**Store Management System**



**Session 2023 - 2027**

**Submitted by:**

Muhammad Talha 2023-CS-12

**Supervised by:**

Dr. Awais Hassan & Mam Maida Mirza

**Course:**

CSC-102 Programming Fundamentals

**Department of Computer Science**

**University of Engineering and Technology**

**Lahore Pakistan**

* **Introduction:**

# Objective:

This app is an efficient and secure store management system that caters to the functionalities required by both administrator and regular users.

* + Contribution to CS:

This project contributes to the field of computer science by implementing a real-world application of a secure store management system.

* + **Output Expectations:**

# User-Friendly Interface

The console interface is designed to be easily understandable for users, they do not require any additional information before using this software.

# Security Measures

Ensuring the safety of user’s data is crucial. To enhance security, I have implemented password masking, making it unreadable for any unauthorized individuals.

# Database Management

Efficient functions and restrictions are implemented, ensuring data consistency of both administrator and users.

* **Users of Application:**

There are two types of users in this application. An Admin who have the authority over the application and the Client who can access his/her account for the use of application.

## Admin

The admin holds full authority over the application. They can add or remove items, view products records, change the prices, and even change the stock if necessary.

## Client/User

The client has the capability to register their account, enabling them to perform various functions such as viewing products, add to cart, view bill, or check prices of products. Additionally, they have the ability to view a bill of selected products. Moreover, Client can also change his password.

* **Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **User Type** | **Functions** | **Action Performed** |
| **Admin** | Add New Item | Add a new item in store |
| Remove an item | Remove an item in store |
| View the Available Products | View the Current list of products |
| Update Prices | Update the prices of products |
| Update Stocks | Update the stock of products |
| Check the Review of Products | Check the feedback about products |
| Add a Delivery Area | Add a new delivery area or can change |
| Remove Delivery Area | Remove a delivery area |
| Change Password | change the password of own account |
| Log Out | Log out from the account |
|  | | |
| **Client/User** | View Products | View the available products |
| Add an item to cart | Add the items to cart |
| View the cart | View the list of items added in the cart |
| Check Prices | Check the prices of products |
| Check the Stock | For desired quantity |
| Review Products | Give feedback to items |
| Check Delivery Area | Check the area where delivery is possible |
| Print Bill | View the bill of items |
| Pay Bill | Pay the Bill |
| Change Password | Used to Change Password |
| Log Out | Log out from the account |

* **Wireframes**

****

Figure main page

****

Figure Admin's menu

****

Figure user's menu

* **Data Structures:**
  + **For Storing User’s Info**

string NameDB[100];

string PasswordDB[100];

string RoleDB[100];

int usercount=0;

* + **For Stroring Products**

string Items[100];

int Price[100];

int Quantity[100];

int productCount;

* + **Extra variables that user uses:**

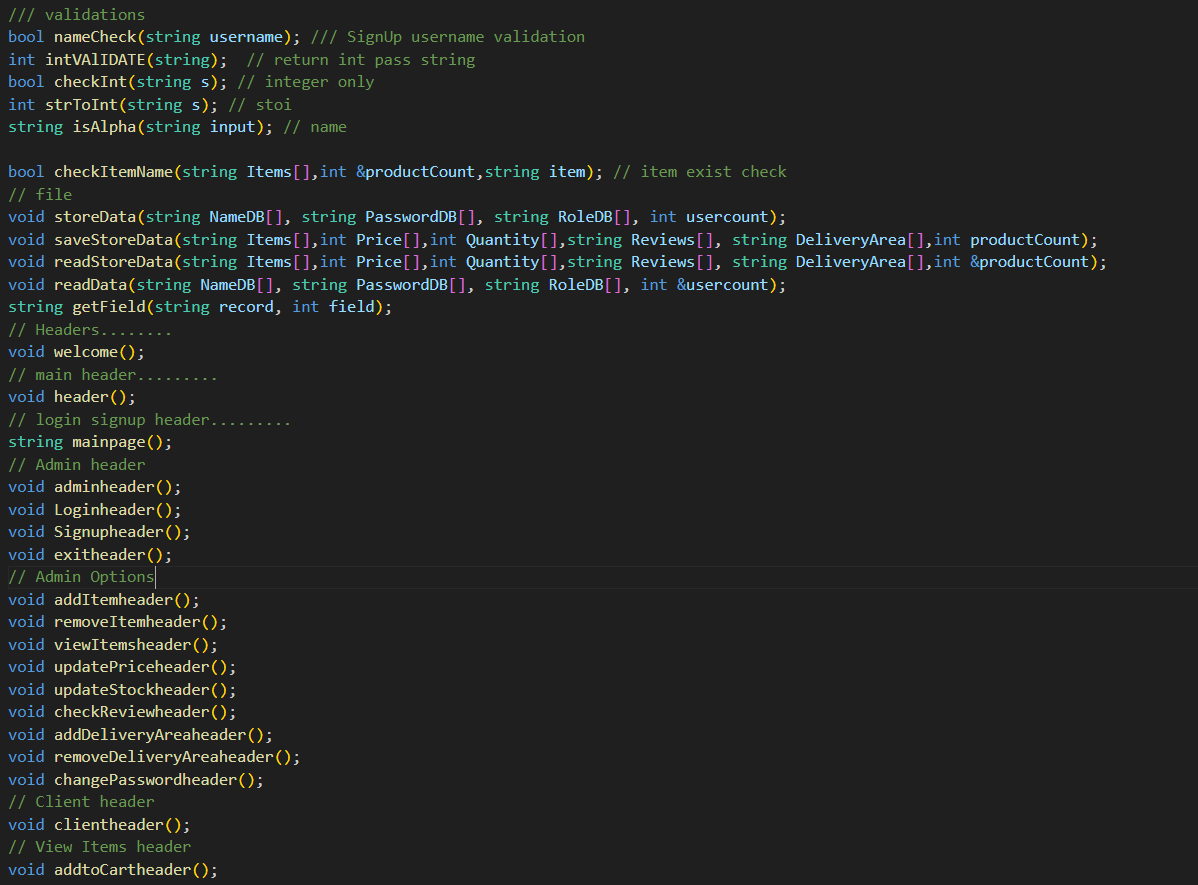
string Reviews[100];

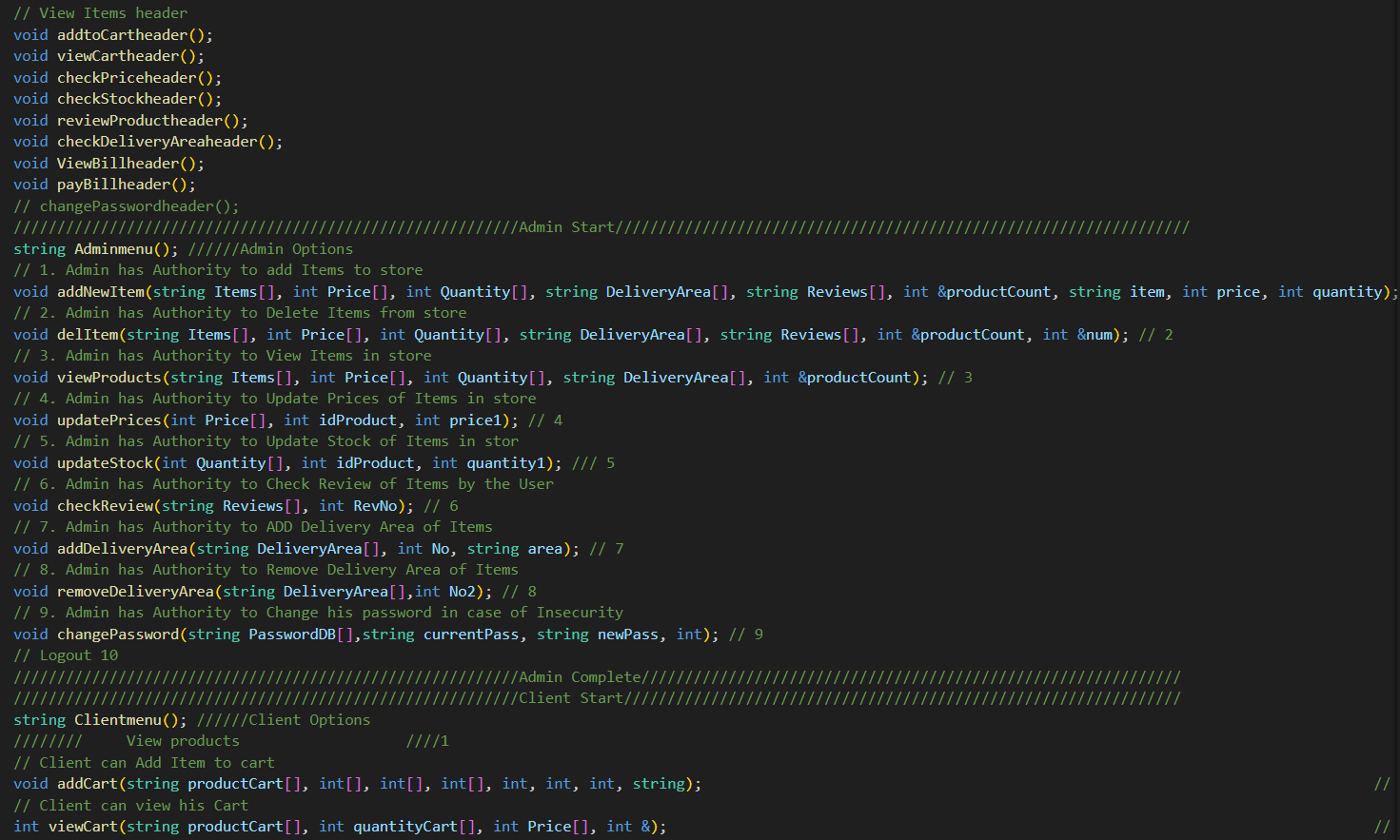
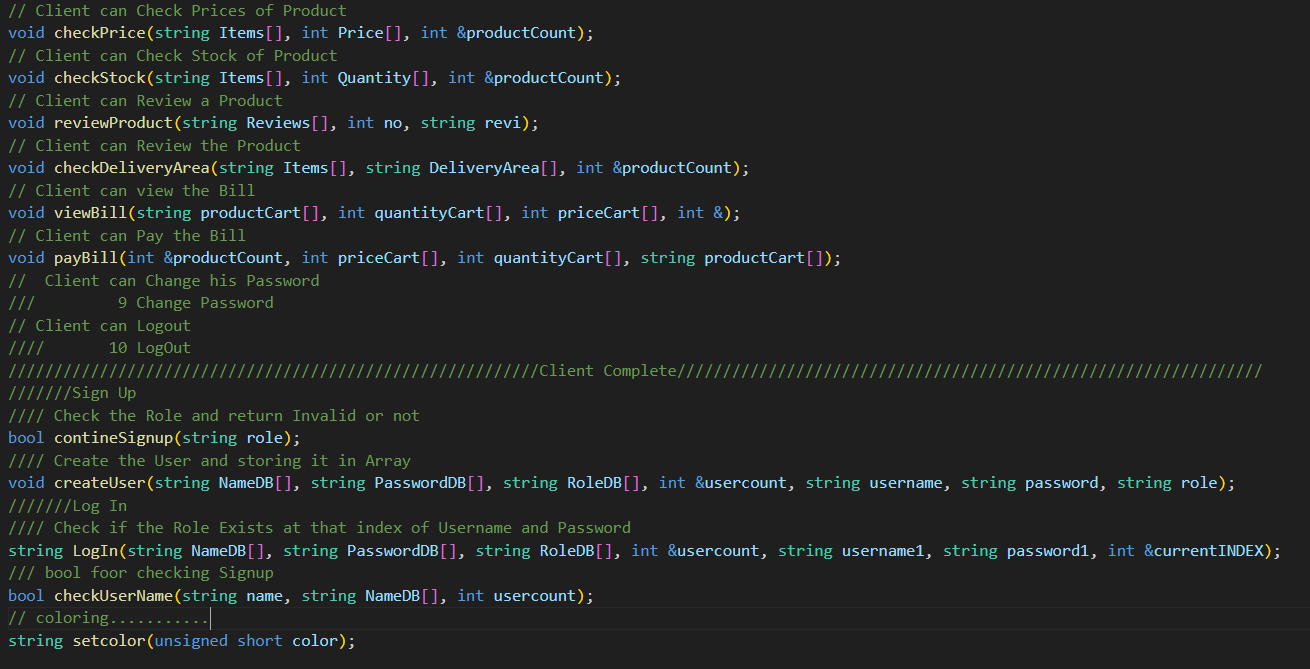
string DeliveryArea[100];

