LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4051NI Fundamentals of Computing

60% Individual Coursework

2023 Spring

Student Name: Sumit Pandit

London Met ID: 22067569

College ID: NP01CP4A22069

Assignment Due Date: Friday, July 28, 2023

Assignment Submission Date: Thursday, July 27, 2023

Word Count: 242

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

Table of Contents:

1. Introduction:

1.1 Introduction to Python

Python, the brainchild of Guido van Rossum, emerged onto the programming scene in 1991, quickly gaining widespread popularity and becoming a cornerstone of the coding world. Renowned for its versatility, Python serves a multitude of purposes, from system scripting to software development, web applications (server-side), and even handling complex arithmetic operations

As the most favored high-level programming language today, Python offers a seamless blend of procedural and object-oriented programming paradigms. One of its standout features is its concise syntax, allowing developers to achieve more with less code compared to languages like Java. Furthermore, Python's unique indentation requirement ensures that code remains highly readable, leading to cleaner and more maintainable projects.

The influence of Python can be felt across the tech industry, with giants like Google, Amazon, Facebook, Instagram, Dropbox, Uber, and numerous others entrusting their critical projects to this versatile language. Its widespread adoption stems not only from its ease of use but also its vast ecosystem of libraries and frameworks, enabling developers to accomplish tasks with remarkable efficiency and speed. With Python, innovation knows no bounds.

1.2 Introduction to Project

The project aims to develop an efficient laptop shop management system using Python, empowering the computer store to effectively handle their laptop inventory. This system will facilitate seamless buying and selling operations, enabling the store to purchase laptops from manufacturers and offer them to customers. The core functionalities include parsing and updating data from a text file containing laptop information. The system will be equipped to modify the file upon sales or orders, generating detailed receipts or invoices for each transaction. Throughout the development, a strong emphasis will be placed on implementing various data structures and operations in Python to optimize the system's performance.

To ensure its effectiveness and reliability, rigorous testing will be conducted, encompassing scenarios to validate the system's capabilities. Error messages will be intelligently designed to guide users whenever they enter invalid data. Ease of use is a key consideration, with a user-friendly interface enabling the smooth entry of customer details by the administrator. Comprehensive documentation, including algorithms, flowcharts, pseudocode, and utilized data structures, will be provided to enhance clarity and understanding.

Through this project, valuable insights will be gained through research and development, culminating in a well-rounded conclusion that highlights the system's successes and potential areas for future improvement.

2. Algorithm:

Step 1: Start

Step 2: Display welcome message.

Step 3: Ask if the user wants to purchase the laptops, sell the laptops, or exit the system.

Step 4: If purchase, go to step 7- 10.

Step 5: If sell, go to step 11- 15.

Step 6: If exit, go to step 16.

Step 7: Display the details of the laptops.

Step 8: Ask the user to enter the laptop id and no of units. If not valid, display appropriate message and repeat step 8.

Step 9: Make the calculations and generate the invoice and update the text file accordingly.

Step 10: Ask if the user want to purchase laptop again. If yes, repeat step 3. If no, exit the system.

Step 11: Display the details of the laptops.

Step 12: Ask the user to enter the name of costumer, mobile no. of customer, laptop id and no. of units. If not valid, display appropriate and repeat step 12.

Step 13: Ask the user’s choice for shipping.

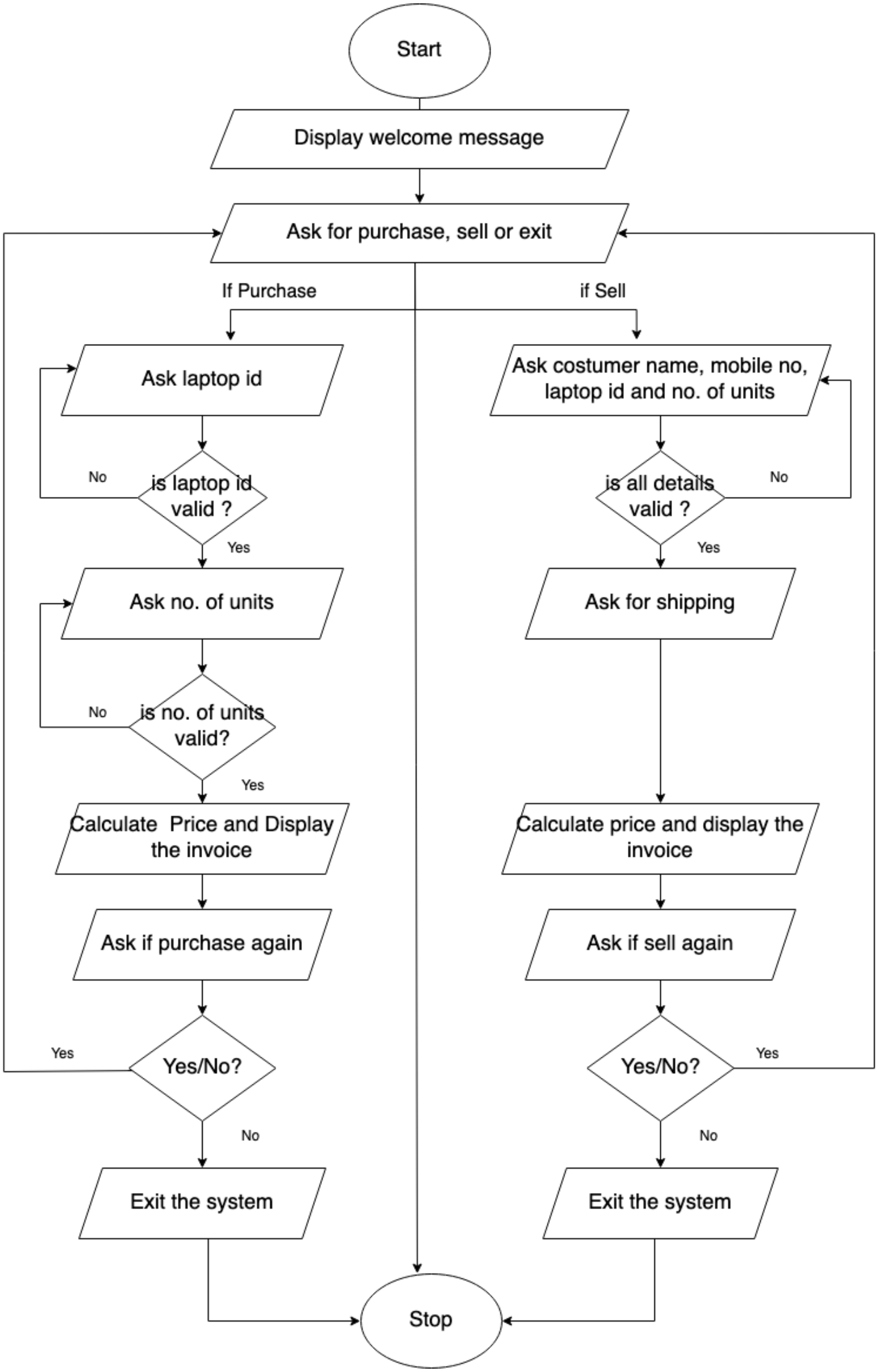
Step 14: Calculate the price and prepare invoice for the user and update the text file accordingly.

+Step 15: Ask if the user want to sell laptop again. If yes, repeat step 3. If no, exit the system.

Step 16: Exit the program.

Step 17: Stop

3.Flow Chart:


Start
Display welcome message
Ask for purchase, sell or exit
If Purchase
if Sell
Ask laptop id
is laptop id valid ?
No
Yes
Ask no. of units
is no. of units valid?
No
Yes
Calculate Price and Display the invoice
Ask if purchase again
Yes/No?
Yes
No
Exit the system
Ask costumer name, mobile no, laptop id and no. of units
is all details valid ?
No
Yes
Ask for shipping
Calculate price and display the invoice
Ask if sell again
Yes/No?
Yes
No
Exit the system
Stop

4. Pseudocode:

Pseudocode serves as a comprehensive and clear explanation of the tasks an algorithm needs to accomplish, presented in a structured and understandable manner using regular English rather than a specific programming language. It plays a crucial role in the development process of a program by providing a detailed description of the concept, enabling programmers and designers to define the logic in a precise manner. This well-defined template then serves as a foundation for writing actual code in a chosen programming language during the subsequent development stages. Overall, pseudocode acts as a bridge between abstract ideas and concrete implementation, facilitating the creation of efficient and effective programs.

```
algorithm read_laptop_file()
    OPEN "laptop.txt" file for reading
    CREATE an empty dictionary laptop_dictionary


    FOR each line in the file
        INCREMENT index by 1
        SPLIT the line by ',' and STRIP any leading/trailing whitespaces
        ADD the line data as a list to laptop_dictionary with key=index
    END FOR


    CLOSE the file
    RETURN laptop_dictionary
end algorithm


algorithm datetime()
```

```
    IMPORT datetime library

    GET the current date and time in the format "YYYY-MM-DD HH:MM"

    RETURN the formatted date and time

end algorithm


algorithm purchase_laptop(choice)

    IF choice is "1" THEN

        DISPLAY "Laptop Details:"

        CALL read_laptop_file() to read laptop data into laptop_dictionary


        WHILE True

            TRY

                GET purchase_id as an integer input

                IF purchase_id is in laptop_dictionary THEN

                    CALL purchase_update(purchase_id, purchase_quantity)

                    BREAK

                ELSE

                    DISPLAY "Please provide a valid laptop ID!"

                END IF

            EXCEPT ValueError

                DISPLAY "Invalid input. Enter a valid input."

            END TRY

        END WHILE
```

```
      WHILE True
         TRY
            GET purchase_quantity as an integer input
            IF purchase_quantity < 0 THEN
               DISPLAY "Please provide a non-negative Quantity!"
            ELSE IF purchase_quantity <= laptop_dictionary[purchase_id][3] THEN
               CALL purchase_update(purchase_id, purchase_quantity)
               BREAK
            ELSE
               DISPLAY "Please provide a valid Quantity!"
            END IF
         EXCEPT ValueError
            DISPLAY "Invalid input. Enter a valid input."
         END TRY
      END WHILE
   ELSE
      DISPLAY "Invalid input. Try again!"
   END IF
end algorithm


algorithm sell_laptop(choice)
   IF choice is "2" THEN
```

```
DISPLAY "Laptop Details:"
CALL read_laptop_file() to read laptop data into laptop_dictionary

WHILE True
    TRY
        GET customer_name as input
        GET customer_number as an integer input
        GET sell_id as an integer input

        IF sell_id is in laptop_dictionary THEN
            GET sell_quantity as an integer input

            IF sell_quantity < 0 THEN
                DISPLAY "Please provide a non-negative Quantity"
            ELSE IF sell_quantity <= laptop_dictionary[sell_id][3] THEN
                CALL sell_update(sell_id, sell_quantity, customer_name, customer_number)
                BREAK
            ELSE
                DISPLAY "The quantity you are looking for is not available at the moment."
            END IF
        ELSE
            DISPLAY "Please provide a valid laptop ID!"
```

```
                END IF
            EXCEPT ValueError
                DISPLAY "Invalid input. Enter a valid input."
            END TRY
        END WHILE
    ELSE
        DISPLAY "Invalid input. Try again!"
    END IF
end algorithm

algorithm main()
    DISPLAY "Welcome to Gada Electronics"
    WHILE True
        DISPLAY "Select an option to continue:"
        DISPLAY "1. Purchase from manufacturer"
        DISPLAY "2. Sell to customer"
        DISPLAY "3. Exit"

        GET choice as input

        IF choice is "1" THEN
            CALL purchase_laptop(choice)
            DISPLAY "The laptop has been purchased."
            BREAK
        ELSE IF choice is "2" THEN
            CALL sell_laptop(choice)
            DISPLAY "The laptop has been sold."
            BREAK
        ELSE IF choice is "3" THEN
            DISPLAY "Thank you for using the system."
            RETURN
        ELSE
            DISPLAY "Invalid choice"
        END IF
    END WHILE

    WHILE True
        GET re_choice as input
        IF re_choice is "y" THEN
            CALL main()
        ELSE IF re_choice is "n" THEN
            DISPLAY "Thank you for using the System. Have a good day!"
            BREAK
        ELSE
            DISPLAY "Invalid input. Please enter 'y' or 'n'"
        END IF
    END WHILE
end algorithm

# Call the main function to start the program
CALL main()

end algorithm
```

5. Data Structures:

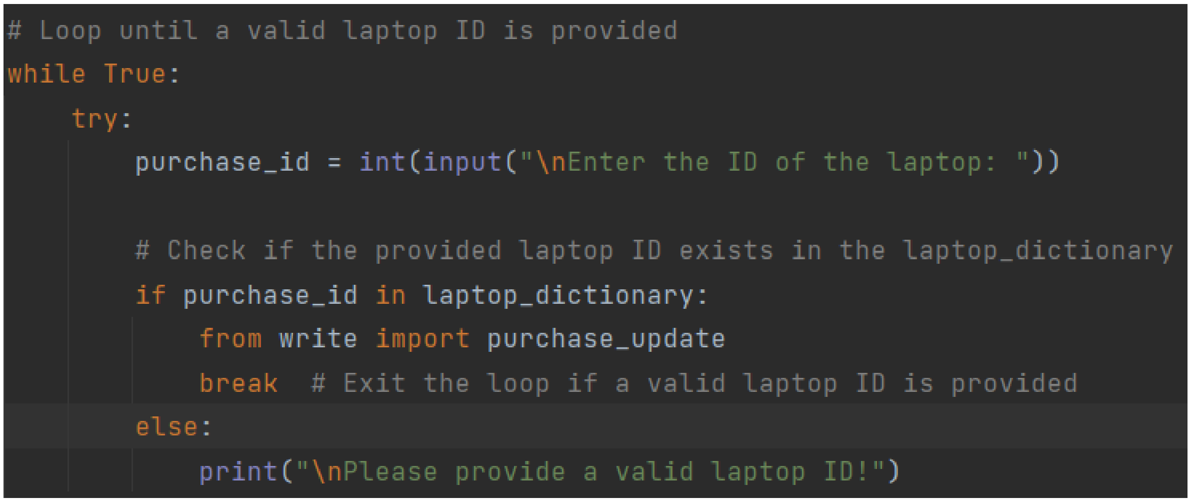
Data structures play a pivotal role in Python programming as they serve as essential tools for efficiently storing and organizing data in a manageable manner. These structures act as containers capable of holding diverse values, including numbers, text, or even other data structures. Python offers eight major data structures that are widely used: Integer, String, Boolean, Float, List, Dictionary, Tuples, and Set.

