RETAIL MANAGEMENT SYSTEM

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

NIKITHA MARAMRAJU - 1602-19-737-145

A. SRAVYA REDDY – 1602-19-737-175

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahim Bagh, Hyderabad-31

2020

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, NIKITHA MARAMRAJU and A. SRAVYA REDDY bearing hall ticket numbers, 1602-19-737-145 aand1602-19-737-175, hereby declare that the project report entitled “RETAIL MANAGEMENT SYSTEM” is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

NIKITHA MARAMRAJU

1602-19-737-145

A. SRAVYA REDDY

1602-19-737-175

(Faculty In-Charge) (Head.Dept IT)

ACKNOWLEDGEMENTS

We are overwhelmed in all humbleness and gratefulness to acknowledge our debt to all those who have helped us put these ideas, well above the level of simplicity and into something concrete. We would like to thank our college who gave us the golden opportunity to do this wonderful project "Retail Management System" which helped us in learning many new things. We are really obliged to them.

We owe our respectable regards to Ms. L. Divya, Assistant Professor, Vasavi College of Engineering, Dr. Ramesh Vasappanavara, Professor, Vasavi College of Engineering, for their support, guidance and valuable suggestions in the project.

We wish to express our affectionate gratitude to our friends, Srilekhya Turlapati and Soma Santosh Kumar for their warm friendship, co-operation and encouragement.

NIKITHA MARAMRAJU

1602-19-737-145

A. SRAVYA REDDY

Date: 18-12-2020 1602-19-737-175

ABSTRACT

Nowadays, people generally prefer online shopping rather than roaming at different places and purchasing the products, because by doing so it saves a lot of time and fuel. Online shopping is a process in which people (specifical customers) are being provided with the option of purchasing goods and services directly from the seller, all-in real-time environment. Our project practically shows the functionality in any online shopping system.

This mini project in C called “RETAIL MANAGEMENT SYSTEM” is a simple, user-friendly application designed to show the practical demonstration of features used in Online Shopping System. In this project there are two ends- The Retailer end and The Customer end. The main goal of this project is to record stock, add new products, refill stock items, search for products at the retailer end and allows to view products, make purchase and receive invoice within the system at the customer end.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Problem domain in general

There was a time, in the not-so-distant past, when the only option for purchasing groceries was to head to the local grocery store. Fast forward to today, and consumers have a growing number of online grocery shopping options.

Nowadays lead busy lives and are always on the lookout for ways to maximize their time. So, as anticipated, saving time is the most popular reason consumers purchase groceries online, with 72% indicating it's a key reason they choose to do so.

Grocery shopping is a sensory experience—much more so than other product categories. Consumers who shop in a store have the opportunity to touch, smell and even taste items to assess their quality and freshness. It's much more difficult to judge the quality of fresh food items—such as meat and produce—when shopping online. So, it comes as no surprise that shelf-stable goods are the most popular grocery items to purchase online.

So now comes the importance of rating and reviews. Regardless of what a customer is buying, not everyone but most of them prefer to go through the reviews. So, creating an image is very important which gradually increases the number of customers trusting you and thereby increasing the profit margins.

1.2 Project Introduction

We have designed a mini project, using C programming including some of the basic features like viewing stock, editing stock, making purchase, displaying invoice, etc. This project "Retail Management System In C" is complete and totally-error free clean code. The main goal of this project is to record stock, add new products, refill stock items, search for products at the retailer end and allows to view products, make purchase and receive invoice within the system at the customer end.

The source code of this “Retail Management System in C” is about 1500 lines. Also, it is password protected. This project is just like real store Management software with all simple and basic features. It doesn’t consist of high graphics. It is a simple one and users can understand it very easily.

1.3 List of features in the project

Our Project basically has two ends – Retailer-end and customer-end.

Features of Retailer:

- View the product details such as product-id, name, price, stock available present the store.
- Add new products to the store with their specifications.
- Search the product by – ID, name, stock and price.
- Add stock to the existing products in the store.

Features of Customer:

- View the catalogue of Products with their product-id, Price and stock available.
- Choose the items to be purchased.
- Purchase the products by entering the stock required.
- Get the Invoice(bill) of the purchased products.

2. TECHNOLOGY

All computer software needs certain hardware components and also other software resources to be present, in order for computers to be used efficiently. These pre-requisites are known as System Requirements. System Requirements are of two types – Software Requirements and Hardware Requirements.

2.1. Software Requirements

Software Requirements deal with defining the software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These preconditions are generally not included in the software package and need to be installed separately.

In order to use Retail Management System, one should have the following software requirements:

- Operating System: Windows 7 and above
- C compiler: GNU Compiler Collection (GCC).
- Editor: Any Text Editor that supports C language

2.2. Hardware Requirements

Hardware Requirements refer to the common set of requirements defined by any operating system or software application and are usually the physical computer resources. In this we look into the architecture, processing power, memory, secondary memory, display adapter and peripherals.

In order to use Retail Management System, one should have the following hardware requirements:

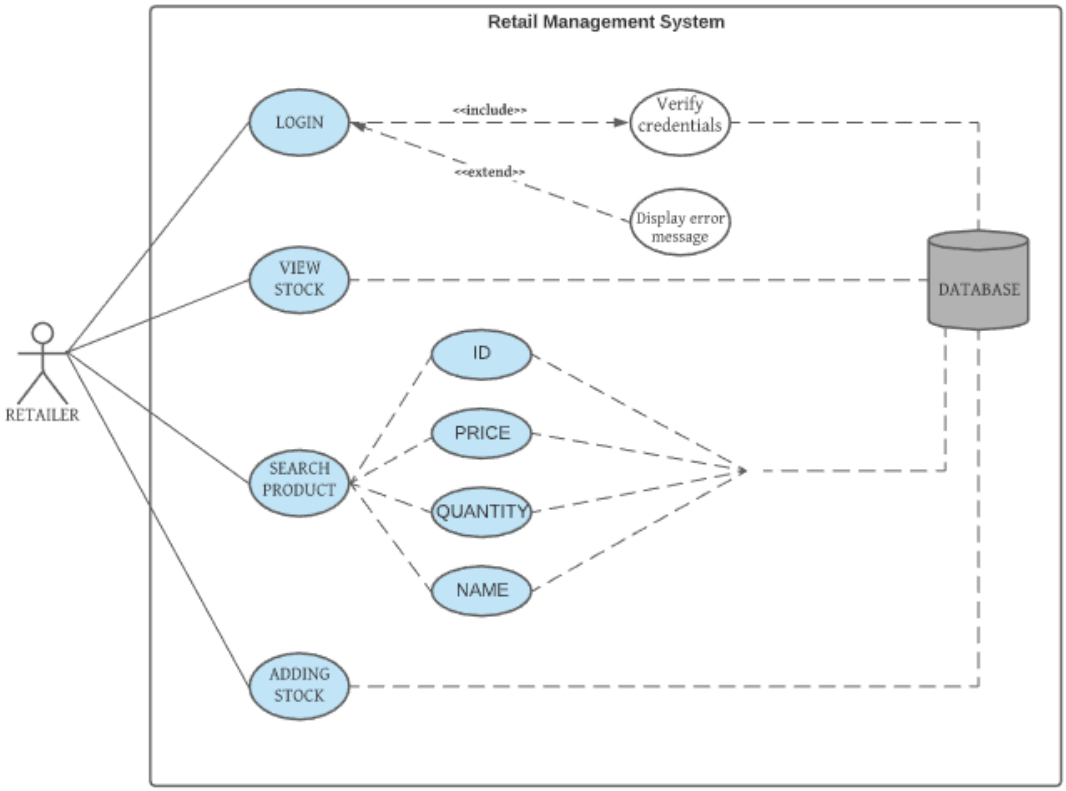
- Processor: Intel Core i5 and above
- Memory: 4 GB RAM and above

3. PROPOSED WORK

3.1. DESIGN.

3.1.1. RETAILER USE CASES

A retailer can login, view products available, add new products, search product, add stock to the existing products.



3.1.1.1. LOGIN

The login interface of the retailer is hard-coded. The retailer can login and view the catalogue of options only on entering the correct password. On entering the wrong password, the system prompts an error message and asks the retailer to re-enter the password again.

3.1.1.2. VIEW PRODUCT DETAILS

When the retailer chooses this option, list of products available for sale along with their details a printed on the screen. The details of the products are stored in a file which are retrieved when this function is called. Initially when no products are available i.e., when the file is empty, it displays a message "STOCK IS EMPTY".

3.1.1.3. ADD NEW PRODUCTS

When the retailer chooses this option, the system asks to enter a product ID. The system checks if the entered product ID already exists in the stock of the products available. If the ID entered by the retailer already exists system displays an error message the retailer to enter another ID. If the ID entered by the retailer is not a pre-existing one then the system asks the retailer to enter the details of the product i.e., price of the product, quantity of the product and name of the product. All these details are saved into the file.

3.1.1.4. SEARCH FOR A PRODUCT

When the retailer chooses this option, the system displays for different choices - searching by product ID, searching by price range, searching by quantity range and searching by name. The retailer has to make his choice.

3.1.1.4.1. By product ID

When the retailer selects the option to search a product by its ID, the system prompts the retailer to enter a product ID. The system checks if the product ID entered by the retailer matches with any of the product ID in the pre-existing stock. If the ID matches, it displays the details of the product. If the product ID does not match with any of the IDs of the pre-existing stock displays an error message “PRODUCT NOT FOUND” and returns back to the catalogue.

3.1.1.4.2. By price range

When the retailer selects the option to search a product by its price, the system prompts the retailer to enter the lower and upper price range. The system checks if the

price of any of the products lies in the range entered by the retailer. If the price matches, it displays all the details of the product. If no products fall in the price range as entered by the retailer, the system displays error message “PRODUCT NOT FOUND” and returns back to the catalogue.