string productCart[100];

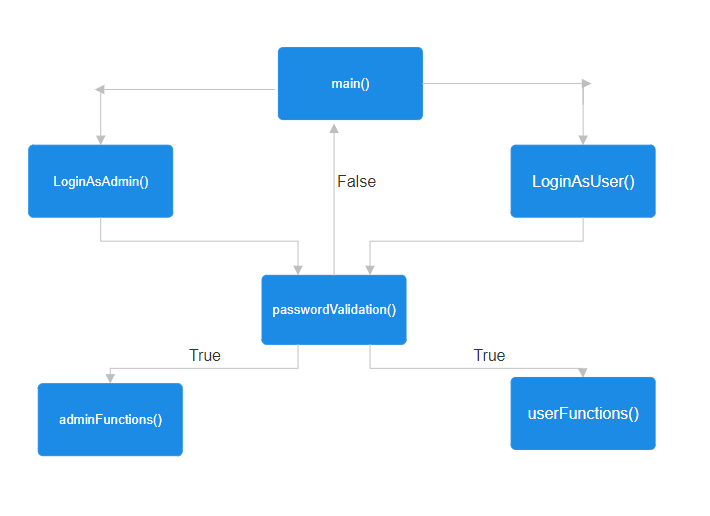
int quantityCart[100];

* + **Function Prototypes**





* **Functions Working Flow**



* **Complete Code of the Business Application**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <windows.h>

#include <string>

using namespace std;

/// validations

bool nameCheck(string username); /// SignUp username validation

int intVAlIDATE(string); // return int pass string

bool checkInt(string s); // integer only

int strToInt(string s); // stoi

string isAlpha(string input); // name

bool checkItemName(string Items[],int &productCount,string item); // item exist check

// file

void storeData(string NameDB[], string PasswordDB[], string RoleDB[], int usercount);

void saveStoreData(string Items[],int Price[],int Quantity[],string Reviews[], string DeliveryArea[],int productCount);

void readStoreData(string Items[],int Price[],int Quantity[],string Reviews[], string DeliveryArea[],int &productCount);

void readData(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount);

string getField(string record, int field);

// Headers........

void welcome();

// main header.........

void header();

// login signup header.........

string mainpage();

// Admin header

void adminheader();

void Loginheader();

void Signupheader();

void exitheader();

// Admin Options

void addItemheader();

void removeItemheader();

void viewItemsheader();

void updatePriceheader();

void updateStockheader();

void checkReviewheader();

void addDeliveryAreaheader();

void removeDeliveryAreaheader();

void changePasswordheader();

// Client header

void clientheader();

// View Items header

void addtoCartheader();

void viewCartheader();

void checkPriceheader();

void checkStockheader();

void reviewProductheader();

void checkDeliveryAreaheader();

void ViewBillheader();

void payBillheader();

// changePasswordheader();

//////////////////////////////////////////////////////////Admin Start//////////////////////////////////////////////////////////////////

string Adminmenu(); //////Admin Options

// 1. Admin has Authority to add Items to store

void addNewItem(string Items[], int Price[], int Quantity[], string DeliveryArea[], string Reviews[], int &productCount, string item, int price, int quantity); // 1

// 2. Admin has Authority to Delete Items from store

void delItem(string Items[], int Price[], int Quantity[], string DeliveryArea[], string Reviews[], int &productCount, int &num); // 2

// 3. Admin has Authority to View Items in store

void viewProducts(string Items[], int Price[], int Quantity[], string DeliveryArea[], int &productCount); // 3

// 4. Admin has Authority to Update Prices of Items in store

void updatePrices(int Price[], int idProduct, int price1); // 4

// 5. Admin has Authority to Update Stock of Items in stor

void updateStock(int Quantity[], int idProduct, int quantity1); /// 5

// 6. Admin has Authority to Check Review of Items by the User

void checkReview(string Reviews[], int RevNo); // 6

// 7. Admin has Authority to ADD Delivery Area of Items

void addDeliveryArea(string DeliveryArea[], int No, string area); // 7

// 8. Admin has Authority to Remove Delivery Area of Items

void removeDeliveryArea(string DeliveryArea[],int No2); // 8

// 9. Admin has Authority to Change his password in case of Insecurity

void changePassword(string PasswordDB[],string currentPass, string newPass, int); // 9

// Logout 10

//////////////////////////////////////////////////////////Admin Complete//////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////Client Start////////////////////////////////////////////////////////////////

string Clientmenu(); //////Client Options

//////// View products ////1

// Client can Add Item to cart

void addCart(string productCart[], int[], int[], int[], int, int, int, string); // 2

// Client can view his Cart

int viewCart(string productCart[], int quantityCart[], int Price[], int &); // 3

// Client can Check Prices of Product

void checkPrice(string Items[], int Price[], int &productCount); // 4

// Client can Check Stock of Product

void checkStock(string Items[], int Quantity[], int &productCount); // 5

// Client can Review a Product

void reviewProduct(string Reviews[], int no, string revi); // 6

// Client can Review the Product

void checkDeliveryArea(string Items[], string DeliveryArea[], int &productCount); // 7

// Client can view the Bill

void viewBill(string productCart[], int quantityCart[], int priceCart[], int &);

// Client can Pay the Bill

void payBill(int &productCount, int priceCart[], int quantityCart[], string productCart[]); // 8

// Client can Change his Password

/// 9 Change Password

// Client can Logout

//// 10 LogOut

//////////////////////////////////////////////////////////Client Complete////////////////////////////////////////////////////////////////

///////Sign Up

//// Check the Role and return Invalid or not

bool contineSignup(string role);

//// Create the User and storing it in Array

void createUser(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount, string username, string password, string role);

///////Log In

//// Check if the Role Exists at that index of Username and Password

string LogIn(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount, string username1, string password1, int &currentINDEX);

/// bool foor checking Signup

bool checkUserName(string name, string NameDB[], int usercount);

// coloring...........

string setcolor(unsigned short color);

/// COLOR SCHEMES

// 0: Black

// 1: Blue

// 2: Green

// 3: Cyan

// 4: Red

// 5: Magenta

// 6: Yellow/Brown

// 7: White

// 8: Gray

// 9: Bright Blue

// 10: Bright Green

// 11: Bright Cyan

// 12: Bright Red

// 13: Bright Magenta

// 14: Bright Yellow

// 15: Bright White

int main()

{

/////Variables.......