By gaining a comprehensive understanding of these data structures and leveraging them effectively, developers can write code that is cleaner, more efficient, and well-organized. These data structures empower programmers to perform various operations on data, such as sorting, searching, filtering, and more, simplifying the programming process and enhancing code manageability. Ultimately, mastering data structures in Python equips programmers with valuable tools to handle data in a more streamlined and effective manner.

5.1 Int:

In Python, the int data type serves as a means to store whole numbers, capable of representing integers of any magnitude. Python's dynamic nature automatically optimizes the internal representation of the integer based on its value, allowing for efficient memory usage. To convert a string containing numerical characters into an integer, Python provides the int() function. This function takes a string input as its argument and returns an integer value that corresponds to the same numerical value as the input string. A common use case for the int() function is when a Python program requires user input as a string, and the program needs to convert it into an integer for further calculations or processing.

For example, suppose we have a variable named "num" that holds the string "32". By calling int(num), Python will convert this string into an integer with a value of 32, facilitating mathematical operations and manipulations with the converted integer value.



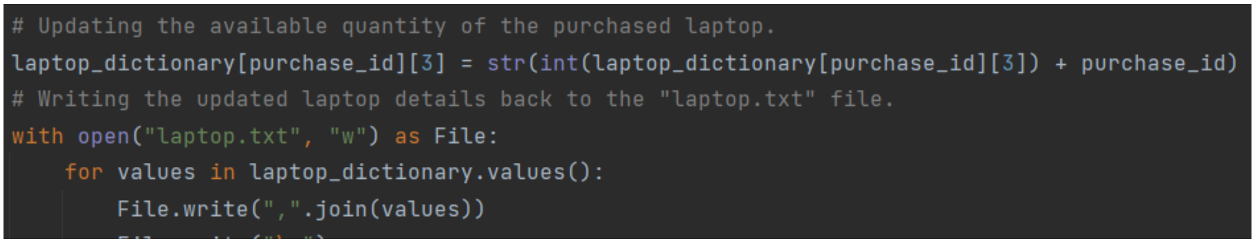
```python
# Loop until a valid laptop ID is provided
while True:
    try:
        purchase_id = int(input("\nEnter the ID of the laptop: "))

        # Check if the provided laptop ID exists in the laptop_dictionary
        if purchase_id in laptop_dictionary:
            from write import purchase_update
            break  # Exit the loop if a valid laptop ID is provided
        else:
            print("\nPlease provide a valid laptop ID!")
```

Fig: Use of Int in program

5.2 String

In Python, a string is a variable type specifically designed for storing text values. Being immutable, once created, the content of a string cannot be altered. Strings are encapsulated within either single quotes ('example') or double quotes ("example"). To construct new strings by merging existing ones, the plus (+) operator can be utilized, a process commonly referred to as string concatenation. Accessing individual characters within a string can be achieved using square brackets alongside the index of the desired character.

Should the need arise to convert other data types into strings, the str() function is readily available for this purpose. In summary, strings represent a foundational data type in Python, extensively employed for the storage and manipulation of textual data.



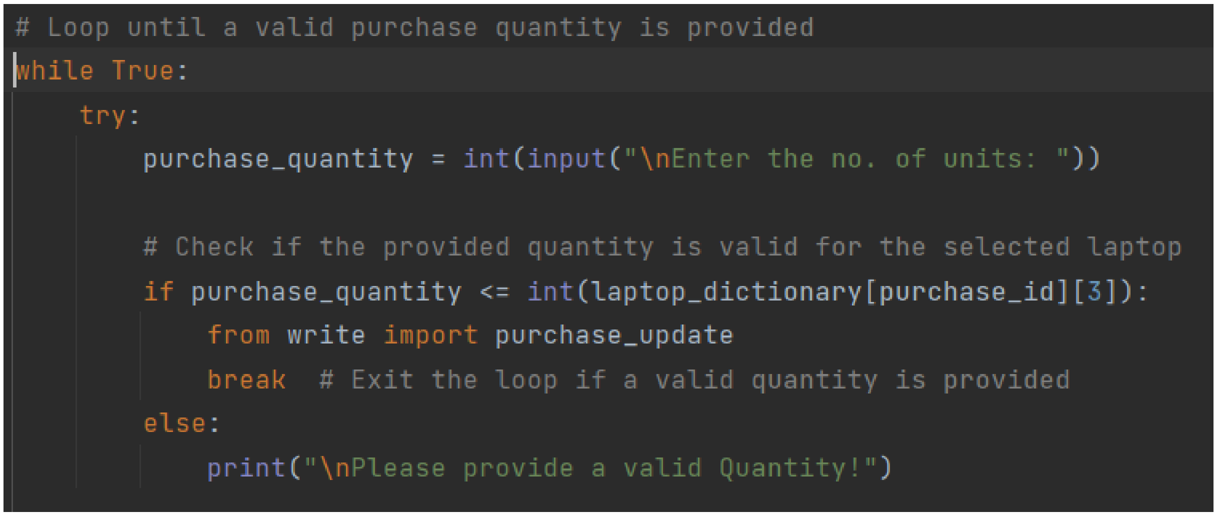
```
# Updating the available quantity of the purchased laptop.
laptop_dictionary[purchase_id][3] = str(int(laptop_dictionary[purchase_id][3]) + purchase_id)
# Writing the updated laptop details back to the "laptop.txt" file.
with open("laptop.txt", "w") as File:
    for values in laptop_dictionary.values():
        File.write(",".join(values))
```

Fig: use of string in program

5.3 Boolean

In Python, a Boolean is a fundamental data type that holds one of two possible values: True or False. Booleans serve as representations of true and false concepts in programming, akin to the numeric values 1 and 0. They play a crucial role in logical operations and control flow statements, enabling programmers to make decisions based on conditions that evaluate to either true or false.

Essentially, Booleans form the building blocks of complex programs, as they enable the creation of intricate logic and decision-making processes that determine the flow and behavior of the code based on the true or false outcomes of various conditions.



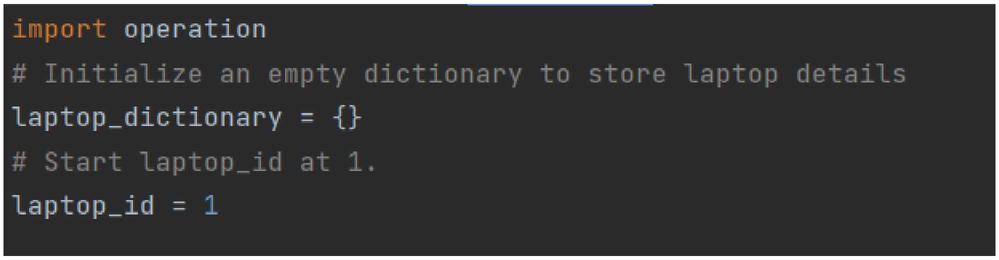
```
# Loop until a valid purchase quantity is provided
while True:
    try:
        purchase_quantity = int(input("\nEnter the no. of units: "))

        # Check if the provided quantity is valid for the selected laptop
        if purchase_quantity <= int(laptop_dictionary[purchase_id][3]):
            from write import purchase_update
            break  # Exit the loop if a valid quantity is provided
        else:
            print("\nPlease provide a valid Quantity!")
```

Fig: Use of Boolean in program

5.4 Dictionary

In Python, a dictionary is a data structure that holds a collection of key-value pairs, ensuring that each key is unique. It offers an alternative approach to store data compared to lists, where we typically need to keep track of indices. Each element within a dictionary comprises two parts: a key and its corresponding value. Dictionaries are mutable, allowing for easy addition or removal of items as needed. As they are unordered collections, the concept of order or position does not apply to dictionaries. Instead, items can be accessed using their respective keys. To denote dictionaries, curly braces are used.

Overall, dictionaries serve as a valuable tool for organizing data in Python, providing a convenient and efficient means to retrieve values based on their associated keys. This makes dictionaries a powerful choice for various applications where fast and accurate data access is crucial.



```
import operation
# Initialize an empty dictionary to store laptop details
laptop_dictionary = {}
# Start laptop_id at 1.
laptop_id = 1
```

Fig: Use of dictionary in program

5.5 Float

In Python, a float is a numerical data type that represents numbers with decimal points. It is specifically designed for storing decimal values and is considered a primitive data type. Unlike integers, floats require more memory space since computers need to convert the decimal part of the number into binary format for storage and processing.

5.6 List

In Python, lists are versatile data structures used to store and manipulate collections of items. They offer the flexibility to hold elements of different types and can be easily modified. To create a list, items are enclosed within square brackets [].

Lists can be extensively manipulated, providing functionalities such as adding or removing items, accessing specific elements by their index, and sorting the list. Due to their dynamic nature and usefulness in various programming tasks, lists hold significant importance in Python and are widely utilized throughout the language.

5.7 Tuples:

Tuples in Python share similarities with lists; however, once created, they remain unchangeable. They are particularly valuable when there is a need to store data that should not be modified, such as constants. Tuples are enclosed within parentheses and can accommodate elements of any data type. While it is not possible to append elements to tuples, it is feasible to swap their existing elements. Notably, a tuple must consist of at least two elements to be considered a tuple.

5.8 Sets:

Sets in Python bear resemblance to mathematical sets. They serve as collections of unique elements, characterized by being unordered and mutable. Sets are especially advantageous when it comes to swiftly checking for the existence of a specific element, as they possess an optimized method for this purpose. However, it's important to note that sets can only store immutable types. In Python, sets are denoted by curly braces, and their elements are separated by commas. It is crucial to differentiate sets from dictionaries, as dictionaries store key-value pairs, whereas sets solely focus on unique elements.

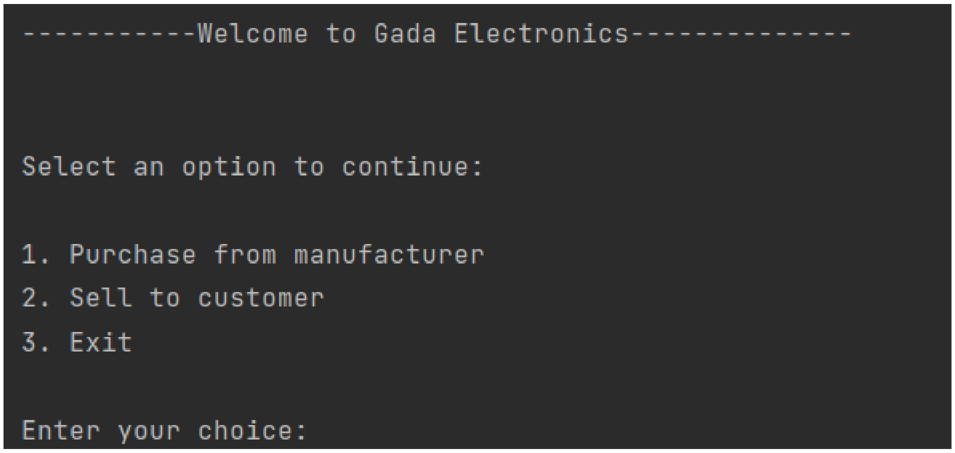
6. Program:

6.1 Implementation of the program for purchasing laptops

The application commences by presenting users with a comprehensive list of available laptops, showcasing essential details such as brand, model, specifications, and corresponding prices. Users are offered three choices: purchasing a laptop from the manufacturer, selling an existing laptop, or exiting the program.

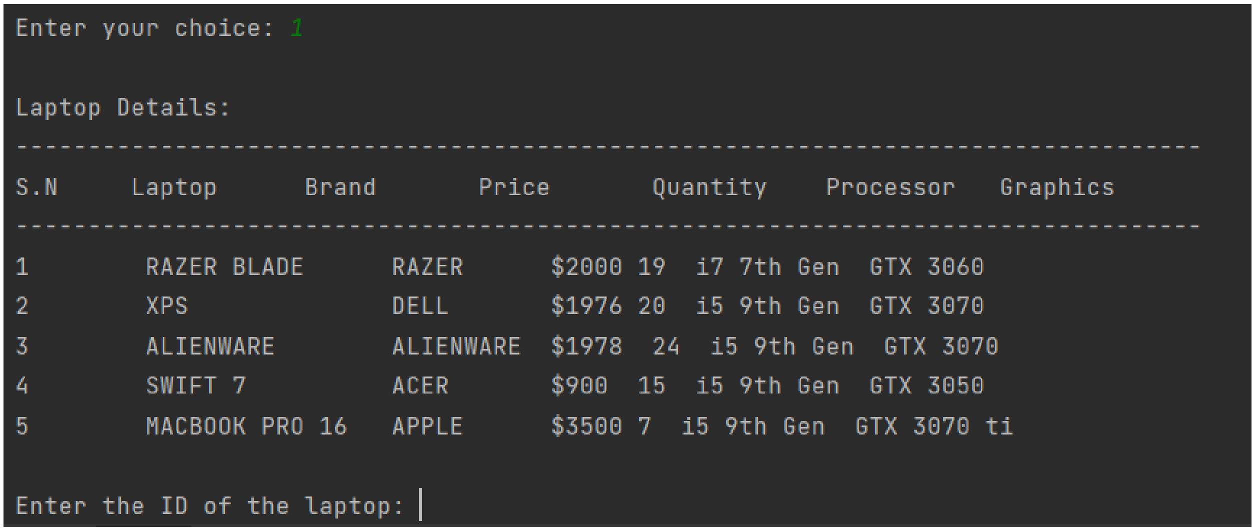
If the user opts to buy a laptop, the application guides them through the process of selecting the desired model and specifying the quantity they wish to purchase. Following this, the program updates the inventory by adding the purchased quantity to the existing stock. Upon the completion of the transaction, a distinctive bill is generated, encompassing all pertinent information, such as the laptop's name, brand, customer name, date and time of purchase, net amount, 13% VAT, and gross amount. Furthermore, if desired, the purchased laptop can be added to the user's list of owned laptops. The program accommodates customers who make multiple laptop purchases within a single transaction, efficiently printing all the acquired laptops on a consolidated bill for enhanced convenience.

In addition to purchasing laptops, the program offers a selling feature, enabling users to input the brand, model, and condition of the laptop they wish to sell. Subsequently, a bill is generated, displaying crucial details of the sold laptop, including the selling price and date and time of the sale. Overall, this application ensures a seamless and user-friendly experience for managing laptop sales and purchases, empowering users to make informed decisions and effortlessly navigate through their transactions.



