# ZAHRA FABRICS

A logo of a university of engineering and technology

Description automatically generated

# Session 2023 – 2027

# Submitted by:

Muhammad Hassaan Shahid 2023-CS-79

# Supervised by:

Dr. Muhammad Awais Hassan

# Course:

CSC-102 Programming Fundamentals

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

# TITLE

# Zahra Fabrics

# The "Zahra Fabrics" application serves as an online clothing shop, designed to facilitate a seamless shopping experience for two primary user types: the Customer (or Client) and the Admin. The application's overarching goal is to enable users to browse, select, and purchase various clothing items via a digital interface. This implementation of online shopping significantly contributes to the field of computer science by utilizing technology to streamline the process of buying clothing through a screen, emphasizing convenience and accessibility.

# Users of Application

**Customer (Client):**

* Objective: Access a wide array of available clothing items for both men and women.
* Functionalities:
* View a tabular list displaying diverse clothing items.
* Add preferred clothing items to their selection.
* Access the shopping cart to review purchased items.
* View the total amount reflecting the selected items.
* Remove unwanted items from the shopping cart.
* Explore available discounts through a dedicated menu.
* Access various payment options via a designated menu.
* Choose from available delivery areas for shipping.
* Log out from the application after completing tasks.

**Admin:**

**Objective:** Control and manage the application's functionalities and inventory.

**Functionalities:**

* View a tabular list displaying all available clothing items.
* Modify prices of listed items as necessary.
* Manage available stock by adjusting quantities.
* Review and possibly respond to user ratings (ranging from 1 to 5) for each item.
* Add or delete items from the clothing menu as needed.
* Update discount percentages for promotional purposes.
* Modify payment options and associated prices.
* Adjust available delivery areas as per requirements.
* Log out from the administrative control panel.

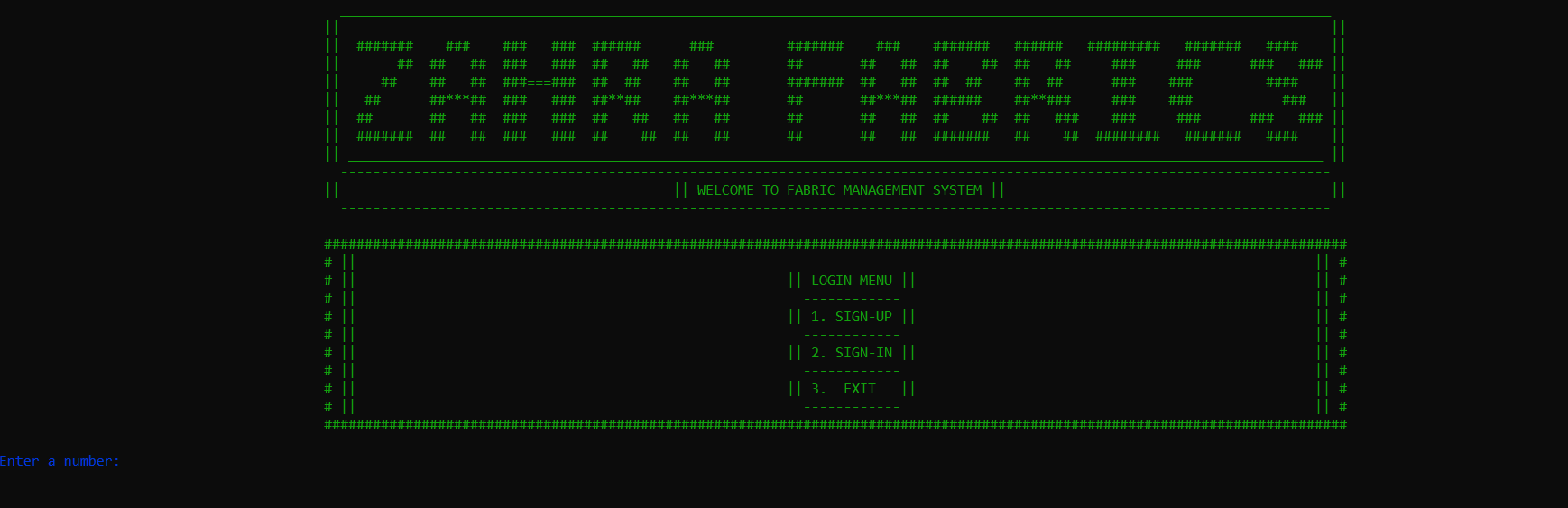
# TABLE

# Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | User Type | Required Function | Result of Action Performed |
| 1 | Admin | View List of Clothes | A tabular list of clothes appears |
| Change prices | Change prices of items |
| Change Available Stock | Increment or decrement in the quantity available of items |
| Check Reviews | Review from 1 to 5 for each item appears |
| Add or delete an item | Addition or Deletion in items of clothing menu |
| Update Discounts | Changing in discount percentages appear |
| Update Payment Options | It will update the payment options and prices |
| Add or Remove Delivery Areas | Changing in available delivery areas |
| Log out | It will log the user out |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | User Type | Required Function | Result of Action Performed |
| 2 | Customer | View List of Clothing | A tabular list of clothes appears |
| Add an item | Addition in item of clothing |
| View cart | List of purchased items appears |
| View result | Total amount appears |
| Delete an item of clothing | Deletion in cart items |
| View Discounts | Discount menu appears |
| View payment options | Payment Option Menu appears |
| Select Delivery Area | List of Delivery Areas appear |
| Log out | It will log the user out |

# Wireframes



**Figure 1: Login Screen**



**Figure 2: Options For Admin**



**Figure 3: Options for User**

* **Functions Working Flow:**
* **Weakness in the Business Application:**

1. Less designing in code like color.
2. Single responsibilities are not meet.
3. File handling problem

* **Future Directions**

1. Improving myself by learning more tricks.

**Variables:** string names[50]; // user name string password[50]; // password string userClass[50]; // classtype string product\_Names[maxProducts]; string product\_Categories[maxProducts]; int reviews[maxProducts]; int product\_Quantities[maxProducts]; int product\_Prices[maxProducts]; string product\_Qualities[maxProducts]; int cart[maxProducts]; string array[maxMonths]; string onths[maxMonths]; int profits[maxMonths]; string productsremaining[maxMonths]; string productsolds[maxMonths]; string pricesbeforesale[maxMonths]; string quantitysold[maxMonths]; string pricesaftersale[maxMonths]; string worker\_names[maxWorkers]; string orker\_codes[maxWorkers]; string worker\_ranks[maxWorkers]; string orker\_performances[maxWorkers]; string worker\_hours[maxWorkers]; string worker\_sellings[maxWorkers]; string addressDetails[addressSize];