/// User Variables

string NameDB[100];

string PasswordDB[100];

string RoleDB[100];

int usercount = 0;

int currentINDEX = 0;

/// Admin || Client /// Variable

int productCount = 0;

string Items[100];

int Price[100];

int Quantity[100];

string Reviews[100];

string DeliveryArea[100];

string productCart[100];

int quantityCart[100];

int priceCart[100];

readData(NameDB, PasswordDB, RoleDB, usercount);

readStoreData(Items,Price,Quantity,Reviews,DeliveryArea,productCount);

// main

// option main.... for SignUp

string optionmain = "";

// loginRole.... for check login role

string loginRole;

/// Print Welcome Header/...........

welcome();

/// When Logout calls it comes here....

Logout:

while (optionmain != "3")

{

system("cls");

header();

/// Get what LOGIN SIGNUP EXIT RETURNS/////........

optionmain = mainpage();

if (optionmain == "3")

{

/// Exit

system("cls");

header();

cout << "\n\n";

exitheader();

Sleep (500);

storeData(NameDB, PasswordDB, RoleDB, usercount);

saveStoreData(Items,Price,Quantity,Reviews,DeliveryArea,productCount);

return 0;

}

else if (optionmain == "2")

/// Sign Up

{

Signupheader();

string username, password, role;

cout << "\t\t\t\t\t\t\t\t\tEnter User Name (Only Alphabets are Allowed): ";

username = isAlpha(username);

if (!checkUserName(username, NameDB, usercount))

{

Password:

cout << "\t\t\t\t\t\t\t\t\tEnter Password: ";

cin >> password;

if(password.length()>=6&&password.length()<=16)

{

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

if(password[i]==','){

cout<<"\t\t\t\t\t\t\tInvalid\n";

getch();

goto Password;

}

}

}

cout << "\t\t\t\t\t\t\t\t\tEnter Role (Admin or Client): ";

cin >> role;

if (contineSignup(role) == true)

{

createUser(NameDB, PasswordDB, RoleDB, usercount, username, password, role);

cout << "\t\t\t\t\t\t\t\t\tPress Any Key to Go Back to Main Page: ";

getch();

}

}

else

{

cout << "\t\t\t\t\t\t\t\t\tUser Already Exists\n";

cout << "\t\t\t\t\t\t\t\t\tPress Any Key to Continue...";

getch();

}

}

else if (optionmain == "1")

{

// Login

Loginheader();

string username1, password1, role1;

cout << "\t\t\t\t\t\t\t\t\tEnter Username: ";

cin >> username1;

cout << "\t\t\t\t\t\t\t\t\tEnter Password: ";

cin >> password1;

loginRole = LogIn(NameDB, PasswordDB, RoleDB, usercount, username1, password1, currentINDEX);

if (loginRole != "nill")

{

break;

}

else

{

cout << "\t\t\t\t\t\t\t\t\tInvalid Credentials...\n";

cout << "\t\t\t\t\t\t\t\t\tPress Any Key to Go Back";

getch();

}

}

else

{

cout <<"\t\t\t\t\t\t\t\t\tWrong Option..\n";

cout <<"\t\t\t\t\t\t\t\t\tPress Any Key to Continue.....";

getch();

}

}

// Outside Loop

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

{

// loginAdmin();

while (true)

{

// Get what Admin menu returns..

string adminOption = Adminmenu();

if (adminOption == "1")

{

system("cls");

addItemheader();

string item;

int price = 0,quantity = 0;

cout << "\t\t\t\t\t\tEnter name of the item: ";

item = isAlpha(item);

if (checkItemName(Items,productCount,item))

{

cout <<"\t\t\t\t\t\tProduct exists already\n";

cout <<"\t\t\t\t\t\tPress Any Key to Contine......";

getch();

}

else

{

string a = "\t\t\t\t\t\tEnter price of the item: ";

price = intVAlIDATE(a);

string b ="\t\t\t\t\t\tEnter the Quantity: ";

quantity = intVAlIDATE(b);

addNewItem(Items, Price, Quantity, DeliveryArea, Reviews, productCount, item, price, quantity);

cout << "\t\t\t\t\t\tPress Any Key to Continue...";

getch();

}

}

else if (adminOption == "2")

{

system("cls");

removeItemheader();

if (Items[0] == "")

{

cout << endl;

cout << "First add Items to Delete.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

int num;

string del = "\nEnter the item number you want to remove: ";

num = intVAlIDATE(del);

if ((num > productCount) || (num <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

delItem(Items, Price, Quantity, DeliveryArea, Reviews, productCount, num);

}

}

cout << "Press Any Key To Continue...";

getch();

}

else if (adminOption == "3")

{

system("cls");

viewItemsheader();

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to view the list.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

}

// viewProducts();

cout << "Press Any Key to Continue.....";

getch();

}

else if (adminOption == "4")

{

system("cls");

updatePriceheader();

int idProduct;

int price1;

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to update the price.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string upPrice = "Enter the No. of the Item whom you want to update price: ";

idProduct = intVAlIDATE(upPrice);

if ((idProduct > productCount) || (idProduct <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

string upPrice1 = "Enter the updated Price: ";

price1 = intVAlIDATE(upPrice1);

updatePrices(Price, idProduct, price1);

}

}

cout << "Press Any Key to Continue.....";

getch();

// updatePrices()

}

else if (adminOption == "5")

{

system("cls");

updateStockheader();

int idProduct;

int quantity1;

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to update the stock.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string upStock = "Enter the No. of the Item whom you want to update Stock: ";

idProduct = intVAlIDATE(upStock);

if ((idProduct > productCount) || (idProduct <= 0))

{

cout << "\nYou Entered an Incorrect Number.";

Sleep(1500);

}

else

{

string upStock1 = "Enter the updated Stock: ";

quantity1 = intVAlIDATE(upStock1);

updateStock(Quantity, idProduct, quantity1);

}

}

cout << "Press Any Key to Continue.....";

getch();

// updateStock();

}

else if (adminOption == "6")

{

system("cls");

checkReviewheader();

int RevNo;

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to Check Review.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas" << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string chkReview = "Enter the No. of Product whom you want to Check Review: ";

RevNo = intVAlIDATE(chkReview);

if ((RevNo > productCount) || (RevNo <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

cout << endl;

checkReview(Reviews, RevNo);

}

}

cout << "\nPress Any Key To Continue...";

getch();

// check Review

}

else if (adminOption == "7")

{

system("cls");

addDeliveryAreaheader();

int No;

string area;

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to Add or Change Delivery Areas.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string upDeliveryA = "Enter the No. of the Product whom you want to Add Delivery Area: ";

No = intVAlIDATE(upDeliveryA);

if ((No > productCount) || (No <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

cout << "Enter the Area: ";

area = isAlpha(area);

addDeliveryArea( DeliveryArea, No, area);

}

}

cout << "Press Any Key to Continue...";

getch();

}

else if (adminOption == "8")

{

system("cls");

removeDeliveryAreaheader();

int No2;

if (Items[0] == "")

{

cout << endl;

cout << "Add Items to Remove Delivery Areas.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string remDeliveryA = "Enter the No. of Item whom you want to remove Delivery Area: ";

No2 = intVAlIDATE(remDeliveryA);

if ((No2 > productCount) || (No2 <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

removeDeliveryArea(DeliveryArea, No2);

}

}

cout << "\n\t\t\t\t\t\tPress Any Key to Continue..";

getch();

}

else if (adminOption == "9")

{

system("cls");

changePasswordheader();

string currentPass;

string newPass;

cout << "\t\t\t\t\t\tEnter current Password: ";

cin >> currentPass;

changePassword(PasswordDB, currentPass, newPass, currentINDEX);

cout << "\n\t\t\t\t\t\tPress Any Key to Continue..";

getch();

// changePassword();

}

else if (adminOption == "10")

{

goto Logout;

}

else

{

cout << "\t\t\t\t\t\tError...\n";

cout << "\t\t\t\t\t\tPlease Enter Correct Option....";

Sleep(1000);

}

}

}

if (loginRole == "Client" || loginRole == "client")

{

// loginClient();

while (true)

{

////get what Client Menu Returns

string clientOption = Clientmenu();

if (clientOption == "1")

{

system("cls");

viewItemsheader();

if (Items[0] == "")

{

cout << endl;

cout << "All the Products are Sold Out.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

}

// viewProducts();

cout << "Press Any Key to Continue.....";

getch();

}

else if (clientOption == "2")

{

system("cls");

addtoCartheader();

int n2;

int added = 0;

int prc = 0;

if (Items[0] == "")

{

cout << "\t\t\t\t\t\tNo Items in the Store. We are sorry.\n";

cout << "\t\t\t\t\t\tPress Any Key to Continue....";

getch();

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string buy = "Enter the item No you want to buy: ";

n2 = intVAlIDATE(buy);

if ((n2 > productCount) || (n2 <= 0))

{

cout << "You Entered an Incorrect Number.\n";

Sleep(1500);

}

else

{

string name3 = "";

string entQuantity = "Enter the Quantity :";

added = intVAlIDATE(entQuantity);

if (added > Quantity[n2 - 1])

{

cout << "Sorry, we do not have enough stock.\n";

Sleep(500);

}

else

{

name3 = Items[n2 - 1];

prc = Price[n2 - 1] \* added;

addCart(productCart, Quantity, quantityCart, priceCart, n2, added, prc, name3);

}

}

}

cout << "Press Any Key To Continue...";

getch();

}

else if (clientOption == "3")

{

system("cls");

viewCartheader();

viewCart(productCart, quantityCart, priceCart, productCount);

cout << "\t\t\t\t\t\tPress Any Key to Continue....";

getch();

}

else if (clientOption == "4")

{

system("cls");

checkPriceheader();

if (Items[0] == "")

