Pharmacy
Inventory & Billing System

CT/2020/018

E.D. Thaveesh Vihanga Adithya

Structured Programming II

GitHub Repository

https://github.com/thaveeshvihanga/Pharmacy_Inventory_and_Billing_System1.git

August 2023

Faculty of Computing and Technology

University of Kaleniya.

Pharmacy Inventory and Billing System

Introduction

I built Simple Inventory and Billing System for pharmacy. Specifically, For Suwajaya Pharmacy, Kotte road, Nugegoda. This simple system can perform various tasks, such as adding new products to the inventory, tracking stock levels, generating purchase orders, managing suppliers, processing product returns, handling sales transactions, and generating sales reports. Each task has its own function separately to complete the task. There are 8 main tasks, So I first wrote Separate Functions for Tasks then I created User Interface after that. I used simple data Structures in C programming Language such as pointers, structures and arrays also I used for loop, while loop, if statements.

Features

- Product Management: The system allows the addition of new products to the inventory along with essential details like name, generic name, batch number, expiry date, quantity, and price.
- View Inventory : Pharmacy staff can monitor all the stocks that are available.
- Inventory Tracking: Pharmacy staff can monitor the stock levels of products and receive alerts for items with low quantities to ensure timely restocking.
- Supplier Management: Pharmacy administrators can add new suppliers and maintain their contact details and terms of business within the system.
- Product Search: The software enables easy retrieval of product information based on either product name or batch number, facilitating quick and accurate search results.
- Sales Transactions: The system supports seamless sales processing, allowing staff to add products to an invoice, select payment methods (cash or credit card), and generate receipts for customers.
- Product Returns and Refunds: Pharmacy staff can process product returns, update the inventory with the returned quantities, and handle refunds appropriately.
- Sales Reports: The system generates comprehensive sales reports, providing insights into product sales, quantities sold, revenue generated, and overall performance.
- **For each of above key Features there is a function.**

Product Management

For this I use function called "addProduct" that allows users to add new products to the inventory. It takes two parameters: an array of struct Product called inventory and a pointer to an integer productCount representing the current number of products in the inventory. The function enables the user to input details of a new product, such as its name, generic name, batch number, expiry date, quantity, and price. It then stores this information in the inventory array.

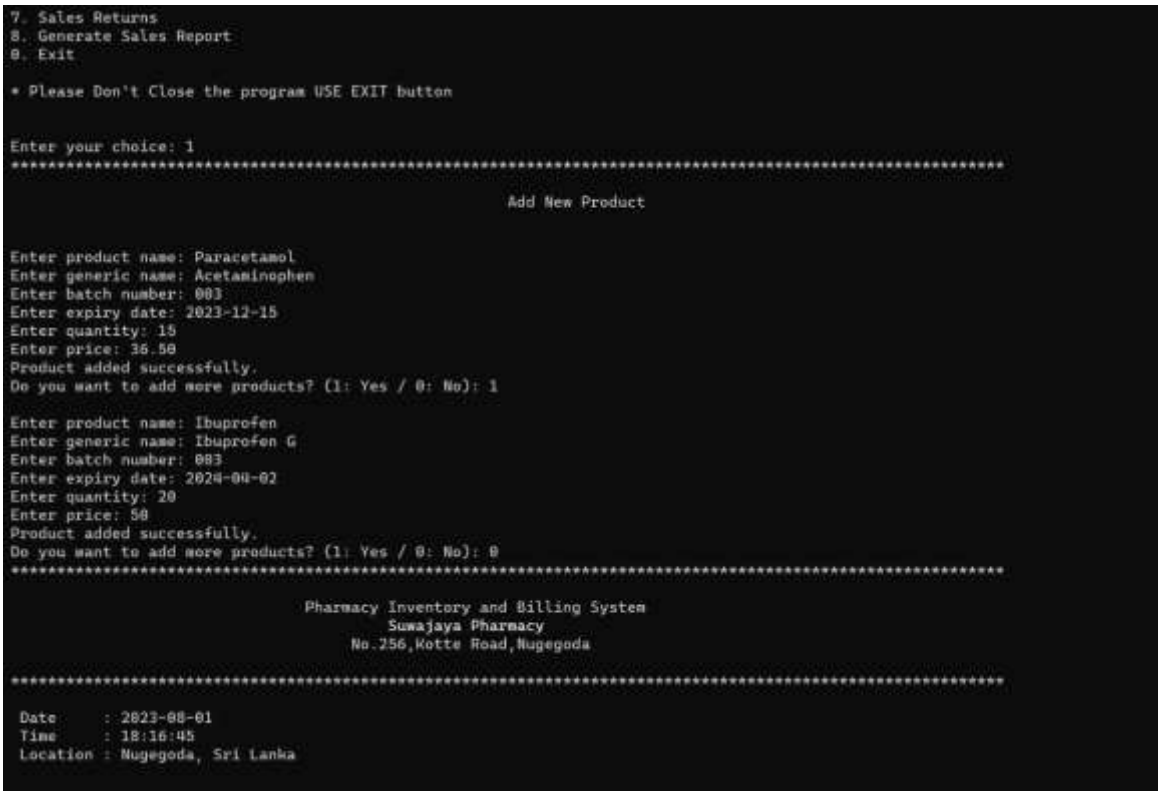
Code :

```
// Function to add a new product to the inventory
void addProduct(struct Product inventory[], int *productCount) {
    if (*productCount == MAX_PRODUCTS) {
        printf("Inventory is full. Cannot add more products.\n");
        return;
    }

    printf("*****************************************************************************************************\n");
    printf("                                        Add New Product                                              \n");

    int choice;
    do {
        // Prompt the user to enter product details
        printf("\nEnter product name: ");
        scanf(" %[^\n]", inventory[*productCount].name);

        printf("Enter generic name: ");
        scanf(" %[^\n]", inventory[*productCount].generic_name);

        printf("Enter batch number: ");
        scanf("%d", &inventory[*productCount].batch_number);

        printf("Enter expiry date: ");
        scanf(" %[^\n]", inventory[*productCount].expiry_date);

        printf("Enter quantity: ");
        scanf("%d", &inventory[*productCount].quantity);

        printf("Enter price: ");
        scanf("%f", &inventory[*productCount].price);

        (*productCount)++;


        printf("Product added successfully.\n");

        // Ask the user if they want to add more products
        printf("Do you want to add more products? (1: Yes / 0: No): ");
        scanf("%d", &choice);
    } while (choice == 1);
```