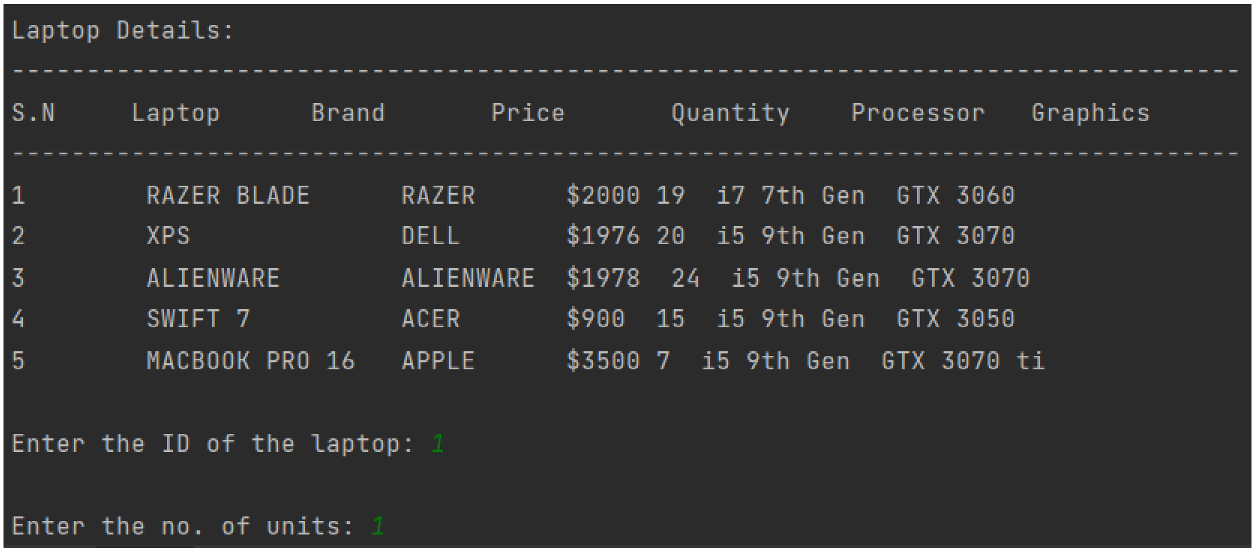
```
-----------Welcome to Gada Electronics--------------



Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice:
```

Fig: Program User Interface



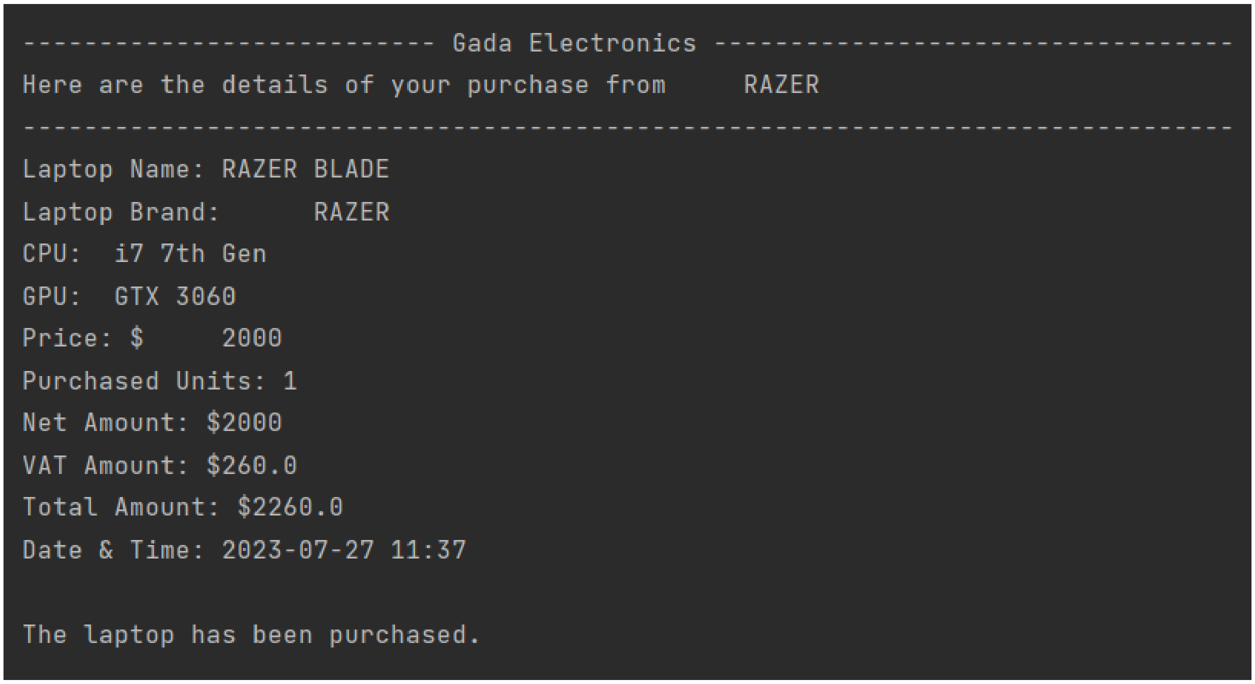
```
Enter your choice: 1

Laptop Details:
----------------------------------------------------------------------------------
S.N      Laptop       Brand        Price        Quantity     Processor    Graphics
----------------------------------------------------------------------------------
1         RAZER BLADE      RAZER      $2000 19   i7 7th Gen   GTX 3060
2         XPS              DELL       $1976 20   i5 9th Gen   GTX 3070
3         ALIENWARE        ALIENWARE  $1978  24   i5 9th Gen   GTX 3070
4         SWIFT 7          ACER       $900  15   i5 9th Gen   GTX 3050
5         MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen   GTX 3070 ti

Enter the ID of the laptop: 
```

Fig: When user enters 1



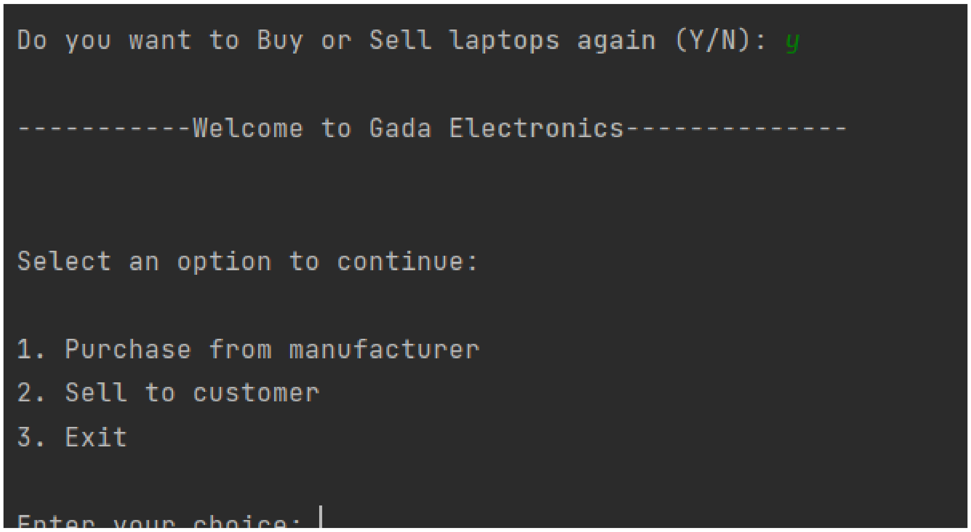
```
Laptop Details:
----------------------------------------------------------------------------------
S.N      Laptop       Brand        Price        Quantity     Processor    Graphics
----------------------------------------------------------------------------------
1         RAZER BLADE      RAZER      $2000 19   i7 7th Gen   GTX 3060
2         XPS              DELL       $1976 20   i5 9th Gen   GTX 3070
3         ALIENWARE        ALIENWARE  $1978  24   i5 9th Gen   GTX 3070
4         SWIFT 7          ACER       $900  15   i5 9th Gen   GTX 3050
5         MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen   GTX 3070 ti

Enter the ID of the laptop: 1

Enter the no. of units: 1
```

Fig: Laptop id and no. of units validity check



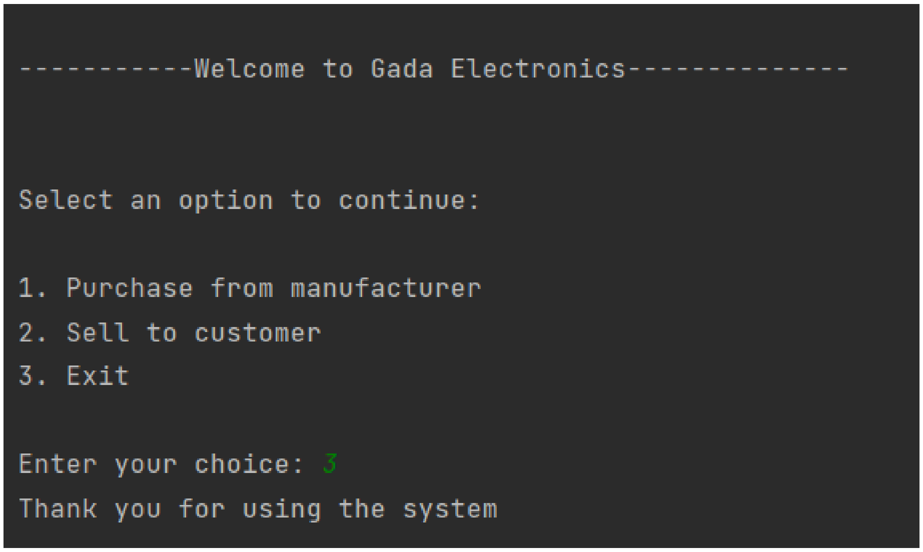
```
--------------------------- Gada Electronics ----------------------------------
Here are the details of your purchase from     RAZER
-------------------------------------------------------------------------------
Laptop Name: RAZER BLADE
Laptop Brand:      RAZER
CPU:  i7 7th Gen
GPU:  GTX 3060
Price: $     2000
Purchased Units: 1
Net Amount: $2000
VAT Amount: $260.0
Total Amount: $2260.0
Date & Time: 2023-07-27 11:37

The laptop has been purchased.
```

Fig: Laptop purchase bill generation



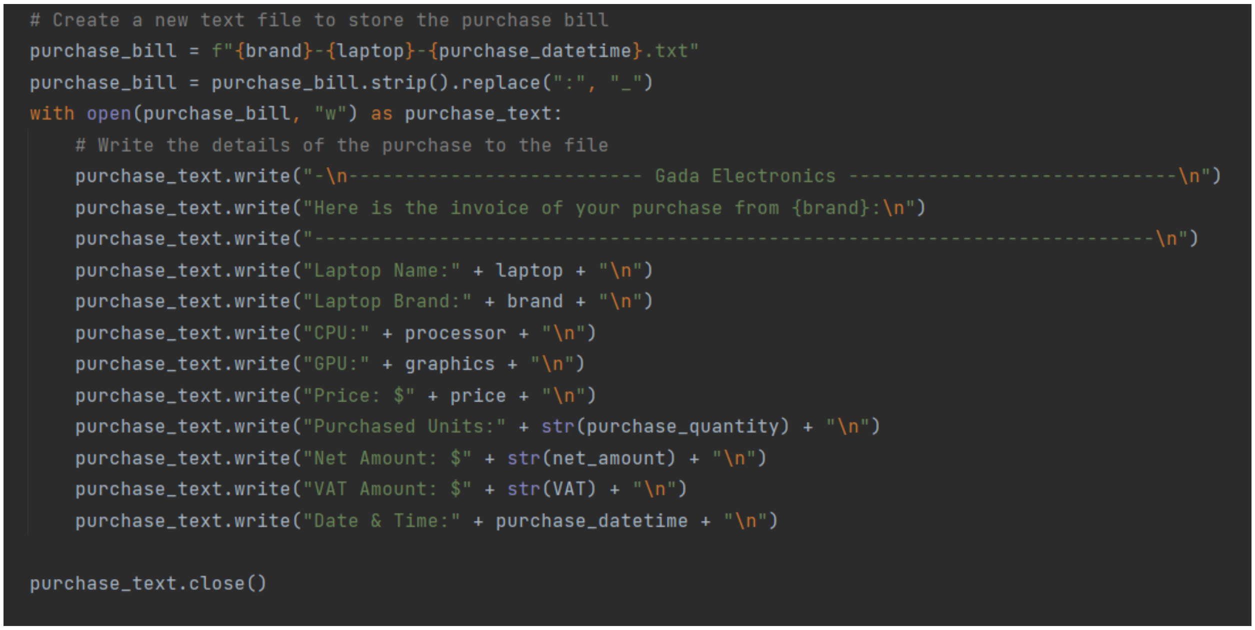
```
Do you want to Buy or Sell laptops again (Y/N): y

-----------Welcome to Gada Electronics--------------


Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice:
```

Fig: Multiple purchase



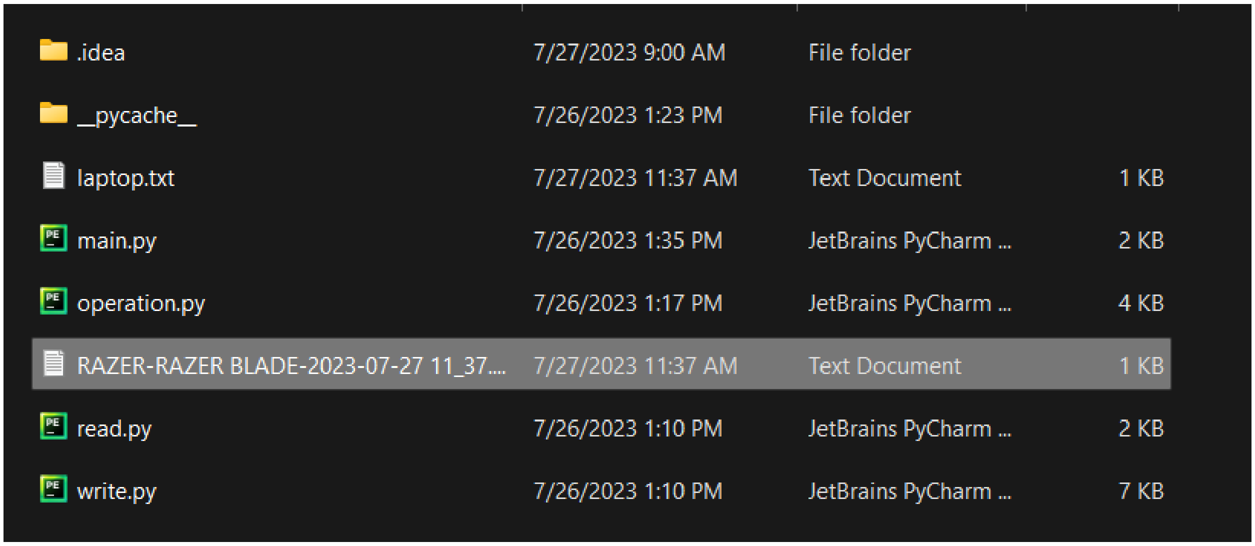
```
-----------Welcome to Gada Electronics--------------



Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice: 3
Thank you for using the system
```