{

cout << "\t\t\t\t\t\tNo Items in the Store. We are sorry.\n";

}

else

{

cout << "\t\t\t\t\t\tProduct Name\t\tPrice\n";

checkPrice(Items, Price, productCount);

}

cout << "\t\t\t\t\t\tPress Any Key To Continue...";

getch();

}

else if (clientOption == "5")

{

system("cls");

checkStockheader();

if (Items[0] == "")

{

cout << "\t\t\t\t\t\tNo Items in the Store. We are sorry.\n";

}

else

{

cout << "\t\t\t\t\t\tProduct Name\t\tQuantity\n";

checkStock(Items, Quantity, productCount);

}

cout << "\t\t\t\t\t\tPress Any Key To Continue...";

getch();

}

else if (clientOption == "6")

{

system("cls");

reviewProductheader();

int no;

string revi;

if (Items[0] == "")

{

cout << "\t\t\t\t\t\tNo Items in the Store. We are sorry.\n";

}

else

{

cout << "No. Products Name\t\tPrice\t\tQuantity\t\tDelivery Areas " << endl;

viewProducts(Items, Price, Quantity, DeliveryArea, productCount);

string noReview = "Enter the No of product you want to review: ";

no = intVAlIDATE(noReview);

if ((no > productCount) || (no <= 0))

{

cout << "You Entered an Incorrect Number.";

Sleep(1500);

}

else

{

cout << "Enter Review: ";

cin.clear();

cin.sync();

getline(cin, revi);

reviewProduct(Reviews, no, revi);

}

}

cout << "\t\t\t\t\t\tPress Any Key To Continue...";

getch();

}

else if (clientOption == "7")

{

system("cls");

checkDeliveryAreaheader();

if (Items[0] == "")

{

cout << "\t\t\t\t\t\tNo Items in the Store. We are sorry.\n";

}

else

{

cout << "\t\t\t\t\t\tProduct Name\t\tDeliveryArea\n";

checkDeliveryArea(Items, DeliveryArea, productCount);

}

cout << "\t\t\t\t\t\tPress Any Key To Continue...";

getch();

}

else if (clientOption == "8")

{

system("cls");

ViewBillheader();

viewBill(productCart, quantityCart, priceCart, productCount);

cout << "\t\t\t\t\t\tPress Any Key to Continue..";

getch();

}

else if (clientOption == "9")

{

system("cls");

payBillheader();

payBill(productCount, priceCart, quantityCart, productCart);

}

else if (clientOption == "10")

{

system("cls");

changePasswordheader();

string currentPass;

string newPass;

cout << "\t\t\t\t\t\tEnter current Password: ";

cin >> currentPass;

int idx;

changePassword(PasswordDB, currentPass, newPass, currentINDEX);

cout << "\n\t\t\t\t\t\tPress Any Key to Continue..";

getch();

}

else if (clientOption == "11")

{

goto Logout;

}

else

{

cout << "\t\t\t\t\t\tError... \n";

cout << "\t\t\t\t\t\tPlease Enter Correct Option....";

Sleep(1000);

}

}

}

}

// Login Functions.....

string LogIn(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount, string username1, string password1, int &currentINDEX)

{

string Role = "nill";

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

{

if (NameDB[i] == username1 && PasswordDB[i] == password1)

{

Role = RoleDB[i];

currentINDEX = i;

}

}

return Role;

}

////Login Menu........

/// Admin Menu........

string Adminmenu()

{

system("cls");

adminheader();

setcolor(10);

string op1 = "";

cout << endl;

cout << endl;

cout << "\t\t\t\t\t\t\t1. Add an Item" << endl;

cout << "\t\t\t\t\t\t\t2. Delete an Item" << endl;

cout << "\t\t\t\t\t\t\t3. View Items" << endl;

cout << "\t\t\t\t\t\t\t4. Update Prices" << endl;

cout << "\t\t\t\t\t\t\t5. Update Stock" << endl;

cout << "\t\t\t\t\t\t\t6. Check Reviews" << endl;

cout << "\t\t\t\t\t\t\t7. Add Delivery Areas" << endl;

cout << "\t\t\t\t\t\t\t8. Remove Delivery Areas" << endl;

cout << "\t\t\t\t\t\t\t9. Change Password" << endl;

cout << "\t\t\t\t\t\t\t10. Log Out" << endl;

cout << endl;

cout << "\t\t\t\t\t\t\tPlease Select an Option...";

cin >> op1;

setcolor(15);

return op1;

}

//////Client Menu..................

string Clientmenu()

{

system("cls");

clientheader();

setcolor(15);

string op2 = "";

cout << endl

<< endl;

cout << endl;

cout << "\t\t\t\t\t\t1. View the list of Available Items" << endl;

cout << "\t\t\t\t\t\t2. Add an item to Cart" << endl;

cout << "\t\t\t\t\t\t3. View cart" << endl;

cout << "\t\t\t\t\t\t4. Check Prices" << endl;

cout << "\t\t\t\t\t\t5. Check Stock" << endl;

cout << "\t\t\t\t\t\t6. Review a Product" << endl;

cout << "\t\t\t\t\t\t7. Check Delivery Areas" << endl;

cout << "\t\t\t\t\t\t8. Print Bill" << endl;

cout << "\t\t\t\t\t\t9. Pay Bill" << endl;

cout << "\t\t\t\t\t\t10. Change Password" << endl;

cout << "\t\t\t\t\t\t11. Log Out" << endl;

cout << endl;

cout << "\t\t\t\t\t\tPlease Select an Option...";

cin >> op2;

setcolor(15);

return op2;

}

/// Admin Functions..........

void addNewItem(string Items[], int Price[], int Quantity[], string DeliveryArea[], string Reviews[], int &productCount, string item, int price, int quantity) // 1

{

Items[productCount] = item;

Price[productCount] = price;

Quantity[productCount] = quantity;

Reviews[productCount] = "N/A";

DeliveryArea[productCount] = "N/A";

productCount++;

}

void delItem(string Items[], int Price[], int Quantity[], string DeliveryArea[], string Reviews[], int &productCount, int &num) // 2

{

for (int i = num - 1; i < productCount; i++)

{

Items[i] = Items[i + 1];

Quantity[i] = Quantity[i + 1];

Price[i] = Price[i + 1];

DeliveryArea[i] = DeliveryArea[i + 1];

Reviews[i] = Reviews[i + 1];

}

productCount--;

}

void viewProducts(string Items[], int Price[], int Quantity[], string DeliveryArea[], int &productCount) // 3

{

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

{

if (Items[i] != "")

cout << i + 1 << " " << Items[i] << "\t\t\t" << Price[i] << "\t\t" << Quantity[i] << "\t\t\t" << DeliveryArea[i] << endl;

}

}

void updatePrices(int Price[], int idProduct, int price1) // 4

{

Price[idProduct - 1] = price1;

}

void updateStock(int Quantity[], int idProduct, int quantity1) // 5

{

Quantity[idProduct - 1] = quantity1;

}

void checkReview(string Reviews[],int RevNo) // 6

{

cout << "\t\t\t\t\t\t" << Reviews[RevNo - 1];

}

void addDeliveryArea(string DeliveryArea[], int No,string area) // 7

{

for (int i = 0; i < No; i++)

{

DeliveryArea[No - 1] = area;

// productCount ++;

}

}

void removeDeliveryArea(string DeliveryArea[], int No2) // 8

{

DeliveryArea[No2 - 1] = "N/A";

}

void changePassword(string PasswordDB[],string currentPass, string newPass, int currentINDEX) // 9

{

if (currentPass == PasswordDB[currentINDEX])

{

cout << "\t\t\t\t\t\tEnter New Password: ";

cin >> newPass;

PasswordDB[currentINDEX] = newPass;

cout << "\t\t\t\t\t\tPassword Changed Successfully...";

}

else

cout << "\t\t\t\t\t\tCurrent Password is Not Correct....";

}

/////Client

void addCart(string productCart[], int Quantity[], int quantityCart[], int priceCart[], int n2, int added, int prc, string name3)

{

productCart[n2 - 1] = name3;

quantityCart[n2 - 1] = added;

priceCart[n2 - 1] = prc;

Quantity[n2 - 1] -= added;

}

int viewCart(string productCart[], int quantityCart[], int priceCart[], int &productCount)

{

int bill = 0;

bool iscartEmpty = false;

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

{

if (quantityCart[i] != 0)

{

iscartEmpty = false;

break;

}

iscartEmpty = true;

}

if (iscartEmpty)

{

cout << "\t\t\t\t\t\tThe Cart is Empty..." << endl;

}

else

{

cout << "\t\t\t\t\t\tProduct Name\t\tQuantity\t\tPrice\n";

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

{

if (quantityCart[i] != 0)

{

cout << "\t\t\t\t\t\t" << productCart[i] << "\t\t\t" << quantityCart[i] << "\t\t\t" << priceCart[i] << "\n";

bill = bill + priceCart[i];

}

}

}

return bill;

}

void checkPrice(string Items[], int Price[], int &productCount)

{

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

{

cout << "\t\t\t\t\t\t" << Items[i] << "\t\t\t" << Price[i] << "\n";

}

}

void checkStock(string Items[], int Quantity[], int &productCount)

{

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

{

cout << "\t\t\t\t\t\t" << Items[i] << "\t\t\t" << Quantity[i] << "\n";

}

}

void reviewProduct(string Reviews[], int no, string revi)

{

for (int i = 0; i < no; i++)

{

Reviews[no - 1] = revi;

}

}

void checkDeliveryArea(string Items[], string DeliveryArea[], int &productCount)

