CS4051NI/CC4059NI Fundamentals of Computing

70% Individual Coursework

Milestone 3

2024/25 Spring

Student Name: Aditya Raj Bohora

London Met ID: 24046345

College ID: NP01NT4A240056

Assignment Due Date: 14th May

Assignment Submission Date: 14th May

Word Count: 542

Project File Links:

Onedrive Drive Link:	Keep Google Drive URL of your Project Here with Anyone in Organization can View Option Enabled

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.

turnitin Page 2 of 28 - Integrity Overview Submission ID trn:oid:::3618:95800143

13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **24** Not Cited or Quoted 13%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
12% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Table of Content

Contents

Table of Figures

1) Introduction

The objective of this coursework is to develop a proper skin care product sale system by programming in python, and we'll also be describing the program. In this coursework we'll be including many things like algorithm, data structure and the program itself. All of this will be done to create a proper and well-functioning skin care sale system. The program will consist of products in a table format after viewing the product, one can also buy product, the process to buy product would be to enter the correct number to select buy product after this the customer would be asked to enter their name the product they want and the quantity they want. The program also consists of restocking products. This report has its own aims and objective, and the technologies used which will be discussed below.

1.1) Aims and objective

1) To create a simple system that helps manage its product stock and sales. By this system it will be comparatively easier to manage and handle all the product and the stocks including the sales too.
2) One of the main aim and objective is also to show all the available products in the shop and their selling price including brand and country of origin. The shops collection of new or existing products to be showcased and the price to be visible to everyone.
3) To reduce the number of stocks of the products after each sale will also be the aims and objective is this report. It is equally important to reduce the number of stocks after each sale to make it organized.
4) To keep everything saved in simple txt files for easy use. By doing this thing gets simpler organized and easily accessible.

1.2) Technologies used while doing coursework

1) Python: This is the main and the only programming language used in this coursework. The language is very simple, its simplicity, readability and wide range of
libraries makes it ideal for implementing features like data processing, file handling and
Invoice generation which was also created.

2) IDLE (Integrated Development and Learning Environment): IDLE is a code editor for
Python. It was used to write, test and debugging. This provided a straightforward interface which was suitable for quick execution and inspection of python scripts.

3) Draw.io: Draw.io was used to design systems architecture and flow diagrams. It helped me with the flowchart designing and to visualizing the structure of the flow chart.
Draw.io also did aid in better planning and communication of the system design.

4) Microsoft Word: Word was used to prepare the documentation for the coursework. It helped in documenting the aims and objectives, data structure algorithm flowchart and many more.

2.Discussion and Analysis

2.1) Algorithm

.1) Main program

Step 1- Start the program

Step 2- Load all the product data from the products.txt file into a list.

Step 3- Display the Welcome Message to the visitors.

Step 4- Show Main Menu that leads you to options

Step 5- Now get the user's input

Step 6- Move forward with the request if the choice is valid or else print error

Step 7- Shows all of the four components of the program

Step 8- Starts the process to buy product

Step 9- Asks the customer to enter their name. This is also important to create a proper invoice of the product they buy.

Step 10- Enter a product id to purchase

Step 11- Once sold the stock gets updated

Step 12- This process generates and saves a sale invoice with the details like date and time. The invoice is automatically generated.