Fig: Termination of program



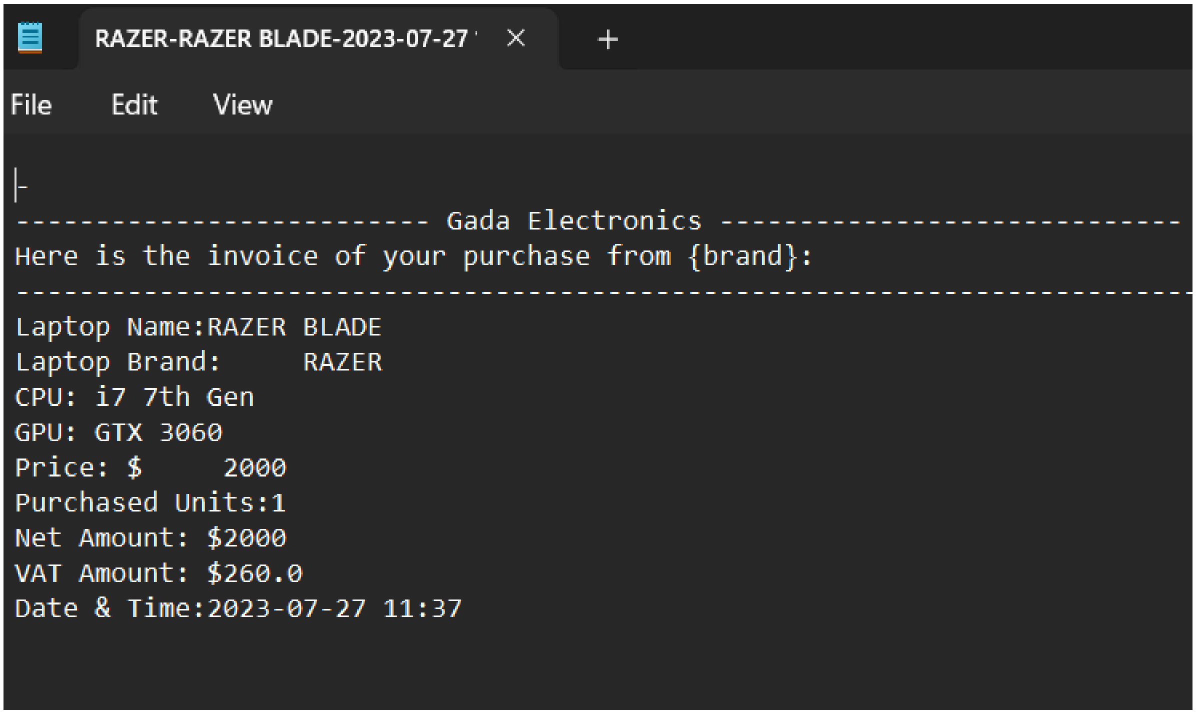
```python
# Create a new text file to store the purchase bill
purchase_bill = f"{brand}-{laptop}-{purchase_datetime}.txt"
purchase_bill = purchase_bill.strip().replace(":", "_")
with open(purchase_bill, "w") as purchase_text:
    # Write the details of the purchase to the file
    purchase_text.write("-\n-------------------------- Gada Electronics -----------------------------\n")
    purchase_text.write("Here is the invoice of your purchase from {brand}:\n")
    purchase_text.write("--------------------------------------------------------------------------\n")
    purchase_text.write("Laptop Name:" + laptop + "\n")
    purchase_text.write("Laptop Brand:" + brand + "\n")
    purchase_text.write("CPU:" + processor + "\n")
    purchase_text.write("GPU:" + graphics + "\n")
    purchase_text.write("Price: $" + price + "\n")
    purchase_text.write("Purchased Units:" + str(purchase_quantity) + "\n")
    purchase_text.write("Net Amount: $" + str(net_amount) + "\n")
    purchase_text.write("VAT Amount: $" + str(VAT) + "\n")
    purchase_text.write("Date & Time:" + purchase_datetime + "\n")

purchase_text.close()
```

Fig; Creation of text file



.idea	7/27/2023 9:00 AM	File folder	
__pycache__	7/26/2023 1:23 PM	File folder	
laptop.txt	7/27/2023 11:37 AM	Text Document	1 KB
main.py	7/26/2023 1:35 PM	JetBrains PyCharm ...	2 KB
operation.py	7/26/2023 1:17 PM	JetBrains PyCharm ...	4 KB
RAZER-RAZER BLADE-2023-07-27 11_37....	7/27/2023 11:37 AM	Text Document	1 KB
read.py	7/26/2023 1:10 PM	JetBrains PyCharm ...	2 KB
write.py	7/26/2023 1:10 PM	JetBrains PyCharm ...	7 KB

Fig: Created text file in folder



```
-
------------------------- Gada Electronics -----------------------------
Here is the invoice of your purchase from {brand}:
-------------------------------------------------------------------------
Laptop Name:RAZER BLADE
Laptop Brand:     RAZER
CPU: i7 7th Gen
GPU: GTX 3060
Price: $     2000
Purchased Units:1
Net Amount: $2000
VAT Amount: $260.0
Date & Time:2023-07-27 11:37
```

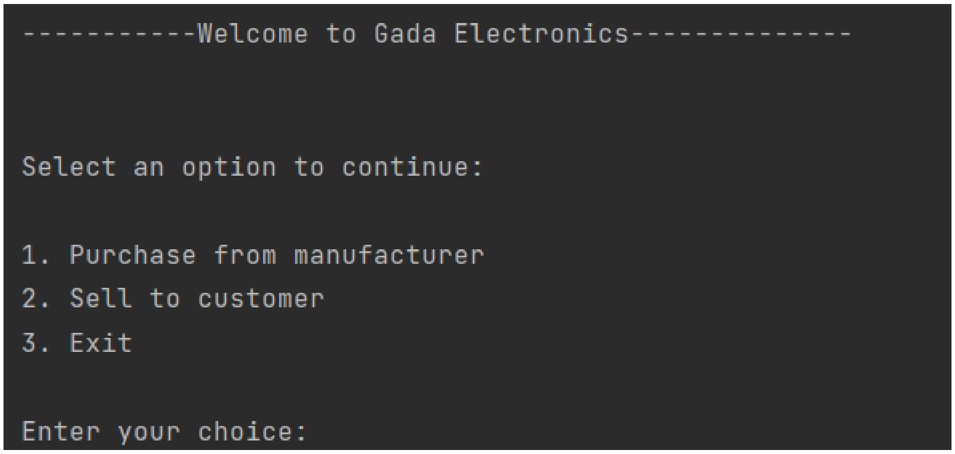
Fig: Purchase bill displayed in text file

6.2 Implementation of the program for selling laptops

At the outset, the program showcases an extensive list of available laptops, offering users the flexibility to choose between purchasing, selling, or exiting the system. Should the user opt to sell a laptop to a customer, the program provides step-by-step guidance, prompting the selection of the laptop's brand, model, and condition. Subsequently, the inventory is updated accordingly, reducing the stock, and a customized bill is generated, encompassing vital information such as the laptop's name, brand, customer name, date and time of sale, net amount, gross amount, and shipping cost (if applicable). If the customer elects to have the purchased laptop(s) shipped, the program efficiently calculates the shipping cost based on the destination and incorporates it into the total amount due. The generated bill comprehensively includes all acquired laptops, along with the shipping cost and other relevant details.

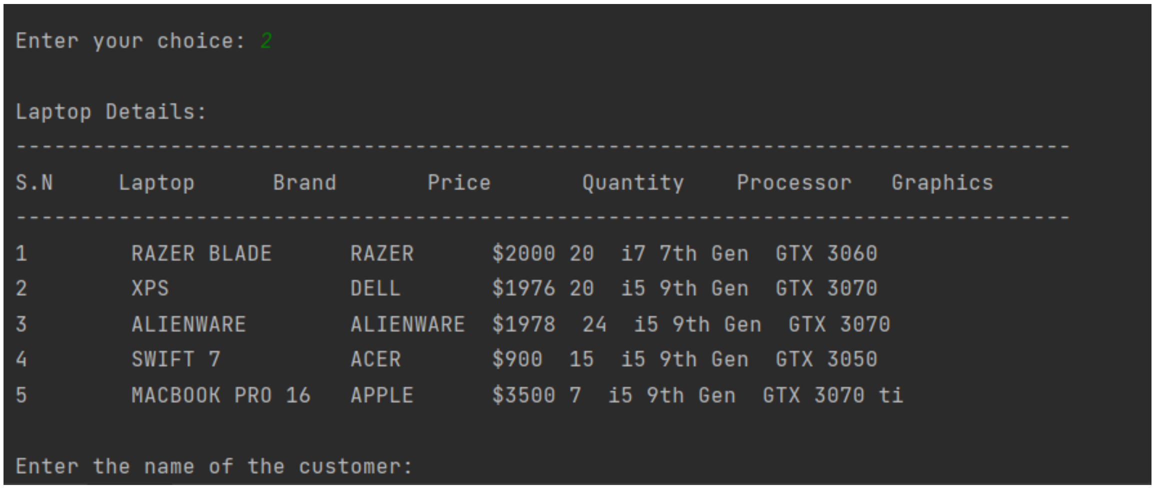
Notably, for customers making multiple laptop purchases in a single transaction, the program streamlines the billing process by consolidating all the purchased laptops onto a single bill, enhancing customer convenience.

Emphasis is placed on ensuring that each generated bill is unique and comprehensive, providing a seamless and user-friendly experience for both the user and the customer to efficiently track and manage transactions. With a robust system in place, the program facilitates smooth and reliable laptop sales and purchases, catering to the diverse needs of both buyers and sellers.



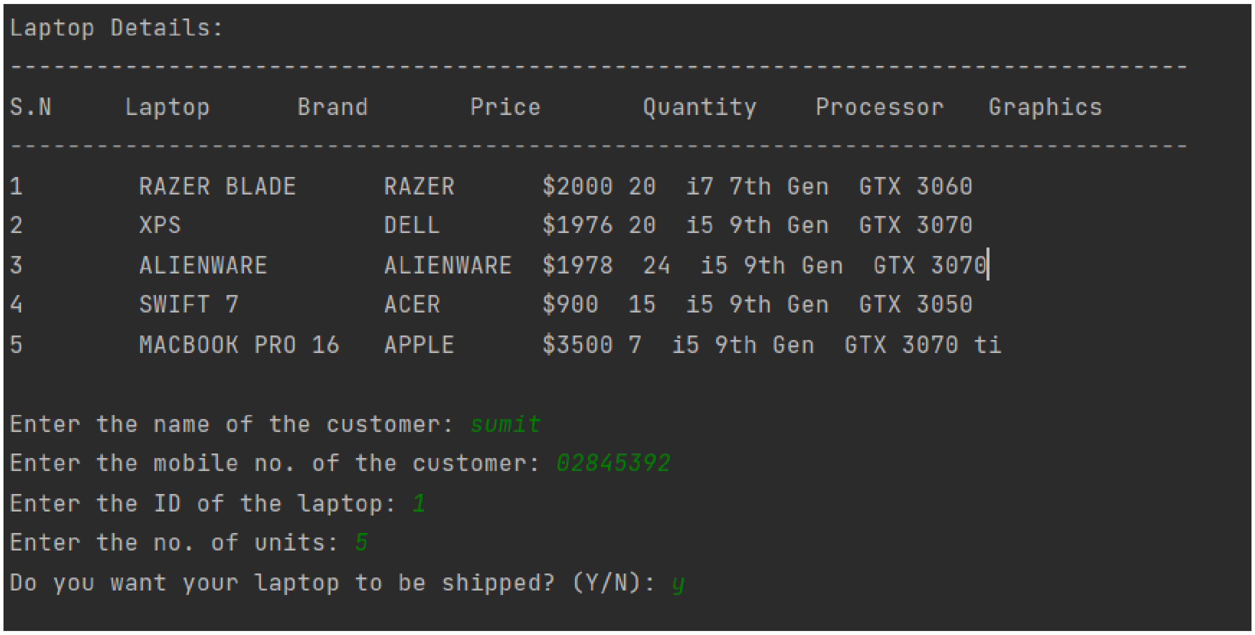
```
-----------Welcome to Gada Electronics--------------


Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice:
```

Fig: program user interface



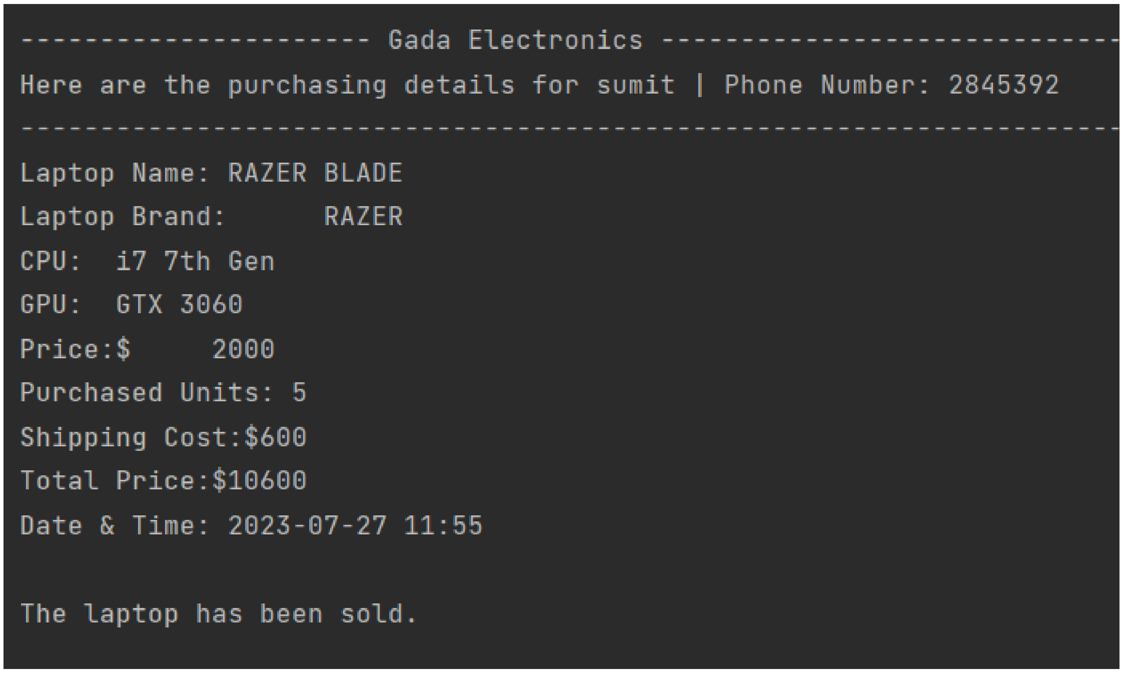
```
Enter your choice: 2

Laptop Details:
----------------------------------------------------------------------------------
S.N     Laptop      Brand       Price       Quantity    Processor   Graphics
----------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 20  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978  24  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen  GTX 3070 ti

Enter the name of the customer:
```