{

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

{

cout << "\t\t\t\t\t\t" << Items[i] << "\t\t\t" << DeliveryArea[i] << "\n";

}

}

void viewBill(string productCart[], int quantityCart[], int priceCart[], int &productCount)

{

int payableBill = viewCart(productCart, quantityCart, priceCart, productCount);

cout << "\n\t\t\t\t\t\t##################################################\n"

<< endl;

cout << "\t\t\t\t\t\tYour total Bill is: " << payableBill;

cout << endl;

}

void payBill(int &productCount, int priceCart[], int quantityCart[], string productCart[])

{

int payableBill = viewCart(productCart, quantityCart, priceCart, productCount);

cout << "\n\t\t\t\t\t\t##################################################\n"

<< endl;

cout << "\t\t\t\t\t\tYour total Bill is: " << payableBill << "\n";

cout << "\t\t\t\t\t\tPress Any Key to Pay Bill.....";

getch();

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

{

quantityCart[i] = 0;

}

cout << "\n\t\t\t\t\t\tBill paid.....";

getch();

}

/// Sign Up Functions.........

bool contineSignup(string role)

{

bool signUp = true;

if (!(role == "Admin" || role == "admin" || role == "Client" || role == "client"))

{

cout << "\t\t\t\t\t\tInvalid Role\n";

signUp = false;

cout << "\t\t\t\t\t\tPress Any Key to Go Back to Main Page: ";

getch();

}

return signUp;

}

void createUser(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount, string username, string password, string role)

{

NameDB[usercount] = username;

PasswordDB[usercount] = password;

RoleDB[usercount] = role;

usercount++;

}

bool checkUserName(string name, string NameDB[], int usercount)

{

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

{

if (name == NameDB[i])

return true;

}

return false;

}

// Header

void header()

{

setcolor(14);

cout <<" "<<endl;

cout <<" /$$$$$$ /$$$$$$$$ /$$$$$$ /$$$$$$$ /$$$$$$$$ /$$ /$$ /$$$$$$ /$$ /$$ /$$$$$$ /$$$$$$ /$$$$$$$$ /$$ /$$ /$$$$$$$$ /$$ /$$ /$$$$$$$$ /$$$$$$ /$$ /$$ /$$$$$$ /$$$$$$$$ /$$$$$$$$ /$$ /$$ "<<endl;

cout <<" /$$\_\_ $$|\_\_ $$\_\_//$$\_\_ $$| $$\_\_ $$| $$\_\_\_\_\_/ | $$$ /$$$ /$$\_\_ $$| $$$ | $$ /$$\_\_ $$ /$$\_\_ $$| $$\_\_\_\_\_/| $$$ /$$$| $$\_\_\_\_\_/| $$$ | $$|\_\_ $$\_\_/ /$$\_\_ $$| $$ /$$//$$\_\_ $$|\_\_ $$\_\_/| $$\_\_\_\_\_/| $$$ /$$$ "<<endl;

cout <<"| $$ \\\_\_/ | $$ | $$ \\ $$| $$ \\ $$| $$ | $$$$ /$$$$| $$ \\ $$| $$$$| $$| $$ \\ $$| $$ \\\_\_/| $$ | $$$$ /$$$$| $$ | $$$$| $$ | $$ | $$ \\\_\_/ \\ $$ /$$/| $$ \\\_\_/ | $$ | $$ | $$$$ /$$$$ "<<endl;

cout <<"| $$$$$$ | $$ | $$ | $$| $$$$$$$/| $$$$$ | $$ $$/$$ $$| $$$$$$$$| $$ $$ $$| $$$$$$$$| $$ /$$$$| $$$$$ | $$ $$/$$ $$| $$$$$ | $$ $$ $$ | $$ | $$$$$$ \\ $$$$/ | $$$$$$ | $$ | $$$$$ | $$ $$/$$ $$ "<<endl;

cout <<" \\\_\_\_\_ $$ | $$ | $$ | $$| $$\_\_ $$| $$\_\_/ | $$ $$$| $$| $$\_\_ $$| $$ $$$$| $$\_\_ $$| $$|\_ $$| $$\_\_/ | $$ $$$| $$| $$\_\_/ | $$ $$$$ | $$ \\\_\_\_\_ $$ \\ $$/ \\\_\_\_\_ $$ | $$ | $$\_\_/ | $$ $$$| $$ "<<endl;

cout <<" /$$ \\ $$ | $$ | $$ | $$| $$ \\ $$| $$ | $$\\ $ | $$| $$ | $$| $$\\ $$$| $$ | $$| $$ \\ $$| $$ | $$\\ $ | $$| $$ | $$\\ $$$ | $$ /$$ \\ $$ | $$ /$$ \\ $$ | $$ | $$ | $$\\ $ | $$ "<<endl;

cout <<"| $$$$$$/ | $$ | $$$$$$/| $$ | $$| $$$$$$$$ | $$ \\/ | $$| $$ | $$| $$ \\ $$| $$ | $$| $$$$$$/| $$$$$$$$| $$ \\/ | $$| $$$$$$$$| $$ \\ $$ | $$ | $$$$$$/ | $$ | $$$$$$/ | $$ | $$$$$$$$| $$ \\/ | $$ "<<endl;

cout <<" \\\_\_\_\_\_\_/ |\_\_/ \\\_\_\_\_\_\_/ |\_\_/ |\_\_/|\_\_\_\_\_\_\_\_/ |\_\_/ |\_\_/|\_\_/ |\_\_/|\_\_/ \\\_\_/|\_\_/ |\_\_/ \\\_\_\_\_\_\_/ |\_\_\_\_\_\_\_\_/|\_\_/ |\_\_/|\_\_\_\_\_\_\_\_/|\_\_/ \\\_\_/ |\_\_/ \\\_\_\_\_\_\_/ |\_\_/ \\\_\_\_\_\_\_/ |\_\_/ |\_\_\_\_\_\_\_\_/|\_\_/ |\_\_/ "<<endl;

cout <<" "<<endl;

cout << endl

<< endl;

cout << endl

<< endl;

setcolor(15);

}

// MainPage header

string mainpage()

{

setcolor(11);

cout << "\t\t\t\t\t\t\t\t\t \_ \_ \_\_\_ \_\_\_\_ \_\_\_ \_ \_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t / | | | / \_ \\ / \_\_\_|\_ \_| \\ | | " << endl;

cout << "\t\t\t\t\t\t\t\t\t | | | | | | | | | \_ | || \\| | " << endl;

cout << "\t\t\t\t\t\t\t\t\t | |\_ | |\_\_| |\_| | |\_| || || |\\ | " << endl;

cout << "\t\t\t\t\t\t\t\t\t |\_(\_) |\_\_\_\_\_\\\_\_\_/ \\\_\_\_\_|\_\_\_|\_| \\\_| " << endl;

cout << endl;

cout << "\t\t\t\t\t\t\t\t\t \_\_\_\_ \_\_\_\_ \_ \_ \_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t|\_\_\_ \\ / \_\_\_|(\_) \_\_ \_ \_ \_\_ | | | |\_ \_\_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t \_\_) | \\\_\_\_ \\| |/ \_` | '\_ \\ | | | | '\_ \\ " << endl;

cout << "\t\t\t\t\t\t\t\t\t / \_\_/ \_ \_\_\_) | | (\_| | | | | | |\_| | |\_) |" << endl;

cout << "\t\t\t\t\t\t\t\t\t|\_\_\_\_\_(\_) |\_\_\_\_/|\_|\\\_\_, |\_| |\_| \\\_\_\_/| .\_\_/ " << endl;

cout << "\t\t\t\t\t\t\t\t\t |\_\_\_/ |\_| " << endl;

cout << endl;

cout << "\t\t\t\t\t\t\t\t\t \_\_\_\_\_ \_\_\_\_\_ \_ \_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t|\_\_\_ / | \_\_\_\_|\_ \_(\_) |\_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t |\_ \\ | \_| \\ \\/ / | \_\_| " << endl;

cout << "\t\t\t\t\t\t\t\t\t \_\_\_) | | |\_\_\_ > <| | |\_ " << endl;

cout << "\t\t\t\t\t\t\t\t\t|\_\_\_\_(\_) |\_\_\_\_\_/\_/\\\_\\\_|\\\_\_| " << endl;

cout << "\t\t\t\t\t\t\t\t\t " << endl;

cout << endl

<< endl;

cout << "\t\t\t\t\t\t\t\t\tEnter your Option: ";

string option;

cin >> option;

return option;

setcolor(15);

}

void adminheader()

{

setcolor(13);

cout <<"\t\t\t\t\t /$$$$$$ /$$$$$$$ /$$ /$$ /$$$$$$ /$$ /$$ "<<endl;

cout <<"\t\t\t\t\t /$$\_\_ $$| $$\_\_ $$| $$$ /$$$|\_ $$\_/| $$$ | $$ "<<endl;

cout <<"\t\t\t\t\t| $$ \\ $$| $$ \\ $$| $$$$ /$$$$ | $$ | $$$$| $$ "<<endl;

cout <<"\t\t\t\t\t| $$$$$$$$| $$ | $$| $$ $$/$$ $$ | $$ | $$ $$ $$ "<<endl;

cout <<"\t\t\t\t\t| $$\_\_ $$| $$ | $$| $$ $$$| $$ | $$ | $$ $$$$ "<<endl;

cout <<"\t\t\t\t\t| $$ | $$| $$ | $$| $$\\ $ | $$ | $$ | $$\\ $$$ "<<endl;

cout <<"\t\t\t\t\t| $$ | $$| $$$$$$$/| $$ \\/ | $$ /$$$$$$| $$\\ $$ "<<endl;

cout <<"\t\t\t\t\t|\_\_/ |\_\_/|\_\_\_\_\_\_\_/ |\_\_/ |\_\_/|\_\_\_\_\_\_/|\_\_/ \\\_\_/ "<<endl;

cout << endl << endl;

setcolor(15);

