ELECTRONIC STORE INVENTORY 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

SYED MUNEEBULLAH HUSSAINI – 1602-19-737-178

PRASUN REDDY – 1602-19-737-147

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2020

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, SYED MUNEEBULLAH HUSSAINI and PRASUN REDDY bearing hall ticket numbers, 1602-19-737-178 and 1602-19-737-147, hereby declare that the project report entitled "ELECTRONIC STORE INVENTORY 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.

SYED MUNEEBULLAH HUSSAINI

1602-19-737-178

PRASUN REDDY

1602-19-737-147

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

ACKNOWLEDGEMENTS

Our Mini Project would not have been successful without the help of several people. We are extremely thankful to our college, Vasavi College of Engineering, for providing the opportunity to implement our project, “Electronic Store Inventory Management System”.

We would like to express our gratitude to Ms. Divya Lingineni, Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Ms. DRL Prasanna, Assistant Professor, Department of Information Technology, Vasavi College of Engineering and Dr. Ramesh Vasappanavara, Professor, Department of Information Technology, Vasavi College of Engineering, for their esteemed guidance, moral support and invaluable advice provided by them for the success of this Mini Project.

Sincerely,

SYED MUNEEBULLAH HUSSAINI 1602-19-737-178

PRASUN REDDY 1602-19-737-147

ABSTRACT

Big Electronic Companies like Samsung, Apple etc. have large databases and multiple systems to manage their databases and inventory. This project is designed to implement a similar inventory management system for a small-scale electronics store. Store Employees will be able to use this project to keep track of inventory, add or remove products, control prices and manage the store, essentially. Employees will also be able to use this system to communicate better with and help their customers. They will be able to compare different features of different products as well.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 About The Project

"Electronic Store Inventory Management System" is a console-based C Project intended to be used by electronic stores to manage store inventory, shipments, purchases and act as an information source for various products offered by the store.

1.2 Problem Domain

Most electronic stores are equipped with software facilities connected to their inventory which allow them to control purchases, shipments and inventory. However, such software is usually only available for store employees to use and cannot be availed by customers. They also have very limited functionality and are not usually up to the standards required by commercial businesses.

"Electronic Store Inventory Management System" intends to limit these drawbacks and make it easier to manage and control inventory, shipments and purchases store while also increasing customer-employee cohesion by providing information related to various products offered by the store. This project can be used by both employees and customers and has a flexible and vast list of features which make it easier for both the store employee and the customer.

1.3 Project Features

1. **Employee and Customer Authorization –** The project features an authorization system for both employees and customers. Pre-registered employees can login and access the employee interface. Employees cannot utilize employee-specific features without successfully logging in. Customers can either register for a new account or login with existing accounts. Customers can access the customer interface without logging in however they gain access to a few extra features if they choose to continue as a registered customer.
2. **Manage Shipments –** Employees can manage wholesale product shipments to the store via this project. The project will display the amount (corresponding to wholesale prices) needed to complete these shipments and will also log each

shipment separately. Employees can also view these mentioned previous shipments at any time along with the date, time and cost for each shipment.

3. **Product Purchases –** Customers can access the 'Buy Products' interface to select which products they want to purchase. After selection of each product, a list of features for that product will be displayed along with its price. The customer can then choose whether to confirm the product's selection or not and can then continue to select other products. A final list of products chosen along with total price will be displayed once the customer has completed selection. The selection and payment can then be confirmed by an employee (who needs to log in) to complete the purchase.
4. **Check Cash Balance –** Store employees have the option to view the current cash balance of the store at any time, provided they are logged in.
5. **Check Stocks –** Store employees can check the current stock of any product they select.
6. **Product Features –** Both customers and employees can view features for a single product or compare different features of different products of the same type. The project will display a concise list of features for each product alongside each other if the compare products option is selected.
7. **Previous Purchases –** Registered customers can view a list of all of their previous purchases made using that account only with date, time and price.
8. **Redeem Coupons –** Registered customers can redeem coupon codes to get a discount on their purchase if the system is able to verify the code.

2. TECHNOLOGY

2.1 Software Requirements

The minimum software requirements to utilize the “Electronic Store Inventory Management System” are:

- Operating System: Windows XP or higher
- GNU Compiler Collection (GCC)
- Text Editor

2.2 Hardware Requirements

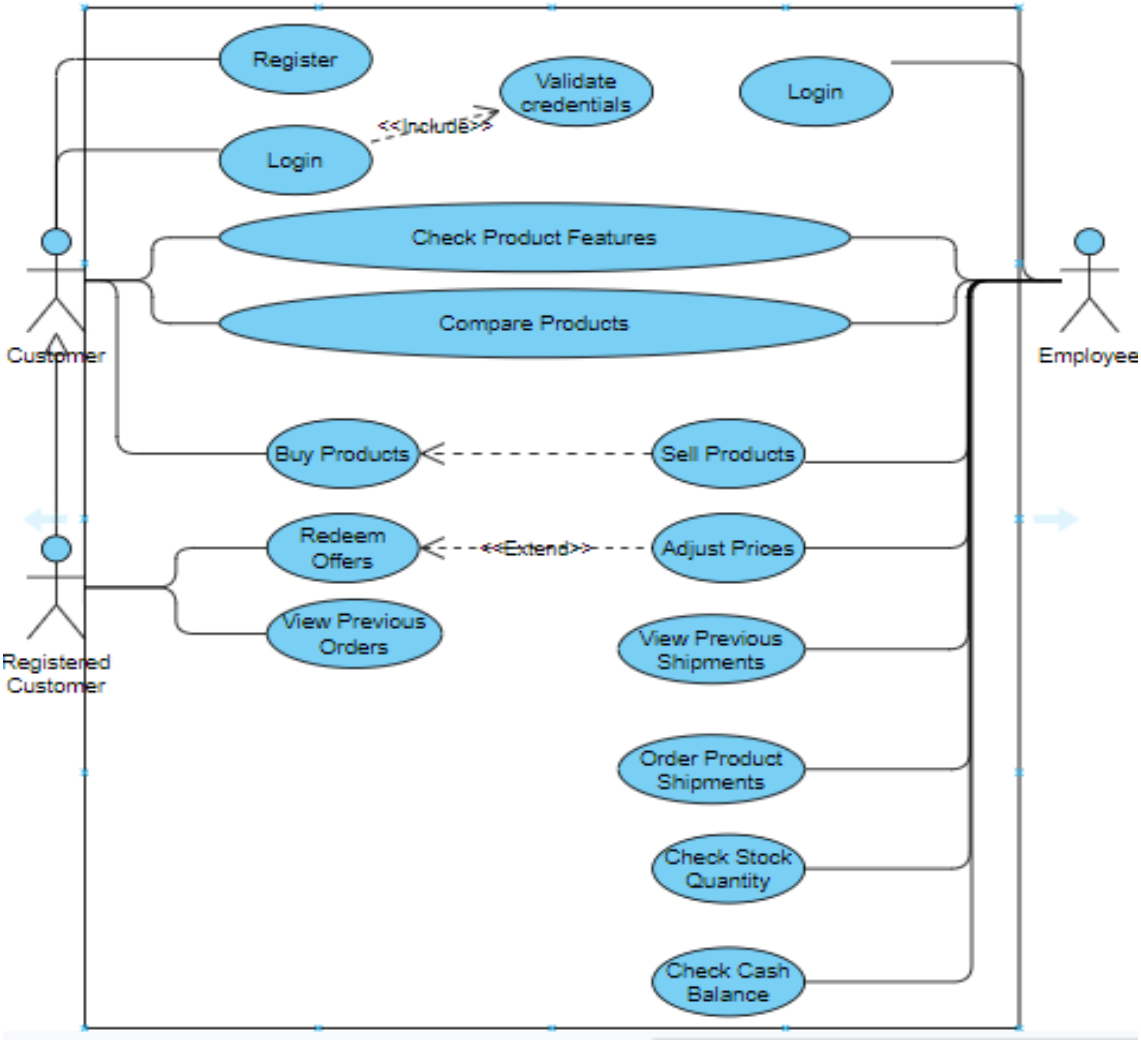
The minimum hardware requirements to utilize the “Electronic Store Inventory Management System” are:

- Minimum 4MB RAM
- Minimum 25MB Free Storage
- Processor: Intel 386 and its equivalent or higher

3. PROPOSED WORK

3.1 Design

3.1.1 Use Case Diagram



3.1.2. Actors

3.1.1.1 Employee

The Employee actor is the store employee and has access to manage product shipments, selling products and product features. Employees need to be logged in to be recognized as employees.

3.1.1.2 Customer

The Customer actor is a customer visiting the store to make a purchase or to survey the products offered by the store. A customer has access to buying

products and product features. Customers can continue with or without logging in. The Customer actor also extends the ‘Registered Customer’ actor.

3.1.2.3 Registered Customer

The Registered Customer actor is a customer who has successfully logged in. Registered Customers have access to more features than regular customers.

Actors Wise Use Cases

3.1.3. Employee Use Cases

3.1.1.3 UC01 Employee Login

Employees need to successfully login to be recognized as employees and access the Employee interface. The Employee is first prompted by the System to enter credentials – Employee ID. The System then checks if the Employee ID is valid and exists. If invalid, the Employee is directed back to the previous menu. If it is valid, the Employee is prompted to enter the password. The System validates the input password by matching it with the existing password corresponding to the previously entered Employee ID. If password validation is successful, login is successful and the Employee continues to the Employee interface. If password validation is unsuccessful, the Employee is directed back to the previous menu along with a prompt.

User:	System:
Enters credentials.	
	Validates credentials from among store employees.
Prompted if invalid and returned to previous menu.	

3.1.1.4 UC02 Check Stocks

Logged In Employees can check the current stock of any selected product. The Employee first has to select the ‘Check Stocks’ option from the Employee interface and is then directed to the ‘Check Stocks’ section. The system displays the product menu and the user is prompted to select a product. The system then displays the current quantity of the product selected and also displays a warning if quantity is low.

User:	System:
Selects check stocks option.	
	Displays product menu. Prompts user to select.
Selects products group or product.	
	Displays stock of selected product.
Can continue to view stock for other products.	

3.1.1.5 UC03 Order Shipments

Logged In Employees can order new shipments for products at wholesale price. The Employee first has to select the ‘Order Shipments’ option from the Employee interface and is then directed to the ‘Order Shipments’ section. The Employee has to enter the number of products to be ordered. The system

displays the product menu for the Employee to select products and quantity for each. If sufficient cash balance is available, the shipment is completed and stocks and cash balance are updated.

User:	System:
Selects order shipments option.	
	Displays wholesale price of products. Prompts user to select product(s) and quantities.
Selects product(s) and quantity for each.	
	Stocks and cash balance are updated.

3.1.1.6 UC04 View Previous Shipments

Logged In Employees can view previous shipments of products. The Employee first has to select the 'View Previous Shipments' option from the Employee interface and is then directed to the 'View Previous Shipments' section. The system then displays the details of previous shipments along with cost, date and time for each shipment.

User:	System:
Selects View Previous Shipments option.	
	Displays details of previous shipments.

3.1.1.7 UC05 Sell Products

Employees can sell products provided a customer has initiated a purchase. The Employee is to confirm the purchase initiation and then forward the payment request. The Employee then has to confirm payment completion to complete the purchase. Stocks and cash balance are then updated and the Employee is warned if stock is low for any of the bought products.

User:	System:
	Displays customer request to purchase product.
Confirms purchase and forwards payment request.	
	Prompts customer to complete payment.
Completes product sale.	
	Updates stock.

3.1.1.8 UC06 Check Cash Balance

Logged In Employees can check the current cash balance. The Employee first has to select the 'Check Cash Balance' option from the Employee interface and is then directed to the 'Check Cash Balance' section. The system then displays the current cash balance.

User:	System:
Selects Check Balance option.	
	Cash Balance is displayed.

3.1.4. Customer Use Cases

3.1.4.1. UC07 Customer Register

Customers can register for a new account. The Customer first has to select the 'Create New Account' option from the Customer interface and is then directed to the 'Create New Account' section. The Customer is prompted to enter an Account name. If the Account name is not taken, the Customer is prompted to enter and confirm password. If password is confirmed, the system creates a new account for the Customer

User:	System:
Enters customer details.	
	Registers customer.

3.1.4.2. UC08 Customer Login

Customers can login to their account if they already have an existing one. The Customer first has to select the ‘Log In’ option from the Customer interface and is then directed to the ‘Log In’ section. The Customer is first prompted by the System to enter credentials – Account Name. The System then checks if the Account Name exists. If invalid, the Customer is directed back to the previous menu. If it is valid, the Customer is prompted to enter the password. The System validates the input password by matching it with the existing password corresponding to the previously entered Account name. If password validation is successful, login is successful and the Customer can continue as a Logged In Customer. If password validation is unsuccessful, the Customer is directed back to the previous menu along with a prompt.

User:	System:
Enters credentials.	
	Validates credentials from among registered customers.
Prompted if invalid and returned to previous menu.	

3.1.4.3. **UC09 Buy Products**

Customers can purchase products. The Customer first has to select the ‘Buy Products’ option from the Customer interface and is then directed to the ‘Buy Products’ section. The System then displays the product menu and the Customer is prompted to make a selection. The System displays a list of features and price for the product. The Customer can choose whether to confirm selection of the current product or not. The Customer can then choose whether to continue product selection or not. The Customer then has to enter quantity for each product selected. Once product selection is completed, the System displays the total price of the products selected. The Customer is once again prompted to confirm the purchase initiation. The Employee is then prompted to confirm the purchase initiation. The Customer is prompted to complete payment and then the purchase is completed. Stocks and Cash Balance are then updated.

User:	System:
	Displays product menu.
Selects product(s) to purchase.	
	Displays price.
Selects whether to initiate purchase or not.	
	Prompts customer for payment after employee confirmation
Completes payment.	

3.1.5. Registered Customer Use Cases

3.1.5.1. UC10 View Previous Purchases

Registered Customers can view previous purchases they have made with that account. The Customer first has to select the 'View Previous Purchases' option from the Logged In Customer interface and is then directed to the 'View Previous Purchases' section. The System then displays a list of previous purchases made with that account along with price, date and time.

User:	System:
Selects View Previous Purchases option.	
	Displays details of previous purchases.

