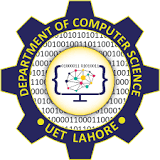
**Service Nest**

**Session 2023-2027**

**Submitted by:**

FajarShahzad 2023-CS-07

**Supervised by:**

Dr.Awais Hassan

**Course:**

CSC-102 Programming Fundamentals

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents:**

[● Description:](#_heading=h.pcgh0q8stajd)

[● User of the Application:](#_heading=h.qwlfxu5t9m)

[● Functional Requirements:](#_heading=h.2a2x7l9oqard)

[As Admin:](#_heading=h.615q1rca8p5n)

[As Customer:](#_heading=h.gj4b2mx0mm8r)

[● Wireframes:](#_heading=h.rce7y19rt7s4)

[● Data Structures(Parallel Arrays and Variables):](#_heading=h.vnx12oy7xxy7)

[Variables:](#_heading=h.g4rg9fk0fg1w)

[Parallel Arrays:](#_heading=h.lsr8tzb39ly6)

[● Function Prototypes:](#_heading=h.66m86mcidwkh)

[● Function Working Flow:](#_heading=h.qtgmhru1uvof)

[● Code Of Application:](#_heading=h.a5w22sz65q2p)

[● Weakness in the Business Application:](#_heading=h.6tg4zx51bbhr)

[● Future Directions:](#_heading=h.1u484x9bk3d5)

# Description:

My Application is about my own digital store on Daraz.I started this store on Daraz in 2022.Now I want to digitize my store and have my own application where users can check the list of products, order the products, and many more. This helps them to interact with the interface directly and enjoy the services provided by my store. The final output is the bill of the order.

# User of the Application:

The major users of my application are the customers and the store admin.

* **Customer:**Who has personal Customer login I'd to use the application.
* **Admin:** Who can manage the whole store and can edit the functionalities of the store system.

# Functional Requirements:

## As Admin:

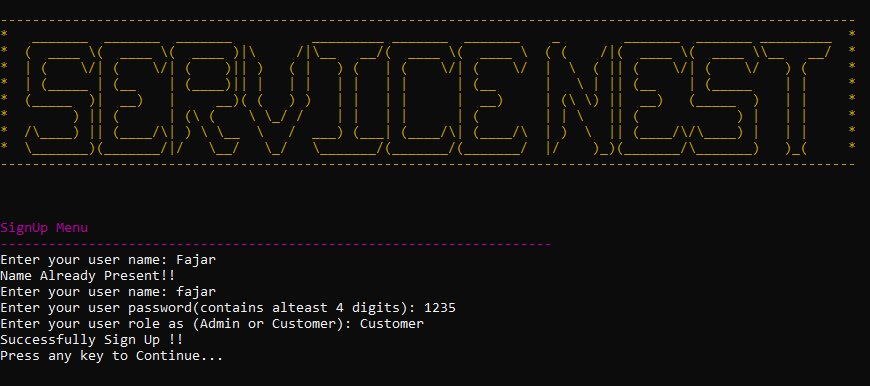
| ***As*** | ***I want to perform*** | ***So that I can*** |
| --- | --- | --- |
| Admin | Add a new product | Add the new product according to the new requirements |
| Admin | Remove the product | Remove the outdated product |
| Admin | View Product List | View the list of available products |
| Admin | View Customer List | To have a check on store customers |
| Admin | View Store Balance | To calculate the total balance and profit of the store |
| Admin | Remove Customer | To remove the customer |
| Admin | View Customer Reviews | To have check on store ratings and reviews |
| Admin | WithDraw Store Profit | To withdraw the profit amount and maintain the balance |
| Admin | View Customer Complaints | To overcome the complaints  of the customers |
| Admin | Logout | To logout from the app |

## As Customer:

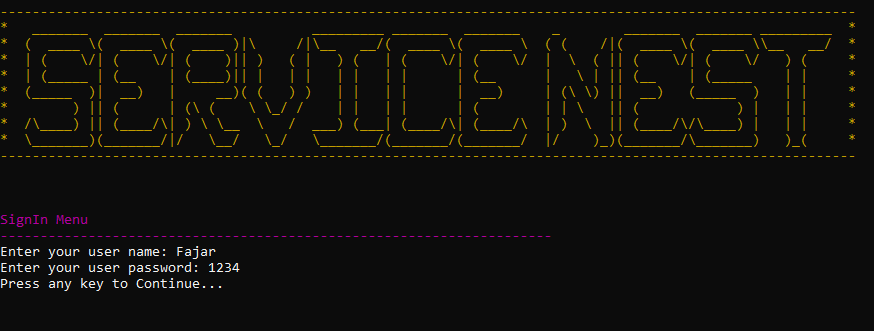
| ***As*** | ***I want to perform*** | ***So that I can*** |
| --- | --- | --- |
| Customer | View Product List | To view the list of products |
| Customer | Add To Cart | To add product into cart |
| Customer | View Cart | To View products from the cart |
| Customer | Shipping | To ready the product for shipping |
| Customer | Billing | To view the total bill |
| Customer | Delivery Time | To view the delivery time according to bill |
| Customer | Give Reviews | To give reviews according to the experience |
| Customer | Give Complaints | To give complaints according to requirement |
| Customer | Change Name | To change the user name |
| Customer | Change Password | To change the user password |
| Customer | Logout | To exit from the application |

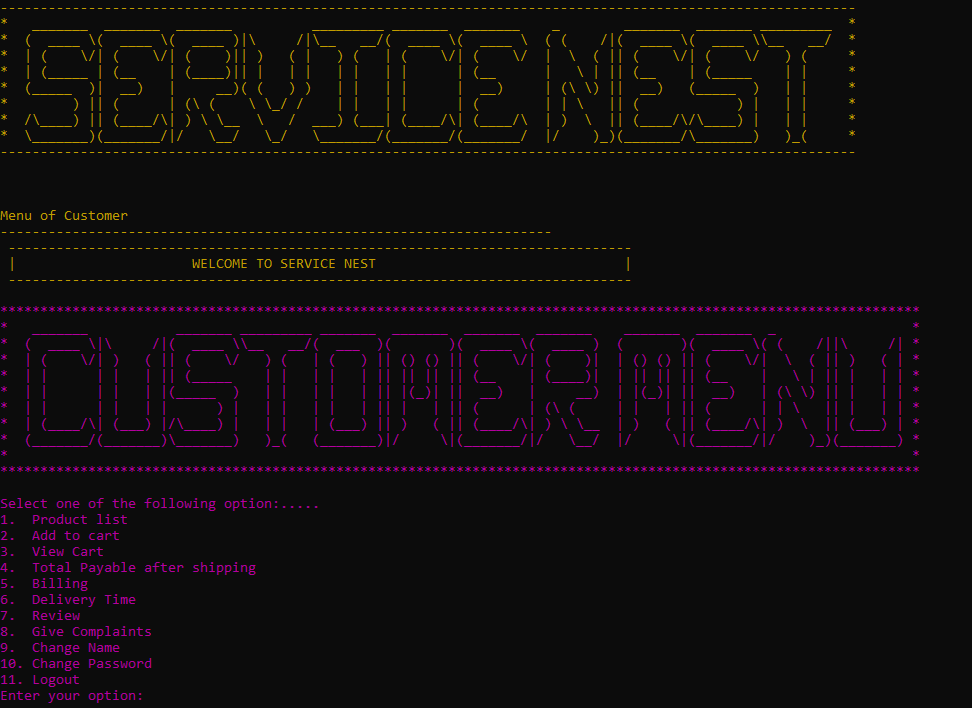
# Wireframes:

# 

**Fig1.1:Login Page**

**Fig1.2:SignUp Menu**

**Fig1.3:SignIn Menu**

**Fig1.4:Customer Menu**

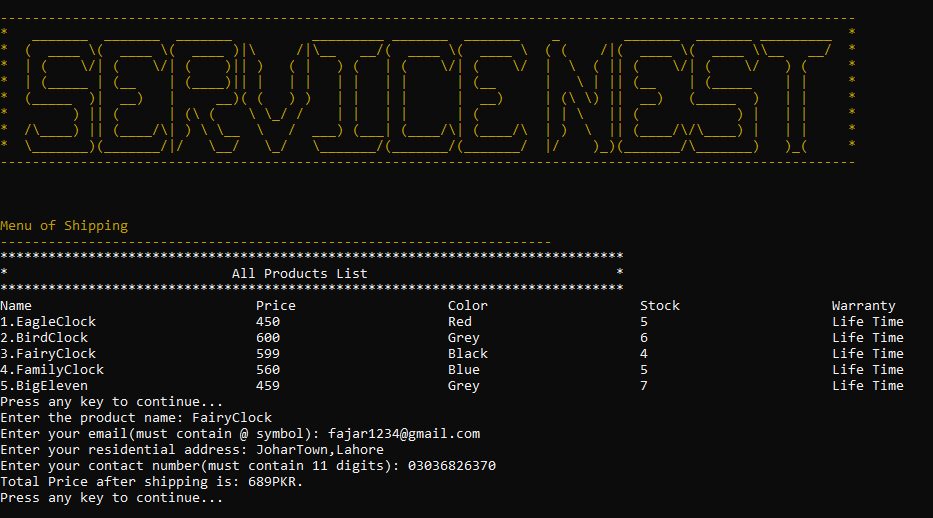


**Fig1.4.1:Products List**

****

**Fig1.4.2:Add To Cart Menu**

**Fig1.4.3:View Cart Menu**

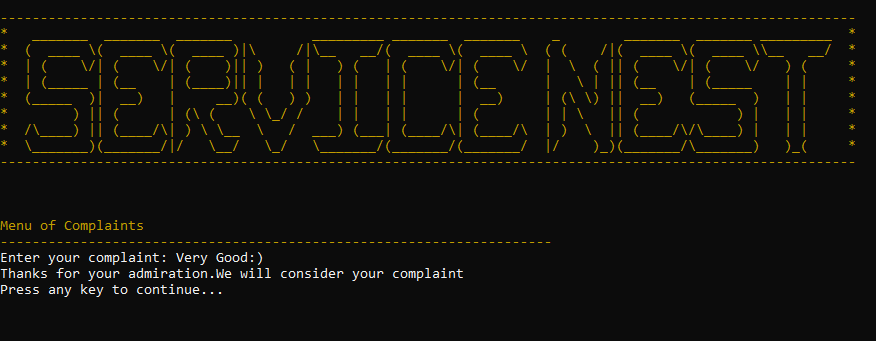
****

**Fig1.4.4:Shipping Menu**

**Fig1.4.5:Billing Menu**

**Fig1.4.6:Delivery Time Menu**

**Fig1.4.7:Review Menu**

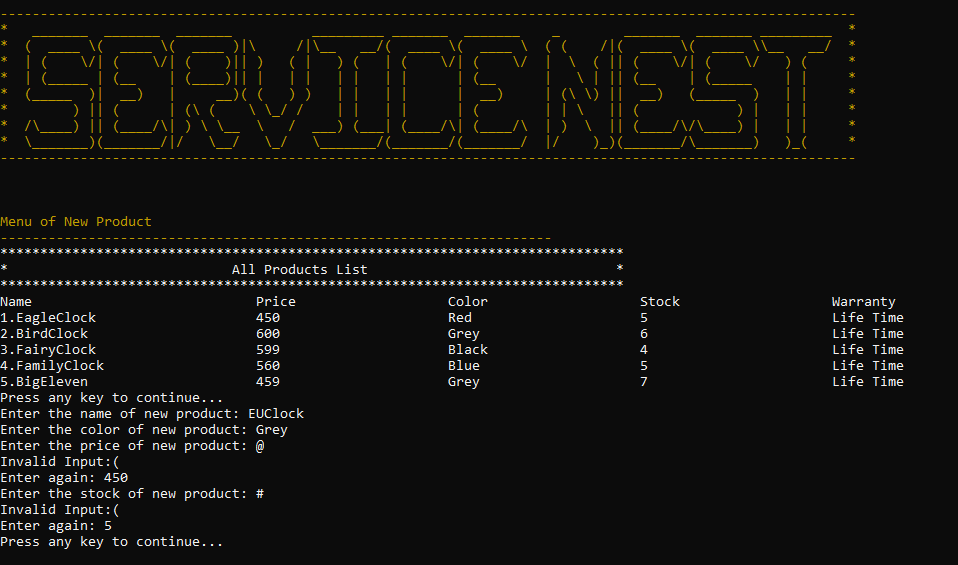
**Fig1.4.8:Complaints Menu**

**Fig1.4.9:Change Name Menu**

**Fig1.4.10:Change PasswordMenu**

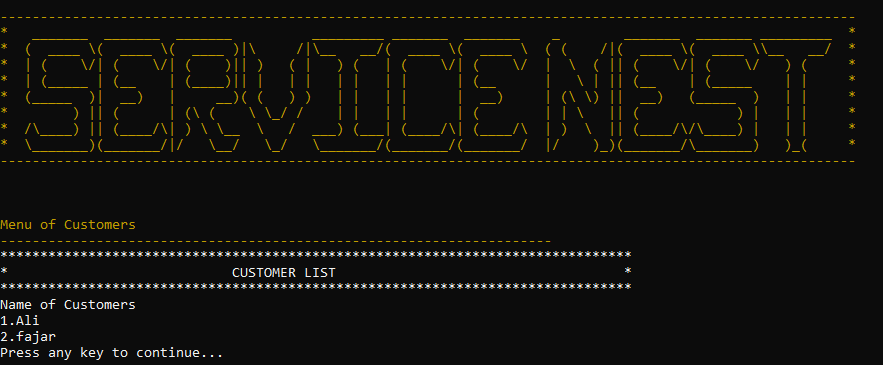
**Fig1.4.11:Logout Menu**

**Fig1.5:Admin Menu**

**Fig1.5.1:New Product Menu**

****

**Fig1.5.2:Updated Product List**

****

**Fig1.5.3:Customer List Menu**

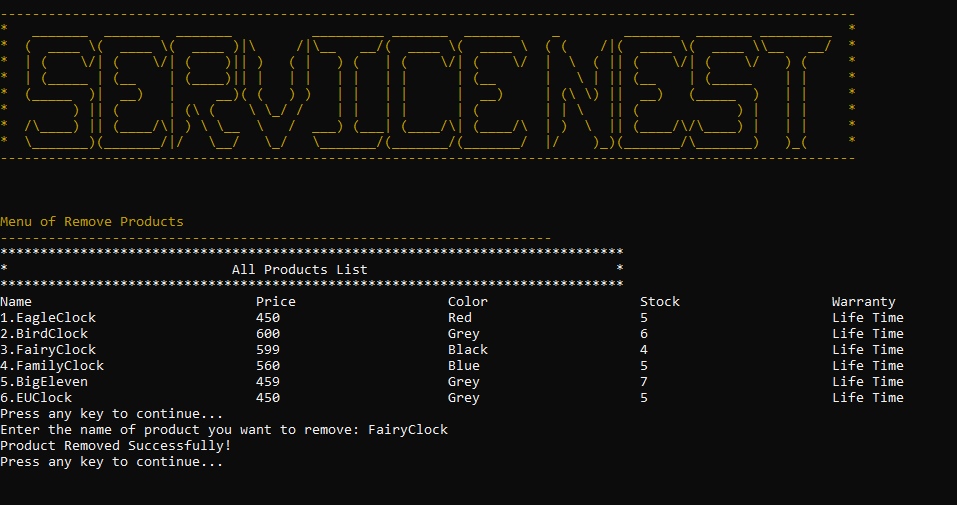
****

**Fig1.5.4:Balance Menu**

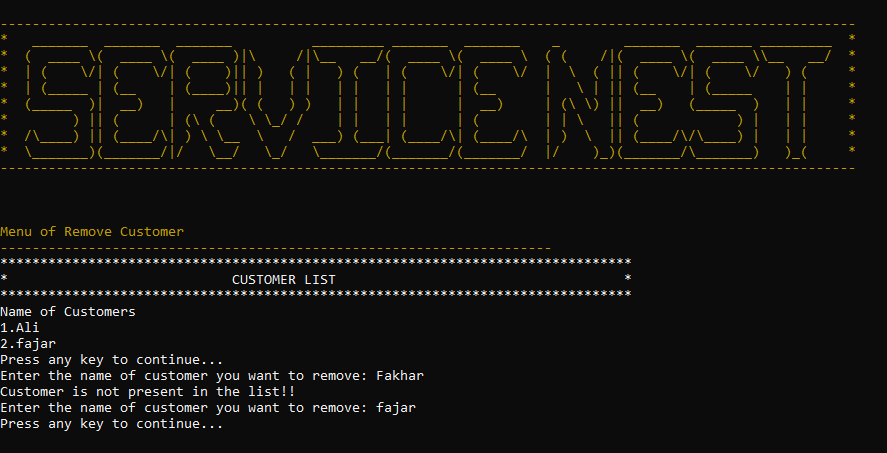
**Fig1.5.5:Reviews Menu**

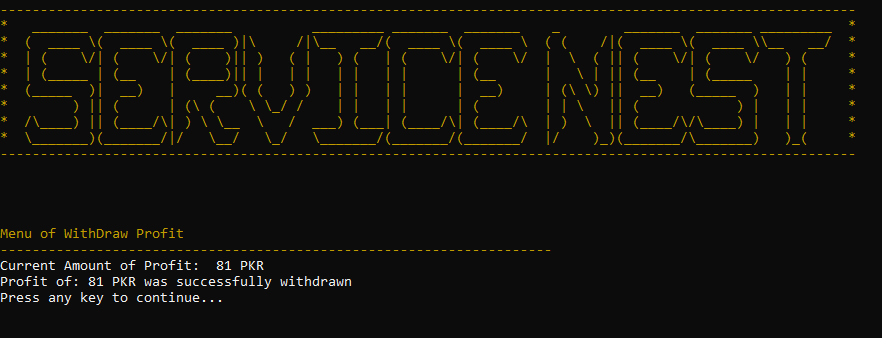
****

**Fig1.5.6:Store Complaints Menu**

****

**Fig1.5.7:Remove Product Menu**

**Fig1.5.8:Remove Customer Menu**

**Fig1.5.9:WithDraw Profit Menu**

**Fig1.5.10:Logout Menu**

# Data Structures(Parallel Arrays and Variables):

## Variables:

1. string name;
2. int price;
3. string stars;
4. string option;
5. int arrSize = 100;
6. string File\_Name;
7. int cartSize=0;
8. int personCount = 0;
9. int currentIndex = 0;
10. int idx = 0;
11. int reviewIndex = 0;
12. int productsCount=0;
13. string option1;

## Parallel Arrays:

1. string customerName[arrSize];
2. string Upassword[arrSize];
3. string Urole[arrSize];
4. string cart\_Product[arrSize];
5. string cart\_Price[arrSize];
6. string cart\_Color[arrSize];
7. string reviews[arrSize];
8. string complaint[arrSize];
9. float balance[arrSize] ;
10. string productName[arrSize] ;
11. string productColor[arrSize] ;
12. int productPrice[arrSize];
13. int productStock[arrSize] ;

# Function Prototypes:

1. void clearScreen();
2. void thanks();
3. void printheader();
4. string customerMenu();
5. string adminMenu();
6. string menu();
7. void adminInterface(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[], int personCount, string customerName[], string Urole[], float balance[], int currentIndex, string reviews[], string complaint[], int idx, int reviewIndex, string[], string[]);
8. void customerInterface(int productsCount, string productName[], int productPrice[], string productColor[], int productStock[], float balance[], int &currentIndex, string name, int price, string stars, string reviews[], string complaint[], int &idx, int &reviewIndex, string Upass[], string[],int &cartSize,string cart\_Product[],string cart\_Color[],string cart\_Price[]);
9. string signIN(string Name, string password, string Upassword[], string Urole[], string costumerName[], int personCount, int &currentIndex);
10. bool signUP(string Name, string password, string role, string costumerName[], string Upassword[], string Urole[], int &personCount, int arrSize);
11. void productname(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[]);
12. string addToCart(string name, int productsCount, string productName[], int productPrice[], string productColor[], int productStock[],int &cartSize,string cart\_Product[],string cart\_Color[],string cart\_Price[]);
13. void deliveryfunction(int price);
14. void shipping(string productName[], int productPrice[], int productsCount, int &Total, float balance[], int currentIndex);
15. void deliveryTime(int price);
16. void billing(float balance[], int currentIndex);
17. void review(string stars, string reviews[], int &reviewIndex);
18. void customerList(int personCount, string customerName[], string Upassword[], string Urole[]);