Step 13- Press 0 to finish the progress

Step 14- Enter to restock products

Step 15- Restock options pops ups

Step 16- To finally restock

Step 17- Enter the product id for conformation

Step 18- Input the quantity of the product

Step 19- Product is restocked successfully

Step 20- Starts the process to add new product

Step 21- Enter the new product name to add

Step 22- Enter the brand of the product

Step 23 – Enter the cost price that doubles when it gets to table

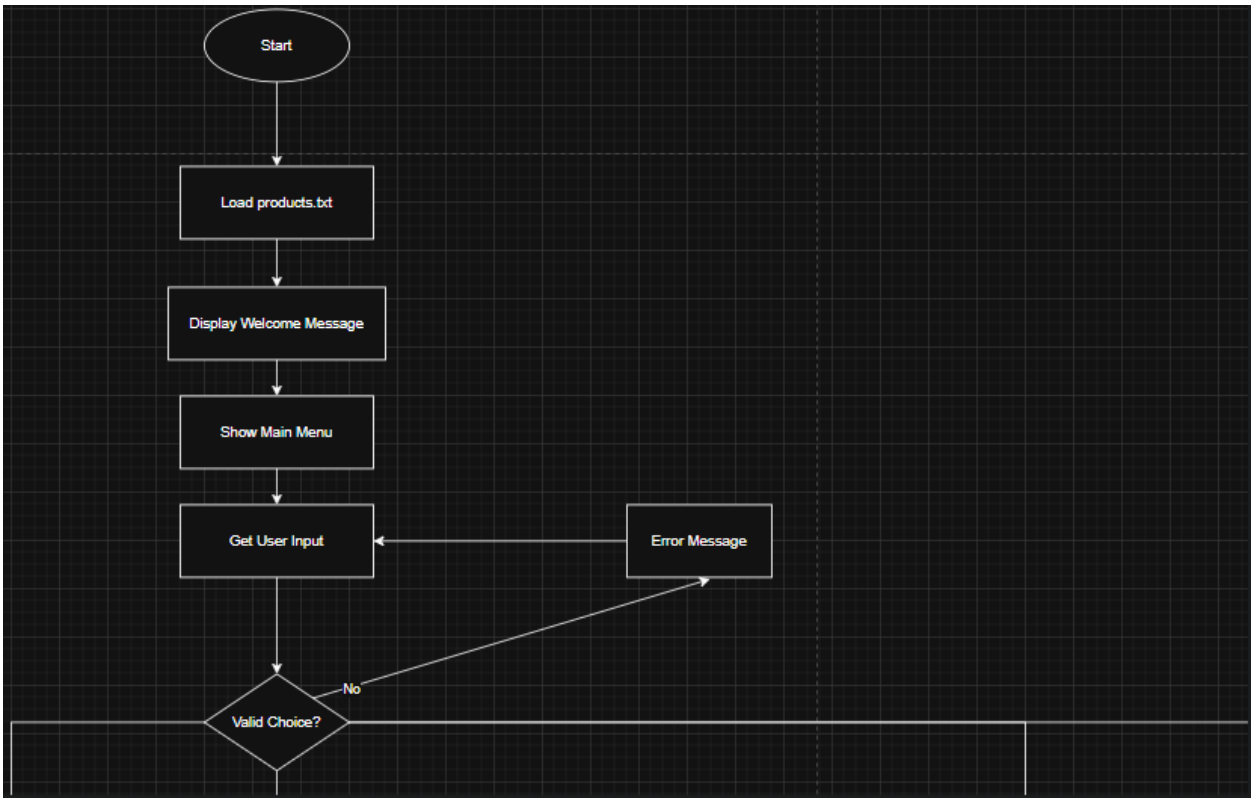
Step 24- Enter the quantity to add

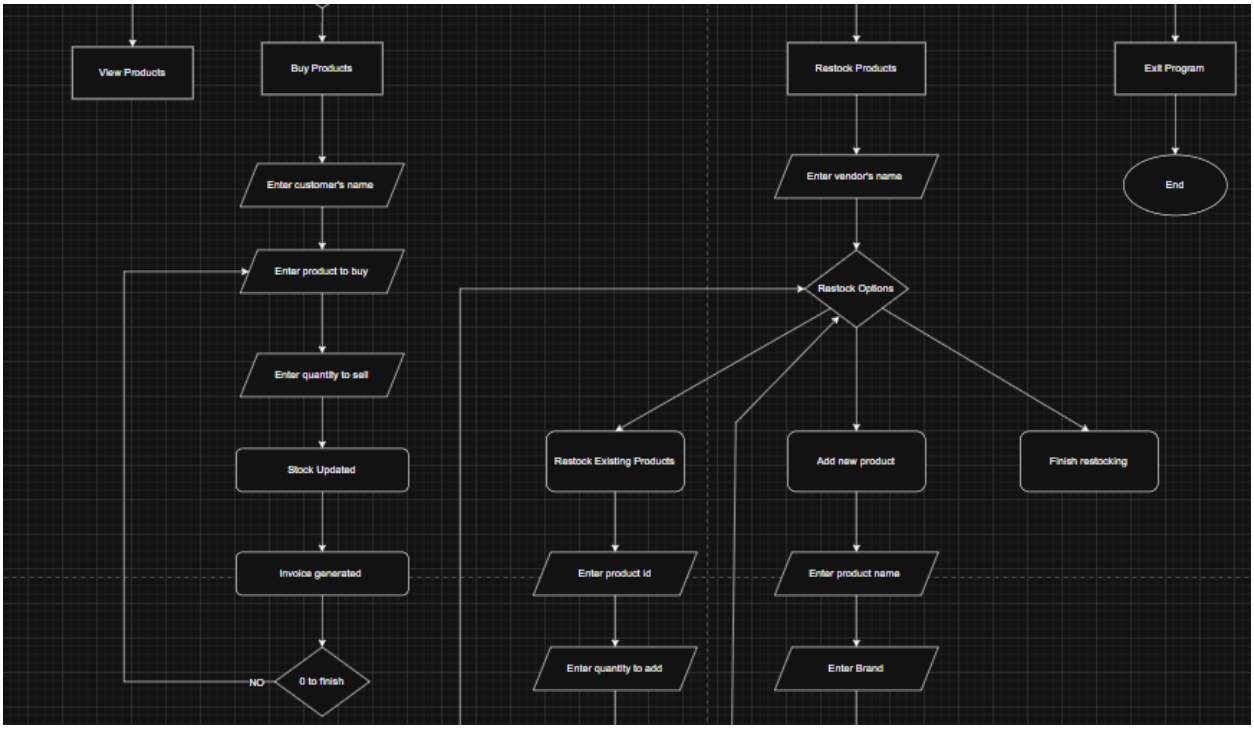
Step 25- Input the country of origin

Step 26- New product is finally then added

Step 27- End

2.2) Flowchart





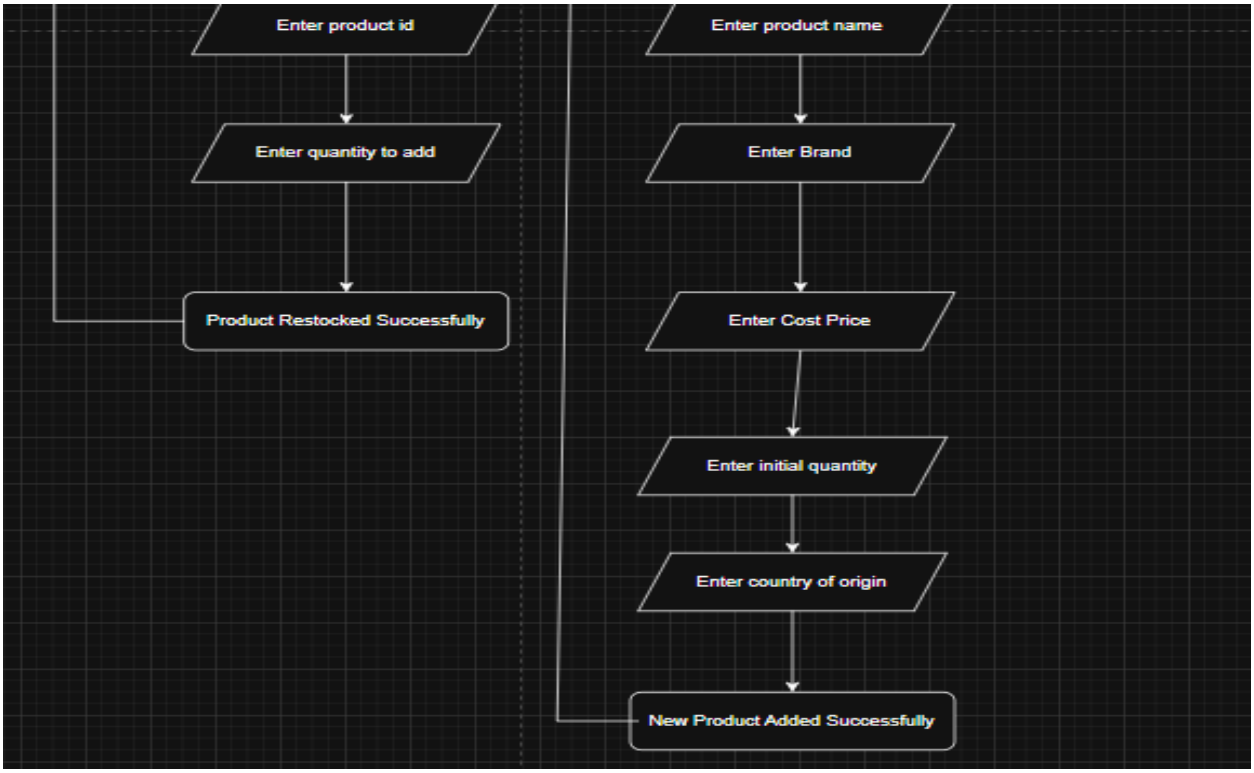


Figure 1 Flowchart of the program

2.3) Pseudocode

```
PROGRAM WeCareCosmetics

    // Main function

    FUNCTION main()

        PRINT welcome banner and store information

        products = LOAD_PRODUCTS()



        WHILE True

            PRINT main menu options:

                1. View Products
```

```
        2. Buy Products

        3. Restock Products

        4. Exit



    GET user choice



    IF choice == 1

        DISPLAY_PRODUCTS(products)

    ELSE IF choice == 2

        SELL_PRODUCTS(products)

    ELSE IF choice == 3

        RESTOCK_PRODUCTS(products)

    ELSE IF choice == 4

        PRINT goodbye message

        EXIT

    ELSE

        PRINT invalid choice message

    END IF

END WHILE
```

```
END FUNCTION


// Product management functions
FUNCTION LOAD_PRODUCTS()
    OPEN products.txt file
    INITIALIZE empty products list


    FOR EACH line in file
        REMOVE newline character
        SPLIT line by commas into parts
        IF valid product data (5 parts)
            CREATE product dictionary with:
                name, brand, quantity, cost, country
            ADD product to products list
        END IF
    END FOR


    RETURN products list
END FUNCTION
```

```
FUNCTION SAVE_PRODUCTS(products)
    OPEN products.txt file for writing
    FOR EACH product in products
        WRITE product data as comma-separated line
    END FOR
    CLOSE file
END FUNCTION


// Display functions
FUNCTION DISPLAY_PRODUCTS(products)
    PRINT table header with columns:
        Product, Brand, Price (Rs), In Stock, Country