Fig: when user enters 2



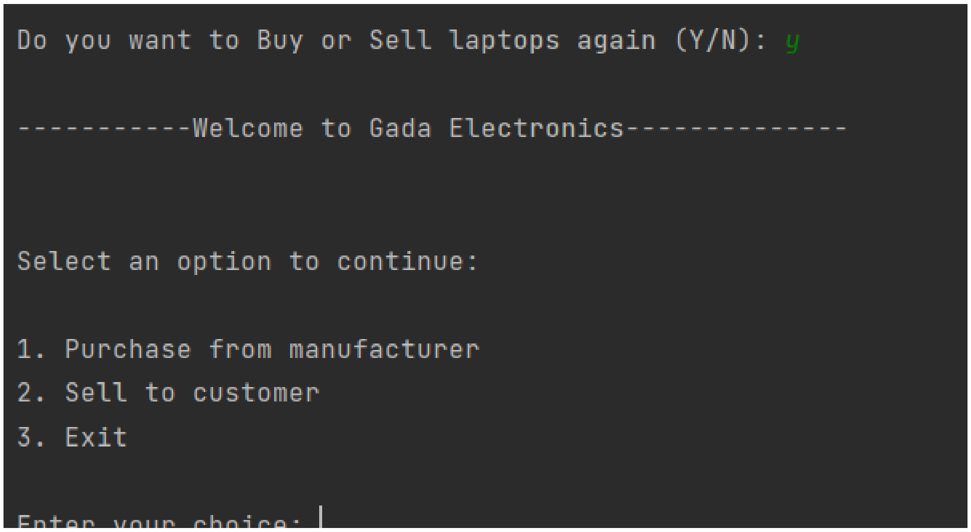
```
Laptop Details:
----------------------------------------------------------------------------------
S.N     Laptop      Brand       Price       Quantity    Processor    Graphics
----------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 20  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978  24  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen  GTX 3070 ti

Enter the name of the customer: sumit
Enter the mobile no. of the customer: 02845392
Enter the ID of the laptop: 1
Enter the no. of units: 5
Do you want your laptop to be shipped? (Y/N): y
```

Fig: customer name, number, laptop id and units validity check



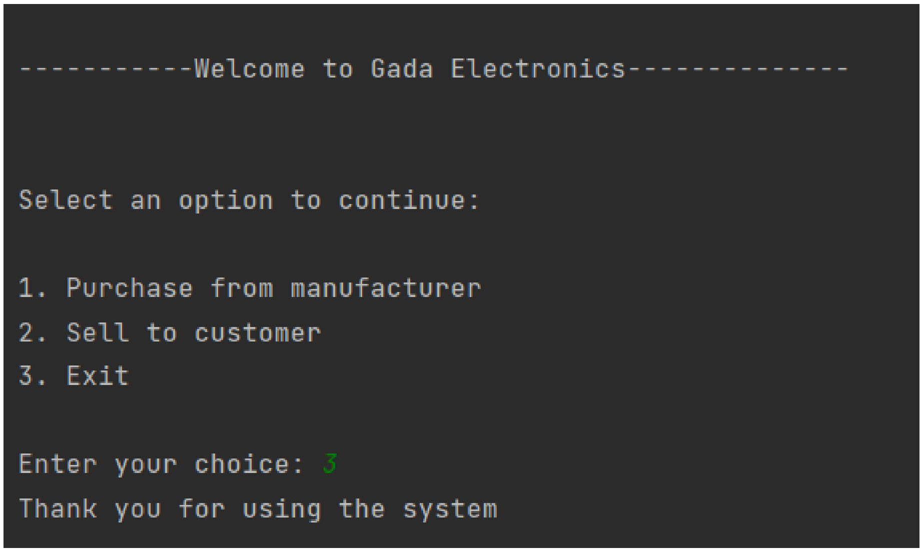
```
---------------------- Gada Electronics ----------------------------
Here are the purchasing details for sumit | Phone Number: 2845392
--------------------------------------------------------------------
Laptop Name: RAZER BLADE
Laptop Brand:      RAZER
CPU:  i7 7th Gen
GPU:  GTX 3060
Price:$     2000
Purchased Units: 5
Shipping Cost:$600
Total Price:$10600
Date & Time: 2023-07-27 11:55

The laptop has been sold.
```

Fig: sell bill generation



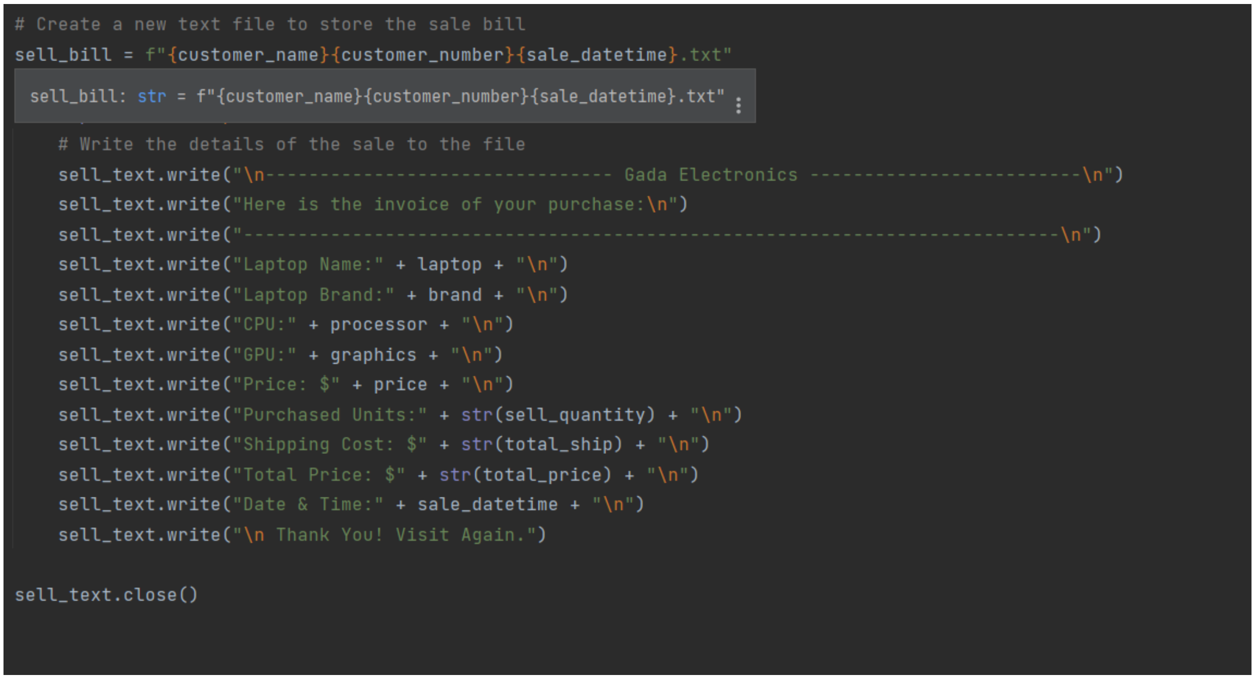
```
Do you want to Buy or Sell laptops again (Y/N): y

-----------Welcome to Gada Electronics--------------



Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice:
```

Fig: Multiple sell



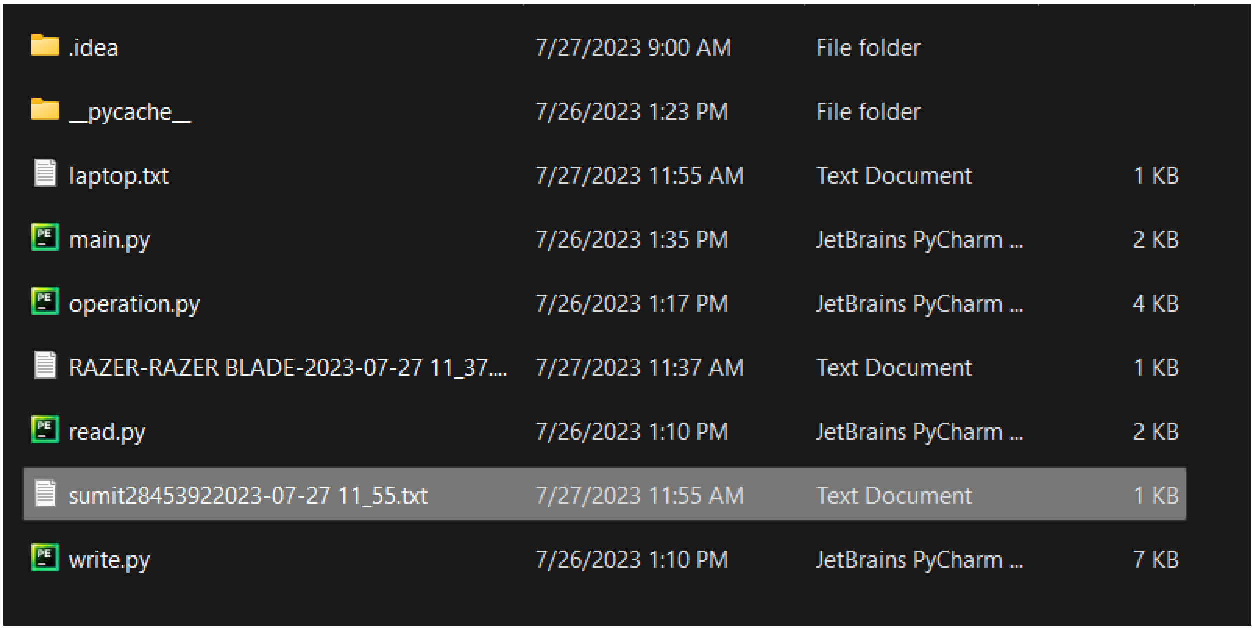
```
-----------Welcome to Gada Electronics--------------



Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice: 3
Thank you for using the system
```

Fig: Termination of program



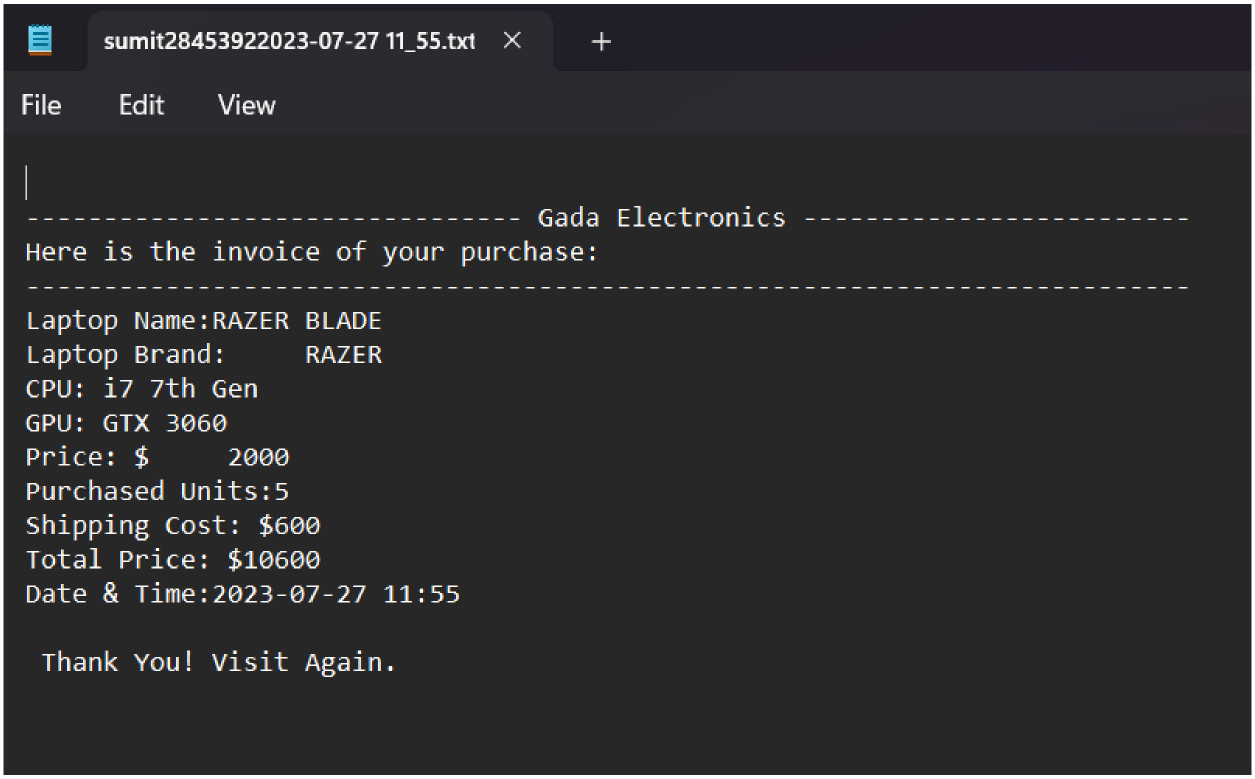
```
# Create a new text file to store the sale bill
sell_bill = f"{customer_name}{customer_number}{sale_datetime}.txt"
 sell_bill: str = f"{customer_name}{customer_number}{sale_datetime}.txt"
    # Write the details of the sale to the file
    sell_text.write("\n-------------------------------- Gada Electronics -------------------------\n")
    sell_text.write("Here is the invoice of your purchase:\n")
    sell_text.write("---------------------------------------------------------------------------\n")
    sell_text.write("Laptop Name:" + laptop + "\n")
    sell_text.write("Laptop Brand:" + brand + "\n")
    sell_text.write("CPU:" + processor + "\n")
    sell_text.write("GPU:" + graphics + "\n")
    sell_text.write("Price: $" + price + "\n")
    sell_text.write("Purchased Units:" + str(sell_quantity) + "\n")
    sell_text.write("Shipping Cost: $" + str(total_ship) + "\n")
    sell_text.write("Total Price: $" + str(total_price) + "\n")
    sell_text.write("Date & Time:" + sale_datetime + "\n")
    sell_text.write("\n Thank You! Visit Again.")

sell_text.close()
```

Fig: Creation of text file



.idea	7/27/2023 9:00 AM	File folder	
__pycache__	7/26/2023 1:23 PM	File folder	
laptop.txt	7/27/2023 11:55 AM	Text Document	1 KB
main.py	7/26/2023 1:35 PM	JetBrains PyCharm ...	2 KB
operation.py	7/26/2023 1:17 PM	JetBrains PyCharm ...	4 KB
RAZER-RAZER BLADE-2023-07-27 11_37....	7/27/2023 11:37 AM	Text Document	1 KB
read.py	7/26/2023 1:10 PM	JetBrains PyCharm ...	2 KB
sumit28453922023-07-27 11_55.txt	7/27/2023 11:55 AM	Text Document	1 KB
write.py	7/26/2023 1:10 PM	JetBrains PyCharm ...	7 KB

Fig: Created text file in folder



```
-------------------------------- Gada Electronics --------------------------
Here is the invoice of your purchase:
----------------------------------------------------------------------------
Laptop Name:RAZER BLADE
Laptop Brand:     RAZER
CPU: i7 7th Gen
GPU: GTX 3060
Price: $     2000
Purchased Units:5
Shipping Cost: $600
Total Price: $10600
Date & Time:2023-07-27 11:55

 Thank You! Visit Again.
```

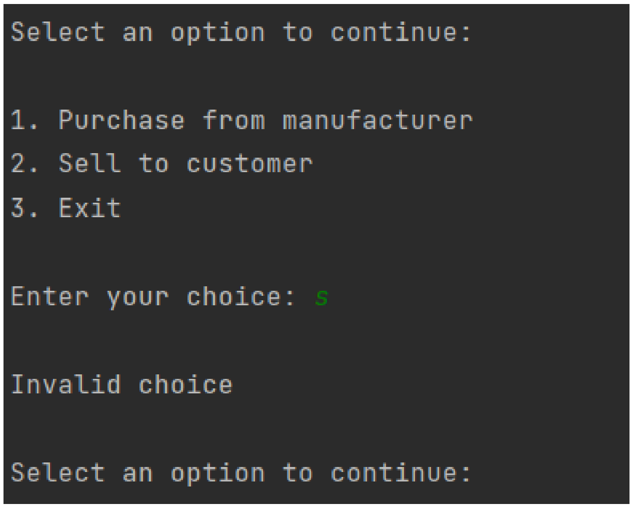
Fig: Sell bill displayed in text file

7. Testing:

7.1 Test 1

Objective	To show the implementation of try, except.
Action	Enter invalid input.
Expected Result	The system will display invalid input and ask to input again.