19. void viewBalance(float balance[], int personCount);
20. void ViewReviews(string reviews[], int currentIndex);
21. void AddNewProduct(int &productsCount, string productName[], string productColor[], int productPrice[], int productStock[]);
22. void ViewComplaints(string complaint[], int idx);
23. void viewCart(string cart\_Product[],string cart\_Color[],string cart\_Price[],int cartSize);
24. void complaints(string complaint[], int &idx);
25. void removeProduct(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[]);
26. void subMenu(string submenu);
27. void removeCustomer(int &personCount, string customerName[], string Urole[], string Upassword[]);
28. void subMenuBeforeMain(string submenu);
29. void withDrawProfit(float balance[], int personCount);
30. void changePassword(string Upass[], int currentIndex);
31. void changeName(string customerName[], int currentIndex);
32. bool validateEmail(string email);
33. bool containsOnlyAlphabets(string word);
34. int validateINT();
35. string setcolor(unsigned short color);
36. int strToInt(string s);

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILE HANDALING\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

1. void StoreLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int personCount);
2. void LoadLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int &personCount);
3. string getField(string record, int field);
4. void StoreReviewsFile(string reviews[], int reviewIndex);
5. void LoadReviews(string reviews[],string File\_Name, int &reviewIndex);
6. void StoreComplaintsFile(string complaint[], int idx);
7. void LoadComplaint(string complaint[],string File\_Name, int &idx);
8. void StoreProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],int productsCount);
9. void LoadProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],string File\_Name,int &productsCount);

# Function Working Flow:

Here is the working flow of my application



# Code Of Application:

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ALL LIBRARIES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

#include <iostream>

#include <conio.h>

#include <iomanip>

#include <string>

#include <windows.h>

#include <fstream>

using namespace std;

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ALL FUNCTION'S PROTOTYOES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

// use to clear the previous screen

void clearScreen();

// print thanks

void thanks();

// print thanks

void thanks();

// print header

void printheader();

// print Welcome

void welcome();

// show customer options

string customerMenu();

// show admin options

string adminMenu();

// print loginMenu

string menu();

// print adminInterface

void adminInterface(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[], int personCount, string customerName[], string Urole[], float balance[], int currentIndex, string reviews[], string complaint[], int idx, int reviewIndex, string[], string[]);

// print customerInterface

void customerInterface(int productsCount, string productName[], int productPrice[], string productColor[], int productStock[], float balance[], int &currentIndex, string name, int price, string stars, string reviews[], string complaint[], int &idx, int &reviewIndex, string Upass[], string[],int &cartSize,string cart\_Product[],string cart\_Color[],string cart\_Price[]);

/// signin menu

string signIN(string Name, string password, string Upassword[], string Urole[], string costumerName[], int personCount, int &currentIndex);

///sign up menu

bool signUP(string Name, string password, string role, string costumerName[], string Upassword[], string Urole[], int &personCount, int arrSize);