**Prototype:**

void addProductDetails(int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);int readUserDataFromFile(string names[], string passwords[], string userClass[], int& indexCount);bool numbers(string integer);bool strings(string word);void viewProducts(int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);void addWorkerDetails(int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[]);void profit(int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);string get\_username();string get\_pass();void signup(string names[],string password[],string userClass[],int indexCount);void AddPaymentMethod(int numMethods );string get\_userType();int user(int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods,int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);int optionsForAdmin(int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods,int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);void viewWorkersDetails(int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);int optionsForUser(int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods,int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);int search\_Array(string array[],string object, int indexCount);void displayMonthlyReport(int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[]);void mostProfitableMonth(int profits[],int maxMonths,string months[]);void overallProfitLoss(int profits[],int maxMonths);void login(int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods,int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);void logout(int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods,int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);void viewReviews(int newReview,int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);void iewPaymentMethods(int numMethods,string methods);void addReview(int newReview,int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]);string viewDeliveryAddress(string addressDetails[]);string inputDeliveryAddress();void viewCart(int cart[],string product\_Names[],int maxProducts,int product\_Prices[]);void addToCart(int maxProducts,int cart[]);void applyDiscount(int product\_Prices[],string voucher,float discountPercentage,int maxProducts);void addDiscountCode();void viewDiscountCode(float discountPercentage,string voucher);void view();string setcolor(int color);string getField(string record, int field);int blue = 1, green = 2, cyan = 3, red = 4, brown = 6, lightwhite = 7, lightblue = 9, lightgreen = 10, lightcyan = 11, lightred = 12, yellow = 14, white = 15;void logoSignUp();void logoSignIn();void viewAdmin();void viewUser();string getField(string get,int field);

**Code:**

int user(string userAddress[],int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods[],int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[])

{

int number = 0;

loadDataFromUserFile("user\_data.txt",names,password,userClass);

loadProductDetails( maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

loadWorkerDetails( maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings);

loadProfitDetails( maxMonths, months, profits, pricesbeforesale, pricesaftersale, product\_Quantities);

loadPaymentMethods( numMethods, methods);

loadDeliveryAddress();

loadCart(maxProducts, cart);

loadDiscountInfo( voucher, discountPercentage);

string input;

while (number != 3) {

system("cls"); // clear previos screen

setcolor(green);

Sleep(100);cout << "\t\t\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << endl;

Sleep(100);cout << "\t\t\t\t\t|| ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ####### ### ### ### ###### ### ####### ### ####### ###### ######### ####### #### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ## ## ## ### ### ## ## ## ## ## ## ## ## ## ## ## ### ### ### ### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ## ## ## ###===### ## ## ## ## ####### ## ## ## ## ## ## ### ### #### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ## ##\*\*\*## ### ### ##\*\*## ##\*\*\*## ## ##\*\*\*## ###### ##\*\*### ### ### ### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ## ## ## ### ### ## ## ## ## ## ## ## ## ## ## ### ### ### ### ### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| ####### ## ## ### ### ## ## ## ## ## ## ## ####### ## ## ######## ####### #### ||" << endl;

Sleep(100);cout << "\t\t\t\t\t|| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ||" << endl;

Sleep(100);cout << "\t\t\t\t\t -------------------------------------------------------------------------------------------------------------------------- " << endl;

Sleep(100);cout << "\t\t\t\t\t|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

Sleep(100);cout << "\t\t\t\t\t -------------------------------------------------------------------------------------------------------------------------- " << endl << endl;

Sleep(100);

Sleep(70);cout << "\t\t\t\t\t##############################################################################################################################" << endl;

Sleep(70);cout << "\t\t\t\t\t# || ------------ || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || || LOGIN MENU || || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || ------------ || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || || 1. SIGN-UP || || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || ------------ || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || || 2. SIGN-IN || || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || ------------ || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || || 3. EXIT || || #" << endl;

Sleep(70);cout << "\t\t\t\t\t# || ------------ || #" << endl;

Sleep(70);cout << "\t\t\t\t\t##############################################################################################################################" << endl <<endl;

Sleep(100);

setcolor(white);

setcolor(blue);

cout << "Enter a number: ";

getline(cin, input);

if (input.empty() || !numbers(input)) {

setcolor(red);

cout << "Invalid input! Please enter a valid number." << endl;

setcolor(white);

Sleep(400);

setcolor(white);

system("cls");

logo();

continue;

}

number = stoi(input); //conversion

if (number == 3) {

setcolor(blue);

cout << "Thanks for visiting..." << endl;

Sleep(400);

setcolor(white);

break;

} else if (number == 1) {

system("cls");

logoSignUp();

signup(names, password,userClass,indexCount); // for signup

} else if (number == 2) {

system("cls");

logoSignIn();

// for sign in

login( userAddress, newReview, addressSize, addressDetails,names, password, userClass, voucher, discountPercentage, numMethods, methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

} else {

cout << "Invalid option! Please enter a valid number." << endl;

Sleep(400);

setcolor(white);

system("cls");

logo();

}

}

return -1;

}

void signup(string names[],string password[],string userClass[],int indexCount)

{

setcolor(blue);

const int maxUsers = 50;

if (indexCount < maxUsers) {

names[indexCount] = get\_username();

password[indexCount] = get\_pass();

userClass[indexCount] = get\_userType();

indexCount++;

// Writing to a file

ofstream outFile("user\_data.txt", ios::app); // Open file in append mode

if (outFile.is\_open()) {

outFile << names[indexCount - 1] << "," << password[indexCount - 1] << "," << userClass[indexCount - 1] << endl;

outFile.close();

} else {

cout << "Unable to open file for writing!" << endl;

}

cout << "Signup successful!" << endl;

Sleep(400);

setcolor(white);

system("cls");

logoSignUp();

} else {

cout << "Maximum user capacity reached." << endl;

