UNIVERSITY OF MAURITIUS

ICDT 1201Y
COMPUTER PROGRAMMING ASSIGNMENT

CASHIER SYSTEM FOR A SMALL SUPERMARKET

Toushal Sampat
2413826

TABLE OF CONTENT

Introduction ____ 2
o The Problem ____ 2
o Proposed Solution ____ 2
o Scope ____ 2
o Distribution of Task ____ 3
Solution Design ____ 3
o Pseudocodes/Flowcharts ____ 3
Implementation & Testing ____ 6
o System requirements ____ 6
o Implementation of each component (different modules) ____ 7
o Test Plan & Scenarios (Unit, integration, system) ____ 9
o Sample Screenshots ____ 10
Conclusion ____ 13
o Achievements ____ 13
o Challenges and problems encountered ____ 13
o Future Work ____ 13
 References ____ 14
 Appendix ____ 15

1. Intoduction

1.1 The Problem

For long manual checkout in a small supermarket has been very slow, inefficient in many ways- errors when calculating the total for each client. It has been a bottleneck for checkout operators since they must look up for each product’s price in paper sheets/ lists and can be tiring and frustrating if a particular product is not recorded. Moreover, if the product’s price has changed, that price must also be manually overwritten in the checkout lists.

Checkouts must make use of calculators to calculate client sales. This is prone to errors and mistakes since that checkout must be aware of what they are doing. Also, not all checkouts are able to work perfectly with a calculator as some calculators are different to other calculators and this can cause confusion, and mistakes arise.

Furthermore, if a client is purchasing lots of products a one time, that checkout along with the clients in the queue will be frustrated and tired of waiting since the checkout process is taking too long.

1.2 Proposed Solution

The application will be implemented in Python and will consists of OOPs and files. Files will be used to access, read, write, and store data. This will be available in the local storage and can be easily edited to update certain data if or when required.

The proposed solution is to make a small application that will make the checkout process smoother and faster for everyone involved. It will be a general application that will be standard to anyone and that can be used with little training. It will also be less prone to making errors when calculating the total compared to the old method used by the cashier.

The application will take the user -cashier- inputs (product barcodes and quantity) and will retrieve information (name and price) from the appropriate files. It will then calculate the total for that sale. Once it is done, it will show the total and will allow the cashier to input the amount that is paid by the client and, if required, will compute and display the difference between what the client has paid and the total amount that the cashier will have to return back.

The application will allow the user to add new products to the product files that will later be read during a sale. It will also allow the user to modify the price of any product easily. Moreover, it will allow the cashier to make refund of any product that exists in the product file and, once the product is refunded, it will save the refund details in a refund file. The cashier will be able to generate report of today’s sale and finance using data from both the sale and refund file. A list of all products details can be printed to view all sellable products.

1.3 Scope (Boundaries and Functionalities)

The boundaries of the application:

1. It will be a basic POS that will handle basic checkout operation.
2. The application is designed to be used by a single cashier at any time.
3. The application will not have any login authentication.
4. The application will use files to access, store, and write data.
5. The application will run in the terminal window of the local computer.

The application will have the following functionalities**:**

1. Product Management that will be used to add new product or update existing product price from the product file.
2. Sale system that will take in certain parameter (product barcode and quantity) and read the product file to get required data.
3. Payment system that will take the retrieved data of that product along with quantity and compute a total.
4. Paying system that will take as a parameter the amount the client is paying and will calculate and display the difference in cash that needs to be returned.
5. Receipt system that will print the sale details of the latest sale
6. Sale History system that will save all sales in a file.
7. Refund system that will allow the cashier to process refunds of products and save the refund details in a refund file.
8. Today's report system that will generate a report of today's sales and refunds and also provide a net sale.

1.4 Distribution of work

Toushal Sampat- All

2. <u>Solution Design</u>

```
START
IMPORT datetime
today = "datetime.today's date"
CLASS Cashier_system
    FUNCTION Options
        PRINT options
        INPUT option

        IF option IS "Add" THEN CALL add_product
        ELSE IF option IS "Update" THEN CALL update_price
        ELSE IF option IS "Sale" THEN CALL sale
        ELSE IF option IS "Last Sale" THEN CALL last_sale
        ELSE IF option IS "Products Details" THEN CALL products_details
        ELSE IF option IS "Refund" THEN CALL refund
        ELSE IF option IS "Today Report" THEN CALL today_report
        ELSE PRINT Application Ended and CLOSE
            END

    FUNCTION add_products
        OPEN product_file IN a+ MODE
        product_file READLINES
        INPUT barcode
        IF barcode IS EMPTY THEN
            CLOSE product_file
            RETURN Cashier_system.Options
        ELSE IF barcode IS NOT NUMERIC THEN
            CLOSE product_file
            RETURN Cashier_system.Options

        FOR products IN product_file
            IF barcode EXISTS THEN
                CLOSE product_file
                RETURN Cashier_system.Options
        INPUT product_name
        IF product_name IS EMPTY THEN
            CLOSE product_file
            RETURN Cashier_system.Options
        WHILE TRUE
            INPUT product_price
            IF product_price IS EMPTY THEN
                CLOSE product_file
                RETURN Cashier_system.Options
            ELSE IF product_price < 0 or product_price IS NOT NUMERIC THEN
```

```
                PRINT input error message
                CONTINUE
            BREAK
        SAVE product details IN LIST AS [product_barcode, product_name, product_price]
        WRITE LIST IN product_file
        CLOSE product_file
        RETURN Cashier_system.Options

FUNCTION update_price
        OPEN product_file IN r+ MODE
        product_file READLINES
        INPUT barcode

        IF barcode IS EMPTY THEN
            CLOSE product_file
            RETURN Cashier_system.Options
        ELSE IF barcode IS NOT NUMERIC THEN
            CLOSE product_file
            RETURN Cashier_system.Options
        FOR products IN product_file
            IF barcode IS == AS product_barcode IN product_file THEN
                INPUT new_price
                IF new_price IS EMPTY THEN
                    CLOSE product_file
                    RETURN Cashier_system.Options
                IF new_price IS LESS THAN 0 THEN
                    CLOSE product_file
                    RETURN Cashier_system.Options
                product_price = new_price
        IF product NOT IN product_file THEN
            PRINT barcode does not exists message
        SAVE new_price IN product_file
        CLOSE product_file
        RETURN Cashier_system.Options

FUNCTION sale
        OPEN product_file IN r MODE
        OPEN sale_file IN a MODE
        product_file READLINES
        INPUT barcode
        receipt AS [today]
        sale_end = FALSE
        total = 0
        IF barcode IS EMPTY THEN
            CLOSE product_file
            CLOSE sale_file
            RETURN Cashier_system.Options
        ELSE IF barcode IS NOT NUMERIC THEN
            CLOSE product_file
            CLOSE sale_file
            RETURN Cashier_system.Options
        IF sale_end == FALSE THEN
            FOR lines IN product_file
                IF barcode EQUAL product_barcode IN product_file THEN
                    INPUT quantity
                    IF quantity IS EMPTY THEN
                        quantity EQUAL 1
                    ELSE IF quantity LESS THAN 1 THEN
                        PRINT quantity must be greater 0 message
                        CONTINUE
                    sum = 0
                    sum = product_price * quantity
                    total = total + sum
                    SAVE sale_detail AS [product_barcode, product_name, product_price, quantity , sum]
                    APPEND sale_detail IN receipt
                    INPUT barcode
                    IF barcode IS EMPTY THEN
                        sale_end = TRUE
                    BREAK
            PRINT total
            remaining = total
            total_cash = 0
            Change = 0
            IF sale_end == TRUE THEN
                WHILE remaining > 0
                    INPUT cash
                    IF cash IS EMPTY THEN
                        cash = remaining
                        total_cash += cash
                        change = 0.00
                        PRINT exact cash
                        BREAK
                    ELSE IF
                        IF cash < 0 THEN
                            PRINT cash must be +ve
                            CONTINUE
                        total_cash += cash
                        IF cash < remaining THEN
                            remaining -= cash
                            PRINT remaining
                        ELSE
                            change = cash – remaining
                            remaining = 0
                            BREAK
            SAVE payment AS [total, total_cash, change]
            APPEND payment TO receipt
            WRITE receipt to sale_file
            CLOSE product_file
            CLOSE sale_file
            RETURN Cashier_system.Options

FUNCTION last_sale
        OPEN sale_file IN r MODE
        sale_file READLINES

        last_line = sale.file[-1]
        FOR items IN last_line
            PRINT barcode, name, price, quantity, total
        PRINT Total
        PRINT Cash
        PRINT Change
        CLOSE sale_file
        RETURN Cashier_system.Options

FUNCTION product_details
        OPEN product_file IN r MODE
        product_file READLINES
        FOR line in product_file
            barcode, name, price = line
            PRINT barcode, name, price
        CLOSE product_file
        RETURN Cashier_system.Options

FUNCTION refund
        OPEN product_file IN r MODE
        OPEN refund_file IN a MODE
        product_file READLINES
        INPUT user_barcode
        IF barcode IS EMPTY THEN
            CLOSE product_file
            CLOSE refund_file
            RETURN Cashier_system.Options
        IF barcode IS NOT NUMERIC THEN
            PRINT barcode is not numeric message
            CLOSE product_file
            CLOSE refund_file
            RETURN Cashier_system.Options
        ELSE
            found = FALSE
            FOR barcode IN product_file
                IF barcode IN product_file == user_barcode THEN
                    found = TRUE
                    INPUT quantity
                    IF quantity IS EMPTY THEN
                        CLOSE product_file
                        CLOSE refund_file
                        RETURN Cashier_system.Options
                    refund_amount = product_price * quantity
                    SAVE refund_details AS [today, barcode, product_name, price, quantity, refund_amount]
                    WRITE refund_details TO refund_file
                    PRINT refund_amount
                    BREAK
            IF found == FALSE THEN
                PRINT barcode not exists
        CLOSE product_file
        CLOSE refund_file
        RETURN Cashier_system.Options

FUNCTION today_report
        OPEN sale_file IN r MODE
        OPEN refund_file IN r MODE
        sale_file READLINES
        refund_file READLINES
        Total = 0
        Sale_count
        FOR sale IN sale_file
                IF sale_date IS today and sale_total IS NOT 0 THEN
                        Total += sale_total
                        Sale_count += 1
        Refund = 0
        Refund_count = 0
        FOR refund IN refund_file
                IF refund_date IS today THEN
                        Refund += refund_amount
                        Refund_count += 1
        PRINT Total
        PRINT Sale_count
        PRINT Refund
        PRINT Refund_count
        PRINT net_total = Total – Refund
        CLOSE sale_file
        CLOSE refund_file
        RETURN Cashier_system.Options
```

3. Implementation & Testing

3.1 System Requirements

The cashier system is designed to run on minimal hardware, making it suitable for small supermarket that uses basic computer setups.