///show product list

void productname(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[]);

///add to cart

string addToCart(string name, int productsCount, string productName[], int productPrice[], string productColor[], int productStock[],int &cartSize,string cart\_Product[],string cart\_Color[],string cart\_Price[]);

/// take cost of product and return to deliveryTime

void deliveryfunction(int price);

/// calculate shipping price

void shipping(string productName[], int productPrice[], int productsCount, int &Total, float balance[], int currentIndex);

/// show delivery time

void deliveryTime(int price);

/// show total bill

void billing(float balance[], int currentIndex);

/// give reviews on the bases of stars

void review(string stars, string reviews[], int &reviewIndex);

/// show customer list to admin

void customerList(int personCount, string customerName[], string Upassword[], string Urole[]);

/// show totalBalance and Profit To admin

void viewBalance(float balance[], int personCount);

///show reviews to admin

void ViewReviews(string reviews[], int currentIndex);

/// allow to add new product list

void AddNewProduct(int &productsCount, string productName[], string productColor[], int productPrice[], int productStock[]);

/// show complaints to admin

void ViewComplaints(string complaint[], int idx);

/// Not working yet

void viewCart(string cart\_Product[],string cart\_Color[],string cart\_Price[],int cartSize);

/// allow customer to give reviews

void complaints(string complaint[], int &idx);

/// allow admin to remove product

void removeProduct(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[]);

/// Static menu for better UI

void subMenu(string submenu);

/// allow admin to remove customer

void removeCustomer(int &personCount, string customerName[], string Urole[], string Upassword[]);

/// Static menu for better UI

void subMenuBeforeMain(string submenu);

/// allow admin to withdraw profit

void withDrawProfit(float balance[], int personCount);

/// allow customer to change password

void changePassword(string Upass[], int currentIndex);

/// allow customer to change Name

void changeName(string customerName[], int currentIndex);

/// CheckEmailValidity

bool validateEmail(string email);

bool containsOnlyAlphabets(string word);

int validateINT();

/// Static color setting function for better UI

/// COLOR INT VALUES

/// -------------------------------

/// BLACK 0

/// BLUE 1

/// GREEN 2

/// CYAN 3

/// RED 4

/// MAGENTA 5

/// BROWN 6

/// LIGHTGRAY 7

/// DARKGRAY 8

/// LIGHTBLUE 9

/// LIGHTGREEN 10

/// LIGHTCYAN 11

/// LIGHTRED 12

/// LIGHTMAGENTA 13

/// YELLOW 14

/// WHITE 15

string setcolor(unsigned short color);

int strToInt(string s);

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILE HANDALING \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

void StoreLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int personCount);

void LoadLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int &personCount);

string getField(string record, int field);

void StoreReviewsFile(string reviews[], int reviewIndex);

void LoadReviews(string reviews[],string File\_Name, int &reviewIndex);

void StoreComplaintsFile(string complaint[], int idx);

void LoadComplaint(string complaint[],string File\_Name, int &idx);

void StoreProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],int productsCount);

void LoadProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],string File\_Name,int &productsCount);

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN FUNCTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

int main()

{

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VARIABLES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

string name;

int price;

string stars;

string option;

int arrSize = 100;

string File\_Name;

int cartSize=0;

int personCount = 0;

int currentIndex = 0;

int idx = 0;

int reviewIndex = 0;

int productsCount=0;

string option1;

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PARALLEL ARRAYS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

string customerName[arrSize];

string Upassword[arrSize];

string Urole[arrSize];

string cart\_Product[arrSize];

string cart\_Price[arrSize];

string cart\_Color[arrSize];

string reviews[arrSize];

string complaint[arrSize];

float balance[arrSize] ;

string productName[arrSize] ;

string productColor[arrSize] ;

int productPrice[arrSize];

int productStock[arrSize] ;

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* loading files\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

LoadLoginFile(customerName, Upassword, Urole, File\_Name, personCount);

LoadReviews(reviews,File\_Name,reviewIndex);

LoadComplaint(complaint,File\_Name,idx);

LoadProductsFile(productName,productColor,productPrice,productStock,File\_Name,productsCount);

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* While Loop start\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

while (option != "3")

{

printheader();

subMenuBeforeMain("Login");

option = menu();

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* option 1 for signin\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8///

if (option == "1")

{

system("cls");

string Uname;

string Userpassword;

string role;

printheader();

subMenuBeforeMain("SignIn");

cout << "Enter your user name: ";

cin.clear();

cin.sync();

getline(cin >> ws, Uname);

cout << "Enter your user password: ";

cin.clear();

cin.sync();

getline(cin >> ws, Userpassword);

role = signIN(Uname, Userpassword, Upassword, Urole, customerName, personCount, currentIndex);

if (role == "admin" || role == "Admin")

{

clearScreen();

adminInterface(productsCount, productName, productPrice, productColor, productStock, personCount, customerName, Urole, balance, currentIndex, reviews, complaint, idx, reviewIndex, customerName, Upassword);

}

else if (role == "customer" || role == "Customer")

{

clearScreen();

customerInterface(productsCount, productName, productPrice, productColor, productStock, balance, currentIndex, name, price, stars, reviews, complaint, idx, reviewIndex, Upassword, customerName,cartSize,cart\_Product,cart\_Color,cart\_Price);

}

else if ((role != "Customer"||role!="customer")&&(role!="Admin"||role!="admin"))

{

cout<<"Your role is not valid"<<endl;

}

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* option 2 for signup\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

else if (option == "2")

{

system("cls");

string Name, password, role;

printheader();

subMenuBeforeMain("SignUp");

again:

cout << "Enter your user name: ";

cin.clear();

cin.sync();

getline(cin >> ws, Name);

bool nameValidated = containsOnlyAlphabets(Name);

if (!nameValidated)

{

cout << "Invalid Name!!" << endl;

Sleep(600);

goto again;

}

for(int i=0;i<personCount;i++)

{

if(Name==customerName[i])

{

cout<<"Name Already Present!!"<<endl;

goto again;

}

}

again1:

cout << "Enter your user password(contains alteast 4 digits): ";

cin.clear();

cin.sync();

getline(cin >> ws, password);

if(password.length()!=4)

{

cout<<"Invalid length!!"<<endl;

goto again1;

}

else

{

for(int i=0;i<personCount;i++)

{

if(password==Upassword[i])

{

cout<<"Password Already Present!!"<<endl;

goto again1;

}

}

}

again2:

cout << "Enter your user role as (Admin or Customer): ";

cin.clear();

cin.sync();

getline(cin >> ws, role);

for(int i=0;i<personCount;i++)

{

if((role!="Customer"&&role!="customer")&&(role!="Admin"&&role!="admin"))

{

cout<<"Invalid Role!!"<<endl;

goto again2;

}

}

bool isValid = signUP(Name, password, role, customerName, Upassword, Urole, personCount, arrSize);

if (isValid)

{

cout << "Successfully Sign Up !!" << endl;

}

if (!isValid)

{

cout << "Try Again !! " << endl;

}

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*option 3 for storing files and exit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

else if (option == "3")

{

StoreLoginFile( customerName, Upassword, Urole, File\_Name, personCount);

StoreReviewsFile(reviews,reviewIndex);

StoreComplaintsFile(complaint,idx);

StoreProductsFile(productName,productColor,productPrice,productStock,productsCount);

return 0;

}

clearScreen();

}

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* UI ENHANCING FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

void printheader()

{

setcolor(6);

cout << R"(

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

\* \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \*

\* ( \_\_\_\_ \( \_\_\_\_ \( \_\_\_\_ )|\ /|\\_\_ \_\_/( \_\_\_\_ \( \_\_\_\_ \ ( ( /|( \_\_\_\_ \( \_\_\_\_ \\\_\_ \_\_/ \*