Sleep(400);

setcolor(15); // Assuming white color code is 15

system("cls");

logoSignUp();

}

}

string get\_username() {

setcolor(blue);

string user;

cout << "Enter Username: ";

getline(cin, user);

while(user.empty() ||!strings(user)) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, user);

}

return user;

setcolor(white);

}

string get\_pass()

{

setcolor(blue);

cout << "PASSWORD: ";

string pass;

getline(cin,pass);

while (pass.empty() ||!strings(pass)) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, pass);

}

return pass;

setcolor(white);

}

string get\_userType()

{

setcolor(blue);

string type;

cout << "Class Type (Admin/admin or User/user): ";

getline(cin,type);

while (type.empty() || !strings(type)) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, type);

}

return type;

}

int readUserDataFromFile(string names[], string passwords[], string userClass[], int& indexCount) {

ifstream inFile("user\_data.txt");

int count = 0;

string record;

if (inFile.is\_open()) {

while (getline(inFile, record) && count < indexCount) {

names[count] = getField(record, 1);

passwords[count] = getField(record, 2);

userClass[count] = getField(record, 3);

count++;

}

inFile.close();

}

else {

cout << "Unable to open file for reading!" << endl;

}

return count;

}

void login(string userAddress[],int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods[],int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

string username = get\_username();

string pass = get\_pass();

string line;

ifstream file("user\_data.txt");

string userclass;

while(getline(file, line))

{

if (getField(line, 1) == username && getField(line, 2) == pass)

{

userclass = getField(line, 3);

}

}

if ((userclass == "admin" || userclass == "Admin") ) {

cout << "Admin login successful!" << endl;

Sleep(400);

setcolor(white);

system("cls");

view();

optionsForAdmin( userAddress, newReview,addressSize, addressDetails, names, password, userClass, voucher, discountPercentage, numMethods,methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

} else if (userclass == "user" || userclass == "User") {

cout<<endl;

cout << "User login successful!" << endl ;

Sleep(400);

setcolor(white);

system("cls");

view();

optionsForUser(userAddress,newReview, addressSize, addressDetails,names, password, userClass,voucher,discountPercentage,numMethods, methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

} else {

cout << "Invalid entry for login." << endl ;

Sleep(400);

setcolor(white);

}

}

int optionsForAdmin(string userAddress[],int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods[],int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

int option;

string input;

cout << endl;

while (true) {

system("cls");

viewAdmin();

//admin options

setcolor(blue);

Sleep(150);cout << "\t\t\t\t\t\t\t\t\t\t || Enter one of the following number ||" << endl << endl;

Sleep(150);cout << " 1: Add Product Details." << endl;

Sleep(150);cout << " 2: View Added Product Details." << endl;

Sleep(150);cout << " 3: Add Workers Details." << endl;

Sleep(150);cout << " 4: View Added Workers Details." << endl;

Sleep(150);cout << " 5: View Profit gain." << endl;

Sleep(150);cout << " 6: Display Monthly Report." << endl;

Sleep(150);cout << " 7: Display Most Profitable Month." << endl;

Sleep(150);cout << " 8: Display Overall Profit Loss." << endl;

Sleep(150);cout << " 9: View Reviews on all Products." << endl;

Sleep(150);cout << "10: Add Payment Methods." << endl;

Sleep(150);cout << "11: View Added Payment Methods." << endl;

Sleep(150);cout << "12. Add Discount Code." << endl;

Sleep(150);cout << "13: Logout." << endl<<endl;

cout << "Enter your choice: ";

getline(cin, input);

if (input.empty() ||!numbers(input)) {

setcolor(red);

cout << "Invalid input! Please enter a valid option." << endl;

setcolor(white);

Sleep(400);

setcolor(white);

system("cls");

view();

continue;

} setcolor(white);

option = stoi(input); //conversion

if (option >= 1 && option <= 14) {// numbers are only between 0 and 15

if (option == 1) {

system("cls");

viewAdmin();

//to add product

addProductDetails( maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices,product\_Qualities,cart);

}

else if (option == 2) {

system("cls");

viewAdmin();

// to view added product

viewProducts( maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else if (option == 3) {

system("cls");

viewAdmin();

// add worker details

addWorkerDetails(maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings);

}

else if (option == 4) {

system("cls");

viewAdmin();

// view added worker details

viewWorkersDetails(maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else if (option == 5) {

system("cls");

viewAdmin();

// view profit achieve

profit(maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else if (option == 6) {

system("cls");

viewAdmin();

// monthly report

displayMonthlyReport( maxMonths, array, object, months, profits,productsremaining, productsolds, pricesbeforesale,quantitysold, pricesaftersale);

}

else if (option == 7) {

system("cls");

viewAdmin();

mostProfitableMonth(profits, maxMonths, months);

}

else if (option == 8) {

system("cls");

viewAdmin();

overallProfitLoss( profits, maxMonths);

}

else if (option == 9) {

system("cls");

viewAdmin();

// view costomer reviews

viewReviews(newReview, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else if (option == 10) {

system("cls");

viewAdmin();

//add payment methods for customer

AddPaymentMethod(numMethods );

}

else if (option == 11) {

system("cls");

viewAdmin();

//view added payment methods

viewPaymentMethods(numMethods, methods);

}

else if (option == 12) {

system("cls");

viewAdmin();

//add discount for customer

addDiscountCode();;

}

else if (option == 13) {

logout(userAddress,newReview,addressSize, addressDetails, names, password, userClass, voucher, discountPercentage, numMethods, methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

} else {

setcolor(red);

cout << "Invalid option! Please enter a valid choice ." << endl;

Sleep(400);

setcolor(white);

system("cls");

void viewAdmin();

}

}

}

void logo()

{

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

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

cout << "\t\t\t\t\t# || ------------ || #" << endl;

cout << "\t\t\t\t\t# || || LOGIN MENU || || #" << endl;

cout << "\t\t\t\t\t# || ------------ || #" << endl;

cout << "\t\t\t\t\t# || || 1. SIGN-UP || || #" << endl;

cout << "\t\t\t\t\t# || ------------ || #" << endl;

cout << "\t\t\t\t\t# || || 2. SIGN-IN || || #" << endl;

cout << "\t\t\t\t\t# || ------------ || #" << endl;

cout << "\t\t\t\t\t# || || 3. EXIT || || #" << endl;

cout << "\t\t\t\t\t# || ------------ || #" << endl;