3.1.1.4.3. By quantity range

When the retailer selects the option to search a product by its quantity, the system prompts the retailer to enter the lower and upper quantity range. The system checks if the quantity of any of the products lies in the range entered by the retailer. If the quantity matches, it displays all the details of the product. If no products fall in the quantity range as entered by the retailer, the system displays error message “PRODUCT NOT FOUND” and returns back to the catalogue.

3.1.1.4.4. By name

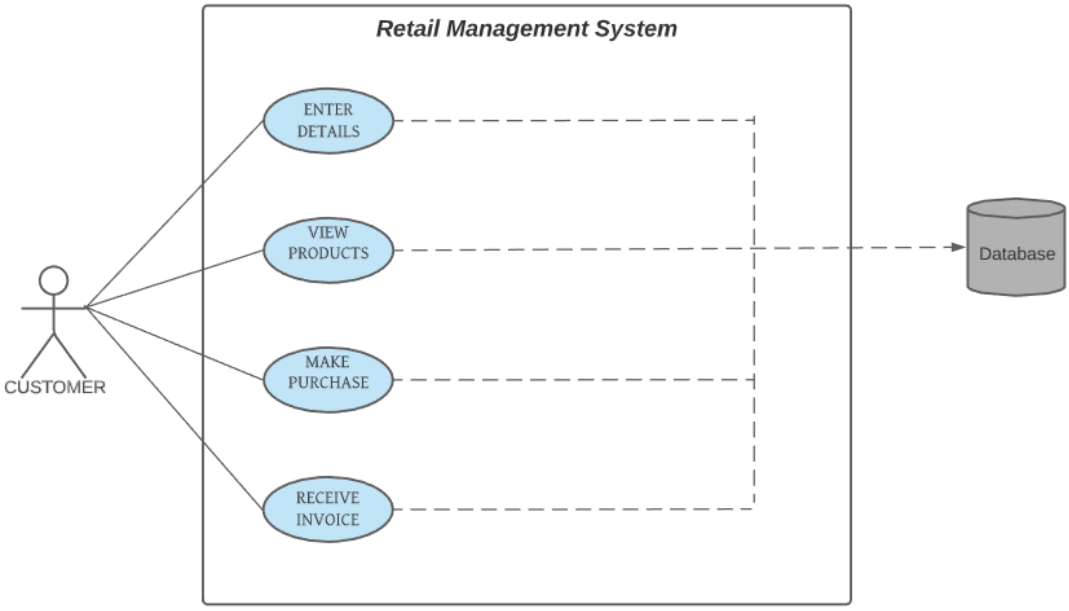
When the retailer selects the option to search a product by its name, the system prompts the retailer to enter a product name. The system checks if the product name entered by the retailer matches with any of the product name in the pre-existing stock. If the name matches, it displays the details of the product. If the product name does not match with any of the IDs of the pre-existing stock displays an error message “PRODUCT NOT FOUND” and returns back to the catalogue.

3.1.1.5. ADDING STOCK TO EXISTING PRODUCT

When the retailer chooses this option, system asks the retailer to enter the product ID. The system checks for the validity of the product ID. If the code is valid it displays the current stock available of the particular product and asks retailer wishes to update the stock. If the retailer chooses yes, it updates the

stock as per the quantity given by the user and prints the updated quantity. If the retailer chooses no, it goes back to the main menu. If in case the product ID entered by the retailer is invalid it displays an error message and asks the retailer to enter the ID again.

3.1.2. CUSTOMER USE CASES



A customer can sign-in, view the catalogue of products with their specifications, purchase a product, receive the invoice.

3.1.2.1. SIGN-UP

In this sign in option, the system prompts the customer to enter his name and mobile number. The system checks if the mobile number is valid i.e., contains only 10 digits and no characters and proceeds to the purchase section. If the phone number entered is invalid it prompts the customer to enter his details again.

3.1.2.2. VIEW CATALOGUE OF PRODUCTS

The details of the stock of products available is retrieved from the file and all the details of the products available for sale are displayed on the screen and the customer can choose to make purchase or can choose to exit.

3.1.2.3. MAKE PURCHASE

In this purchase option, the system prompts the customer if he wishes to make purchase. If the customer chooses yes, then the system asks him to enter the number of products he wishes to purchase and the IDs of products he wishes to purchase. The system then displays the quantity available for each product and prompts the user to enter the quantity he wishes to buy. If the quantity entered by the user is sufficient enough, stock is automatically deducted. If the quantity entered by the user is insufficient, displays an error message and prompts the customer to change the quantity entered. If the customer chooses no when he is asked to make purchase, then he is redirected to the main menu.

3.1.2.4. GET THE INVOICE

When the customer selects this option, the system displays the invoice which consist of the name of the customer, phone number of the customer, date and time of billing along with product name and price purchased. The total amount is also displayed.

3.2. Implementation

3.2.1. Module-Wise Code

3.2.1.1 Login function of Retailer

- **Function name:** void login_retailer()
- **Functionality:** This user-defined function is to enable the retailer to enter a password and access the account and perform desired options like add stock, search product, update stock etc.

```
void login_retailer()
{
        system("CLS");

        char pass[13] = "miniproject", inputpass[30], input_char;
        int counter = 0, count = 0;

        outputcursorposition(74, 15);
        printf("
        \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
        \xB2                                                    LOGIN
        \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
        \xB2");

        outputcursorposition(74, 23);
```

```
printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

outputcursorposition(72, 18);

printf("\t        ENTER THE PASSWORD ");

outputcursorposition(92, 20);


while (input_char != 13)

{

        input_char = getch();


        if (input_char != 13 && input_char != 8)

        {

                printf("*");

                inputpass[counter] = input_char;

                counter++;

        }


        else if (input_char == 8) //backspace ASCII

        {

                if (counter > 0)

                {

                        counter--;
```

```
                inputpass[counter] = '\0';
                printf("\b \b"); // \b is a backspace
            }
        }

        if (input_char == 13) //CR(carriage return) , which is not supposed to be a password character
        {
            break;
        }
}

inputpass[counter] = '\0';

if (strcmp(inputpass, pass) == 0)
{
        outputcursorposition(74, 26);
        printf("REDIRECTING YOU TO THE CATALOUGE.....");
        Sleep(2000);
        display_retailer();
}
else
{
        outputcursorposition(74, 26);
```

```
            printf("\t\tINCORRECT PASSWORD !!");
            outputcursorposition(60, 28);
            printf("\t\t PRESS ANY KEY TO RE-ENTER THE PASSWORD : ");
            getch();
            login_retailer();
        }
}
```

3.2.1.2. Sign-up function of customer

- **Function name:** void entry_customer()
- **Functionality:** This user-defined function is used to enable the customer to enter his details – name and valid phone number which is saved to print them on the invoice later.

```
void entry_customer()
{
    system("CLS");
    char temp_ph[10];
    int counter = 1;

    outputcursorposition(74, 15);
    printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2 SIGN-IN
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2");
```

```
outputcursorposition(74, 24);
printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

outputcursorposition(74, 17);
printf("           Please enter your details");

outputcursorposition(74, 19);
printf("      Enter your name: ");
getchar();
gets(name);

outputcursorposition(74, 21);
printf("      Enter your phone number: ");
gets(phone_number);

strcpy(temp_ph, phone_number);

for (int i = 0; temp_ph[i] != '\0'; i++)
{
      if (temp_ph[i] == '0' || temp_ph[i] == '1' || temp_ph[i] == '2' ||
temp_ph[i] == '3' || temp_ph[i] == '5' || temp_ph[i] == '6' || temp_ph[i] ==
'7' || temp_ph[i] == '8' || temp_ph[i] == '9')
      {
            counter = counter + 1;
      }
}

if (counter == 10 && strlen(temp_ph))
{
```

```
            outputcursorposition(74, 26);
            printf("      Press any key to view the catalogue: ");
            getch();

            customer_menu();
        }

        else
        {
            outputcursorposition(68, 26);
            printf("       INVALID PHONE NUMBER, ENTER YOUR DETAILS
        AGAIN !");

            outputcursorposition(74, 28);
            printf("\t   PRESS ANY KEY TO ENTER AGAIN : ");
            getchar();

            entry_customer();
        }
    }
```

3.2.1.3. Display functions

- **Function name:** void display_main();
- **Functionality:** This user-defined function opens the file with data (product details), reads the data and sets the cursor indentations.

```
void display_main()
{
	system("CLS");

	int i = 26, j = 1;
	FILE *file;

	display_heading();
	file = fopen("record.txt", "rb");
	rewind(file);
	fflush(file);

	while (fread(&item, sizeof(item), 1, file))
	{
		display_file(i, j);
		i++;
		j++;
	}
```

```
        if (i == 26)
        {
            outputcursorposition(65, 20);
            printf("\t\t\tSTOCK IS EMPTY");
        }

        fclose(file);

        outputcursorposition(64, 33);

        printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
        \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
        B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
        2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
        xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

        outputcursorposition(64, 36);
        printf("Press any key to go back to catalogue: ");
        getch();

        display_retailer();
}
```

- **Function name:** void display_heading();
- **Functionality:** This user-defined function is used to display the headings like product name, product ID, product price on the output screen.

```
void display_heading()
{
        int i;

       outputcursorposition(64, 9);

       printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
       \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 ITEM
       DETAILS
       \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
       B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

       outputcursorposition(64, 14);
          printf("   S.No.     Item Name     Item ID        Price       Quantity left");
}
```

- **Function name:** void display_file(int i, int j)
- **Functionality:** This user-defined function retrieves data (product details) from the file and prints them on the screen.

```
void display_file(int i, int j)
{
       outputcursorposition(64, i - 10);
       printf("%4d", j);
       printf("%17s", item.name);
       printf("%12s", item.code);
       printf("%14.2f", item.rate);
       printf("%13d", item.quantity);}
```