Hardware Requirements

- Any basic computer architecture or tablet having the minimum of 1 GHz processor, 1 GB RAM.
- At least 100 MB of free storage for application and text files.

Functional Requirements

- The system shall add new products by inputting a new barcode, product name, and price and store them in a file.
- The system shall process sale by inputting a barcode and quantities, and retrieves the price associated to that barcode and calculate the total cost and ends the sale by pressing ENTER.
- The system shall allow for cash input, calculate change based on total, display all details of the payment and save the sale details in the sale file.
- The system shall take in as input a barcode and a price, check if barcode exists in product file and then change the price of the existing product to have the new price.
- The system shall display details of the latest sale.
- The system shall take in as input a barcode and quantities, then retrieve information related to that barcode in the product file, then calculate and display the refund information for that product and save the refund details of that product.
- The system shall generate a daily report by retrieving information from sale and refunds files.
- The system shall display the information of all products stored in the product file.

Non-Functional Requirements

- The system shall take less than 5 minutes to learn and use.
- The system shall make any operation with at most 2 seconds.

- The system shall handle missing files and display any errors.
- The system shall validate any input to avoid any errors in the future.

3.2 Implementation of each component

The cashier system is implemented as a class using object-oriented programming having different method that have different functions that they execute.

3.2.1 Options Menu (`Options()`)

- **Purpose:** It is the main menu that displays the options and allow cashier to select an operation and redirects them to such operation.
- **Implementation:** This method uses `print()` to prints out the options like "A - Add Product" and "S - Sale" that the cashier can select using the `input("Enter your option: ")`. The `.lower()` is use to make the input to lowercase and uses `if-elif-else` to match with the validation like "a" or "add product", to then calls the method like `Add_product()` or `Sale()`. IF input is invalid, it prints "INVALID option entered" and recalls itself. If an empty input is detected "", the application prints "Application ENDED and CLOSE" and exits with `exit()`.
- **Key Features:** Support dual input style and loops back when encounter errors

3.2.2 Add product (`Add_product()`)

- **Purpose:** To add new products to the product file for later use.
- **Implementation:** Opens `product_file.txt` in `a+` (read and append) mode. Prompts for a barcode with `input("Enter a product barcode or press ENTER to EXIT: ")`. If input is empty "", it prints "ADD operation ENDED." and return to `Options()`. It uses `.isnumeric()` to verify if input is not, it prints "Barcode should be only numeric. ADD operation ABORTED." and return to `Options()`. If it is, then it check if barcode is already existing in file and if so it print "Barcode ALREADY EXISTS. Product name: ", else it prompts for input of name and price using `input("Enter product name OR press ENTER to ABORT Add: ")` and `input("Enter product price OR press ENTER to ABORT Add: ")`. If both input are empty, it prints "ADD operation ENDED." and return to `Options()`. If price is -ve or not numeric, it print "Price cannot be negative(-ve)." or "Price should be numeric(1234)." and loops until is valid. Once done, it prints the added product as "Product Added: [barcode, name, price]".
- **Key Features:** Validates inputs and confirm addition.

3.3.3 Update price (`Update_price()`)

- **Purpose:** Update the price of an existing product.
- **Implementation:** Opens `product_file.txt` in `r+` (read and write) mode. Prompts for a barcode with `input("Enter a product barcode or press ENTER to EXIT: ")`. If input is empty "", it prints "UPDATE operation ENDED." and return to `Options()`. It uses `.isnumeric()` to verify if input is not, it prints "Barcode should be only numeric. UPDATE operation ABORTED." and return to `Options()`. For a match, it prompts the user to input a price `input("Enter a new price or press ENTER to END: ")`. If price is -ve it prints "Price cannot be negative. UPDATE ENDED. No price change." and return to `Options()`. Else, it update the price and prints the updated list [barcode, price, new_price] and rewrites the file. If no match found, it print "Barcode not found." and return to `Options()`.
- **Key Features:** Validate exists and confirms update with print.

3.3.4 Sale (`Sale()`)

- **Purpose:** Handles the sale of products, quantity, and also handles the transactions (cash handling) and save the sale details in file.
- **Implementation:** Opens `product_file.txt` in `r` (read) mode and opens `sale_file.txt` in `a` (append) mode. Starts with timestamps list, and prompts for a barcode with `input("Enter`

product barcode or press ENTER to END Sale: "). If input is empty, it return to Options() or if not numeric, it prints "Barcode must be numeric(e.g 1234).". For a match, it prompts for quantity input("Enter qty or press ENTER if prod_qty is 1: "). If empty it default the quantity to 1 and if quantity is invalid, it prints "Quantity must be greater 0." or "Quantity must be +ve and numeric." It calculates the subtotal of that product using the product price and input quantity and repeats until an empty "" barcode is input and it ends the sale. All selling product is save in a list. It prints the total "Total Amount = Rs %0.2f" % Total. Then it prompts for cash input("Enter cash amount: "). If empty "", it assumes that the exact payment, prints "Exact amount of cash paid." and ends. If negative or invalid it prints "Cash cannot be -ve." or "Invalid cash amount entered.". For partial payment, it prints "Remaining Rs %0.2f" % remainig or for excess "Return Rs %0.2f" % change. It appends the payment details to the list of sale which is then append to the sale_file.txt.

- **Key Features:** Loops for multiple items and confirms totals. Supports flexible cash inputs with feedback. Save details of each sale in a file.

3.3.5 Receipt (Last_sale())

- **Purpose:** Display the latest sale receipt.
- **Implementation:** Opens sale_file.txt in r (read) mode, it reads the last line last_line = eval(lines[-1].strip()) and breaks the list into lines with all details of the sale. Prints the timestamps, then the product details and lastly the payment details.
- **Key Features:** Display a proper receipt of the sale.

3.3.6 Refund (Refund())

- **Purpose:** Process refund of a product.
- **Implementation:** Opens product_file.txt in r (read) mode and opens refund_file.txt in a (append) mode. Prompt for a barcode input("Enter Barcode to refund or press ENTER to END: "). If input is empty, file is close, prints "REFUND operation ENDED." and return to Options(), or if barcode is not numeric it prints "Barcode should be numeric.". For a match, it prompts for quantity input("Enter quantity or press ENTER to END: ") and if empty it prints "REFUND operations CANCELLED." and returns to Options(). Otherwise, it calculate the refund using the price and quantity and prints "Refund completed. Return Rs", refund_sum. It appends the refund details in the file as [refund_date, product[0], product[1], product[2], user_refund_qty, float(refund_sum)]. If no barcode matches, it prints "Barcode does not exist in record "
- **Key Features:** Validate and confirm refund, and display refund amount.

3.3.7 Product Details (Products_details())

- **Purpose:** Lists all details of all products in the file.
- **Implementation:** Opens product_file.txt in r (read) mode and parses each line as barcode, name, price = prod_details. It then prints each as barcode name price (e.g. 12345, Milk 2kg, 350.0) using the print("{:<15}{:<30}Rs {:>7.2f}".format(barcode, name, price)). Once completed it close the file and return to Options(). If an error occurs when reading the lines, it print "Error in reading sale details from file. Line(s) in file is corrupt.".
- **Key Features:** Displays all sellable products.

3.3.8 Today's Report (Today_report())

- **Purpose:** Summarizes and display daily sales and refund.
- **Implementation:** Opens product_file.txt and refund_file.txt in r (read) mode. Calculates totals and counts for today's sales and refunds. It filters all lines that have today's date for sale as if sale[0][0] == y_today and for refund as if refund[0] == y_today, then does the calculation. It calculates the net_total = Total_cash - refund_cash. Prints the details as

("Today Sale Report: Rs",Total_cash), ("Today sale made: ", sale_count), ("Today Refund Total Rs", refund_cash), ("Total refund made: ", refund_count), ("Net Sale Rs ", net_total).

- **Key Features:** Provide a concise daily summary.

3.4 Test Plans & Scenarios

Test plan & scenarios are performed in 3 levels-Unit, Integration and System- to ensure that the application is functioning properly, interacting with other modules and able to read and write in text files.

3.4.3 Unit Testing

1. `Options`
   Test 1: Valid input. Input: a or add product.
   Test 2: Invalid input. Input: q.
   Test 3: Exit. Input: “” (ENTER).
2. `Add_product`
   Test 1: Add a product. Input: barcode 6091231231231, name Milk 2Kg, price 255.5.
   Test 2: Duplicate barcode. Input: barcode 6091231231231.
   Test 3: Invalid barcode. Input: qwerty
   Test 4: Negative price. Input: barcode 6097897897897, name Eggs *24pcs, price -240.
3. `Update_price`
   Test 1: Update price. Input: barcode 6091231231231, price 300.0.
   Test 2: Invalid barcode. Input: 999.
   Test 3: Negative price. Input: barcode 6091231231231, price -300.
4. `Sale`
   Test 1: Single item sale. Input: barcode 6091231231231, qty 2, cash 1000.
   Test 2: Early exit. Input: barcode “” (ENTER).
   Test 3: Partial payment. Input: barcode 6097897897897, qty 1, cash 200, then 100.
   Test 4: Invalid barcode. Input: xyz.
5. `Last_sale`
   Test 1: Display receipt.
   Test 2: Empty file.
6. Refund
   Test 1: Refund product. Input: barcode 6091231231231, qty 2.
   Test 2: Invalid barcode. Input: 999.
7. `Today_report`
   Test 1: Generate report.
   Test 2: No sales/refunds.
8. `Product_details`
   Test 1: List products.
   Test 2: Corrupt line. Input: Add invalid to file.