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

setcolor(white);

}

void logoSignUp()

{

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

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(white);

}

void logoSignIn()

{

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

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

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

cout << "\t\t\t\t\t + ======================== +" <<endl;

setcolor(white);

}

void loadProductDetails(int maxProducts, string product\_Names[], string product\_Categories[], int reviews[], int product\_Quantities[], int product\_Prices[], string product\_Qualities[], int cart[]) {

ifstream inputFile("products.txt");

string line;

int count = 0;

if (inputFile.is\_open()) {

while (getline(inputFile, line)) {

stringstream ss(line);

string read;

int index = count++;

getline(ss, read, ',');

product\_Names[index] = read;

getline(ss, read, ',');

product\_Categories[index] = read;

getline(ss, read, ',');

product\_Quantities[index] = stoi(read);

getline(ss, read, ',');

product\_Prices[index] = stoi(read);

getline(ss, read);

product\_Qualities[index] = read;

}

inputFile.close();

} else {

cout << "Unable to open file!" << endl;

}

}

void addProductDetails(int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

setcolor(blue);

view();

string product\_Quantity[maxProducts];

string product\_Price[maxProducts];

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* ENTER PRODUCT DETAILS \*\*\*\* ||" << endl << endl;

int numProducts;

string input;

cout << "Enter the number of products: ";

getline(cin, input);

while (input.empty() ||!numbers(input)) {

setcolor(red);

cout << "Invalid input! Please enter a valid number." << endl;

setcolor(white);

setcolor(white);

getline(cin, input);

continue;

}

numProducts = stoi(input);

if (numProducts > maxProducts) {

cout << "Exceeded the maximum number of products." << endl;

setcolor(white);

return;

}

for (int i = 0; i < numProducts; i++) {

cout << "Enter details for Product " << i + 1 << ":" << endl;

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

getline(cin, product\_Names[i]);

while(product\_Names[i].empty() ||!strings(product\_Names[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, product\_Names[i]);

i--;

continue;

}

cout << "Enter the category of the product: ";

getline(cin, product\_Categories[i]);

while(product\_Categories[i].empty() ||!strings(product\_Categories[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, product\_Categories[i]);

i--;

continue;

}

cout << "Enter the quantity of the product: ";

getline(cin, product\_Quantity[i]);

while(product\_Quantity[i].empty() ||!numbers(product\_Quantity[i])) {

cout << "Invalid input for quantity! Please enter a valid number." << endl;

setcolor(white);

i--;

continue;

}

product\_Quantities[i] = stoi(product\_Quantity[i]);

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

getline(cin, product\_Price[i]);

while(product\_Price[i].empty() ||!numbers(product\_Price[i])) {

cout << "Invalid input for price! Please enter a valid number." << endl;

setcolor(white);

i--;

continue;

}

product\_Prices[i] = stoi(product\_Price[i]);

cout << "Check whether it is damaged or not: ";

getline(cin, product\_Qualities[i]);

while(product\_Qualities[i].empty() || !strings(product\_Qualities[i])) {

cout << "Invalid input for quality!" << endl;

cin.ignore();

i--;

continue;

}

// Write the product details to a file

ofstream outputFile("products.txt", ios::app);

outputFile << product\_Names[i] << "," << product\_Categories[i] << "," << product\_Quantities[i] << "," << product\_Prices[i] << "," << product\_Qualities[i] << endl;

outputFile.close();

cout<<endl;

system("cls");

view();

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* ENTER PRODUCT DETAILS \*\*\*\* ||" << endl << endl;

}

}

void view(){

system("cls");

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

setcolor(white);

}

void viewAdmin(){

system("cls");

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

setcolor(white);

}

void viewUser(){

system("cls");

setcolor(green);

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|| || WELCOME TO FABRIC MANAGEMENT SYSTEM || ||" << endl;

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

setcolor(white);

}

void viewProducts(int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]){

view();

string record;

setcolor(blue);

ifstream inputFile("products.txt");

if (!inputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* PRODUCT DETAILS \*\*\*\* ||" << endl << endl;

cout << "NAME CATEGORY QUANTITY PRICE QUALITY" << endl;

while (getline(inputFile, record))

{

cout << getField(record, 1) << " " << getField(record, 2) << " "<< getField(record, 3) << " " << getField(record, 4) << " "<< getField(record, 5) << endl;

}

inputFile.close();

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

getch();

setcolor(white);

}

void addReview(int newReview,int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

void view();

setcolor(blue);

string productName, productCategory;

bool found = false;

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

getline(cin,productName);

while(productName.empty() || !strings(productName) ) {

setcolor(red);

cout << "Invalid input! Please enter valid Product Name." << endl;

setcolor(white);

cin.ignore();

continue;

}

cout << "Enter the category of the product: ";

getline(cin,productCategory);

while(productCategory.empty() || !strings(productCategory)) {

setcolor(red);

cout << "Invalid input! Please enter valid Product Category." << endl;

setcolor(white);

getline(cin,productCategory);

continue;

}

for (int i = 0; i < maxProducts; ++i) {

if (productName == product\_Names[i] && productCategory == product\_Categories[i]) {

found = true;

cout << "Enter the review for " << productName << " (" << productCategory << "): ";

cin>>newReview;

reviews[i] = newReview;

cout << "Review added successfully!" << endl;

break;

}

}

if (!found) {

cout << "Product not found!" << endl;

}

// Writing the updated reviews to a file

ofstream outputFile("reviews.txt");

if (!outputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

for (int i = 0; i < maxProducts; ++i) {

outputFile << product\_Names[i] << "," << product\_Categories[i] << "," << reviews[i] << "," << product\_Quantities[i] << "," << product\_Prices[i] << "," << product\_Qualities[i] << endl;

}

setcolor(white);

outputFile.close();

}

void viewReviews(int newReview,int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

void view();

setcolor(blue);

ifstream inputFile("reviews.txt");

if (!inputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* Reviews for All Products \*\*\*\* ||" << endl << endl;

for (int i = 0; i < newReview; ++i) {

cout << "Product: " << product\_Names[i] << " (" << product\_Categories[i] << ") - Reviews: ";

for (int j = 0; j < newReview; ++j) {

cout << "\*";