Actual Result	The system displayed invalid input and asked to input again.
Conclusion	Testing was successful.

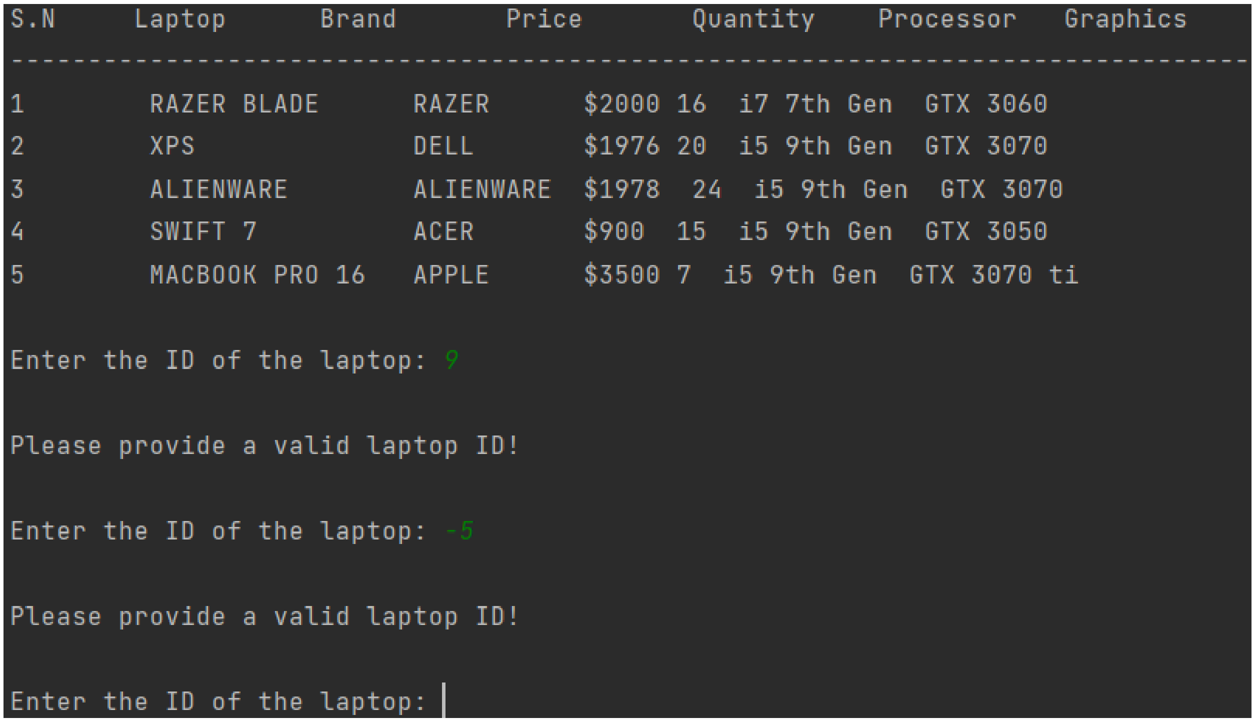


```
Select an option to continue:

1. Purchase from manufacturer
2. Sell to customer
3. Exit

Enter your choice: s

Invalid choice

Select an option to continue:
```

Fig: Implementation of try and except

7.2 Test 2

Objective	To show invalid message when wrong ID is entered
Action	• Non-existed costume ID was entered. • Negative Id was entered.
Expected Result	The program should show an invalid message
Actual Result	The program showed an invalid message.
Conclusion	The test was successful.



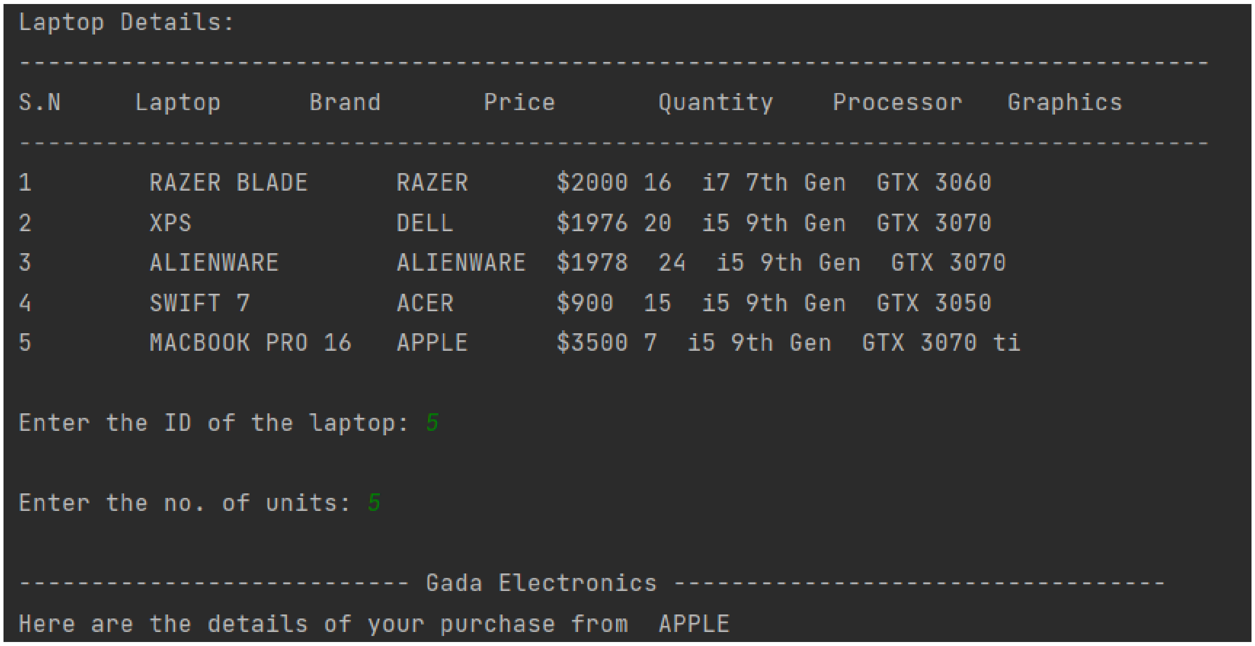
```
S.N     Laptop      Brand       Price       Quantity    Processor   Graphics
--------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 16  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978  24  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen  GTX 3070 ti

Enter the ID of the laptop: 9

Please provide a valid laptop ID!

Enter the ID of the laptop: -5

Please provide a valid laptop ID!

Enter the ID of the laptop: 
```

Fig: Entering invalid laptop id

7.3 Test 3

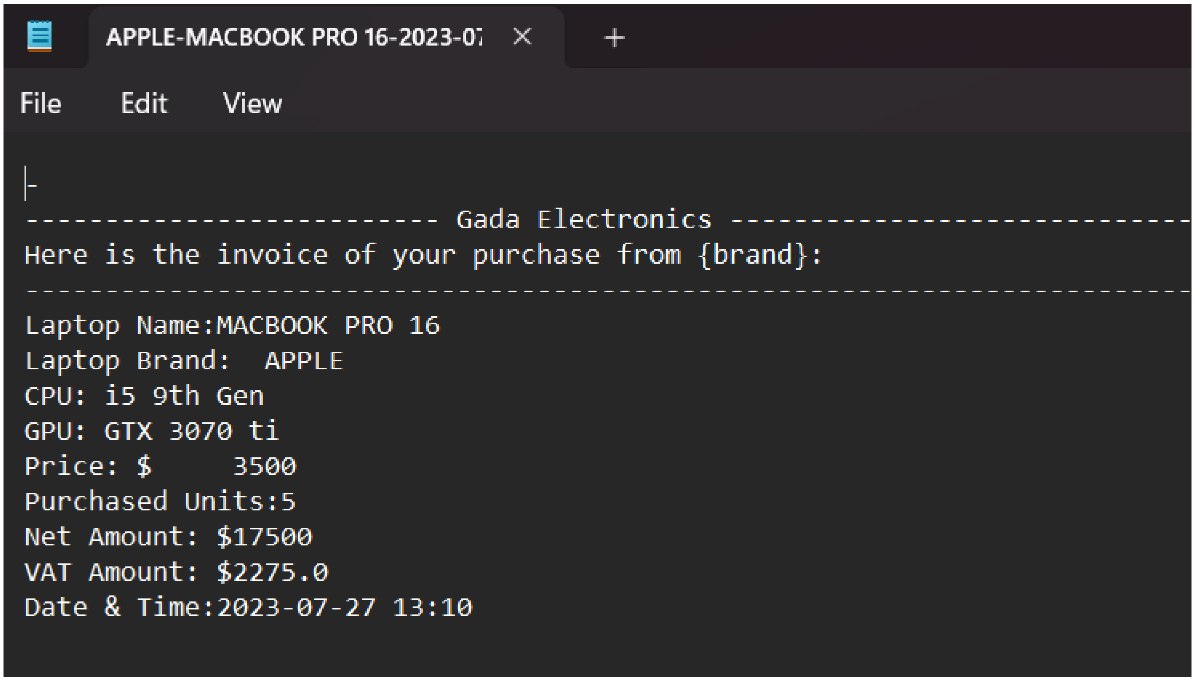
Objective	To test a successful purchase.
Action	The asked credentials were entered.
Expected Result	The purchase should be successful and ask to check the text file.
Actual Result	The purchase was successful, and the check text file message was displayed.

Conclusion	Testing was successful.



```
Laptop Details:
----------------------------------------------------------------------------------
S.N     Laptop      Brand       Price       Quantity    Processor   Graphics
----------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 16  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978  24  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 7  i5 9th Gen  GTX 3070 ti

Enter the ID of the laptop: 5

Enter the no. of units: 5

--------------------------- Gada Electronics ----------------------------------
Here are the details of your purchase from  APPLE
```

Fig: Purchasing laptops

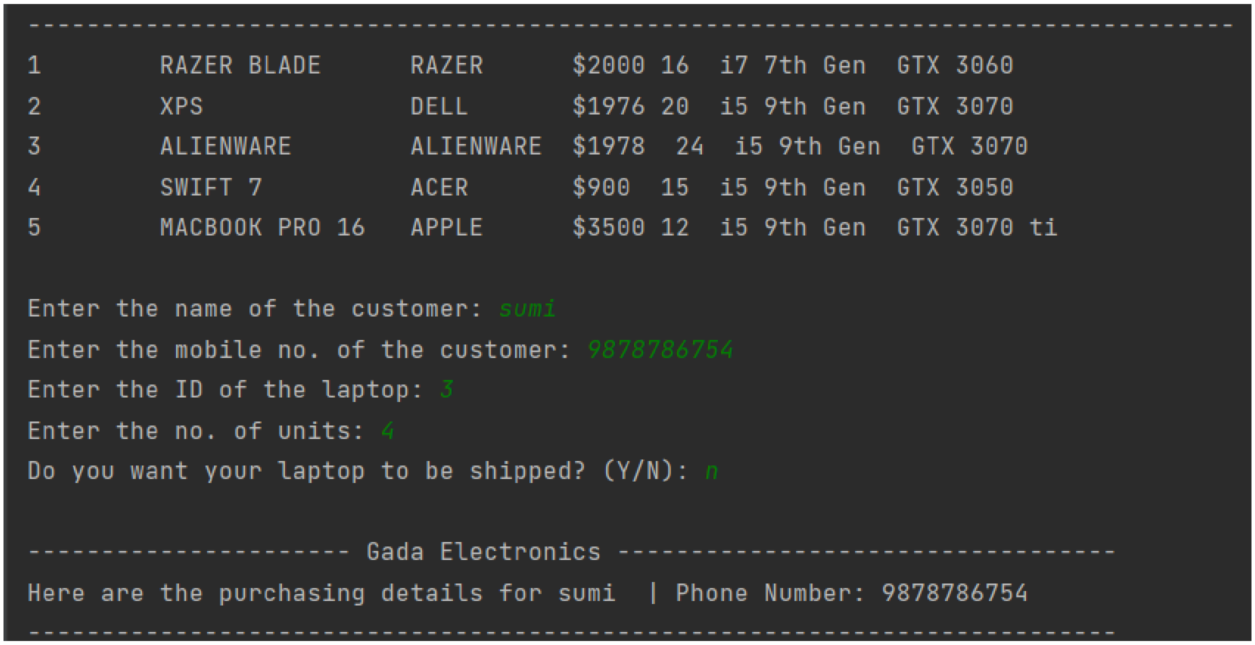


```
-
-------------------------- Gada Electronics -----------------------------
Here is the invoice of your purchase from {brand}:
--------------------------------------------------------------------------
Laptop Name:MACBOOK PRO 16
Laptop Brand:  APPLE
CPU: i5 9th Gen
GPU: GTX 3070 ti
Price: $     3500
Purchased Units:5
Net Amount: $17500
VAT Amount: $2275.0
Date & Time:2023-07-27 13:10
```

