# 

**Fashion Is U**

A logo of a university of engineering and technology

Description automatically generated

# Session 2023 – 2027

# Submitted by:

Mustafa Noor 2023-CS-17

# Supervised by:

Dr. Muhammad Awais Hassan

# Course:

CSC-102 Programming Fundamentals

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

Table of Contents

[ Fashion Is U 3](#_Toc153819554)

[ Users Of Application 3](#_Toc153819555)

[ Functional Requirements 3](#_Toc153819556)

[ Wireframes 4](#_Toc153819557)

[ Data Structures (Parallel Arrays) 17](#_Toc153819558)

[ Function Prototypes 17](#_Toc153819559)

[ Function Working Flow 21](#_Toc153819560)

[ Complete Code of Fashion Is U 22](#_Toc153819561)

[ Weakness in the Business Application 22](#_Toc153819562)

[ Future Directions 22](#_Toc153819563)

# Fashion Is U

* This application is a clothes shop, and it will provide customers with different types of clothing items. The main objective is to have an application that can interact with the user in such a way that at the end of the process he/she will be able to buy clothing items. Online shopping is an application of computer science, and it contributes to the computer science field by making a system that enables the user to buy clothing items through a screen. At the end of the project, it should be able to have a vast variety of clothing items for both men and women to buy.

# Users Of Application

This application will have two users:

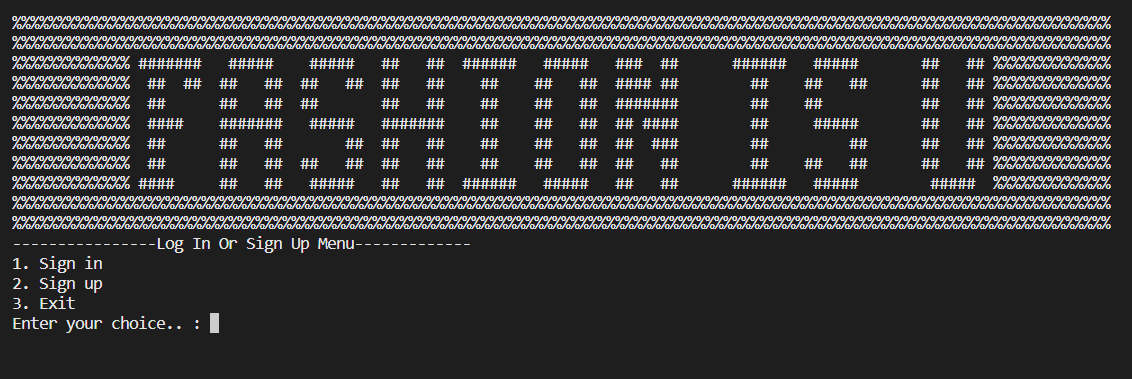
* Client or Customer: This user will have access to all the clothing items that are available.
* Employee: The employee can make changes in the function displayed to the customers.

# Functional Requirements

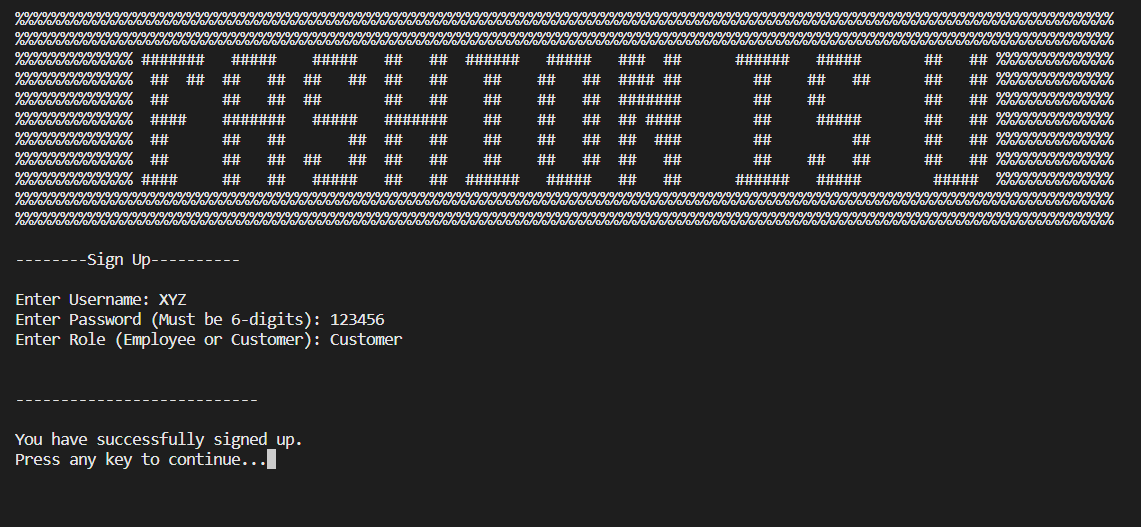
|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | User Type | Required Function | Result of Action Performed |
| 1 | Employee | 1. View List of Clothes | A tabular list of clothes appears. |
| 1. Add an item of Clothing | A new item of clothing appears. |
| 3. Change Available Stock | Change the quantity available of items |
| 4. Check Reviews | Review given by customers appear. |
| 5. Delete an item | Deletion in items of clothing menu |
| 6. Change Name of an Item | Change the name of already made clothing item. |
| 7. See Current Customers | List of Customers with their details comes on. |
| 8. Add Delivery Area | Addition in available delivery areas |
| 9. Remove a Delivery Area | Delete a delivery Area |
| 10. Change Contact Information | This will change the contact information displayed to the customer. |
| 11. Log out | This will log the user out. |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | User Type | Required Function | Result of Action Performed |
| 2 | Customer | 1.View List of Clothing | A tabular list of clothes appears |
| 2.View Cart | List of Purchased Items |
| 3.Select Payment Options | Selects A Payment Method |
| 4.Select Delivery Area | Selects a Delivery Area from given List. |
| 5.View the Bill | A bill in tabular form is shown to the customer. |
| 6. Change the Quantity | Here customer can change the quantity of a cart item. |
| 7.Pay the Bill | The customer can Pay the Bill. |
| 8. Check Status of Order | The customer can check the status of already paid order. |
| 9. Leave a review. | The customer can give a review. |
| 10.Contact Us | The customer is given contact details. |
| 11. Log Out | It logs the user out. |