    FOR EACH product in products
        CALCULATE selling price (cost * 2)
        PRINT product info with proper spacing
    END FOR
END FUNCTION
```

```
// Sales functions
FUNCTION SELL_PRODUCTS(products)
    GET customer name
    INITIALIZE total_amount to 0
    INITIALIZE empty invoice_lines list

    WHILE True
        DISPLAY available products with numbers

        GET product choice from user
        IF choice is 0
            BREAK
        END IF

        SELECT product based on choice
        IF product out of stock
            PRINT message
            CONTINUE
```

```
        END IF


        GET quantity to sell
        CALCULATE free items (quantity // 3)
        CALCULATE total items (quantity + free items)


        IF not enough stock
            PRINT error message
            CONTINUE
        END IF


        UPDATE product quantity
        CALCULATE subtotal
        ADD to total_amount
        ADD transaction to invoice_lines
    END WHILE


    IF items were purchased
        GENERATE invoice file with:
```

```
            Customer info
            Date
            Purchased items
            Total amount
        SAVE_PRODUCTS(products)
    END IF
END FUNCTION


// Restocking functions
FUNCTION RESTOCK_PRODUCTS(products)
    GET vendor name
    INITIALIZE total_amount to 0
    INITIALIZE empty invoice_lines list


    WHILE True
        PRINT restock options:
            1. Restock existing
            2. Add new product
            3. Finish
```

```
GET user choice

IF choice == 1
    DISPLAY products with IDs
    GET product ID to restock
    IF valid ID
        GET quantity to add
        UPDATE product quantity
        CALCULATE subtotal
        ADD to total_amount
        ADD transaction to invoice_lines
    END IF

ELSE IF choice == 2
    GET new product details:
        name, brand, cost, quantity, country
    CREATE new product
    ADD to products list
```

```
            SAVE_PRODUCTS(products)
            ADD transaction to invoice_lines


        ELSE IF choice == 3
            BREAK


        ELSE
            PRINT invalid choice
        END IF
    END WHILE


    IF items were restocked
        GENERATE restock invoice file
        SAVE_PRODUCTS(products)
    END IF
  END FUNCTION
END PROGRAM
```