}

cout << endl;

}

inputFile.close();

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

getch();

setcolor(white);

}

void loadWorkerDetails(int maxWorkers, string worker\_names[], string worker\_codes[], string worker\_ranks[], string worker\_performances[], string worker\_hours[], string worker\_sellings[]) {

ifstream inputFile("worker\_details.txt");

if (!inputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

int i = 0;

string line;

while (getline(inputFile, line) && i < maxWorkers) {

istringstream iss(line);

string read;

getline(iss, read, ',');

worker\_names[i] = read;

getline(iss, read, ',');

worker\_codes[i] = read;

getline(iss, read, ',');

worker\_ranks[i] = read;

getline(iss, read, ',');

worker\_performances[i] = read;

getline(iss, read, ',');

worker\_hours[i] = read;

getline(iss, read);

worker\_sellings[i] = read;

i++;

}

inputFile.close();

}

void addWorkerDetails(int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[]) {

void view();

setcolor(blue);

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* ENTER WORKER DETAILS \*\*\*\* ||" << endl << endl;

string num;

int numWorkers;

cout << "Enter the number of workers: ";

getline(cin,num);

while(num.empty() ||!numbers(num)) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin,num);

continue;

}

numWorkers = stoi(num);

if (numWorkers > maxWorkers) {

cout << "Exceeded the maximum number of workers." << endl;

setcolor(white);

Sleep(500);

return;

}

for (int i = 0; i < numWorkers; i++) {

cout << endl;

setcolor(blue);

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* WORKER DETAILS \*\*\*\* ||" << i + 1 << ":" << endl;

cout << "Enter the name of the worker: ";

getline(cin, worker\_names[i]);

while (worker\_names[i].empty() || !strings(worker\_names[i])) {

setcolor(red);

cout << "Invalid input! Please enter a valid name for the worker." << endl;

setcolor(white);

setcolor(white);

i--;

continue;

}

cout << "Enter the code of the worker: ";

getline(cin,worker\_codes[i]);

while(worker\_codes[i].empty() ||!numbers(worker\_codes[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, worker\_codes[i]);

if (worker\_codes[i].empty() || stoi(worker\_codes[i]) < 0) {

setcolor(red);

cout << "Invalid input! Please enter a positive integer for the worker code." << endl;

setcolor(white);

setcolor(white);

i--;

continue;

}

}

cout << "Enter the rank of the worker: ";

getline(cin, worker\_ranks[i]);

while(worker\_ranks[i].empty() ||!numbers(worker\_ranks[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin,worker\_ranks[i]);

while(!cin ) {

setcolor(red);

cout << "Invalid input! Please enter a positive integer for the worker rank." << endl;

setcolor(white);

i--;

continue;

}

}

cout << "Enter the performance of the worker: ";

getline(cin, worker\_performances[i]);

while(worker\_performances[i].empty() ||!strings(worker\_performances[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, worker\_performances[i]);

}

cout << "Enter the working hours of this worker: ";

getline(cin,worker\_hours[i]);

while(worker\_hours[i].empty() ||!numbers(worker\_hours[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, worker\_hours[i]);

i--;

continue;

}

cout << "Enter the daily basis selling of this worker: ";

getline(cin,worker\_sellings[i]);

while(worker\_sellings[i].empty() ||!numbers(worker\_sellings[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, worker\_sellings[i]);

if (!cin ) {

setcolor(red);

cout << "Invalid input! Please enter a positive integer for the selling value." << endl;

setcolor(white);

setcolor(white);

i--;

continue;

}

}

}

// Writing the worker details to a file

ofstream outputFile("worker\_details.txt");

if (!outputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

for (int i = 0; i < numWorkers; ++i) {

outputFile << worker\_names[i] << "," << worker\_codes[i] << "," << worker\_ranks[i] << "," << worker\_performances[i] << "," << worker\_hours[i] << "," << worker\_sellings[i] << endl;

}

outputFile.close();

setcolor(white);

}

void viewWorkersDetails(int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]){

setcolor(blue);

cout << "\t\t\t\t\t\t\t\t\t\t || \*\*\*\* WORKER DETAILS \*\*\*\* ||" << endl << endl;

cout << "NAME CODE RANK PERFORMANCE WORKING HOURS SELLING" << endl;

ifstream inputFile("worker\_details.txt");

for (int i = 0; i < maxProducts && !product\_Names[i].empty(); i++ ){

if (inputFile.is\_open()) {

while (i < maxWorkers && getline(inputFile, worker\_names[i], ',') && getline(inputFile, worker\_codes[i], ',') && getline(inputFile, worker\_ranks[i], ',') && getline(inputFile, worker\_performances[i], ',') && getline(inputFile, worker\_hours[i], ',') && getline(inputFile, worker\_sellings[i]))

{

cout << worker\_names[i] << " " << worker\_codes[i] << " " << worker\_ranks[i] << " " << worker\_performances[i] << " " << worker\_hours[i] << " " << worker\_sellings[i] << endl << endl;

i++;

}

inputFile.close();

}

}

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

getch();

setcolor(white);

}

void loadProfitDetails(int maxMonths, string months[], int profits[], string pricesbeforesale[], string pricesaftersale[], int product\_Quantities[]) {

ifstream inputFile("profits.txt");

if (!inputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

int i = 0;

string line;

while (getline(inputFile, line) && i < maxMonths) {

istringstream iss(line);

string read;

getline(iss, read, ',');

months[i] = read;

getline(iss, read, ',');

product\_Quantities[i] = stoi(read);

getline(iss, read, ',');

pricesbeforesale[i] = read;

getline(iss, read, ',');

pricesaftersale[i] = read;

getline(iss, read);

profits[i] = stoi(read);

i++;

}

inputFile.close();

}

void profit(int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[]) {

void view();

setcolor(blue);

ofstream outputFile("profits.txt", ios::app);

for (int i = 0; i < maxMonths; i++) {

cout << "Enter details for Month " << i + 1 << ":" << endl;

getline(cin, months[i]);

// Input validation for month (1-12)

while (months[i].empty() ||!numbers(months[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, months[i]);

continue;

}

string input;

cout << "Quantity of products: ";

getline(cin, input);