3.4.4 Integration Testing

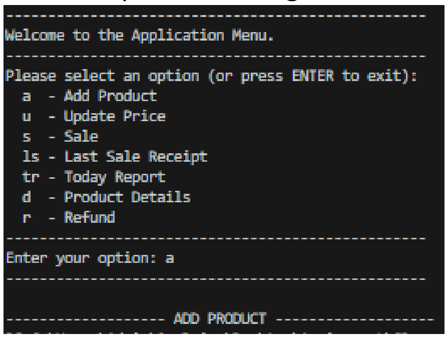
1. **Scenario 1: Product lifecycle**
   Steps: Add product (barcode 6091212121212, name Chocolate, price 25.0), make a sale (barcode 6091212121212, qty 2, cash 100), check Last_sale, update price to 20, make another sale (qty 1, cash 25).
2. **Scenario 2: Sale and refund flow**
   Steps: Make two sales (barcode 6091212121212, qty 2, cash 100; qty 1, cash 25), refund (barcode 6091212121212, qty 1), run Today_report.

3.4.5 System Testing

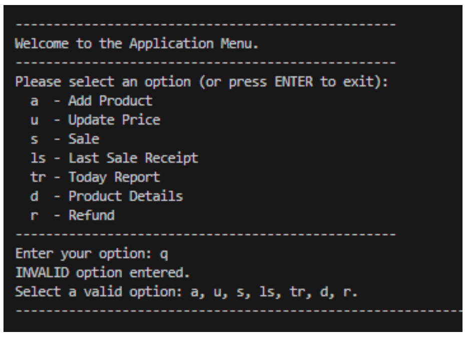
Scenario: Full day simulation
Precondition: Empty files.
Steps: Add 10 products (e.g., 6091231231231:”Milk”:255.5, 456:”Bread”:1.50, etc.), make 10 sales (e.g., 6 items, 7 items, 3 items,….), refund 2 items (e.g., 6091231231231,qty 1….), run Today_report.

3.5 Sample Screenshot

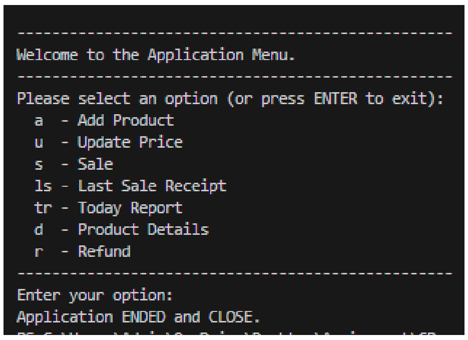
- Options Testing



```
--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: a
--------------------------------------------------

------------------ ADD PRODUCT ------------------
```

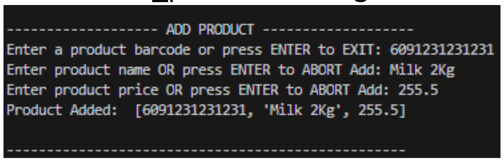


```
--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: q
INVALID option entered.
Select a valid option: a, u, s, ls, tr, d, r.
--------------------------------------------------
```

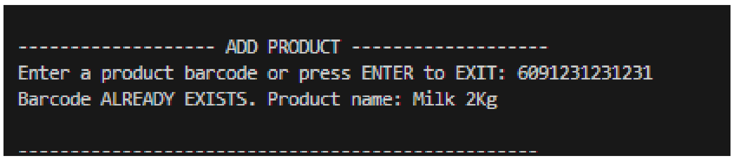


```
--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option:
Application ENDED and CLOSE.
```

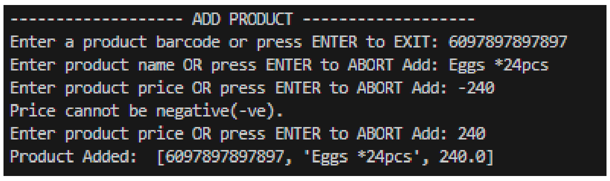
- Add_product Testing



```
------------------ ADD PRODUCT ------------------
Enter a product barcode or press ENTER to EXIT: 6091231231231
Enter product name OR press ENTER to ABORT Add: Milk 2Kg
Enter product price OR press ENTER to ABORT Add: 255.5
Product Added:  [6091231231231, 'Milk 2Kg', 255.5]

--------------------------------------------------
```

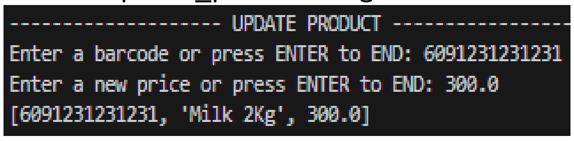


```
------------------- ADD PRODUCT -------------------
Enter a product barcode or press ENTER to EXIT: 6091231231231
Barcode ALREADY EXISTS. Product name: Milk 2Kg

--------------------------------------------------
```

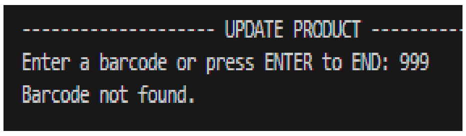


```
------------------ ADD PRODUCT ------------------
Enter a product barcode or press ENTER to EXIT: 6097897897897
Enter product name OR press ENTER to ABORT Add: Eggs *24pcs
Enter product price OR press ENTER to ABORT Add: -240
Price cannot be negative(-ve).
Enter product price OR press ENTER to ABORT Add: 240
Product Added:  [6097897897897, 'Eggs *24pcs', 240.0]
```

- Update_price Testing

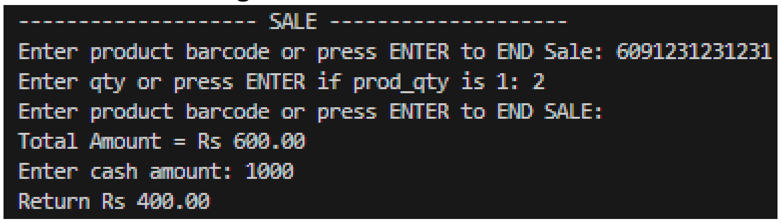


```
------------------- UPDATE PRODUCT ---------------
Enter a barcode or press ENTER to END: 6091231231231
Enter a new price or press ENTER to END: 300.0
[6091231231231, 'Milk 2Kg', 300.0]
```

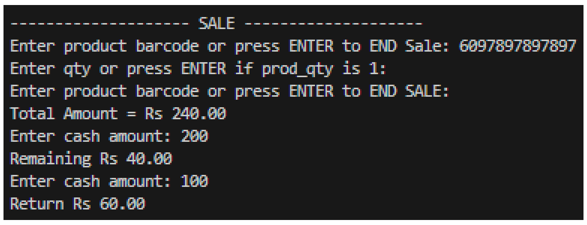


```
------------------- UPDATE PRODUCT ---------
Enter a barcode or press ENTER to END: 999
Barcode not found.
```

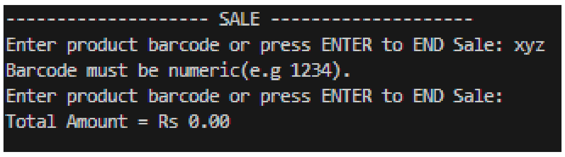
- Sale Testing



```
-------------------- SALE --------------------
Enter product barcode or press ENTER to END Sale: 6091231231231
Enter qty or press ENTER if prod_qty is 1: 2
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 600.00
Enter cash amount: 1000
Return Rs 400.00
```

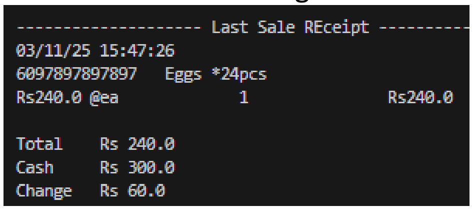


```
------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: 6097897897897
Enter qty or press ENTER if prod_qty is 1:
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 240.00
Enter cash amount: 200
Remaining Rs 40.00
Enter cash amount: 100
Return Rs 60.00
```

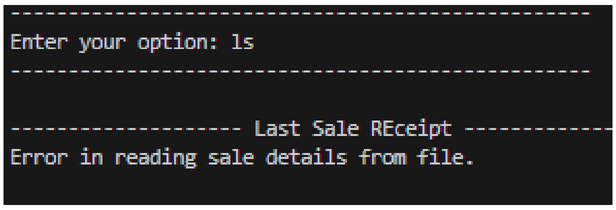


```
------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: xyz
Barcode must be numeric(e.g 1234).
Enter product barcode or press ENTER to END Sale:
Total Amount = Rs 0.00
```

- Last Sale Testing

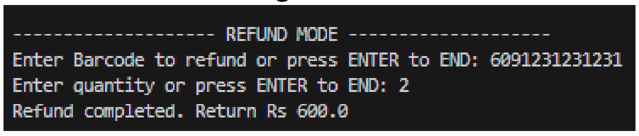


```
------------------- Last Sale REceipt ---------
03/11/25 15:47:26
6097897897897    Eggs *24pcs
Rs240.0 @ea             1              Rs240.0

Total      Rs 240.0
Cash       Rs 300.0
Change     Rs 60.0
```

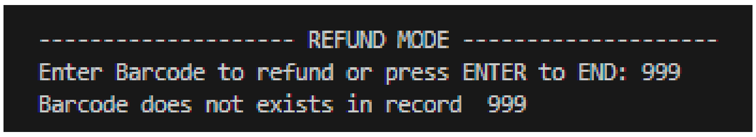


```
--------------------------------------------------
Enter your option: ls
--------------------------------------------------

-------------------- Last Sale REceipt ------------
Error in reading sale details from file.
```

- Refund Testing

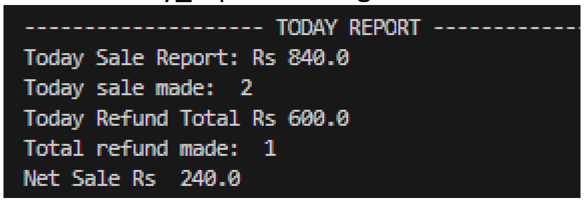


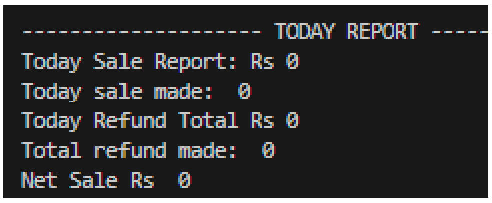
```
------------------- REFUND MODE -------------------
Enter Barcode to refund or press ENTER to END: 6091231231231
Enter quantity or press ENTER to END: 2
Refund completed. Return Rs 600.0
```