- It takes two parameters: inventory, an array of struct Product, and productCount, a pointer to an integer.It takes two parameters, inventory, an array of struct Product

- If the inventory is full, it prints a message to inform the user and immediately returns from the function using the return statement, preventing further product additions and The function starts by displaying a header to indicate that the user is adding a new product.
- This is a loop that allows the user to add multiple products one after the other until they choose to stop. Within the loop, the user is prompted to enter details of a new product, including its name, generic name, batch number, expiry date, quantity, and price. The input values are stored in the corresponding fields of the inventory array at the index *productCount, which is incremented after each product addition.
- The user is asked whether they want to add more products. If the user enters 1 (Yes), the loop continues, allowing the user to add another product. If the user enters 0 (No), the loop ends, and the function returns to the main program.

View :


7. Sales Returns
8. Generate Sales Report
0. Exit
* Please Don't Close the program USE EXIT button
Enter your choice: 1
Add New Product
Enter product name: Paracetamol
Enter generic name: Acetaminophen
Enter batch number: 003
Enter expiry date: 2023-12-15
Enter quantity: 15
Enter price: 36.50
Product added successfully.
Do you want to add more products? (1: Yes / 0: No): 1
Enter product name: Ibuprofen
Enter generic name: Ibuprofen G
Enter batch number: 003
Enter expiry date: 2024-04-02
Enter quantity: 20
Enter price: 50
Product added successfully.
Do you want to add more products? (1: Yes / 0: No): 0
Pharmacy Inventory and Billing System
Suwajaya Pharmacy
No.256,Kotte Road,Nugegoda
Date : 2023-08-01
Time : 18:16:45
Location : Nugegoda, Sri Lanka

View Inventory

I used a function called viewInventory, which is responsible for displaying the current inventory of products at "Suwajaya Pharmacy." The function takes three parameters: an array of struct Product called inventory, an integer productCount representing the number of products in the inventory.

Code :

```
121
122  // Function to view the remaining inventory
123
124 void viewInventory(struct Product inventory[], int productCount, struct Supplier suppliers[]) {
125
126     printf("*****************************************************************************************************************\n\n");
127     printf("                                        Suwajaya Pharmacy - Inventory                                          \n\n");
128     printf("*****************************************************************************************************************\n\n");
129     printf("Product Name       | Batch Number | Quantity | Total Value (Rs.)\n");
130     printf("\n-----------------------------------------------------------------------------------------------------------------\n");
131
132     float totalValue = 0.0;
133
134     for (int i = 0; i < productCount; i++) {
135         int batchNumber = inventory[i].batch_number;
136         char* productName = inventory[i].name;
137         int quantity = inventory[i].quantity;
138         float itemValue = quantity * inventory[i].price;
139
140         printf("%-20s | %-12d | %-8d | %-16.2f\n", productName, batchNumber, quantity, itemValue);
141         totalValue += itemValue;
142     }
143
144     printf("\n-----------------------------------------------------------------------------------------------------------------\n");
145     printf("Total Value (All Products)                          | %-16.2f\n", totalValue);
146     printf("\n*****************************************************************************************************************\n\n");
147 }
148
149
```

- It takes three parameters: inventory, an array of struct Product, productCount, an integer representing the number of products, and suppliers, an array of struct Supplier.
- the table header that describes the columns in the inventory table. It includes the product name, batch number, quantity, and total value.
- A variable totalValue is initialized to 0.0 to store the cumulative total value of all
- For each product, the code extracts the product's batch number, name, quantity, and price from the inventory array.
- It then calculates the total value of that particular product by multiplying its quantity with its price.
- The totalValue is updated by adding the itemValue, which is the total value of the current product.
- After displaying all product details, code prints the total value of all products in the inventory.

View :

```
2. View Inventory
3. Track Stock
4. Add Supplier
5. Search Product
6. Sales Invoice
7. Sales Returns
8. Generate Sales Report
0. Exit

* Please Don't Close the program USE EXIT button


Enter your choice: 2
*****************************************************************************************************************

                                        Suwajaya Pharmacy - Inventory

*****************************************************************************************************************

Product Name       | Batch Number | Quantity | Total Value (Rs.)
-----------------------------------------------------------------------------------------------------------------
0                  | 1            | 5        | 750.00
a                  | 2            | 12       | 1440.00
Paracetamol        | 3            | 15       | 547.50
Ibuprofen          | 3            | 20       | 1000.00
-----------------------------------------------------------------------------------------------------------------
Total Value (All Products)                          | 3737.50

*****************************************************************************************************************

*****************************************************************************************************************
```

Inventory Tracking

I created a function named trackStock, which is responsible for tracking the stock levels of products in the inventory at "Suwajaya Pharmacy." The function takes two parameters: an array of struct Product called inventory and an integer productCount representing the number of products in the inventory.

Code :

```
106
107  // Function to track stock levels and generate alerts
108  void trackStock(struct Product inventory[], int productCount) {
109      printf("*****************************************************************************************************************\n\n");
110      printf("\n                                        Stock Alert                                                          \n\n");
111      printf("\nProducts with low stock levels:\n");
112
113      for (int i = 0; i < productCount; i++) {
114          if (inventory[i].quantity < 10) {
115              printf("Product: %s (Batch Number: %d)\n", inventory[i].name, inventory[i].batch_number);
116          }
117      }
118
119      printf("\n*****************************************************************************************************************\n");
120  }
```

- It takes two parameters: inventory, an array of struct Product, and productCount, an integer representing the number of products in the inventory.
- For each product, the code checks if its current quantity (stored in the quantity field of the struct Product) is less than 10, indicating low stock.
- If the condition is true, it means the product is low in stock, and the product name along with its batch number is printed .