# Wireframes



**Figure 1: Sub Menu Before Main Menu**



**Figure 2: Sign Up Menu**

A person holding a sign

Description automatically generated

**Figure 3: Sign In Menu**

A screen shot of a computer

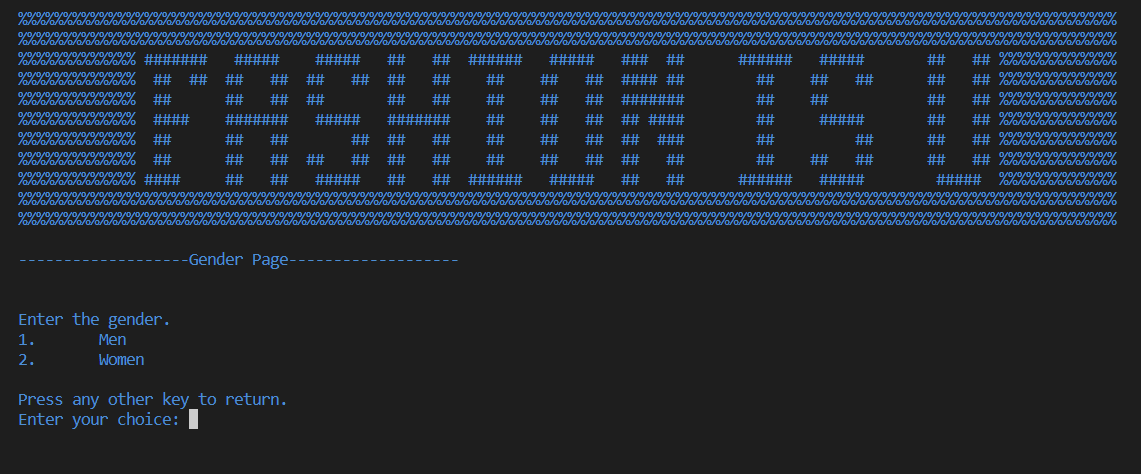
Description automatically generated

**Figure 4: Customer Menu**

A screenshot of a computer screen

Description automatically generated

**Figure 5: Employee Menu**



**Figure 6: Gender Page**

A screen shot of a computer screen

Description automatically generated**Figure 7: Male Items Page**

A screen shot of a computer

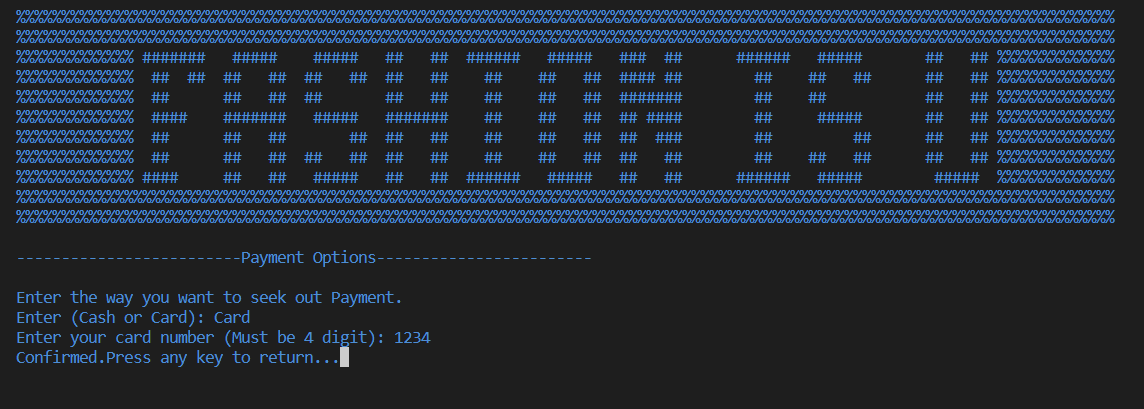
Description automatically generated

**Figure 8: Women Items Page**

A screen shot of a computer

Description automatically generated

**Figure 9: Cart**

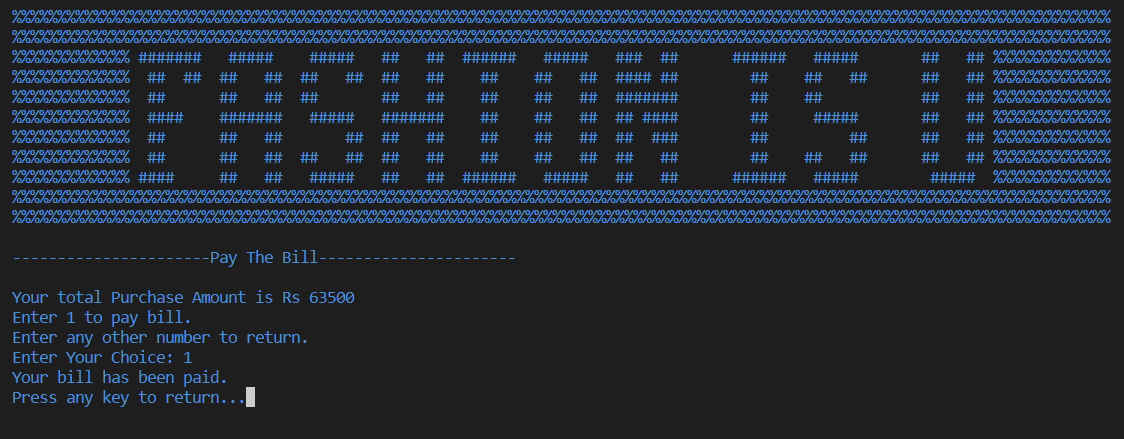
****

**Figure 10: Payment Options Page**

A screen shot of a computer screen

Description automatically generated

**Figure 11: Delivery Area Page**

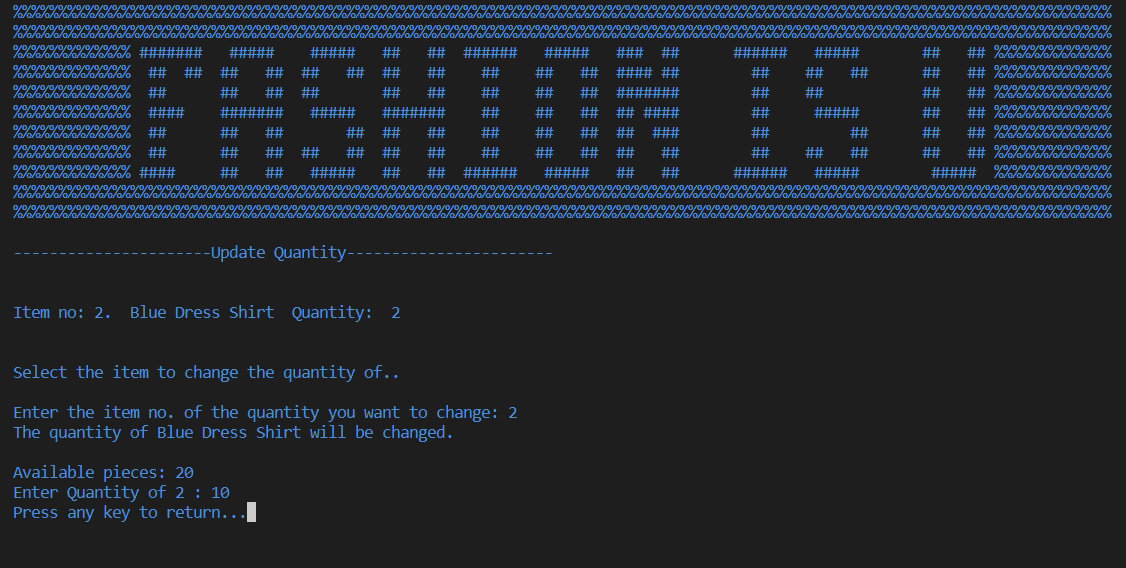


**Figure 12: Pay the Bill Page**

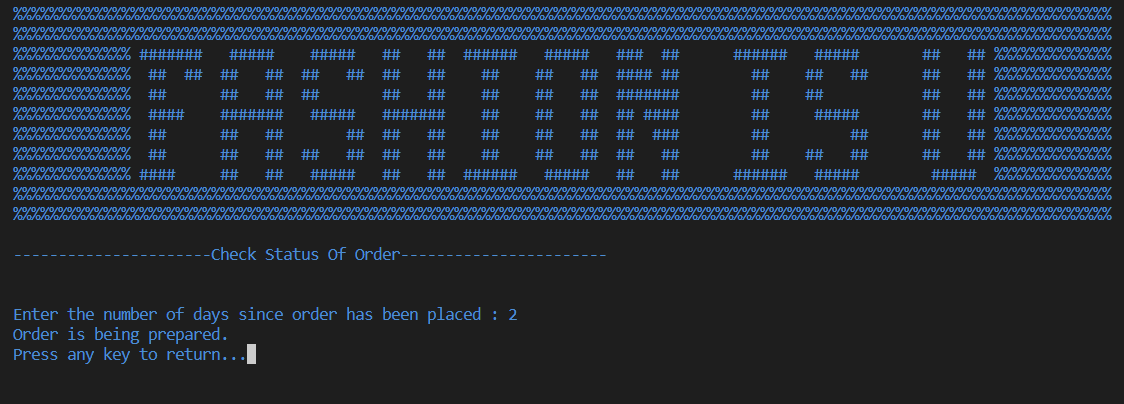
**A screen shot of a computer screen

Description automatically generated**

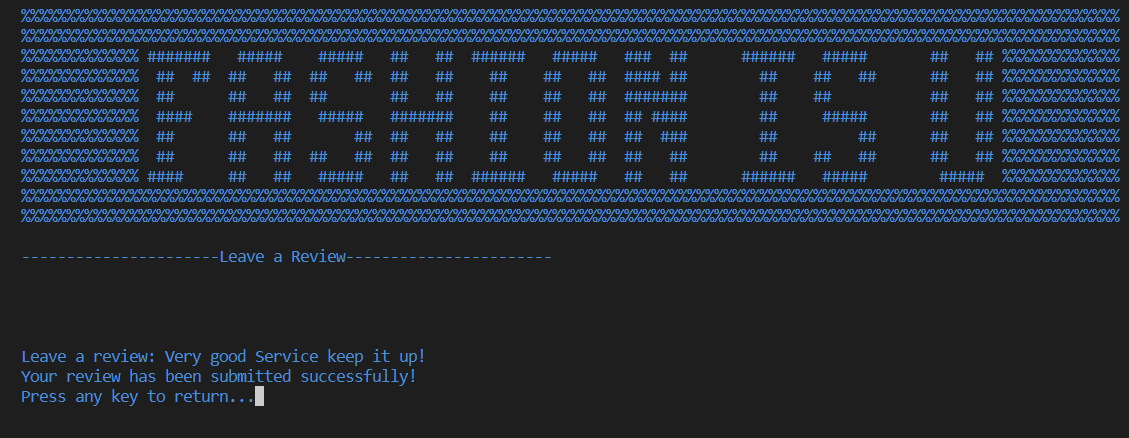
**Figure 13: View the Bill**



**Figure 14: Update Quantity Page**



**Figure 15: Check Status of Order**

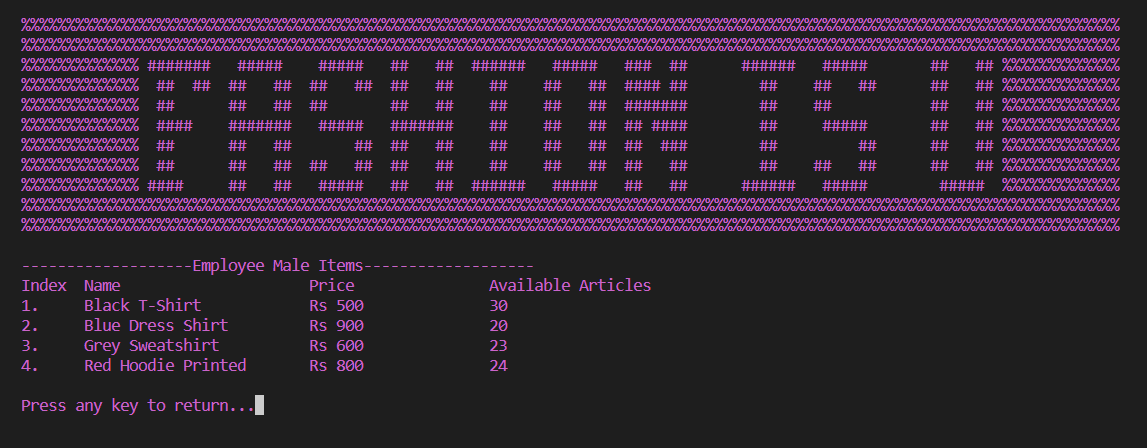
****

**Figure 16: Leave a Review Page**

A screenshot of a computer screen

Description automatically generated

**Figure 17: Contact Us Page**

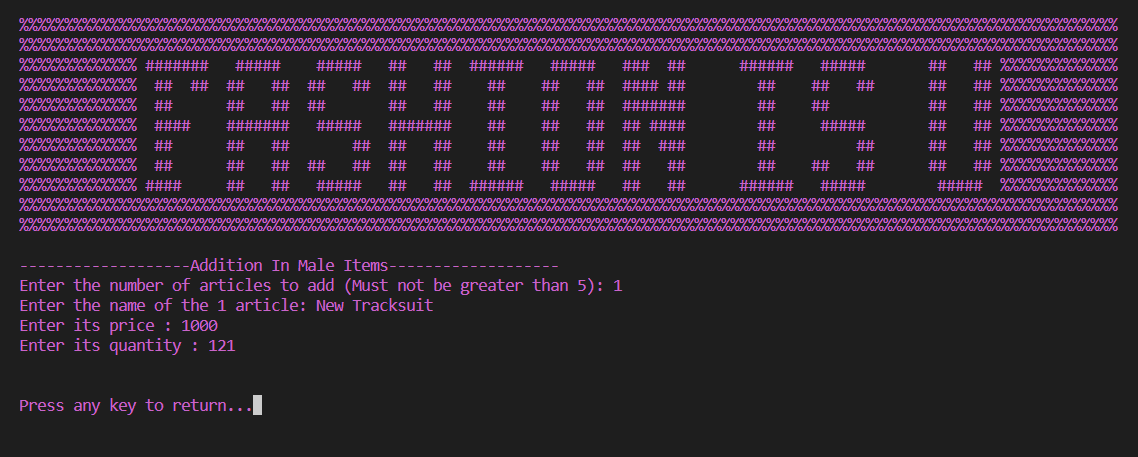


**Figure 18: Employee Male Items Page**

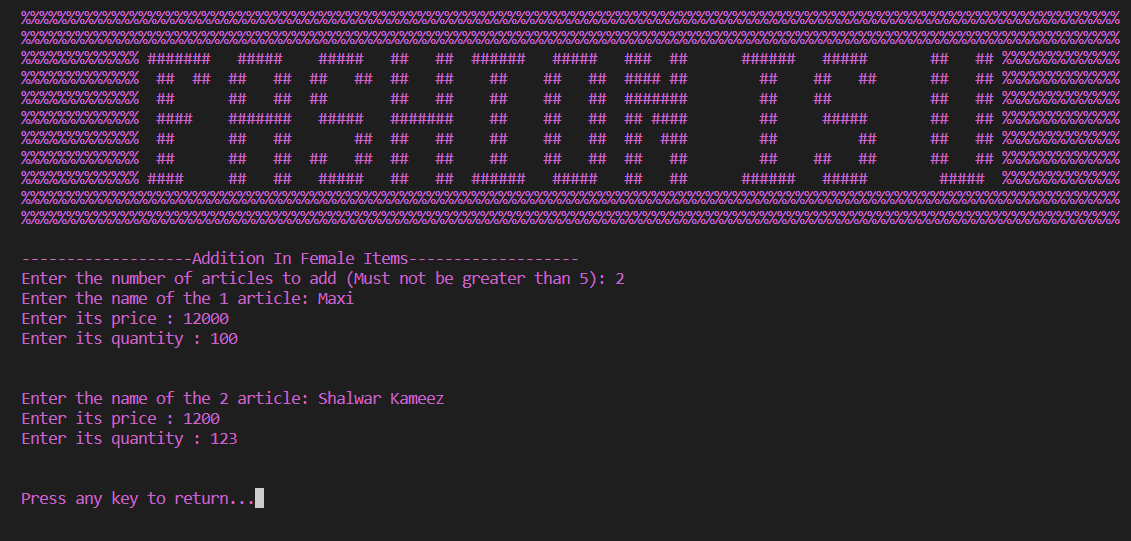
A screenshot of a computer screen

Description automatically generated

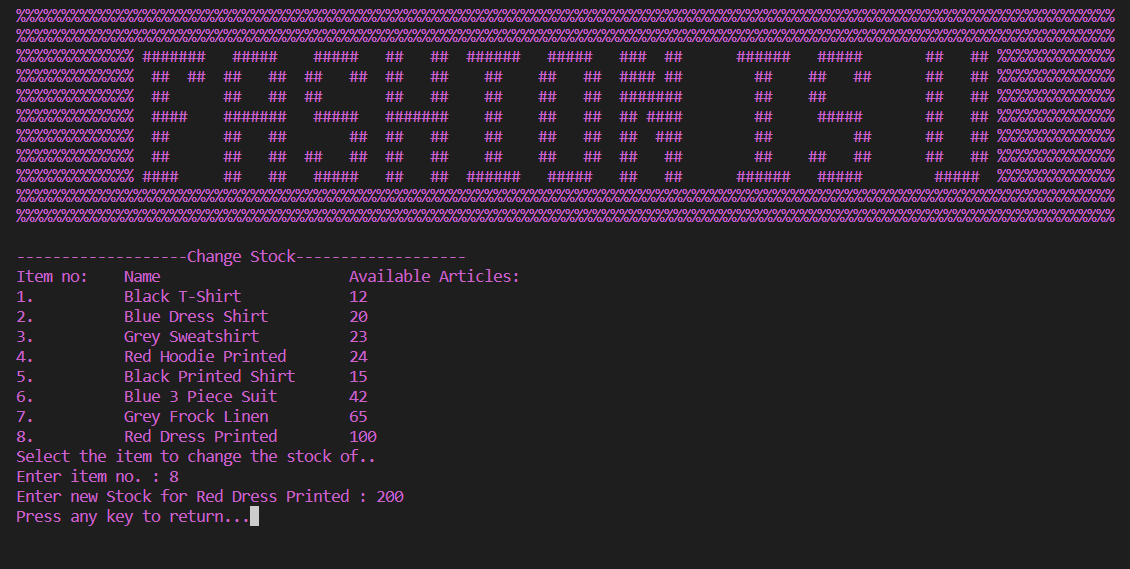
**Figure 19: Employee Female Items Page**



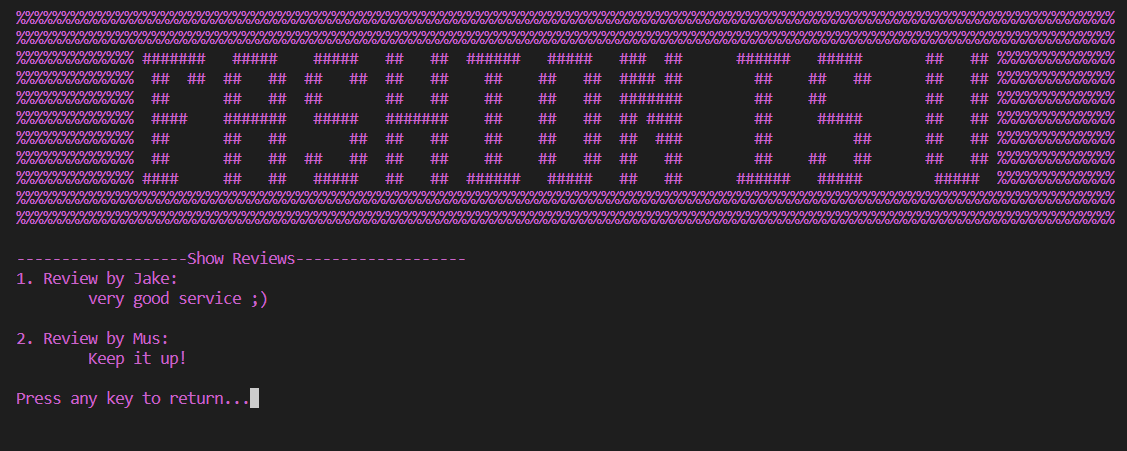
**Figure 20: Addition in Male items Page**