2.4) Data Structure

Used data structure according to my program

1) List: List is a data structure used to store a collection of items in a single variable. Lists are also mutable (changeable) and can hold items of many data types.
Products = []- This stores all the product dictionaries.
Restocked_items = This stores items during restocking.

2) Dictionary: Similar to list, dictionary is also a built-in data structure that stores data in key –value pairs. Some of the key features of a dictionary are-
- It is defined using curly brackets {}.
- Each item is a key: value pair.
- These values can be of any data type.

Unused data structure

1) Tuples: A tuple is also a built-in data type that is used to store a collection of items. They are ordered but immutable meaning it cannot be changed once created. It can also contain elements of different data types. Tuples are denoted by parentheses ().
2) Sets: Set is data type that is used to store a collection of unique items. They are unordered and mutable unlike tuples they can be modified even after created. Sets are donated using curly brackets {}.

3) Program

3.1) Implementation of the program

The We Care Cosmetics program is implemented as a very modular Python application divided into four key components that works together to manage the product inventory, sales and restocking operations. This architecture

follows distinct modules for data loading (read.py), data saving (write.py), business logic (operation.py) and program execution (main.py).
The 4 components that the module is divided into are:

1) View product- View product is used to show us the available product in the store with its price brand quantity available and its origin as well. It shows complete detail about the product in a very proper manner.

2) Buy products- This is the section where customer's make their purchase. Just like the View product Buy product also shows the table in nearly the same manner. The system ask's the customer to enter their name before accessing the table they are then able to choose freely about what they want. The Buy product also has a **promotional logic of "Buy 3 Get 1 Free".**

3) Restock products – Unlike the two of the user input, restock is quite different as it does not show any table when clicked but rather asks the admin to choose an option between restocking and adding. Adding is done to add new product to the table.

4) Exit- This is to simply terminate the program.

3.2) Screenshot of programs

```
******************************************
                WE CARE COSMETICS
******************************************
                DURBARMARG, KATHMANDU | Phone no: 9842322211
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         A VERY DELIGHTFUL GREETINGS! Crafted for the queen in you
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Discover our Signature Collection

==== Welcome to We Care Cosmetics ====
1. View Products
2. Buy Products
3. Restock Products
4. Exit
Enter your choice (1-4): 1

Available Products:
--------------------------------------------------------------------------------
Product             Brand          Price (Rs)In Stock  Country
--------------------------------------------------------------------------------
FaceSerum           Neutro         1800.0    350       USA
HerbalToner         Biotiq         600.0     150       India
LipBalms            Nivea          240.0     250       Germany
Moisturizer         Olay           1000.0    130       USA
FacePack            WOW            700.0     100       India
Anti-AgeCream       Lotus          1300.0    100       India
RosewaterSpray      Dabur          160.0     200       India
FaceSheetMask       Ceta           300.0     160       SouthKorea
AloeNightGel        Troy           800.0     140       India
VitaminEOil         Boly           1200.0    90        UK
AppleWash           Apple          2000.0    100       France
--------------------------------------------------------------------------------
```