View:

```
*****************************************************************************************************************

*****************************************************************************************************************

                              Pharmacy Inventory and Billing System
                                      Suwajaya Pharmacy
                                  No.256,Kotte Road,Nugegoda

*****************************************************************************************************************

 Date     : 2023-08-01
 Time     : 20:00:04
 Location : Nugegoda, Sri Lanka


1. Add Product
2. View Inventory
3. Track Stock
4. Add Supplier
5. Search Product
6. Sales Invoice
7. Sales Returns
8. Generate Sales Report
0. Exit

* Please Don't Close the program USE EXIT button

Enter your choice: 3
*****************************************************************************************************************

                                          Stock Alert


Products with low stock levels:

Product: g (Batch Number: 1)

*****************************************************************************************************************
*****************************************************************************************************************
```

Supplier Management

I created addSupplier function. The addSupplier function allows users to add a new supplier to the supplier database. It takes two parameters:
suppliers: This is an array of struct Supplier objects, representing the existing list of suppliers in the database.
supplierCount: This is a pointer to an integer variable that represents the current number of suppliers in the suppliers array. It keeps track of how many suppliers are already present in the database.

Code :

```
150
151  // Function to add a new supplier to the supplier database
152 void addSupplier(struct Supplier suppliers[], int *supplierCount) {
153     if (*supplierCount >= MAX_SUPPLIERS) {
154
155         printf("\n**********************************************************************************************************************\n\n");
156         printf("\n                              Supplier database is full. Cannot add more suppliers.                                  \n");
157             return;
158     }
159
160     // Prompt the user to enter supplier details
161     printf("\n**********************************************************************************************************************\n\n");
162     printf("                                              Enter Supplier Detailes                                                   \n\n");
163     printf("\nEnter supplier name: ");
164     scanf(" %[^\n]", suppliers[*supplierCount].name);
165
166     printf("Enter contact details: ");
167     scanf(" %[^\n]", suppliers[*supplierCount].contact_details);
168
169     printf("Enter terms: ");
170     scanf(" %[^\n]", suppliers[*supplierCount].terms);
171
172
173     // Increment the supplier count
174     (*supplierCount)++;
175
176     printf("                                          Supplier added successfully.                                                  \n");
177 }
178
```

- The function first checks if the maximum capacity of the supplier database has been reached by comparing the value of *supplierCount with the maximum number of suppliers allowed, which is defined as MAX_SUPPLIERS. If the database is already full, the function prints a message indicating that no more suppliers can be added, and it immediately returns without making any changes to the database.
- If the supplier database is not full, the function proceeds to prompt the user to enter details for the new supplier. It displays a series of prompts asking for the supplier's name, contact details, and terms. The user's input is then read using scanf and stored in the appropriate fields of the next available struct Supplier object in the suppliers array (at the index *supplierCount).
- After successfully adding the new supplier, the function increments *supplierCount by one, indicating that the number of suppliers in the database has increased.
- Finally, the function prints a message to inform the user that the new supplier has been added successfully to the database.
- the "terms" could include payment terms (e.g., payment due date, payment methods accepted), delivery terms (e.g., delivery schedule, shipping costs), or any other terms that govern the business relationship between the pharmacy and the supplier

View :

```
1. Add Product
2. View Inventory
3. Track Stock
4. Add Supplier
5. Search Product
6. Sales Invoice
7. Sales Returns
8. Generate Sales Report
0. Exit

* Please Don't Close the program USE EXIT button

Enter your choice: 4

**********************************************************************************************************************

                                        Enter Supplier Detailes

Enter supplier name: mohan
Enter contact details: 0112568823
Enter terms: -
                                      Supplier added successfully.
**********************************************************************************************************************
```

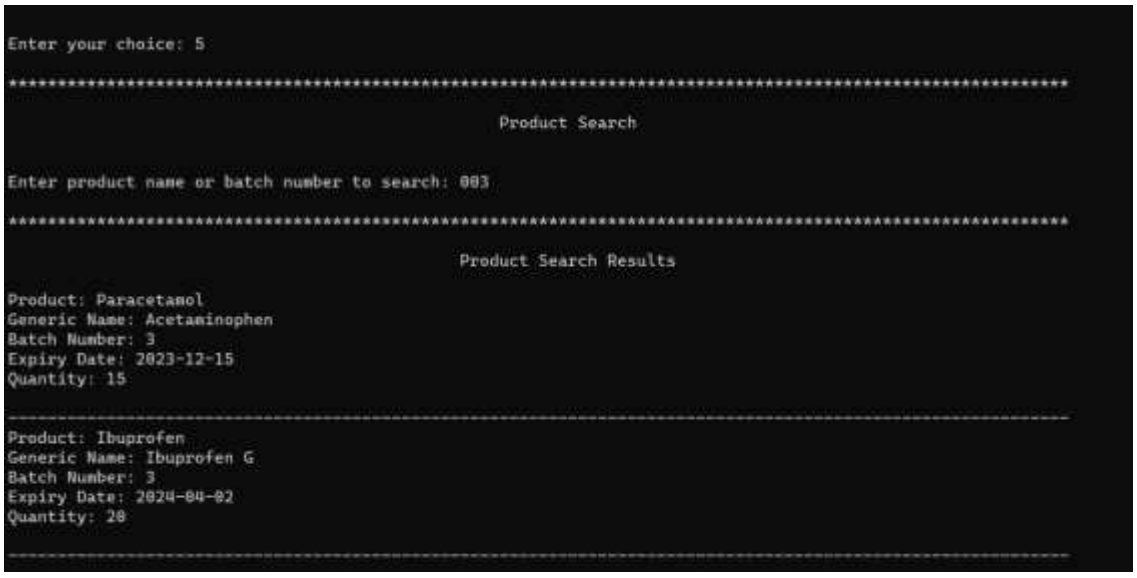
Product Search

The searchProduct function is designed to search for products in the inventory based on user input. It takes two parameters: inventory, an array of struct Product, and productCount, an integer representing the number of products in the inventory.