3.1.5.2. UC11 Redeem Coupons

Registered Customers can redeem coupon codes, if they have any, for discounts in product purchases. The Customer, if Logged In, is asked if they have a coupon code before completing a purchase. If they have a coupon code, the Logged In Customer is then prompted to enter the code. The system validates the code and if successful, a discount is applied to the total price of the product purchase.

User:	System:
Enters coupon code	
	Checks coupon validity. Reduces product price if coupon is valid.

3.1.6. Common Use Cases

These Use Cases are common to both Employee and Customer actors.

3.1.6.1. UC12 View Product Features

Both Employees and Customers have the option to view product features for a single product. The User first has to select the 'View Product Features' from the corresponding interface. The System then displays the product menu and the User is prompted to select a product. The System then displays a list of features for the product selected.

User:	System:
Selects View Product Features option.	
	Displays product menu. Prompts user to select product.
Selects product.	
	Displays product features.

3.1.6.2. UC13 Compare Product Features

Both Employees and Customers have the option to compare product features for several products. The User first has to select the 'Compare Product Features' from the corresponding interface. The User is then prompted to select the number of products to compare. The System then displays the product menu and the User is prompted to select products. The System then displays the features of the products selected alongside each other for comparison.

User:	System:
Selects Compare Product Features option.	
	Displays product menu. Prompts user to select product.
Selects products.	
	Displays product comparison.

3.2. Implementation

3.2.1 Functions

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
#include<time.h>

int cash_balance = 0; //External variable to store Cash Balance which can be accessed throughout the program

void clear(); //Clears screen when called

void balance(); //Updates cash_balance variable and displays Cash Balance when called

void Employee(); //Displays Employee Interface if Login is successful
int EmployeeLogin(); //Verifies Employee Identity

void CheckStocks(); //Enables Employee to Check Stock Quantity
void OrderShipments(); //Enables Employee to Order New Shipments
void ViewShipments(); //Enables Employee to View Previous Shipments

void ViewFeatures(); //Enables User to View Features for a single product

void CompareProducts(); //Enables User to Compare Features of several products

//CompareProducts() function branches into 2 functions to handle mobile Feature Comparison and Laptop Feature Comparison
//CompareProducts() also reads the number of products to be compared and passes them to CompareMobiles() and CompareLaptops()
void CompareMobiles(int n);
void CompareLaptops(int n);

void Customer(); //Displays Customer Interface
void CustomerRegistry(); //Enables Customers to create new Customer Accounts
```

```c
void CustomerLogin(); //Verifies Customer Identity if Customer Account already exists

void LoggedInCustomer(char Account[]); //Displays Logged In Customer Interface, Account name is passed to this function
void RegularCustomer(); //Displays Regular Customer Interface

void BuyProducts(int flag, char Account[]); //Enables Customer to Buy Products
//flag variable is used to toggle between Logged In and Registered Customer, Account name is passed if Customer is Logged In

void PreviousPurchases(char Account[35]); //Enables Logged In Customer to View Previous Purchases made using that Customer Account
int RedeemCoupons(char coupon[]); //Called by BuyProducts() Function if Customer has a Coupon Code and verifies the code

void set_quantity(char id[], int quantity); //Changes Quantity of a product depending on New Purchase/Shipment
int get_quantity(char id[]); //Returns Quantity of selected product

void features(char id[]); //Displays features of product selected when Customer is selecting products to purchase

void product_name(char id[], char name[]); //Assigns Product Name to name[] String
int product_price(char id[]); //Returns Product Price

char* set_product(); //Used across the program to enable User to select product
//set_product() function branches into mobile() and laptop() functions which display Menus to select Mobile/Laptop brand

char* mobiles();
char* laptops();
//mobile() and laptop() functions branch into several functions corresponding to each brand

//Each brand function finally enables User to select product and then returns its ID as a string
char* oneplus_mobiles();
char* xiaomi_mobiles();
char* samsung_mobiles();
```

```c
char* apple_mobiles();

char* hp_laptops();
char* dell_laptops();
char* asus_laptops();
char* lenovo_laptops();
```

3.2.2 Main Module

```c

int main()
{
    int c;
    char s[100];

    //Cash Balance is assigned to the cash_balance variable at the start of the program
    FILE *Balance;
    Balance = fopen("balance.txt", "r");
    fscanf(Balance, "%s", s);
    cash_balance = atoi(s);
    fclose(Balance);

    do //Displays Initial Menu to continue as Employee or Customer or to Exit the Program
    {
        printf("\n\n\t\t\t\tINVENTORY MANAGEMENT SYSTEM\n");
        printf("\n\t\tPRESS:");
        printf("\n\t\t1 TO CONTINUE AS EMPLOYEE");
        printf("\n\t\t2 TO CONTINUE AS CUSTOMER");
        printf("\n\t\t3 TO EXIT");
        printf("\n\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        system("cls");
        switch(c)
        {
            case 1: Employee();
                    clear();
                    break;
            case 2: Customer();
                    clear();
```

```c
                    break;
            case 3: printf("\n\t\tEXITING THE SYSTEM...\n");
                    break;
            default:printf("\n\t\tINVALID INPUT...\n");
                    break;
        }
    }while(c != 3);
    return 0;
}

void balance() //Updates cash_balance variable and displays Cash Balance when called
{
    FILE *Balance;
    printf("\n\t\tCURRENT CASH BALANCE: %d", cash_balance);
    Balance = fopen("balance.txt", "w"); //Cash Balance is stored in balance.txt
    fprintf(Balance, "%d", cash_balance);
    fclose(Balance);
    clear();
}

```

3.2.3 Employee Module

```c
void Employee() //Displays Employee Interface when called from Main Menu of main() function, if login is successful
{
    int c;
    c = EmployeeLogin(); //Verifies Employee Identity by calling EmployeeLogin() function
    system("cls");
    if(c == 0) //Program returns to main() function if login fails
    {
        printf("\n\n\t\tLOG IN FAILED DUE TO INCORRECT ID OR PASSWORD\n");
        return;
    }
    printf("\n\t\tLOGGED IN AS EMPLOYEE\n"); //Employee Interface is displayed if Login is successful
    do
    {
        printf("\n\n\t\t\t\tINVENTORY MANAGEMENT SYSTEM\n");
        printf("\n\t\t\t\t\tEMPLOYEE MENU");
        printf("\n\t\tPRESS:");
        printf("\n\t\t1 TO CHECK STOCKS");\
        printf("\n\t\t2 TO CHECK CASH BALANCE");
        printf("\n\t\t3 TO ORDER PRODUCT SHIPMENTS");
        printf("\n\t\t4 TO VIEW PREVIOUS SHIPMENTS");
        printf("\n\t\t5 TO VIEW PRODUCT FEATURES");
        printf("\n\t\t6 TO COMPARE PRODUCT FEATURES");
        printf("\n\t\t7 to EXIT TO MAIN MENU");
        printf("\n\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        system("cls");
        switch(c)
        {
            case 1: CheckStocks();
                    clear();
                    break;
            case 2: balance();
                    break;
            case 3: OrderShipments();
                    clear();
                    break;
            case 4: ViewShipments();
                    clear();
                    break;
            case 5: ViewFeatures();
                    clear();
                    break;
            case 6: CompareProducts();
                    clear();
                    break;
            case 7: printf("\n\t\tEXITING TO MAIN MENU...\n");
                    break;
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(c != 7);
}

int EmployeeLogin() //Verifies Employee Identity
{
    char EmployeeIDs[10][10] = {"101", "102", "103", "104", "105", "106", "107", "108"}; //Valid Employee identities are stored in 2D String
    char Account[10], File[10], Password[31], CorrectPassword[31], *s = ".txt", c;
    int i;
    FILE *EmployeeAccount;
    printf("\n\t\tENTER EMPLOYEE ID: "); //Employee ID is taken as input
    scanf("%s", Account);
```

```
    strcpy(File, Account);
    strcat(File, s);
    EmployeeAccount = fopen(File, "r");
    for(i = 0; i < 8; i++)
    {
        if(strcmp(EmployeeIDs[i], Account) == 0)
        {
            printf("\n\t\tENTER PASSWORD: "); //Password is taken as input
            scanf("%s", Password);
            //Passowrd is verified
            fscanf(EmployeeAccount, "%s", CorrectPassword);
            fclose(EmployeeAccount);
            if(strcmp(CorrectPassword, Password) == 0)
            {
                return 1; //Function returns 1 if Password is correct
            }
            else
            {
                return 0; //Function returns 0 if Password is wrong
            }
        }
    }
    return 0; //Function returns 0 if ID entered is not found
}

void CheckStocks() //Enables Employee to Check Stock Quantity
{
    char id[10], name[100];
    printf("\n\n\t\t\t\t\tCHECK STOCKS");
    strcpy(id, set_product());
```

```
    product_name(id, name);
    printf("\n\t\tPRODUCT: %s", name);
    printf("\n\t\tQUANTITY: %d", get_quantity(id)); //Quantity is Displayed using get_quantity() function
}

void OrderShipments() //Enables Employee to Check Stock Quantity
{
    //Time variables
    time_t rawtime;
    struct tm *timeinfo;

    char *s, *a = "";
    int i, n, cost = 0;
    FILE *fp;
    printf("\n\n\t\t\t\t\tORDER SHIPMENTS");
    printf("\n\n\t\tENTER NUMBER OF DIFFERENT PRODUCTS: "); //Number of Products is taken as input
    scanf("%d", &n);
    char Products[n][10];
    int NewQuantity[n];
    int OldQuantity[n];
    int Prices[n];
    for(i = 0; i < n; i++)
    {
        printf("\n\n\t\tSELECT PRODUCT %d\n", i + 1);
        strcpy(Products[i], set_product()); //Product is Selected
        OldQuantity[i] = get_quantity(Products[i]); //Current Quantity of product is stored in array
        Prices[i] = product_price(Products[i]) * 0.8; //Product Price is stored in array
        printf("\n\n\t\tENTER PRODUCT QUANTITY: ");
        scanf("%d", &NewQuantity[i]); //New Quantity is stored
    }
```

```

    //Summary of Products selected is displayed
    printf("\n\n\t\tPRODUCT ID\t\t\tQUANTITY\tPRICE");
    for(i = 0; i < n; i++)
    {
        printf("\n\t\t%s  \t\t\t\t%d\t\t%d", Products[i], NewQuantity[i], Prices[i]);
        cost = cost + (NewQuantity[i] * Prices[i]);
    }

    //Total Cost is Displayed
    printf("\n\n\t\tTOTAL COST: %d", cost);

    if(cost > cash_balance) //Shipment is cancelled if it cannot be afforded
    {
        printf("\n\n\t\tINSUFFICIENT BALANCE AVAILABLE");
        return;
    }

    //Shipment is confirmed
    printf("\n\n\t\tENTER 1 TO CONFIRM SHIPMENT: ");
    scanf("%d", &i);
    if(i != 1)
    {
        printf("\n\n\t\tCONFIRMATION FAILED...");
        return;
    }

    //Cash Balance is updated
    cash_balance = cash_balance - cost;
    balance();
```

```
267
268     printf("\n\n\t\tSHIPMENT CONFIRMED");
269
270     //Shipment Details are stored in file along with date and time
271     fp = fopen("PreviousShipments.txt", "a");
272     time(&rawtime);
273     timeinfo = localtime(&rawtime);
274     fprintf(fp, "\n\n\t\t%s\n", asctime(timeinfo));
275     fprintf(fp, "\n\t\tPRODUCT ID\t\t\tQUANTITY\tPRICE%s", a);
276     for(i = 0; i < n; i++)
277     {
278         set_quantity(Products[i], OldQuantity[i] + NewQuantity[i]);
279         fprintf(fp, "\n\t\t\t%s  \t\t\t\t%d\t\t\t%d", Products[i], NewQuantity[i], Prices[i]);
280     }
281     fclose(fp);
282 }
283
284 void ViewShipments() //Enables Employee to View Previous Shipments
285 {
286     FILE *fp;
287     char c;
288     printf("\n\n\t\t\t\t\tSHIPMENT HISTORY");
289     fp = fopen("PreviousShipments.txt", "r"); //Previous Shipment details are stored in PreviousShipments.txt
290     c = getc(fp);
291     while(c != EOF)
292     {
293         putch(c);
294         c = getc(fp);
295     }
296     fclose(fp);
```

```
297 }
298
```

3.2.4 Customer Module

```
299 void Customer() //Displays Customer Interface when called from Main Menu of main() function
300 {
301     int c;
302     do
303     {
304         printf("\n\n\t\t\t\tINVENTORY MANAGEMENT SYSTEM\n");
305         printf("\n\t\t\t\t\tCUSTOMER MENU");
306         printf("\n\t\tPRESS:");
307         printf("\n\t\t1 TO CREATE A NEW ACCOUNT");
308         printf("\n\t\t2 TO LOG IN USING AN EXISTING ACCOUNT");
309         printf("\n\t\t3 TO CONTINUE WITHOUT LOGGING IN");
310         printf("\n\t\t4 TO RETURN TO MAIN MENU");
311         printf("\n\n\t\tENTER YOUR CHOICE: ");
312         scanf("%d", &c);
313         system("cls");
314         switch(c)
315         {
316             case 1: CustomerRegistry();
317                     clear();
318                     break;
319             case 2: CustomerLogin();
320                     clear();
321                     break;
322             case 3: RegularCustomer();
323                     clear();
324                     break;
325             case 4: printf("\n\t\tEXITING TO MAIN MENU...\n");
326                     break;
327             default:printf("\n\t\tINVALID INPUT...\n");
328         }
```

```
329     }while(c!=4);
330 }
331
332 void CustomerRegistry() //Enables Customers to create new Customer Accounts
333 {
334     char Account[35], Purchases[37], Password[35], Password1[35], Password2[35], *s = ".txt";
335     FILE *CustomerAccount;
336     printf("\n\t\tENTER ACCOUNT NAME (Max 30 Characters): "); //New Account Name is taken as input
337     scanf("%s", Account);
338     strcpy(Purchases, Account);
339     strcat(Account, ".txt");
340     strcat(Purchases, "PH.txt");
341     CustomerAccount = fopen(Account, "r");
342     if(CustomerAccount != NULL)
343     {
344         printf("\n\t\tACCOUNT NAME TAKEN\n");
345         fclose(CustomerAccount);
346         return;
347     }
348     printf("\n\t\tENTER PASSWORD (Max 30 Characters): "); //Password is taken as input
349     scanf("%s", Password1);
350     printf("\n\t\tCONFIRM PASSWORD: "); //Password is confirmed
351     scanf("%s", Password2);
352     if(strcmp(Password1, Password2) == 0) //New Account is created if Password is confirmed
353     {
354         strcpy(Password, Password1);
355         CustomerAccount = fopen(Account, "w");
356         fprintf(CustomerAccount, "%s", Password);
357         printf("\n\t\tNEW CUSTOMER ACCOUNT SUCCESSFULLY CREATED\n");
358         fclose(CustomerAccount);
```