// Input validation for quantity of products

while (input.empty() ||!numbers(input)) {

getline(cin, input);

cin.ignore();

continue;

}

product\_Quantities[i] = stoi(input);

// Input validation for total price of sold products before sale

cout << "Total price of sold products before sale: ";

getline(cin, pricesbeforesale[i]);

while (pricesbeforesale[i].empty() ||!numbers(pricesbeforesale[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, pricesbeforesale[i]);

if (!cin || stoi(pricesbeforesale[i]) < 0) {

setcolor(red);

cout << "Invalid input! Please enter again." << endl;

setcolor(white);

i--;

}

continue;

}

// Input validation for total price of sold products after selling

cout << "Total price of sold products after selling: ";

getline(cin, pricesaftersale[i]);

while (pricesaftersale[i].empty() ||!numbers(pricesaftersale[i])) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, pricesaftersale[i]);

if (!cin || stoi(pricesaftersale[i]) < 0) {

setcolor(red);

cout << "Invalid input! Please enter again." << endl;

setcolor(white);

i--;

}

continue;

}

// Calculate profits

profits[i] = stoi(pricesaftersale[i]) - stoi(pricesbeforesale[i]);

// Determine company status based on profit or loss

if (profits[i] < stoi(pricesbeforesale[i])) {

cout << "\t\t\t\t\t\t\t\t\t\t Status: Company is in Loss." << endl << endl;

} else if (profits[i] > stoi(pricesbeforesale[i])) {

cout << "\t\t\t\t\t\t\t\t\t\t Status: Company is in Profit." << endl << endl;

}

// Write profit details to file

outputFile << months[i] << ","<< product\_Quantities[i] << "," << pricesbeforesale[i] << "," << pricesaftersale[i] << "," << profits[i] << endl;

}

setcolor(white);

outputFile.close();

}

void displayMonthlyReport(int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[]) {

void view();

setcolor(blue);

ifstream inputFile("profits.txt");

int i = 0;

if (inputFile.is\_open()) {

while (i < maxMonths &&

getline(inputFile, months[i], ',') && inputFile >> profits[i] &&inputFile.ignore() &&getline(inputFile, productsremaining[i])) {

cout << "Month " << i + 1 << " - " << months[i] << ":" << endl;

cout << "Profit/Loss: " << profits[i] << endl;

cout << "Products Remaining: " << productsremaining[i] << endl;

cout << endl;

i++;

}

inputFile.close();

} else {

cout << "Unable to open file." << endl;

// Handle error if unable to open file

}

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

getch();

setcolor(white);

}

void mostProfitableMonth(int profits[],int maxMonths,string months[]) {

void view();

setcolor(blue);

ifstream inputFile("profits.txt");

int maxProfit = 0;

int maxIndex = 0;

int currentProfit = 0;

string currentMonth;

if (inputFile.is\_open()) {

for (int i = 0; i < maxMonths; i++) {

// Assuming profits are stored in the profits array

inputFile >> currentMonth >> currentProfit;

if (currentProfit > maxProfit) {

maxProfit = currentProfit;

maxIndex = i;

}

}

inputFile.close();

if (maxProfit > 0) {

cout << "The most profitable month is: " << months[maxIndex] << " (Profit: " << maxProfit << ")" << endl << endl;

} else {

cout << "No profit data found in the file." << endl << endl;

}

} else {

cout << "Unable to open file." << endl;

// Handle error if unable to open file

}

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

getch();

setcolor(white);

}

void overallProfitLoss(int profits[],int maxMonths) {

void view();

setcolor(blue);

ifstream inputFile("profits.txt");

int totalProfit = 0;

int currentProfit = 0;

if (inputFile.is\_open()) {

while (inputFile >> currentProfit) {

totalProfit += currentProfit;

}

inputFile.close();

cout << "Overall Profit/Loss for the year: " << totalProfit << endl << endl;

} else {

cout << "Unable to open file." << endl;

// Handle error if unable to open file

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

getch();

setcolor(white);

}

}

int optionsForUser(string userAddress[],int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods[],int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[])

{

int option;

setcolor(blue);

string input;

while (true) {

viewUser();

setcolor(blue);

Sleep(150);cout << "\t\t\t\t\t\t\t\t\t\t || Enter one of the following number ||" << endl << endl;

Sleep(150);cout << " 1: View Available Products." << endl;

Sleep(150);cout << " 2: Add Product to cart." << endl;

Sleep(150);cout << " 3: View Cart." << endl;

Sleep(150);cout << " 4: Add Delivery Address." << endl;

Sleep(150);cout << " 5: View Delivery Address." << endl;

Sleep(150);cout << " 6: View Discount Voucher Code." << endl;

Sleep(150);cout << " 7: Apply Discount Voucher." << endl;

Sleep(150);cout << " 8: View payment method." << endl;

Sleep(150);cout << " 9: Add Reviews." << endl;

Sleep(150);cout << "10: Logout." << endl;

cout << "Enter your choice: ";

getline(cin, input);

if(input.empty() ||!numbers(input)) {

setcolor(red);

cout << "Invalid input! Please enter a valid option." << endl;

setcolor(white);

Sleep(400);

setcolor(white);

system("cls");

viewUser();

getline(cin, input);

continue;

}

option = stoi(input);

if (option >= 1 && option <= 11)

{

if (option == 1)

{

system("cls");

viewUser();

viewProducts( maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else if (option == 2)

{

system("cls");

viewUser();

addToCart(maxProducts, cart);

}

else if (option == 3)

{

system("cls");

viewUser();

viewCart( cart, product\_Names, maxProducts, product\_Prices);

}

else if (option == 4)

{

system("cls");

viewUser();

inputDeliveryAddress();

}

else if (option == 5)

{

system("cls");

viewUser();

viewDeliveryAddress();

}

else if (option == 6)

{

system("cls");

viewUser();

viewDiscountCode(discountPercentage,voucher);

}

else if (option == 7)

{

system("cls");

viewUser();

applyDiscount(product\_Prices,voucher,discountPercentage, maxProducts);

}

else if (option == 8)

{

system("cls");

viewUser();

viewPaymentMethods(numMethods,methods);

}

else if (option == 9)

{

system("cls");

viewUser();

addReview(newReview, maxProducts, product\_Names,product\_Categories, reviews, product\_Quantities, product\_Prices,product\_Qualities,cart);