Code :

- The function starts by declaring a character array searchQuery of size 100 to store the user's search input. It also initializes an integer variable found to 0. The found variable serves as a flag to indicate if a matching product is found during the search.
- The user enters their search query, which is stored in the searchQuery array.
- The function checks if the search query is empty (length is 0). If the search query is empty, it means the user did not provide any input, and an error message is displayed.
- If the search query is not empty, the function proceeds to display a header for the search results section and iterates through the inventory array to find matching products.
- During the iteration, the function compares the search query (case-insensitive) with each product's name and batch number. If a match is found, the product details are displayed, including the name, generic name, batch number, expiry date, and quantity.
- If at least one matching product is found, the found flag is set to 1.
- After the iteration, if no matching products are found (i.e., found remains 0), a message is displayed indicating that no matching products were found.

View :



Sales Transactions

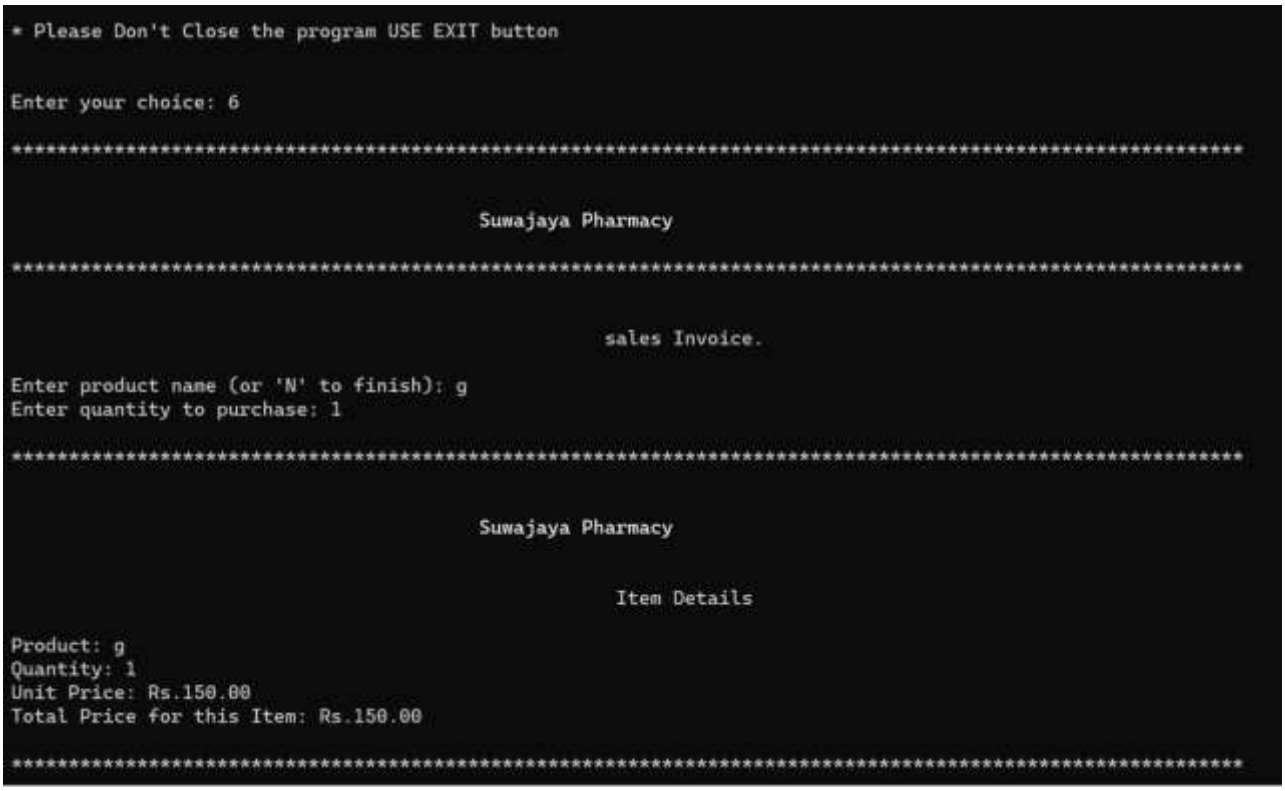
The Salesandinvoice function is responsible for handling the sales process and generating a sales invoice for the products purchased by the customer. It takes four parameters: inventory, an array of struct Product, productCount, an integer representing the number of products in the inventory, suppliers, an array of struct Supplier, and supplierCount, an integer representing the number of suppliers.

Code :

- It enters a loop to process the customer's product selection. The customer is prompted to enter the name of the product they want to purchase. If the user enters "N" (case-insensitive), the loop breaks, indicating that the customer has finished adding products to the invoice.
- For each product entered by the customer, the function checks if the product name matches any product in the inventory (case-insensitive) and whether the requested quantity is valid (greater than 0 and available in the inventory).
- If the product name is not found or the requested quantity is invalid, an error message is displayed, and the customer is prompted to re-enter the product details.

- If a valid product is found, its details are displayed, including the name, quantity, unit price, and total price for the current item. The total price of all purchased items is updated accordingly, and the product's quantity is deducted from the inventory.
- The loop continues until the customer indicates they are finished adding products (by entering "N").
- After the loop, the function checks if any items were added to the invoice (i.e., productFound is 0). If not, a message is displayed, indicating that no items were added, and the function returns.
- If items were added to the invoice, the function prompts the user to select a payment method from the available options.
- The function checks if the selected payment method is valid. If not, an error message is displayed, and the function returns.
- After a valid payment method is chosen, the function displays a confirmation message of the successful payment and thanks the customer for their purchase.
- The function then generates and issues the sales receipt. It includes details such as the receipt number, date, time, total amount paid, and the chosen payment method.
- The receipt is displayed on the screen, and the receipt number is incremented for future transactions.