3.2.1.4. Retailer Use-case functions

- **Function name:** void display_retailer()
- **Functionality:** This user-defined function displays the option catalogue of the retailer – viewing stock, adding new products, adding stock and searching for products available.

```
void display_retailer()
{
	system("CLS");

	int choice;

	outputcursorposition(64, 11);
	printf("
	\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
	B2\xB2\xB2\xB2\xB2\xB2 OPTION-CATALOGUE
	\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
	B2\xB2\xB2\xB2\xB2\xB2");

	outputcursorposition(64, 14);
	printf("\t\t\t  1. VIEW PRODUCT DETAILS");

	outputcursorposition(64, 16);
	printf("\t\t\t  2. ADD NEW PRODUCTS");

	outputcursorposition(64, 18);
	printf("\t\t\t  3. SEARCH A PRODUCT");

	outputcursorposition(64, 20);
	printf("\t\t\t  4. ADD STOCK ");
```

```
outputcursorposition(64, 22);
printf("\t\t\t  5. GO BACK TO MAIN");

outputcursorposition(64, 24);
printf("\t\t\t  6. EXIT");

outputcursorposition(64, 27);
printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

outputcursorposition(70, 30);
printf("ENTER YOUR CHOICE : ");
scanf("%d", &choice);

switch (choice)
{
      case (1):
      display_main();
      break;

      case (2):
      add();
      break;

      case (3):
      search_main();
      break;

      case (4):
```

```
            add_stock();
            break;

            case (5):
            welcome_options();
            break;

            case (6):
            thankyou_display();
            break;

            default:
            printf("INVALID ENTRY!");
            break;
        }
}
```

- **Function name:** void add_stock()
- **Functionality:** This user-defined function is used to add stock to a pre-existing product. The retailer can access this function. This function displays the available stock and asks for increment and thereby makes changes in the file where data (product details) are stored.

```
void add_stock()
{
    system("CLS");

    FILE *file;
    char id[20];
    int stock;

    outputcursorposition(64, 11);
```

```
printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2 ADDING STOCK
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");

file = fopen("record.txt", "rb+");
rewind(file);

outputcursorposition(64, 14);
printf("      Enter product ID: ");
scanf("%s", id);
fflush(file);

while (fread(&item, sizeof(item), 1, file) == 1)
{
      if ((strcmp(item.code, id) == 0))
      {
            outputcursorposition(64, 17);
            printf("      Stock available is: %d", item.quantity);

            outputcursorposition(64, 20);
            printf("      Would you like to add stock [Y/N]: ");

            if (toupper(getche()) == 'Y')
            {
                  outputcursorposition(64, 22);
                  printf("      Enter the new stock to be added: ");
                  scanf("%d", &stock);

                  int size = sizeof(item.quantity);
                  item.quantity += stock;
```

```
                fseek(file, -size, 1);
                fwrite(&item.quantity, sizeof(item.quantity), 1, file);
                fclose(file);

                outputcursorposition(64, 25);
                printf("      Stock available now is: %d", item.quantity);

                outputcursorposition(64, 28);
                printf("
        \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
        B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
        2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
        xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
        B2\xB2\xB2");
          }

          else
          {
                break;
          }
    }

    else
    {
          outputcursorposition(64, 17);
          printf("      PRODUCT NOT FOUND");
    }
}

outputcursorposition(74, 31);
printf("      Press any key to go back to catalogue: ");
```

```
    getch();
        display_retailer();
}
```

- **Function name:** void add()
- **Functionality:** This user-defined function is used to add new products to the stock. The retailer can access this function. This function asks for the details of product and adds them to the file.

```
void add()
{
    system("CLS");

    FILE *file;
    char y[20], x[12];

    outputcursorposition(64, 11);
    printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2 ADD NEW PRODUCT
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2");

    outputcursorposition(64, 14);
    printf("      Would you like to add a new product [Y/N]: ");

    while (toupper(getche()) == 'Y')
    {
        getchar();
        Sleep(1500);
        getchar();
```

```
        file = fopen("record.txt", "ab");

        system("CLS");

        add_check(y);

        strcpy(item.code, y);

        outputcursorposition(64, 17);
        printf("      Enter name of the item: ");
        scanf("%s", item.name);

        outputcursorposition(64, 19);
        printf("      Enter rate of the item: ");
        scanf("%f", &item.rate);

        outputcursorposition(64, 21);
        printf("      Enter quantity of the item: ");
        scanf("%d", &item.quantity);

        outputcursorposition(64, 24);
        printf("      New product added !!");

        outputcursorposition(64, 27);
        printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

        fseek(file, 0, SEEK_END);
```

```
            fwrite(&item, sizeof(item), 1, file);
            fclose(file);

            outputcursorposition(64, 30);
            printf("      Would you like to add another product [Y/N]: ");
        }

        display_retailer();
}
```

- **Function name**: void add_check(char y[])
- **Functionality**: This user-defined function is used to check if a product ID, when entered by the user, already exists.

```
void add_check(char y[])
{
        int flag;
        FILE *file;
        file = fopen("record.txt", "rb");

        while (1)
        {
            system("CLS");

            flag = 1;

            outputcursorposition(64, 11);
            printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 ADD PRODUCT DETAILS
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
```

```
        rewind(file);

        outputcursorposition(64, 14);
        printf("      Enter ID of the product: ");
        scanf(" %[^\n]", y);

        while (fread(&item, sizeof(item), 1, file) == 1)
        {
            if (strcmp(y, item.code) == 0)
            {
                flag = 0;

                outputcursorposition(64, 16);
                printf("      Code already exists...");

                outputcursorposition(64, 20);
                printf("      Press any key to enter a different ID: ");
                getch();

                break;
            }
        }

        if (flag == 1)
        {
            break;
        }
    }

    fclose(file);
}
```

- **Function name:** void search_main()
- **Functionality:** This user-defined function is used to display the various search options available to the customer.

```
void search_main()
{
	system("CLS");

	char ch;
	int choice;

	outputcursorposition(64, 11);
	printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2 SEARCHING A PRODUCT
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2");

	outputcursorposition(64, 14);
	printf("\t\t\t   1. SEARCH BY ID");

	outputcursorposition(64, 16);
	printf("\t\t\t   2. SEARCH BY PRICE");

	outputcursorposition(64, 18);
	printf("\t\t\t   3. SEARCH BY QUANTITY");

	outputcursorposition(64, 20);
	printf("\t\t\t   4. SEARCH BY NAME");

	outputcursorposition(64, 22);
```

```
    printf("\t\t\t   5. GO BACK TO CATALOGUE");

    outputcursorposition(64, 25);
    printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2");

    outputcursorposition(64, 28);
    printf("      ENTER YOUR CHOICE : ");
    scanf("%d", &choice);

    switch (choice)
    {
    case (1):
        search_id();
        break;

    case (2):
        search_price();
        break;

    case (3):
        search_quantity();
        break;

    case (4):
        search_name();
        break;
```

```
        case (5):
            display_retailer();
            break;

        default:
            break;
        }
}
```

- **Function name:** void search_id()
- **Functionality:** This user-defined function is used to enable the retailer to search a product from the available products by product ID.

```
void search_id()
{
        system("CLS");

        int i = 26, j = 1;
        char x[10] = {0};
        FILE *file;

        outputcursorposition(64, 11);
        printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY ID
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2");

        file = fopen("record.txt", "rb");
        rewind(file);
```

```
outputcursorposition(64, 14);
printf("      Enter item code: ");
scanf("%s", x);

fflush(file);

while (fread(&item, sizeof(item), 1, file))
{
   if ((strcmp(item.code, x) == 0))
   {
      system("CLS");
      display_heading();
      display_file(i, j);
      i++;
      j++;
   }
}

if (i == 26)
{
   outputcursorposition(64, 17);
   printf("      PRODUCT NOT FOUND...");
}

outputcursorposition(74, 23);
printf("      Enter any key to go back  : ");
getch();
fclose(file);
search_main();}
```

- **Function name:** void search_price()
- **Functionality:** This user-defined function is used to enable the retailer to search a product from the available products by product price range.

```
void search_price()
{
	system("CLS");

	int i = 26, j = 1;
	float a, b;
	FILE *file;

	outputcursorposition(64, 11);
	printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY PRICE
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");

	file = fopen("record.txt", "rb");
	rewind(file);

	outputcursorposition(64, 14);
	printf("      Enter lower limit of range of price: ");
	scanf("%f", &a);

	outputcursorposition(64, 16);
	printf("      Enter upper limit of range of price: ");
	scanf("%f", &b);

	fflush(file);
```

```
    system("CLS");

    while (fread(&item, sizeof(item), 1, file))
    {
       if (item.rate >= a && item.rate <= b)
       {
          display_heading();
          display_file(i, j);
          i++;
          j++;
       }
    }

    if (i == 26)
    {
       outputcursorposition(64, 11);
       printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY PRICE
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
       outputcursorposition(64, 15);
       printf("      PRODUCT NOT FOUND");
    }

    outputcursorposition(74, 25);
    printf("      Press any key to go to back  : ");
    getch();

    fclose(file);
    search_main();
}
```

- **Function name:** void search_quantity()
- **Functionality:** This user-defined function is used to enable the retailer to search a product from the available products by product quantity range.

```
void search_quantity()
{
	system("CLS");

	int i = 26, j = 1;
	int a, b;
	FILE *file;

	outputcursorposition(64, 11);
	printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY QUANTITY
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

	file = fopen("record.txt", "rb");
	rewind(file);

	outputcursorposition(64, 14);
	printf("      Enter lower limit of range of quantity: ");
	scanf("%d", &a);

	outputcursorposition(64, 16);
	printf("      Enter upper limit of range of quantity: ");
	scanf("%d", &b);

	fflush(file);
```

```
    system("CLS");

    while (fread(&item, sizeof(item), 1, file))
    {
        if (item.quantity >= a && item.quantity <= b)
        {
            display_heading();
            display_file(i, j);
            i++;
            j++;
        }
    }

    if (i == 26)
    {
        outputcursorposition(64, 11);
        printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY QUANTITY
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
        outputcursorposition(64, 15);
        printf("      PRODUCT NOT FOUND");
    }

    outputcursorposition(74, 25);
    printf("      Press any key to go back  : ");
    getch();

    fclose(file);
    search_main();
}
```