}

else if (option == 10)

{

logout(userAddress,newReview, addressSize, addressDetails, names, password, userClass,voucher, discountPercentage,numMethods,methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

else

{ cout << endl;

cout << "Invalid Option! Please enter a valid option."<<endl;

Sleep(400);

setcolor(white);

system("cls");

viewUser();

}

}

}

setcolor(white);

}

void logout(string userAddress[],int newReview,int addressSize,string addressDetails[],string names[],string password[],string userClass[],string voucher,float discountPercentage,int numMethods,string methods[],int indexCount,int maxMonths,string array[],string object,string months[],int profits[],string productsremaining[],string productsolds[],string pricesbeforesale[],string quantitysold[],string pricesaftersale[],int maxWorkers,string worker\_names[],string worker\_codes[],string worker\_ranks[],string worker\_performances[],string worker\_hours[],string worker\_sellings[],int maxProducts, string product\_Names[],string product\_Categories[],int reviews[],int product\_Quantities[],int product\_Prices[],string product\_Qualities[],int cart[])

{

setcolor(red);

int x = 100;

for(int i = 0; i <= x;i++){

cout<<"Logging out... ";

cout<<i<<"%";

Sleep(5);

cout << '\r';

setcolor(red);

}

user(userAddress,newReview, addressSize,addressDetails,names, password, userClass, voucher, discountPercentage, numMethods, methods,indexCount,maxMonths, array, object, months, profits, productsremaining, productsolds, pricesbeforesale, quantitysold, pricesaftersale,maxWorkers, worker\_names, worker\_codes, worker\_ranks, worker\_performances, worker\_hours, worker\_sellings, maxProducts, product\_Names, product\_Categories, reviews, product\_Quantities, product\_Prices, product\_Qualities, cart);

}

void loadPaymentMethods(int numMethods, string methods[]) {

ifstream inputFile("payment\_methods.txt");

if (!inputFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

int i = 0;

string line;

while (getline(inputFile, line) && i < numMethods) {

methods[i] = line;

i++;

}

inputFile.close();

}

void AddPaymentMethod(int numMethods ) {

setcolor(blue);

const int maxMethods = 10;

string methods[maxMethods];

cout << "Enter the number of payment methods: ";

cin >> numMethods;

while ( numMethods <= 0 || numMethods > maxMethods) {

setcolor(red);

cout << "Invalid input! Please enter a positive number within the valid range." << endl;

setcolor(white);

cout << "Enter the number of payment methods: ";

cin >> numMethods;

continue;

}

cin.ignore(); // Clear the input buffer

cout << "Enter payment methods:" << endl;

ofstream outputFile("payment\_methods.txt", ios::app);

if (outputFile.is\_open()) {

for (int i = 0; i < numMethods; ++i) {

cout << "Method " << i + 1 << ": ";

getline(cin, methods[i]);

while (methods[i].empty()) {

setcolor(red);

cout << "Invalid input! Payment method cannot be empty. Please enter again: ";

setcolor(white);

getline(cin, methods[i]);

}

outputFile << methods[i] << endl;

}

outputFile.close();

cout << "Payment Methods added successfully!" << endl;

} else {

cout << "Unable to open file for writing." << endl;

// Handle error if unable to open file

}

Sleep(300);

setcolor(white);

}

void viewPaymentMethods(int numMethods,string methods[]){

setcolor(blue);

ifstream inputFile("payment\_methods.txt");

if (inputFile.is\_open()) {

string method;

cout << "Payment Methods:" << endl;

int i = 0;

while (getline(inputFile, method)) {

cout << i << ": " << method << endl;

++i;

}

inputFile.close();

} else {

cout << "Unable to open file." << endl;

// Handle error if unable to open file

}

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

getch();

setcolor(white);

}

void viewDeliveryAddress() {

ifstream fin("address.txt");

setcolor(blue);

if (fin.is\_open()) {

string line;

cout << "Address Details:\n";

while (getline(fin, line)) {

cout << line << endl;

}

fin.close();

} else {

cout << "Unable to open the file." << endl;

}

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

setcolor(white);

}

void loadDeliveryAddress() {

ifstream fin("address.txt");

if (fin.is\_open()) {

string line;

cout << "Loaded Address Details:\n";

while (getline(fin, line)) {

cout << line << endl;

}

fin.close();

} else {

cout << "Unable to open the file." << endl;

}

}

void inputDeliveryAddress() {

setcolor(blue);

ofstream fout("address.txt", ios::app); // Open file in append mode

if (fout.is\_open()) {

string country, city, state, postalCode, streetAddress;

cout << "Enter Country: ";

getline(cin, country);

cout << "Enter City: ";

getline(cin, city);

cout << "Enter State: ";

getline(cin, state);

cout << "Enter Postal Code: ";

getline(cin, postalCode);

cout << "Enter Street Address: ";

getline(cin, streetAddress);

// Write address details to the file

fout << "Country: " << country << endl;

fout << "City: " << city << endl;

fout << "State: " << state << endl;

fout << "Postal Code: " << postalCode << endl;

fout << "Street Address: " << streetAddress << endl;

fout.close();

cout << "Address details saved to address.txt" << endl;

} else {

cout << "Error opening the file!" << endl;

}

setcolor(white);

}

void loadCart(int maxProducts, int cart[]) {

ifstream cartFile("cart.txt");

if (!cartFile.is\_open()) {

cout << "Error opening the file!" << endl;

return;

}

// Initialize cart array with zeros

for (int i = 0; i < maxProducts; ++i) {

cart[i] = 0;

}

int productIndex, quantity;

while (cartFile >> productIndex >> quantity) {

if (productIndex >= 0 && productIndex < maxProducts && quantity >= 0) {

cart[productIndex] += quantity;

}

}

cartFile.close();

}

void addToCart(int maxProducts,int cart[]) {

int numProducts;

setcolor(blue);

cout << "Enter the number of products you want to add to the cart: ";

while (!(cin >> numProducts) || numProducts <= 0 || numProducts > maxProducts) {

setcolor(red);

cout << "Invalid input! Please enter a positive number within the valid range." << endl;

setcolor(white);

cout << "Enter the number of products you want to add to the cart: ";