Fig: Purchase bill generation

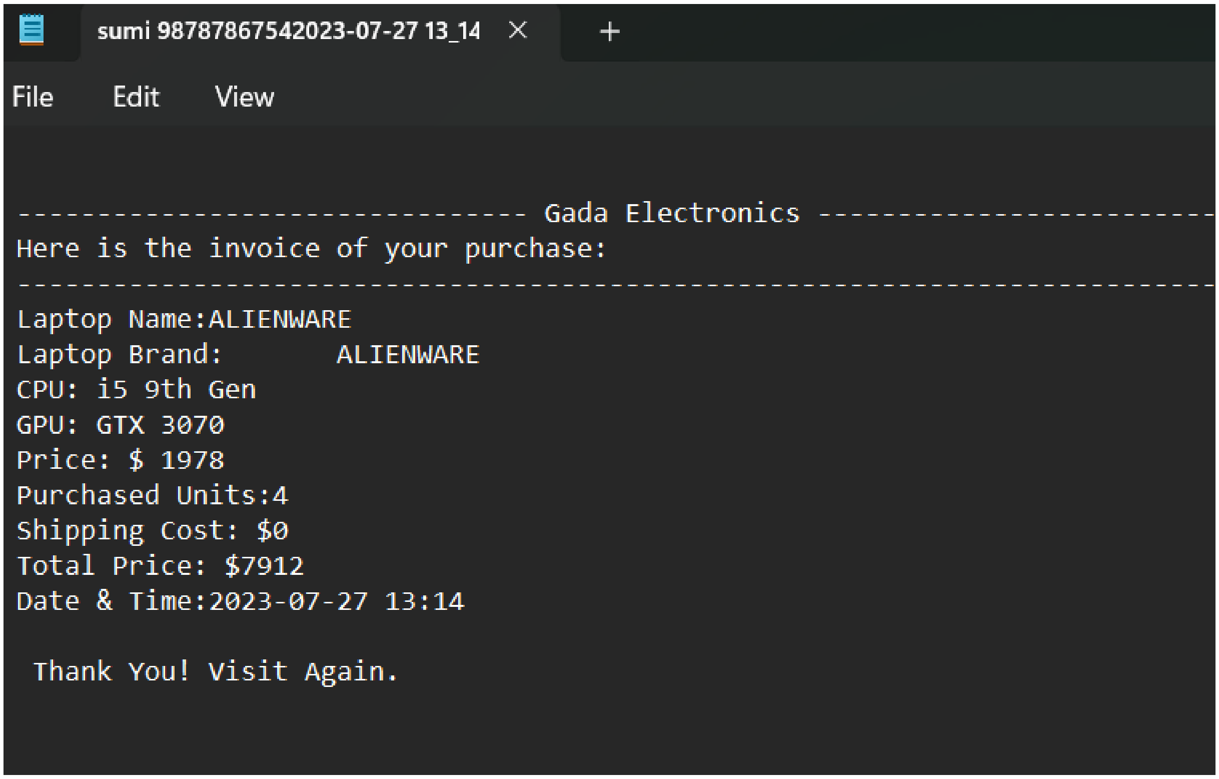
7.4 Test 4:

Objective	To test a successful sale.
Action	The asked credentials were entered.
Expected Result	The sale should be successful and ask to check the invoice.
Actual Result	The sale was successful, and the check invoice message was displayed.
Conclusion	Testing was successful.



```
----------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 16  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978  24  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 12  i5 9th Gen  GTX 3070 ti

Enter the name of the customer: sumi
Enter the mobile no. of the customer: 9878786754
Enter the ID of the laptop: 3
Enter the no. of units: 4
Do you want your laptop to be shipped? (Y/N): n

---------------------- Gada Electronics ----------------------------------
Here are the purchasing details for sumi  | Phone Number: 9878786754
--------------------------------------------------------------------------
```

Fig: Selling laptops

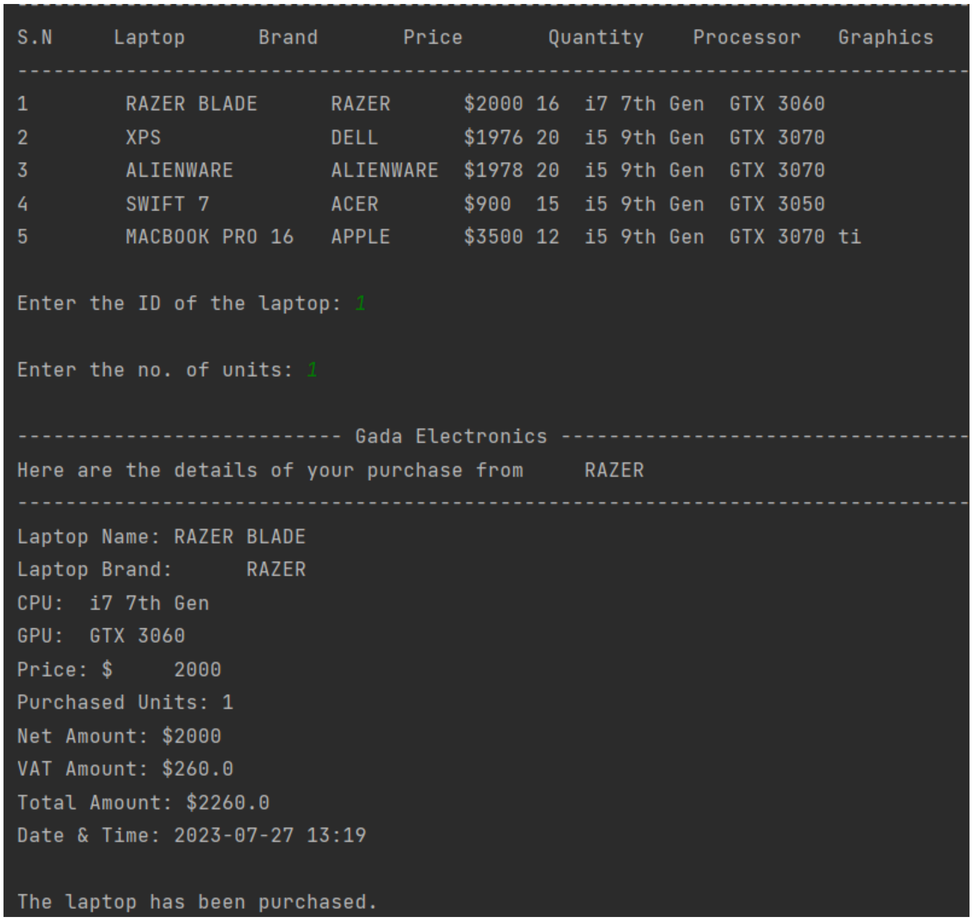


```
-------------------------------- Gada Electronics -------------------------
Here is the invoice of your purchase:
---------------------------------------------------------------------------
Laptop Name:ALIENWARE
Laptop Brand:       ALIENWARE
CPU: i5 9th Gen
GPU: GTX 3070
Price: $ 1978
Purchased Units:4
Shipping Cost: $0
Total Price: $7912
Date & Time:2023-07-27 13:14

 Thank You! Visit Again.
```

Fig: Sell bill generation

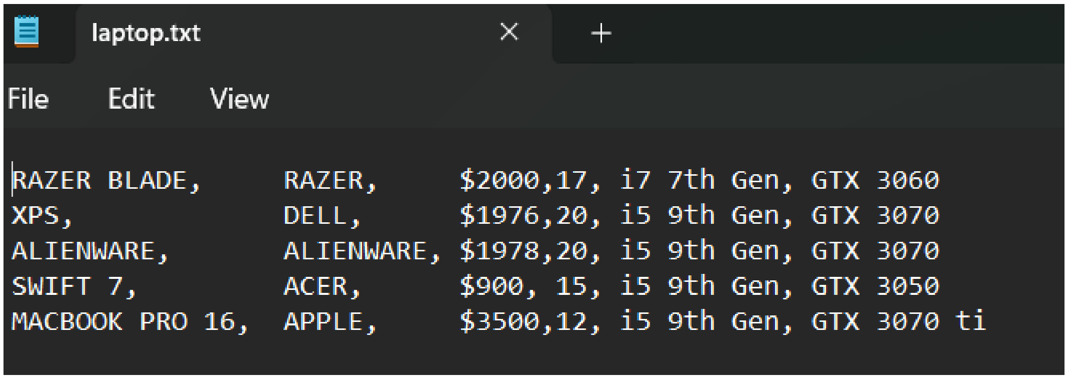
7.5 Test 5

Objective	To show the update in stock of laptop
Action	• laptops are purchased. • laptops are sold.
Expected Result	Quantity of the laptop should be increased while purchasing laptops. Quantity of the costume should be reduced while selling laptops
Actual Result	Quantity of laptops was increased. Quantity of laptops was reduced.
Conclusion	The test was successful.



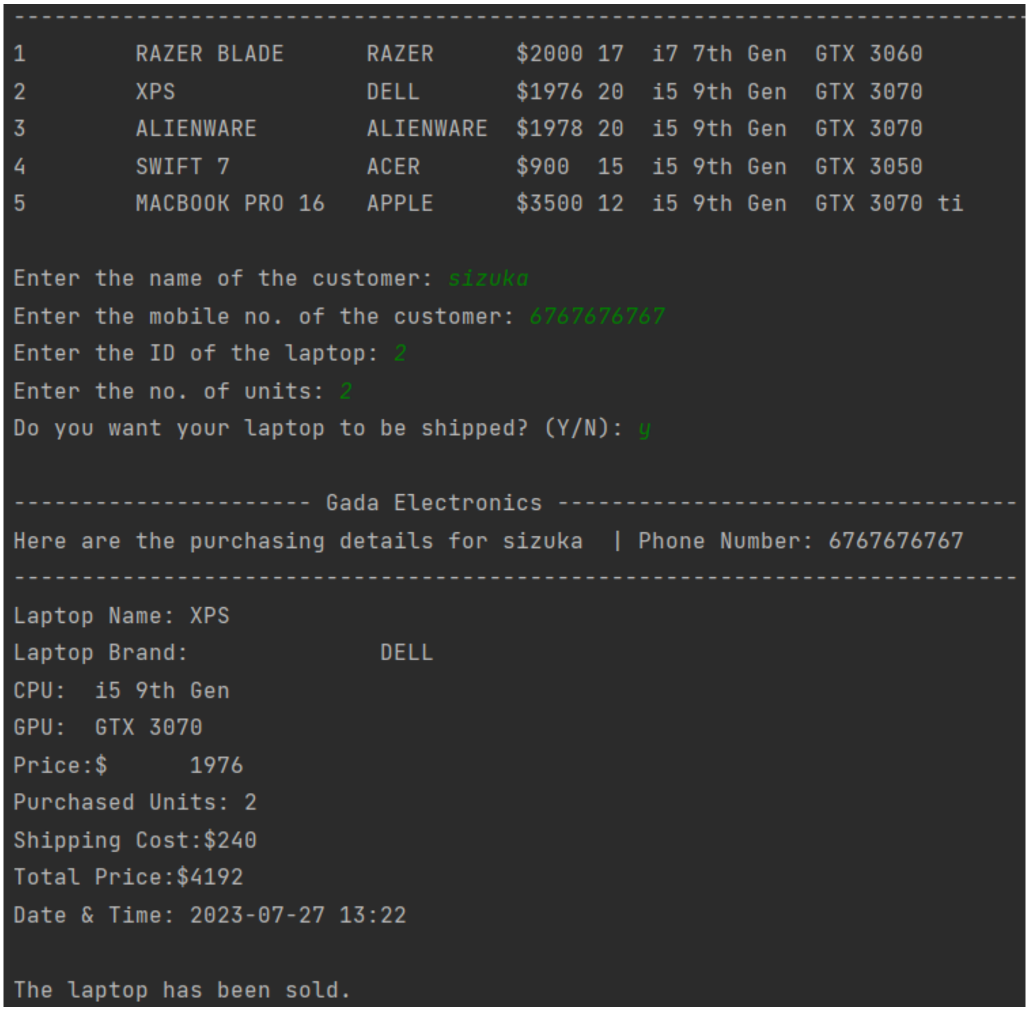
```
-------------------------------------------------------------------------------
S.N     Laptop      Brand       Price       Quantity     Processor    Graphics
-------------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 16  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978 20  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 12  i5 9th Gen  GTX 3070 ti

Enter the ID of the laptop: 1

Enter the no. of units: 1

--------------------------- Gada Electronics ---------------------------------
Here are the details of your purchase from     RAZER
-------------------------------------------------------------------------------
Laptop Name: RAZER BLADE
Laptop Brand:      RAZER
CPU:  i7 7th Gen
GPU:  GTX 3060
Price: $     2000
Purchased Units: 1
Net Amount: $2000
VAT Amount: $260.0
Total Amount: $2260.0
Date & Time: 2023-07-27 13:19

The laptop has been purchased.
```

Fig: Purchasing a laptop



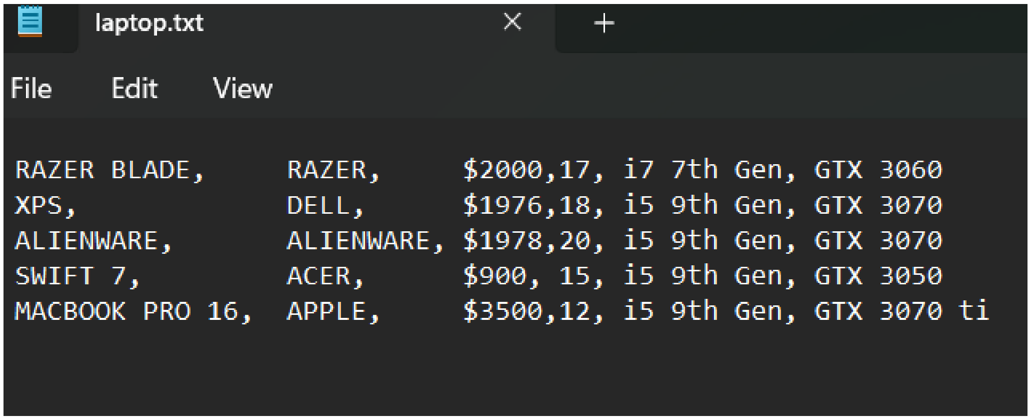
```
RAZER BLADE,     RAZER,     $2000,17, i7 7th Gen, GTX 3060
XPS,             DELL,      $1976,20, i5 9th Gen, GTX 3070
ALIENWARE,       ALIENWARE, $1978,20, i5 9th Gen, GTX 3070
SWIFT 7,         ACER,      $900, 15, i5 9th Gen, GTX 3050
MACBOOK PRO 16,  APPLE,     $3500,12, i5 9th Gen, GTX 3070 ti
```

Fig: Updated stock of laptop after purchasing



```
---------------------------------------------------------------------------
1        RAZER BLADE      RAZER      $2000 17  i7 7th Gen  GTX 3060
2        XPS              DELL       $1976 20  i5 9th Gen  GTX 3070
3        ALIENWARE        ALIENWARE  $1978 20  i5 9th Gen  GTX 3070
4        SWIFT 7          ACER       $900  15  i5 9th Gen  GTX 3050
5        MACBOOK PRO 16   APPLE      $3500 12  i5 9th Gen  GTX 3070 ti

Enter the name of the customer: sizuka
Enter the mobile no. of the customer: 6767676767
Enter the ID of the laptop: 2
Enter the no. of units: 2
Do you want your laptop to be shipped? (Y/N): y

---------------------- Gada Electronics ----------------------------------
Here are the purchasing details for sizuka  | Phone Number: 6767676767
--------------------------------------------------------------------------
Laptop Name: XPS
Laptop Brand:               DELL
CPU:  i5 9th Gen
GPU:  GTX 3070
Price:$      1976
Purchased Units: 2
Shipping Cost:$240
Total Price:$4192
Date & Time: 2023-07-27 13:22

The laptop has been sold.
```

Fig: Selling two laptops



```
RAZER BLADE,     RAZER,     $2000,17, i7 7th Gen, GTX 3060
XPS,             DELL,      $1976,18, i5 9th Gen, GTX 3070
ALIENWARE,       ALIENWARE, $1978,20, i5 9th Gen, GTX 3070
SWIFT 7,         ACER,      $900, 15, i5 9th Gen, GTX 3050
MACBOOK PRO 16,  APPLE,     $3500,12, i5 9th Gen, GTX 3070 ti
```

Fig: Updated stock after selling two laptops

8. Conclusion:

The Fundamentals of Computing coursework was an invaluable learning journey that equipped me with essential skills in algorithm design, flowchart creation, pseudocode, and Python programming as a whole. Throughout this coursework, I not only honed my technical abilities but also developed crucial time management skills and learned the importance of seeking help and overcoming challenges.