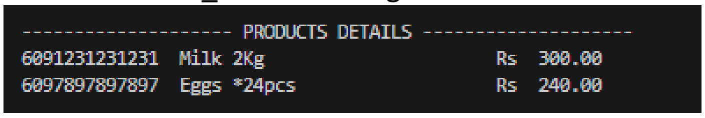
```
-------------------- REFUND MODE --------------------
Enter Barcode to refund or press ENTER to END: 999
Barcode does not exists in record  999
```

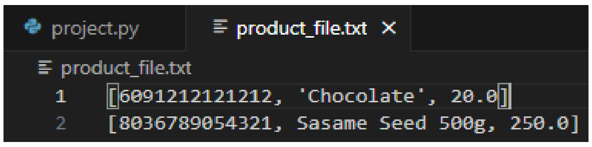
- Today_report Testing


```
-------------------- TODAY REPORT ------------
Today Sale Report: Rs 840.0
Today sale made:  2
Today Refund Total Rs 600.0
Total refund made:  1
Net Sale Rs  240.0
```

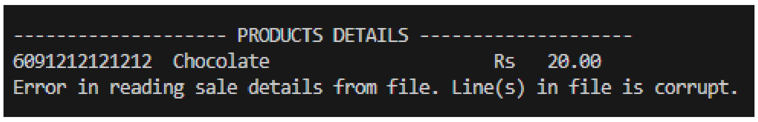

```
-------------------- TODAY REPORT ----
Today Sale Report: Rs 0
Today sale made:  0
Today Refund Total Rs 0
Total refund made:  0
Net Sale Rs  0
```

- Product_details Testing

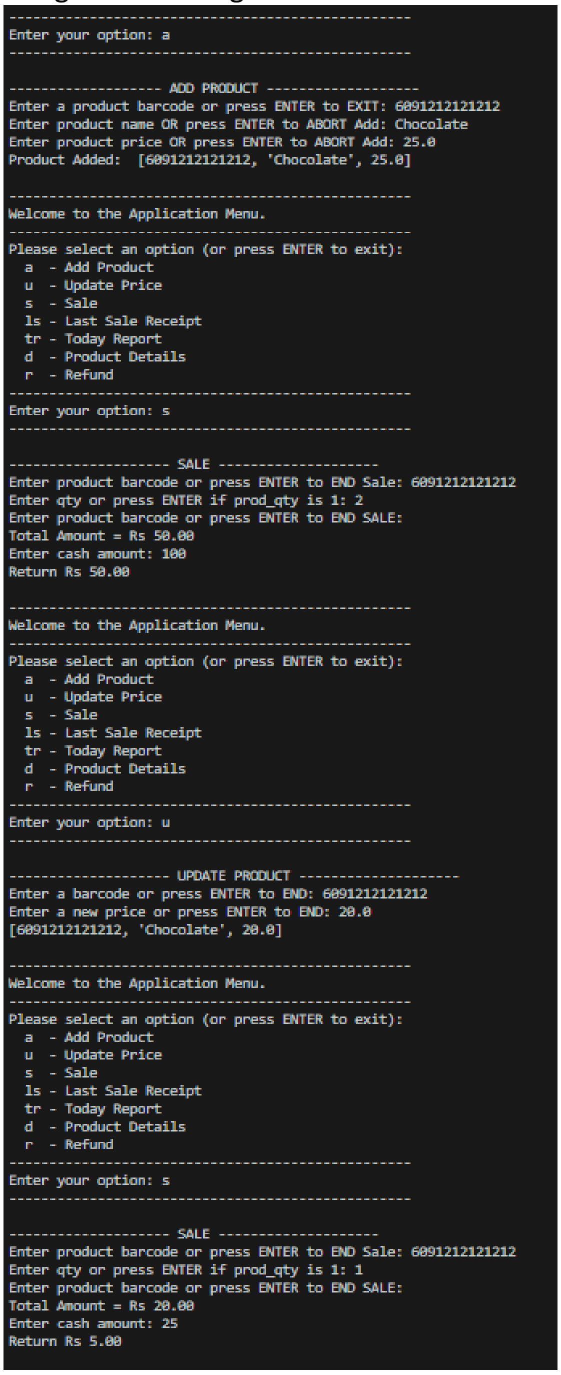

```
------------------- PRODUCTS DETAILS -------------------
6091231231231  Milk 2Kg                   Rs  300.00
6097897897897  Eggs *24pcs                Rs  240.00
```


project.py   product_file.txt ×

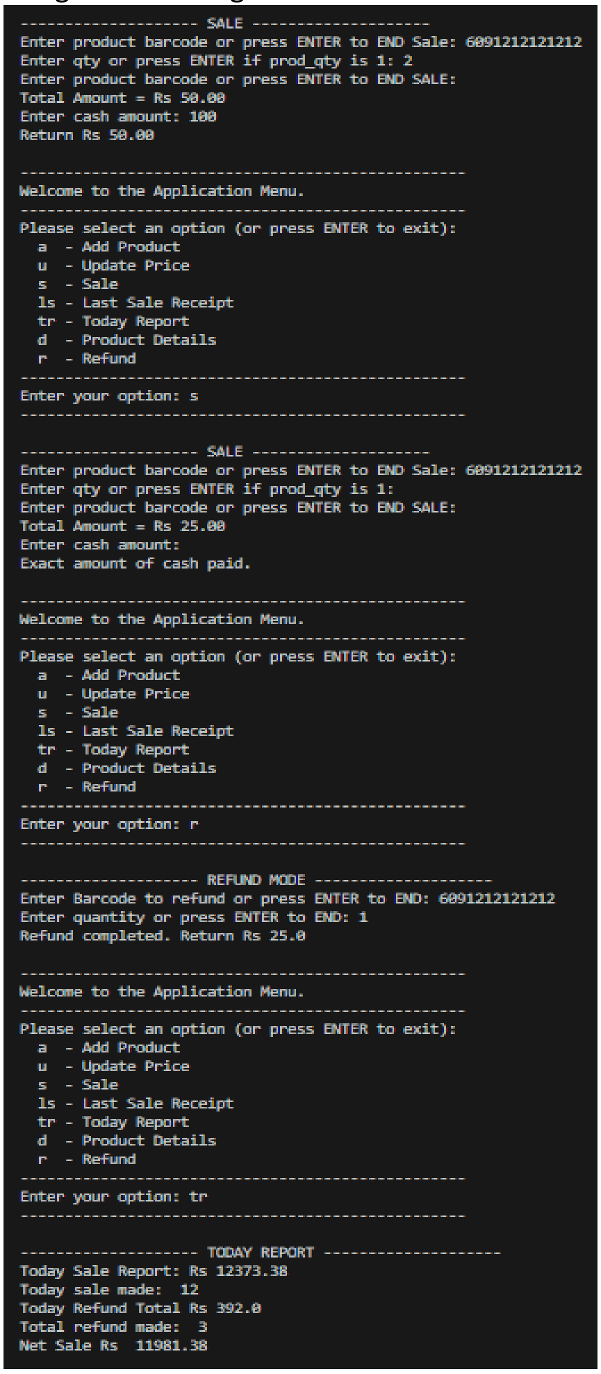
product_file.txt
```
[6091212121212, 'Chocolate', 20.0]
[8036789054321, Sasame Seed 500g, 250.0]
```


```
------------------- PRODUCTS DETAILS -------------------
6091212121212  Chocolate                  Rs   20.00
Error in reading sale details from file. Line(s) in file is corrupt.
```

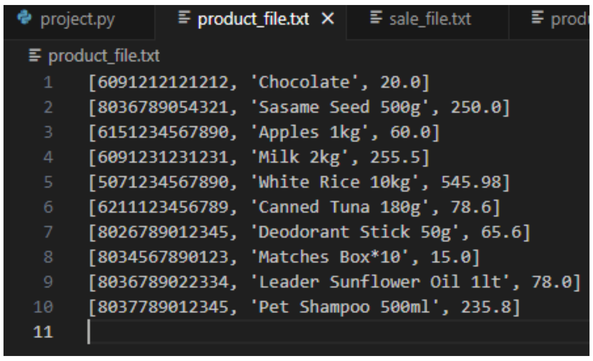
Integration Testing- Scenario 1


```
--------------------------------------------------
Enter your option: a
--------------------------------------------------

------------------ ADD PRODUCT ------------------
Enter a product barcode or press ENTER to EXIT: 6091212121212
Enter product name OR press ENTER to ABORT Add: Chocolate
Enter product price OR press ENTER to ABORT Add: 25.0
Product Added:  [6091212121212, 'Chocolate', 25.0]

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: s
--------------------------------------------------

------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: 6091212121212
Enter qty or press ENTER if prod_qty is 1: 2
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 50.00
Enter cash amount: 100
Return Rs 50.00

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: u
--------------------------------------------------

------------------ UPDATE PRODUCT ------------------
Enter a barcode or press ENTER to END: 6091212121212
Enter a new price or press ENTER to END: 20.0
[6091212121212, 'Chocolate', 20.0]

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: s
--------------------------------------------------

------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: 6091212121212
Enter qty or press ENTER if prod_qty is 1: 1
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 20.00
Enter cash amount: 25
Return Rs 5.00
```

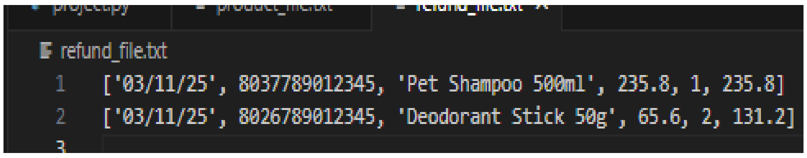
Integration Testing- Scenario 2


```
------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: 6091212121212
Enter qty or press ENTER if prod_qty is 1: 2
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 50.00
Enter cash amount: 100
Return Rs 50.00

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: s
--------------------------------------------------

------------------- SALE -------------------
Enter product barcode or press ENTER to END Sale: 6091212121212
Enter qty or press ENTER if prod_qty is 1:
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 25.00
Enter cash amount:
Exact amount of cash paid.

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: r
--------------------------------------------------

------------------- REFUND MODE -------------------
Enter Barcode to refund or press ENTER to END: 6091212121212
Enter quantity or press ENTER to END: 1
Refund completed. Return Rs 25.0

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: tr
--------------------------------------------------

------------------- TODAY REPORT -------------------
Today Sale Report: Rs 12373.38
Today sale made:  12
Today Refund Total Rs 392.0
Total refund made:  3
Net Sale Rs  11981.38
```