\* | ( \/| ( \/| ( )|| ) ( | ) ( | ( \/| ( \/ | \ ( || ( \/| ( \/ ) ( \*

\* | (\_\_\_\_\_ | (\_\_ | (\_\_\_\_)|| | | | | | | | | (\_\_ | \ | || (\_\_ | (\_\_\_\_\_ | | \*

\* (\_\_\_\_\_ )| \_\_) | \_\_)( ( ) ) | | | | | \_\_) | (\ \) || \_\_) (\_\_\_\_\_ ) | | \*

\* ) || ( | (\ ( \ \\_/ / | | | | | ( | | \ || ( ) | | | \*

\* /\\_\_\_\_) || (\_\_\_\_/\| ) \ \\_\_ \ / \_\_\_) (\_\_\_| (\_\_\_\_/\| (\_\_\_\_/\ | ) \ || (\_\_\_\_/\/\\_\_\_\_) | | | \*

\* \\_\_\_\_\_\_\_)(\_\_\_\_\_\_\_/|/ \\_\_/ \\_/ \\_\_\_\_\_\_\_/(\_\_\_\_\_\_\_/(\_\_\_\_\_\_\_/ |/ )\_)(\_\_\_\_\_\_\_/\\_\_\_\_\_\_\_) )\_( \*

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

)" << endl;

setcolor(15);

}

void welcome()

{

setcolor(6);

cout << setw(79) << "------------------------------------------------------------------------------" << endl;

cout << setw(79) << "| WELCOME TO SERVICE NEST |" << endl;

cout << setw(79) << "------------------------------------------------------------------------------" << endl;

setcolor(15);

}

void thanks()

{

setcolor(6);

cout <<setw(79) << "-------------------------------------------------------------------------------" << endl;

cout <<setw(79) << "| THANKS FOR VISITING! |" << endl;

cout << setw(79) <<"-------------------------------------------------------------------------------" << endl;

setcolor(15);

}

string customerMenu()

{

welcome();

string option1;

setcolor(13);

cout << R"(

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_ \*

\* ( \_\_\_\_ \|\ /|( \_\_\_\_ \\\_\_ \_\_/( \_\_\_ )( )( \_\_\_\_ \( \_\_\_\_ ) ( )( \_\_\_\_ \( ( /||\ /| \*

\* | ( \/| ) ( || ( \/ ) ( | ( ) || () () || ( \/| ( )| | () () || ( \/| \ ( || ) ( | \*

\* | | | | | || (\_\_\_\_\_ | | | | | || || || || (\_\_ | (\_\_\_\_)| | || || || (\_\_ | \ | || | | | \*

\* | | | | | |(\_\_\_\_\_ ) | | | | | || |(\_)| || \_\_) | \_\_) | |(\_)| || \_\_) | (\ \) || | | | \*

\* | | | | | | ) | | | | | | || | | || ( | (\ ( | | | || ( | | \ || | | | \*

\* | (\_\_\_\_/\| (\_\_\_) |/\\_\_\_\_) | | | | (\_\_\_) || ) ( || (\_\_\_\_/\| ) \ \\_\_ | ) ( || (\_\_\_\_/\| ) \ || (\_\_\_) | \*

\* (\_\_\_\_\_\_\_/(\_\_\_\_\_\_\_)\\_\_\_\_\_\_\_) )\_( (\_\_\_\_\_\_\_)|/ \|(\_\_\_\_\_\_\_/|/ \\_\_/ |/ \|(\_\_\_\_\_\_\_/|/ )\_)(\_\_\_\_\_\_\_) \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

)"

<< endl;

cout << "Select one of the following option:....." << endl;

cout << "1. Product list " << endl;

cout << "2. Add to cart " << endl;

cout << "3. View Cart" << endl;

cout << "4. Total Payable after shipping " << endl;

cout << "5. Billing " << endl;

cout << "6. Delivery Time " << endl;

cout << "7. Review " << endl;

cout << "8. Give Complaints " << endl;

cout << "9. Change Name" << endl;

cout << "10. Change Password" << endl;

cout << "11. Logout" << endl;

again:

cout << "Enter your option: ";

cin.clear();

cin.sync();

getline(cin >> ws, option1);

int Option1=strToInt(option1);

if(Option1<1||Option1 >11)

{

cout<<setw(55)<<"Invalid option!!"<<endl;

goto again;

}

setcolor(15);

return option1;

}

string adminMenu()

{

string option;

setcolor(6);

cout << R"(

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_

( \_\_\_ )( \_\_ \ ( )\\_\_ \_\_/( ( /| ( )( \_\_\_\_ \( ( /||\ /|

| ( ) || ( \ )| () () | ) ( | \ ( | | () () || ( \/| \ ( || ) ( |

| (\_\_\_) || | ) || || || | | | | \ | | | || || || (\_\_ | \ | || | | |

| \_\_\_ || | | || |(\_)| | | | | (\ \) | | |(\_)| || \_\_) | (\ \) || | | |

| ( ) || | ) || | | | | | | | \ | | | | || ( | | \ || | | |

| ) ( || (\_\_/ )| ) ( |\_\_\_) (\_\_\_| ) \ | | ) ( || (\_\_\_\_/\| ) \ || (\_\_\_) |

|/ \|(\_\_\_\_\_\_/ |/ \|\\_\_\_\_\_\_\_/|/ )\_) |/ \|(\_\_\_\_\_\_\_/|/ )\_)(\_\_\_\_\_\_\_)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

)" << endl;

cout << "Select one of the following option....." << endl;

cout << "1.Add New Product " << endl;

cout << "2.View Products List " << endl;

cout << "3.View Customer List" << endl;

cout << "4.View Balance" << endl;

cout << "5.View Customer Reviews" << endl;

cout << "6.View Customer Complaints" << endl;

cout << "7.Remove Product" << endl;

cout << "8.Remove Customer" << endl;

cout << "9.Withdraw Profit" << endl;

cout << "10.Logout" << endl;

again:

cout << "Enter your option: ";

cin.clear();

cin.sync();

getline(cin >> ws, option);

int Option=strToInt(option);

if(Option<1||Option >10)

{

cout<<setw(55)<<"Invalid option!!"<<endl;

goto again;

}

setcolor(15);

return option;

}

void subMenuBeforeMain(string submenu)

{

setcolor(13);

string message = submenu + " Menu";

cout << message << endl;

cout << "---------------------------------------------------------------------" << endl;

setcolor(15);

}

void subMenu(string submenu)

{

setcolor(6);

string message = "Menu of " + submenu;

cout << message << endl;

cout << "---------------------------------------------------------------------" << endl;

setcolor(15);

}

void clearScreen()

{

cout << "Press any key to Continue...";

getch();

system("cls");

}

string setcolor(unsigned short color)

{

HANDLE hcon = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hcon, color);

return "";

}

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOGIN MENU FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

string menu()

{

string option;

setcolor(13);

cout << setw(79) << R"( \_\_ \_\_\_\_\_ \_ \_\_\_\_\_ )" << endl;

cout << setw(79) << R"(/ | / \_\_\_(\_) |\_ \_| )" << endl;

cout << setw(79) << R"(`| | \ `--. \_ \_\_ \_ \_ \_\_ | | \_ \_\_ )" << endl;

cout << setw(79) << R"( | | `--. \ |/ \_` | '\_ \ | || '\_ \ )" << endl;

cout << setw(79) << R"(\_| |\_\_\ \\_\_/ / | (\_| | | | | \_| || | | | )" << endl;

cout << setw(79) << R"(\\_\_\_\_(\_)\_\_\_\_/|\_|\\_\_, |\_| |\_| \\_\_\_/\_| |\_| )" << endl;

cout << setw(79) << R"( \_\_/ | )" << endl;

cout << setw(79) << R"( |\_\_\_/ )" << endl;

cout << setw(79) << R"( \_\_\_\_\_ \_\_\_\_\_ \_ \_ \_ )" << endl;

cout << setw(79) << R"(/ \_\_ \ / \_\_\_(\_) | | | | )" << endl;