Figure 2 When Selecting on View Products

```
==== Welcome to We Care Cosmetics ====
1. View Products
2. Buy Products
3. Restock Products
4. Exit
Enter your choice (1-4): 2
Enter customer name: adi

Available Products:
--------------------------------------------------------------------------------
No.  Product            Brand          Price (Rs)In Stock  Country
--------------------------------------------------------------------------------
1    FaceSerum          Neutro         1800.0    350       USA
2    HerbalToner        Biotiq         600.0     150       India
3    LipBalms           Nivea          240.0     250       Germany
4    Moisturizer        Olay           1000.0    130       USA
5    FacePack           WOW            700.0     100       India
6    Anti-AgeCream      Lotus          1300.0    100       India
7    RosewaterSpray     Dabur          160.0     200       India
8    FaceSheetMask      Ceta           300.0     160       SouthKorea
9    AloeNightGel       Troy           800.0     140       India
10   VitaminEOil        Boly           1200.0    90        UK
11   AppleWash          Apple          2000.0    100       France
--------------------------------------------------------------------------------
Enter product number to buy (or '0' to finish): 1
Enter quantity to sell for FaceSerum (Available: 350): 10
Product added to invoice.
```

Figure 3 Process of buying a product

```
Available Products:
--------------------------------------------------------------------------------
No.  Product            Brand          Price (Rs)In Stock  Country
--------------------------------------------------------------------------------
1    FaceSerum          Neutro         1800.0    337       USA
2    HerbalToner        Biotiq         600.0     150       India
3    LipBalms           Nivea          240.0     250       Germany
4    Moisturizer        Olay           1000.0    130       USA
5    FacePack           WOW            700.0     100       India
6    Anti-AgeCream      Lotus          1300.0    100       India
7    RosewaterSpray     Dabur          160.0     200       India
8    FaceSheetMask      Ceta           300.0     160       SouthKorea
9    AloeNightGel       Troy           800.0     140       India
10   VitaminEOil        Boly           1200.0    90        UK
11   AppleWash          Apple          2000.0    100       France
--------------------------------------------------------------------------------
Enter product number to buy (or '0' to finish):
```

Figure 4 Stock decreasing after buying a product

```
************************************
                WE CARE COSMETICS
************************************
                DURBARMARG, KATHMANDU | Phone no: 9842322211
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         A VERY DELIGHTFUL GREETINGS! Crafted for the queen in you
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Discover our Signature Collection

==== Welcome to We Care Cosmetics ====
1. View Products
2. Buy Products
3. Restock Products
4. Exit
Enter your choice (1-4): 3
Enter supplier/vendor name: dan

Restock Options:
1. Restock existing products
2. Add new product
3. Finish restocking
Enter your choice (1-3): |
```

Figure 5 Main interface of Restock products

```
Restock Options:
1. Restock existing products
2. Add new product
3. Finish restocking
Enter your choice (1-3): 1

Available Products:
--------------------------------------------------------------------------------
ID   Product             Brand          Price     Stock
--------------------------------------------------------------------------------
1    FaceSerum           Neutro         900.0     350
2    HerbalToner         Biotiq         300.0     150
3    LipBalms            Nivea          120.0     250
4    Moisturizer         Olay           500.0     130
5    FacePack            WOW            350.0     100
6    Anti-AgeCream       Lotus          650.0     100
7    RosewaterSpray      Dabur          80.0      200
8    FaceSheetMask       Ceta           150.0     160
9    AloeNightGel        Troy           400.0     140
10   VitaminEOil         Boly           600.0     90
11   AppleWash           Apple          1000.0    100
--------------------------------------------------------------------------------
Enter product ID to restock (0 to cancel): 1

Selected: FaceSerum (Neutro) - Current stock: 350
Enter quantity to add: 10
Product restocked successfully!

Restock Options:
1. Restock existing products
2. Add new product
3. Finish restocking
Enter your choice (1-3):
```

Figure 6 Process of restocking Products

__pycache__	5/14/2025 10:07 AM	File folder	
invoice_restock_ad_20250514065055	5/14/2025 6:50 AM	Text Document	1 KB
invoice_sale_afd_20250514065015	5/14/2025 6:50 AM	Text Document	1 KB
main	5/14/2025 9:53 AM	Python File	3 KB
operation	5/14/2025 10:07 AM	Python File	12 KB
products	5/14/2025 10:21 AM	Text Document	1 KB
read	5/14/2025 9:56 AM	Python File	2 KB
write	5/14/2025 9:57 AM	Python File	1 KB

Figure 7 Invoice getting generated

```
Supplier: ad
Date: 2025-05-14 06:50:55.779629

Restocked Items:
Anti-AgeCream   Lotus   1        Rate: Rs. 650.0  Subtotal: Rs. 650.0

Total Restock Cost: Rs. 650.0
```

Figure 8 Screenshot of Restock product invoice

4) Testing

4.1) Test 1

Objectives	Verify the program handles invalid input gracefully.
Action	Run the program Select "2. Buy Products" When asked for quantity, enter "abc" (non-numeric input)
Expected Result	For invalid product number: "Invalid input. Try again." message appears
Actual Result	Program displays "Invalid input. Try again." for invalid product number
Conclusion	Input validation works as expected.

```
Available Products:
--------------------------------------------------------------------------------
No.  Product            Brand          Price (Rs)In Stock  Country
--------------------------------------------------------------------------------
1    FaceSerum          Neutro         1800.0    350       USA
2    HerbalToner        Biotiq         600.0     150       India
3    LipBalms           Nivea          240.0     250       Germany
4    Moisturizer        Olay           1000.0    130       USA
5    FacePack           WOW            700.0     100       India
6    Anti-AgeCream      Lotus          1300.0    104       India
7    RosewaterSpray     Dabur          160.0     200       India
8    FaceSheetMask      Ceta           300.0     160       SouthKorea
9    AloeNightGel       Troy           800.0     140       India
10   VitaminEOil        Boly           1200.0    90        UK
11   AppleWash          Apple          2000.0    100       France
--------------------------------------------------------------------------------
Enter product number to buy (or '0' to finish): 1
Enter quantity to sell for FaceSerum (Available: 350): abc
Invalid quantity.
```