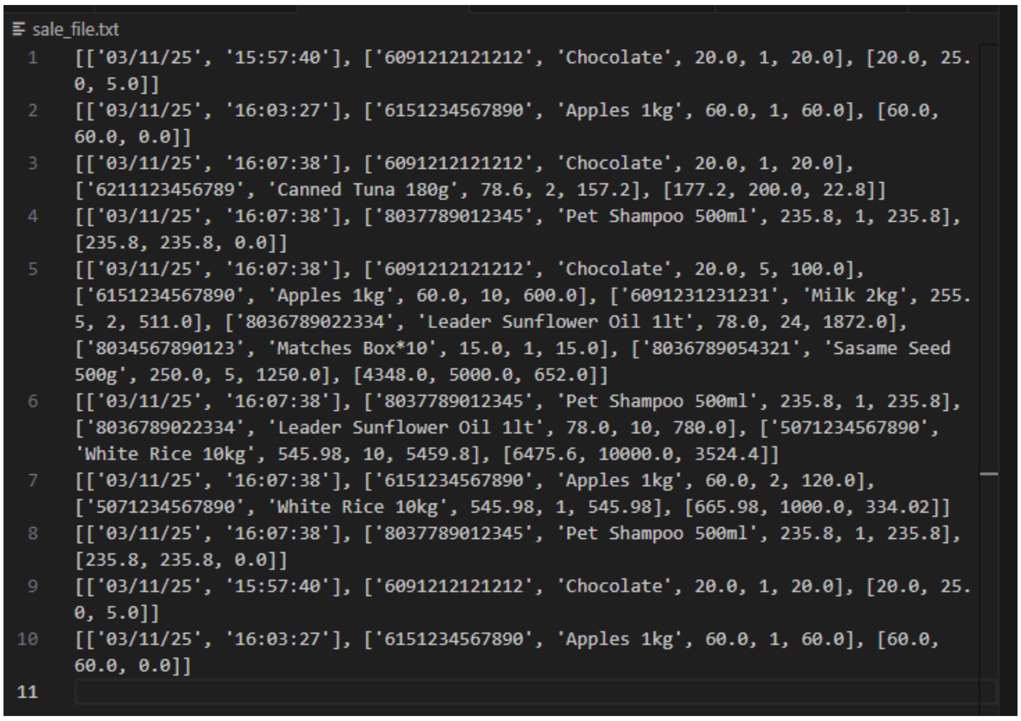
- System Testing



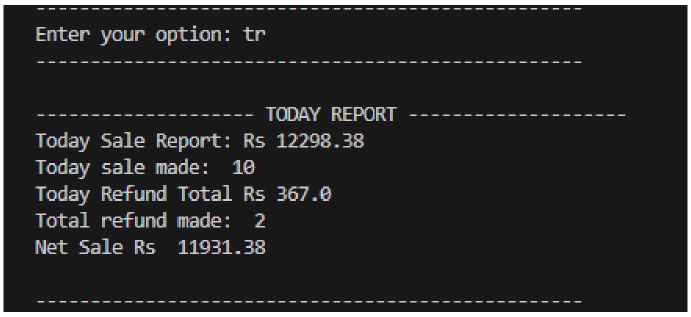
```
product_file.txt
[6091212121212, 'Chocolate', 20.0]
[8036789054321, 'Sasame Seed 500g', 250.0]
[6151234567890, 'Apples 1kg', 60.0]
[6091231231231, 'Milk 2kg', 255.5]
[5071234567890, 'White Rice 10kg', 545.98]
[6211123456789, 'Canned Tuna 180g', 78.6]
[8026789012345, 'Deodorant Stick 50g', 65.6]
[8034567890123, 'Matches Box*10', 15.0]
[8036789022334, 'Leader Sunflower Oil 1lt', 78.0]
[8037789012345, 'Pet Shampoo 500ml', 235.8]
```



```
refund_file.txt
['03/11/25', 8037789012345, 'Pet Shampoo 500ml', 235.8, 1, 235.8]
['03/11/25', 8026789012345, 'Deodorant Stick 50g', 65.6, 2, 131.2]
```



```
sale_file.txt
[['03/11/25', '15:57:40'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0], [20.0, 25.0, 5.0]]
[['03/11/25', '16:03:27'], ['6151234567890', 'Apples 1kg', 60.0, 1, 60.0], [60.0, 60.0, 0.0]]
[['03/11/25', '16:07:38'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0], ['6211123456789', 'Canned Tuna 180g', 78.6, 2, 157.2], [177.2, 200.0, 22.8]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8], [235.8, 235.8, 0.0]]
[['03/11/25', '16:07:38'], ['6091212121212', 'Chocolate', 20.0, 5, 100.0], ['6151234567890', 'Apples 1kg', 60.0, 10, 600.0], ['6091231231231', 'Milk 2kg', 255.5, 2, 511.0], ['8036789022334', 'Leader Sunflower Oil 1lt', 78.0, 24, 1872.0], ['8034567890123', 'Matches Box*10', 15.0, 1, 15.0], ['8036789054321', 'Sasame Seed 500g', 250.0, 5, 1250.0], [4348.0, 5000.0, 652.0]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8], ['8036789022334', 'Leader Sunflower Oil 1lt', 78.0, 10, 780.0], ['5071234567890', 'White Rice 10kg', 545.98, 10, 5459.8], [6475.6, 10000.0, 3524.4]]
[['03/11/25', '16:07:38'], ['6151234567890', 'Apples 1kg', 60.0, 2, 120.0], ['5071234567890', 'White Rice 10kg', 545.98, 1, 545.98], [665.98, 1000.0, 334.02]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8], [235.8, 235.8, 0.0]]
[['03/11/25', '15:57:40'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0], [20.0, 25.0, 5.0]]
[['03/11/25', '16:03:27'], ['6151234567890', 'Apples 1kg', 60.0, 1, 60.0], [60.0, 60.0, 0.0]]
```



```
---------------------------------------------------
Enter your option: tr
---------------------------------------------------

-------------------- TODAY REPORT --------------------
Today Sale Report: Rs 12298.38
Today sale made:  10
Today Refund Total Rs 367.0
Total refund made:  2
Net Sale Rs  11931.38

---------------------------------------------------
```

4. Conclusion

Achievement

Completing this project, I successfully make a small python program for a Cashier System that uses files to handle data, learn how to properly use OOP and also achieved the expected outputs results of the functions implemented. Through testing, I was able to refine the optimize each module to reduce errors.

Challenges and problems encountered

However, the project was not without its challenges. There were most of the time where data would write in the files but won't read afterward since it was not properly closed. There were times when reading and converting the lines would not output anything since it was not always properly parses. Some problems would sometimes require using other tools to verify if its good or having any bugs. While they were demanding, they provided valuable insights on how to solve similar problems that were happening in other modules.

Future Work

For the future, several improvements can be made to optimize and enhance the performance.

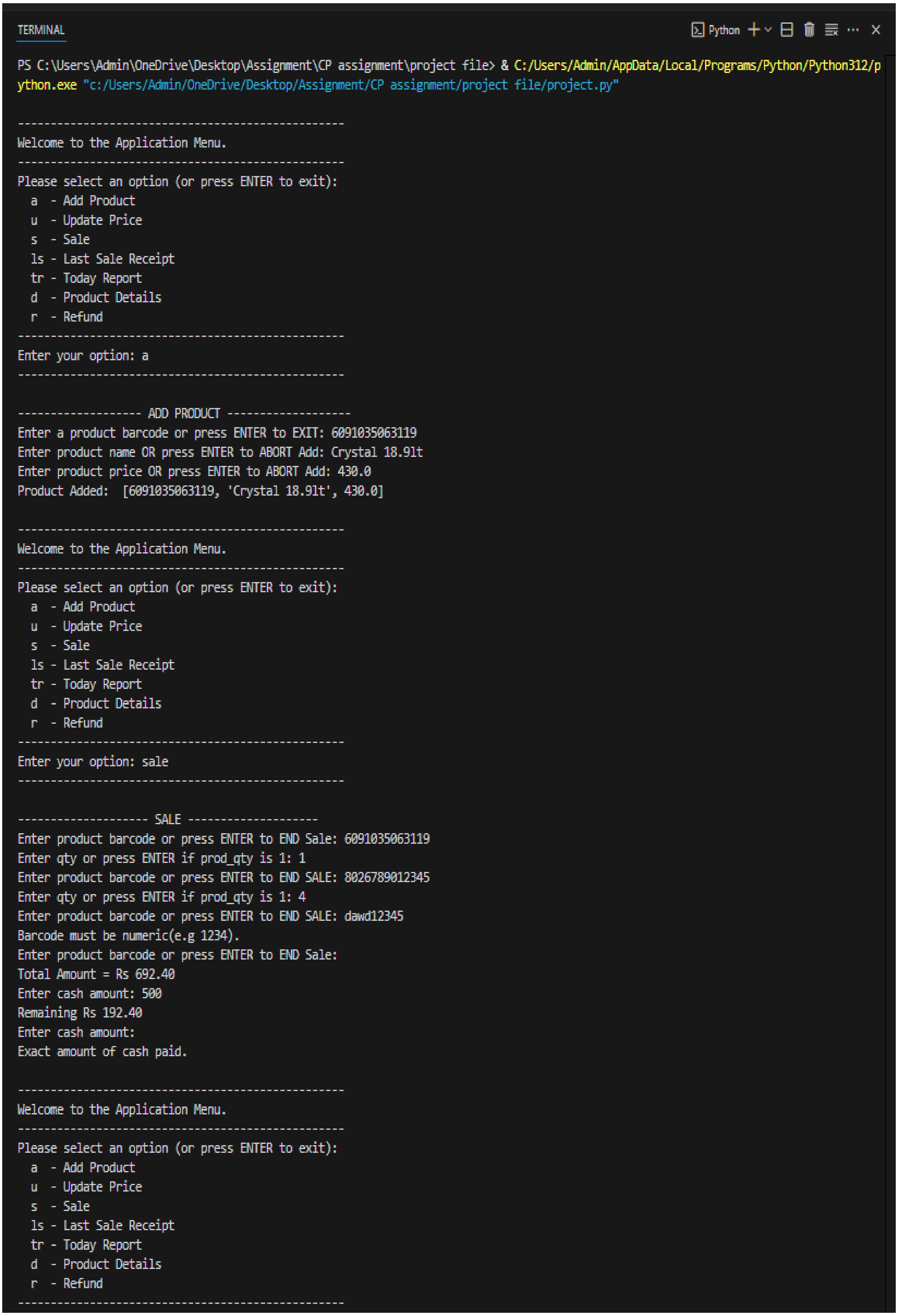
- Improve the user interface from CLI to a proper UI that have proper functions
- Improve the functionality of the system to be able to save daily reports, allow for modifying of product quantity or removing a product during sale operation.
- Improve files from .txt to other type such as .json.
- Improve the system to connect to a local computer or server that will contain all files that can be access by all computers. Hence allowing for all users to have access to same files.
- Improve the filter date option to allow for retrieval of other days sale reports.
- Add a function that will calculate/sum quantity of all product sold and generate a report of most selling.
- Add a function that will also show the net profit in a day.