cout << setw(79) << R"(`' / /' \ `--. \_ \_\_ \_ \_ \_\_ | | | |\_ \_\_ )" << endl;

cout << setw(79) << R"( / / `--. \ |/ \_` | '\_ \ | | | | '\_ \ )" << endl;

cout << setw(79) << R"(./ /\_\_\_\_\ \\_\_/ / | (\_| | | | || |\_| | |\_) |)" << endl;

cout << setw(79) << R"(\\_\_\_\_\_(\_)\\_\_\_\_/|\_|\\_\_, |\_| |\_| \\_\_\_/| .\_\_/ )" << endl;

cout << setw(79) << R"( \_\_/ | | | )" << endl;

cout << setw(79) << R"( |\_\_\_/ |\_| )" << endl;

cout << setw(79) << R"( \_\_\_\_\_ \_\_\_\_\_ \_ \_ )" << endl;

cout << setw(79) << R"(|\_\_\_\_ || \_\_\_| (\_) | )" << endl;

cout << setw(79) << R"( / /| |\_\_\_\_ \_\_\_| |\_ )" << endl;

cout << setw(79) << R"( \ \| \_\_\_\_\/ /| \_\_| )" << endl;

cout << setw(79) << R"(.\_\_\_/ /| |\_\_ > < | | | )" << endl;

cout << setw(79) << R"(\\_\_\_\_(\_)\_\_\_\_/\_/\\_\_|\\_|\_| )" << endl;

cout<<endl;

again:

cout <<setw(55)<< "Enter your option: ";

cin.clear();

cin.sync();

getline(cin >> ws, option);

if(option<"1"||option >"3")

{

cout<<setw(55)<<"Invalid option!!"<<endl;

goto again;

}

setcolor(15);

return option;

}

string signIN(string Name, string password, string Upassword[], string Urole[], string customerName[], int personCount, int &currentIndex)

{

string result = "undefined";

for (int i = 0; i < personCount; i++)

{

if (customerName[i] == Name && Upassword[i] == password)

{

result = Urole[i];

currentIndex = i;

break;

}

}

return result;

}

bool signUP(string Name, string password, string role, string customerName[], string Upassword[], string Urole[], int &personCount, int arrSize)

{

bool isPresent = false;

for (int i = 0; i < personCount; i++)

{

if (Name == customerName[i] && password == Upassword[i])

isPresent = true;

}

if (isPresent == true)

{

return 0;

}

else if (personCount < arrSize)

{

customerName[personCount] = Name;

Upassword[personCount] = password;

Urole[personCount] = role;

personCount++;

return true;

}

else

{

return false;

}

}

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTION FOR BOTH END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

void productname(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[])

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* All Products List \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Name\t\t\t\tPrice\t\t\tColor\t\t\tStock\t\t\tWarranty " << endl;

for (int i = 0; i < productsCount; i++)

{

cout << i + 1 << "." << productName[i] <<"\t\t\t" <<productPrice[i] <<"\t\t\t"<< productColor[i] <<"\t\t\t" << productStock[i]<<"\t\t\t" << "Life Time" << endl;

}

cout << "Press any key to continue...";

getch();

}

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTIONS FOR CUSTOMER END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

string addToCart(string name, int productsCount, string productName[], int productPrice[], string productColor[], int productStock[],int &cartSize, string cart\_Product[],string cart\_Color[],string cart\_Price[])

{

again:

cout << "Enter the product name: ";

cin.clear();

cin.sync();

getline(cin >> ws, name);

bool found=false;

for (int i = 0; i < productsCount; i++)

{

if (name == productName[i]&&productStock[i]>0)

{

cout << "Price: " << productPrice[i] << endl;

cout << "Color: " << productColor[i] << endl;

// Add the product to the cart

cart\_Product[cartSize]=productName[i];

cart\_Price[cartSize]=to\_string(productPrice[i]);

cart\_Color[cartSize]=productColor[i];

cartSize++;

productStock[i] = productStock[i] - 1;

cout << "Successfully Added to Cart!" << endl;

cout << "Press any key to continue...";

getch();

found=true;

}

}

if (!found)

{

cout << "Product not found or out of stock." << endl;

Sleep(600);

goto again;

}

return "";

}

void viewCart(string cart\_Product[],string cart\_Color[],string cart\_Price[],int cartSize)

{

int j = 1;

cout<<"Name\t\t\t\tColor\t\t\t\tPrice"<<endl;

for (int i = 0; i < cartSize; i++)

{

cout << j << "." << cart\_Product[i]<<"\t\t\t"<<cart\_Color[i]<<"\t\t\t\t"<<cart\_Price[i] << endl;

j++;

}

cout << "Press any key to continue...";

getch();

}

void deliveryTime(int price)

{

again:

cout << "Enter your Product total cost: ";

cin >> price;

if(price<=0)

{

cout<<"Invalid Price:(" <<endl;

Sleep(600);

goto again;

}

deliveryfunction(price);

}

void deliveryfunction(int price)

{

if (price <= 450)

{

cout << "Delivery Time is 3 Days." << endl;

}

if (price >= 451 && price <= 640)

{

cout << "Delivery Time is 5 Days." << endl;

}

if (price >= 641)

{

cout << "Delivery Time is 7 Days." << endl;

}

cout << "Press any key to continue...";

getch();

}

void billing(float balance[], int currentIndex)

{

cout << "Your total bill is: " << balance[currentIndex] << endl;

cout << "Press any key to continue...";

getch();

}

void review(string stars, string reviews[], int &reviewIndex)

{

string result;

again:

cout << "Enter your ratings out of 5: ";

cin >> stars;

bool reviewValidated = containsOnlyAlphabets(stars);

if(!reviewValidated)

{

if(stars>="1"&&stars<="5")

{

if (stars <= "1")

{

result = "POOR";

}

else if (stars <= "2")

{

result = "NOT SATISFACTORY";

}

else if (stars <= "3")

{

result = "GOOD";

}

else if (stars <= "4")

{

result = "VERY GOOD";

}

else if (stars <= "5")

{

result = "HIGHLY SATISFIED";

}

}

else

{

cout<<"Invalid Input"<<endl;

Sleep(600);

goto again;

}

}

else

{

cout<<"Invalid Input"<<endl;

Sleep(600);

goto again;

}

cout << result << endl;

/// Adding reviews to the particular array

reviews[reviewIndex] = result;

reviewIndex++;

cout << "Press any key to continue...";

getch();

}

void complaints(string complaint[], int &idx)

{

string Complaint;

cout << "Enter your complaint: ";

cin.clear();

cin.sync();

getline(cin >> ws, Complaint);

string result = Complaint;

complaint[idx] = result;

idx++;

cout << "Thanks for your admiration.We will consider your complaint" << endl;

cout << "Press any key to continue...";

getch();

}

void shipping( string productName[], int productPrice[], int productsCount, int &Total, float balance[], int currentIndex)

{

again2:

string ProductName;

cout << "Enter the product name: ";

cin >> ProductName;

bool found=false;

for (int i = 0; i < productsCount; i++)

{

if (productName[i] == ProductName)

{

found=true;

again:

string email;

cout << "Enter your email(must contain @ symbol): ";

cin.clear();

cin.sync();

getline(cin >> ws, email);

bool addTheRateSymbol = validateEmail(email);

if (addTheRateSymbol == false)

{

cout << "Invalid Email" << endl;

goto again;

}

string address;

cout << "Enter your residential address: ";

cin.clear();

cin.sync();

getline(cin >> ws, address);

again1:

string number;

cout << "Enter your contact number(must contain 11 digits): ";

cin >> number;

if(number.length() !=11)

{

cout<<"Invalid Number"<<endl;

goto again1;

continue;

}

/// Calculating the price of product after shipping

int shippingprice = 90;

Total += productPrice[i] + shippingprice;

balance[currentIndex] += Total;

cout << "Total Price after shipping is: " << balance[currentIndex] << "PKR." << endl;

break;