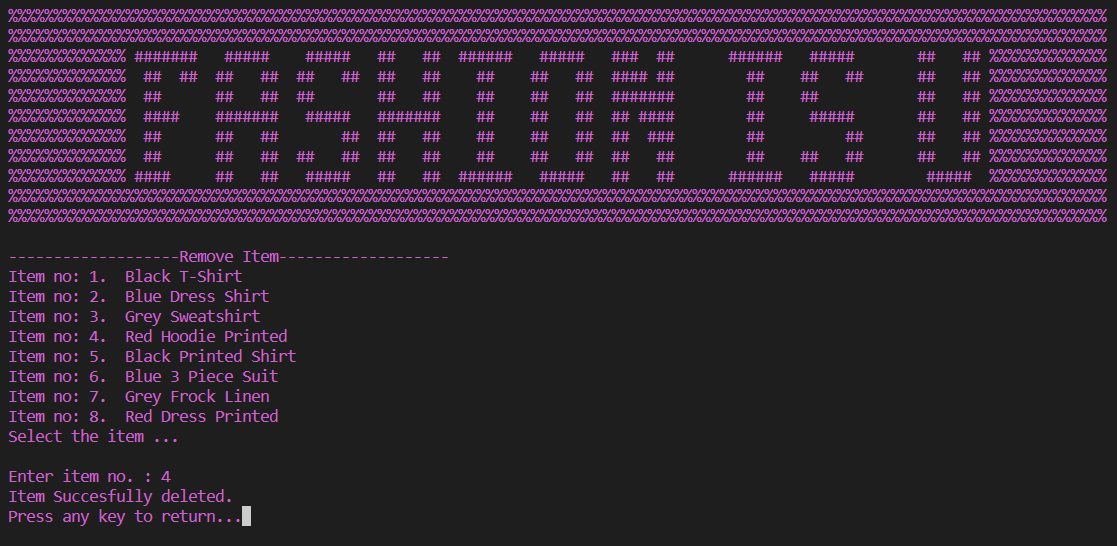
**Figure 21: Addition in Female Items Page**

****

**Figure 22: Change Stock Page**

****

**Figure 23: Check Reviews Page**

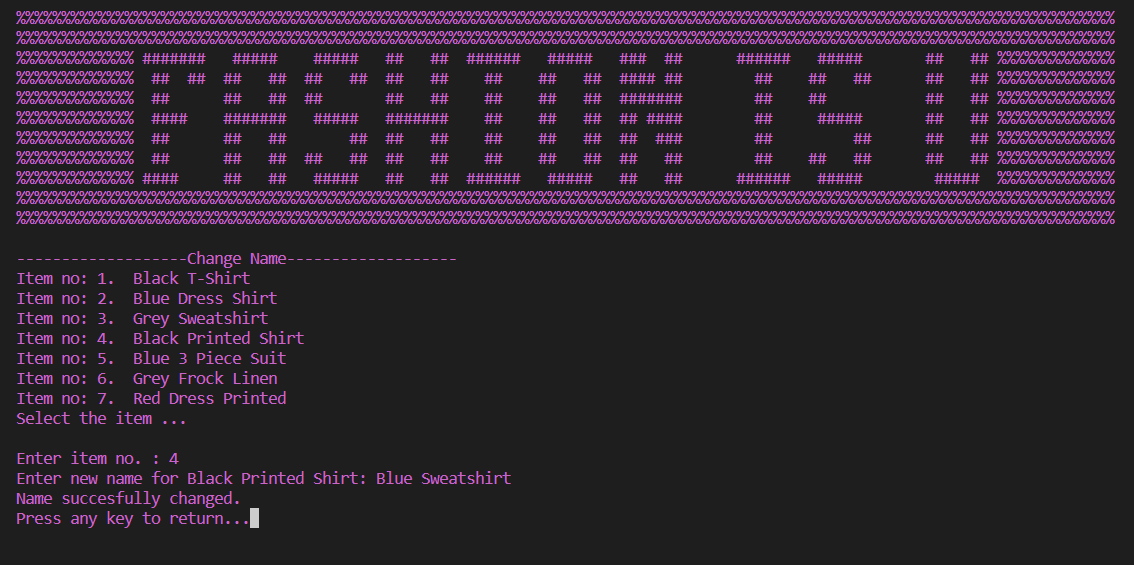
****

**Figure 24: Delete an Item Page**

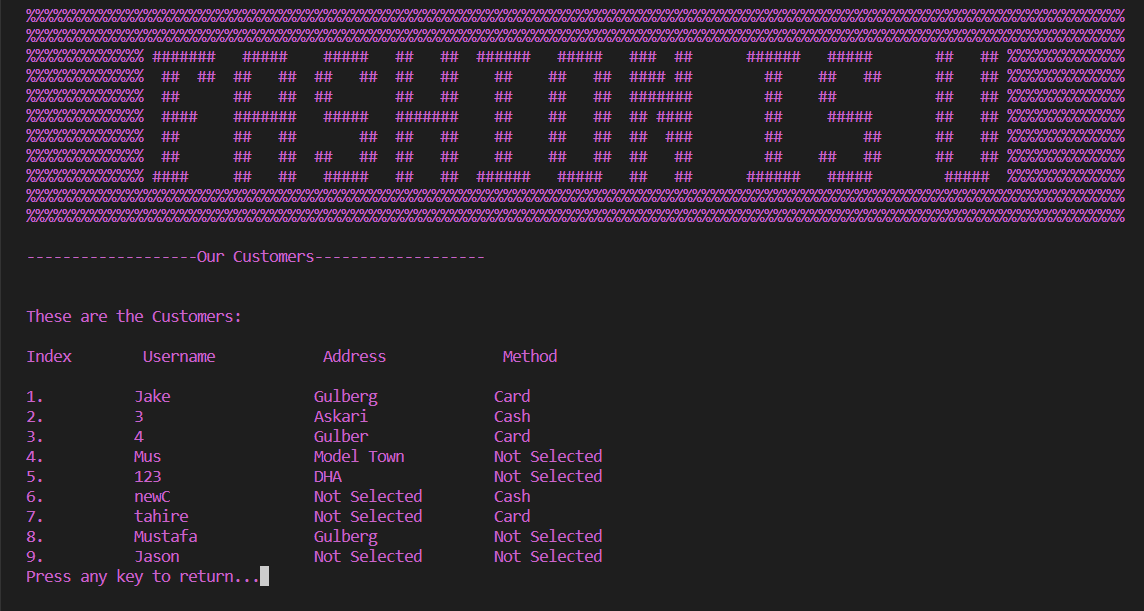
A screenshot of a computer screen

Description automatically generated

**Figure 25: Change Contact Us Information**



**Figure 26: Change Name of an Item**



**Figure 27: See Current Customers**

A screenshot of a computer screen

Description automatically generated

**Figure 28: Add Delivery Area**

**A screenshot of a computer

Description automatically generated**

**Figure 29: Remove Delivery Area**

# Data Structures (Parallel Arrays)

Credentials

* + string username[range]; (idx)
  + string password[range]; (idx)
  + string role[range]; (idx)

   Customer Related

* + string customerArr[range]; (cusIndex)
  + int quantforMen[range][30]; (cusIndex) (menq)
  + int quantforWomen[range][30]; (cusIndex) (womenq)
  + int billPaidcount[range]; (cusIndex)
  + int totalM[range]; (cusIndex)
  + int totalW[range]; (cusIndex)
  + int finalTotal[range]; (cusIndex)
  + string reviews[range]; (cusIndex)
  + string userArea[range]; (cusIndex)
  + string delivery[range]; (cusIndex)
  + int cardno[range]; (cusIndex)

   Store Related

* + string arrM[range]; (menq)
  + int priceM[range]; (menq)
  + int availableM[range]; (menq)
  + string arrW[range]; (womenq)
  + int priceW[range]; (womenq)
  + int availableW[range]; (womenq)
  + string deliveryAreas[range]; (areas)

# Function Prototypes

//prints the header

* void printHeader();

// prints employee menu

* string employeemenu();

// print customermenu

* string customermenu();

// these are customers functions

* void printMitems(int availableM[], int &menq, int cardindex, int quantforMen[][30]);
* void printWitems(int availableW[], int &womenq, int cardindex, int quantforWomen[][30]);
* void bill(string arrM[], string arrW[], int totalM[], int totalW[], int priceM[], int priceW[], int menq, int womenq, int cardindex, int finalTotal[], string delivery[], string userArea[], bool deliveryop, bool delArea, int quantforMen[][30], int quantforWomen[][30]);
* void PayBill(int totalM[], int totalW[], int priceM[], int priceW[], int availableM[], int availableW[], int menq, int womenq, bool &delArea, bool &deliveryop, bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30], int billPaidcount[]);
* void cart(string arrM[], string arrW[], int menq, int womenq, int cardindex, int quantforMen[][30], int quantForWomen[][30]);
* void status(bool billpaid, int billPaidcount[], int cardindex);
* void updatequantity(string arrM[], string arrW[], int menq, int womenq, int availableM[], int availableW[], int counter, int cardindex, int quantforMen[][30], int quantforWomen[][30]);

// these are related to sign in and sign up and finding of indexes

* string signupMenu(string username[], string &name, string &password1, int range);
* string signinMenu(string username[], string password[], string role[], int &cardindex, int range, int idx, int &reviewindex, int cusIndex, string customerArr[]);
* bool checkUser(string name, string username[], int range);
* int usernameInd(string name, int idx, string username[]);
* // these are some more customers functions
* void deliveryoptions(bool &deliveryop, string delivery[], int cardindex, int cardno[]);
* void deliveryArea(int areas, string deliveryAreas[], string userArea[], bool &delArea, int cardindex);
* void resetCart(int menq, int womenq, int cardindex, int quantforMen[][30], int quantforWomen[][30]);
* void leaveReview(string reviews[], int &reviewindex, int cardindex);

// these are admin functions

* string employeeGenderPage();
* void employeeMitems(string arrM[], int priceM[], int availableM[], int menq);
* void employeeWitems(string arrW[], int priceW[], int availableW[], int womenq);
* void returnforAll();
* void addMitem(int &menq, string arrM[], int priceM[], int availableM[]);
* void addWitem(int &womenq, string arrW[], int priceW[], int availableW[]);
* void changeStock(int menq, string arrM[], int availableM[], string arrW[], int womenq, int availableW[]);
* void checkReviews(string reviews[], int cusCount, string customerArr[], int reviewindex, int cusIndex);
* void changeName(int menq, int womenq, string arrM[], string arrW[]);
* void removeItem(int &menq, int &womenq, string arrM[], string arrW[], int priceM[], int priceW[], int availableM[], int availableW[]);
* void seeCustomer(int cusIndex, string userArea[], string delivery[], string customerArr[]);
* void addDeliveryArea(int &areas, string deliveryAreas[]);
* void removeAddress(int &areas, string deliveryAreas[]);

// these are for user inteface during sign in and sign up

* void clearScreen();
* void signinWindow(string &name, string &password1);
* void signupWindow(string &name, string &password1);
* string takeRole();
* void congratsforSignup();
* void congratsforSignin();

// these functions mostly take input

* int takeNumToAdd();
* string takeNametoAdd(int counter);
* string takePricetoAdd();
* int makePriceAccordingToCriteria(int price, string convertprice);
* string takeQStockToAdd();
* int makeQStockAccordingtoCriteria(int available, string convertstock);
* string takeStock();
* string newStockforWomen(int idx, string arrW[]);
* string newStockforMen(int idx, string arrM[]);
* int makeValueAccToCriteria(int value, string convert);

// this shows reviews

* void showreviews(string reviews[], int cusCount, string customerArr[], int &counter, int cusIndex);
* void newNameforWomen(int idx, int var, string arrW[]);
* void newNameforMen(int idx, int var, string arrM[]);
* // this shows the list of customers
* void showCustomersList(int cusIndex, string userArea[], string delivery[], string customerArr[], int &counter);

// these are for validation of addition in number of addresses

* string takeNumberofAddress();
* int restrictNumberofAddress(int number);
* string takeNumberofAddressToRemove();
* string takeitem();
* // these are for validation of quantity bought
* int restrictQforMen(int quantity, int availableM[], int idx, string convert);
* string takeQuantityforMen(int men, int availableM[], int idx);
* int restrictQforWomen(int quantity, int availableW[], int idx, string convert);
* string takeQuantityforWomen(int women, int availableW[], int idx);

