**E-Commerce Management System**



Session: 2022 – 2025

**Submitted by:**

Muhammad Furqan 2022R/2021-CS-199

**Supervised by:**

Professor Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Here you can find the major parts of your Business Application documentation**

* **Short Description of project**
  + In this project, there are two types of users i.e., seller and customer. The main

objective is to develop a business application through which a user can order a product and have different options like add to cart, discounts, etc. In the upcoming era, the need for the online shopping will go too much higher and its increasing day by day, so, my plan to develop an E-Commerce application is clearly the much more need of these services in the upcoming future.

* **Users of Application (minimum 2 users for your project)**
  + There are two users of this application, these are,
  + Customers: Customers can buy different products and may have different type of discounts and after purchasing a product can give a review.
  + Seller: Seller can list different products, give different discounts, remove a product and update a product. It can also give tracking Information.
* **Functional Requirements**
  + Functional requirements are a detailed description of what a system, product, or service must do to meet the needs and expectations
  + Use the following format to write the Functional Requirements
* **Wireframes**

Wireframes for E-Commerce Management System:

**Login Page**



SIGNUP MENU



SELLER MENU



IF ENTER 1



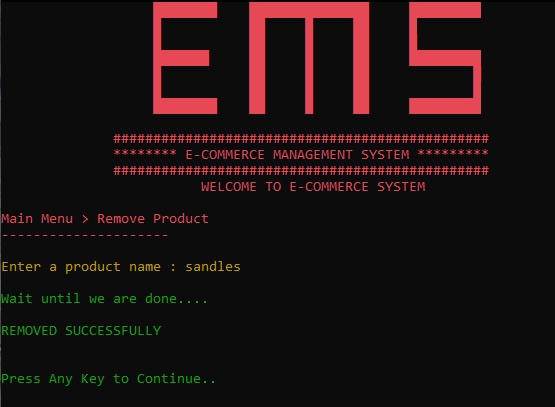
IF ENTER 2



IF ENTER 3



IF ENTER 4



IF ENTER 5



IF ENTER 6



IF ENTER 7



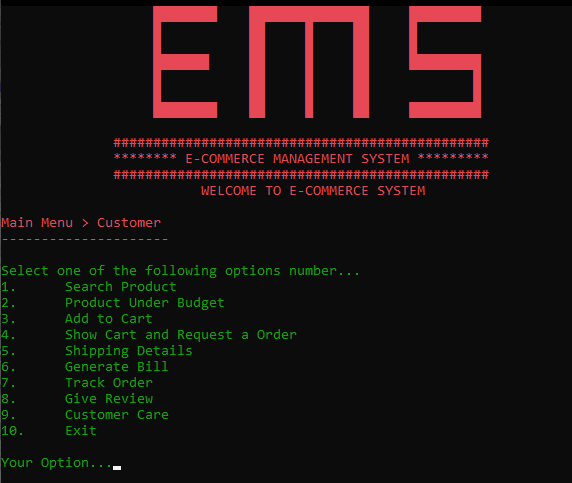
IF ENTER 8



IF ENTER 9



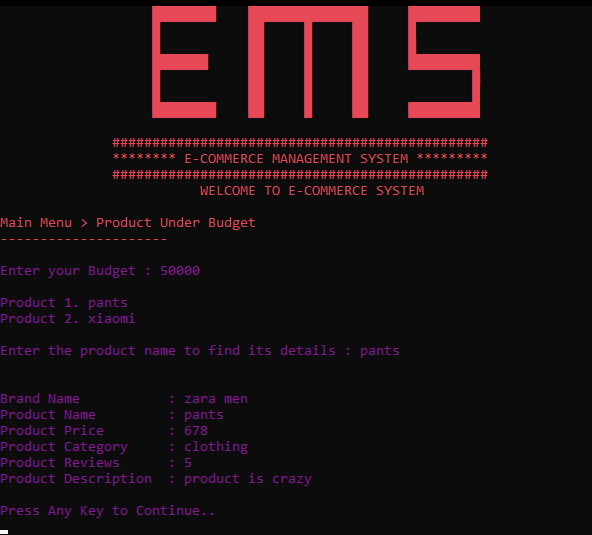
CUSTOMER LOGIN MENU



IF ENTER 1



IF ENTER 2



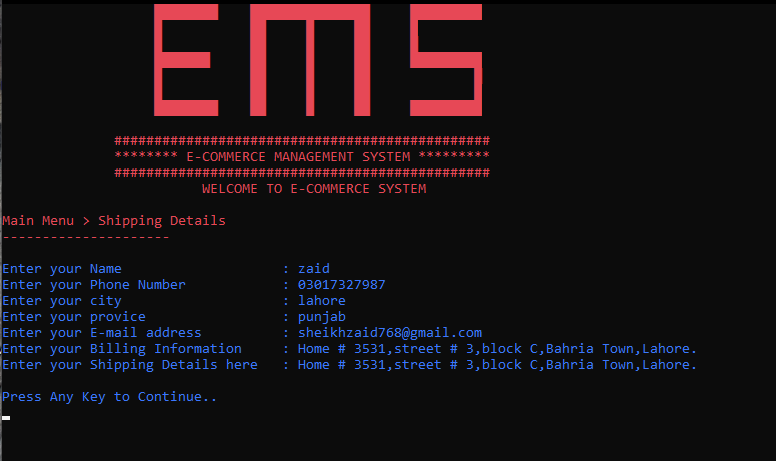
IF ENTER 3



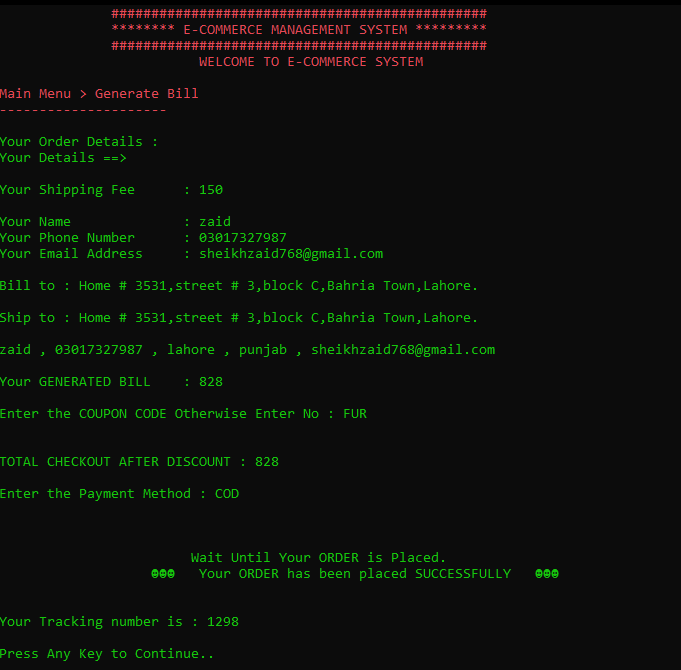
IF ENTER 4



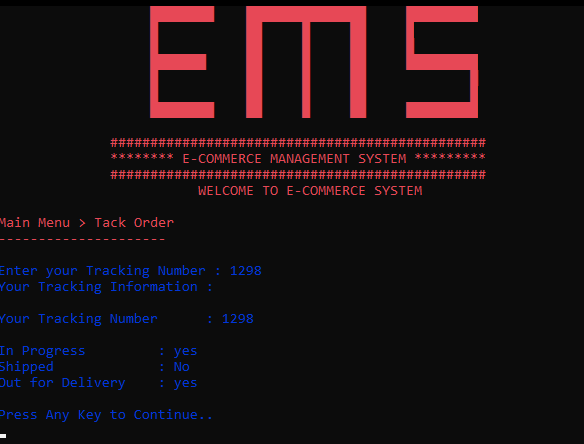
IF ENTER 5



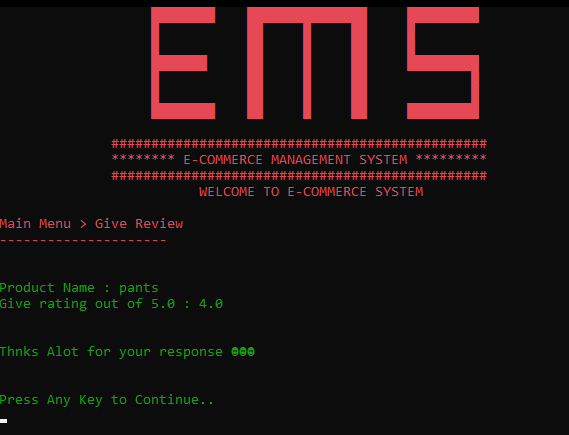
IF ENTER 6



IF ENTER 7



IF ENTER 8



IF ENTER 9



* **Function Prototypes**

//------------------------------------User Defined Functions--------------------------------------//