}

void clientheader()

{

setcolor(14);

cout <<"\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t /$$$$$$ /$$ /$$$$$$ /$$$$$$$$ /$$ /$$ /$$$$$$$$ "<<endl;

cout <<"\t\t\t\t\t /$$\_\_ $$| $$ |\_ $$\_/| $$\_\_\_\_\_/| $$$ | $$|\_\_ $$\_\_/ "<<endl;

cout <<"\t\t\t\t\t| $$ \\\_\_/| $$ | $$ | $$ | $$$$| $$ | $$ "<<endl;

cout <<"\t\t\t\t\t| $$ | $$ | $$ | $$$$$ | $$ $$ $$ | $$ "<<endl;

cout <<"\t\t\t\t\t| $$ | $$ | $$ | $$\_\_/ | $$ $$$$ | $$ "<<endl;

cout <<"\t\t\t\t\t| $$ $$| $$ | $$ | $$ | $$\\ $$$ | $$ "<<endl;

cout <<"\t\t\t\t\t| $$$$$$/| $$$$$$$$ /$$$$$$| $$$$$$$$| $$ \\ $$ | $$ "<<endl;

cout <<"\t\t\t\t\t \\\_\_\_\_\_\_/ |\_\_\_\_\_\_\_\_/|\_\_\_\_\_\_/|\_\_\_\_\_\_\_\_/|\_\_/ \\\_\_/ |\_\_/ "<<endl;

cout <<"\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t "<<endl;

setcolor(15);

}

void Loginheader()

{

system("cls");

setcolor(14);

cout <<"\t\t\t\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t\t\t\t /$$ /$$$$$$ /$$$$$$ /$$$$$$ /$$ /$$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ /$$\_\_ $$ /$$\_\_ $$|\_ $$\_/| $$$ | $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ | $$ \\ $$| $$ \\\_\_/ | $$ | $$$$| $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ | $$ | $$| $$ /$$$$ | $$ | $$ $$ $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ | $$ | $$| $$|\_ $$ | $$ | $$ $$$$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ | $$ | $$| $$ \\ $$ | $$ | $$\\ $$$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$$$$$$$| $$$$$$/| $$$$$$/ /$$$$$$| $$ \\ $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t|\_\_\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ |\_\_\_\_\_\_/|\_\_/ \\\_\_/"<<endl;

cout <<"\t\t\t\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void Signupheader()

{

system("cls");

setcolor(14);

cout <<"\t\t\t\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t\t\t\t /$$$$$$ /$$$$$$ /$$$$$$ /$$ /$$ /$$ /$$ /$$$$$$$ "<<endl;

cout <<"\t\t\t\t\t\t\t\t /$$\_\_ $$|\_ $$\_/ /$$\_\_ $$| $$$ | $$ | $$ | $$| $$\_\_ $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$ \\\_\_/ | $$ | $$ \\\_\_/| $$$$| $$ | $$ | $$| $$ \\ $$"<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$$$$$ | $$ | $$ /$$$$| $$ $$ $$ | $$ | $$| $$$$$$$/"<<endl;

cout <<"\t\t\t\t\t\t\t\t \\\_\_\_\_ $$ | $$ | $$|\_ $$| $$ $$$$ | $$ | $$| $$\_\_\_\_/ "<<endl;

cout <<"\t\t\t\t\t\t\t\t /$$ \\ $$ | $$ | $$ \\ $$| $$\\ $$$ | $$ | $$| $$ "<<endl;

cout <<"\t\t\t\t\t\t\t\t| $$$$$$/ /$$$$$$| $$$$$$/| $$ \\ $$ | $$$$$$/| $$ "<<endl;

cout <<"\t\t\t\t\t\t\t\t \\\_\_\_\_\_\_/ |\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ |\_\_/ \\\_\_/ \\\_\_\_\_\_\_/ |\_\_/ "<<endl;

cout <<"\t\t\t\t\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void storeData(string NameDB[], string PasswordDB[], string RoleDB[], int usercount)

{

fstream storeFILE;

storeFILE.open("Users.txt", ios::out);

for (int x = 0; x < usercount; x++)

{

if (x == usercount - 1)

storeFILE << NameDB[x] << "," << PasswordDB[x] << "," << RoleDB[x];

else

storeFILE << NameDB[x] << "," << PasswordDB[x] << "," << RoleDB[x] << "\n";

}

storeFILE.close();

}

void saveStoreData(string Items[],int Price[],int Quantity[],string Reviews[], string DeliveryArea[], int productCount)

{

fstream saveDatafile;

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

for (int x = 0; x < productCount; x++)

{

if (x == productCount - 1)

saveDatafile << Items[x] << "," << Price[x] << "," << Quantity[x] << "," << Reviews[x] << "," << DeliveryArea[x];

else

saveDatafile << Items[x] << "," << Price[x] << "," << Quantity[x] << "," << Reviews[x] << "," << DeliveryArea[x] <<"\n";

}

saveDatafile.close();

}

void welcome()

{

setcolor(12);

system("cls");

cout <<" "<<endl;

cout <<" "<<endl;

cout <<"WWWWWWWW WWWWWWWWEEEEEEEEEEEEEEEEEEEEEELLLLLLLLLLL CCCCCCCCCCCCC OOOOOOOOO MMMMMMMM MMMMMMMMEEEEEEEEEEEEEEEEEEEEEE "<<endl;

cout <<"W::::::W W::::::WE::::::::::::::::::::EL:::::::::L CCC::::::::::::C OO:::::::::OO M:::::::M M:::::::ME::::::::::::::::::::E "<<endl;

cout <<"W::::::W W::::::WE::::::::::::::::::::EL:::::::::L CC:::::::::::::::C OO:::::::::::::OO M::::::::M M::::::::ME::::::::::::::::::::E "<<endl;

cout <<"W::::::W W::::::WEE::::::EEEEEEEEE::::ELL:::::::LL C:::::CCCCCCCC::::CO:::::::OOO:::::::OM:::::::::M M:::::::::MEE::::::EEEEEEEEE::::E "<<endl;

cout <<" W:::::W WWWWW W:::::W E:::::E EEEEEE L:::::L C:::::C CCCCCCO::::::O O::::::OM::::::::::M M::::::::::M E:::::E EEEEEE "<<endl;

cout <<" W:::::W W:::::W W:::::W E:::::E L:::::L C:::::C O:::::O O:::::OM:::::::::::M M:::::::::::M E:::::E "<<endl;

cout <<" W:::::W W:::::::W W:::::W E::::::EEEEEEEEEE L:::::L C:::::C O:::::O O:::::OM:::::::M::::M M::::M:::::::M E::::::EEEEEEEEEE "<<endl;

cout <<" W:::::W W:::::::::W W:::::W E:::::::::::::::E L:::::L C:::::C O:::::O O:::::OM::::::M M::::M M::::M M::::::M E:::::::::::::::E "<<endl;

cout <<" W:::::W W:::::W:::::W W:::::W E:::::::::::::::E L:::::L C:::::C O:::::O O:::::OM::::::M M::::M::::M M::::::M E:::::::::::::::E "<<endl;

cout <<" W:::::W W:::::W W:::::W W:::::W E::::::EEEEEEEEEE L:::::L C:::::C O:::::O O:::::OM::::::M M:::::::M M::::::M E::::::EEEEEEEEEE "<<endl;

cout <<" W:::::W:::::W W:::::W:::::W E:::::E L:::::L C:::::C O:::::O O:::::OM::::::M M:::::M M::::::M E:::::E "<<endl;

cout <<" W:::::::::W W:::::::::W E:::::E EEEEEE L:::::L LLLLLLC:::::C CCCCCCO::::::O O::::::OM::::::M MMMMM M::::::M E:::::E EEEEEE "<<endl;

cout <<" W:::::::W W:::::::W EE::::::EEEEEEEE:::::ELL:::::::LLLLLLLLL:::::L C:::::CCCCCCCC::::CO:::::::OOO:::::::OM::::::M M::::::MEE::::::EEEEEEEE:::::E "<<endl;

cout <<" W:::::W W:::::W E::::::::::::::::::::EL::::::::::::::::::::::L CC:::::::::::::::C OO:::::::::::::OO M::::::M M::::::ME::::::::::::::::::::E "<<endl;

cout <<" W:::W W:::W E::::::::::::::::::::EL::::::::::::::::::::::L CCC::::::::::::C OO:::::::::OO M::::::M M::::::ME::::::::::::::::::::E "<<endl;

cout <<" WWW WWW EEEEEEEEEEEEEEEEEEEEEELLLLLLLLLLLLLLLLLLLLLLLL CCCCCCCCCCCCC OOOOOOOOO MMMMMMMM MMMMMMMMEEEEEEEEEEEEEEEEEEEEEE "<<endl;

cout <<" "<<endl;

cout <<" "<<endl;

cout << endl

<< endl;

cout << "\t\t\t\t\tPress Any Key To Continue...";

getch();

setcolor(15);

}

void exitheader()