```
        CustomerAccount = fopen(Purchases, "w"); //New Purchase History File is created for Customer
        fclose(CustomerAccount);
    }
    else
    {
        printf("\n\t\tPASSWORD CONFIRMATION FAILED\n");
    }
}

void CustomerLogin() //Verifies Customer Identity if Customer Account already exists
{
    char Account[35], File[40], Password[35], CorrectPassword[35], c;
    int i;
    FILE *CustomerAccount;
    printf("\n\t\tENTER CUSTOMER ACCOUNT NAME: "); //Account Name is taken as input
    scanf("%s", Account);
    strcpy(File, Account);
    strcat(File, ".txt");
    CustomerAccount = fopen(File, "r");
    if(CustomerAccount == NULL) //If Account is not found
    {
        printf("\n\t\tACCOUNT NAME DOES NOT EXIST\n");
        return;
    }
    printf("\n\t\tENTER PASSWORD: "); //Password is taken as input
    scanf("%s", Password);
    fscanf(CustomerAccount, "%s", CorrectPassword);
    fclose(CustomerAccount);
    if(strcmp(CorrectPassword, Password) == 0) //Password is verified
    {
        printf("\n\t\tLOGGED IN AS REGISTERED CUSTOMER\n");
        clear();
        LoggedInCustomer(Account); //Program moves to Logged In Customer Interface if Login is successful
        return;
    }
    printf("\n\t\tINCCORECT PASSWORD");
    return; //Program returns to Customer Interface if Login fails
}

void LoggedInCustomer(char Account[35]) //Displays Logged In Customer Interface, Account name is passed to this function
{
    int c;
    do
    {
        printf("\n\n\t\t\t\tINVENTORY MANAGEMENT SYSTEM\n");
        printf("\n\t\t\t\tCUSTOMER MENU");
        printf("\n\t\tPRESS:");
        printf("\n\t\t1 TO BUY PRODUCTS");
        printf("\n\t\t2 TO VIEW PREVIOUS PURCHASES");
        printf("\n\t\t3 TO VIEW PRODUCT FEATURES");
        printf("\n\t\t4 TO COMPARE PRODUCT FEATURES");
        printf("\n\t\t5 TO EXIT");
        printf("\n\n\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        system("cls");
        switch(c)
        {
            case 1: BuyProducts(1, Account);
                    clear();
                    break;
            case 2: PreviousPurchases(Account);
                    clear();
                    break;
            case 3: ViewFeatures();
                    clear();
                    break;
            case 4: CompareProducts();
                    clear();
                    break;
            case 5: printf("\n\t\tEXITING TO MAIN MENU...\n");
                    break;
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(c!=5);
}

void RegularCustomer() //Displays Regular Customer Interface
{
    int c;
    do
    {
        printf("\n\n\t\t\t\tINVENTORY MANAGEMENT SYSTEM\n");
        printf("\n\t\t\t\tCUSTOMER MENU");
        printf("\n\t\tPRESS:");
        printf("\n\t\t1 TO BUY PRODUCTS");
        printf("\n\t\t2 TO VIEW PRODUCT FEATURES");
        printf("\n\t\t3 TO COMPARE PRODUCT FEATURES");
        printf("\n\t\t4 TO EXIT");
        printf("\n\n\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
```

```
        switch(c)
        {
            case 1: BuyProducts(0, " ");
                    clear();
                    break;
            case 2: ViewFeatures();
                    clear();
                    break;
            case 3: CompareProducts();
                    clear();
                    break;
            case 4: printf("\n\t\tEXITING TO MAIN MENU...\n");
                    break;
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(c!=4);
}

void BuyProducts(int flag, char Account[35]) //Enables Customer to Buy Products
//flag variable is used to toggle between Logged In and Registered Customer, Account name is passed if Customer is Logged In
{
    //Time variables
    time_t rawtime;
    struct tm *timeinfo;

    char *s, *a = "", name[100], File[40], coupon[10];
    int i, x, price, cost = 0, c = 1, n = 0;
    FILE *fp;
    printf("\n\n\t\t\t\t\tBUY PRODUCTS");
    char Products[10][10];
```

```
    int Quantity[10];
    int OldQuantity[10];
    int Prices[10];
    do
    {
        printf("\n\n\t\tSELECT PRODUCT %d\n", n + 1);
        strcpy(Products[n], set_product()); //Product is Selected

        //Product Name, Price and Features are Displayed
        product_name(Products[n], name);
        price = product_price(Products[n]);
        printf("\n\n\t\tPRODUCT: %s", name);
        printf("\n\n\t\tPRICE: %d", price);
        printf("\n\t\tFEATURES:\n");
        features(Products[n]);

        //Customer can confirm selection of product
        printf("\n\t\tENTER 1 TO SELECT THIS PRODUCT TO BUY: ");
        scanf("%d", &c);
        if(c == 1)
        {
            printf("\n\n\t\tPRODUCT: %s SELECTED", name);
            n++;
        }
        else
        {
            printf("\n\n\t\tPRODUCT NOT SELECTED");
        }

        //Customer can choose to continue selection
```

```
        printf("\n\t\tENTER 1 TO CONTINUE SELECTING OTHER PRODUCTS: ");
        scanf("%d", &c);
    }while(c == 1);
    printf("\n\n\t\tPRODUCT SELECTION HAS STOPPED");
    if(n == 0)
    {
        return;
    }
    i = 0;
    while(i < n)
    {
        OldQuantity[i] = get_quantity(Products[i]); //Current Quantity of product is stored in array
        Prices[i] = product_price(Products[i]); //Product Price is stored in array
        product_name(Products[i], name);
        printf("\n\n\t\tPRODUCT: %s", name);
        printf("\n\t\tENTER PRODUCT QUANTITY: "); //Product Quantity is taken as input
        scanf("%d", &Quantity[i]);
        if(Quantity[i] > OldQuantity[i]) //Program checks if sufficient stocks are available for purchase
        {
            printf("\n\n\t\t\t\tINSUFFICIENT STOCKS AVAILABLE");
            printf("\n\n\t\t\t\tPLEASE ENTER QUANTITY EQUAL TO OR LESSER THAN %d", OldQuantity[i]);
        }
        else //Iterator is incremented only if sufficient stocks are available
        {
            i++;
        }
    }

    //Summary of Products Selected is displayed
    printf("\n\n\t\tPRODUCT ID\t\t\tQUANTITY\tPRICE");
```

```
    for(i = 0; i < n; i++)
    {
        printf("\n\t\t%s  \t\t\t\t%d\t\t%d", Products[i], Quantity[i], Prices[i]);
        cost = cost + (Quantity[i] * Prices[i]);
    }

    //Customer is asked for Coupon Code if they are Logged In
    if(flag == 1)
    {
        printf("\n\n\t\tENTER 1 IF YOU HAVE A COUPON: ");
        scanf("%d", &c);
        if(c == 1)
        {
            printf("\n\t\tENTER COUPON CODE: "); //Coupon Code is taken as input
            scanf("%s", coupon);
            if(RedeemCoupons(coupon)) //Coupon Code is verified
            {
                printf("\n\t\t10%% DISCOUNT APPLIED");
                cost = cost * 0.9; //Discount is applied if Coupon Code is valid
            }
        }
    }

    //Total Cost is Displayed
    printf("\n\n\t\tTOTAL COST: %d", cost);

    //Purchase Initation is confirmed
    printf("\n\n\t\tENTER 1 TO INITIATE PURCHASE: ");
    scanf("%d", &i);
    if(i != 1)
    {
        printf("\n\n\t\tINITIATION FAILED...");
        return;
    }

    //Employee Confirmation is taken after logging in
    printf("\n\n\t\tEMPLOYEE CONFIRMATION REQUIRED");
    printf("\n\n\t\tEMPLOYEE LOG IN");
    int t = 0;
    do
    {
        if(EmployeeLogin())
        {
            clear();
            t = 1;
            printf("\n\n\t\tEMPLOYEE ENTER 1 TO CONFIRM PURCHASE INITATION: ");
            scanf("%d", &c);
        }
        else
        {
            clear();
            printf("\n\n\t\tEMPLOYEE LOG IN FAILED");
            printf("\n\t\tPLEASE TRY AGAIN");
        }
    }while(t == 0);

    //Customer is prompted to complete payment if Employee confirmation is successful
    if(c == 1)
    {
        printf("\n\n\t\tCUSTOMER IS TO COMPLETE PAYMENT");
        //Payment Confirmation is done
        printf("\n\n\t\tENTER 1 TO CONFIRM PAYMENT HAS BEEN COMPLETED: ");
        scanf("%d", &c);

        //Final Purchase Summary is displayed
        if(c == 1)
        {
            printf("\n\n\t\tPRODUCT ID\t\t\tQUANTITY\tPRICE");
            for(i = 0; i < n; i++)
            {
                printf("\n\t\t%s  \t\t\t\t%d\t\t%d", Products[i], Quantity[i], Prices[i]);
            }
            printf("\n\n\t\tTOTAL COST: %d", cost);
            cash_balance = cash_balance + cost;
            balance();
            printf("\n\n\t\tPURCHASE COMPLETED");

            //All products purchased are checked to see if stocks are low
            for(i = 0; i < n; i++)
            {
                set_quantity(Products[i], OldQuantity[i] - Quantity[i]);
                x = get_quantity(Products[i]);
            }

            //Purchase Summary is stored in Purchase History file of Customer if logged in
            if(flag == 1)
            {
                strcpy(File, Account);
                strcat(File, "PH.txt");
                fp = fopen(File, "a");
```

```
                    time(&rawtime);
                    timeinfo = localtime(&rawtime);
                    fprintf(fp, "\n\n\t\t%s\n", asctime(timeinfo));
                    fprintf(fp, "\n\t\tPRODUCT ID\t\t\tQUANTITY\tPRICE%s", a);
                    for(i = 0; i < n; i++)
                    {
                        fprintf(fp, "\n\t\t%s  \t\t\t\t%d\t\t\t%d", Products[i], Quantity[i], Prices[i]);
                    }
                    fclose(fp);
                }
            }
            else
            {
                printf("\n\n\t\tPURCHASE NOT COMPLETED");
            }
        }
        else
        {
            printf("\n\n\t\tPURCHASE NOT COMPLETED");
        }
    }

    void PreviousPurchases(char Account[35]) //Enables Logged In Customer to View Previous Purchases made using that Customer Account
    {
        FILE *fp;
        char File[40], c;
        strcpy(File, Account);
        strcat(File, "PH.txt");
        printf("\n\n\t\t\t\tPURCHASE HISTORY OF ACCOUNT: %s\n", Account);
        fp = fopen(File, "r"); //Purchase History of Customer is stored in File with PH as suffix to Customer Account name
        c = getc(fp);
        while(c != EOF)
        {
            putch(c);
            c = getc(fp);
        }
        fclose(fp);
    }

    int RedeemCoupons(char coupon[10]) //Called by BuyProducts() Function if Customer has a Coupon Code and verifies the code
    {
        char Coupons[5][10] = {"XY101", "ES323", "TY003", "MH178", "PR149"}; //Valid Coupon Codes are stored in 2D String
        int i;
        for(i = 0; i < 5; i++)
        {
            if(strcmp(Coupons[i], coupon))
            {
                return 1; //Function returns 1 if Coupon Code is valid
            }
        }
        return 0; //Function returns 0 if Coupon Code is invalid
    }
```

3.2.5 Product Features Module

```
void ViewFeatures() //Enables User to View Features for a single product
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT PRODUCT");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT MOBILE");
        printf("\n\t\t2 TO SELECT LAPTOP");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: CompareMobiles(2);
                    return;
            case 2: CompareLaptops(2);
                    return;
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

void CompareProducts() //Enables User to Compare Features for several products
{
    int c, n;
    do{
        printf("\n\t\t\t\tSELECT PRODUCT");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT MOBILE");
        printf("\n\t\t2 TO SELECT LAPTOP");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);

        //Number of products is read and passed to according mobiles/laptops function
        printf("\n\n\t\tENTER NUMBER OF PRODUCTS TO COMPARE: ");
        scanf("%d", &n);
        n = n + 1;
        switch(c)
        {
            case 1: CompareMobiles(n);
                    return;
            case 2: CompareLaptops(n);
                    return;
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

void CompareMobiles(int n) //Enables User to Compare Features for several mobiles
{
    int i, j;
    char c, mob[10], file[15], s[100];
    FILE *Mobiles[n];
    Mobiles[0] = fopen("mobile.txt", "r");

    //File Pointer array is used to point towards features file of each mobile selected
    for(i = 1; i < n; i++)
    {
        printf("\n\n\t\t\t\tSELECT MOBILE %d\n", i);
        strcpy(mob, mobiles());
        strcat(mob, "F.txt");
        Mobiles[i] = fopen(mob, "r");
    }

    //Features for each mobile are displayed side by side by printing each file 1 line at a time
    printf("\n\t\t\t\tMOBILES FEATURES:\n\n");
    for(i = 0; i < 8; i++)
    {
        for(j = 0; j < n; j++)
        {
            printf("\t");
            c = getc(Mobiles[j]);
            while(c != 10)
            {
                putch(c);
                c = getc(Mobiles[j]);
            }
        }
        printf("\n");
    }
    for(i = 0; i < n; i++)
    {
        fclose(Mobiles[i]);
    }
}

void CompareLaptops(int n) //Enables User to Compare Features for several laptops
{
    int i, j;
    char c, lap[10], file[15], s[100];
```

```
    FILE *Laptops[n];
    Laptops[0] = fopen("laptop.txt", "r");

    //File Pointer array is used to point towards features file of each mobile selected
    for(i = 1; i < n; i++)
    {
        printf("\n\n\t\t\t\tSELECT LAPTOP %d\n", i);
        strcpy(lap, laptops());
        strcat(lap, "F.txt");
        Laptops[i] = fopen(lap, "r");
    }