// functions for login Menu

void topHeader();

void subMenuBeforeMainMenu(string submenu);

void submenu();

bool signUp(string name, string password, string role);

string signIn(string name, string password);

int loginMenu();

void clearScreen();

// Functions for seller

void sellerInterface();

int sellerMenu();

// Functions for Customer

void customerInterface();

int customerMenu();

// file handling for Logins

void storeUser(string userName, string password, string role);

void loadSignup();

// Seller Menus

void sellerMenu1(); // done

void sellerMenu2(); // done

void sellerMenu3(); // done

void sellerMenu4(); // done

int sellerMenu4subMenu(string removeProduct); // done

void sellerMenu5(); // done

void sellerMenu6();

void sellerMenu7(); // done

void sellerMenu8(); // done

void sellerMenu9(); // done

void sellerMenu9SubMenu1(int option); // done

void sellerMenu9SubMenu2(); // done

// Seller Menus File Handling

void StoreSellerMenu1(); // done

void LoadSellerMenu1(); // done

void StoreSellerMenu4(); // done

void StoreSellerMenu5(); // done

void StoreSellerMenu7(); // done

void LoadSellerMenu7(); // done

void StoreSellerMenu8(); // done

void LoadSellerMenu8(); // done

// Customer Menus

void customerMenu1(); // done

void customerMenu2(); // done

void customerMenu3(); // done

void customerMenu4(); // done

void customerMenu5(); // done

void customerMenu6(); // done

void customerMenu7(); // done

void customerMenu8(); // done

void customerMenu9(); // done

void customerMenu9SubMenu1(int option);

void customerMenu9SubMenu2();

// Customer Menus File Handling

void StoreCustomerMenu3(); // done

void LoadCustomerMenu3(); // done

void StoreCustomerMenu5(); // done

void LoadCustomerMenu5(); // done

void StoreCustomerMenu7(); // done

void LoadCustomerMenu7(); // done

* **Functions Working Flow**

LoadSignup()

main()

Signin()

Signup()

LoadSellerMenu1()

SellerInterface()

CustomerInterface()

LoadSellerMenu7()

CustomerMenu()

SellerMenu()

LoadSellerMenu8()

LoadCustomerMenu3()

SellerMenu1()

CustomerMenu1()

SellerMenu2()

CustomerMenu2()

LoadCustomerMenu5()

SellerMenu3()

CustomerMenu3()

SellerMenu4()

LoadCustomerMenu7()

CustomerMenu4()

SellerMenu5()

CustomerMenu5()

SellerMenu6()

CustomerMenu6()

SellerMenu7()

CustomerMenu7()

SellerMenu8()

CustomerMenu8()

SellerMenu9()

CustomerMenu9()

SellerMenu10()

CustomerMenu10()

* **Complete Code of the Business Application**

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

#include <conio.h>

using namespace std;

//--------------------------------------- Global Variables--------------------------------------//

// Variables for Login Menu

const int userArrSize = 10;

string users[userArrSize];

string passwords[userArrSize];

string roles[userArrSize];

int usersCount = 0;

// Variables for Product Detail

string productBrand[100];

string productName[100];

string productCategory[100];

string productDetails[100];

float productPrice[100];

float productReviews[100];

int productCartNumber;

// Variables for Shipping Details

string nameShipping;

string numberShipping;

string cityShipping;

string provinceShipping;

string emailShipping;

string shippingInfo;

string billingInfo;

float shippingFees = 150;

// Variable for Emoji Character

char emoji = 2;

// Variable for Shippment and Payment method

int paymentMethods;

int shippingCompany;

string shipping[100];

string payment[100];

// Variables for Cart and Order Details

int placeCart;

int placeOrderedindex = 0;

int reviewCount = 0;

int orderPlacement = 0; // Order place hone k bad is me increment ho jaye ga jis se seller ki menu3 me addition ho jaye gi

int productCount = 0;

int cartCount = 0;

int addToCartProducts[100];

int totalCheckout = 0;

int numberOfProductsOrdered;

int orderPlacedArray[100];

// Variable for Tracking Information

int trackingNumber = 0;

string inProgress = "No";

string shipped = "No";

string delivered = "No";

// Variables for Discount

string couponDiscount;

float discountCouponPercentage;

int numberOfItemsDiscount;

float discountItemPercentage;

int checkOutDiscount;

float discountCheckOutPercentage;

//------------------------------------User Defined Functions--------------------------------------//