{

setcolor(10);

cout <<"\t\t\t "<<endl;

cout <<"\t\t\t \_\_\_\_\_ \_ \_ \_\_ \_\_ \_\_\_ \_ \_ \_ \_ \_ "<<endl;

cout <<"\t\t\t|\_ \_| |\_ \_\_ \_ \_ \_ | |\_\_ \_\_\_ / \_|\_\_\_ \_ \_ \\ \\ / (\_)\_\_(\_) |\_ \_ \_ \_\_ \_ | | | |\_\_| |"<<endl;

cout <<"\t\t\t | | | ' \\/ \_` | ' \\| / /(\_-< | \_/ \_ \\ '\_| \\ V /| (\_-< | \_| ' \\/ \_` | | |\_| (\_-<\_|"<<endl;

cout <<"\t\t\t |\_| |\_||\_\\\_\_,\_|\_||\_|\_\\\_\\/\_\_/ |\_| \\\_\_\_/\_| \\\_/ |\_/\_\_/\_|\\\_\_|\_||\_\\\_\_, | \\\_\_\_//\_\_(\_)"<<endl;

cout <<"\t\t\t |\_\_\_/ "<<endl;

cout << endl <<endl;

setcolor(15);

}

void addItemheader()

{

setcolor(14);

cout <<"\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t \_ \_\_\_ \_\_\_ \_\_\_ \_\_\_\_\_ \_\_\_ \_\_ \_\_ "<<endl;

cout <<"\t\t\t\t\t /\_\\ | \\| \\ |\_ \_|\_ \_| \_\_| \\/ |"<<endl;

cout <<"\t\t\t\t\t / \_ \\| |) | |) | | | | | | \_|| |\\/| |"<<endl;

cout <<"\t\t\t\t\t/\_/ \\\_\\\_\_\_/|\_\_\_/ |\_\_\_| |\_| |\_\_\_|\_| |\_|"<<endl;

cout <<"\t\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void removeItemheader()

{

setcolor(14);

cout <<"\t\t\t\t "<<endl;

cout <<"\t\t\t\t \_\_\_ \_ \_ \_ \_ "<<endl;

cout <<"\t\t\t\t| \\ \_\_\_| |\_\_\_| |\_ \_\_\_ (\_) |\_ \_\_\_ \_ \_\_ "<<endl;

cout <<"\t\t\t\t| |) / -\_) / -\_) \_/ -\_) | | \_/ -\_) ' \\ "<<endl;

cout <<"\t\t\t\t|\_\_\_/\\\_\_\_|\_\\\_\_\_|\\\_\_\\\_\_\_| |\_|\\\_\_\\\_\_\_|\_|\_|\_| "<<endl;

cout <<"\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void viewItemsheader()

{

setcolor(14);

cout <<"\t\t\t\t "<<endl;

cout <<"\t\t\t\t\_\_ \_\_\_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t\t\\ \\ / (\_)\_\_\_\_\_ \_\_ \_\_ |\_ \_| |\_ \_\_\_ \_ \_\_ \_\_\_ "<<endl;

cout <<"\t\t\t\t \\ V /| / -\_) V V / | || \_/ -\_) ' \\(\_-< "<<endl;

cout <<"\t\t\t\t \\\_/ |\_\\\_\_\_|\\\_/\\\_/ |\_\_\_|\\\_\_\\\_\_\_|\_|\_|\_/\_\_/ "<<endl;

cout <<"\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void updatePriceheader()

{

setcolor(14);

cout <<"\t\t\t\t \_ \_ \_ \_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t\t| | | |\_ \_\_ \_\_| |\_\_ \_| |\_ \_\_\_ | \_ \\\_ \_(\_)\_\_ \_\_\_ "<<endl;

cout <<"\t\t\t\t| |\_| | '\_ \\/ \_` / \_` | \_/ -\_) | \_/ '\_| / \_/ -\_)"<<endl;

cout <<"\t\t\t\t \\\_\_\_/| .\_\_/\\\_\_,\_\\\_\_,\_|\\\_\_\\\_\_\_| |\_| |\_| |\_\\\_\_\\\_\_\_|"<<endl;

cout <<"\t\t\t\t |\_| "<<endl;

cout <<"\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void updateStockheader()

{

setcolor(14);

cout <<"\t\t\t\t \_ \_ \_ \_ \_\_\_ \_ \_ "<<endl;

cout <<"\t\t\t\t| | | |\_ \_\_ \_\_| |\_\_ \_| |\_ \_\_\_ / \_\_| |\_ \_\_\_ \_\_| |\_\_ "<<endl;

cout <<"\t\t\t\t| |\_| | '\_ \\/ \_` / \_` | \_/ -\_) \\\_\_ \\ \_/ \_ \\/ \_| / / "<<endl;

cout <<"\t\t\t\t \\\_\_\_/| .\_\_/\\\_\_,\_\\\_\_,\_|\\\_\_\\\_\_\_| |\_\_\_/\\\_\_\\\_\_\_/\\\_\_|\_\\\_\\"<<endl;

cout <<"\t\t\t\t |\_| "<<endl;

cout <<"\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void checkReviewheader()

{

setcolor(14);

cout<<"\t\t\t\t "<<endl;

cout<<"\t\t\t\t \_\_\_ \_ \_ \_\_\_ \_ "<<endl;

cout<<"\t\t\t\t / \_\_| |\_ \_\_\_ \_\_| |\_\_ | \_ \\\_\_\_\_\_ \_(\_)\_\_\_\_\_ \_\_ \_\_ "<<endl;

cout<<"\t\t\t\t| (\_\_| ' \\/ -\_) \_| / / | / -\_) V / / -\_) V V / "<<endl;

cout<<"\t\t\t\t \\\_\_\_|\_||\_\\\_\_\_\\\_\_|\_\\\_\\ |\_|\_\\\_\_\_|\\\_/|\_\\\_\_\_|\\\_/\\\_/ "<<endl;

cout<<"\t\t\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void addDeliveryAreaheader()

{

setcolor(14);

cout <<"\t\t "<<endl;

cout <<"\t\t \_ \_ \_ \_\_\_ \_ \_\_\_ \_ \_ \_ "<<endl;

cout <<"\t\t /\_\\ \_\_| |\_\_| | \_\_\_ \_ \_ / \_\_| |\_ \_\_ \_ \_ \_ \_\_ \_ \_\_\_ | \\ \_\_\_| (\_)\_ \_\_\_\_\_ \_ \_ \_ \_ /\_\\ \_ \_ \_\_\_ \_\_ \_ "<<endl;

cout <<"\t\t / \_ \\/ \_` / \_` | / \_ \\ '\_| | (\_\_| ' \\/ \_` | ' \\/ \_` / -\_) | |) / -\_) | \\ V / -\_) '\_| || | / \_ \\| '\_/ -\_) \_` |"<<endl;

cout <<"\t\t/\_/ \\\_\\\_\_,\_\\\_\_,\_| \\\_\_\_/\_| \\\_\_\_|\_||\_\\\_\_,\_|\_||\_\\\_\_, \\\_\_\_| |\_\_\_/\\\_\_\_|\_|\_|\\\_/\\\_\_\_|\_| \\\_, | /\_/ \\\_\\\_| \\\_\_\_\\\_\_,\_|"<<endl;

cout <<"\t\t |\_\_\_/ |\_\_/ "<<endl;

cout << endl << endl;

setcolor(15);

}

void removeDeliveryAreaheader()

{

setcolor(14);

cout <<"\t\t \_\_\_ \_\_\_ \_ \_ \_ "<<endl;

cout <<"\t\t| \_ \\\_\_\_ \_ \_\_ \_\_\_\_\_ \_\_\_\_\_ | \\ \_\_\_| (\_)\_ \_\_\_\_\_ \_ \_ \_ \_ /\_\\ \_ \_ \_\_\_ \_\_ \_ "<<endl;

cout <<"\t\t| / -\_) ' \\/ \_ \\ V / -\_) | |) / -\_) | \\ V / -\_) '\_| || | / \_ \\| '\_/ -\_) \_` |"<<endl;

cout <<"\t\t|\_|\_\\\_\_\_|\_|\_|\_\\\_\_\_/\\\_/\\\_\_\_| |\_\_\_/\\\_\_\_|\_|\_|\\\_/\\\_\_\_|\_| \\\_, | /\_/ \\\_\\\_| \\\_\_\_\\\_\_,\_|"<<endl;

cout <<"\t\t |\_\_/ "<<endl;

cout <<"\t\t "<<endl;

cout << endl << endl;

setcolor(15);

}

void changePasswordheader()

{

setcolor(14);

cout <<"\t\t\t \_\_\_ \_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t / \_\_| |\_ \_\_ \_ \_ \_ \_\_ \_ \_\_\_ | \_ \\\_\_ \_ \_\_\_\_\_\_\_ \_\_ \_\_\_\_\_ \_ \_ \_\_| |"<<endl;

cout <<"\t\t\t| (\_\_| ' \\/ \_` | ' \\/ \_` / -\_) | \_/ \_` (\_-<\_-< V V / \_ \\ '\_/ \_` |"<<endl;

cout <<"\t\t\t \\\_\_\_|\_||\_\\\_\_,\_|\_||\_\\\_\_, \\\_\_\_| |\_| \\\_\_,\_/\_\_/\_\_/\\\_/\\\_/\\\_\_\_/\_| \\\_\_,\_|"<<endl;

cout <<"\t\t\t |\_\_\_/ "<<endl;

cout << endl << endl;

setcolor(15);

}

void addtoCartheader()

{

setcolor(14);

cout <<"\t\t\t\t \_ \_ \_ \_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t\t /\_\\ \_\_| |\_\_| | | |\_ \_\_\_ / \_\_|\_\_ \_ \_ \_| |\_ "<<endl;

cout <<"\t\t\t\t / \_ \\/ \_` / \_` | | \_/ \_ \\ | (\_\_/ \_` | '\_| \_|"<<endl;

cout <<"\t\t\t\t/\_/ \\\_\\\_\_,\_\\\_\_,\_| \\\_\_\\\_\_\_/ \\\_\_\_\\\_\_,\_|\_| \\\_\_|"<<endl;

cout <<"\t\t\t\t "<<endl;

setcolor(15);