References

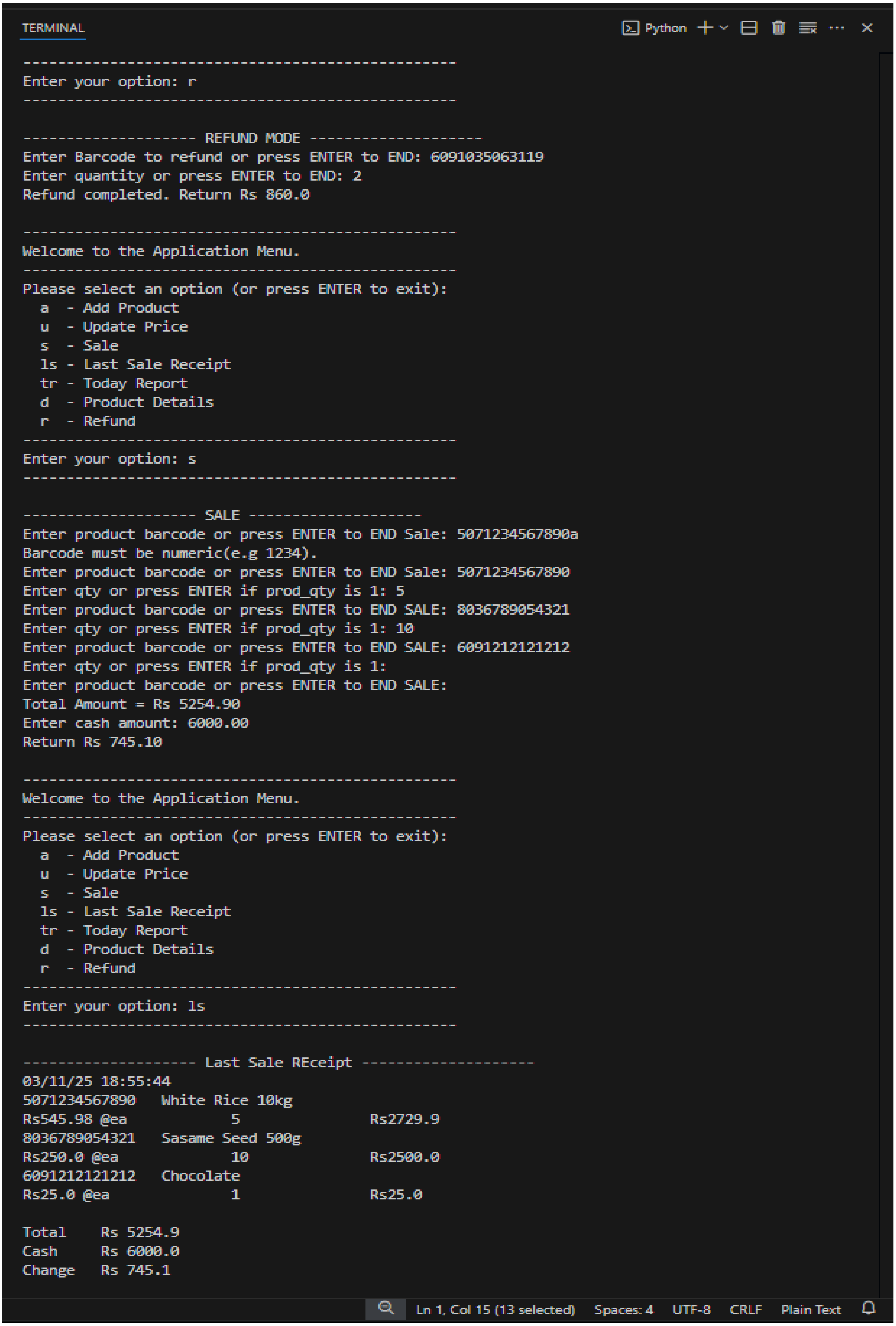
W3Schools.com (no date). https://www.w3schools.com/python/python_datetime.asp (Accessed: March 11, 2025).

GeeksforGeeks (2025) *File handling in Python*. https://www.geeksforgeeks.org/file-handling-python/ (Accessed: March 11, 2025).

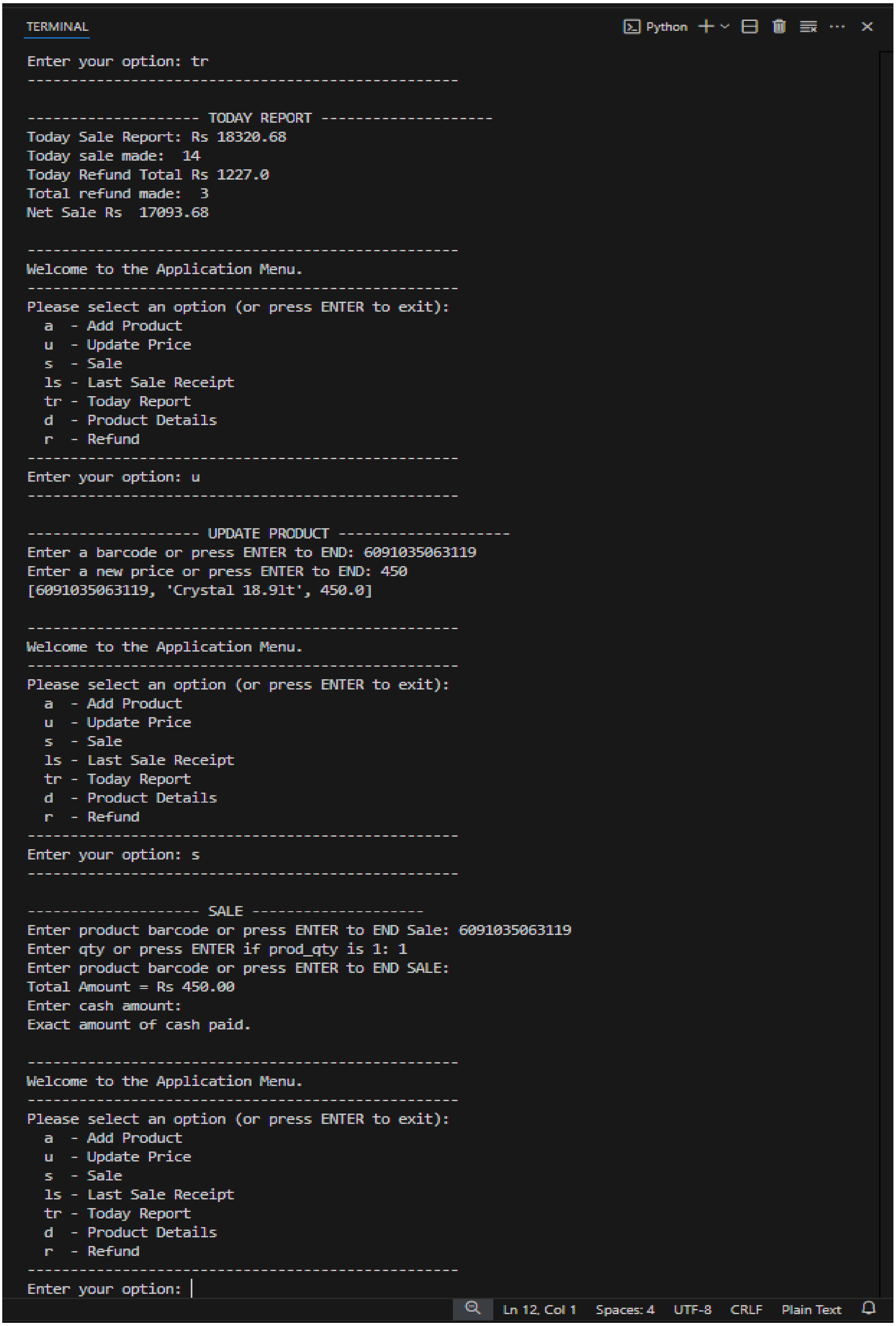
Deutch, K. (2024) *What is a POS system and how does it work?* https://squareup.com/us/en/the-bottom-line/operating-your-business/what-pos-system (Accessed: March 11, 2025).



```
PS C:\Users\Admin\OneDrive\Desktop\Assignment\CP assignment\project file> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/p
ython.exe "c:/Users/Admin/OneDrive/Desktop/Assignment/CP assignment/project file/project.py"

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: a
--------------------------------------------------


------------------ ADD PRODUCT ------------------
Enter a product barcode or press ENTER to EXIT: 6091035063119
Enter product name OR press ENTER to ABORT Add: Crystal 18.91t
Enter product price OR press ENTER to ABORT Add: 430.0
Product Added:  [6091035063119, 'Crystal 18.91t', 430.0]

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: sale
--------------------------------------------------


-------------------- SALE --------------------
Enter product barcode or press ENTER to END Sale: 6091035063119
Enter qty or press ENTER if prod_qty is 1: 1
Enter product barcode or press ENTER to END SALE: 8026789012345
Enter qty or press ENTER if prod_qty is 1: 4
Enter product barcode or press ENTER to END SALE: dawd12345
Barcode must be numeric(e.g 1234).
Enter product barcode or press ENTER to END Sale:
Total Amount = Rs 692.40
Enter cash amount: 500
Remaining Rs 192.40
Enter cash amount:
Exact amount of cash paid.

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
```