    //Features for each mobile are displayed side by side by printing each file 1 line at a time
    printf("\n\t\t\t\tLAPTOPS FEATURES:\n\n");
    for(i = 0; i < 8; i++)
    {
        for(j = 0; j < n; j++)
        {
            printf("\t");
            c = getc(Laptops[j]);
            while(c != 10)
            {
                putch(c);
                c = getc(Laptops[j]);
            }
        }
        printf("\n");
    }
    for(i = 0; i < n; i++)
    {
```

```
        fclose(Laptops[i]);
    }
}

```

3.2.6 Product Details Module

```
void set_quantity(char id[10], int quantity) //Changes Quantity of a product depending on New Purchase/Shipment
{
    FILE *Product;
    char file[10], s[5];
    strcpy(file, id);
    strcat(file, ".txt");
    Product = fopen(file, "w"); //Product file is opened
    itoa(quantity, s, 10);
    fprintf(Product, "%s", s);
    fclose(Product);
}

int get_quantity(char id[10]) //Returns Quantity of selected prodcut
{
    FILE *Product;
    int quantity;
    char file[10], s[5], name[100];
    strcpy(file, id);
    strcat(file, ".txt");
    Product = fopen(file, "r"); //Product file is opened
    fscanf(Product, "%s", s);
    quantity = atoi(s);
    fclose(Product);

    //Warning is displayed if stocks of product are low
    if(quantity < 10)
    {
        printf("\n\t\tPRODUCT STOCK IS LOW");
        product_name(id, name);
        printf("\n\t\tPRODUCT: %s", name);
```

```
        printf("\n\t\tQUANTITY: %d", quantity);
        printf("\n\t\tORDER SHIPMENTS AT EARLIEST\n");
    }
    return quantity;
}

void features(char id[10]) //Displays features of product selected when Customer is selecting products to purchase
{
    FILE *Product, *fp;
    char c, file[10];
    int i;
    strcpy(file, id);
    strcat(file, "F.txt");
    Product = fopen(file, "r");

    //Product Features are printed
    printf("\n\n\t\t\t\tPRODUCT FEATURES:\n\n");
    if(id[0] == 'M')
    {
        fp = fopen("mobile.txt", "r");
        for(i = 0; i < 8; i++)
        {
            printf("\t");
            c = getc(fp);
            while(c != 10)
            {
                putch(c);
                c = getc(fp);
            }
            printf("\t");
```

```
            c = getc(Product);
            while(c != 10)
            {
                putch(c);
                c = getc(Product);
            }
            printf("\n");
        }
    }
    else
    {
        fp = fopen("laptop.txt", "r");
        for(i = 0; i < 8; i++)
        {
            printf("\t");
            c = getc(fp);
            while(c != 10)
            {
                putch(c);
                c = getc(fp);
            }
            printf("\t");
            c = getc(Product);
            while(c != 10)
            {
                putch(c);
                c = getc(Product);
            }
        }
        printf("\n");
    }
    fclose(fp);
    fclose(Product);
}

void product_name(char id[10], char name[100]) //Assigns Product Name to name[] String
{
    FILE *Product;
    char c, file[10];
    int i = 0;
    strcpy(file, id);
    strcat(file, "N.txt");
    Product = fopen(file, "r");
    c = fgetc(Product);
    while(c != EOF)
    {
        name[i] = c;
        i++;
        c = fgetc(Product);
    }
    name[i] = '\0';
    fclose(Product);
}

int product_price(char id[10]) //Returns Product Price
{
    FILE *Product;
    char s[10], file[10];
    int price;
    strcpy(file, id);
    strcat(file, "P.txt");
    Product = fopen(file, "r");
    fscanf(Product, "%s", s);
    price = atoi(s);
    fclose(Product);
    return price;
}
```

3.2.7 Product Selection Module

```
933  char* set_product()
934  //Used across the program to enable User to select product
935  //set_product() function branches into mobile() and laptop() functions which display Menus to select Mobile/Laptop brand
936  //mobile() and laptop() functions branch into several functions corresponding to each brand
937  //Each brand function finally enables User to select product and then returns its ID as a string
938  {
939      int c;
940      do{
941          printf("\n\t\t\t\tSELECT PRODUCT");
942          printf("\n\n\t\tPRESS:");
943          printf("\n\t\t1 TO SELECT MOBILE");
944          printf("\n\t\t2 TO SELECT LAPTOP");
945          printf("\n\t\tENTER YOUR CHOICE: ");
946          scanf("%d", &c);
947          switch(c)
948          {
949              case 1: return mobiles();
950              case 2: return laptops();
951              default:printf("\n\t\tINVALID INPUT...\n");
952          }
953      }while(1);
954  }
955
956  char* mobiles()
957  {
958      int c;
959      do{
960          printf("\n\t\t\t\tSELECT MOBILE BRAND");
961          printf("\n\n\t\tPRESS:");
962          printf("\n\t\t1 TO SELECT ONEPLUS");
```

```
963          printf("\n\t\t2 TO SELECT XIAOMI");
964          printf("\n\t\t3 TO SELECT SAMSUNG");
965          printf("\n\t\t4 TO SELECT APPLE");
966          printf("\n\t\tENTER YOUR CHOICE: ");
967          scanf("%d", &c);
968          switch(c)
969          {
970              case 1: return oneplus_mobiles();
971              case 2: return xiaomi_mobiles();
972              case 3: return samsung_mobiles();
973              case 4: return apple_mobiles();
974              default:printf("\n\t\tINVALID INPUT...\n");
975          }
976      }while(1);
977  }
978
979  char* oneplus_mobiles()
980  {
981      int c;
982      do{
983          printf("\n\t\t\t\tSELECT ONEPLUS MOBILE");
984          printf("\n\n\t\tPRESS:");
985          printf("\n\t\t1 TO SELECT ONEPLUS NORD 5G");
986          printf("\n\t\t2 TO SELECT ONEPLUS 8");
987          printf("\n\t\t3 TO SELECT ONEPLUS 8T");
988          printf("\n\t\t4 TO SELECT ONEPLUS 8 PRO");
989          printf("\n\t\tENTER YOUR CHOICE: ");
990          scanf("%d", &c);
991          switch(c)
992          {
```

```
993              case 1: return "M001";
994              case 2: return "M002";
995              case 3: return "M003";
996              case 4: return "M004";
997              default:printf("\n\t\tINVALID INPUT...\n");
998          }
999      }while(1);
1000 }
1001
1002 char* xiaomi_mobiles()
1003 {
1004     int c;
1005     do{
1006         printf("\n\t\t\t\tSELECT XIAMO MOBILE");
1007         printf("\n\n\t\tPRESS:");
1008         printf("\n\t\t1 TO SELECT REDMI 9");
1009         printf("\n\t\t2 TO SELECT REDMI 9A");
1010         printf("\n\t\t3 TO SELECT REDMI 9 POWER");
1011         printf("\n\t\t4 TO SELECT REDMI 9 PRIME");
1012         printf("\n\t\t5 TO SELECT REDMI NOTE 9");
1013         printf("\n\t\t6 TO SELECT REDMI NOTE 9 PRO");
1014         printf("\n\t\t7 TO SELECT REDMI NOTE 9 PRO MAX");
1015         printf("\n\t\tENTER YOUR CHOICE: ");
1016         scanf("%d", &c);
1017         switch(c)
1018         {
1019             case 1: return "M101";
1020             case 2: return "M102";
1021             case 3: return "M103";
1022             case 4: return "M104";
```

```
            case 5: return "M105";
            case 6: return "M106";
            case 7: return "M107";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}
char* samsung_mobiles()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT SAMSUNG MOBILE");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT SAMSUNG S10");
        printf("\n\t\t2 TO SELECT SAMSUNG S10 PLUS");
        printf("\n\t\t3 TO SELECT SAMSUNG S20");
        printf("\n\t\t4 TO SELECT SAMSUNG S20+");
        printf("\n\t\t5 TO SELECT SAMSUNG NOTE 10");
        printf("\n\t\t6 TO SELECT SAMSUNG NOTE 20");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return "M201";
            case 2: return "M202";
            case 3: return "M203";
            case 4: return "M204";
            case 5: return "M205";
            case 6: return "M206";
            default:printf("\n\t\tINVALID INPUT...\n");
```

```
        }
    }while(1);
}
char* apple_mobiles()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT APPLE MOBILE");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT iPHONE 12");
        printf("\n\t\t2 TO SELECT iPHONE 12 PRO");
        printf("\n\t\t3 TO SELECT iPHONE 12 PRO MAX");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return "M301";
            case 2: return "M302";
            case 3: return "M303";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

char* laptops()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT LAPTOP BRAND");
        printf("\n\n\t\tPRESS:");
```

```
        printf("\n\t\t1 TO SELECT HP");
        printf("\n\t\t2 TO SELECT DELL");
        printf("\n\t\t3 TO SELECT ASUS");
        printf("\n\t\t4 TO SELECT LENOVO");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return hp_laptops();
            case 2: return dell_laptops();
            case 3: return asus_laptops();
            case 4: return lenovo_laptops();
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

char* hp_laptops()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT HP LAPTOP");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT HP 15");
        printf("\n\t\t2 TO SELECT HP 15S");
        printf("\n\t\t3 TO SELECT HP PAVILION x360");
        printf("\n\t\t4 TO SELECT HP PAVILION GAMING");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
```

```
        {
            case 1: return "L001";
            case 2: return "L002";
            case 3: return "L003";
            case 4: return "L004";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

char* dell_laptops()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT DELL LAPTOP");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT DELL INSPIRON 3505");
        printf("\n\t\t2 TO SELECT DELL INSPIRON 3583");
        printf("\n\t\t3 TO SELECT DELL INSPIRON 3593");
        printf("\n\t\t4 TO SELECT DELL INSPIRON 3595");
        printf("\n\t\t5 TO SELECT DELL INSPIRON 5408");
        printf("\n\t\t6 TO SELECT DELL G3 3500");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return "L101";
            case 2: return "L102";
            case 3: return "L103";
            case 4: return "L104";
```

```
            case 5: return "L105";
            case 6: return "L106";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}
char* asus_laptops()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT ASUS LAPTOP");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT ASUS ZENBOOK 13");
        printf("\n\t\t2 TO SELECT ASUS ZENBOOK 14");
        printf("\n\t\t3 TO SELECT ASUS VIVOBOOK 15");
        printf("\n\t\t4 TO SELECT ASUS TUF GAMING");
        printf("\n\t\t5 TO SELECT ASUS ROG GAMING");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return "L201";
            case 2: return "L202";
            case 3: return "L203";
            case 4: return "L204";
            case 5: return "L205";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}
```

```
char* lenovo_laptops()
{
    int c;
    do{
        printf("\n\t\t\t\tSELECT LENOVO LAPTOP");
        printf("\n\n\t\tPRESS:");
        printf("\n\t\t1 TO SELECT LENOVO IDEAPAD SLIM 3");
        printf("\n\t\t2 TO SELECT LENOVO IDEAPAD S145");
        printf("\n\t\t3 TO SELECT LENOVO IDEAPAD S340");
        printf("\n\t\t4 TO SELECT LENOVO LEGION 5i");
        printf("\n\t\t5 TO SELECT LENOVO LEGION Y540");
        printf("\n\t\t6 TO SELECT LENOVO THINKPAD E14");
        printf("\n\t\t7 TO SELECT LENOVO THINKPAD E15");
        printf("\n\t\tENTER YOUR CHOICE: ");
        scanf("%d", &c);
        switch(c)
        {
            case 1: return "L301";
            case 2: return "L302";
            case 3: return "L303";
            case 4: return "L304";
            case 5: return "L305";
            case 6: return "L306";
            case 7: return "L307";
            default:printf("\n\t\tINVALID INPUT...\n");
        }
    }while(1);
}

```

3.2.8 Miscellaneous