// these are related to print bill funtion mentioned above

* void printBillForMen(int menq, int totalM[], int priceM[], int &sumM, string arrM[], int cardindex, int quantforMen[][30]);
* void printBillForWomen(int womenq, int totalW[], int priceW[], int &sumW, string arrW[], int cardindex, int quantforWomen[][30]);
* string billRem1(bool deliveryop, string delivery[], int cardindex);
* string billRem1(bool delArea, string delivery[], int cardindex);
* int findTotalForBill(int menq, int womenq, int totalM[], int priceM[], int totalW[], int priceW[], int cardindex, int quantforMen[][30], int quantforWomen[][30]);
* string takeChoiceForBill();
* string reasonsForBill(bool deliveryop, bool delArea);
* string billPaid(int menq, int availableM[], int womenq, int availableW[], bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30]);
* string takeDelArea();
* int restrictCard(int &a, bool &deliveryop, int cardindex, int cardno[], string convert);
* string payOp();

// these are related to customer info

* void contactForCustomer(string phoneN, string email);
* void setContactInfo(string &phoneN, string &email);
* void changeInfo(string &phoneN, string &email);
* int findCustomerIndex(string name, int cusIndex, string customerArr[]);

// these are for validation

* bool validateint(string convert);
* int strToInt(string convert);
* string intToStr(int num);
* bool checkingForcomma(string sen);
* bool checkingforInteger(string sen);
* bool checkforEmpty(string sen);
* bool checkingforAtthesymbol(string sen);
* bool checkingforspace(string sen);
* string retrictPassword(string sen);
* string restrictAddressName(string address);
* string restrictnewNameforCloth(string sen);
* string signupName(string name);

// these are for saving records

* void saveRecordsofCred(int range, string username[], string password[], string role[], int idx, int cusIndex, int cardindex);
* void saveCustomerInfo(string customerArr[], int billPaidcount[], int totalM[], int totalW[], int finalTotal[], string userArea[], string delivery[], int cardno[], int cusIndex, string reviews[]);
* void saveRecordsofWomenitems(string arrW[], int priceW[], int availableW[], int womenq);
* void saveRecordsofMenitems(string arrM[], int priceM[], int availableM[], int menq);
* void saveAdresses(int areas, string deliveryAreas[]);
* void saveContactInfo(string phoneN, string email);

// these are for loading those records

* void retrieveCredentialsRec(int range, string username[], string password[], string role[], int &idx, int cusIndex, int cardindex);
* void retrieveRecOfMenitems(string arrM[], int priceM[], int availableM[], int &menq);
* void retrieveRecOfWomenitems(string arrW[], int priceW[], int availableW[], int &womenq);
* void retrieveAdress(int &areas, string deliveryAreas[]);
* void retriveinfoCustomer(string customerArr[], int billPaidcount[], int totalM[], int totalW[], int finalTotal[], string userArea[], string delivery[], int cardno[], int &cusIndex, string reviews[]);
* void retrieveContactInfo(string &phoneN, string &email);
* string getField(string record, int field);

# Function Working Flow

This is a diagram of the working flow of the Business Application Fashion Is U.

**Fashion Is U Flow diagram:**



# Complete Code of Fashion Is U

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include <fstream>

using namespace std;

// prints the header

void printHeader();

// prints employee menu

string employeemenu();

// print customermenu

string customermenu();

// these are customers functions

void printMitems(int availableM[], int &menq, int cardindex, int quantforMen[][30]);

void printWitems(int availableW[], int &womenq, int cardindex, int quantforWomen[][30]);

void bill(string arrM[], string arrW[], int totalM[], int totalW[], int priceM[], int priceW[], int menq, int womenq, int cardindex, int finalTotal[], string delivery[], string userArea[], bool deliveryop, bool delArea, int quantforMen[][30], int quantforWomen[][30]);

void PayBill(int totalM[], int totalW[], int priceM[], int priceW[], int availableM[], int availableW[], int menq, int womenq, bool &delArea, bool &deliveryop, bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30], int billPaidcount[]);

void cart(string arrM[], string arrW[], int menq, int womenq, int cardindex, int quantforMen[][30], int quantForWomen[][30]);

void status(bool billpaid, int billPaidcount[], int cardindex);

void updatequantity(string arrM[], string arrW[], int menq, int womenq, int availableM[], int availableW[], int counter, int cardindex, int quantforMen[][30], int quantforWomen[][30]);

// these are related to sign in and sign up and finding of indexes

string signupMenu(string username[], string &name, string &password1, int range);

string signinMenu(string username[], string password[], string role[], int &cardindex, int range, int idx, int &reviewindex, int cusIndex, string customerArr[]);

bool checkUser(string name, string username[], int range);

int usernameInd(string name, int idx, string username[]);

// these are some more customers functions

void deliveryoptions(bool &deliveryop, string delivery[], int cardindex, int cardno[]);

void deliveryArea(int areas, string deliveryAreas[], string userArea[], bool &delArea, int cardindex);

void resetCart(int menq, int womenq, int cardindex, int quantforMen[][30], int quantforWomen[][30]);

void leaveReview(string reviews[], int &reviewindex, int cardindex);

// these are admin functions

string employeeGenderPage();

void employeeMitems(string arrM[], int priceM[], int availableM[], int menq);

void employeeWitems(string arrW[], int priceW[], int availableW[], int womenq);

void returnforAll();

void addMitem(int &menq, string arrM[], int priceM[], int availableM[]);

void addWitem(int &womenq, string arrW[], int priceW[], int availableW[]);

void changeStock(int menq, string arrM[], int availableM[], string arrW[], int womenq, int availableW[]);

void checkReviews(string reviews[], string customerArr[], int reviewindex, int cusIndex);

void changeName(int menq, int womenq, string arrM[], string arrW[]);

void removeItem(int &menq, int &womenq, string arrM[], string arrW[], int priceM[], int priceW[], int availableM[], int availableW[]);

void seeCustomer(int cusIndex, string userArea[], string delivery[], string customerArr[]);

void addDeliveryArea(int &areas, string deliveryAreas[]);

void removeAddress(int &areas, string deliveryAreas[]);

// these are for user inteface during sign in and sign up

void clearScreen();

void signinWindow(string &name, string &password1);

void signupWindow(string &name, string &password1);

string takeRole();

void congratsforSignup();

void congratsforSignin();

// these functions mostly take input

int takeNumToAdd();

string takeNametoAdd(int counter);

string takePricetoAdd();

int makePriceAccordingToCriteria(int price, string convertprice);

string takeQStockToAdd();

int makeQStockAccordingtoCriteria(int available, string convertstock);

string takeStock();

string newStockforWomen(int idx, string arrW[]);

string newStockforMen(int idx, string arrM[]);

int makeValueAccToCriteria(int value, string convert);

// this shows reviews

void showreviews(string reviews[], string customerArr[], int &counter, int cusIndex);

void newNameforWomen(int idx, int var, string arrW[]);

void newNameforMen(int idx, int var, string arrM[]);

// this shows the list of customers

void showCustomersList(int cusIndex, string userArea[], string delivery[], string customerArr[], int &counter);

// these are for validation of addition in number of addresses

string takeNumberofAddress();

int restrictNumberofAddress(int number);

string takeNumberofAddressToRemove();

string takeitem();

// these are for validation of quantity bought

int restrictQforMen(int quantity, int availableM[], int idx, string convert);

string takeQuantityforMen(int men, int availableM[], int idx);

int restrictQforWomen(int quantity, int availableW[], int idx, string convert);

string takeQuantityforWomen(int women, int availableW[], int idx);

// these are related to print bill funtion mentioned above

void printBillForMen(int menq, int totalM[], int priceM[], int &sumM, string arrM[], int cardindex, int quantforMen[][30]);

void printBillForWomen(int womenq, int totalW[], int priceW[], int &sumW, string arrW[], int cardindex, int quantforWomen[][30]);

string billRem1(bool deliveryop, string delivery[], int cardindex);

string billRem1(bool delArea, string delivery[], int cardindex);

int findTotalForBill(int menq, int womenq, int totalM[], int priceM[], int totalW[], int priceW[], int cardindex, int quantforMen[][30], int quantforWomen[][30]);

string takeChoiceForBill();

string reasonsForBill(bool deliveryop, bool delArea);

string billPaid(int menq, int availableM[], int womenq, int availableW[], bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30]);

string takeDelArea();

int restrictCard(int &a, bool &deliveryop, int cardindex, int cardno[], string convert);

string payOp();

// these are related to customer info

void contactForCustomer(string phoneN, string email);

void setContactInfo(string &phoneN, string &email);

void changeInfo(string &phoneN, string &email);

int findCustomerIndex(string name, int cusIndex, string customerArr[]);

// these are for validation

int strToInt(string convert);

string intToStr(int num);

bool checkingForcomma(string sen);

bool checkingforInteger(string sen);

bool checkforEmpty(string sen);

bool checkingforAtthesymbol(string sen);

bool checkingforspace(string sen);

string retrictPassword(string sen);

string restrictAddressName(string address);

string restrictnewNameforCloth(string sen);

string signupName(string name);

// these are for saving records

void saveRecordsofCred(string username[], string password[], string role[], int idx);

void saveCustomerInfo(string customerArr[], int billPaidcount[], string userArea[], string delivery[], int cusIndex, string reviews[]);

void saveRecordsofWomenitems(string arrW[], int priceW[], int availableW[], int womenq);

void saveRecordsofMenitems(string arrM[], int priceM[], int availableM[], int menq);

void saveAdresses(int areas, string deliveryAreas[]);

void saveContactInfo(string phoneN, string email);

// these are for loading those records

void retrieveCredentialsRec(string username[], string password[], string role[], int &idx);

void retrieveRecOfMenitems(string arrM[], int priceM[], int availableM[], int &menq);

void retrieveRecOfWomenitems(string arrW[], int priceW[], int availableW[], int &womenq);

void retrieveAdress(int &areas, string deliveryAreas[]);

void retriveinfoCustomer(string customerArr[], int billPaidcount[], string userArea[], string delivery[], int &cusIndex, string reviews[]);

void retrieveContactInfo(string &phoneN, string &email);

string getField(string record, int field);

main()

{

system("color 0F");

//Credentials

int range = 30;

string username[range];

string password[range];

string role[range];

int idx = 0;

int cusIndex = 0;

int cardindex;

//Customer Related

string customerArr[range];

int quantforMen[range][30];

int quantforWomen[range][30];

int billPaidcount[range];

int totalM[range];

int totalW[range];

int finalTotal[range];

string reviews[range];

int reviewindex;

string customer[range];

string userArea[range];

int cusCount = 0;

string delivery[range];

int cardno[range];

//Store Related

int menq = 0, womenq = 0;

int areas = 0;

bool deliveryop = false, delArea = false, billpaid = false;

string name, password1;

string phoneN = "0423-123456", email = "fashionisu@gmail.com";

string arrM[range];

int priceM[range];

int availableM[range];

string arrW[range];

int priceW[range];

int availableW[range];

string deliveryAreas[range];

// retrieving from saved files

retrieveCredentialsRec(username, password, role,idx);

retrieveRecOfMenitems(arrM, priceM, availableM, menq);

retrieveRecOfWomenitems(arrW, priceW, availableW, womenq);

retrieveAdress(areas, deliveryAreas);

retriveinfoCustomer(customerArr, billPaidcount,userArea,delivery, cusIndex, reviews);

retrieveContactInfo(phoneN, email);

string op;

while (true)

{

clearScreen();

cout << "----------------Log In Or Sign Up Menu-------------" << endl; // Sub Menu Before Employee Menu or Customer Menu

cout << "1. Sign in" << endl;

cout << "2. Sign up" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice.. : ";

cin >> op;

if (op == "1")

{

clearScreen();

string role1 = signinMenu(username, password, role, cardindex, range, idx, reviewindex, cusIndex, customerArr); // recieves role

if (role1 == "Employee" || role1 == "employee")

{

string employeechoice;

system("color 05");

while (true)

{

clearScreen();

cout << endl;

cout << "----------Employee Menu-----------" << endl;

employeechoice = employeemenu();

cout << endl;

if (employeechoice == "1")

{

clearScreen();

cout << endl;

cout << "----------Employee Gender Page-----------" << endl;

string gender = employeeGenderPage();

cout << endl;

if (gender == "1")

{

clearScreen();

cout << endl;

cout << "-------------------Employee Male Items-------------------" << endl;

employeeMitems(arrM, priceM, availableM, menq);

}

else if (gender == "2")

{

clearScreen();

cout << endl;

cout << "-------------------Employee Female Items-------------------" << endl;

employeeWitems(arrW, priceW, availableW, womenq);

}

else

{

Sleep(300);

}

}

else if (employeechoice == "2")

{

clearScreen();

cout << endl;

cout << "----------Employee Gender Page-----------" << endl;

string gender = employeeGenderPage();

cout << endl;

if (gender == "1")

{

clearScreen();

cout << endl;

cout << "-------------------Addition In Male Items-------------------" << endl;

addMitem(menq, arrM, priceM, availableM);

}

else if (gender == "2")

{

clearScreen();

cout << endl;

cout << "-------------------Addition In Female Items-------------------" << endl;

addWitem(womenq, arrW, priceW, availableW);

}

else

{

Sleep(300);

}

}

else if (employeechoice == "3")

{

clearScreen();

cout << endl;

cout << "-------------------Change Stock-------------------" << endl;

cout << left << setw(12) << "Item no:" << setw(25) << "Name" << setw(10) << "Available Articles: " << endl;

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

{

cout << left << setw(12) << to\_string(idx + 1) + ". " << setw(25) << arrM[idx] << setw(10) << availableM[idx] << endl; //print Men items