}

}

if(!found)

{

cout << "Product Not Found"<<endl;

goto again2;

}

cout << "Press any key to continue...";

getch();

}

bool validateEmail(string email)

{

bool found = false;

for(int i = 0 ; i < email.length(); i++)

{

if (email[i] == '@')

{

found=true;

break;

}

}

return found;

}

void changeName(string customerName[], int currentIndex)

{

again:

string newName;

cout << "Enter new name: ";

cin.clear();

cin.sync();

getline(cin >> ws, newName);

bool newNameValidated = containsOnlyAlphabets(newName);

if(newNameValidated)

{

if(customerName[currentIndex] == newName)

{

cout<<"Name is same!!"<<endl;

goto again;

}

/// Changing name with the name at current index

customerName[currentIndex] = newName;

cout << "Changed Successfully!!" << endl;

cout << "Press any key to continue...";

getch();

}

else

{

cout<<"Invalid Input:("<<endl;

Sleep(600);

goto again;

}

}

void changePassword(string Upass[], int currentIndex)

{

again:

string newPass;

cout << "Enter new password(contains 4 digits only): ";

cin.clear();

cin.sync();

getline(cin >> ws, newPass);

if(newPass.length()!=4)

{

cout<<"Invalid length!!!"<<endl;

goto again;

}

if(Upass[currentIndex] == newPass)

{

cout<<"Password is same!!"<<endl;

goto again;

}

Upass[currentIndex] = newPass;

cout << "Changed Successfully!!" << endl;

cout << "Press any key to continue...";

getch();

}

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTIONS FOR ADMIN END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

void customerList(int personCount, string customerName[], string Urole[])

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* CUSTOMER LIST \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Name of Customers" << endl;

int j = 1;

for (int i = 0; i < personCount; i++)

{

if (Urole[i] == "customer" || Urole[i] == "Customer")

{

cout << j << "." << customerName[i] << endl;

j++;

}

}

cout << "Press any key to continue...";

getch();

cout<<endl;

}

void viewBalance(float balance[], int personCount)

{

float BALANCE = 0;

for (int i = 0; i < personCount; i++)

{

BALANCE += balance[i];

}

cout << "Current Store Balance: " << BALANCE << "PKR" << endl;

cout << "Current Amount of Profit: " << (BALANCE \* 15 / 100) << "PKR" << endl;

cout << "Press any key to continue...";

getch();

}

void ViewReviews(string reviews[], int reviewIndex)

{

int j = 1;

for (int i = 0; i < reviewIndex; i++)

{

cout << j << "." << reviews[i] <<endl;

j++;

}

cout << "Press any key to continue...";

getch();

}

void ViewComplaints(string complaint[], int idx)

{

int j = 1;

for (int i = 0; i < idx; i++)

{

cout << j << "." << complaint[i] << endl;

j++;

}

cout << "Press any key to continue...";

getch();

}

void AddNewProduct(int &productsCount, string productName[], string productColor[], int productPrice[], int productStock[])

{

productname(productsCount, productName, productPrice, productColor, productStock);

cout<<endl;

again:

string name;

cout << "Enter the name of new product: ";

cin.clear();

cin.sync();

getline(cin >> ws, name);

bool nameValidated = containsOnlyAlphabets(name);

if(nameValidated)

{

for(int i=0;i<productsCount;i++)

{

if(name==productName[i])

{

cout<<"Product is Already Present!!"<<endl;

goto again;

}

}

}

else

{

cout<<"Invalid Input:("<<endl;

goto again;

}

again3:

string color;

cout << "Enter the color of new product: ";

cin.clear();

cin.sync();

getline(cin >> ws, color);

bool colorValidated = containsOnlyAlphabets(color);

if(!colorValidated)

{

cout<<"Invalid Input:("<<endl;

goto again3;

}

again1:

cout << "Enter the price of new product: ";

int validatedPrice = validateINT();

if(validatedPrice<=0)

{

cout<<"Invalid Price!!"<<endl;

goto again1;

}

again2:

cout << "Enter the stock of new product: ";

int stock = validateINT();

if(stock<=0)

{

cout<<"Invalid Stock!!"<<endl;

goto again2;

}

productName[productsCount] = name;

productColor[productsCount] = color;

productPrice[productsCount] = validatedPrice;

productStock[productsCount] = stock;

productsCount++;

cout << "Press any key to continue...";

getch();

}

int validateINT()

{

AGAIN:

string number;

int temp = 0;

cin.clear();

cin.sync();

getline(cin >> ws, number);

try{

temp = stoi(number);

}

catch(invalid\_argument){

cout<<"Invalid Input:("<<endl;

Sleep(600);

cout << "Enter again: ";

goto AGAIN;

}

catch(out\_of\_range){

cout<<"Invalid Input:("<<endl;

Sleep(600);

cout << "Enter again: ";

goto AGAIN;

}

return temp;

}

void withDrawProfit(float balance[], int personCount)

{

cout << "Current Amount of Profit: " << (balance[personCount] \* 15 / 100) << "PKR" << endl;

/// removing the balance from current index

balance[personCount]=0;

cout << "Profit of: " << (balance[personCount] \* 15 / 100) << " PKR was successfully withdrawn" << endl;

cout << "Press any key to continue...";

getch();

}

void removeCustomer(int &personCount, string customerName[], string Urole[], string Upassword[])

{

customerList(personCount, customerName, Urole);

again:

string name;

cout << "Enter the name of customer you want to remove: ";

cin.clear();

cin.sync();

getline(cin >> ws, name);

bool nameValidated = containsOnlyAlphabets(name);

bool found = false;

int indexToRemove = 0;

if(nameValidated)

{

for (int i = 0; i < personCount; i++)

{

if (name == customerName[i])

{

found = true;

indexToRemove = i;

break;

}

}

}

else

{

cout<<"Invalid Input:("<<endl;

goto again;

}

if (found)

{

for (int i = indexToRemove; i < personCount; i++)

{

customerName[i] = customerName[i + 1];

Urole[i] = Urole[i + 1];

Upassword[i] = Upassword[i + 1];

}

personCount--;

}

else

{

cout<<"Customer is not present in the list!!"<<endl;

goto again;

}

cout << "Press any key to continue...";

getch();

}

void removeProduct(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[])

{

productname(productsCount, productName, productPrice, productColor, productStock);

cout<<endl;

again:

string name;

bool found = false;

cout << "Enter the name of product you want to remove: ";

cin.clear();

cin.sync();

getline(cin >> ws, name);

for (int i = 0; i < productsCount; i++)

{

if (name == productName[i])

{

for (int j = i; j < productsCount; j++)

{

productName[j] = productName[j + 1];

productPrice[j] = productPrice[j + 1];

productColor[j] = productColor[j + 1];

productStock[j] = productStock[j + 1];

}

productsCount--;

cout << "Product Removed Successfully!" << endl;

found = true;

break;

}

}

if (!found)

{

cout<<"Product is not present in the list!!"<<endl;

goto again;

}

cout << "Press any key to continue...";

getch();

}

void adminInterface(int &productsCount, string productName[], int productPrice[], string productColor[], int productStock[], int personCount, string customerName[], string Urole[], float balance[], int currentIndex, string reviews[], string complaint[], int idx, int reviewIndex, string Uname[], string Upassword[])

{

string adminOption;

while (adminOption != "10")

{

system("cls");

printheader();

subMenu("Admin");

adminOption = adminMenu();

if (adminOption == "1")

{

system("cls");

printheader();

subMenu("New Product");

AddNewProduct(productsCount, productName, productColor, productPrice, productStock);

}

if (adminOption == "2")

{

system("cls");

printheader();

subMenu("Products");

productname(productsCount, productName, productPrice, productColor, productStock);

}

if (adminOption == "3")

{

system("cls");

printheader();

subMenu("Customers");

customerList(personCount, customerName, Urole);

}

if (adminOption == "4")