```
void clear()
{
    printf("\n\n\t\tPRESS ANY KEY TO CONTINUE");
    getch();
    system("cls");
}
```

3.2.9 Github Link

https://github.com/MuneebHussaini/Mini-Project-ESIMS.git

3.2.10 Github Folder Structure

- esims.c is the C File
- esims.exe is the executable File
- Each Product has 4 files – Stock, Name, Price, Features
    - Stock File is indicated by product ID.

      Ex: L101
    - Name File is indicated by product ID followed by suffix N.

      Ex: L101N
    - Price File is indicated by product ID followed by suffix P.

      Ex: L101P
    - Features File is indicated by product ID followed by suffix F.

      Ex: L101F
- Each Employee ID has an account which stores the account password
- Each Customer Account has 2 files, 1 storing the account password and the other containing details of previous purchases
- mobile.txt and laptop.txt contain a generic list of features
- balance.txt stores the cash balance
- PreviousShipments.txt stores details of previous shipments


MuneebHussaini Add files via upload
ac37a4f 3 hours ago
4 commits
101.txt Add files via upload 17 hours ago
102.txt Add files via upload 17 hours ago
103.txt Add files via upload 17 hours ago
104.txt Add files via upload 17 hours ago
105.txt Add files via upload 17 hours ago
106.txt Add files via upload 17 hours ago
107.txt Add files via upload 17 hours ago
108.txt Add files via upload 17 hours ago
L001.txt Add files via upload 17 hours ago
L001F.txt Add files via upload 17 hours ago
L001N.txt Add files via upload 17 hours ago
L001P.txt Add files via upload 17 hours ago
L002.txt Add files via upload 17 hours ago
L002F.txt Add files via upload 17 hours ago
L002N.txt Add files via upload 17 hours ago


L002P.txt Add files via upload 17 hours ago
L003.txt Add files via upload 17 hours ago
L003F.txt Add files via upload 17 hours ago
L003N.txt Add files via upload 17 hours ago
L003P.txt Add files via upload 17 hours ago
L004.txt Add files via upload 17 hours ago
L004F.txt Add files via upload 17 hours ago
L004N.txt Add files via upload 17 hours ago
L004P.txt Add files via upload 17 hours ago
L101.txt Add files via upload 17 hours ago
L101F.txt Add files via upload 17 hours ago
L101N.txt Add files via upload 17 hours ago
L101P.txt Add files via upload 17 hours ago
L102.txt Add files via upload 17 hours ago
L102F.txt Add files via upload 17 hours ago
L102N.txt Add files via upload 17 hours ago

L102P.txt	Add files via upload	17 hours ago
L103.txt	Add files via upload	17 hours ago
L103F.txt	Add files via upload	17 hours ago
L103N.txt	Add files via upload	17 hours ago
L103P.txt	Add files via upload	17 hours ago
L104.txt	Add files via upload	17 hours ago
L104F.txt	Add files via upload	17 hours ago
L104N.txt	Add files via upload	17 hours ago
L104P.txt	Add files via upload	17 hours ago
L105.txt	Add files via upload	17 hours ago
L105F.txt	Add files via upload	17 hours ago
L105N.txt	Add files via upload	17 hours ago
L105P.txt	Add files via upload	17 hours ago
L106.txt	Add files via upload	17 hours ago
L106F.txt	Add files via upload	17 hours ago
L106N.txt	Add files via upload	17 hours ago

L106P.txt	Add files via upload	17 hours ago
L201.txt	Add files via upload	17 hours ago
L201F.txt	Add files via upload	17 hours ago
L201N.txt	Add files via upload	17 hours ago
L201P.txt	Add files via upload	17 hours ago
L202.txt	Add files via upload	17 hours ago
L202F.txt	Add files via upload	17 hours ago
L202N.txt	Add files via upload	17 hours ago
L202P.txt	Add files via upload	17 hours ago
L203.txt	Add files via upload	17 hours ago
L203F.txt	Add files via upload	17 hours ago
L203N.txt	Add files via upload	17 hours ago
L203P.txt	Add files via upload	17 hours ago
L204.txt	Add files via upload	17 hours ago
L204F.txt	Add files via upload	17 hours ago
L204N.txt	Add files via upload	17 hours ago


L204P.txt Add files via upload 17 hours ago
L205.txt Add files via upload 17 hours ago
L205F.txt Add files via upload 17 hours ago
L205N.txt Add files via upload 17 hours ago
L205P.txt Add files via upload 17 hours ago
L301.txt Add files via upload 17 hours ago
L301F.txt Add files via upload 17 hours ago
L301N.txt Add files via upload 17 hours ago
L301P.txt Add files via upload 17 hours ago
L302.txt Add files via upload 17 hours ago
L302F.txt Add files via upload 17 hours ago
L302N.txt Add files via upload 17 hours ago
L302P.txt Add files via upload 17 hours ago
L303.txt Add files via upload 17 hours ago
L303F.txt Add files via upload 17 hours ago
L303N.txt Add files via upload 17 hours ago
L303P.txt Add files via upload 17 hours ago


L304.txt Add files via upload 17 hours ago
L304F.txt Add files via upload 17 hours ago
L304N.txt Add files via upload 17 hours ago
L304P.txt Add files via upload 17 hours ago
L305.txt Add files via upload 17 hours ago
L305F.txt Add files via upload 17 hours ago
L305N.txt Add files via upload 17 hours ago
L305P.txt Add files via upload 17 hours ago
L306.txt Add files via upload 17 hours ago
L306F.txt Add files via upload 17 hours ago
L306N.txt Add files via upload 17 hours ago
L306P.txt Add files via upload 17 hours ago
L307.txt Add files via upload 17 hours ago
L307F.txt Add files via upload 17 hours ago
L307N.txt Add files via upload 17 hours ago
L307P.txt Add files via upload 17 hours ago

M001.txt	Add files via upload	3 minutes ago
M001F.txt	Add files via upload	3 minutes ago
M001N.txt	Add files via upload	3 minutes ago
M001P.txt	Add files via upload	3 minutes ago
M002.txt	Add files via upload	3 minutes ago
M002F.txt	Add files via upload	3 minutes ago
M002N.txt	Add files via upload	3 minutes ago
M002P.txt	Add files via upload	3 minutes ago
M003.txt	Add files via upload	3 minutes ago
M003F.txt	Add files via upload	3 minutes ago
M003N.txt	Add files via upload	3 minutes ago
M003P.txt	Add files via upload	3 minutes ago
M004.txt	Add files via upload	3 minutes ago
M004F.txt	Add files via upload	3 minutes ago
M004N.txt	Add files via upload	3 minutes ago
M004P.txt	Add files via upload	3 minutes ago
M101.txt	Add files via upload	3 minutes ago

M101F.txt	Add files via upload	3 minutes ago
M101N.txt	Add files via upload	3 minutes ago
M101P.txt	Add files via upload	3 minutes ago
M102.txt	Add files via upload	3 minutes ago
M102F.txt	Add files via upload	3 minutes ago
M102N.txt	Add files via upload	3 minutes ago
M102P.txt	Add files via upload	3 minutes ago
M103.txt	Add files via upload	3 minutes ago
M103F.txt	Add files via upload	3 minutes ago
M103N.txt	Add files via upload	3 minutes ago
M103P.txt	Add files via upload	3 minutes ago
M104.txt	Add files via upload	3 minutes ago
M104F.txt	Add files via upload	3 minutes ago
M104N.txt	Add files via upload	3 minutes ago
M104P.txt	Add files via upload	3 minutes ago
M105.txt	Add files via upload	3 minutes ago
M105F.txt	Add files via upload	3 minutes ago

M105N.txt	Add files via upload	3 minutes ago
M105P.txt	Add files via upload	3 minutes ago
M106.txt	Add files via upload	3 minutes ago
M106F.txt	Add files via upload	3 minutes ago
M106N.txt	Add files via upload	3 minutes ago
M106P.txt	Add files via upload	3 minutes ago
M107.txt	Add files via upload	3 minutes ago
M107F.txt	Add files via upload	3 minutes ago
M107N.txt	Add files via upload	3 minutes ago
M107P.txt	Add files via upload	3 minutes ago
M201.txt	Add files via upload	3 minutes ago
M201F.txt	Add files via upload	3 minutes ago
M201N.txt	Add files via upload	3 minutes ago
M201P.txt	Add files via upload	3 minutes ago
M202.txt	Add files via upload	3 minutes ago
M202F.txt	Add files via upload	3 minutes ago
M202N.txt	Add files via upload	3 minutes ago
M202P.txt	Add files via upload	3 minutes ago
M203.txt	Add files via upload	3 minutes ago
M203F.txt	Add files via upload	3 minutes ago
M203N.txt	Add files via upload	3 minutes ago
M203P.txt	Add files via upload	3 minutes ago
M204.txt	Add files via upload	3 minutes ago
M204F.txt	Add files via upload	3 minutes ago
M204N.txt	Add files via upload	3 minutes ago
M204P.txt	Add files via upload	3 minutes ago
M205.txt	Add files via upload	3 minutes ago
M205F.txt	Add files via upload	3 minutes ago
M205N.txt	Add files via upload	3 minutes ago
M205P.txt	Add files via upload	3 minutes ago
M206.txt	Add files via upload	3 minutes ago
M206F.txt	Add files via upload	3 minutes ago
M206N.txt	Add files via upload	3 minutes ago

M206P.txt	Add files via upload	4 minutes ago
M301.txt	Add files via upload	4 minutes ago
M301F.txt	Add files via upload	4 minutes ago
M301N.txt	Add files via upload	4 minutes ago
M301P.txt	Add files via upload	4 minutes ago
M302.txt	Add files via upload	4 minutes ago
M302F.txt	Add files via upload	4 minutes ago
M302N.txt	Add files via upload	4 minutes ago
M302P.txt	Add files via upload	4 minutes ago
M303.txt	Add files via upload	4 minutes ago
M303F.txt	Add files via upload	4 minutes ago
M303N.txt	Add files via upload	4 minutes ago
M303P.txt	Add files via upload	4 minutes ago
PreviousShipments.txt	Add files via upload	4 minutes ago
balance.txt	Add files via upload	17 hours ago

esims.c	Add files via upload	3 hours ago
esims.exe	Add files via upload	3 hours ago
laptop.txt	Add files via upload	17 hours ago
mobile.txt	Add files via upload	4 minutes ago
muneeb.txt	Add files via upload	4 minutes ago
muneebPH.txt	Add files via upload	4 minutes ago
prasun.txt	Add files via upload	4 minutes ago
prasunPH.txt	Add files via upload	4 minutes ago

3.3 Testing

3.3.1 Employee Use Cases

3.3.1.1 UC01 Employee Login

Test case ID: TC0101		Use case ID: **UC01**
Test case title: Employee Login		
Test case description: User attempts to Login with valid Employee ID and correct password		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters valid Employee ID and correct password associated with it.	"LOGGED IN AS EMPLOYEE" Confirmation Message is displayed.	"LOGGED IN AS EMPLOYEE" Confirmation Message is displayed.

Test case ID: TC0102		Use case ID: **UC01**
Test case title: Employee Login		
Test case description: User attempts to Login with valid Employee ID but incorrect password		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters valid Employee ID but incorrect password.	"LOG IN FAILED DUE TO INCORRECT ID OR PASSWORD" Error Message is displayed and User is returned to previous Menu.	"LOG IN FAILED DUE TO INCORRECT ID OR PASSWORD" Error Message is displayed and User is returned to previous Menu.

Test case ID: TC0103		Use case ID: **UC01**
Test case title: Employee Login		
Test case description: User attempts to Login with invalid Employee ID		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters invalid Employee ID.	"LOG IN FAILED DUE TO INCORRECT ID OR PASSWORD" Error Message is displayed and User is returned to previous Menu.	"LOG IN FAILED DUE TO INCORRECT ID OR PASSWORD" Error Message is displayed and User is returned to previous Menu.

3.3.1.2 UC02 Check Stocks

Test case ID: TC0201		Use case ID: **UC02**
Test case title: Check Stocks		
Test case description: Employee Checks Stocks for a product with sufficient stocks.		