View :



```
Enter product name (or 'N' to finish): a
Enter quantity to purchase: 5

*****************************************************************************************************

                                    Suwajaya Pharmacy

                                            Item Details

Product: a
Quantity: 5
Unit Price: Rs.120.00
Total Price for this Item: Rs.600.00

*****************************************************************************************************

Enter product name (or 'N' to finish): n

Select Payment Method:
1. Cash
2. Credit Card
Enter your choice: 1

Payment Method: Cash

                              Payment successful. Thank you for your purchase!

                                    Suwajaya Pharmacy
                                       No.256,Kotte Road,Nugegoda

*****************************************************************************************************

Receipt Number: 1
Date and Time: 01/08/2023 21:02
Total Amount: Rs.750.00
Payment Method: Cash
Thank you for your purchase!

*****************************************************************************************************
```

Product Returns and Refunds

The handleReturns function handles the process of product returns in the inventory of a pharmacy. It takes two parameters: inventory, an array of struct Product, and productCount, an integer representing the number of products in the inventory.

Code :

```
329 void handleReturns(struct Product inventory[], int productCount) {
330     char productName[100];
331     int returnedQuantity;
332
333     // Prompt the user to enter the product name and quantity returned
334     printf("*****************************************************************************************************************\n\n");
335     printf("                                                Product Return                                                   \n\n");
336     printf("\nEnter product name for return: ");
337     scanf(" %[^\n]", productName);
338
339     printf("Enter quantity returned: ");
340     scanf("%d", &returnedQuantity);
341
342     // Find the product in the inventory and update the quantity
343     int foundProductIndex = -1;
344     for (int i = 0; i < productCount; i++) {
345         if (strcasecmp(productName, inventory[i].name) == 0) {
346             foundProductIndex = i;
347             break;
348         }
349     }
350
351     if (foundProductIndex == -1 || returnedQuantity <= 0) {
352         printf("                                Invalid product name or quantity for return.                          \n");
353         return;
354     }
355
356     inventory[foundProductIndex].quantity += returnedQuantity;
357
358     printf("\n                                Product return processed successfully.                                   \n");
359 }
360
```

- It displays a prompt to the user, asking them to enter the name of the product they want to return and the quantity they are returning.
- The user's input for the product name and returned quantity is read from the console.
- The function then searches for the product in the inventory based on the entered product name (case-insensitive). If the product is found, its index in the inventory array is stored in the variable foundProductIndex.
- If the product is not found in the inventory or the returned quantity is invalid (less than or equal to zero), the function displays an error message and returns without further processing.
- If the product is found and the returned quantity is valid, the inventory is updated by adding the returned quantity to the current quantity of the product at the foundProductIndex in the inventory array.
- After updating the inventory, the function displays a message indicating that the product return was processed successfully.

View :

```
* Please Don't Close the program USE EXIT button


Enter your choice: 7
*****************************************************************************************************************

                                                Product Return


Enter product name for return: g
Enter quantity returned: 3

                                Product return processed successfully.
*****************************************************************************************************************
```

Sales Reports

The generateSalesReport function is responsible for generating a sales report for the pharmacy's inventory. It takes four parameters: inventory, an array of struct Product, productCount, an integer representing the number of products in the inventory, suppliers, an array of struct Supplier, and supplierCount, an integer representing the number of suppliers.

Code :

```c
// Function to generate sales reports
void generateSalesReport(struct Product inventory[], int productCount, struct Supplier suppliers[], int supplierCount) {

    printf("*****************************************************************************************************\n\n");
    printf("                                        Sales Report                                                 \n\n");

    // Print the table header
    printf("%-20s | %-12s | %-10s | %-16s | %-12s\n", "Product", "Batch Number", "Quantity", "Price Per Unit", "Total Revenue");
    printf("\n\n-----------------------------------------------------------------------------------------------------\n\n");

    // Variables to store total sales and revenue
    int totalSales = 0;
    float totalRevenue = 0.0;

    for (int i = 0; i < productCount; i++) {
        int productSales = inventory[i].quantity; // Assuming each sold item represents a sale
        float productRevenue = productSales * inventory[i].price;

        printf("%-20s | %-12d | %-10d | $%-15.2f | $%-12.2f\n", inventory[i].name, inventory[i].batch_number, productSales, inventory[i].price, productRevenue);

        // Accumulate the total sales and revenue
        totalSales += productSales;
        totalRevenue += productRevenue;
    }

    printf("\n-----------------------------------------------------------------------------------------------------\n\n");
    printf("%-20s | %-12s | %-10d | %-16s | $%-12.2f\n", "Overall Total", "", totalSales, "", totalRevenue);
    printf("\n\n*****************************************************************************************************\n");
}
```

- It prints a table header that includes column titles: "Product," "Batch Number," "Quantity," "Price Per Unit," and "Total Revenue."
- The function then initializes two variables, totalSales (to store the total quantity of products sold) and totalRevenue (to store the total revenue generated from sales), both initially set to 0.
- It enters a loop that iterates through each product in the inventory.
- For each product, the function calculates the total sales (assuming each sold item represents a sale) and the total revenue for that specific product based on its quantity and price.
- The function then prints a row in the table containing the product name, batch number, quantity sold, price per unit, and the total revenue generated from selling that product.
- It then prints the overall total, which includes the total quantity of products sold (totalSales) and the total revenue generated from all sales (totalRevenue).