- **Function name:** void search_name()
- **Functionality:** This user-defined function is used to enable the retailer to search a product from the available products by product name.

```
void search_name()
{
        system("CLS");

        int i = 26, j = 1;
        char name[10];
        FILE *file;

        outputcursorposition(64, 11);
        printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2 SEARCHING BY NAME
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2");

        file = fopen("record.txt", "rb");
        rewind(file);

        outputcursorposition(64, 14);
        printf("      Enter item name: ");
        scanf("%s", name);

        fflush(file);

        while (fread(&item, sizeof(item), 1, file))
        {
            if ((strcmp(item.name, name) == 0))
            {
```

```
            system("CLS");
            display_heading();
            display_file(i, j);
            i++;
            j++;
        }
    }

    if (i == 26)
    {
        outputcursorposition(64, 17);
        printf("      PRODUCT NOT FOUND...");
    }

    outputcursorposition(74, 23);
    printf("      Enter any key to go back  : ");
    getch();

    fclose(file);

    search_main();
}
```

3.2.1.5. Customer Use-case functions

- **Function name:** void customer_menu()
- **Functionality:** This user-defined function is used to display the option catalogue of customer – view catalogue, make purchase, receive invoice, exit.

```
void customer_menu()
{
        system("CLS");

        struct id_entry c;
        int input_choice, x;

        outputcursorposition(64, 11);
        printf("
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2 OPTION-CATALOGUE
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2");

        outputcursorposition(64, 14);
        printf("\t\t   1. VIEW THE CATALOGUE OF ITEMS    ");

        outputcursorposition(64, 16);
        printf("\t\t   2. PURCHASE SECTION OF SELECTED ITEMS   ");

        outputcursorposition(64, 18);
        printf("\t\t   3. GET INVOICE  ");

        outputcursorposition(64, 20);
        printf("\t\t   4. GO BACK TO MAIN  ");
```

```
    outputcursorposition(64, 22);
    printf("\t\t   5. EXIT  ");

    outputcursorposition(64, 25);
    printf("
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

    outputcursorposition(64, 28);
    printf("      ENTER YOUR CHOICE : ");
    scanf("%d", &input_choice);

    switch (input_choice)
    {
            case 1:
            item_menu_customer();
            break;

            case 2:
            customer_purchase(no_of_items);
            break;

            case 3:
            display_invoice();
            break;

            case 4:
            welcome_options();
            break;
```

```
                case 5:
                thankyou_display();
                break;

                default:
                printf("INVALID ENTRY!");
                break;
        }
}
```

- **Function name:** int item_menu_customer()
- **Functionality:** This user-defined function is used to enable the customer to choose to purchase or no. It inputs the IDs of products the customer wishes to purchase and also quantity.

```
int item_menu_customer()
{
        system("CLS");

        int in;

        outputcursorposition(60, 5);
        cus_item_display();

        outputcursorposition(60, 32);
        printf("\t\t     WOULD YOU LIKE TO MAKE PURCHASE ? [Y/N] :
      ");

        if (getchar() == 'Y' || getchar() == 'y')
        {
                fname = fopen("item_name.txt", "w+");
```

```
        famt = fopen("item_amounts.txt", "w+");

        fclose(fname);
        fclose(famt);

        outputcursorposition(60, 35);
        printf("\t ENTER THE NUMBER OF ITEMS YOU WANT TO
PURCHASE : ");
        scanf("%d", &no_of_items);

        outputcursorposition(60, 37);
        printf("\t ENTER THE PRODUCT ID(s) : ");

        for (int i = 0; i < no_of_items; i++)
        {
                scanf("%d", &c.id[i]);
        }

        outputcursorposition(60, 39);
        printf("\t PRESS 1 TO PROCEED TO PURCHASE SECTION,
PRESS 2 TO RETURN BACK TO THE CATALOGUE : ");
        scanf("%d", &in);

        if (in == 1)
        {
                customer_purchase(no_of_items);
        }

        else
        {
                customer_menu();
        }
```

```
        }

        else
        {
                outputcursorposition(64, 34);
                printf("      PRESS ANY KEY TO RETURN BACK TO MAIN
    MENU :");
                getch();

                customer_menu();
        }

        return no_of_items;
}
```

- **Function name:** void cus_item_display()
- **Functionality:** This user-defined function is used to display the stock of products available on the screen at the customer end.

```
void cus_item_display()
{
        FILE *fp, *file;
        int i = 26, j = 1;

        fopen("record.txt", "rb");

        display_heading();

        file = fopen("record.txt", "rb");
        rewind(file);
        fflush(file);
```

```
while (fread(&item, sizeof(item), 1, file))
{
        display_file(i, j);
        i++;
        j++;

        if ((j % 20) == 0)
        {
                outputcursorposition(64, 30);
                printf("Press any key to see more....");

                getch();
                system("CLS");

                display_heading();

                i = 26;

                continue;
        }
}

if (i == 26)
{
        outputcursorposition(60, 18);
        printf("\t\t\tSTOCK IS EMPTY");
}

fclose(file);

outputcursorposition(64, 30);
```

```
    printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
    \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
    B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB
    2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
    xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
}
```

- **Function name:** void deduct_stock(int, char[])
- **Functionality:** This user-defined function is used to make changes to the quantity of product after purchase is made by the customer.

```
void deduct_stock(int stock, char temp[10])
{
    fp = fopen("record.txt", "rb+");

    while (fread(&item, sizeof(item), 1, fp) == 1)
    {
        if (strcmp(item.code, temp) == 0)
        {
            int size = sizeof(item.quantity);

            item.quantity -= stock;

            fseek(fp, -size, 1);
            fwrite(&item.quantity, sizeof(item.quantity), 1, fp);
            fclose(fp);
        }
    }
}
```

- **Function name:** int check_quantity(int)
- **Functionality:** This user-defined function is used to check if the quantity the customer wishes to purchase is available for sale or not.

```
int check_quantity(int i)
{
	char code[10], temp[10];

	itoa(c.id[i], code, 10);
	strcpy(temp, code);

	fp = fopen("record.txt", "rb");

	while (fread(&item, sizeof(item), 1, fp))
	{
		if (strcmp(item.code, temp) == 0)
		{
			return item.quantity;
		}
	}

	fclose(fp);
}
```

- **Function name:** void customer_purchase(int)
- **Functionality:** This user-defined function enables the customer to view the amount of stock available for sale and enter the quantity he desires to purchase.

```
void customer_purchase(int num)

{
        char code[10];
        char temp[10];
        system("CLS");
        int input;
        fp1 = fopen("inputquantity.txt", "w+");

        outputcursorposition(55, 9);
        printf("\t\t   ------------------------ PURCHASE ------------------------ ");
        outputcursorposition(55, 11);

        int ypos = 11;
        for (int i = 0; i < num; i++)
        {
                stock:
                ypos = ypos + 2;
                outputcursorposition(55, ypos);
                printf("\tITEM ID: %d  -  QUNATITY AVAILABALE: %d ",
    c.id[i], check_quantity(i));
                printf("\t------>\t ENTER QUANTITY: ");
                scanf("%d", &input);
                itoa(c.id[i], code, 10);
                strcpy(temp, code);

                if (check_quantity(i) < input)
                {
```

```
                outputcursorposition(55, ypos + 2);
                printf(" \t                STOCK INSUFFICIENT!!");
                outputcursorposition(55, ypos + 4);
                printf("\t  WOULD YOU LIKE TO MAKE CHANGES
        IN THE QUANTITY YOU ENTERED? [Y/N] ");

                if (getchar() == 'Y' || getchar() == 'y')
                {
                        ypos = ypos + 4;
                        goto stock;
                }

                else
                {
                        ypos = ypos + 4;
                        printf("\t
_____________________________________________________________
_____________________");
                        continue;
                }
        }

        else if (check_quantity(i) >= input)
        {
                ypos = ypos + 2;
                outputcursorposition(55, ypos);

  printf("_____________________________________________________
_____________________");
                fprintf(fp1, "%d\n", input);
                deduct_stock(input, temp);
                calculate_amt(input, temp);
```

```
            }
        }
        printf("press any key to go back to main");
        getchar();
        customer_menu();
        fclose(fp1);
}
```

- **Function name:** int calculate_amt(int, char[])
- **Functionality:** This user-defined function is used to compute the amount as per the purchases made by the customer.

```
int calculate_amt(int stock, char id[10])
{
        int item_amt = 0;

        fp = fopen("record.txt", "rb+");
        famt = fopen("item_amounts.txt", "a+");
        fname = fopen("item_name.txt", "a+");

        while (fread(&item, sizeof(item), 1, fp) == 1)
        {
                if (strcmp(item.code, id) == 0)
                {
                        item_amt = stock * item.rate;

                        fprintf(famt, "%d\n", item_amt);
                        fprintf(fname, "%s\n", item.name);

                        fclose(fp);
                        fclose(famt);
                        fclose(fname);
```

```
            }
        }

        final_amount += item_amt;
}
```