Figure 9Testing of Test1

4.2) Test 2

Objectives	Verify product purchase logic handles edge cases.
Action	Run the program Select "2. Buy Products" Choose an available product Enter "-5" for quantity (negative value) Then enter quantity greater than available stock
Expected Result	For negative value: "Quantity must be greater than zero."
Actual Result	Negative quantity shows proper error message.
Conclusion	Purchase validation works correctly.

```
Available Products:
--------------------------------------------------------------------------------
No.  Product            Brand          Price (Rs)In Stock  Country
--------------------------------------------------------------------------------
1    FaceSerum          Neutro         1800.0    350       USA
2    HerbalToner        Biotiq         600.0     150       India
3    LipBalms           Nivea          240.0     250       Germany
4    Moisturizer        Olay           1000.0    130       USA
5    FacePack           WOW            700.0     100       India
6    Anti-AgeCream      Lotus          1300.0    104       India
7    RosewaterSpray     Dabur          160.0     200       India
8    FaceSheetMask      Ceta           300.0     160       SouthKorea
9    AloeNightGel       Troy           800.0     140       India
10   VitaminEOil        Boly           1200.0    90        UK
11   AppleWash          Apple          2000.0    100       France
--------------------------------------------------------------------------------
Enter product number to buy (or '0' to finish): 1
Enter quantity to sell for FaceSerum (Available: 350): -5
Quantity must be greater than zero.
```

Figure 10 Testing of Test 2

4.3) Test 3

Objectives	Verify complete purchase process and invoice generation for multiple products.
Action	Run the program Select "2. Buy Products" Choose first product (e.g., FaceSerum) Enter quantity: 4 (should get 1 free) Choose second product (e.g., LipBalms) Enter quantity: 2 (no free item) Enter '0' to finish purchase Enter customer name: "Khan"
Expected Result	Multiple invoice to be created with one free item.
Actual Result	Multiple invoice created.

Conclusion	Purchase process works correctly for multiple products "Buy 3 Get 1 Free" calculation is accurate Invoice file generated with proper details

4.4) Test 4

Objectives	Verify complete sales process and restock documentation.
Action	Run the program Select "3. Restock Products" Enter vendor name: "Adil" Select "1. Restock existing products" Choose FaceSerum (ID 1) Add quantity: 10 Select "2. Add new product" Name: "AppleWash" Brand: "Apple" Cost: 1000 Quantity: 100 Country: "France" Select "3. Finish restocking"
Expected Result	The Restock to function properly.
Actual Result	The Restock functioned properly.
Conclusion	Restocking existing products works correctly and adding new products functions as expected Restock invoice contains all transaction details

4.5) Test 5

Objectives	Verify real-time stock updates during purchases and restocking.
Action	Check initial stock in products.txt (e.g., FaceSerum: 167) Run program and select "2. Buy Products" Purchase FaceSerum with quantity 3 (should get 1 free) Complete purchase Check updated products.txt
Expected Result	New product to be added at product.txt.
Actual Result	New product added at product.txt.
Conclusion	Stock updates work perfectly in both directions.

5) Conclusion

With this now we've come to the end of this report. The main objective of this coursework was to build a proper functioning skincare product store that would sell products restock products and the invoices to be added too. There are four things to the very first user input that being- View products, buy products, Restock products and Exit. Buy product section is for the customers to buy the products and Restock section is for the admin to restock the products. This is how the whole wholesale cosmetics is built and how it functions.

6) Apendix

Main.py

```
"""
main.py - Main entry point for the We Care Cosmetics shop system
Handles the main menu and program flow
"""

from operation import display_products, sell_products, restock_products
from read import load_products

def show_welcome():
    """Displays the welcome banner and shop information"""
    print("\n" + "*" * 50)
    print("\t\tWE CARE COSMETICS".center(50))
    print("*" * 50)
    print("\tDURBARMARG, KATHMANDU | Phone: 9842322211".center(50))
    print("~" * 50)
    print("\tA VERY DELIGHTFUL GREETINGS!".center(50))
    print("\tCrafted for the queen in you".center(50))
    print("~" * 50)
    print("Discover our Signature Collection".center(50) + "\n")

def main():
    """Main program loop"""
    show_welcome()
    products = load_products()

    while True:
        print("\n==== Main Menu ====")
        print("1. View Products")
```

```
        print("2. Sell Products")
        print("3. Restock Products")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == "1":
            display_products(products)
        elif choice == "2":
            products = sell_products(products)
        elif choice == "3":
            products = restock_products(products)
        elif choice == "4":
            print("\nThank you for using We Care Cosmetics system!")
            print("Have a wonderful day!\n")
            break
        else:
            print("Invalid choice. Please enter 1-4.")

if __name__ == "__main__":
    main()
```