View :

```
Enter your choice: 8
*****************************************************************************************************

                                        Sales Report

Product              | Batch Number | Quantity   | Price Per Unit   | Total Revenue


-----------------------------------------------------------------------------------------------------

g                    | 1            | 5          | Rs.150.00        | Rs.750.00
a                    | 2            | 12         | Rs.120.00        | Rs.1440.00
Paracetamol          | 3            | 15         | Rs.36.50         | Rs.547.50
Ibuprofen            | 3            | 20         | Rs.50.00         | Rs.1000.00

-----------------------------------------------------------------------------------------------------

Overall Total        |              | 52         |                  | Rs.3737.50


*****************************************************************************************************
*****************************************************************************************************
```

Full Code

```
#include <stdio.h>
#include <string.h>
#include <time.h>

// Constants
#define MAX_PRODUCTS 1000
#define MAX_SUPPLIERS 1000



// Define structures to represent product and supplier information
struct Product {
    char name[100];
    char generic_name[100];
    int batch_number;
    char expiry_date[20];
    int quantity;
    float price;
};

struct Supplier {
    char name[100];
    char contact_details[100];
    char terms[100];

};

//load data from file
void loadData(struct Product inventory[], int *productCount, struct Supplier suppliers[], int *supplierCount) {
    FILE* file = fopen("inventory_data.txt", "r");

    if (file == NULL) {
        printf("Error opening the file for loading data.\n");
        return;
    }
```

```
    // Read product count
    fscanf(file, "Product Count: %d\n", productCount);

    // Read product data
    for (int i = 0; i < *productCount; i++) {
        fscanf(file, "Product Name: %[^\n]\n", inventory[i].name);
        fscanf(file, "Generic Name: %[^\n]\n", inventory[i].generic_name);
        fscanf(file, "Batch Number: %d\n", &inventory[i].batch_number);
        fscanf(file, "Expiry Date: %[^\n]\n", inventory[i].expiry_date);
        fscanf(file, "Quantity: %d\n", &inventory[i].quantity);
        fscanf(file, "Price: %f\n", &inventory[i].price);
    }

    // Read supplier count
    fscanf(file, "Supplier Count: %d\n", supplierCount);

    // Read supplier data
    for (int i = 0; i < *supplierCount; i++) {
        fscanf(file, "Supplier Name: %[^\n]\n", suppliers[i].name);
        fscanf(file, "Contact Details: %[^\n]\n", suppliers[i].contact_details);
        fscanf(file, "Terms: %[^\n]\n", suppliers[i].terms);
    }

    fclose(file);
}


// Function to add a new product to the inventory
void addProduct(struct Product inventory[], int *productCount) {
    if (*productCount >= MAX_PRODUCTS) {
        printf("Inventory is full. Cannot add more products.\n");
        return;
    }
```

```
printf("*****************************************************************************************************
*********\n\n");
   printf("                                Add New Product                                \n\n");

   int choice;
   do {
      // Prompt the user to enter product details
      printf("\nEnter product name: ");
      scanf(" %[^\n]", inventory[*productCount].name);

      printf("Enter generic name: ");
      scanf(" %[^\n]", inventory[*productCount].generic_name);

      printf("Enter batch number: ");
      scanf("%d", &inventory[*productCount].batch_number);

      printf("Enter expiry date: ");
      scanf(" %[^\n]", inventory[*productCount].expiry_date);

      printf("Enter quantity: ");
      scanf("%d", &inventory[*productCount].quantity);

      printf("Enter price: ");
      scanf("%f", &inventory[*productCount].price);

      (*productCount)++;



      printf("Product added successfully.\n");

      // Ask the user if they want to add more products
      printf("Do you want to add more products? (1: Yes / 0: No): ");
      scanf("%d", &choice);
   } while (choice == 1);
```

```
}

// Function to track stock levels and generate alerts
void trackStock(struct Product inventory[], int productCount) {

printf("**********************************************************************************************************
*********\n\n");
   printf("\n                                Stock Alert                                \n\n");
   printf("\nProducts with low stock levels:\n");

   for (int i = 0; i < productCount; i++) {
      if (inventory[i].quantity < 10) {
         printf("\n\nProduct: %s (Batch Number: %d)\n", inventory[i].name, inventory[i].batch_number);
      }
   }


printf("\n**********************************************************************************************************
**********\n");
}

// Function to view the remaining inventory

void viewInventory(struct Product inventory[], int productCount, struct Supplier suppliers[]) {


printf("**********************************************************************************************************
*********\n\n");
   printf("                          Suwajaya Pharmacy- Inventory                          \n\n");

printf("**********************************************************************************************************
*********\n\n");
   printf("Product Name      | Batch Number | Quantity | Total Value (Rs.)\n");
   printf("\n----------------------------------------------------------------------------------------------------\n");

   float totalValue = 0.0;
```

```
    for (int i = 0; i < productCount; i++) {
        int batchNumber = inventory[i].batch_number;
        char* productName = inventory[i].name;
        int quantity = inventory[i].quantity;
        float itemValue = quantity * inventory[i].price;

        printf("%-20s | %-12d | %-8d | %-16.2f\n", productName, batchNumber, quantity, itemValue);
        totalValue += itemValue;
    }

    printf("\n-------------------------------------------------------------------------------------------------------\n");
    printf("Total Value (All Products)                    | %-16.2f\n", totalValue);

printf("\n*****************************************************************************************************\n\n");
}



// Function to add a new supplier to the supplier database
void addSupplier(struct Supplier suppliers[], int *supplierCount) {
    if (*supplierCount >= MAX_SUPPLIERS) {


printf("\n*****************************************************************************************************\n\n");
    printf("\n                    Supplier database is full. Cannot add more suppliers.                    \n");
        return;
    }

    // Prompt the user to enter supplier details

printf("\n*****************************************************************************************************\n\n");
    printf("                              Enter Supplier Detailes                              \n\n");
    printf("\nEnter supplier name: ");
    scanf(" %[^\n]", suppliers[*supplierCount].name);
```

```
    printf("Enter contact details: ");
    scanf(" %[^\n]", suppliers[*supplierCount].contact_details);