- **Function name:** void box_invoice()
- **Functionality:** This user-defined function is for the design of the box structure to display the invoice on the screen.

```
void box_invoice()
{
        system("CLS");

        int i;

        outputcursorposition(76, 3);
        printf("---------- INVOICE ----------");

        outputcursorposition(66, 7);
        printf("NAME: %s", name);

        outputcursorposition(105, 7);
        printf(" DATE: %s", __DATE__);

        outputcursorposition(66, 9);
        printf("CONTACT NO.: %s", phone_number);

        outputcursorposition(105, 9);
        printf(" TIME: %s", __TIME__);

        outputcursorposition(65, 11);
```

```
for (int i = 0; i < 60; i++)
{
	printf("\xcd");
}

outputcursorposition(65, 12);
for (i = 0; i < 19; i++)
{
	outputcursorposition(65, i + 12);
	printf("\xba");
}

outputcursorposition(65, 31);
for (int i = 0; i < 60; i++)
{
	printf("\xcd");
}

outputcursorposition(125, 12);
for (i = 0; i < 19; i++)
{
	outputcursorposition(125, i + 12);
	printf("\xba");
}

outputcursorposition(65, 11);
printf("\xc9");

outputcursorposition(125, 11);
printf("\xbb");

outputcursorposition(65, 31);
```

```
        printf("\xc8");

        outputcursorposition(125, 31);
        printf("\xbc");

        outputcursorposition(66, 13);
        for (int i = 0; i < 59; i++)
        {
                printf("\xcd");
        }

        outputcursorposition(66, 12);
        printf(" S.NO.\t\tPRODUCT NAME\t\t  TOTAL PRICE");
}
```

- **Function name:** void display_invoice();
- **Functionality:** This user-defined function is used to retrieve the product details from file and print them on the screen as per the purchases made by the customer.

```
void display_invoice()
{
        char names[20][20], id[20][20], name[20];
        int amount[20], i = 0, j = 0, k = 0;
        int amt, ypos = 15, ypos1 = 15;
        float total;

        fname = fopen("item_name.txt", "r");
        famt = fopen("item_amounts.txt", "r");

        while (fgets(name, sizeof(name), fname) != NULL)
        {
```

```
        strcpy(names[i], name);
        i++;
}

while (fscanf(famt, "%d\n", &amt) != EOF)
{
        amount[k] = amt;
        k++;
 }

 fclose(fname);
 fclose(famt);

 box_invoice();

 for (int q = 0; q < i; q++)
 {
        outputcursorposition(66, ypos);
        printf("%4d %24s", q + 1, names[q]);

         ypos = ypos + 2;
  }

  for (int q = 0; q < i; q++)
  {
        outputcursorposition(105, ypos1);
        printf("%13d", amount[q]);

        ypos1 = ypos1 + 2;
   }

   outputcursorposition(66, 27);
```

```
        {
                for (i = 0; i < 59; i++)
                {
                        printf("-");
                }
        }

        total = final_amount;

        outputcursorposition(76, 29);
        printf("        TOTAL AMOUNT : %.2f", total);

        outputcursorposition(70, 33);
        printf("------------ THANK YOU FOR PURCHASING ------------");

        outputcursorposition(72, 37);
        printf("Press any key to go back to Option-Catalogue : ");
        getch();

        customer_menu();
}
```

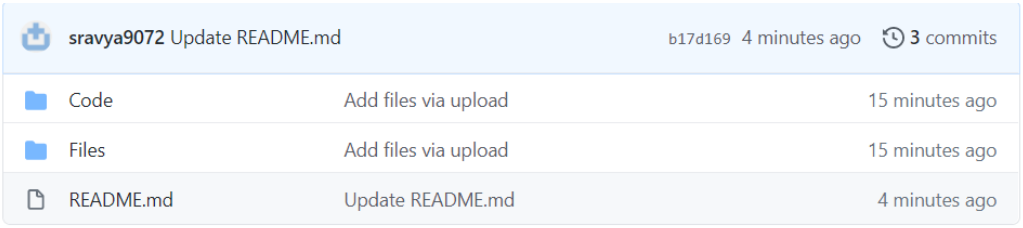
3.2.2. GitHub structure/folder

We have segregated the files of our project into folders namely - Code and Files. Code folder contains the main C file and the Files folder contains the data files (.txt format) which we used in our project. Our repository also contains a README file which has a brief description of our project. We have also uploaded report of our project on our github repository.

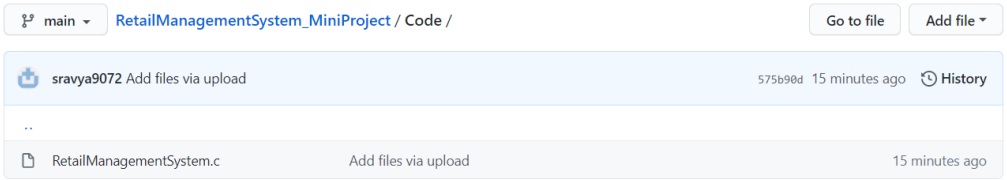
GitHub Repository Link:

https://github.com/sravya9072/RetailManagementSystem_MiniProject

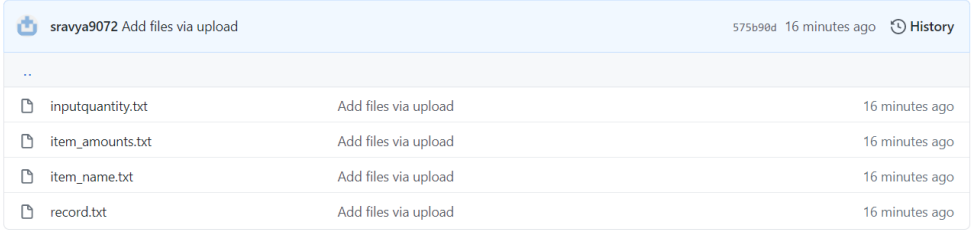
https://github.com/Nikki8502/RetailManagementSystem-Miniproject



- Code folder contents:



- Files folder contents:



3.3. Testing

3.3.1. Retailer Test Cases

- **FOR LOGIN**

Test case ID: TC01		Use case ID: **UC01**
Test case title: Login (Retailer)		
Test case description: User has to enter password		
Test steps	**Expected result**	**Actual result**
The system prompts the user to enter password. User enters the password Password entered is not same as the hard-coded one.	An error message saying "INCORRECT PASSWORD, ENTER PASSWORD AGAIN" is displayed on the screen.	An error message saying "INCORRECT PASSWORD, ENTER PASSWORD AGAIN" is displayed on the screen.

Test case ID: TC02		Use case ID: **UC01**
Test case title: Login (Retailer)		
Test case description: User has to enter password		
Test steps	**Expected result**	**Actual result**
The system prompts the user to enter password. User enters the password Password entered is same as the hard-coded one.	A message saying "REDIRECTING TO OPTION CATALOGUE" is displayed on the screen and system displays option catalogue.	A message saying "REDIRECTING TO OPTION CATALOGUE" is displayed on the screen and system displays option catalogue.

- **VIEW PRODUCTS**

Test case ID: TC03		Use case ID: **UC02**
Test case title: View products (Retailer)		
Test case description: User can view the products available for sale		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to view products.	Initially, if there are no products added to the stock, a message saying "STOCK IS EMPTY" is displayed on the screen.	Initially, if there are no products added to the stock, a message saying "STOCK IS EMPTY" is displayed screen.

Test case ID: TC04		Use case ID: **UC02**
Test case title: View products (Retailer)		
Test case description: User can view the products available for sale		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to view products.	The list of all products available for sale along with their details are displayed on the screen.	The list of all products available for sale along with their details are displayed on the screen.

- **ADDING PRODUCTS**

Test case ID: TC05		Use case ID: **UC03**
Test case title: Add new products (Retailer)		
Test case description: User can add new products to the stock.		
Test steps	**Expected result**	**Actual result**
System displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to add new products. System prompts the user to enter new product ID. Users enters an ID which isn't a pre-existing one, and then system prompts to enter other product details.	Product is successfully added to the stock i.e., added to the file.	Product is successfully added to the stock i.e., added to the file.

Test case ID: TC06		Use case ID: **UC03**
Test case title: Add new products (Retailer)		
Test case description: User can add new products to the stock.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to add new products. System prompts the user to enter new product ID. System checks if the ID. User enters a pre-existing ID.	An error message saying "CODE EXISTS. ENTER ANOTHER ONE" is displayed on the screen and system prompts user to enter another code.	An error message saying "CODE EXISTS. ENTER ANOTHER ONE" is displayed on the screen and system prompts user to enter another code.

- **SEARCH BY ID**

Test case ID: TC07		Use case ID: **UC04**
Test case title: Search products by ID (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by ID. The system prompts the user to enter a product ID. User enters a product ID which is an existing one.	The details of the product which matches the ID given by the user is displayed on the screen.	The details of the product which matches the ID given by the user is displayed on the screen.

Test case ID: TC08		Use case ID: **UC04**
Test case title: Search products by ID (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by ID. The system prompts the user to enter a product ID. User enters a product ID which is not an existing one or an invalid one.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.

- **SEARCH BY PRICE**

Test case ID: TC09		Use case ID: **UC05**
Test case title: Search products by price range (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by price. The system prompts the user to enter a price range. User enters valid lower and upper limit of the price range.	The details of all the product(s) which fall under the range given by the user is/are displayed on the screen.	The details of all the product(s) which fall under the range given by the user is/are displayed on the screen.

Test case ID: TC10		Use case ID: **UC05**
Test case title: Search products by price range (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by price. The system prompts the user to enter a price range. User enters valid lower and upper limit of the price range.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.

- **SEARCH BY QUANTITY**

Test case ID: TC11		Use case ID: **UC06**
Test case title: Search products by quantity range (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by quantity. The system prompts the user to enter a quantity range. User enters valid lower and upper limit of the quantity range.	The details of all the product(s) which fall under the range given by the user is/are displayed on the screen.	The details of all the product(s) which fall under the range given by the user is/are displayed on the screen.