}

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

{

cout << left << setw(12) << to\_string(idx + 1 + menq) + ". " << setw(25) << arrW[idx] << setw(10) << availableW[idx] << endl; // print Women items

}

changeStock(menq, arrM, availableM, arrW, womenq, availableW);

}

else if (employeechoice == "4")

{

clearScreen();

cout << endl;

cout << "-------------------Show Reviews-------------------" << endl;

checkReviews(reviews, customerArr, reviewindex, cusIndex);

}

else if (employeechoice == "6")

{

clearScreen();

cout << endl;

cout << "-------------------Change Name-------------------" << endl;

changeName(menq, womenq, arrM, arrW);

}

else if (employeechoice == "5")

{

clearScreen();

cout << endl;

cout << "-------------------Remove Item-------------------" << endl;

removeItem(menq, womenq, arrM, arrW, priceM, priceW, availableM, availableW);

}

else if (employeechoice == "7")

{

clearScreen();

cout << endl;

cout << "-------------------Our Customers-------------------" << endl;

seeCustomer(cusIndex, userArea, delivery, customerArr);

}

else if (employeechoice == "8")

{

clearScreen();

cout << endl;

cout << "-------------------Add Delivery Area-------------------" << endl;

cout << "These are the delivery Areas: " << endl;

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

{

cout << idx + 1 << ". " << deliveryAreas[idx] << endl;

}

addDeliveryArea(areas, deliveryAreas);

}

else if (employeechoice == "9")

{

clearScreen();

cout << endl;

cout << "-------------------Remove Delivery Area-------------------" << endl;

cout << endl

<< endl;

cout << "Select the number of address to remove: " << endl

<< endl;

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

{

cout << idx + 1 << ". " << deliveryAreas[idx] << endl;

}

removeAddress(areas, deliveryAreas);

}

else if (employeechoice == "10")

{

clearScreen();

cout << endl;

cout << "-------------------Change Contact US Information-------------------" << endl;

cout << endl

<< endl;

cout << "Our Contact Number: " << phoneN << endl;

cout << "Our Email Address: " << email << endl;

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

changeInfo(phoneN, email);

}

else if (employeechoice == "11")

{

system("color 0F");

break;

}

else

{

cout << "Incorrect Input...";

Sleep(300);

}

}

}

else if (role1 == "Customer" || role1 == "customer")

{

system("color 01");

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

{

if (quantforMen[cardindex][x] > availableM[x])

{

quantforMen[cardindex][x] = 0;

}

}

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

{

if (quantforWomen[cardindex][x] > availableW[x])

{

quantforWomen[cardindex][x] = 0;

}

}

string customerchoice;

while (true)

{

clearScreen();

cout << endl;

cout << "----------Customer Menu-----------" << endl;

customerchoice = customermenu();

if (customerchoice == "1")

{

clearScreen();

cout << endl;

cout << "-------------------Gender Page-------------------" << endl;

string gender = employeeGenderPage();

if (gender == "1")

{

clearScreen();

cout << endl;

cout << "-------------------Male Items Page-------------------" << endl;

cout << endl

<< endl;

cout << "Enter your choice: " << endl;

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

{ // print men items

if (availableM[idx] == 0)

{

cout << left << setw(12) << to\_string(idx + 1) + ". " << setw(25) << arrM[idx] << setw(20) << " Rs " + to\_string(priceM[idx]) << "\t (Out of Stock)" << endl;

}

else

{

cout << left << setw(12) << to\_string(idx + 1) + ". " << setw(25) << arrM[idx] << setw(20) << " Rs " + to\_string(priceM[idx]) << endl;

}

}

cout << endl;

printMitems(availableM, menq, cardindex, quantforMen);

}

else if (gender == "2")

{

clearScreen();

cout << endl;

cout << "-------------------Women Items Page-------------------" << endl;

cout << endl

<< endl;

cout << "Enter your choice: " << endl;

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

{ // print women items

if (availableW[idx] == 0)

{

cout << left << setw(12) << to\_string(idx + 1) + ". " << setw(25) << arrW[idx] << setw(20) << " Rs " + to\_string(priceW[idx]) << "\t (Out of Stock)" << endl;

}

else

{

cout << left << setw(12) << to\_string(idx + 1) + ". " << setw(25) << arrW[idx] << setw(20) << " Rs " + to\_string(priceW[idx]) << endl;

}

}

printWitems(availableW, womenq, cardindex, quantforWomen);

}

else

{

Sleep(300);

}

}

else if (customerchoice == "5")

{

clearScreen();

cout << endl;

cout << "----------------------Print The Bill----------------------" << endl

<< endl;

cout << left << setw(25) << " Item" << setw(20) << "Price" << setw(20) << "Quantity" << setw(20) << "Total Price" << endl;

bill(arrM, arrW, totalM, totalW, priceM, priceW, menq, womenq, cardindex, finalTotal, delivery, userArea, deliveryop, delArea, quantforMen, quantforWomen);

}

else if (customerchoice == "7")

{

clearScreen();

cout << endl;

cout << "----------------------Pay The Bill----------------------" << endl

<< endl;

PayBill(totalM, totalW, priceM, priceW, availableM, availableW, menq, womenq, delArea, deliveryop, billpaid, cardindex, quantforMen, quantforWomen, billPaidcount);

}

else if (customerchoice == "3")

{

clearScreen();

cout << endl;

cout << "-------------------------Payment Options------------------------" << endl;

cout << endl;

deliveryoptions(deliveryop, delivery, cardindex, cardno);

}

else if (customerchoice == "2")

{

clearScreen();

cout << endl;

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

cout << endl;

cout << left << setw(25) << " Item " << setw(20) << "Quantity " << endl;

cart(arrM, arrW, menq, womenq, cardindex, quantforMen, quantforWomen);

}

else if (customerchoice == "4")

{

clearScreen();

cout << endl;

cout << "----------------------Select Delivery Area-----------------------" << endl;

cout << endl;

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

{

cout << idx + 1 << ". " << deliveryAreas[idx] << endl;

}

deliveryArea(areas, deliveryAreas, userArea, delArea, cardindex);

}

else if (customerchoice == "6")

{

clearScreen();

cout << endl;

cout << "----------------------Update Quantity-----------------------" << endl;

cout << endl

<< endl;

int counter = 0;

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

{

if (quantforMen[cardindex][idx] != 0) //printing the men items bought by the customer

{

cout << "Item no: " << idx + 1 << ". " << arrM[idx] << " Quantity: " << quantforMen[cardindex][idx] << endl;

counter++;

}

}

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

{

if (quantforWomen[cardindex][idx] != 0) //printing the women items bought by the customer

{

cout << "Item no: " << idx + 1 + menq << ". " << arrW[idx] << " Quantity: " << quantforWomen[cardindex][idx] << endl;

counter++;

}

}

cout << endl

<< endl;

cout << "Select the item to change the quantity of.." << endl

<< endl;

updatequantity(arrM, arrW, menq, womenq, availableM, availableW, counter, cardindex, quantforMen, quantforWomen);

}

else if (customerchoice == "8")

{

clearScreen();

cout << endl;

cout << "----------------------Check Status Of Order-----------------------" << endl;

cout << endl

<< endl;

status(billpaid, billPaidcount, cardindex);

}

else if (customerchoice == "11")

{

billpaid = false;

system("color 0F");

break;

}

else if (customerchoice == "9")

{

clearScreen();

cout << endl;

cout << "----------------------Leave a Review-----------------------" << endl;

cout << endl

<< endl;

leaveReview(reviews, reviewindex, cardindex);

}

else if (customerchoice == "10")

{

clearScreen();

cout << endl;

cout << "----------------------Contact Us-----------------------" << endl;

cout << endl

<< endl;

contactForCustomer(phoneN, email);

}

else

{

cout << "Incorrect Input.." << endl;

Sleep(300);

}

}

}

}

else if (op == "2")

{

clearScreen();

string role1 = signupMenu(username, name, password1, range); //recieves role from signupMenu()

if ((role1 == "Employee") || (role1 == "Customer") || (role1 == "employee") || (role1 == "customer"))

{

if (role1 == "Customer" || role1 == "customer")

{

customer[cusCount] = name;

cusCount++;

}

if (role1 == "customer")

{

role1 = "Customer";

}

else if (role1 == "employee")

{

role1 = "Employee";

}

role[idx] = role1;

username[idx] = name;

password[idx] = password1;

idx++;

if (role1 == "Customer")

{

customerArr[cusIndex] = name; //puts name in customerArr

cusIndex++;

}

congratsforSignup();

}

else

{

cout << "Incorrect role...";

Sleep(300);

}

}

else if (op == "3")

{

// saving records

saveCustomerInfo(customerArr,billPaidcount,userArea,delivery, cusIndex, reviews);

saveRecordsofCred(username, password, role, idx);

saveRecordsofMenitems(arrM,priceM, availableM,menq);

saveRecordsofWomenitems(arrW, priceW, availableW, womenq);

saveAdresses(areas, deliveryAreas);

saveContactInfo(phoneN, email);

break;

}

else

{

cout << "Incorrect Input...";

Sleep(300);

}

}

}

// saving records of credentials

void saveRecordsofCred(string username[], string password[], string role[], int idx)

{

fstream file;

file.open("Credentials.txt", ios::out);

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

{

file << role[x];

file << ",";

file << username[x];

file << ",";

file << password[x];

if (x != idx-1)

{

file<<'\n';

}

}

file.close();

}

// saves data related to customer

void saveCustomerInfo(string customerArr[], int billPaidcount[], string userArea[], string delivery[], int cusIndex, string reviews[])

{

fstream file;

file.open("CustomerDetails.txt", ios::out);

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

{

file << customerArr[x];

file << ",";

file << userArea[x];

file << ",";

file << delivery[x];

file << ",";

file << billPaidcount[x];

file << ",";

file << reviews[x];

if (x != cusIndex-1)

{

file<<'\n';

}

}

file.close();

}

// this will retrieve customer info

void retriveinfoCustomer(string customerArr[], int billPaidcount[],string userArea[], string delivery[], int &cusIndex, string reviews[])

{

string record="";

fstream file;

file.open("CustomerDetails.txt", ios::in);

while(!file.eof())

{

getline(file,record);

customerArr[cusIndex] = getField(record,1);

userArea[cusIndex] = getField(record,2);

delivery[cusIndex] = getField(record,3);

billPaidcount[cusIndex] = stoi(getField(record,4));

reviews[cusIndex] = getField(record,5);

cusIndex=cusIndex+1;

}

file.close();

}

// save store record of Men items

void saveRecordsofMenitems(string arrM[], int priceM[], int availableM[], int menq)

{

fstream file;

file.open("Menitems.txt", ios::out);

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

{

file << arrM[x];

file << ",";

file << priceM[x];

file << ",";

file << availableM[x];

if(x != menq-1)

{

file << '\n';

}

}

file.close();

}

// save store records of Women items

void saveRecordsofWomenitems(string arrW[], int priceW[], int availableW[], int womenq)

{

fstream file;

file.open("Womenitems.txt", ios::out);

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

{

file << arrW[x];

file << ",";

file << priceW[x];

file << ",";

file << availableW[x];

if(x != womenq-1)

{

file << '\n';

}

}

file.close();

}

// save change in contact information

void saveContactInfo(string phoneN, string email)

{

fstream file;

file.open("ContactInfo.txt", ios::out);

file << phoneN;

file << ",";

file << email;

file.close();

}

// retrieves contact us informations

void retrieveContactInfo(string &phoneN, string &email)

{

string record="";

fstream file;

file.open("ContactInfo.txt", ios::in);

getline(file,record);

phoneN=getField(record,1);

email=getField(record,2);

file.close();

}

// saves the addresses offered by the store

void saveAdresses(int areas, string deliveryAreas[])

{

fstream file;

file.open("DeliveryArea.txt", ios::out);

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

{

file << deliveryAreas[x];

if(x!=areas-1)

{

file << endl;

}

}

file.close();

}

// it retrieves the credentials

void retrieveCredentialsRec(string username[], string password[], string role[], int &idx)

{

string record="";

fstream file;

file.open("Credentials.txt", ios::in);

if(file.fail())

{

cout << "Error opening the file.";

}

while(!file.eof())

{

getline(file, record);

role[idx]=getField(record,1);

username[idx]=getField(record,2);

password[idx]=getField(record,3);

idx=idx+1;

}

file.close();

}

void retrieveRecOfMenitems(string arrM[], int priceM[], int availableM[], int &menq)

{

string record="";

fstream file;

file.open("Menitems.txt", ios::in);

if(file.fail())

{

cout << "Error opening the file.";

}

while(!file.eof())

{

getline(file,record);

arrM[menq]=getField(record,1);

priceM[menq]=stoi(getField(record,2));

availableM[menq]=stoi(getField(record,3));

menq++;

}

file.close();

}

void retrieveRecOfWomenitems(string arrW[], int priceW[], int availableW[], int &womenq)

{

string record="";

fstream file;

file.open("Womenitems.txt", ios::in);

if(file.fail())

{

cout << "Error opening the file.";

}

while(!file.eof())

{

getline(file,record);

arrW[womenq]=getField(record,1);

priceW[womenq]=stoi(getField(record,2));

availableW[womenq]=stoi(getField(record,3));

womenq++;