    printf("Enter terms: ");
    scanf(" %[^\n]", suppliers[*supplierCount].terms);


    // Increment the supplier count
    (*supplierCount)++;

    printf("                    Supplier added successfully.                    \n");
}



// Function to search for products by name or batch number
void searchProduct(struct Product inventory[], int productCount) {
    char searchQuery[100];
    int found = 0; // Flag to indicate if a matching product is found

printf("\n*********************************************************************************************
***********\n\n");
    printf("                    Product Search                    \n\n");
    printf("\nEnter product name or batch number to search: ");
    scanf(" %[^\n]", searchQuery);

    // Check if the search query is empty
    if (strlen(searchQuery) == 0) {

        printf("Invalid search query. Please enter a valid product name or batch number.\n");
        return;
    }

printf("\n*********************************************************************************************
***********\n\n");
```

```
    printf("                    Product Search Results                    \n\n");
    for (int i = 0; i < productCount; i++) {

        // Check if the search query matches the product name or batch number (case-insensitive)
        if (strcasecmp(searchQuery, inventory[i].name) == 0 || atoi(searchQuery) == inventory[i].batch_number) {
            printf("Product: %s\n", inventory[i].name);
            printf("Generic Name: %s\n", inventory[i].generic_name);
            printf("Batch Number: %d\n", inventory[i].batch_number);
            printf("Expiry Date: %s\n", inventory[i].expiry_date);
            printf("Quantity: %d\n", inventory[i].quantity);
            printf("\n--------------------------------------------------------------------------------------------------------\n");
            found = 1; // Set the flag to indicate a match is found
        }
    }

    if (!found) {
        printf("\n                    No matching products found.                    \n");
    }


printf("\n*****************************************************************************************************
***********\n\n");


}

// Function to handle sales, generate invoices, and issue receipts
void Salesandinvoice(struct Product inventory[], int productCount, struct Supplier suppliers[], int supplierCount) {
    const char* paymentMethods[] = {"Cash", "Credit Card"};
    int paymentMethodChoice;
    float totalPrice = 0.0;
    int productFound = 0;

    // Display a welcome message and initialize the receipt details


printf("\n*****************************************************************************************************
***********\n\n");
```

```
    printf("\n\033[1m                    Suwajaya Pharmacy                    \033[0m\n\n");

printf("**************************************************************************************************
*********\n\n");
    printf("\n                          sales Invoice.                    \n");


    do {
        char productName[100];
        int quantity;
        int foundProductIndex =-1;

        // Prompt the user to enter the product name and quantity
        printf("\nEnter product name (or 'N' to finish): ");
        scanf(" %[^\n]", productName);

        // Check if the user entered "N" to finish adding products
        if (strcasecmp(productName, "N") == 0) {
            break;
        }

        printf("Enter quantity to purchase: ");
        scanf("%d", &quantity);

        // Find the product in the inventory and get its supplier details
        for (int i = 0; i < productCount; i++) {
            if (strcasecmp(productName, inventory[i].name) == 0) {
                foundProductIndex = i;
                break;
            }
        }

        if (foundProductIndex ==-1 || quantity > inventory[foundProductIndex].quantity || quantity <= 0) {
            printf("Invalid product name or quantity. Please try again.\n");
            continue; // Go back to the beginning of the loop to re-enter the product details
        } else {
            // Calculate the total price for the purchase of this item
```

```
            float itemPrice = quantity * inventory[foundProductIndex].price;
            totalPrice += itemPrice;
            productFound = 1;

            // Display the product details for the current item

printf("\n*****************************************************************************************************
***********\n\n");
            printf("\n\033[1m                    Suwajaya Pharmacy                    \033[0m\n\n");
            printf("\n                    Item Details                    \n");
            printf("\nProduct: %s\n", inventory[foundProductIndex].name);
            printf("Quantity: %d\n", quantity);
            printf("Unit Price: Rs.%.2f\n", inventory[foundProductIndex].price);
            printf("Total Price for this Item: Rs.%.2f\n", itemPrice);

printf("\n*****************************************************************************************************
***********\n\n");

            // Update the inventory by deducting the sold quantity
            inventory[foundProductIndex].quantity-= quantity;
        }
    } while (1);

    if (!productFound) {
        printf("\n                    No items added to the invoice.                    \n");
        return;
    }

    // Prompt the user to select a payment method
    printf("\nSelect Payment Method:\n");
    for (int i = 0; i < 2; i++) {
        printf("%d. %s\n", i + 1, paymentMethods[i]);
    }
    printf("Enter your choice: ");
    scanf("%d", &paymentMethodChoice);

    // Check if the selected payment method is valid
```

```
    if (paymentMethodChoice < 1 || paymentMethodChoice > 2) {
        printf("Invalid payment method choice.\n");
        return;
    }

    printf("\nPayment Method: %s\n", paymentMethods[paymentMethodChoice- 1]);
    printf("\n                    Payment successful. Thank you for your purchase!                    \n");

    // Generate and issue the receipt
    time_t now;
    time(&now);
    struct tm* timeInfo = localtime(&now);
    static int receiptCounter = 1;

    printf("\n\033[1m                    Suwajaya Pharmacy                    \033[0m");
    printf("\n                    No.256,Kotte Road,Nugegoda                    \n\n");

printf("**********************************************************************************************************
*********\n\n\n");

    printf("Receipt Number: %d\n", receiptCounter++);
    printf("Date and Time: %02d/%02d/%d %02d:%02d\n", timeInfo->tm_mday, timeInfo->tm_mon + 1, timeInfo->tm_year + 1900,
timeInfo->tm_hour, timeInfo->tm_min);
    printf("Total Amount: Rs.%.2f\n", totalPrice);
    printf("Payment Method: %s\n", paymentMethods[paymentMethodChoice- 1]);
    printf("Thank you for your purchase!\n");

printf("\n**********************************************************************************************************
**********\n\n");
}

// Function to process product returns and refunds
void handleReturns(struct Product inventory[], int productCount) {
```

```
    char productName[100];
    int returnedQuantity;