{

system("cls");

printheader();

subMenu("Store Balance");

viewBalance(balance, personCount);

}

if (adminOption == "5")

{

system("cls");

printheader();

subMenu("Store Reviews");

ViewReviews(reviews, reviewIndex);

}

if (adminOption == "6")

{

system("cls");

printheader();

subMenu("Store Complaints");

ViewComplaints(complaint, idx);

}

if (adminOption == "7")

{

system("cls");

printheader();

subMenu("Remove Products");

removeProduct(productsCount, productName, productPrice, productColor, productStock);

}

if (adminOption == "8")

{

system("cls");

printheader();

subMenu("Remove Customer");

removeCustomer(personCount, customerName, Urole, Upassword);

}

if (adminOption == "9")

{

system("cls");

printheader();

subMenu("WithDraw Profit");

withDrawProfit(balance, personCount);

}

if (adminOption == "10")

{

system("cls");

printheader();

break;

}

}

}

void customerInterface(int productsCount, string productName[], int productPrice[], string productColor[], int productStock[], float balance[], int &currentIndex, string name, int price,string stars, string reviews[], string complaint[], int &idx, int &reviewIndex, string Upassword[], string customerName[],int &cartSize,string cart\_Product[],string cart\_Color[],string cart\_Price[])

{

string option1;

int Total = 0;

while (option1 != "11")

{

system("cls");

printheader();

subMenu("Customer");

option1 = customerMenu();

if (option1 == "1")

{

system("cls");

printheader();

subMenu("Products");

productname(productsCount, productName, productPrice, productColor, productStock);

}

if (option1 == "2")

{

system("cls");

printheader();

subMenu("Add To Cart");

productname(productsCount, productName, productPrice, productColor, productStock);

cout<<endl;

addToCart(name, productsCount, productName, productPrice, productColor, productStock,cartSize,cart\_Product,cart\_Color,cart\_Price);

}

if (option1 == "3")

{

system("cls");

printheader();

subMenu("View Cart");

viewCart(cart\_Product,cart\_Color,cart\_Price,cartSize);

}

if (option1 == "4")

{

system("cls");

printheader();

subMenu("Shipping");

productname(productsCount, productName, productPrice, productColor, productStock);

cout<<endl;

shipping( productName, productPrice, productsCount, Total, balance, currentIndex);

}

if (option1 == "5")

{

system("cls");

printheader();

subMenu("Billing");

billing(balance, currentIndex);

}

if (option1 == "6")

{

system("cls");

printheader();

subMenu("Delivery");

deliveryTime(price);

}

if (option1 == "7")

{

system("cls");

printheader();

subMenu("Reviews");

review(stars, reviews, reviewIndex);

}

if (option1 == "8")

{

system("cls");

printheader();

subMenu("Complaints");

complaints(complaint, idx);

}

if (option1 == "9")

{

system("cls");

printheader();

subMenu("Change Name");

changeName(customerName, currentIndex);

}

if (option1 == "10")

{

system("cls");

printheader();

subMenu("Change Password");

changePassword(Upassword, currentIndex);

}

if (option1 == "11")

{

system("cls");

printheader();

thanks();

}

}

}

bool containsOnlyAlphabets(string word) // It validates if the required input are alphabets

{

bool result = true;

for (int i = 0; i < word.length(); i++)

{

int asciiCode = word[i];

if (asciiCode < 65 || asciiCode > 122)

{

result = false;

break;

}

}

return result;

}

int strToInt(string s)

{

int result = 0;

for (int i = 0; s[i] != '\0'; i++)

{

result = result \* 10 + (s[i] - '0');

}

return result;

}

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

string getField(string record, int field)

{

int commaCount = 0;

string item="";

for (int x = 0; x < record.length(); x++)

{

if (record[x] == ',')

{

commaCount = commaCount + 1;

}

else if (commaCount == field)

{

item = item + record[x];

}

else if(commaCount>field)

{

break;

}

}

return item;

}

void LoadLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int &personCount)

{

string line="";

fstream loginData;

loginData.open("LoginFile.txt", ios::in);

if (loginData.fail())

return;

while (!loginData.eof())

{

getline(loginData, line);

customerName[personCount] = getField(line, 0); // data variable holding the whole line , 0 is index of username in file

Upassword[personCount] = getField(line, 1);

Urole[personCount] = getField(line, 2);

personCount = personCount + 1;

}

loginData.close();

}

void StoreLoginFile(string customerName[], string Upassword[], string Urole[], string File\_Name, int personCount)

{

string line;

fstream loginData;

loginData.open("LoginFile.txt", ios::out);

for (int i = 0; i < personCount; i++)

{

if(i==personCount-1)// last storage don't add new line

loginData << customerName[i] << "," << Upassword[i] << "," << Urole[i];

else

loginData << customerName[i] << "," << Upassword[i] << "," << Urole[i] << endl;

}

loginData.close();

}

void StoreReviewsFile(string reviews[], int reviewIndex)

{

string line;

fstream ReviewsData;

ReviewsData.open("Reviews.txt",ios::out);

for(int i=0;i<reviewIndex;i++)

{

if(i==reviewIndex-1)// last storage don't add new line

ReviewsData<<reviews[i];

else

ReviewsData<<reviews[i]<<endl;

}

ReviewsData.close();

}

void LoadReviews(string reviews[],string File\_Name, int &reviewIndex)

{

string line="";

fstream ReviewsData;

ReviewsData.open("Reviews.txt",ios::in);

if (ReviewsData.fail())

return;

while(!ReviewsData.eof())

{

getline(ReviewsData,line);

reviews[reviewIndex]=getField(line,0);

reviewIndex=reviewIndex+1;

}

ReviewsData.close();

}

void StoreComplaintsFile(string complaint[], int idx)

{

string line;

fstream ComplaintsData;

ComplaintsData.open("Complaints.txt",ios::out);

for(int i=0;i<idx;i++)

{

if(i==idx-1)// last storage don't add new line

ComplaintsData<<complaint[i];

else

ComplaintsData<<complaint[i]<<endl;

}

ComplaintsData.close();

}

void LoadComplaint(string complaint[],string File\_Name, int &idx)

{

string line="";

fstream ComplaintsData;

ComplaintsData.open("Complaints.txt",ios::in);

if (ComplaintsData.fail())

return;

while(!ComplaintsData.eof())

{

getline(ComplaintsData,line);

complaint[idx]=getField(line,0);

idx=idx+1;

}

ComplaintsData.close();

}

void StoreProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],int productsCount)

{

string line;

fstream ProductsData;

ProductsData.open("Products.txt",ios::out);

for(int i=0;i<productsCount;i++)

{

if(i==productsCount-1)// last storage don't add new line

ProductsData<<productName[i]<<","<<productColor[i]<<","<<to\_string(productPrice[i])<<","<<to\_string(productStock[i]);

else

ProductsData<<productName[i]<<","<<productColor[i]<<","<<to\_string(productPrice[i])<<","<<to\_string(productStock[i])<<endl;

}

ProductsData.close();

}

2void LoadProductsFile(string productName[],string productColor[],int productPrice[] ,int productStock[],string File\_Name,int &productsCount)

{

string line="";

fstream ProductsData;

ProductsData.open("Products.txt",ios::in);

if (ProductsData.fail())

return;

while(!ProductsData.eof())

{

getline(ProductsData,line);

productName[productsCount]=getField(line,0);

productColor[productsCount]=getField(line,1);

productPrice[productsCount]=stoi(getField(line,2));

productStock[productsCount]=stoi(getField(line,3));

productsCount=productsCount+1;

}

ProductsData.close();

}

# Weakness in the Business Application:

1. This application can be extended on a large scale with more effective functions and database.
2. Bit of functions are not SRF(Single Responsibility Function).
3. Bit of UI enhancement is required.

# Future Directions:

1. I will complete the possible functions in future.
2. I will beautify my code.
3. I will try to make it as much flexible as a code should be.