}

file.close();

}

void retrieveAdress(int &areas, string deliveryAreas[])

{

string record="";

fstream file;

file.open("DeliveryArea.txt", ios::in);

if(file.fail())

{

cout << "Error opening the file.";

}

while(!file.eof())

{

getline(file,record);

deliveryAreas[areas]=record;

areas++;

}

file.close();

}

// it finds field which is present in csv

string getField(string record, int field)

{

int comma=1;

string result="";

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

{

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

{

comma=comma+1;

}

else if(comma==field)

{

result=result+record[x];

}

}

return result;

}

// prints the header of the store

void printHeader()

{

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% ####### ##### ##### ## ## ###### ##### ### ## ###### ##### ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% ## ## ## ## ## ## ## ## ## ## ## #### ## ## ## ## ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% ## ## ## ## ## ## ## ## ## ####### ## ## ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% #### ####### ##### ####### ## ## ## ## #### ## ##### ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% ## ## ## ## ## ## ## ## ## ## ### ## ## ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%% #### ## ## ##### ## ## ###### ##### ## ## ###### ##### ##### %%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% " << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% " << endl;

}

string signinMenu(string username[], string password[], string role[], int &cardindex, int range, int idx, int &reviewindex, int cusIndex, string customerArr[])

{

string name;

string password1;

string role1;

int index;

signinWindow(name, password1);

if (checkUser(name, username, range))

{

index = usernameInd(name, idx, username);

cardindex = findCustomerIndex(name, cusIndex, customerArr);

reviewindex = index;

if (password1 == password[index])

{

role1 = role[index];

congratsforSignin();

}

else

{

cout << "User Not Found.";

Sleep(300);

}

}

else

{

cout << "User does not exist.";

Sleep(300);

}

return role1;

}

void signupWindow(string &name, string &password1)

{

cout << endl;

cout << "--------Sign Up----------";

cout << endl

<< endl;

cout << "Enter Username: ";

cin.clear();

cin.sync();

getline(cin,name);

name=signupName(name);

cout << "Enter Password (Must be 6-digits): ";

cin.clear();

cin.sync();

getline(cin,password1);

password1=retrictPassword(password1);

}

string takeRole()

{

string role1;

cout << "Enter Role (Employee or Customer): ";

cin >> role1;

cout << endl;

return role1;

}

void congratsforSignup()

{

cout << endl;

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

cout << endl;

cout << "You have successfully signed up." << endl;

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

getch();

}

void congratsforSignin()

{

cout << endl;

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

cout << endl;

cout << "You have successfully signed in." << endl;

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

getch();

}

string signupMenu(string username[], string &name, string &password1, int range)

{

string role1;

signupWindow(name, password1);

if (checkUser(name, username, range))

{

cout << "Username Already Taken..";

Sleep(300);

clearScreen();

role1 = "";

}

else

{

role1 = takeRole();

}

return role1;

}

// finds the customer index from their name

int findCustomerIndex(string name, int cusIndex, string customerArr[])

{

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

{

if (name == customerArr[x])

{

return x;

}

}

}

// finds the user index from their name

int usernameInd(string name, int idx, string username[])

{

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

{

if (name == username[x])

{

return x;

}

}

}

// verifies if the user exist or not

bool checkUser(string name, string username[], int range)

{

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

{

if (name == username[x])

{

return true;

}

}

return false;

}

// prints employee menu

string employeemenu()

{

string employeechoice;

cout << endl

<< endl;

cout << "Enter one of the following options number..." << endl;

cout << "1. \t View List Of Clothes" << endl;

cout << "2. \t Add an item of Clothing" << endl;

cout << "3. \t Change Available Stock" << endl;

cout << "4. \t Check Reviews" << endl;

cout << "5. \t Delete an item of Clothing" << endl;

cout << "6. \t Change Name of an Item" << endl;

cout << "7. \t See Current Customers" << endl;

cout << "8. \t Add Delivery Area" << endl;

cout << "9. \t Remove a Delivery Area" << endl;

cout << "10. \t Change Contact Us Information" << endl;

cout << "11. \t Log out" << endl;

cout << "Enter your choice: ";

cin >> employeechoice;

return employeechoice;

}

//prints the customer Menu

string customermenu()

{

string customerchoice;

cout << endl

<< endl;

cout << "Enter one of the following options number..." << endl;

cout << "1. \t View List Of Clothes" << endl;

cout << "2. \t View Cart" << endl;

cout << "3. \t Select Payment Options" << endl;

cout << "4. \t Select Delivery Area" << endl;

cout << "5. \t View the Bill" << endl;

cout << "6. \t Change Quantity of an item" << endl;

cout << "7. \t Pay the Bill" << endl;

cout << "8. \t Check Status of Placed Order" << endl;

cout << "9. \t Leave a review." << endl;

cout << "10. \t Contact Us " << endl;

cout << "11. \t Log out" << endl;

cout << "Enter your choice: ";

cin >> customerchoice;

return customerchoice;

}

string takeitem()

{

string item;

cout << "Enter number: ";

cin >> item;

return item;

}

string takeQuantityforMen(int men, int availableM[], int idx)

{

string quantity;

cout << "Available pieces: " << availableM[idx] << endl;

cout << "Enter Quantity of " << men << " : ";

cin >> quantity;

return quantity;

}

string takeQuantityforWomen(int women, int availableW[], int idx)

{

string quantity;

cout << "Available pieces: " << availableW[idx] << endl;

cout << "Enter Quantity of " << women << " : ";

cin >> quantity;

return quantity;

}

int restrictQforMen(int quantity, int availableM[], int idx, string convert)

{

if ((quantity > availableM[idx] || quantity < 0 ) || !checkingforInteger(convert))

{

while ((quantity > availableM[idx] || quantity < 0) || !checkingforInteger(convert))

{

cout << "Not Possible.." << endl;

cout << "Enter again: ";

cin >> convert;

quantity=strToInt(convert);

}

}

return quantity;

}

int restrictQforWomen(int quantity, int availableW[], int idx, string convert)

{

if (quantity > availableW[idx] || quantity < 0 || !checkingforInteger(convert))

{

while (quantity > availableW[idx] || quantity < 0 || !checkingforInteger(convert))

{

cout << "Not Possible.." << endl;

cout << "Enter again: ";

cin >> convert;

quantity=strToInt(convert);

}

}

return quantity;

}

// prints Men items for customers

void printMitems(int availableM[], int &menq, int cardindex, int quantforMen[][30])

{

string convert=takeitem();

int men=strToInt(convert);

if (men > 0 && men <= menq)

{

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

{

if (men == idx + 1)

{

if (availableM[idx] != 0 && quantforMen[cardindex][idx] == 0)

{

string convert = takeQuantityforMen(men, availableM, idx); // variables for validation

int quantity=strToInt(convert);

quantity = restrictQforMen(quantity, availableM, idx, convert);

quantforMen[cardindex][idx] = quantity;

}

else if (quantforMen[cardindex][idx] != 0)

{

cout << "Already bought.." << endl; // if the item is already bought

Sleep(300);

break;

}

else

{

cout << "Not Possible..." << endl;

Sleep(300);

break;

}

}

}

}

else

{

cout << "Incorrect Input...";

Sleep(200);

}

returnforAll();

}

// print Women items for customers

void printWitems(int availableW[], int &womenq, int cardindex, int quantforWomen[][30])

{

string convert = takeitem(); // variable convert for validation

int women=strToInt(convert);

if (women > 0 && women <= womenq)

{

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

{

if (women == idx + 1)

{

if (availableW[idx] != 0 && quantforWomen[cardindex][idx] == 0)

{

string convert = takeQuantityforWomen(women, availableW, idx); // variable for validation

int quantity=strToInt(convert);

quantity = restrictQforWomen(quantity, availableW, idx, convert);

quantforWomen[cardindex][idx] = quantity;

}

else if (quantforWomen[cardindex][idx] != 0)

{

cout << "Already bought.." << endl; // if the items is already bought

Sleep(300);

break;

}

else

{

cout << "Not Possible..." << endl;

Sleep(300);

break;

}

}

}

}

else

{

cout << "Incorrect Input...";

Sleep(200);

}

returnforAll();

}

// prints cart for customers

void cart(string arrM[], string arrW[], int menq, int womenq, int cardindex, int quantforMen[][30], int quantforWomen[][30])

{

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

{

if (quantforMen[cardindex][idx] != 0 && quantforMen[cardindex][idx] <= 1000)

{

cout << left << setw(25) << arrM[idx] << setw(22) << quantforMen[cardindex][idx] << endl; // prints male items bought

}

}

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

{

if (quantforWomen[cardindex][idx] != 0 && quantforWomen[cardindex][idx] <= 1000)

{

cout << left << setw(25) << arrW[idx] << setw(22) << quantforWomen[cardindex][idx] << endl; //print female items bought

}

}

cout << endl << endl;

returnforAll();

}

// this function is called in bill function, it prints the items from the men menu bought

void printBillForMen(int menq, int totalM[], int priceM[], int &sumM, string arrM[], int cardindex, int quantforMen[][30])

{

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

{

totalM[idx] = quantforMen[cardindex][idx] \* priceM[idx];

sumM += totalM[idx];

if (quantforMen[cardindex][idx] != 0)

{

cout << left << setw(25) << arrM[idx] << setw(20) << "Rs " + to\_string(priceM[idx]) << setw(20) << quantforMen[cardindex][idx] << setw(20) << totalM[idx] << endl;

}

}

}

// this function is called in bill function, it prints the items from the womens menu bought

void printBillForWomen(int womenq, int totalW[], int priceW[], int &sumW, string arrW[], int cardindex, int quantforWomen[][30])

{

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

{

totalW[idx] = quantforWomen[cardindex][idx] \* priceW[idx];

sumW += totalW[idx];

if (quantforWomen[cardindex][idx] != 0)

{

cout << left << setw(25) << arrW[idx] << setw(20) << "Rs " + to\_string(priceW[idx]) << setw(20) << quantforWomen[cardindex][idx] << setw(20) << totalW[idx] << endl;

}

}

}

// it gives the remaining part of bill which gives payment method

string billRem1(bool deliveryop, string delivery[], int cardindex)

{

string part1;

if (deliveryop == 1)

{

part1 = "Your Payment Method: " + (delivery[cardindex]);

}

else

{

part1 = "Your Payment Method: (Not selected yet) ";

}

return part1;

}

// it gives the part of the bill which give delivery address of customer

string billRem2(bool delArea, string userArea[], int cardindex)

{

string part2;

if (delArea == 1)

{

part2 = "Your delivery Address: " + userArea[cardindex];

}

else

{

part2 = "Your deliver Address: (Not selected yet) ";

}

return part2;

}

// this function prints the bill

void bill(string arrM[], string arrW[], int totalM[], int totalW[], int priceM[], int priceW[], int menq, int womenq, int cardindex, int finalTotal[], string delivery[], string userArea[], bool deliveryop, bool delArea, int quantforMen[][30], int quantforWomen[][30])

{

int sumM = 0;

int sumW = 0;

printBillForMen(menq, totalM, priceM, sumM, arrM, cardindex, quantforMen);

printBillForWomen(womenq, totalW, priceW, sumW, arrW, cardindex, quantforWomen);

finalTotal[cardindex] = sumM + sumW;

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

cout << "\t\t \t\t\t\t\t Total Bill: Rs" << finalTotal[cardindex] << endl

<< endl;

cout << billRem1(deliveryop, delivery, cardindex) << endl;

cout << billRem2(delArea, userArea, cardindex) << endl;

returnforAll();

}

// this function finds the total amount of all the items bought by the customer

int findTotalForBill(int menq, int womenq, int totalM[], int priceM[], int totalW[], int priceW[], int cardindex, int quantforMen[][30], int quantforWomen[][30])

{

int sumM = 0, sumW = 0;

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

{

totalM[idx] = quantforMen[cardindex][idx] \* priceM[idx];

sumM += totalM[idx];

}

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

{

totalW[idx] = quantforWomen[cardindex][idx] \* priceW[idx];

sumW += totalW[idx];

}

int total = sumM + sumW;

return total;

}

// it is called in paybill

string takeChoiceForBill()

{

string choice;

cout << "Enter 1 to pay bill." << endl;

cout << "Enter any other number to return." << endl;

cout << "Enter Your Choice: ";

cin >> choice;

return choice;

}

string billPaid(int menq, int availableM[], int womenq, int availableW[], bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30])

{

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

{

availableM[idx] -= quantforMen[cardindex][idx];

}

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

{

availableW[idx] -= quantforWomen[cardindex][idx];

}

billpaid = true;

resetCart(menq, womenq, cardindex, quantforMen, quantforWomen);

string result = "Your bill has been paid.";

return result;

}

// if bill can't be paid it gives the reasons

string reasonsForBill(bool deliveryop, bool delArea)

{

string reason; // deliveryop verifies selection of payment method, delArea verifies the selection of delivery Area

if (deliveryop == 1 && delArea == 0)

{

reason = "Your delivery Area is not selected yet. You can't Pay Bill.";

}

else if (delArea == 1 && deliveryop == 0)

{

reason = "Your payment option is not selected yet. You can't Pay Bill.";

}

else if (delArea == 0 && deliveryop == 0)