TERMINAL
Python
Ln 1, Col 15 (13 selected)   Spaces: 4   UTF-8   CRLF   Plain Text

```
--------------------------------------------------
Enter your option: r
--------------------------------------------------

-------------------- REFUND MODE --------------------
Enter Barcode to refund or press ENTER to END: 6091035063119
Enter quantity or press ENTER to END: 2
Refund completed. Return Rs 860.0

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: s
--------------------------------------------------

-------------------- SALE --------------------
Enter product barcode or press ENTER to END Sale: 5071234567890a
Barcode must be numeric(e.g 1234).
Enter product barcode or press ENTER to END Sale: 5071234567890
Enter qty or press ENTER if prod_qty is 1: 5
Enter product barcode or press ENTER to END SALE: 8036789054321
Enter qty or press ENTER if prod_qty is 1: 10
Enter product barcode or press ENTER to END SALE: 6091212121212
Enter qty or press ENTER if prod_qty is 1:
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 5254.90
Enter cash amount: 6000.00
Return Rs 745.10

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: ls
--------------------------------------------------

-------------------- Last Sale REceipt --------------------
03/11/25 18:55:44
5071234567890   White Rice 10kg
Rs545.98 @ea            5              Rs2729.9
8036789054321   Sasame Seed 500g
Rs250.0 @ea             10             Rs2500.0
6091212121212   Chocolate
Rs25.0 @ea              1              Rs25.0

Total     Rs 5254.9
Cash      Rs 6000.0
Change    Rs 745.1
```



```
Enter your option: tr
--------------------------------------------------


-------------------- TODAY REPORT --------------------
Today Sale Report: Rs 18320.68
Today sale made:  14
Today Refund Total Rs 1227.0
Total refund made:  3
Net Sale Rs  17093.68

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: u
--------------------------------------------------

-------------------- UPDATE PRODUCT --------------------
Enter a barcode or press ENTER to END: 6091035063119
Enter a new price or press ENTER to END: 450
[6091035063119, 'Crystal 18.9lt', 450.0]

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option: s
--------------------------------------------------

-------------------- SALE --------------------
Enter product barcode or press ENTER to END Sale: 6091035063119
Enter qty or press ENTER if prod_qty is 1: 1
Enter product barcode or press ENTER to END SALE:
Total Amount = Rs 450.00
Enter cash amount:
Exact amount of cash paid.

--------------------------------------------------
Welcome to the Application Menu.
--------------------------------------------------
Please select an option (or press ENTER to exit):
  a  - Add Product
  u  - Update Price
  s  - Sale
  ls - Last Sale Receipt
  tr - Today Report
  d  - Product Details
  r  - Refund
--------------------------------------------------
Enter your option:
```

```python
import datetime as dt

x = dt.datetime.now()
y_today = x.strftime("%x")
y_date_time = x.strftime("%x"),x.strftime("%X")

class Cashier_system:
    def Options(self):
        print("\n"+"-" * 50)
        print("Welcome to the Application Menu.")
        print("-" * 50)
        print("Please select an option (or press ENTER to exit):")
        print("  a  - Add Product")
        print("  u  - Update Price")
        print("  s  - Sale")
        print("  ls - Last Sale Receipt")
        print("  tr - Today Report")
        print("  d  - Product Details")
        print("  r  - Refund")
        print("-" * 50)

        user_option = input("Enter your option: ").lower()

        if user_option == "a" or user_option == "add product":
            print("-" * 50)
            print("\n"+"-"*19, "ADD PRODUCT", "-"*19)
            self.Add_product()
        elif user_option == "u" or user_option == "update price":
            print("-" * 50)
            print("\n"+"-"*20, "UPDATE PRODUCT", "-"*20)
            self.Update_price()
        elif user_option == "s" or user_option == "sale":
            print("-" * 50)
            print("\n"+"-"*20, "SALE", "-"*20)
            self.Sale()
        elif user_option == "ls" or user_option == "last sale receipt":
            print("-" * 50)
            print("\n"+"-"*20, "Last Sale REceipt", "-"*20)
            self.Last_sale()
        elif user_option == "tr" or user_option == "today report":
            print("-" * 50)
            print("\n"+"-"*20, "TODAY REPORT", "-"*20)
            self.Today_report()
        elif user_option == "d" or user_option == "product details":
            print("-" * 50)
            print("\n"+"-"*20, "PRODUCTS DETAILS", "-"*20)
            self.Products_details()
        elif user_option == "r" or user_option == "refund":
            print("-" * 50)
            print("\n"+"-"*20, "REFUND MODE", "-"*20)
            self.Refund()
        elif user_option != "":
            print("INVALID option entered.\nSelect a valid option: a, u, s, ls, tr, d, r.")
            print("-" * 70)
            return self.Options()
        else:
            print("Application ENDED and CLOSE.")
            exit()


    def Add_product(self):
        try:
            file_product = open("product_file.txt", "a+")
            file_product.seek(0)
            products = file_product.readlines()

            user_add_barcode = (input("Enter a product barcode or press ENTER to EXIT: "))
            if user_add_barcode == "":
                print("\n"+"ADD operation ENDED.")
                file_product.close()
                return Cashier_system.Options(self)
            elif user_add_barcode.isnumeric() == False:
                print("\n"+"Barcode should be only numeric. ADD operation ABORTED.")
                file_product.close()
                return Cashier_system.Options(self)

            for line in products:
                try:
                    product_list = eval(line.strip())
                    if int(user_add_barcode) == product_list[0]:
                        print("Barcode ALREADY EXISTS. Product name: " + product_list[1])
                        file_product.close()
                        return Cashier_system.Options(self)
                except (ValueError, SyntaxError):
                    print("ERROR IN READING PRODUCT DETAILS IS FILE. PRODUCT DATA IN FILE IS CORRUPT.")
                    file_product.close()
                    return Cashier_system.Options(self)

            user_add_name = input("Enter product name OR press ENTER to ABORT Add: ")
            if user_add_name == "":
                print("ADD operation ABORTED.")
                file_product.close()
                return Cashier_system.Options(self)

            while True:
                user_add_price = input("Enter product price OR press ENTER to ABORT Add: ")
                if user_add_price == "":
                    print("ADD operation ABORTED.")
                    file_product.close()
                    return Cashier_system.Options(self)
                try:
                    if float(user_add_price) < 0:
                        print("Price cannot be negative(-ve).")
                        continue
                    break
                except ValueError:
```

```
                print("Price should be numeric(1234).")

            details = [int(user_add_barcode), user_add_name, float(user_add_price)]
            str_details = str(details)+"\n"
            file_product.write(str_details)
            print("Product Added: ", details)

            file_product.close()
            return Cashier_system.Options(self)

        except(IOError):
            print("Missing File(s).")
            return Cashier_system.Options(self)


    def Sale(self):
        try:
            file_product = open("product_file.txt","r")
            file_product.seek(0)
            file_save = open("sale_file.txt", "a")
            products = file_product.readlines()

            receipt_time = [[x.strftime("%x"),x.strftime("%X")]]

            user_sale_barcode = input("Enter product barcode or press ENTER to END Sale: ")
            if user_sale_barcode == "":
                file_save.close()
                file_product.close()
                return Cashier_system.Options(self)

            sale_end = False
            Total = 0

            while sale_end == False:
                product_exist = False
                if user_sale_barcode.isnumeric() == False:
                    print("Barcode must be numeric(e.g 1234).")
                    user_sale_barcode = input("Enter product barcode or press ENTER to END Sale: ")
                    if user_sale_barcode == "":
                        sale_end = True
                    continue

                for lines in products:
                    try:
                        ind_prod = eval(lines)
                        if int(user_sale_barcode) == ind_prod[0]:
                            product_exist = True

                            user_sale_qty = input("Enter qty or press ENTER if prod_qty is 1: ")
                            if user_sale_qty == "":
                                user_sale_qty = 1
                            elif user_sale_qty.isnumeric() == True:
                                user_sale_qty = int(user_sale_qty)
                                if user_sale_qty < 1:
                                    print("Quantity must be greater 0.")
                                    continue
                            else:
                                print("Quantity must be +ve and numeric.")
                                continue

                            sum = 0
                            sum = ind_prod[2] * user_sale_qty
                            Total += sum
                            cart_list = [user_sale_barcode, ind_prod[1], ind_prod[2], user_sale_qty, float(sum)]
                            receipt_time.append(cart_list)
                            user_sale_barcode = input("Enter product barcode or press ENTER to END SALE: ")

                            if user_sale_barcode == "":
                                sale_end = True
                            break
                    except(ValueError, SyntaxError, IndexError):
                        print("Error in processing input barcode: ", user_sale_barcode)
                        break
                if product_exist == False:
                    print("Barcode not found.",end=" ")
                    user_sale_barcode = input("Enter product barcode or press ENTER to END Sale: ")
                    if user_sale_barcode == "":
                        sale_end = True

            print("Total Amount = Rs %0.2f" % Total)

            remainig = Total
            Total_cash = 0
            Cash = 0
            change = 0.00

            if sale_end == True:
                while remainig > 0:
                    Cash = input("Enter cash amount: ")
                    try:
                        if Cash == "":
                            Cash = remainig
                            Total_cash += Cash
                            change = 0.00
                            print("Exact amount of cash paid.")
                            break

                        else:
                            Cash = float(Cash)
                            if Cash < 0:
                                print("Cash cannot be -ve.")
                                continue
                            Total_cash += Cash
                            if Cash < remainig:
                                remainig -= Cash
                                print("Remaining Rs %0.2f" % remainig)
```

```
                        else:
                            change = Cash - remainig
                            remainig = 0
                            print("Return Rs %0.2f" % change)
                            break
                except(ValueError):
                    print("Invalid cash amount entered.")
                    continue

        cash_mgmt = [float("%0.2f" % Total), float("%0.2f" % Total_cash), float("%0.2f" % change)]
        receipt_time.append(cash_mgmt)

        receipt_str = str(receipt_time) + "\n"
        file_save.write(receipt_str)

        file_save.close()
        file_product.close()
        return Cashier_system.Options(self)

    except(IOError):
        print("Missing Read File, File NOT found.")
        return Cashier_system.Options(self)



def Update_price(self):
    try:
        file_product = open("product_file.txt", "r+")
        product = file_product.readlines()

        user_update_barcode = input("Enter a barcode or press ENTER to END: ")
        if user_update_barcode == "":
            print("Update operation ENDED.")
            file_product.close()
            return Cashier_system.Options(self)

        if user_update_barcode.isnumeric() == False:
            print("Barcode should be numeric. UPDATE operation ABORTED.")
            file_product.close()
            return Cashier_system.Options(self)

        exists = False
        updated_product = []
        for line in product:
            prod_list = eval(line)
            if int(user_update_barcode) == prod_list[0]:
                exists = True
                user_update_price = input("Enter a new price or press ENTER to END: ")
                if user_update_price == "":
                    file_product.close()
                    return Cashier_system.Options(self)
                if float(user_update_price) < 0:
                    print("Price cannot be negative. UPDATE ENDED. No price change.")
```

```python
                        file_product.close()
                        return Cashier_system.Options(self)
                    prod_list[2] = float(user_update_price)
                    updated_product.append(str(prod_list)+"\n")
                    print(prod_list)
                else:
                    updated_product.append(line)

            if exists == False:
                print("Barcode not found.")

            file_product.seek(0)
            file_product.writelines(updated_product)
            file_product.truncate()

            file_product.close()
            return Cashier_system.Options(self)

        except(FileNotFoundError):
            print("Missing Read File, File NOT found.")
            return Cashier_system.Options(self)



    def Last_sale(self):
        try:
            file_save = open("sale_file.txt","r")
            lines = file_save.readlines()

            try:
                last_line = eval(lines[-1].strip())
            except(ValueError, SyntaxError, IndexError):
                print("Error in reading sale details from file.")
                file_save.close()
                return Cashier_system.Options(self)

            print(last_line[0][0], last_line[0][1])

            for items in last_line[1:-1]:
                item_detail = str(items[0]) + "\t" + str(items[1]) + "\n" + "Rs"+ str(items[2]) + " @ea\t\t" +
str(items[3]) + "\t\t" + "Rs"+str(items[4])
                print(item_detail)

            print()
            print("Total \t Rs", last_line[-1][0])
            print("Cash \t Rs", last_line[-1][1])
            print("Change \t Rs",last_line[-1][2])

            file_save.close()
            return Cashier_system.Options(self)

        except(IOError):
            print("Missing Read File, File NOT found.")
        return Cashier_system.Options(self)



def Today_report(self):
    try:
        sale_file = open("sale_file.txt", "r")
        refund_file = open("refund_file.txt", "r")
        sales_lines = sale_file.readlines()
        refund_lines = refund_file.readlines()

        Total_cash = 0
        sale_count = 0
        for sale in sales_lines:
            try:
                if sale.strip():
                    sale = eval(sale.strip())
            except(ValueError, SyntaxError, IndexError):
                print("An error has occur in sale file, a data line is corrupt.")
                sale_file.close()
                refund_file.close()
                return Cashier_system.Options(self)
            if sale[0][0] == y_today:
                if sale[-1][0] != 0:
                    Total_cash += sale[-1][0]
                    sale_count += 1

        refund_cash = 0
        refund_count = 0
        for refund in refund_lines:
            try:
                if refund.strip():
                    refund = eval(refund.strip())
            except(ValueError, SyntaxError, IndexError):
                print("An error has occur in sale file, a data line is corrupt.")
                sale_file.close()
                refund_file.close()
                return Cashier_system.Options(self)
            if refund[0] == y_today:
                refund_cash += refund[5]
                refund_count += 1

        net_total = Total_cash - refund_cash

        print("Today Sale Report: Rs",Total_cash)
        print("Today sale made: ", sale_count)
        print("Today Refund Total Rs", refund_cash)
        print("Total refund made: ", refund_count)
        print("Net Sale Rs ", net_total)

        sale_file.close()
        refund_file.close()
        return Cashier_system.Options(self)
    except(IOError):
        print("Missing Read File, File NOT found.")
        return Cashier_system.Options(self)



def Products_details(self):
    try:
        file_product = open("product_file.txt","r")
        product = file_product.readlines()

        for lines in product:
            try:
                prod_details = eval(lines.strip())
            except(ValueError, SyntaxError, IndexError):
                    print("Error in reading sale details from file. Line(s) in file is corrupt.")
                    file_product.close()
                    return Cashier_system.Options(self)
            barcode, name, price = prod_details
            print("{:<15}{:<30}Rs {:>7.2f}".format(barcode, name, price))
        file_product.close()
        return Cashier_system.Options(self)

    except IOError:
        print("Missing Read File, File NOT found.")
        return Cashier_system.Options(self)



def Refund(self):
    try:
        file_save = open("product_file.txt","r")
        file_save.seek(0)
        file_refund = open("refund_file.txt","a")
        file_product = file_save.readlines()

        user_refund_barcode = input("Enter Barcode to refund or press ENTER to END: ")
        if user_refund_barcode == "":
            file_save.close()
            file_refund.close()
            print("REFUND operation ENDED.")
        elif user_refund_barcode.isnumeric() == False:
            file_save.close()
            file_refund.close()
            print("Barcode should be numeric.")
        else:
            product_found = False
            for product in file_product:
                try:
                    product = eval(product.strip())
                except(ValueError, SyntaxError, IndexError):
                    print("Error in reading sale details from file. Line(s) in file is corrupt.")
```

```
                        file_save.close()
                        file_refund.close()
                        return Cashier_system.Options(self)

                    if int(user_refund_barcode) == product[0]:
                        product_found = True

                        user_refund_qty = input("Enter quantity or press ENTER to END: ")
                        if user_refund_qty == "":
                            print("REFUND operations CANCELLED.")
                            break
                        elif user_refund_qty.isnumeric() == True:
                            user_refund_qty = int(user_refund_qty)
                            if user_refund_qty < 1:
                                print("Quantity must be greater 0.")
                                continue
                        else:
                            print("Quantity must be +ve and numeric.")
                            continue

                        refund_sum = 0
                        refund_date = x.strftime("%x")
                        refund_sum = product[2] * user_refund_qty

                        refund_list = [refund_date, product[0], product[1], product[2], user_refund_qty,
float(refund_sum)]

                        file_refund.write(str(refund_list) + "\n")
                        print("Refund completed. Return Rs", refund_sum)
                        break

                if product_found == False:
                    print("Barcode does not exists in record ", user_refund_barcode)

            file_save.close()
            file_refund.close()
            return Cashier_system.Options(self)

        except(IOError):
            print("An Error has occur.")
            return Cashier_system.Options(self)


if __name__ == "__main__":
    system = Cashier_system()
    system.Options()
```