Test steps	**Expected result**	**Actual result**
1. System prompts Employee to select product.	Current Product Stock is displayed.	Current Product Stock is displayed.

Test case ID: TC0202		Use case ID: **UC02**
Test case title: Check Stocks		
Test case description: Employee Checks Stocks for a product with low stocks.		
Test steps	**Expected result**	**Actual result**
1. System prompts Employee to select product.	Current Product Stock is displayed along with warning for low stocks.	Current Product Stock is displayed along with warning for low stocks.

3.3.1.3 UC03 Order Shipments

Test case ID: TC0301		Use case ID: **UC03**
Test case title: Order Shipments		
Test case description: Employee Orders Shipments with sufficient Cash Balance		
Test steps	**Expected result**	**Actual result**
1. System prompts Employee to select products. 2. Employee selects products and enters quantities such that total cost is less than current cash balance.	System displays summary of Products selected along with total price and Shipment is confirmed.	System displays summary of Products selected along with total price and Shipment is confirmed.

Test case ID: TC0302		Use case ID: **UC03**
Test case title: Order Shipments		
Test case description: Employee Orders Shipments with insufficient Cash Balance		
Test steps	**Expected result**	**Actual result**
1. System prompts Employee to select products. 2. Employee selects products and enters quantities such that total cost is more than current cash balance.	System displays summary of Products selected along with total price. Shipment is blocked due to insufficient cash balance.	System displays summary of Products selected along with total price. Shipment is blocked due to insufficient cash balance.

3.3.1.4 UC04 View Previous Shipments

Test case ID: TC0401		Use case ID: **UC04**
Test case title: View Previous Shipments		
Test case description: Employee Views Shipments		
Test steps	**Expected result**	**Actual result**
1. Employee selects 'View Previous Shipments' option.	System displays details of all Previous Shipments along with price, date and time.	System displays details of all Previous Shipments along with price, date and time.

3.3.1.5 UC05 Sell Products

Test case ID: TC0501		Use case ID: **UC05**
Test case title: Sell Products		
Test case description: Employee confirms purchase.		
Test steps	**Expected result**	**Actual result**
1. Employee is prompted to log in. 2. Employee is prompted to confirm purchase and payment. 3. Employee confirms.	"PURCHASE COMPLETED" Confirmation Message is displayed.	"PURCHASE COMPLETED" Confirmation Message is displayed.

Test case ID: TC0502		Use case ID: **UC05**
Test case title: Sell Products		
Test case description: Employee does confirm purchase.		
Test steps	**Expected result**	**Actual result**
1. Employee is prompted to log in. 2. Employee is prompted to confirm purchase and payment. 3. Employee does not confirm.	"PURCHASE NOT COMPLETED" Error Message is displayed.	"PURCHASE NOT COMPLETED" Error Message is displayed.

3.3.1.6 UC06 Check Cash Balance

Test case ID: TC0601		Use case ID: **UC06**
Test case title: Check Cash Balance		
Test case description: Employee checks cash balance		
Test steps	**Expected result**	**Actual result**
1. Employee selects 'Check Cash Balance' option.	System displays the current cash balance.	System displays the current cash balance.

3.3.2 Customer Use Cases

3.3.2.1 UC07 Customer Register

Test case ID: TC0701		Use case ID: **UC07**
Test case title: Customer Register		
Test case description: User Registers for new account with unique account name.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters unique Account name.	"NEW CUSTOMER ACCOUNT SUCCESSFULLY CREATED" Confirmation Message is displayed.	"NEW CUSTOMER ACCOUNT SUCCESSFULLY CREATED" Confirmation Message is displayed.

Test case ID: TC0702		Use case ID: **UC07**
Test case title: Customer Register		
Test case description: User attempts to register for new account with taken account name.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters taken Account name.	"ACCOUNT NAME TAKEN" Error Message is displayed.	"ACCOUNT NAME TAKEN" Error Message is displayed.

3.3.2.2 UC08 Customer Login

Test case ID: TC0801		Use case ID: **UC08**
Test case title: Customer Login		
Test case description: User attempts to Login with existing account and correct password		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters existing account name and correct password associated with it.	"LOGGED IN AS CUSTOMER" Confirmation Message is displayed.	"LOGGED IN AS CUSTOMER" Confirmation Message is displayed.

Test case ID: TC0802		Use case ID: **UC08**
Test case title: Customer Login		
Test case description: User attempts to Login with existing account but incorrect password		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters existing account name but incorrect password.	"INCORRECT PASSWORD" Error Message is displayed.	"INCORRECT PASSWORD" Error Message is displayed.

Test case ID: TC0803		Use case ID: **UC08**
Test case title: Customer Login		
Test case description: User attempts to Login with non-existing account.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter credentials. 2. User enters non-existing account name.	"ACCOUNT DOES NOT EXIST" Error Message is displayed.	"ACCOUNT DOES NOT EXIST" Error Message is displayed.

3.3.2.3 **UC09 Buy Products**

Test case ID: TC0901		Use case ID: **UC09**
Test case title: Buy Products		
Test case description: Customer completes purchase.		
Test steps	**Expected result**	**Actual result**
1. Customer selects products and enters quantities. 2. Customer initiates and completes purchase.	System prompts Employee for confirmation.	System prompts Employee for confirmation.

Test case ID: TC0902		Use case ID: **UC09**
Test case title: Buy Products		
Test case description: Customer does not complete purchase.		
Test steps	**Expected result**	**Actual result**
1. Customer selects products and enters quantities. 2. Customer does not initiate purchase.	"INITIATION FAILED…" Error Message is displayed.	" INITIATION FAILED…" Error Message is displayed.

3.3.3 Registered Customer Use Cases

3.3.3.1 UC10 View Previous Purchases

Test case ID: TC1001		Use case ID: **UC10**
Test case title: View Previous Purchases		
Test case description: Logged In Customer views Previous Purchases		
Test steps	**Expected result**	**Actual result**
1. Logged In Customer selects 'View Previous Purchases' option.	System displays details of all Previous Purchases for that account along with price, date and time.	System displays details of all Previous Purchases for that account along with price, date and time.

3.3.3.2 UC11 Redeem Coupons

Test case ID: TC1101		Use case ID: **UC11**
Test case title: Redeem Coupons		
Test case description: Logged In Customer enters valid Coupon Code		
Test steps	**Expected result**	**Actual result**
1. System asks Logged In Customer if they have a coupon code. 2. Logged In Customer enters valid Coupon Code.	Discount Message is Displayed and Purchase Initiation is prompted.	Discount Message is Displayed and Purchase Initiation is prompted.

Test case ID: TC1102		Use case ID: **UC11**
Test case title: Redeem Coupons		
Test case description: Logged In Customer does not enter coupon code.		
Test steps	**Expected result**	**Actual result**
1. System asks Logged In Customer if they have a coupon code. 2. Logged In Customer gives negative input.	Purchase Initiation is prompted.	Purchase Initiation is prompted.

3.3.4 Common Use Cases

3.3.4.1 UC12 View Product Features

Test case ID: TC1201		Use case ID: **UC12**
Test case title: View Product Features		
Test case description: User selects Mobile to display features.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to select product. 2. User selects mobile.	Mobile Features are displayed.	Mobile Features are displayed.

Test case ID: TC1202		Use case ID: **UC12**
Test case title: View Product Features		
Test case description: User selects Laptop to display features.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to select product. 2. User selects laptop.	Laptop Features are displayed.	Laptop Features are displayed.

3.3.4.2 UC13 Compare Product Features

Test case ID: TC1301		Use case ID: **UC13**
Test case title: Compare Product Features		
Test case description: User selects Mobiles to display features.		
Test steps	**Expected result**	**Actual result**
+1. System prompts User to enter number of products. 2. System prompts User to select products. 3. User selects mobiles.	Features of Mobiles are displayed side by side.	Features of Mobiles are displayed side by side.

Test case ID: TC1302		Use case ID: **UC13**
Test case title: Compare Product Features		
Test case description: User selects Laptops to display features.		
Test steps	**Expected result**	**Actual result**
1. System prompts User to enter number of products. 2. System prompts User to select products. 3. User selects laptops.	Features of Laptops are displayed side by side.	Features of Laptops are displayed side by side.

4 RESULTS

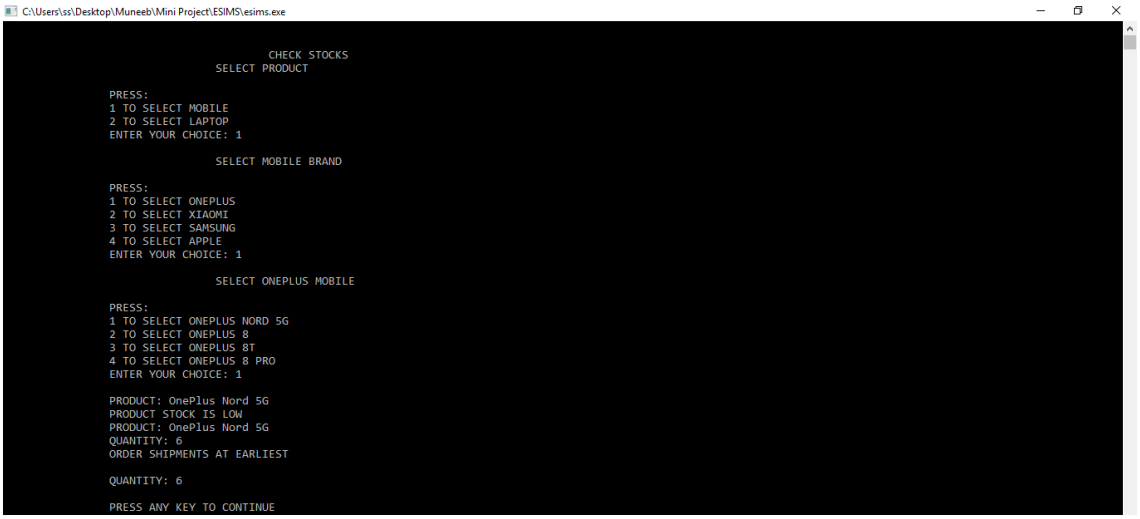
4.2 TC0101

4.3 TC0102

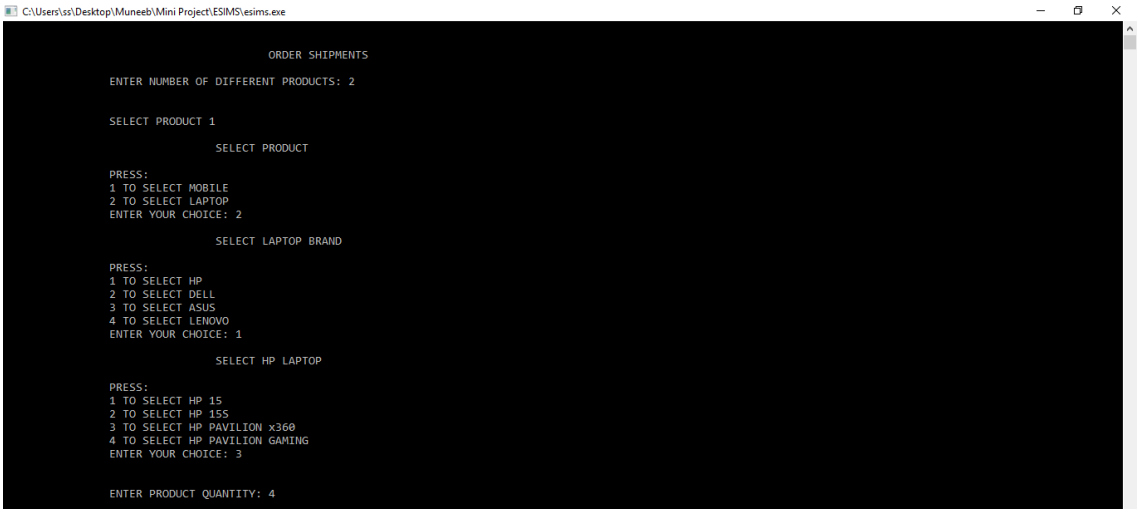
4.4 TC0103

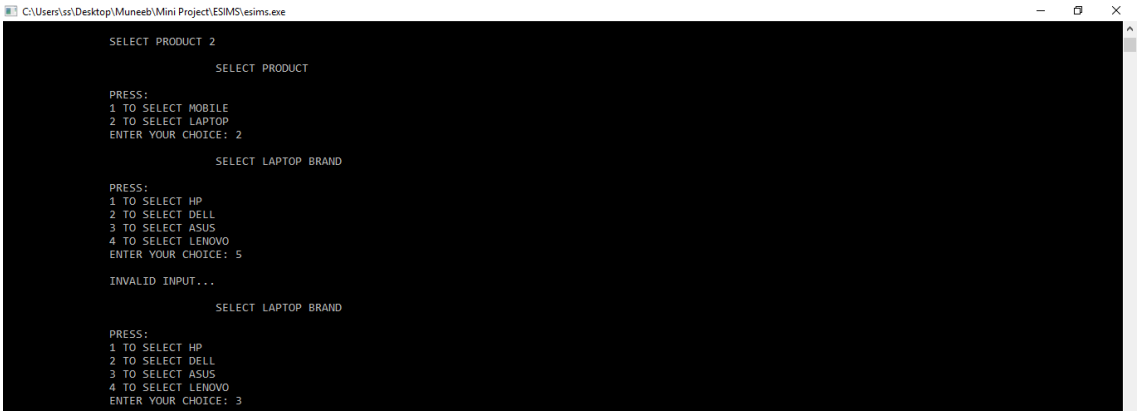
4.5 TC0201

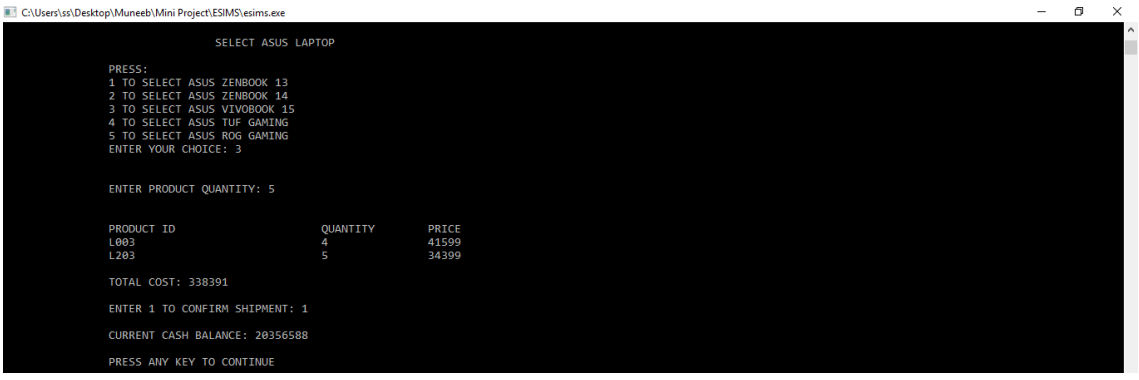
4.6 TC0202



4.7 TC0301

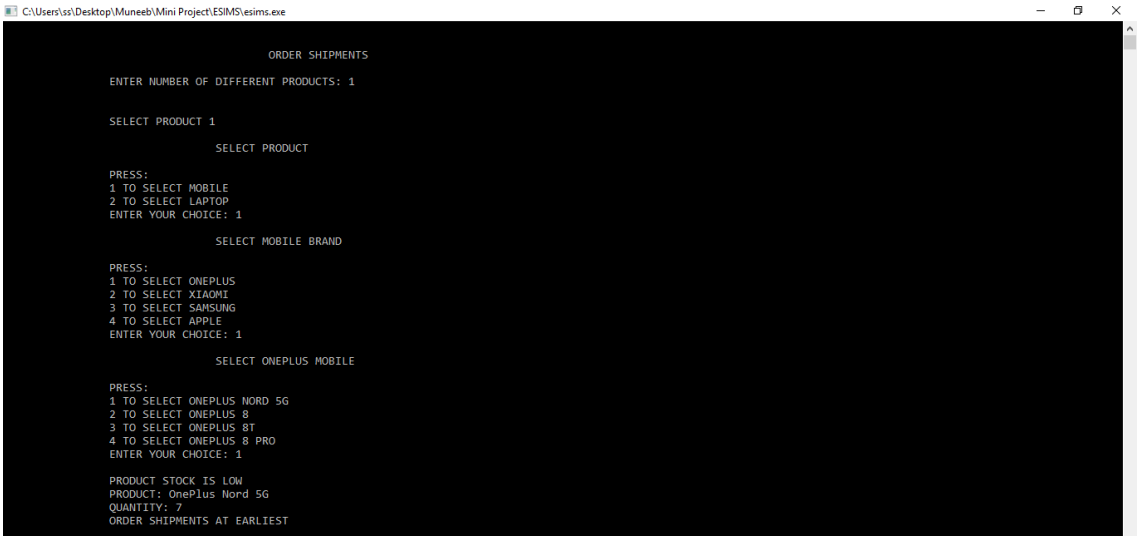

C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
ORDER SHIPMENTS
ENTER NUMBER OF DIFFERENT PRODUCTS: 2
SELECT PRODUCT 1
SELECT PRODUCT
PRESS:
1 TO SELECT MOBILE
2 TO SELECT LAPTOP
ENTER YOUR CHOICE: 2
SELECT LAPTOP BRAND
PRESS:
1 TO SELECT HP
2 TO SELECT DELL
3 TO SELECT ASUS
4 TO SELECT LENOVO
ENTER YOUR CHOICE: 1
SELECT HP LAPTOP
PRESS:
1 TO SELECT HP 15
2 TO SELECT HP 15S
3 TO SELECT HP PAVILION x360
4 TO SELECT HP PAVILION GAMING
ENTER YOUR CHOICE: 3
ENTER PRODUCT QUANTITY: 4


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
SELECT PRODUCT 2
SELECT PRODUCT
PRESS:
1 TO SELECT MOBILE
2 TO SELECT LAPTOP
ENTER YOUR CHOICE: 2
SELECT LAPTOP BRAND
PRESS:
1 TO SELECT HP
2 TO SELECT DELL
3 TO SELECT ASUS
4 TO SELECT LENOVO
ENTER YOUR CHOICE: 5
INVALID INPUT...
SELECT LAPTOP BRAND
PRESS:
1 TO SELECT HP
2 TO SELECT DELL
3 TO SELECT ASUS
4 TO SELECT LENOVO
ENTER YOUR CHOICE: 3

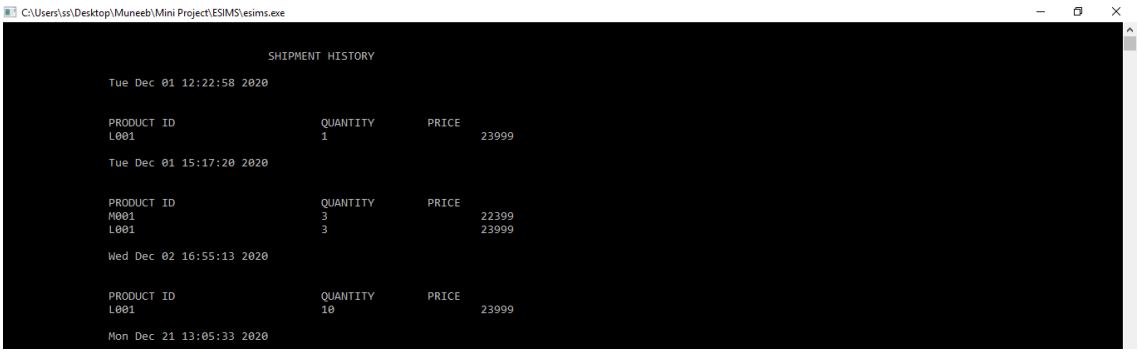

C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
SELECT ASUS LAPTOP
PRESS:
1 TO SELECT ASUS ZENBOOK 13
2 TO SELECT ASUS ZENBOOK 14
3 TO SELECT ASUS VIVOBOOK 15
4 TO SELECT ASUS TUF GAMING
5 TO SELECT ASUS ROG GAMING
ENTER YOUR CHOICE: 3
ENTER PRODUCT QUANTITY: 5
PRODUCT ID QUANTITY PRICE
L003 4 41599
L203 5 34399
TOTAL COST: 338391
ENTER 1 TO CONFIRM SHIPMENT: 1
CURRENT CASH BALANCE: 20356588
PRESS ANY KEY TO CONTINUE


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
SHIPMENT CONFIRMED
PRESS ANY KEY TO CONTINUE

4.8 TC0302



4.9 TC0401



4.10 TC0501

4.11 TC0502