{

reason = "Neither your payment method nor your delivery Area is selected yet. You can't Pay Bill.";

}

else

{

reason = "Your total amount is 0. You can't pay the bill.";

}

return reason;

}

// this function pays the bill and make necessary changes afterwards

void PayBill(int totalM[], int totalW[], int priceM[], int priceW[], int availableM[], int availableW[], int menq, int womenq, bool &delArea, bool &deliveryop, bool &billpaid, int cardindex, int quantforMen[][30], int quantforWomen[][30], int billPaidcount[])

{

int total = findTotalForBill(menq, womenq, totalM, priceM, totalW, priceW, cardindex, quantforMen, quantforWomen);

cout << "Your total Purchase Amount is Rs " << total << endl;

if (delArea == 1 && deliveryop == 1 && total != 0)

{

string convert = takeChoiceForBill();

int choice=strToInt(convert);

if (choice == 1)

{

string result = billPaid(menq, availableM, womenq, availableW, billpaid, cardindex, quantforMen, quantforWomen);

cout << result << endl;

deliveryop = false;

delArea = false;

billPaidcount[cardindex]++;

}

else

{

Sleep(300);

cout << "Incorrect choice." << endl;

}

}

else

{

string reason = reasonsForBill(deliveryop, delArea);

cout << reason << endl;

}

returnforAll();

}

// it is called deliveryoptions function

string payOp()

{

string delivery1;

cout << "Enter the way you want to seek out Payment." << endl;

cout << "Enter (Cash or Card): ";

cin >> delivery1;

return delivery1;

}

// also called in deliveryoptions()

string takeCardNum()

{

string card;

cout << "Enter your card number (Must be 4 digit): ";

cin >> card;

return card;

}

int restrictCard(int &a, bool &deliveryop, int cardindex, int cardno[],string convert)

{

// this is for validation of correct card number added

if(!checkingforInteger(convert) || convert.length()!=4)

{

while(!checkingforInteger(convert) || convert.length()!=4)

{

cout << "Card Number not correct.";

cout << "Enter number again: ";

cin >> convert;

}

}

a=strToInt(convert);

cout << "Confirmed.";

deliveryop = true;

a = cardno[cardindex];

return a;

}

// this function takes delivery option(payment method) and makes necessay changes afterwards

void deliveryoptions(bool &deliveryop, string delivery[], int cardindex, int cardno[])

{

string delivery1 = payOp();

if (delivery1 == "Cash" || delivery1=="cash")

{

cout << "It will be a cash on delivery." << endl;

deliveryop = true;

delivery[cardindex] = "Cash";

}

else if (delivery1 == "Card" || delivery1=="card")

{

delivery[cardindex] = "Card";

string convert= takeCardNum();

int card=strToInt(convert);

int a = card;

card = restrictCard(a, deliveryop, cardindex, cardno,convert);

cardno[cardindex]=card;

}

else

{

cout << "Incorrect Method...";

Sleep(300);

}

returnforAll();

}

// this fucntion takes input for delivery area selection

string takeDelArea()

{

string choice;

cout << "Select One of the delivery Areas: " << endl;

cout << "Enter Choice" << endl;

cin >> choice;

return choice;

}

// this function takes delivery area and makes necessary changes afterwards

void deliveryArea(int areas, string deliveryAreas[], string userArea[], bool &delArea, int cardindex)

{

string convert = takeDelArea();

int choice =strToInt(convert);

if (choice > 0 && choice <= areas)

{

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

{

if (choice == idx + 1)

{

userArea[cardindex] = deliveryAreas[idx];

delArea = true;

}

}

}

else

{

cout << "Incorrect Area...";

Sleep(300);

}

returnforAll();

}

// takes input for the quantity to change

string takeQuantitytoChange()

{

string change;

cout << "Enter the item no. of the quantity you want to change: ";

cin >> change;

return change;

}

// this function changes the quantity of an item which is already bought

void updatequantity(string arrM[], string arrW[], int menq, int womenq, int availableM[], int availableW[], int counter, int cardindex, int quantforMen[][30], int quantforWomen[][30])

{

if (counter != 0) // this counter verifies if something is bought or not

{

string convert = takeQuantitytoChange();

int change=strToInt(convert);

if (change > 0 && change <= menq) // this separates the selection of a male item

{

int var = change - 1;

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

{

if (var == idx)

{

if (quantforMen[cardindex][idx] != 0)

{

cout << "The quantity of " << arrM[idx] << " will be changed. " << endl

<< endl;

string convert = takeQuantityforMen(change, availableM, idx); // these functions are for validations

int quantity=strToInt(convert);

quantity = restrictQforMen(quantity, availableM, idx, convert);

quantforMen[cardindex][idx] = quantity;

}

else

{

cout << "Not a correct option." << endl;

Sleep(300);

break;

}

}

}

}

else if (change > menq && change <= (menq + womenq)) // this separated the seletion of a female item

{

int var = change - menq - 1;

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

{

if (var == idx)

{

if (quantforWomen[cardindex][idx] != 0)

{

cout << "The quantity of " << arrW[idx] << " will be changed. " << endl

<< endl;

string convert = takeQuantityforWomen(change, availableW, idx);

int quantity=strToInt(convert); // these functions are for validations

quantity = restrictQforWomen(quantity, availableW, idx, convert);

quantforWomen[cardindex][idx] = quantity;

}

else

{

cout << "Not a correct option." << endl;

Sleep(300);

break;

}

}

}

}

else

{

cout << "Not a correct option.";

Sleep(300);

}

}

else

{

cout << "Nothing to change." << endl;

}

returnforAll();

}

// this fucntions takes days for another function which tells status of order

string takeDays()

{

string days;

cout << "Enter the number of days since order has been placed : ";

cin >> days;

return days;

}

void status(bool billpaid, int billPaidcount[], int cardindex)

{

if (billpaid == 1 || billPaidcount[cardindex] > 0) // this verifies if bill is paid atleast once or not

{

string convert = takeDays();

int days=strToInt(convert);

if (days<8 && days > 3)

{

cout << "Order will be arriving soon." << endl;

}

else if (days <= 3 && days>0)

{

cout << "Order is being prepared." << endl;

}

else

{

cout << "Not Posiible" << endl;

}

}

else

{

cout << "Your order has not been placed yet." << endl;

}

returnforAll();

}

// this function resets the cart of customer and is called in paybill function

void resetCart(int menq, int womenq, int cardindex, int quantforMen[][30], int quantforWomen[][30])

{

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

{

quantforMen[cardindex][i] = 0;

}

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

{

quantforWomen[cardindex][i] = 0;

}

}

// this takes review for the leave review function

string reviews1()

{

string review1;

cout << endl

<< endl;

cout << "Leave a review: ";

cin.clear();

cin.sync();

getline(cin, review1);

review1=restrictnewNameforCloth(review1);

return review1;

}

// this function accepts the review and adds into reviews array

void leaveReview(string reviews[], int &reviewindex, int cardindex)

{

string review;

review = reviews1();

reviews[cardindex] = review;

reviewindex++;

cout << "Your review has been submitted successfully!" << endl;

returnforAll();

}

// this prints the gender page for employee

string employeeGenderPage()

{

string gender;

cout << endl

<< endl;

cout << "Enter the gender." << endl;

cout << "1. \t Men" << endl;

cout << "2. \t Women" << endl;

cout << endl;

cout << "Press any other key to return." << endl;

cout << "Enter your choice: ";

cin >> gender;

return gender;

}

// this prints male items for employee to see

void employeeMitems(string arrM[], int priceM[], int availableM[], int menq)

{

cout << left << setw(7) << "Index" << setw(25) << "Name" << setw(20) << "Price" << setw(20) << "Available Articles" << endl;

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

{

cout << left << setw(7) << to\_string(idx + 1) + ". " << setw(25) << arrM[idx] << setw(20) << "Rs " + to\_string(priceM[idx]) << setw(20) << availableM[idx] << endl;

}

cout << endl;

returnforAll();

}

// this prints female items for the employee to see

void employeeWitems(string arrW[], int priceW[], int availableW[], int womenq)

{

cout << left << setw(7) << "Index" << setw(25) << "Name" << setw(20) << "Price" << setw(20) << "Available Articles" << endl;

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

{

cout << left << setw(7) << to\_string(idx + 1) + ". " << setw(25) << arrW[idx] << setw(20) << "Rs " + to\_string(priceW[idx]) << setw(20) << availableW[idx] << endl;

}

cout << endl;

returnforAll();

}

void returnforAll() //this is used as a return function for all functions

{

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

getch();

}

// it takes a number to add for the employee in men or women clothing

int takeNumToAdd()

{

string convert;

cout << "Enter the number of articles to add (Must not be greater than 5): ";

cin >> convert;

int number=strToInt(convert);

if(number>5 || number<=0)

{

while(number>5 || number<=0) // this loop is for validation

{

cout << "Not Possible.." << endl;

cout << "Enter again: ";

cin >> convert;

number=strToInt(convert);

}

}

return number;

}

// it takes name for the added item

string takeNametoAdd(int counter)

{

string name;

cout << "Enter the name of the " << counter << " article: ";

cin.clear();

cin.sync();

getline(cin, name);

name=restrictnewNameforCloth(name);

return name;

}

// it takes price for the added item

string takePricetoAdd()

{

string price;

cout << "Enter its price : ";

cin >> price;

return price;

}

// it validated price according to criteria

int makePriceAccordingToCriteria(int price,string convertprice)

{

if (!(price > 0) || !checkingforInteger(convertprice))

{

while (!(price >= 0 ) || !checkingforInteger(convertprice))

{

cout << "Not according to criteria." << endl;

cout << "Enter price again: ";

cin >> convertprice;

price=strToInt(convertprice);

}

}

return price;

}

// it takes quantity for the added item

string takeQStockToAdd()

{

string available;

cout << "Enter its quantity : ";

cin >> available;

return available;

}

// it validates the quantity

int makeQStockAccordingtoCriteria(int available, string convertstock)

{

if (!(available >= 0) || !checkingforInteger(convertstock))

{

while (!(available > 0) || !checkingforInteger(convertstock))

{

cout << "Not according to criteria." << endl;

cout << "Enter quantity again: ";

cin >> convertstock;

available=strToInt(convertstock);

}

}

return available;

}

// it makes employee able to add an item in mens menu and makes necessary changes

void addMitem(int &menq, string arrM[], int priceM[], int availableM[])

{

int number = takeNumToAdd();

string name;

int price;

int available;

int counter = 1;

int a = menq;

menq = number + menq;

for (int idx = a; idx < menq; idx++)

{

name = takeNametoAdd(counter);

string convertprice = takePricetoAdd();

price=strToInt(convertprice);

price = makePriceAccordingToCriteria(price,convertprice); //these are for validation

string convertstock = takeQStockToAdd();

available = strToInt(convertstock);

available = makeQStockAccordingtoCriteria(available,convertstock);

cout << endl << endl;

arrM[idx] = name;

priceM[idx] = price; // these are the addition in arrays

availableM[idx] = available;

counter++;

}

returnforAll();

}

// it makes employee able to add an item in mens menu and makes necessary changes

void addWitem(int &womenq, string arrW[], int priceW[], int availableW[])

{

int number = takeNumToAdd();

string name;

int price;

int available;

int counter = 1;

int a = womenq;

womenq = number + womenq;

for (int idx = a; idx < womenq; idx++)

{

name = takeNametoAdd(counter);

string convertprice = takePricetoAdd();

price = strToInt(convertprice);

price = makePriceAccordingToCriteria(price,convertprice); //these are for validation

string convertstock = takeQStockToAdd();

available = strToInt(convertstock);

available = makeQStockAccordingtoCriteria(available, convertstock);

cout << endl << endl;

arrW[idx] = name;

priceW[idx] = price; // these are the addition in arrays

availableW[idx] = available;

counter++;

}

returnforAll();

}

// it takes input to change the stock of an item by the employee

string takeStock()

{

string change;

cout << "Select the item to change the stock of.." << endl;

cout << "Enter item no. : ";

cin >> change;

return change;

}

// it takes the value of the new stock for men

string newStockforMen(int idx, string arrM[])

{

string value;

cout << "Enter new Stock for " << arrM[idx] << " : ";

cin >> value;

return value;

}

// it takes value for the new stock for women

string newStockforWomen(int idx, string arrW[])

{

string value;

cout << "Enter new Stock for " << arrW[idx] << " : ";

cin >> value;

return value;

}

// it validates the value

int makeValueAccToCriteria(int value,string convert)

{

if (!(value >= 0) || !checkingforInteger(convert))

{

while (!(value > 0) || !checkingforInteger(convert))

{

cout << "Not according to criteria." << endl;

cout << "Enter quantity again: ";

cin >> convert;

value=strToInt(convert);

}

}

return value;

}

// this function changes the stock by the employee and makes necessary changes

void changeStock(int menq, string arrM[], int availableM[], string arrW[], int womenq, int availableW[])

{

int value;

string convert = takeStock();

int change=strToInt(convert);

if (change > 0 && change <= menq) //seperated men items

{

int var = change - 1;

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

{

if (var == idx)

{

string convertM = newStockforMen(idx, arrM);

value = strToInt(convertM);

value = makeValueAccToCriteria(value,convertM);

availableM[idx] = value;