operation.py

```
"""
operation.py - Module containing core business logic operations
Includes functions for product display, sales, and restocking
"""

from write import save_products, generate_sale_invoice, generate_restock_invoice
```

```
from read import load_products
import datetime

def display_products(products):
    """
    Displays available products in a formatted table

    Args:
        products (list): List of product dictionaries
    """
    print("\nAvailable Products:")
    print("-" * 80)
    print(
        'Product'.ljust(20) +
        'Brand'.ljust(15) +
        'Price (Rs)'.ljust(12) +
        'In Stock'.ljust(12) +
        'Country'.ljust(15)
    )
    print("-" * 80)

    for p in products:
        selling_price = p['cost'] * 2
        print(
            p['name'].ljust(20) +
            p['brand'].ljust(15) +
            str(round(selling_price, 2)).ljust(12) +
            str(p['qty']).ljust(12) +
            p['country'].ljust(15)
        )
    print("-" * 80)
```

```
def sell_products(products):
    """
    Handles the product selling process including invoice generation

    Args:
        products (list): List of product dictionaries

    Returns:
        list: Updated list of products
    """
    customer_name = input("Enter customer name: ").strip()
    total_amount = 0
    invoice_lines = []

    while True:
        display_products(products)
        choice = input("\nEnter product number to buy (or '0' to finish): ")

        if not choice.isdigit() or int(choice) < 0 or int(choice) > len(products):
            print("Invalid input. Please try again.")
            continue

        choice = int(choice)
        if choice == 0:
            break

        selected = products[choice-1]
        if selected['qty'] <= 0:
            print(f"Sorry, {selected['name']} is out of stock.")
            continue
```

```
        try:
            sell_qty = int(input(f"Enter quantity to buy (Available: {selected['qty']}): "))
            if sell_qty <= 0:
                print("Quantity must be positive.")
                continue

            free_items = sell_qty // 3
            total_items = sell_qty + free_items

            if total_items > selected['qty']:
                print(f"Not enough stock! You requested {sell_qty} (+{free_items} free) = {total_items}, but only {selected['qty']} available.")
                continue

            selected['qty'] -= total_items
            subtotal = (selected['cost'] * 2) * sell_qty
            total_amount += subtotal

            line = f"{selected['name']}\t{selected['brand']}\t{sell_qty} +{free_items} free\tRs. {subtotal:.2f} (Buy 3 Get 1 Free)"
            invoice_lines.append(line)
            print(f"Added {sell_qty} {selected['name']} (+{free_items} free) to invoice.")

        except ValueError:
            print("Invalid quantity entered.")
            continue

    if total_amount > 0:
        if generate_sale_invoice(customer_name, invoice_lines, total_amount):
            save_products(products)
```

```
    return products

def restock_products(products):
    """
    Handles product restocking including new product addition

    Args:
        products (list): List of product dictionaries

    Returns:
        list: Updated list of products
    """
    vendor_name = input("Enter supplier/vendor name: ").strip()
    total_amount = 0
    invoice_lines = []

    while True:
        print("\nRestock Options:")
        print("1. Restock existing product")
        print("2. Add new product")
        print("3. Finish restocking")
        choice = input("Enter choice (1-3): ")

        if choice == '1':
            display_products(products)
            try:
                product_id = int(input("Enter product ID to restock (0 to cancel): "))
                if product_id == 0:
                    continue
                if 1 <= product_id <= len(products):
```

```
                selected = products[product_id-1]
                add_qty = int(input(f"Enter quantity to add to {selected['name']}: "))
                if add_qty > 0:
                    selected['qty'] += add_qty
                    subtotal = selected['cost'] * add_qty
                    total_amount += subtotal
                    line = f"{selected['name']}\t{selected['brand']}\t{add_qty}\tRate: Rs. {selected['cost']}  Subtotal: Rs. {subtotal:.2f}"
                    invoice_lines.append(line)
                    print(f"Added {add_qty} {selected['name']} to inventory.")
            except ValueError:
                print("Invalid input. Please enter numbers only.")

        elif choice == '2':
            print("\nAdd New Product:")
            name = input("Product name: ").strip()
            brand = input("Brand: ").strip()
            try:
                cost = float(input("Cost price: "))
                qty = int(input("Initial quantity: "))
                country = input("Country of origin: ").strip()

                new_product = {
                    'name': name,
                    'brand': brand,
                    'qty': qty,
                    'cost': cost,
                    'country': country
                }
                products.append(new_product)
                subtotal = cost * qty
```

```
            total_amount += subtotal
            line = f"{name}\t{brand}\t{qty}\tRate: Rs. {cost}  Subtotal: Rs. {subtotal:.2f}"
            invoice_lines.append(line)
            print(f"Added new product: {name}")
        except ValueError:
            print("Invalid cost or quantity entered.")

    elif choice == '3':
        break

    else:
        print("Invalid choice. Please enter 1, 2, or 3.")

if total_amount > 0:
    if generate_restock_invoice(vendor_name, invoice_lines, total_amount):
        save_products(products)

return products
```