```
EMPLOYEE CONFIRMATION REQUIRED

EMPLOYEE LOG IN
ENTER EMPLOYEE ID: 101

ENTER PASSWORD: bentley


PRESS ANY KEY TO CONTINUE
```

```
Select C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

EMPLOYEE ENTER 1 TO CONFIRM PURCHASE INITATION: 0


PURCHASE NOT COMPLETED

PRESS ANY KEY TO CONTINUE
```

4.12 TC0601

```
C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

CURRENT CASH BALANCE: 20362188

PRESS ANY KEY TO CONTINUE
```

4.13 TC0701

```
C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

ENTER ACCOUNT NAME (Max 30 Characters): hello

ENTER PASSWORD (Max 30 Characters): hello

CONFIRM PASSWORD: hello

NEW CUSTOMER ACCOUNT SUCCESSFULLY CREATED


PRESS ANY KEY TO CONTINUE
```

4.14 TC0702

```
Select C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

ENTER ACCOUNT NAME (Max 30 Characters): muneeb

ACCOUNT NAME TAKEN


PRESS ANY KEY TO CONTINUE
```

4.15 TC0801

```
C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

ENTER CUSTOMER ACCOUNT NAME: hello

ENTER PASSWORD: hello

LOGGED IN AS REGISTERED CUSTOMER


PRESS ANY KEY TO CONTINUE
```

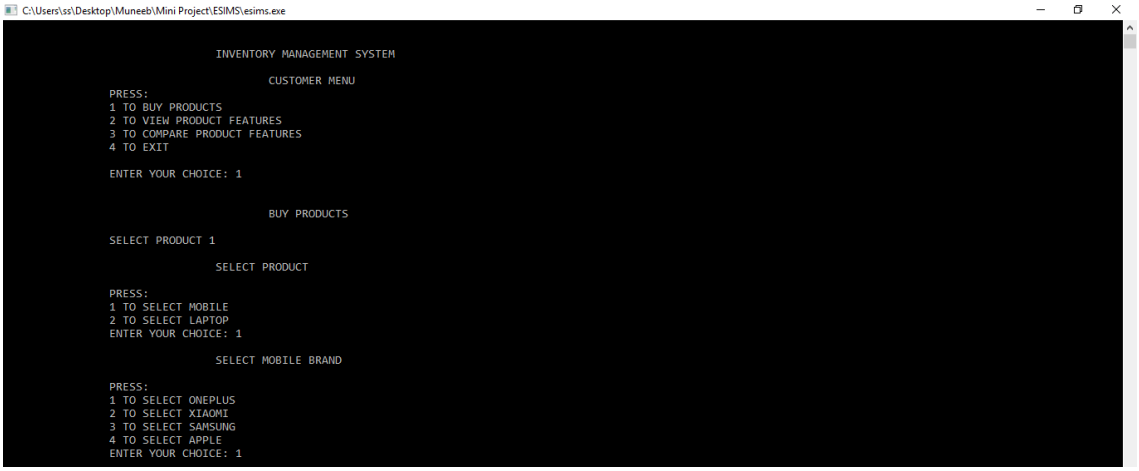
```
C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

                    INVENTORY MANAGEMENT SYSTEM

                              CUSTOMER MENU
PRESS:
1 TO BUY PRODUCTS
2 TO VIEW PREVIOUS PURCHASES
3 TO VIEW PRODUCT FEATURES
4 TO COMPARE PRODUCT FEATURES
5 TO EXIT

ENTER YOUR CHOICE:
```

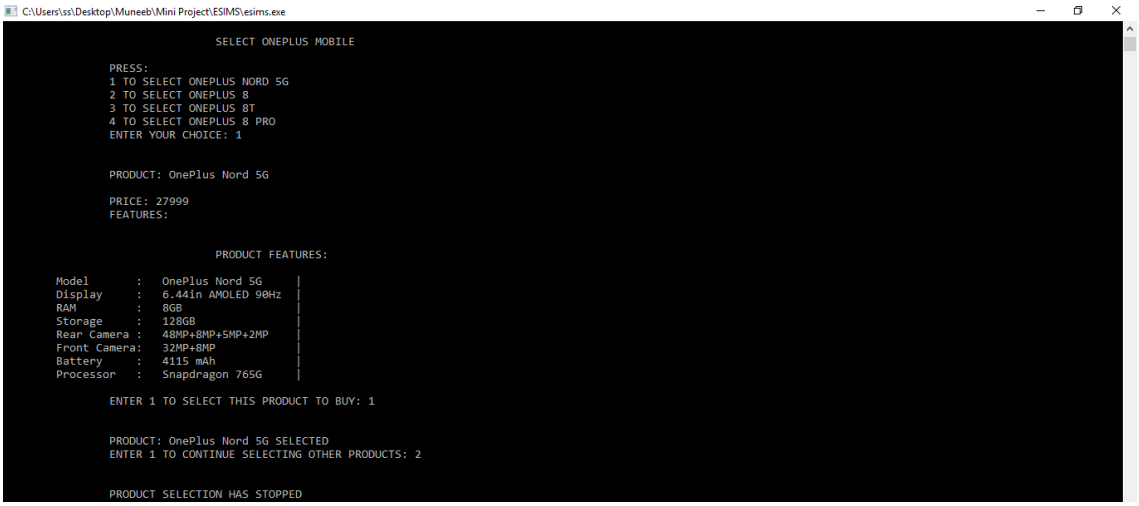
4.16 TC0802

```
C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe

ENTER CUSTOMER ACCOUNT NAME: muneeb

ENTER PASSWORD: hi

INCCORECT PASSWORD

PRESS ANY KEY TO CONTINUE
```

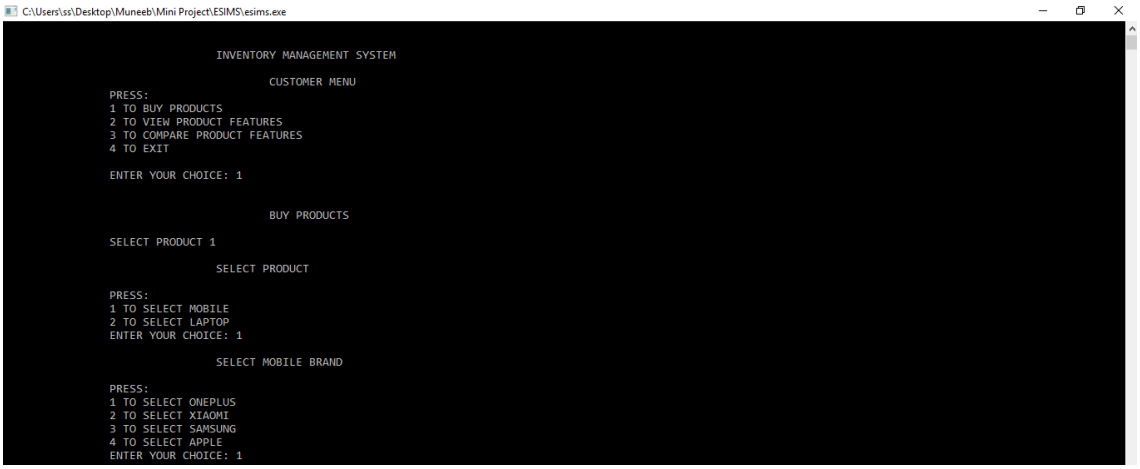
4.17 TC0803

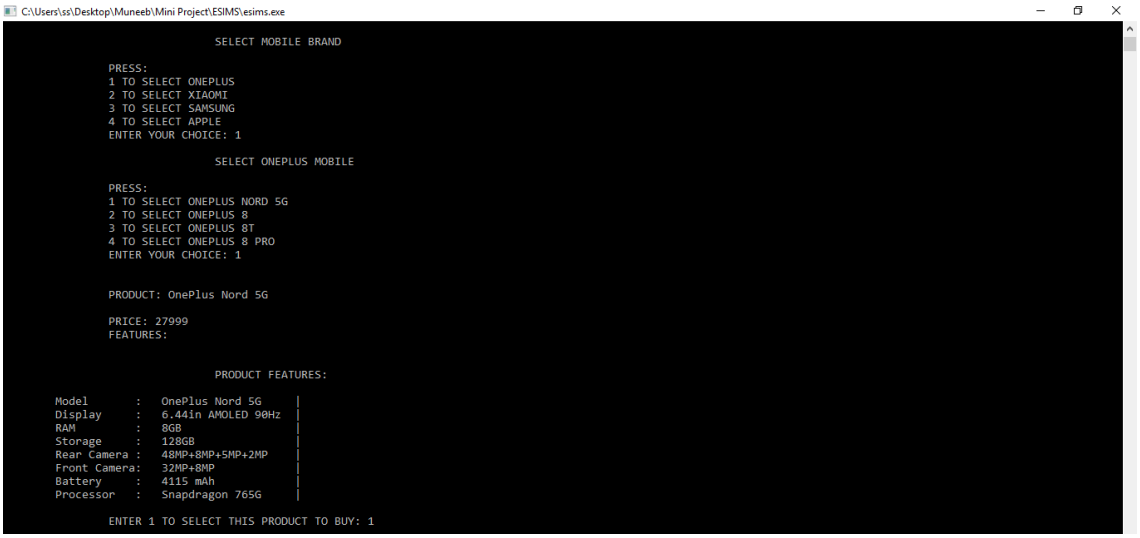
4.18 TC0901





4.19 TC0902





4.20 TC1001

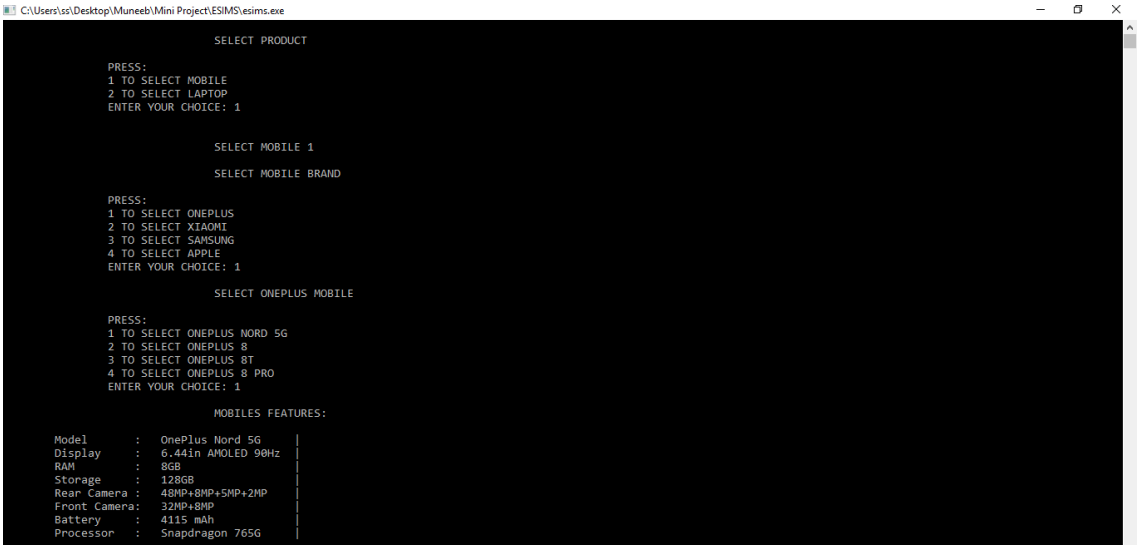
4.21 TC1101

```
PRODUCT ID                    QUANTITY        PRICE
M001                          1               27999

ENTER 1 IF YOU HAVE A COUPON: 1

ENTER COUPON CODE: MH178

10% DISCOUNT APPLIED

TOTAL COST: 25199

ENTER 1 TO INITIATE PURCHASE:
```

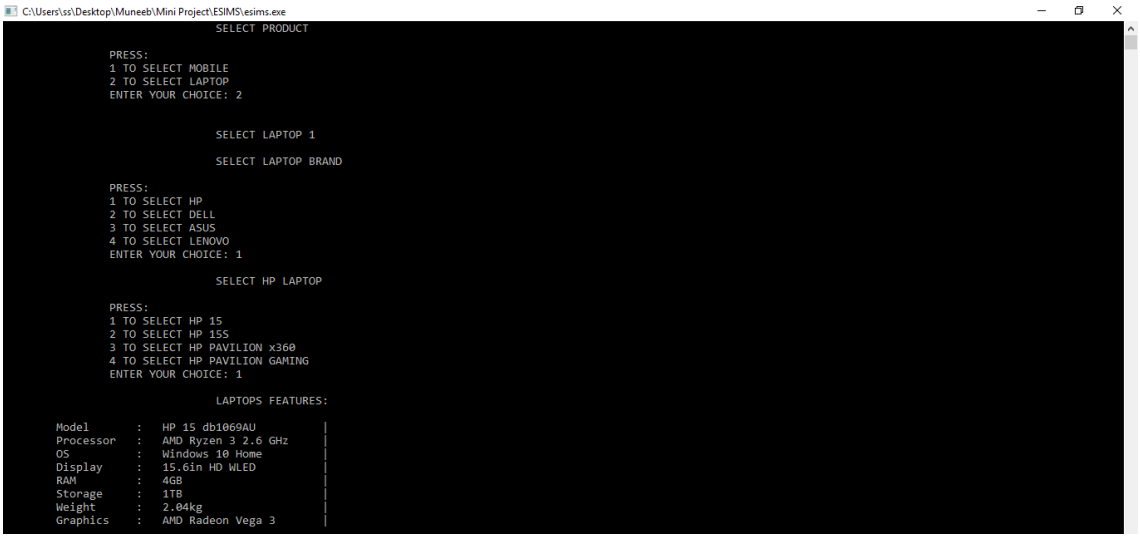
4.22 TC1102

