not
    if(os.path.exists(customerName + "Bill.txt")):
        file = open(customerName + "Bill.txt", "a")
        file.write(f"""
+-----------------------------------------------------------------------------------------------+
|       Laptop Name     :   {stockData[int(userChoice)][0]:<20}                                  |
|       Laptop Brand    :   {stockData[int(userChoice)][1]:<10}                                  |
|       Price           :   {stockData[int(userChoice)][2]:<10}                                  |
|       Quantity        :   {userDemand:<10}                                                     |
+-----------------------------------------------------------------------------------------------+
|       Total Amount    :   ${totalAmount:<10}                                                   |
                    """)
        file.close
    else:
        file = open(customerName + "Bill.txt", "w")
        file.write(f"""
+-----------------------------------------------------------------------------------------------+
|                                Best Buy Shop                                                  |
|                      1234 Main Street, Anytown USA 12345                                      |
|                Phone: 555-555-5555 Email: info@bestbuyshop.com                                |
+-----------------------------------------------------------------------------------------------+
    |                                           Date    :{datetime.datetime.now()}                |
|       Customer Name   :   {customerName:<10}                                                   |
|       Customer Address:   {customerAddress:<10}                                                |
|       Customer Contact:   {customerNumber:<10}                                                 |
+-----------------------------------------------------------------------------------------------+
|       Laptop Name     :   {stockData[int(userChoice)][0]:<20}                                  |
```

```
|       Laptop Brand    :   {stockData[int(userChoice)][1]:10}                                    |
|       Price           :   {stockData[int(userChoice)][2]:<10}                                   |
|       Quantity        :   {userDemand:<10}                                                      |
+-----------------------------------------------------------------------------------------------+
|       Total Amount    :   ${totalAmount:<10}                                                    |
                    """)
        file.close

# add grand when called
def addGrandTotal(customerName, grandTotal):
    file = open(customerName + "Bill.txt" , "a")
    file.write(f"""
|                                           Shipping Cost   :  1200              |
+-----------------------------------------------------------------------------------------------+
|       Grand Total     :   {grandTotal + 1200:<10}                                               |
+-----------------------------------------------------------------------------------------------+
        """)
    file.close

#display bill when called
def vendorBill(vendorName, userChoice, userDemand, totalAmount):
    data = read.get_stockData()
    # checks if file exist or not
    if(os.path.exists(vendorName + "_Bill.txt")):
        data = read.get_stockData()
        file = open(vendorName + "_Bill.txt", "a")
        file.write(f"""
+-----------------------------------------------------------------------------------------------+
|       Laptop Name     : {data[int(userChoice)][0]:<20}                                          |
|       Item Price      : {totalAmount:<10}                                                       |
|       Item Stock      : {str(userDemand):<10}                                                   |
+-----------------------------------------------------------------------------------------------+
|       Total           : {totalAmount:<10}                                                       |
                    """)
        file.close
    else:
        file = open(vendorName + "_Bill.txt", "w")
        file.write(f"""
+-----------------------------------------------------------------------------------------------+
|                                Best Buy Shop                                                  |
|                     1234 Main Street, Anytown USA 12345                                       |
|               Phone: 555-555-5555 Email: info@bestbuyshop.com                                 |
+-----------------------------------------------------------------------------------------------+
|                                           Date    : {datetime.datetime.now()}                   |
+-----------------------------------------------------------------------------------------------+
|       Company Name    : {vendorName:<10}                                                        |
+-----------------------------------------------------------------------------------------------+
|       Laptop Name     : {data[int(userChoice)][0]:<20}                                          |
|       Item Price      : {totalAmount:<10}                                                       |
|       Item Stock      : {userDemand:<10}                                                        |
+-----------------------------------------------------------------------------------------------+
|       Total           : {totalAmount:<10}                                                       |
```

```
                    """)
        file.close

#add on vendor
def addVendorTotal(vendorName, vendorTotal):

    withoutVAT = vendorTotal - (vendorTotal * (13/100))

    file = open(vendorName + "_Bill.txt" , "a")
    file.write(f"""
+-----------------------------------------------------------------------------------------------+
|        Grand Total(Without VAT)    :   {withoutVAT:<10}                                        |
|        Grand Total(With VAT)       :   {vendorTotal:<10}                                       |
+-----------------------------------------------------------------------------------------------+
        """)
    file.close
```

write.py

```
# Importing necessary modules
import read

# increase demand accorodingly
def refillStock(userDemand, userChoice):
    stockData = read.get_stockData()
    stockData[int(userChoice)][3] = " " +str(int(stockData[int(userChoice)][3]) + int(userDemand))
    write_stockData_to_file(stockData)

# decrease demand accorodingly
def changeStock(userDemand, userChoice):
    stockData = read.get_stockData()
    stockData[int(userChoice)][3] = " " +str(int(stockData[int(userChoice)][3]) - int(userDemand))
    write_stockData_to_file(stockData)

# write changes in stock.txt
def write_stockData_to_file(stockData):
    stockData_str = ""
    for sublist in stockData:
        line = ""
        for item in sublist:
            line += str(item) + ","
        line = line.rstrip(",") + "\n"
        stockData_str += line
    file = open("stock.txt", "w")
    file.write(stockData_str)
    file.close
```

read.py

```
# set stock data from stock.txt
def get_stockData():
    data = []
    file = open('stock.txt','r')
    lines = file.readlines()
    for i in range (len(lines)):
        data.append(lines[i].strip('\n').split(','))
    file.close
    return data

# read bill and show for buy
def displayBill(vendorName):
    bill = open(vendorName + "_Bill.txt", "r")
    print(bill.read())
    bill.close()

#  read bill and show for sell
def readData(customerName):
    bill = open(customerName + "Bill.txt", "r")
    print(bill.read())
    bill.close
```