}

}

}

else if (change > menq && change <= (menq + womenq)) // separates women items

{

int var = change - menq - 1;

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

{

if (var == idx)

{

string convertW = newStockforWomen(idx, arrW);

value =strToInt(convertW); // these are for validations

value = makeValueAccToCriteria(value,convertW);

availableW[idx] = value;

}

}

}

else

{

cout << "Not a correct option." << endl;

Sleep(300);

}

returnforAll();

}

// this function shows reviews to the employeer and is called in check reviews function

void showreviews(string reviews[], string customerArr[], int &counter, int cusIndex)

{

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

{

if (reviews[idx] != "")

{

cout << counter << ". Review by " << customerArr[idx] << ": " << endl;

cout << "\t" << reviews[idx] << endl

<< endl;

counter++;

}

}

}

// these fucntion shows reviews or tells that none exist

void checkReviews(string reviews[], string customerArr[], int reviewindex, int cusIndex)

{

int counter = 1;

if (reviewindex != 0)

{

showreviews(reviews, customerArr, counter, cusIndex);

}

else

{

cout << endl

<< endl;

cout << "There are no reviews yet." << endl;

}

returnforAll();

}

// this fucntion takes name of the item whos name is about to be changed

string takeName(int menq, int womenq, string arrM[], string arrW[])

{

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

{

cout << "Item no: " << idx + 1 << ". " << arrM[idx] << endl; //print amen items

}

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

{

cout << "Item no: " << idx + 1 + menq << ". " << arrW[idx] << endl; // print women items

}

cout << "Select the item ..." << endl

<< endl;

string change;

cout << "Enter item no. : ";

cin >> change;

return change;

}

void newNameforMen(int idx, int var, string arrM[]) // it takes the new name for men

{

cout << "Enter new name for " << arrM[idx] << ": ";

cin.clear();

cin.sync();

getline(cin, arrM[var]);

arrM[var]=restrictnewNameforCloth(arrM[var]);

cout << "Name succesfully changed." << endl;

}

string restrictnewNameforCloth(string sen)

{

while(checkforEmpty(sen) || checkingForcomma(sen))

{

if(checkforEmpty(sen))

{

while (checkforEmpty(sen))

{

cout << "It must not be empty." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,sen);

}

}

if(checkingForcomma(sen))

{

while(checkingForcomma(sen))

{

cout << "It must not contain a comma." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin, sen);

}

}

}

return sen;

}

void newNameforWomen(int idx, int var, string arrW[]) // it takes the new name for women

{

cout << "Enter new name for " << arrW[idx] << ": ";

cin.clear();

cin.sync();

getline(cin, arrW[var]);

arrW[var]=restrictnewNameforCloth(arrW[var]);

cout << "Name succesfully changed." << endl;

}

// this function calls other functions and makes the necessary changes regarding name changing

void changeName(int menq, int womenq, string arrM[], string arrW[])

{

string convert = takeName(menq, womenq, arrM, arrW);

int change= strToInt(convert);

if (change > 0 && change <= menq) // seperates men items

{

int var = change - 1;

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

{

if (var == idx)

{

newNameforMen(idx, var, arrM);

}

}

}

else if (change > menq && change <= (menq + womenq)) // seperated women items

{

int var = change - 1 - menq;

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

{

if (var == idx)

{

newNameforWomen(idx, var, arrW);

}

}

}

else

{

cout << "Not a correct option." << endl;

Sleep(300);

}

returnforAll();

}

// this function removes an item of clothing

void removeItem(int &menq, int &womenq, string arrM[], string arrW[], int priceM[], int priceW[], int availableM[], int availableW[])

{

string convert= takeName(menq, womenq, arrM, arrW);

int remove=strToInt(convert);

if (remove > 0 && remove <= menq) // separates men item

{

for (int var=remove-1; var < menq; var++)

{

arrM[var] = arrM[var + 1];

priceM[var] = priceM[var + 1];

availableM[var] = availableM[var + 1];

}

cout << "Item Succesfully deleted." << endl;

menq=menq-1;

}

else if (remove > menq && remove <= (menq + womenq)) // separates women items

{

for (int var = remove - 1 - menq; var < womenq; var++)

{

arrW[var] = arrW[var + 1];

priceW[var] = priceW[var + 1];

availableW[var] = availableW[var + 1];

}

cout << "Item Succesfully deleted." << endl;

womenq = womenq - 1;

}

else

{

cout << "Not a correct option." << endl;

Sleep(300);

}

returnforAll();

}

// it shows the list of customers with their details

void showCustomersList(int cusIndex, string userArea[], string delivery[], string customerArr[], int &counter)

{

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

{

if (userArea[idx] != "" && delivery[idx] != "")

{

cout << left << setw(12) << to\_string(counter) + ". " << setw(20) << customerArr[idx] << setw(20) << userArea[idx] << setw(20) << delivery[idx] << endl;

}

else if (userArea[idx] != "" && delivery[idx] == "")

{

cout << left << setw(12) << to\_string(counter) + ". " << setw(20) << customerArr[idx] << setw(20) << userArea[idx] << setw(20) << "Not Selected" << endl;

}

else if (userArea[idx] == "" && delivery[idx] != "")

{

cout << left << setw(12) << to\_string(counter) + ". " << setw(20) << customerArr[idx] << setw(20) << "Not Selected" << setw(20) << delivery[idx] << endl;

}

else

{

cout << left << setw(12) << to\_string(counter) + ". " << setw(20) << customerArr[idx] << setw(20) << "Not Selected" << setw(20) << "Not Selected" << endl;

}

counter++;

}

}

// it checks if there are customers or not and then calls showcustomers function to print their list

void seeCustomer(int cusIndex, string userArea[], string delivery[], string customerArr[])

{

if (cusIndex != 0)

{

int counter = 1;

cout << endl

<< endl;

cout << "These are the Customers: " << endl

<< endl;

cout << left << setw(12) << "Index" << setw(20) << " Username" << setw(20) << " Address" << setw(20) << " Method" << endl

<< endl;

showCustomersList(cusIndex, userArea, delivery, customerArr, counter);

}

else

{

cout << endl

<< endl;

cout << "There are no Customers yet." << endl;

}

returnforAll();

}

// it takes the number of addresses that employee wants to add

string takeNumberofAddress()

{

string number;

cout << endl

<< endl;

cout << "Enter the number of adresses you want to add (Must be less than 3): ";

cin >> number;

return number;

}

// it validates the number of addresses that the employee wants to add

int restrictNumberofAddress(int number)

{

if (number>3 || number <= 0)

{

string convert = intToStr(number);

while (number>3 || number <= 0)

{

cout << "Not possible..." << endl;

cout << "Enter again: ";

cin >> convert;

number=strToInt(convert);

}

}

return number;

}

// it adds the delivery in delivery Area array and also increments its index

void addDeliveryArea(int &areas, string deliveryAreas[])

{

string convert = takeNumberofAddress();

int number = strToInt(convert);

number = restrictNumberofAddress(number);

int a = areas;

areas = areas + number;

for (int idx = a; idx < areas; idx++)

{

string address;

cout << "Enter Address " << idx + 1 << ": ";

cin.clear();

cin.sync();

getline(cin, address);

address=restrictAddressName(address);

cout << endl

<< endl;

deliveryAreas[idx] = address;

}

returnforAll();

}

string restrictAddressName(string address)

{

while(checkforEmpty(address) || checkingForcomma(address))

{

if(checkforEmpty(address))

{

while (checkforEmpty(address))

{

cout << "It must not be empty." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,address);

}

}

if(checkingForcomma(address))

{

while(checkingForcomma(address))

{

cout << "It must not contain a comma." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin, address);

}

}

}

return address;

}

string signupName(string name)

{

while(checkforEmpty(name) || checkingForcomma(name) || checkingforspace(name))

{

if(checkforEmpty(name))

{

while (checkforEmpty(name))

{

cout << "It must not be empty." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,name);

}

}

if(checkingForcomma(name))

{

while(checkingForcomma(name))

{

cout << "It must not contain a comma." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin, name);

}

}

if(checkingforspace(name))

{

while(checkingforspace(name))

{

cout << "It must not contain a space." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,name);

}

}

}

return name;

}

// it take the chocie of which delivery address is to be removed

string takeNumberofAddressToRemove()

{

string choice;

cout << "Enter your Choice: ";

cin >> choice;

return choice;

}

// it removes the delivery address and decrements the index on delivery area array.

void removeAddress(int &areas, string deliveryAreas[])

{

string convert = takeNumberofAddressToRemove();

int choice = strToInt(convert);

if (choice > 0 && choice <= areas)

{

for(int var=choice-1 ; var < areas; var++)

{

deliveryAreas[var] = deliveryAreas[var + 1];

}

areas = areas - 1;

}

else

{

cout << "Not a corrext option...";

Sleep(300);

}

returnforAll();

}

void clearScreen()

{

system("cls");

printHeader();

}

void signinWindow(string &name, string &password1)

{

cout << endl;

cout << "--------Sign In----------";

cout << endl

<< endl;

cout << "Enter Username: ";

cin.clear();

cin.sync();

getline(cin,name);

if(checkingforspace(name)) // validation

{

cout << "It should not contain space.";

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,name);

}

cout << "Enter Password (6 digits): ";

cin.clear();

cin.sync();

getline(cin,password1);

password1=retrictPassword(password1);

cout << endl

<< endl;

}

// itshows the customer stores contact info

void contactForCustomer(string phoneN, string email)

{

cout << "Our Contact Number: " << phoneN << endl;

cout << "Our Email Address: " << email << endl;

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

returnforAll();

}

// it takes input from the employee regarding changes in contact info

void setContactInfo(string &phoneN, string &email)

{

cout << "Enter new Contact Number: ";

cin.clear();

cin.sync();

getline(cin,phoneN);

if(checkingForcomma(phoneN))

{

while(checkingForcomma(phoneN))

{

cout << "It should not contain comma." << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,phoneN);

}

}

cout << "Enter new Email Address: ";

cin.clear();

cin.sync();

getline(cin,email);

while(checkingforspace(email) || !checkingforAtthesymbol(email))

{

if(checkingforspace(email)) // validatory restrictions

{

while(checkingforspace(email))

{

cout << "It should not contain spaces. " << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,email);

}

}

if(!checkingforAtthesymbol(email)) // validatory restrictions

{

while(!checkingforAtthesymbol(email))

{

cout << "It must contain @ symbol. " << endl;

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,email);

}

}

}

cout << "Info updated." << endl;

}

string takeChoiceForContact()

{

string answer;

cout << "Press 1 to change the Contact Information." << endl;

cout << "Enter any other string to return." << endl;

cout << "Enter your choice: ";

cin >> answer;

return answer;

}

void changeInfo(string &phoneN, string &email) // it changes the contact info as taken from the employee

{

string answer = takeChoiceForContact();

if (answer == "1")

{

setContactInfo(phoneN, email);

}

else

{

Sleep(300);

}

returnforAll();

}

int strToInt(string convert) // these functions are for validations

{

int result = 0;

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

{

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

}

return result;

}

string intToStr(int num) // this for the proper execution of validation

{

if (num == 0) {

return "0";

}

string result = "";

while (num > 0) {

char digit = '0' + (num % 10);

result = digit + result;

num /= 10;

}

return result;

}

bool checkingForcomma(string sen)

{

for (int x=0; sen[x]!='\0'; x++)

{

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

{

return true;

}

}

return false;

}

bool checkingforspace(string sen)

{

for(int x=0; sen[x]!='\0'; x++)

{

if(sen[x] == ' ')

{

return true;

}

}

return false;

}

bool checkingforAtthesymbol(string sen)

{

for(int x=0; sen[x]!='\0'; x++)

{

if(sen[x]=='@')

{

return true;

}

}

return false;

}

bool checkingforInteger(string sen)

{

for (int x=0; sen[x] != '\0'; x++)

{

if (sen[x]< '0' || sen[x] >'9')

{

return false;

}

}

return true;

}

bool checkingforlengthofP(string sen)

{

if(sen.length()!=6)

{

return false;

}

return true;

}

string retrictPassword(string sen)

{

while(checkingforspace(sen) || !checkingforlengthofP(sen) || !checkingforInteger(sen))

{

if(!checkingforlengthofP(sen) || !checkingforInteger(sen))

{

while(!checkingforlengthofP(sen) || !checkingforInteger(sen))

{

cout << "The password is not according to given criteria." << endl;

cout << "Enter password again: ";

cin.clear();

cin.sync();

getline(cin,sen);

}

}

if(checkingforspace(sen))

{

while(checkingforspace(sen))

{

cout << "It should not contain space.";

cout << "Enter again: ";

cin.clear();

cin.sync();

getline(cin,sen);

}

}

}

return sen;

}

// checks for empty spaces

bool checkforEmpty(string sen)

{

if(sen=="")

{

return true;

}

return false;

}

# Weakness in the Business Application

* + The perimeters are way too many for most of the functions.
  + Single functionality is not properly executed in functions.
  + The carts are made in 2d arrays for different customers, so they are not part of the file handling.

# Future Directions

* + Reduce the number of perimeters.
  + Replace void functions.
  + Properly execute single functionality.
  + Gain proper time management.
  + Improve the user interface.