product_file.txt

```
[6091212121212, 'Chocolate', 25.0]
[8036789054321, 'Sasame Seed 500g', 250.0]
[6151234567890, 'Apples 1kg', 60.0]
[6091231231231, 'Milk 2kg', 255.5]
[5071234567890, 'White Rice 10kg', 545.98]
[6211123456789, 'Canned Tuna 180g', 78.6]
[8026789012345, 'Deodorant Stick 50g', 65.6]
[8034567890123, 'Matches Box*10', 15.0]
[8036789022334, 'Leader Sunflower Oil 1lt', 78.0]
[8037789012345, 'Pet Shampoo 500ml', 235.8]
```

Sale_file.txt

```
[['03/11/25', '15:57:40'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0], [20.0,
25.0, 5.0]]
[['03/11/25', '16:03:27'], ['6151234567890', 'Apples 1kg', 60.0, 1, 60.0], [60.0,
60.0, 0.0]]
[['03/11/25', '16:07:38'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0],
['6211123456789', 'Canned Tuna 180g', 78.6, 2, 157.2], [177.2, 200.0, 22.8]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8],
[235.8, 235.8, 0.0]]
[['03/11/25', '16:07:38'], ['6091212121212', 'Chocolate', 20.0, 5, 100.0],
['6151234567890', 'Apples 1kg', 60.0, 10, 600.0], ['6091231231231', 'Milk 2kg',
255.5, 2, 511.0], ['8036789022334', 'Leader Sunflower Oil 1lt', 78.0, 24, 1872.0],
['8034567890123', 'Matches Box*10', 15.0, 1, 15.0], ['8036789054321', 'Sasame Seed
500g', 250.0, 5, 1250.0], [4348.0, 5000.0, 652.0]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8],
['8036789022334', 'Leader Sunflower Oil 1lt', 78.0, 10, 780.0], ['5071234567890',
'White Rice 10kg', 545.98, 10, 5459.8], [6475.6, 10000.0, 3524.4]]
[['03/11/25', '16:07:38'], ['6151234567890', 'Apples 1kg', 60.0, 2, 120.0],
['5071234567890', 'White Rice 10kg', 545.98, 1, 545.98], [665.98, 1000.0, 334.02]]
[['03/11/25', '16:07:38'], ['8037789012345', 'Pet Shampoo 500ml', 235.8, 1, 235.8],
[235.8, 235.8, 0.0]]
[['03/11/25', '15:57:40'], ['6091212121212', 'Chocolate', 20.0, 1, 20.0], [20.0,
25.0, 5.0]]
[['03/11/25', '16:03:27'], ['6151234567890', 'Apples 1kg', 60.0, 1, 60.0], [60.0,
60.0, 0.0]]
[['03/11/25', '16:48:20'], ['6091212121212', 'Chocolate', 25.0, 2, 50.0], [50.0,
100.0, 50.0]]
[['03/11/25', '16:48:20'], ['6091212121212', 'Chocolate', 25.0, 1, 25.0], [25.0,
25.0, 0.0]]
```

refund_file.txt

```
['03/11/25', 8037789012345, 'Pet Shampoo 500ml', 235.8, 1, 235.8]
['03/11/25', 8026789012345, 'Deodorant Stick 50g', 65.6, 2, 131.2]
```