```
PRODUCT ID                    QUANTITY        PRICE
M001                          1               27999

ENTER 1 IF YOU HAVE A COUPON: 0


TOTAL COST: 27999

ENTER 1 TO INITIATE PURCHASE:
```

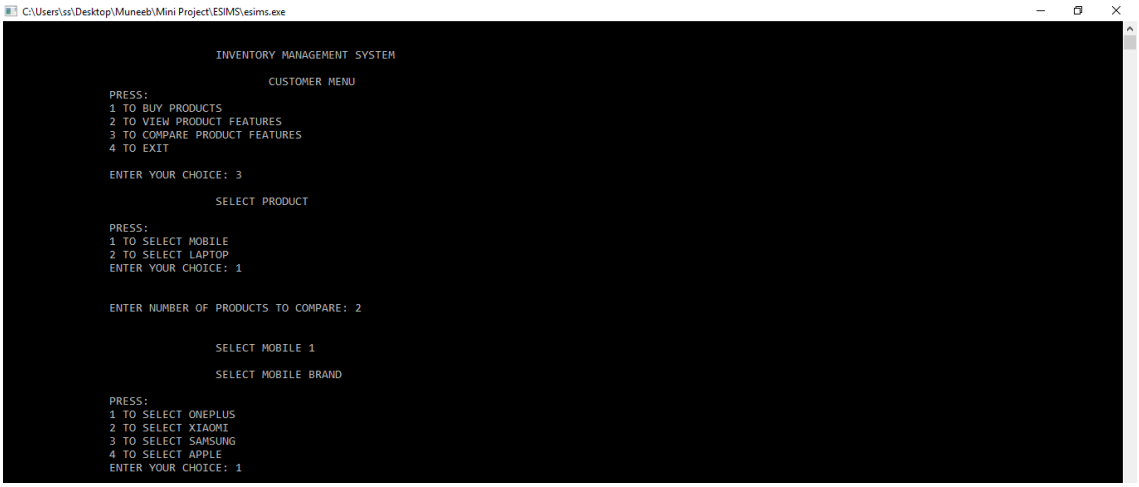
4.23 TC1201

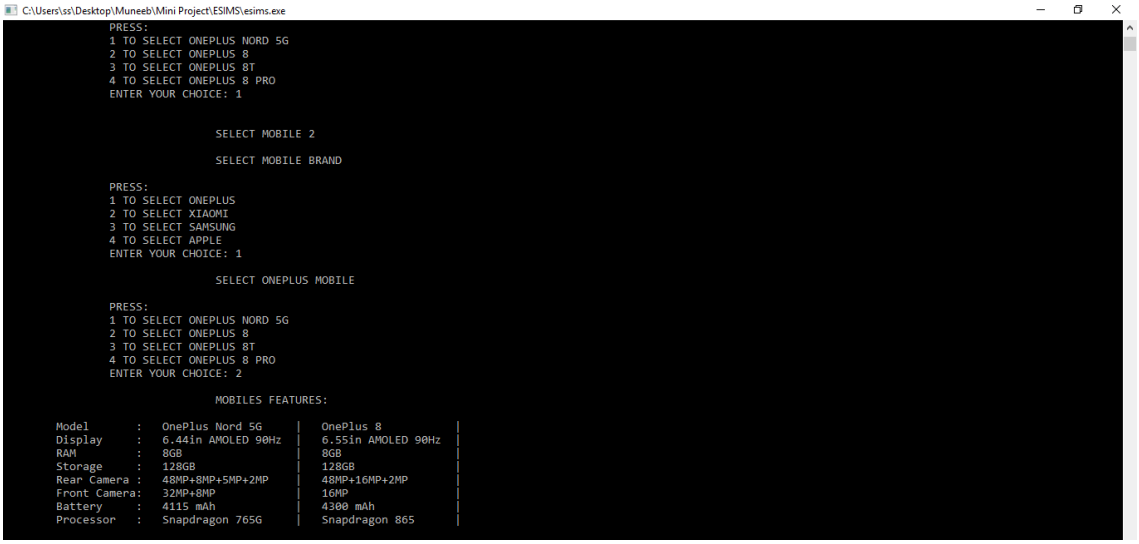


4.24 TC1202

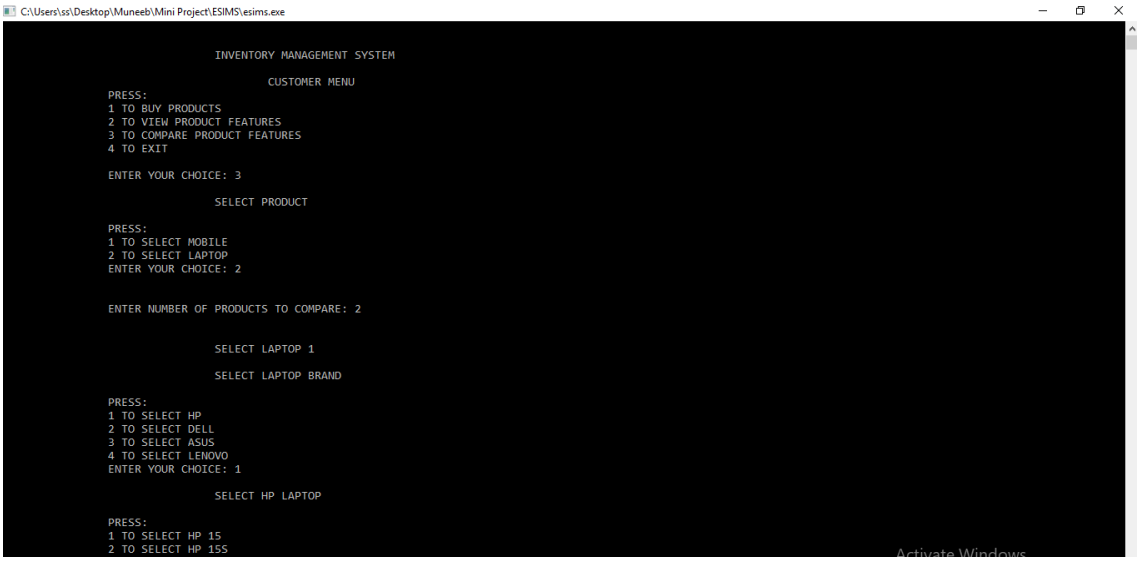


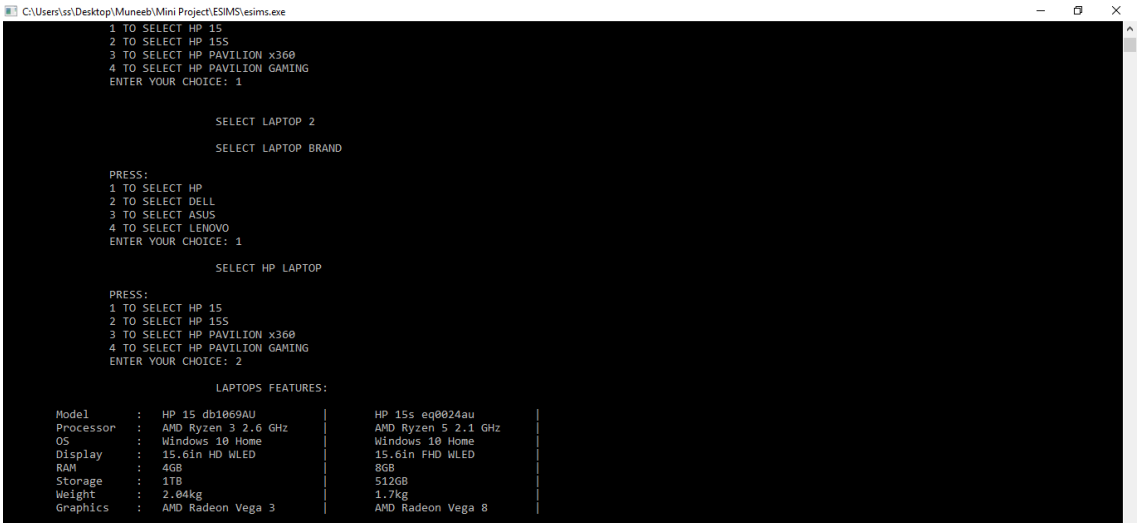
4.25 TC1301


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
INVENTORY MANAGEMENT SYSTEM
CUSTOMER MENU
PRESS:
1 TO BUY PRODUCTS
2 TO VIEW PRODUCT FEATURES
3 TO COMPARE PRODUCT FEATURES
4 TO EXIT
ENTER YOUR CHOICE: 3
SELECT PRODUCT
PRESS:
1 TO SELECT MOBILE
2 TO SELECT LAPTOP
ENTER YOUR CHOICE: 1
ENTER NUMBER OF PRODUCTS TO COMPARE: 2
SELECT MOBILE 1
SELECT MOBILE BRAND
PRESS:
1 TO SELECT ONEPLUS
2 TO SELECT XIAOMI
3 TO SELECT SAMSUNG
4 TO SELECT APPLE
ENTER YOUR CHOICE: 1


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
PRESS:
1 TO SELECT ONEPLUS NORD 5G
2 TO SELECT ONEPLUS 8
3 TO SELECT ONEPLUS 8T
4 TO SELECT ONEPLUS 8 PRO
ENTER YOUR CHOICE: 1
SELECT MOBILE 2
SELECT MOBILE BRAND
PRESS:
1 TO SELECT ONEPLUS
2 TO SELECT XIAOMI
3 TO SELECT SAMSUNG
4 TO SELECT APPLE
ENTER YOUR CHOICE: 1
SELECT ONEPLUS MOBILE
PRESS:
1 TO SELECT ONEPLUS NORD 5G
2 TO SELECT ONEPLUS 8
3 TO SELECT ONEPLUS 8T
4 TO SELECT ONEPLUS 8 PRO
ENTER YOUR CHOICE: 2
MOBILES FEATURES:
Model : OnePlus Nord 5G | OnePlus 8
Display : 6.44in AMOLED 90Hz | 6.55in AMOLED 90Hz
RAM : 8GB | 8GB
Storage : 128GB | 128GB
Rear Camera : 48MP+8MP+5MP+2MP | 48MP+16MP+2MP
Front Camera: 32MP+8MP | 16MP
Battery : 4115 mAh | 4300 mAh
Processor : Snapdragon 765G | Snapdragon 865

4.26 TC1302


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
INVENTORY MANAGEMENT SYSTEM
CUSTOMER MENU
PRESS:
1 TO BUY PRODUCTS
2 TO VIEW PRODUCT FEATURES
3 TO COMPARE PRODUCT FEATURES
4 TO EXIT
ENTER YOUR CHOICE: 3
SELECT PRODUCT
PRESS:
1 TO SELECT MOBILE
2 TO SELECT LAPTOP
ENTER YOUR CHOICE: 2
ENTER NUMBER OF PRODUCTS TO COMPARE: 2
SELECT LAPTOP 1
SELECT LAPTOP BRAND
PRESS:
1 TO SELECT HP
2 TO SELECT DELL
3 TO SELECT ASUS
4 TO SELECT LENOVO
ENTER YOUR CHOICE: 1
SELECT HP LAPTOP
PRESS:
1 TO SELECT HP 15
2 TO SELECT HP 15S


C:\Users\ss\Desktop\Muneeb\Mini Project\ESIMS\esims.exe
1 TO SELECT HP 15
2 TO SELECT HP 15S
3 TO SELECT HP PAVILION x360
4 TO SELECT HP PAVILION GAMING
ENTER YOUR CHOICE: 1
SELECT LAPTOP 2
SELECT LAPTOP BRAND
PRESS:
1 TO SELECT HP
2 TO SELECT DELL
3 TO SELECT ASUS
4 TO SELECT LENOVO
ENTER YOUR CHOICE: 1
SELECT HP LAPTOP
PRESS:
1 TO SELECT HP 15
2 TO SELECT HP 15S
3 TO SELECT HP PAVILION x360
4 TO SELECT HP PAVILION GAMING
ENTER YOUR CHOICE: 2
LAPTOPS FEATURES:
Model : HP 15 db1069AU | HP 15s eq0024au |
Processor : AMD Ryzen 3 2.6 GHz | AMD Ryzen 5 2.1 GHz |
OS : Windows 10 Home | Windows 10 Home |
Display : 15.6in HD WLED | 15.6in FHD WLED |
RAM : 4GB | 8GB |
Storage : 1TB | 512GB |
Weight : 2.04kg | 1.7kg |
Graphics : AMD Radeon Vega 3 | AMD Radeon Vega 8 |

5 ADDITIONAL KNOWLEDGE GAINED

Developing and building this project introduced us to new libraries such as conio.h and time.h. We used conio.h to build a better user interface and time.h to record details. The vast resources required to run this project also helped us improve our file handling and we were also able to successfully implement a File Pointer array.

The project also helped us dive into the depths of the commercial world and helped us learn more about it. We learned which features are important when it comes to different types of electronics.

6 CONCLUSION AND FUTURE WORK

In conclusion, we have successfully developed a project which can be utilized by electronic stores to manage their inventory, shipments and purchases. The project also makes the store experience much more customer and even employee friendly by allowing them to view and compare features of different products so as to make better choices. The Project is flexible and can be adjusted to incorporate different sets of products and prices.

The Project has opened up the possibilities for a much bigger and more comprehensive project for us. In the future, we can rebuild this project using databases and a graphical user interface to give a better feel to the user and make inventory management much easier. The Project also has scope to be connected to proper payment methods.

7 REFERENCES

1. http://geeksforgeeks.com/
2. https://stackoverflow.com/
3. https://www.gsmarena.com/
4. https://www.amazon.in/

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, SYED MUNEEBULLAH HUSSAINI and PRASUN REDDY bearing hall ticket numbers, 1602-19-737-178 and 1602-19-737-147, hereby declare that the project report entitled "ELECTRONIC STORE INVENTORY 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.

SYED MUNEEBULLAH HUSSAINI

1602-19-737-178

PRASUN REDDY

1602-19-737-147

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