}

void viewCartheader()

{

setcolor(14);

cout <<"\t\t\t\t "<<endl;

cout <<"\t\t\t\t\_\_ \_\_\_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t\t\\ \\ / (\_)\_\_\_\_\_ \_\_ \_\_ / \_\_|\_\_ \_ \_ \_| |\_ "<<endl;

cout <<"\t\t\t\t \\ V /| / -\_) V V / | (\_\_/ \_` | '\_| \_|"<<endl;

cout <<"\t\t\t\t \\\_/ |\_\\\_\_\_|\\\_/\\\_/ \\\_\_\_\\\_\_,\_|\_| \\\_\_|"<<endl;

cout <<"\t\t\t\t "<<endl;

setcolor(15);

}

void checkPriceheader()

{

setcolor(14);

cout <<"\t\t\t\t \_\_\_ \_ \_ \_\_\_ \_ "<<endl;

cout <<"\t\t\t\t / \_\_| |\_ \_\_\_ \_\_| |\_\_ | \_ \\\_ \_(\_)\_\_ \_\_\_ \_\_\_"<<endl;

cout <<"\t\t\t\t| (\_\_| ' \\/ -\_) \_| / / | \_/ '\_| / \_/ -\_|\_-<"<<endl;

cout <<"\t\t\t\t \\\_\_\_|\_||\_\\\_\_\_\\\_\_|\_\\\_\\ |\_| |\_| |\_\\\_\_\\\_\_\_/\_\_/"<<endl;

cout <<"\t\t\t\t "<<endl;

setcolor(15);

}

void checkStockheader()

{

setcolor(14);

cout <<"\t\t\t\t \_\_\_ \_ \_ \_\_\_ \_ \_ "<<endl;

cout <<"\t\t\t\t / \_\_| |\_ \_\_\_ \_\_| |\_\_ / \_\_| |\_ \_\_\_ \_\_| |\_\_"<<endl;

cout <<"\t\t\t\t| (\_\_| ' \\/ -\_) \_| / / \\\_\_ \\ \_/ \_ \\/ \_| / /"<<endl;

cout <<"\t\t\t\t \\\_\_\_|\_||\_\\\_\_\_\\\_\_|\_\\\_\\ |\_\_\_/\\\_\_\\\_\_\_/\\\_\_|\_\\\_\\"<<endl;

cout <<"\t\t\t\t "<<endl;

setcolor(15);

}

void reviewProductheader()

{

setcolor(14);

cout <<"\t\t\t\t \_\_\_ \_ \_\_\_ \_ \_ "<<endl;

cout <<"\t\t\t\t| \_ \\\_\_\_\_\_ \_(\_)\_\_\_\_\_ \_\_ \_\_ | \_ \\\_ \_ \_\_\_ \_\_| |\_ \_ \_\_| |\_ "<<endl;

cout <<"\t\t\t\t| / -\_) V / / -\_) V V / | \_/ '\_/ \_ \\/ \_` | || / \_| \_|"<<endl;

cout <<"\t\t\t\t|\_|\_\\\_\_\_|\\\_/|\_\\\_\_\_|\\\_/\\\_/ |\_| |\_| \\\_\_\_/\\\_\_,\_|\\\_,\_\\\_\_|\\\_\_|"<<endl;

cout <<"\t\t\t\t "<<endl;

setcolor(15);

}

void checkDeliveryAreaheader()

{

setcolor(14);

cout <<"\t\t \_\_\_ \_ \_ \_\_\_ \_ \_ \_ "<<endl;

cout <<"\t\t / \_\_| |\_ \_\_\_ \_\_| |\_\_ | \\ \_\_\_| (\_)\_ \_\_\_\_\_ \_ \_ \_ \_ /\_\\ \_ \_ \_\_\_ \_\_ \_ \_\_\_"<<endl;

cout <<"\t\t| (\_\_| ' \\/ -\_) \_| / / | |) / -\_) | \\ V / -\_) '\_| || | / \_ \\| '\_/ -\_) \_` (\_-<"<<endl;

cout <<"\t\t \\\_\_\_|\_||\_\\\_\_\_\\\_\_|\_\\\_\\ |\_\_\_/\\\_\_\_|\_|\_|\\\_/\\\_\_\_|\_| \\\_, | /\_/ \\\_\\\_| \\\_\_\_\\\_\_,\_/\_\_/"<<endl;

cout <<"\t\t |\_\_/ "<<endl;

setcolor(15);

}

void ViewBillheader()

{

setcolor(14);

cout <<"\t\t\t\t\t\_\_ \_\_\_ \_\_\_ \_ \_ \_ "<<endl;

cout <<"\t\t\t\t\t\\ \\ / (\_)\_\_\_\_\_ \_\_ \_\_ | \_ |\_) | |"<<endl;

cout <<"\t\t\t\t\t \\ V /| / -\_) V V / | \_ \\ | | |"<<endl;

cout <<"\t\t\t\t\t \\\_/ |\_\\\_\_\_|\\\_/\\\_/ |\_\_\_/\_|\_|\_|"<<endl;

cout <<"\t\t\t\t\t "<<endl;

setcolor(15);

}

void payBillheader()

{

setcolor(14);

cout <<"\t\t\t\t\t "<<endl;

cout <<"\t\t\t\t\t \_\_\_ \_\_\_ \_ \_ \_ "<<endl;

cout <<"\t\t\t\t\t | \_ \\\_\_ \_ \_ \_ | \_ |\_) | |"<<endl;

cout <<"\t\t\t\t\t | \_/ \_` | || | | \_ \\ | | |"<<endl;

cout <<"\t\t\t\t\t |\_| \\\_\_,\_|\\\_, | |\_\_\_/\_|\_|\_|"<<endl;

cout <<"\t\t\t\t\t |\_\_/ "<<endl;

cout <<" "<<endl;

setcolor(15);

}

string setcolor(unsigned short color)

{

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

SetConsoleTextAttribute(hcon, color);

return "";

}

void readStoreData(string Items[],int Price[],int Quantity[],string Reviews[], string DeliveryArea[],int &productCount)

{

string record = "";

fstream readStoreFILE;

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

if (readStoreFILE.fail())

return;

while (!readStoreFILE.eof())

{

getline(readStoreFILE, record);

Items[productCount] = getField(record, 1);

Price[productCount] = strToInt(getField(record, 2));

Quantity[productCount] = strToInt(getField(record, 3));

Reviews[productCount] = getField(record, 4);

DeliveryArea[productCount] = getField(record, 5);

productCount++;

}

readStoreFILE.close();

}

void readData(string NameDB[], string PasswordDB[], string RoleDB[], int &usercount)

{

string record = "";

fstream readFILE;

readFILE.open("Users.txt", ios::in);

if (readFILE.fail())

return;

while (!readFILE.eof())

{

getline(readFILE, record);

NameDB[usercount] = getField(record, 1);

PasswordDB[usercount] = getField(record, 2);

RoleDB[usercount] = getField(record, 3);

usercount++;

}

readFILE.close();

}

// getfield

string getField(string record, int field)

{

int commaCount = 1;

string result = "";

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

{

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

{

commaCount = commaCount + 1;

}

else if (commaCount == field)

{

result = result + record[x];

}

else if (commaCount > field)

{

break;

}

}

return result;

}

/// Validations

bool nameCheck(string username) /// username when signup

{

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

{

if (!((username[i] >= 'A' && username[i] <= 'Z') || (username[i] >= 'a' && username[i] <= 'z')))

{

return false;

}

}

return true;

}

int intVAlIDATE(string prompt)

{

string num;

while (true)

{

cout << prompt;

cin >> num;

if (checkInt(num))

{

return strToInt(num);

}

else

{

cout << "Invalid Input." << endl;

cout << "Press any key to try again..................." << endl;

getch();

}

}

}

bool checkInt(string s)

{

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

{

if (s[i] < '0' || s[i] > '9')

{

return false;

}

}

return true;

}

int strToInt(string s)

{

int result = 0;

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

{

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

}

return result;

}

string isAlpha(string input) // name and city validation

{

cin.clear();

cin.sync();

getline(cin >> ws, input);

int size;

int check;

bool flap;

while (true)

{

size = input.length();

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

{

check = int(input[i]);

if (((check >= 65 && check <= 90) || (check >= 97 && check <= 122) || input[i] == ' ') && input[i] != ',')

{

flap = true;

}

else

{

flap = false;

break;

}

}

if (flap == true)

{

return input;

}

else

{

cin.clear();

cin.sync();

cout << "\t\t\tWrong input..." << endl;

cout << "\t\t\tEnter Again: ";

getline(cin >> ws, input);

}

}

}

bool checkItemName(string Items[],int &productCount,string item)

{

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

{

if (Items[i] == item)

{

return true;

}

}

return false;

}

* **Weakness in the Business Application**
  + **Limited Error Handling:**

The code lacks comprehensive error handling. For example, if the user enters string data in place of integer type data, the program might not handle it properly, and crash.

* **Comments:**

Some functions may be lacking comments, making it challenging to understand their purpose and functionality.

* **Future Directions**

I envision this application having a Graphical User Interface (GUI) to enhance user convenience compared to a console-based interface. Furthermore, integrating internet connectivity will allow the application to retrieve real-time information from the web, thus enhancing its intelligence and ensuring users have the most up-to-date data at their fingertips.