read.py

```
"""
read.py - Module for handling all read operations in the cosmetics shop system
Includes functions for loading product data
"""

def load_products(filename="products.txt"):
    """
    Loads product data from a text file

    Args:
        filename (str): Name of file to read from (default: products.txt)
```

```
    Returns:
        list: List of product dictionaries
    """
    products = []
    try:
        with open(filename, "r") as f:
            for line in f:
                line = line.strip()
                if line:
                    parts = line.split(",")
                    if len(parts) == 5:
                        product = {
                            'name': parts[0],
                            'brand': parts[1],
                            'qty': int(parts[2]),
                            'cost': float(parts[3]),
                            'country': parts[4]
                        }
                        products.append(product)
        return products
    except FileNotFoundError:
        print("Products file not found. Starting with empty inventory.")
        return []
    except Exception as e:
        print(f"Error loading products: {e}")
        return []
```

write.py

```
"""
write.py - Module for handling all write operations in the cosmetics shop system
```

```
Includes functions for saving products and generating invoices
"""

import datetime

def save_products(products, filename="products.txt"):
    """
    Saves product data to a text file

    Args:
        products (list): List of product dictionaries
        filename (str): Name of file to save to (default: products.txt)
    """
    try:
        with open(filename, "w") as f:
            for p in products:
                line = f"{p['name']},{p['brand']},{p['qty']},{p['cost']},{p['country']}\n"
                f.write(line)
    except Exception as e:
        print(f"Error saving products: {e}")

def generate_sale_invoice(customer_name, invoice_lines, total_amount):
    """
    Generates a sales invoice file

    Args:
        customer_name (str): Name of customer
        invoice_lines (list): List of purchased items
        total_amount (float): Total sale amount
    """
    try:
```

```
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"invoice_sale_{customer_name}_{timestamp}.txt"

    with open(filename, "w") as f:
        # Header section
        f.write("="*50 + "\n")
        f.write("WE CARE COSMETICS\n".center(50) + "\n")
        f.write("DURBARMARG, KATHMANDU\n".center(50))
        f.write("Phone: 9842322211\n".center(50))
        f.write("="*50 + "\n\n")

        # Customer info
        f.write(f"Customer: {customer_name}\n")
        f.write(f"Date: {datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")
        f.write("-"*50 + "\n")

        # Items header
        f.write("ITEM".ljust(20))
        f.write("BRAND".ljust(15))
        f.write("QTY".center(8))
        f.write("PRICE".rjust(10))
        f.write("TOTAL".rjust(12) + "\n")
        f.write("-"*50 + "\n")

        # Items list
        for line in invoice_lines:
            parts = line.split("\t")
            if len(parts) >= 4:
                f.write(parts[0].ljust(20))  # Product name
                f.write(parts[1].ljust(15))  # Brand
                f.write(parts[2].center(8))  # Quantity
```

```
                f.write(parts[3].split()[0].rjust(10))  # Price
                if len(parts) > 3:
                    total = parts[3].split()[-1].replace("Rs.","").strip()
                    f.write(total.rjust(12) + "\n")  # Total

            # Footer
            f.write("-"*50 + "\n")
            f.write("TOTAL AMOUNT:".rjust(40))
            f.write(f"Rs. {total_amount:.2f}".rjust(10) + "\n")
            f.write("="*50 + "\n")

        print(f"\nInvoice successfully saved as: {filename}")
        return True
    except Exception as e:
        print(f"Error generating invoice: {e}")
        return False

def generate_restock_invoice(vendor_name, invoice_lines, total_amount):
    """
    Generates a restock invoice file

    Args:
        vendor_name (str): Name of supplier/vendor
        invoice_lines (list): List of restocked items
        total_amount (float): Total restock amount
    """
    try:
        timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
        filename = f"invoice_restock_{vendor_name}_{timestamp}.txt"

        with open(filename, "w") as f:
```

```
            f.write("="*50 + "\n")
            f.write("WE CARE COSMETICS - RESTOCK\n".center(50))
            f.write("="*50 + "\n\n")
            f.write(f"Supplier: {vendor_name}\n")
            f.write(f"Date: {datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")
            f.write("-"*50 + "\n")
            f.write("ITEM".ljust(25))
            f.write("QTY".ljust(10))
            f.write("UNIT PRICE".ljust(15))
            f.write("TOTAL".rjust(10) + "\n")
            f.write("-"*50 + "\n")

            for line in invoice_lines:
                parts = line.split("\t")
                if len(parts) >= 3:
                    f.write(parts[0].ljust(25))
                    f.write(parts[2].split()[0].ljust(10))
                    f.write(parts[3].split(":")[1].strip().ljust(15))
                    f.write(parts[3].split("Subtotal:")[1].strip().rjust(10) + "\n")

            f.write("-"*50 + "\n")
            f.write("TOTAL RESTOCK COST:".rjust(40))
            f.write(f"Rs. {total_amount:.2f}".rjust(10) + "\n")
            f.write("="*50 + "\n")

        print(f"\nRestock invoice saved as: {filename}")
        return True
    except Exception as e:
        print(f"Error generating restock invoice: {e}")
        return False
```