    // Prompt the user to enter the product name and quantity returned

printf("**********************************************************************************************************
*********\n\n");
    printf("                                Product Return                                \n\n");
    printf("\nEnter product name for return: ");
    scanf(" %[^\n]", productName);

    printf("Enter quantity returned: ");
    scanf("%d", &returnedQuantity);

    // Find the product in the inventory and update the quantity
    int foundProductIndex =-1;
    for (int i = 0; i < productCount; i++) {
        if (strcasecmp(productName, inventory[i].name) == 0) {
            foundProductIndex = i;
            break;
        }
    }

    if (foundProductIndex ==-1 || returnedQuantity <= 0) {
        printf("                        Invalid product name or quantity for return.                \n");
        return;
    }

    inventory[foundProductIndex].quantity += returnedQuantity;

    printf("\n                        Product return processed successfully.                        \n");
}

// Function to generate sales reports
void generateSalesReport(struct Product inventory[], int productCount, struct Supplier suppliers[], int supplierCount) {
```

```
printf("**********************************************************************************************************
*********\n\n");
   printf("                                Sales Report                                \n\n");

   // Print the table header
   printf("%-20s | %-12s | %-10s | %-16s | %-12s\n", "Product", "Batch Number", "Quantity", "Price Per Unit", "Total Revenue");
   printf("\n\n-------------------------------------------------------------------------------------------------------\n\n");

   // Variables to store total sales and revenue
   int totalSales = 0;
   float totalRevenue = 0.0;

   for (int i = 0; i < productCount; i++) {
      int productSales = inventory[i].quantity; // Assuming each sold item represents a sale
      float productRevenue = productSales * inventory[i].price;

      printf("%-20s | %-12d | %-10d | Rs.%-15.2f | Rs.%-12.2f\n", inventory[i].name, inventory[i].batch_number, productSales,
inventory[i].price, productRevenue);

      // Accumulate the total sales and revenue
      totalSales += productSales;
      totalRevenue += productRevenue;
   }

   printf("\n-------------------------------------------------------------------------------------------------------\n\n");
   printf("%-20s | %-12s | %-10d | %-16s | Rs.%-12.2f\n", "Overall Total", "", totalSales, "", totalRevenue);

printf("\n\n**********************************************************************************************************
************\n");
}

void saveData(struct Product inventory[], int productCount, struct Supplier suppliers[], int supplierCount) {
   FILE* file = fopen("inventory_data.txt", "w");

   if (file == NULL) {
      printf("Error opening the file for saving data.\n");
```

```
        return;
    }

    // Save inventory data
    fprintf(file, "Product Count: %d\n", productCount);
    for (int i = 0; i < productCount; i++) {
        fprintf(file, "Product Name: %s\n", inventory[i].name);
        fprintf(file, "Generic Name: %s\n", inventory[i].generic_name);
        fprintf(file, "Batch Number: %d\n", inventory[i].batch_number);
        fprintf(file, "Expiry Date: %s\n", inventory[i].expiry_date);
        fprintf(file, "Quantity: %d\n", inventory[i].quantity);
        fprintf(file, "Price: %.2f\n", inventory[i].price);
    }

    // Save supplier data
    fprintf(file, "Supplier Count: %d\n", supplierCount);
    for (int i = 0; i < supplierCount; i++) {
        fprintf(file, "Supplier Name: %s\n", suppliers[i].name);
        fprintf(file, "Contact Details: %s\n", suppliers[i].contact_details);
        fprintf(file, "Terms: %s\n", suppliers[i].terms);
    }

    fclose(file); // Add the closing parenthesis here
}


int main() {
    // Define arrays to hold product and supplier data
    struct Product inventory[MAX_PRODUCTS];
    struct Supplier suppliers[MAX_SUPPLIERS];

    // Initialize variables to keep track of the number of products and suppliers
    int productCount = 0;
    int supplierCount = 0;
```

```
    loadData(inventory, &productCount, suppliers, &supplierCount);

    // Implement the main program loop where users can interact with the system
    int choice;
    do {
        // Get the current date and time
        time_t now;
        struct tm *localTime;
        char dateStr[128];
        char TimeStr[128];

        time(&now);
        localTime = localtime(&now);

        strftime(dateStr, sizeof(dateStr), "%Y-%m-%d", localTime);
        strftime(TimeStr, sizeof(TimeStr), "%H:%M:%S", localTime);

        // Display a menu of options for the user

printf("*****************************************************************************************************
*********\n");
        printf("\n                    Pharmacy Inventory and Billing System                    ");
        printf("\n\033[1m                          Suwajaya Pharmacy                     \033[0m");
        printf("\n                       No.256,Kotte Road,Nugegoda                    \n\n");

printf("*****************************************************************************************************
*********\n\n");

        // Display the current date and time
        printf(" Date     : %s\n", dateStr);
        printf(" Time     : %s\n", TimeStr);
        printf(" Location : Nugegoda, Sri Lanka\n\n\n");


        printf("1. Add Product\n");
        printf("2. View Inventory\n");
        printf("3. Track Stock\n");
```

```
printf("4. Add Supplier\n");
printf("5. Search Product\n");
printf("6. Sales Invoice\n");
printf("7. Sales Returns\n");
printf("8. Generate Sales Report\n");
printf("0. Exit\n");
printf("\n* Please Don't Close the program USE EXIT button \n");
printf("\n\nEnter your choice: ");
scanf("%d", &choice);

// Use a switch statement to execute the chosen option
switch (choice) {
    case 1:
        addProduct(inventory, &productCount);
        break;
    case 2:
        viewInventory(inventory, productCount, suppliers);
        break;
    case 3:
        trackStock(inventory, productCount);
        break;
    case 4:
        addSupplier(suppliers, &supplierCount);
        break;
    case 5:
        searchProduct(inventory, productCount);
        break;
    case 6:
        Salesandinvoice(inventory, productCount, suppliers, supplierCount);
        break;
    case 7:
        handleReturns(inventory, productCount);
        break;
    case 8:
        generateSalesReport(inventory, productCount, suppliers, supplierCount);
```

```
            break;
        case 0:
            printf("\n                    Exiting the program...                    \n");
            saveData(inventory, productCount, suppliers, supplierCount); // Save the data before exiting
            break;
        default:
            printf("\n                Invalid choice. Please try again.                \n");
            break;
    }
  } while (choice != 0);

  return 0;
}
```

Git Repository link : https://github.com/thaveeshvihanga/Pharmacy_Inventory_and_Billing_System1.git

I have submitted the C file with this document.

Thank you !