Initially, the tasks presented in the coursework were manageable, but as I progressed, I encountered various complexities and difficult scenarios. However, I persevered through hard work and determination, learning how to handle errors and exceptions while utilizing data structures and procedural techniques in Python programming. Implementing these concepts in my coding further enhanced my abilities. The knowledge and skills gained during this coursework hold great value for my future endeavors. Python, being a versatile programming language widely used across various fields, continues to gain popularity. With this in mind, I am eager to continue my exploration of Python and prepare myself for the countless projects and challenging situations that lie ahead.

In summary, the Fundamentals of Computing coursework provided an exceptional learning experience, imparting crucial skills and values necessary to become a proficient programmer. It laid a solid foundation for my proficiency in Python programming, and I am confident that the knowledge acquired will prove invaluable in future projects. I look forward to embracing the opportunities that Python presents and further advancing my capabilities as a skilled programmer.

9. Appendix:

9.1 Main file

```
def main():

    # Welcome message

    print("\n-----------Welcome to Gada Electronics--------------\n")
    while True:
        # Displaying options to the user
        print("\nSelect an option to continue:\n")
        print("1. Purchase from manufacturer")
        print("2. Sell to customer")
        print("3. Exit")

        # User input for the selected option
        choice = input("\nEnter your choice: ")

        if choice == "1":
            # If the user chooses to purchase a laptop, import and call the purchase_laptop
function from the operation module
            from operation import purchase_laptop
            purchase_laptop(choice)
            print("\nThe laptop has been purchased.")
            break

        elif choice == "2":
            # If the user chooses to sell a laptop, import and call the sell_laptop function
from the operation module
            from operation import sell_laptop
            sell_laptop(choice)
            print("\nThe laptop has been sold.")
            break

        elif choice == "3":
            # If the user chooses to exit, display a thank you message and return from the main function, effectively terminating the loop.
            print("Thank you for using the system")
            return

        else:
            # If the user enters an invalid choice, inform them about it.
            print("\nInvalid choice")

        # After each operation, ask the user if they want to buy or sell laptops again
    while True:
        re_choice = input("\nDo you want to Buy or Sell laptops again (Y/N): ")
        if re_choice.lower() == 'y':
            # If the user wants to continue buying or selling laptops, recursively call the main function again.
            main()
        else:
            # If the user does not want to continue, display a goodbye message and break out of the loop, effectively ending the program.
            print("\nThank you for using the System. Have a good day!")
            break

# Call the main function to start the program
main()
```

9.2 Operation file

```
#read laptop.txt and store it in a dictionary
with open("laptop.txt", "r") as file:
    laptop_dictionary = {i + 1: line.strip().split(",") for i, line in enumerate(file)}

# Function to get the current date and time
def datetime():
    import datetime
    return datetime.datetime.now().strftime("%Y-%m-%d %H:%M")

# Function for purchasing laptops
def purchase_laptop(choice):
    if choice == "1":
        print("\nLaptop Details:")
        from read import read_laptop
```

```
# Loop until a valid laptop ID is provided
while True:
    try:
        purchase_id = int(input("\nEnter the ID of the laptop: "))

        # Check if the provided laptop ID exists in the laptop_dictionary
        if purchase_id in laptop_dictionary:
            from write import purchase_update
            break  # Exit the loop if a valid laptop ID is provided
        else:
            print("\nPlease provide a valid laptop ID!")

    except ValueError:
        print("\nInvalid input. Enter a valid input.")

# Loop until a valid purchase quantity is provided
while True:
    try:
        purchase_quantity = int(input("\nEnter the no. of units: "))

        if purchase_quantity < 0:
            print("\nPlease provide a non-negative Quantity!")
        # Check if the provided quantity is valid for the selected laptop
```

```
                elif purchase_quantity <= int(laptop_dictionary[purchase_id][3]):
                    from write import purchase_update
                    break  # Exit the loop if a valid quantity is provided
                else:
                    print("\nPlease provide a valid Quantity!")

            except ValueError:
                print("\nInvalid input. Enter a valid input.")

        # After both loops, the valid 'purchase_id' and 'purchase_quantity' are available, and the 'purchase_update' function is called.
        purchase_update(purchase_id, purchase_quantity)

    else:
        print("Invalid input. Try again!")

# Function for selling laptops
def sell_laptop(choice):
    if choice == "2":
        print("\nLaptop Details:")
        from read import read_laptop

        # Loop until valid customer details and a valid laptop ID are provided
        while True:
            try:
                customer_name = input("\nEnter the name of the customer: ")
                customer_number = int(input("Enter the mobile no. of the customer: "))
                sell_id = int(input("Enter the ID of the laptop: "))

                # Check if the provided laptop ID exists in the laptop_dictionary
                if sell_id in laptop_dictionary:
                    sell_quantity = int(input("Enter the no. of units: "))

                    if sell_quantity < 0:
                        print("\nPlease provide a non-negative Quantity")
                    # Check if the requested sell_quantity is available in stock for the selected
laptop
                    if sell_quantity <= int(laptop_dictionary[sell_id][3]):
                        from write import sell_update
                        # After all details are provided and checked, call the 'sell_update'
function to update the sales data.
                        sell_update(sell_id, sell_quantity, customer_name, customer_number)
                        break
                    else:
                        print("The quantity you are looking for is not available at the moment.")
                else:
                    print("\nPlease provide a valid laptop ID!\n")
            except ValueError:

                print("\nInvalid input. Enter a valid input.\n")


    else:

        print("Invalid input. Try again!")
```

9.3 Write file

```
import operation

# Initialize an empty dictionary to store laptop details

laptop_dictionary = {}

# Start laptop_id at 1.

laptop_id = 1


# Attempt to read the laptop details from the "laptop.txt" file and populate the laptop_dictionary.

try:

    with open("laptop.txt", "r") as file:
        # Read each line from the file and create a dictionary with laptop_id as key and details as values.

        laptop_dictionary = {i + 1: line.strip().split(",") for i, line in enumerate(file)}

except FileNotFoundError:

    # If the file is not found, print an error message.

    print("File not found. Please make sure the 'laptop.txt' file exists.")



# Function to update laptop purchase details.

def purchase_update(purchase_id, purchase_quantity):

    # Extracting laptop details from the laptop_dictionary based on purchase_id.

    laptop = laptop_dictionary[purchase_id][0]

    brand = laptop_dictionary[purchase_id][1]

    price = laptop_dictionary[purchase_id][2].replace("$", "")

    processor = laptop_dictionary[purchase_id][4]

    graphics = laptop_dictionary[purchase_id][5]



    # Calculating the net amount, VAT, and total amount for the purchase.

    net_amount = int(price) * purchase_quantity

    purchase_datetime = operation.datetime()

    VAT = (13 / 100) * net_amount

    total_amount = net_amount + VAT



    # Updating the available quantity of the purchased laptop.
    laptop_dictionary[purchase_id][3] = str(int(laptop_dictionary[purchase_id][3]) + purchase_id)

    # Writing the updated laptop details back to the "laptop.txt" file.

    with open("laptop.txt", "w") as File:

        for values in laptop_dictionary.values():

            File.write(",".join(values))

            File.write("\n")



    # Printing the purchase details

    print("\n---------------------------- Gada Electronics -----------------------------------")

    print("Here are the details of your purchase from" + str(brand))

    print("----------------------------------------------------------------------------------")

    print("Laptop Name:",laptop)

    print("Laptop Brand:",brand)

    print("CPU:",processor)

    print("GPU:",graphics)

    print("Price: $"+price)

    print("Purchased Units:",purchase_quantity)

    print("Net Amount: $" + str(net_amount))

    print("VAT Amount: $" + str(VAT))

    print("Total Amount: $" + str(total_amount))

    print("Date & Time:",purchase_datetime)
    # Create a new text file to store the purchase bill
    purchase_bill = f"{brand}-{laptop}-{purchase_datetime}.txt"
    purchase_bill = purchase_bill.strip().replace(":", "_")
    with open(purchase_bill, "w") as purchase_text:
        # Write the details of the purchase to the file
        purchase_text.write("-\n---------------------------- Gada Electronics -------------------------------\n")
        purchase_text.write("Here is the invoice of your purchase from {brand}:\n")
        purchase_text.write("---------------------------------------------------------------------------------\n")
        purchase_text.write("Laptop Name:" + laptop + "\n")
        purchase_text.write("Laptop Brand:" + brand + "\n")
        purchase_text.write("CPU:" + processor + "\n")
        purchase_text.write("GPU:" + graphics + "\n")
        purchase_text.write("Price: $" + price + "\n")
        purchase_text.write("Purchased Units:" + str(purchase_quantity) + "\n")
        purchase_text.write("Net Amount: $" + str(net_amount) + "\n")
        purchase_text.write("VAT Amount: $" + str(VAT) + "\n")
        purchase_text.write("Date & Time:" + purchase_datetime + "\n")


    purchase_text.close()


# Function to update laptop sell details.
def sell_update(sell_id, sell_quantity, customer_name, customer_number):
```

```
# Updating the available quantity of the sold laptop.

laptop_dictionary[sell_id][3] = str(int(laptop_dictionary[sell_id][3]) - int(sell_quantity))

# Writing the updated laptop details back to the "laptop.txt" file.

with open("laptop.txt", "w") as File:

    for values in laptop_dictionary.values():

        File.write(",".join(values))

        File.write("\n")


# Asking if the laptop should be shipped and calculating the shipping cost.

ship_laptop = input("Do you want your laptop to be shipped? (Y/N): ")

if ship_laptop.lower() == 'y':

    ship_cost = 120

elif ship_laptop.lower() == 'n':

    ship_cost = 0

else:

    print("Invalid Input")

    return 0


# Extracting laptop details from the laptop_dictionary based on sell_id.

laptop = laptop_dictionary[sell_id][0]

brand = laptop_dictionary[sell_id][1]

price = laptop_dictionary[sell_id][2].replace("$", "")

processor = laptop_dictionary[sell_id][4]
```

```
    graphics = laptop_dictionary[sell_id][5]


    # Calculating the total shipping cost and total price for the sell.

    total_ship = ship_cost * sell_quantity

    total_price = (int(price) * sell_quantity) + total_ship

    sale_datetime = operation.datetime()


    # Printing the sell details

    print("\n---------------------- Gada Electronics ----------------------------------")

    print("Here are the purchasing details for", customer_name, "| Phone Number:", customer_number)

    print("--------------------------------------------------------------------------")

    print("Laptop Name:",laptop)

    print("Laptop Brand:",brand)

    print("CPU:",processor)

    print("GPU:",graphics)

    print("Price:$" + price)

    print("Purchased Units:",sell_quantity)

    print("Shipping Cost:$" + str(total_ship))

    print("Total Price:$" + str(total_price))

    print("Date & Time:",sale_datetime)


    # Create a new text file to store the sale bill
    sell_bill = f"{customer_name}{customer_number}{sale_datetime}.txt"

    sell_bill = sell_bill.strip().replace(":", "_")

    with open(sell_bill, "w") as sell_text:

        # Write the details of the sale to the file

        sell_text.write("\n------------------------------- Gada Electronics ------------------------\n")

        sell_text.write("Here is the invoice of your purchase:\n")

        sell_text.write("-------------------------------------------------------------------------\n")

        sell_text.write("Laptop Name:" + laptop + "\n")

        sell_text.write("Laptop Brand:" + brand + "\n")

        sell_text.write("CPU:" + processor + "\n")

        sell_text.write("GPU:" + graphics + "\n")

        sell_text.write("Price: $" + price + "\n")

        sell_text.write("Purchased Units:" + str(sell_quantity) + "\n")

        sell_text.write("Shipping Cost: $" + str(total_ship) + "\n")

        sell_text.write("Total Price: $" + str(total_price) + "\n")

        sell_text.write("Date & Time:" + sale_datetime + "\n")

        sell_text.write("\n Thank You! Visit Again.")


    sell_text.close()
```

9.4 Read file

```
def read_laptop():
    try:
        # Attempt to open the "laptop.txt" file in read mode
        with open("laptop.txt", "r") as file:
            # Print header for the table
            print("--------------------------------------------------------------------------------")
            print("S.N\t\tLaptop\t\tBrand\t\tPrice\t\tQuantity\tProcessor\tGraphics")
            print("--------------------------------------------------------------------------------")

            # Enumerate through each line in the file and display its content in a formatted table
            for id, line in enumerate(file, 1):
                # Replace any commas in the line with spaces and remove leading/trailing whitespaces
                print(id, "\t\t", line.replace(",", " ").strip())

    # Handle the exception when the file is not found
    except FileNotFoundError:
        print("File not found. Please make sure the 'laptop.txt' file exists.")

# Call the function to read and display the contents of the 'laptop.txt' file
read_laptop()
```