Test case ID: TC12		Use case ID: **UC06**
Test case title: Search products by price quantity (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by quantity. The system prompts the user to enter a quantity range. User enters valid lower and upper limit of the quantity range.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.

- **SEARCH BY NAME**

Test case ID: TC13		Use case ID: **UC07**
Test case title: Search products by name (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by name. The system prompts the user to enter a product name. User enters a product name which is an existing one.	The details of the product which matches the name given by the user is displayed on the screen.	The details of the product which matches the name given by the user is displayed on the screen.

Test case ID: TC14		Use case ID: **UC07**
Test case title: Search products by name (Retailer)		
Test case description: User can search for a particular product if available.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to search products by name. The system prompts the user to enter a product name. The system prompts the user to enter a product name. User enters a product name which is not an existing one or an invalid one.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.	An error message saying "PRODUCT NOT AVAILABLE" is displayed on the screen.

- **ADD STOCK**

Test case ID: TC15		Use case ID: **UC08**
Test case title: Add stock to pre-existing products (Retailer)		
Test case description: User can add quantity to the pre-existing products.		
Test steps	**Expected result**	**Actual result**
System displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to add new products. System prompts the user to enter new product ID. Users enters and ID which is a pre-existing one.	System displays available stock and asks user to enter the quantity he wishes to add. User enters the quantity to be added and stock is updated successfully.	System displays available stock and asks user to enter the quantity he wishes to add. User enters the quantity to be added and stock is updated successfully.

Test case ID: TC16		Use case ID: **UC08**
Test case title: Add new products (Retailer)		
Test case description: User can add new products to the stock.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of retailer and prompts the user to make his choice. User selects the option to add new products. System prompts the user to enter new product ID. System checks if the ID. User enters an ID which isn't a pre-existing one.	An error message saying "CODE EXISTS… ENTER ANOTHER ONE" is displayed on the screen and system prompts user to enter another code.	An error message saying "CODE EXISTS… ENTER ANOTHER ONE" is displayed on the screen and system prompts user to enter another code.

3.3.2. Customer Test Cases

- **SIGN-UP**

Test case ID: TC17		Use case ID: **UC10**
Test case title: Sign-up (Customer)		
Test case description: User has to enter his details.		
Test steps	**Expected result**	**Actual result**
The system prompts the user to enter name and phone number. User enters the name and an invalid phone number (having some characters instead of 10 digits).	An error message saying "INCORRECT PASSWORD. ENTER PASSWORD AGAIN" is displayed on the screen.	An error message saying "INCORRECT PASSWORD. ENTER PASSWORD AGAIN" is displayed on the screen.

Test case ID: TC18		Use case ID: **UC10**
Test case title: Sign-up (Customer)		
Test case description: User has to enter his details.		
Test steps	**Expected result**	**Actual result**
The system prompts the user to enter name and phone number. User enters the name and a valid phone number (only 10 digits).	User is directed to option catalogue and is prompted to make his choice.	User is directed to option catalogue and is prompted to make his choice.

- **VIEW CATALOGUE**

Test case ID: TC19		Use case ID: **UC11**
Test case title: View catalogue (Customer)		
Test case description: User can view the products available for sale.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of customer and prompts the user to make his choice. User selects the option to view catalogue of items.	The list of all products available for sale along with their details are displayed on the screen and system prompts the user to select if he wishes to make any purchase. If user chooses yes, he is directed to purchase section.	The list of all products available for sale along with their details are displayed on the screen and system prompts the user to select if he wishes to make any purchase. If user chooses yes, he is directed to purchase section.

Test case ID: TC20		Use case ID: **UC11**
Test case title: View catalogue (Customer)		
Test case description: User can view the products available for sale.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of customer and prompts the user to make his choice. User selects the option to view catalogue of items.	The list of all products available for sale along with their details are displayed on the screen and system prompts the user to select if he wishes to make any purchase. If user chooses no, he is directed back to the option catalogue.	The list of all products available for sale along with their details are displayed on the screen and system prompts the user to select if he wishes to make any purchase. If user chooses no, he is directed back to the option catalogue.

- **MAKE PURCHASE**

Test case ID: TC21		Use case ID: **UC12**
Test case title: Make purchase (Customer)		
Test case description: User can purchase products of his choice.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of customer and prompts the user to make his choice. User selects the option to make purchase. System prompts the user to select if he wishes to make any purchase. The user chooses yes and he is directed to purchase section. User has to enter IDs of products he wishes to buy. Quantity available for each of selected products is displayed and user enters insufficient quantity.	An error message saying insufficient quantity, make changes in quantity is displayed on screen.	An error message saying insufficient quantity, make changes in quantity is displayed on screen.

Test case ID: TC22		Use case ID: **UC12**
Test case title: Make purchase (Customer)		
Test case description: User can purchase products of his choice.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of customer and prompts the user to make his choice. User selects the option to make purchase. System prompts the user to select if he wishes to make any purchase. The user chooses yes and he is directed to purchase section. User has to enter IDs of products he wishes to buy. Quantity available for each of selected products is displayed and user enters sufficient quantity.	The quantity entered is noted and deducted from stock and added to invoice.	The quantity entered is noted and deducted from stock and added to invoice.

- **INVOICE**

Test case ID: TC23		Use case ID: **UC13**
Test case title: View invoice (Customer)		
Test case description: User can view the invoice.		
Test steps	**Expected result**	**Actual result**
The system displays the option catalogue of customer and prompts the user to make his choice. User selects the option to view the invoice.	An invoice with customer details, and purchases made is displayed on the screen.	An invoice with customer details, and purchases made is displayed on the screen.

4. Results (Output Screenshots)

4.1. Welcome Page

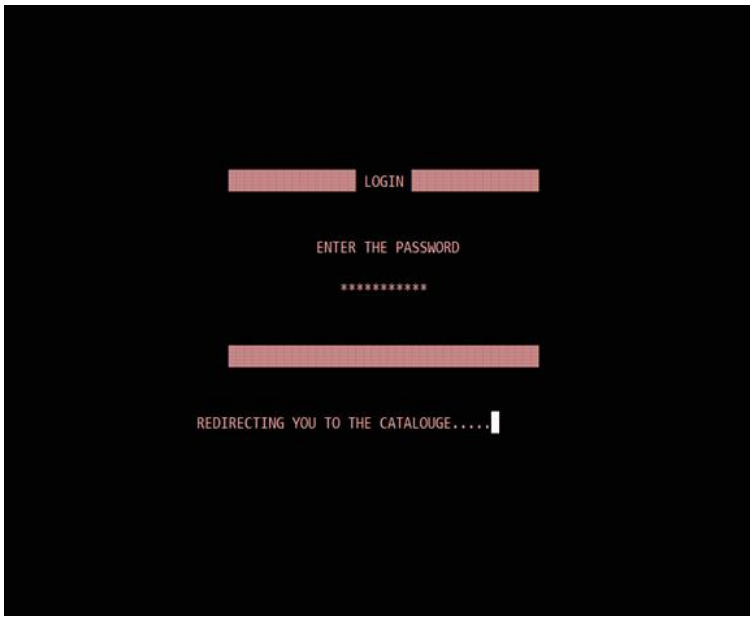
4.2. Main-Menu

4.3. Retailer Test-Cases

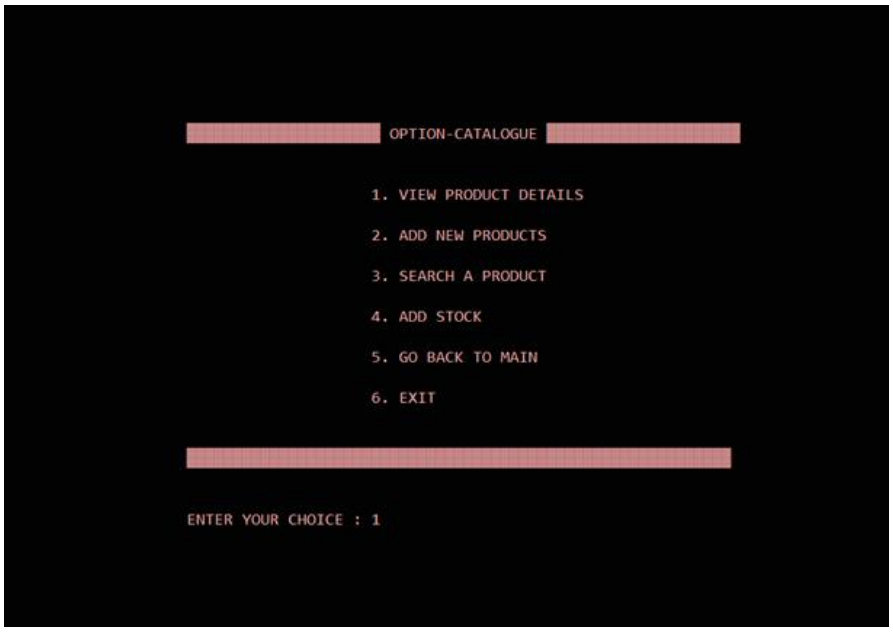
1 – Login

- Wrong password entered

- Correct password entered



#2 – View the main option-catalogue (Retailer)