}

ofstream cartFile("cart.txt", ios::app);

if (cartFile.is\_open()) {

for (int i = 0; i < numProducts; ++i) {

int productIndex;

cout << "Enter the index of the product you want to add to the cart (0 to " << maxProducts - 1 << "): ";

while (!(cin >> productIndex) || productIndex < 0 || productIndex >= maxProducts) {

setcolor(red);

cout << "Invalid input! Please enter a valid product index (0 to " << maxProducts - 1 << "): ";

setcolor(white);

cin.ignore();

}

int quantity;

cout << "Enter the quantity for product " << productIndex << ": ";

while (!(cin >> quantity) || quantity < 0) {

setcolor(red);

cout << "Invalid input! Please enter a non-negative quantity: ";

setcolor(white);

cin.ignore();

}

cart[productIndex] += quantity;

cartFile << "Product " << productIndex << ": " << quantity << endl;

cout << "Product added to cart!" << endl;

}

cartFile.close();

} else {

cout << "Unable to open file for writing." << endl;

// Handle error if unable to open file

}

setcolor(white);

}

void viewCart(int cart[],string product\_Names[],int maxProducts,int product\_Prices[]) {

setcolor(blue);

ifstream cartFile("cart.txt");

if (cartFile.is\_open()) {

string line;

cout << "Cart Contents:" << endl;

while (getline(cartFile, line)) {

// Extract product index and quantity from the line in the file

int productIndex, quantity;

cout << "Product Index: " << productIndex << ", Name: " << product\_Names[productIndex]

<< ", Quantity: " << cart[productIndex] << ", Price: " << product\_Prices[productIndex] << endl;

}

cartFile.close();

} else {

cout << "Unable to open cart file." << endl;

// Handle error if unable to open file

}

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

getch();

setcolor(white);

}

void applyDiscount(int product\_Prices[],string voucher,float discountPercentage,int maxProducts) {

setcolor(blue);

discountPercentage = 0.0;

ifstream inFile("discount\_codes.txt");

if (!inFile.is\_open()) {

cout << "Unable to open the file." << endl;

return;

}

if (!(inFile >> voucher >> discountPercentage)) {

cout << "Invalid format in the file." << endl;

inFile.close();

return;

}

inFile.close();

string voucherCode;

cout<<"Enter the voucher code: ";

getline(cin,voucherCode);

while(!strings(voucherCode)) {

setcolor(red);

cout << "Invalid input! Please enter again: ";

setcolor(white);

getline(cin, voucherCode);

}

if (voucherCode == voucher) {

for (int i = 0; i < maxProducts; ++i) {

product\_Prices[i] = product\_Prices[i]-(product\_Prices[i]\*(discountPercentage/100));

}

cout << "Discount of " << discountPercentage << "% applied to all products using the voucher code." << endl;

} else {

cout << "Invalid voucher code. No discount applied." << endl;

}

cout << "Updated prices after discount:" << endl;

for (int i = 0; i < maxProducts; ++i) {

cout << "Product " << i + 1 << " price: " << product\_Prices[i] << endl;

}

setcolor(white);

}

void loadDiscountInfo(string voucher, float discountPercentage) {

ifstream inFile("discount\_codes.txt");

if (!inFile.is\_open()) {

cout << "Unable to open the file." << endl;

return;

}

if (!(inFile >> voucher >> discountPercentage)) {

cout << "Invalid format in the file." << endl;

inFile.close();

return;

}

inFile.close();

}

void addDiscountCode(){

setcolor(blue);

float discountPercentage;

string voucher;

cout << "Enter the Discount Coupon Code: ";

cin >> voucher;

while (voucher.empty() ||!strings(voucher)) {

cout << "Invalid Entry! Please enter again: ";

getline(cin,voucher);

continue;

}

cout<<"Enter Discount % you get on this voucher: ";

string input;

cin >> input;

while (input.empty() ||!numbers(input) || stoi(input) <= 0 || stoi(input) > 100) {

cout << "Invalid discount percentage! Please enter a value between 1 and 100: ";

cin >> input;

continue;

}

discountPercentage = stof(input);

// Writing the voucher and discount percentage to a file

ofstream outFile("discount\_codes.txt", ios::app); // Open the file in append mode

if (outFile.is\_open()) {

outFile << voucher << " " << discountPercentage << "%" << endl;

outFile.close();

cout << "Discount code added successfully!" << endl;

} else {

cout << "Unable to open the file." << endl;

}

setcolor(white);

Sleep(300);

}

int loadDataFromUserFile(string path,string names[],string password[],string userClass[])

{

ifstream inputFile(path); // Replace "your\_file\_path.txt" with the path to your file

if (!inputFile) {

cerr << "Error opening the file." << endl;

return 1;

}

int index = 0;

string line;

while (getline(inputFile, line) && index < 50) {

istringstream iss(line);

string name, pass, uClass;

if (iss >> name >> pass >> uClass) {

names[index] = name;

password[index] = pass;

userClass[index] = uClass;

index++;

}

}

inputFile.close();

return 0;

}

void viewDiscountCode(float discountPercentage,string voucher) {

setcolor(blue);

// Voucher: ABC123

// Discount%: 10.0

ifstream inFile("discount\_info.txt");

if (!inFile.is\_open()) {

cout << "Unable to open the file." << endl;

return;

}

if (inFile >> voucher >> discountPercentage) {

cout << "The discount voucher " << voucher << " available has a discount of " << discountPercentage << "% which can be applied on all the products available..." << endl;

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

inFile.close();

getch(); // Wait for user input

} else {

cout << "Invalid format in the file." << endl;

}

setcolor(white);

}

// Validations

bool strings(string str) // It validates if the required input are alphabets

{

bool conclusion = true;

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

{

if (!isalpha(str[i]))

{

conclusion = false;

break;

}

}

return conclusion;

}

bool numbers(string number) // It validates if the required input are integers

{

bool integer = true;

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

{

if (!isdigit(number[i]))

{

integer = false;

break;

}

}

return integer;

}

string setcolor(int color)

{

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

SetConsoleTextAttribute(hcon, color);

return "";

}

string getField(string record, int field)

{

int commaCount = 1;

string item;

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

{

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

{

commaCount = commaCount + 1;

}

else if (commaCount == field)

{

item = item + record[x];

}

}

return item;

}