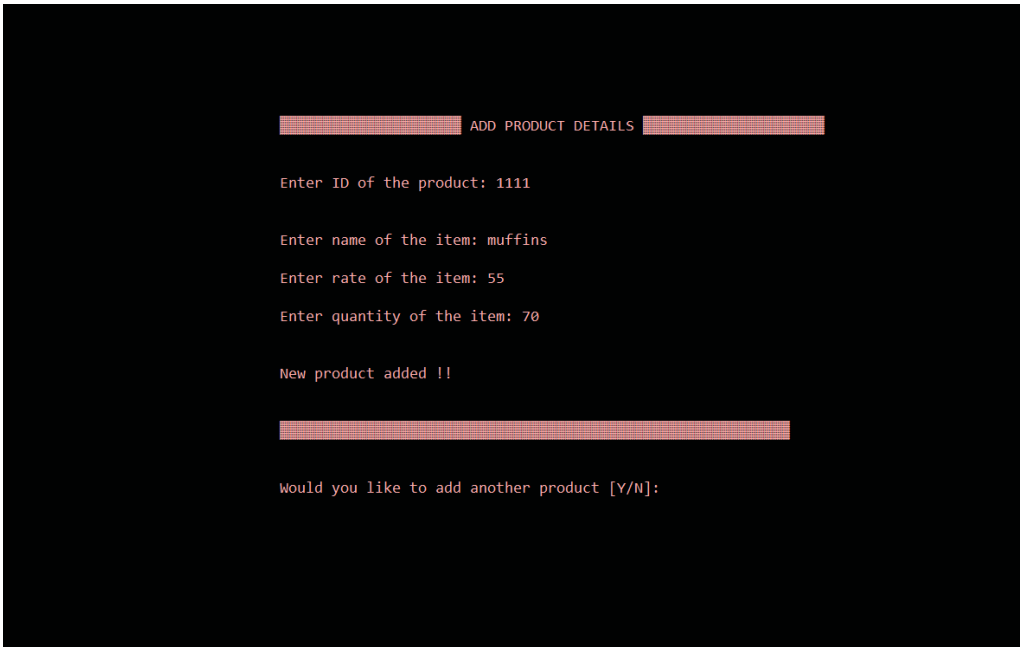
#3 – View the List of items

- Stock is empty

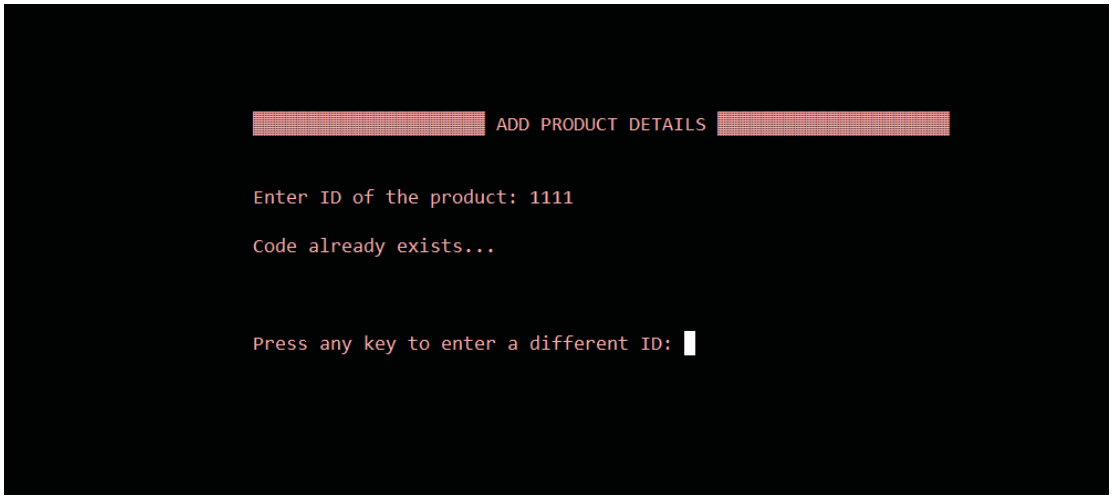
- Stock is non-empty

#4 – Adding products

- Product ID entered is present in the pre-existing stock.



- Product ID entered is not present in the pre-existing stock.



#5 – Search by ID

- Entered product ID not present (Item not found)

- Entered product ID present (Item found)

#5 – Search by Item price

- Item not found in the give price range

- Item found in the give price range

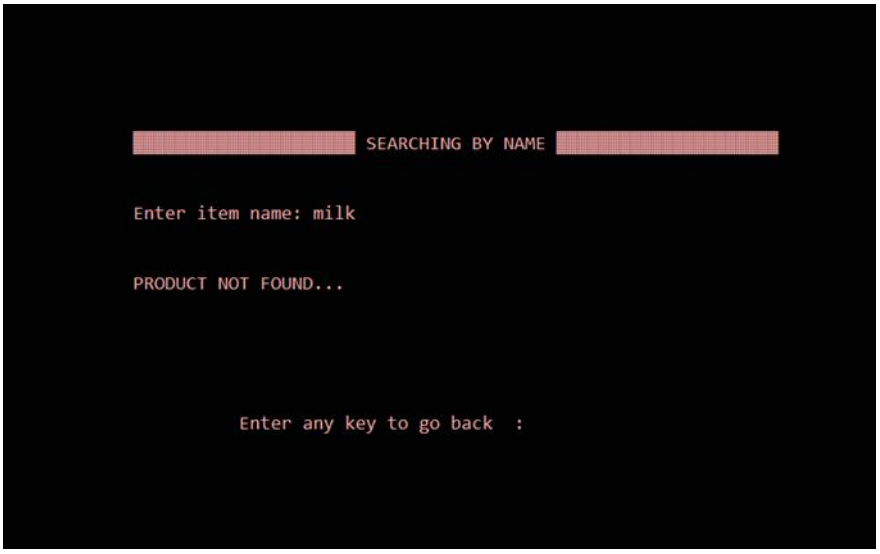
#6 – Search by Item quantity

- Item not found in the give quantity range

- Item found in the give quantity range

#7 – Search by Item Name

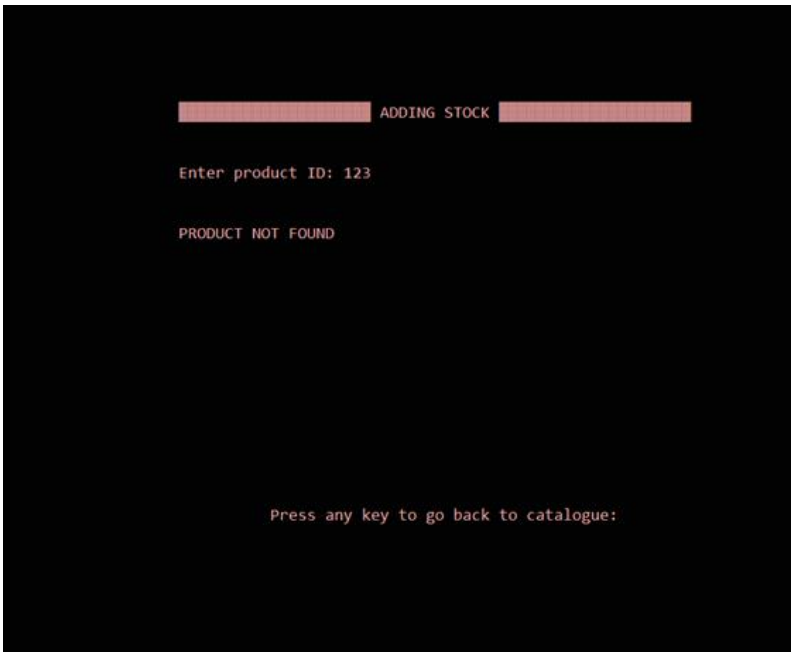
- Item not found with the given name



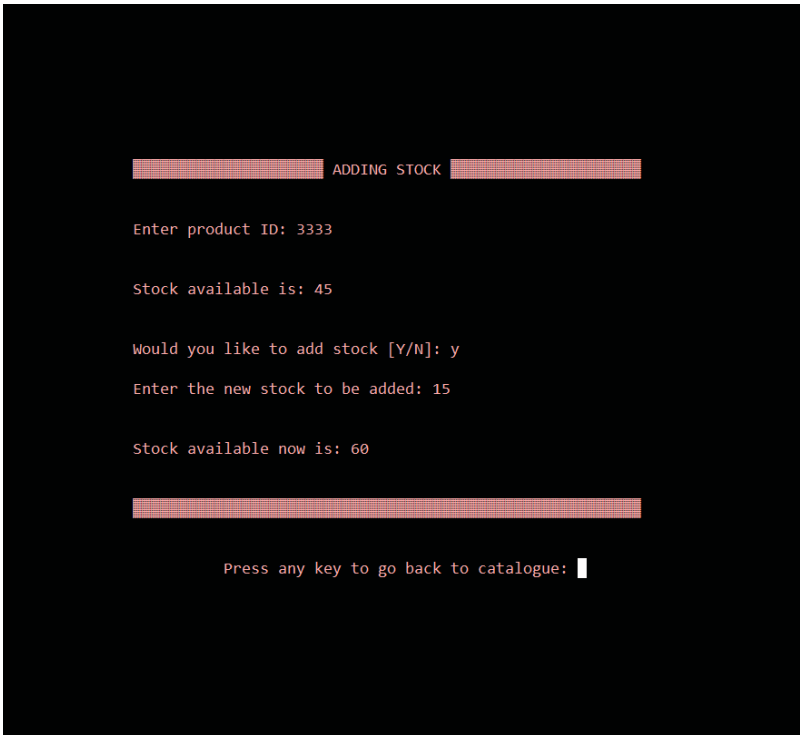
- Item found with the given name

#7 – Adding stock to pre-existing products

- ID entered does not exist



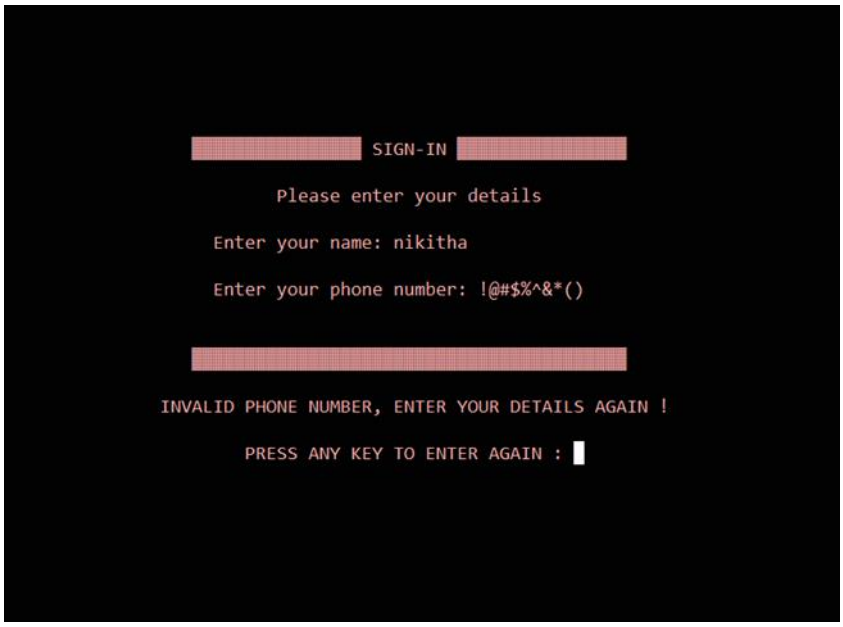
- ID entered exists, Customer adds stock to the given ID

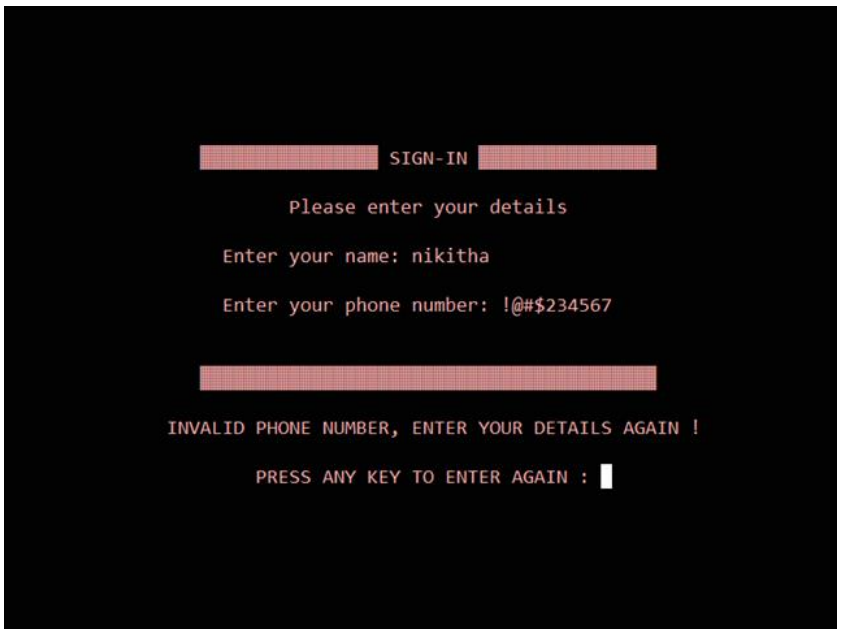


4.4. Customer Test-Cases

#1- Sign-Up

- Wrong format of phone number entered

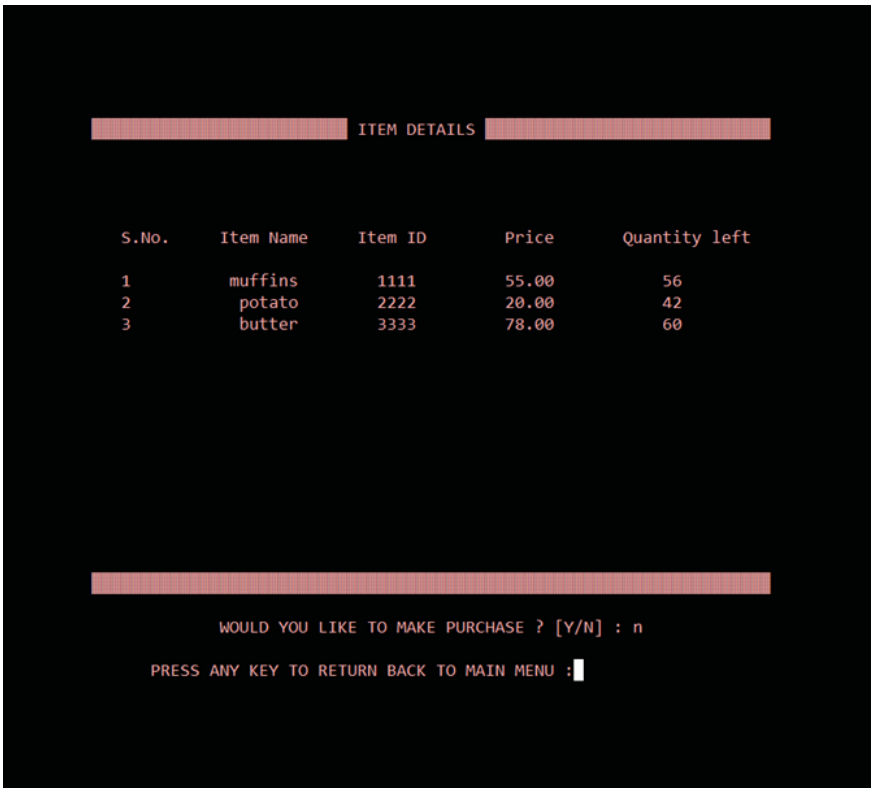




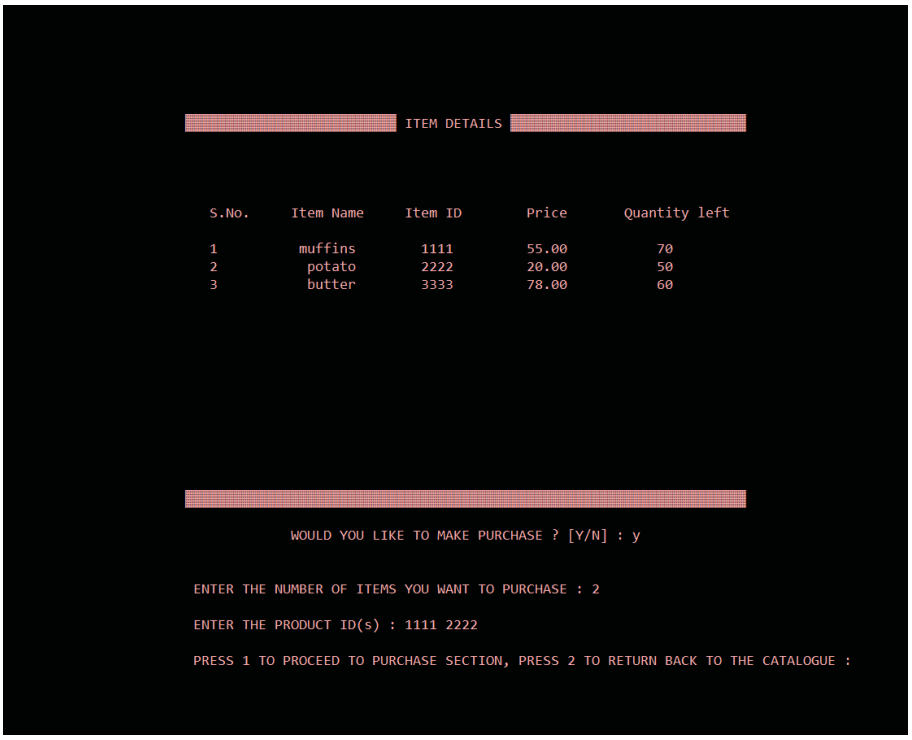
- Correct format of phone number entered

#2 – View the main option-catalogue (Customer)

#3 – Catalogue of products in the store



#4 – Purchase – Selection of Item ID's



#5 – Purchase – Required stock entry

- Insufficient stock entry

- Sufficient stock entry

#6 – Get Invoice

4.5. Thank-you page

5. ADDITIONAL KNOWLEDGE GAINED

This mini project in C programming helped us get clear with the fundamentals of C language and also helped us brush-up the main concepts. We have also learned a few new concepts. Earlier, when we had to print something as an output on the screen, it was the conventional way of printing at default positions (like to the left). But now, we have learned to handle the cursor positions on the output screen and print the desired output in a organised and more structured way.

This project also taught us that writing the code alone does not make the job done, but also writing a smaller and more efficient code is what matters in today's competitive world. When you write a code, you need to be clear enough as to what each and every statement or function in that program does. It is equally important to structure and organise your code such that even a common man will be able to understand.

6. CONCLUSION AND FUTURE WORK

This project “Retail Management System” in C programming, has helped us to once again get back with all the topics we have already learnt in C programming in the previous semester. We have also learned some new topics. In fact, as a part of the curriculum, we only get to write code for some simple problems, but in this mini project we have chosen a real-life problem and tried to build an application with the available known resources. This project helped us to write huge codes, patiently deal with errors and how to solve a problem step-wise analyse it, think of possible solutions and choosing the best and accurate way.

We would like to further extend our project in the future by creating a real-time graphical user interface and also add an additional feature of sending invoice through the mail by learning Client-Server networking concepts. We would also like to include payment in our features. We also would like to make a real-time app using Flutter or React and also a web application using Django and Python.

7. REFERENCES

WEBSITE: www.cprogramming.com

LINK: https://cboard.cprogramming.com/c-programming/42482-setting-cursor-position-c.html

USED FOR: Output cursor position function

WEBSITE: www.stackoverflow.com

LINK: https://stackoverflow.com/questions/39025074/c-program-how-to-print-in-table-format-alignment

USED FOR: Table alignment and Line patterns

WEBSITE: www.programmingsimplified.com

LINK: https://www.programmingsimplified.com/c/dos.h/sleep

USED FOR: sleep function

WEBSITE: www.youtube.com

LINK: https://www.youtube.com/watch?v=4E59FvBzoZY&t=2246s

USED FOR: Hidden characters for password