// functions for login Menu

void topHeader();

void subMenuBeforeMainMenu(string submenu);

void submenu();

bool signUp(string name, string password, string role);

string signIn(string name, string password);

int loginMenu();

void clearScreen();

// Functions for seller

void sellerInterface();

int sellerMenu();

// Functions for Customer

void customerInterface();

int customerMenu();

// file handling for Logins

void storeUser(string userName, string password, string role);

void loadSignup();

// Seller Menus

void sellerMenu1(); // done

void sellerMenu2(); // done

void sellerMenu3(); // done

void sellerMenu4(); // done

int sellerMenu4subMenu(string removeProduct); // done

void sellerMenu5(); // done

void sellerMenu6();

void sellerMenu7(); // done

void sellerMenu8(); // done

void sellerMenu9(); // done

void sellerMenu9SubMenu1(int option); // done

void sellerMenu9SubMenu2(); // done

// Seller Menus File Handling

void StoreSellerMenu1(); // done

void LoadSellerMenu1(); // done

void StoreSellerMenu4(); // done

void StoreSellerMenu5(); // done

void StoreSellerMenu7(); // done

void LoadSellerMenu7(); // done

void StoreSellerMenu8(); // done

void LoadSellerMenu8(); // done

// Customer Menus

void customerMenu1(); // done

void customerMenu2(); // done

void customerMenu3(); // done

void customerMenu4(); // done

void customerMenu5(); // done

void customerMenu6(); // done

void customerMenu7(); // done

void customerMenu8(); // done

void customerMenu9(); // done

void customerMenu9SubMenu1(int option);

void customerMenu9SubMenu2();

// Customer Menus File Handling

void StoreCustomerMenu3(); // done

void LoadCustomerMenu3(); // done

void StoreCustomerMenu5(); // done

void LoadCustomerMenu5(); // done

void StoreCustomerMenu7(); // done

void LoadCustomerMenu7(); // done

// MAIN FUNCTION----------------------------------------------------------------//

int main()

{

int loginOption = 0;

// load all the functions from the file

loadSignup();

LoadSellerMenu1();

LoadSellerMenu7();

LoadSellerMenu8();

LoadCustomerMenu3();

LoadCustomerMenu5();

LoadCustomerMenu7();

while (loginOption != 3)

{

topHeader();

subMenuBeforeMainMenu("Login");

loginOption = loginMenu();

if (loginOption == 1)

{ // signup option Interface

system("cls");

string name;

string password;

string role;

cout << "\n";

topHeader();

subMenuBeforeMainMenu("SignUp");

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

cin >> name;

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

cin >> password;

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

cin >> role;

bool isValid = signUp(name, password, role);

if (isValid)

{

storeUser(name, password, role);

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

SetConsoleTextAttribute(hConsole, 2);

cout

<< "SignedUp Successfully\n\n";

}

if (!isValid)

{

cout << "Users in the System have exceeded the Capacity" << endl;

}

clearScreen();

}

else if (loginOption == 2)

{

// Sign in interface

system("cls");

string name;

string password;

string role;

topHeader();

subMenuBeforeMainMenu("SignIn");

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

cin >> name;

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

cin >> password;

role = signIn(name, password);

if (role == "Seller")

{

clearScreen();

sellerInterface();

}

else if (role == "Customer")

{

clearScreen();

customerInterface();

}

else if (role == "undefined")

{

cout << "You Entered wrong Credentials" << endl;

system("cls");

}

}

}

}

// file handling for signUp function

void storeUser(string userName, string password, string role)

{

fstream file;

file.open("signup.txt", ios::app);

file << userName << "\n"

<< password << "\n"

<< role << "\n";

file.close();

}

// file handling for load of signup

void loadSignup()

{

fstream file;

string uName;

string rolee;

string pass;

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

while (getline(file, uName) && getline(file, pass) && getline(file, rolee))

{

users[usersCount] = uName;

passwords[usersCount] = pass;

roles[usersCount] = rolee;

usersCount++;

}

file.close();

}

void topHeader()

{

char box = 219;

{

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

SetConsoleTextAttribute(hConsole, 12);

cout << " " << box << box << box << box << box << box << box << box << " " << box << box << box << box << box << box << box << box << box << box << box << box << box << box << box << " " << box << box << box << box << box << box << box << box << box << endl;

cout << " " << box << " " << box << box << " " << box << " " << box << box << " " << box << endl;

cout << " " << box << " " << box << box << " " << box << " " << box << box << " " << box << " " << endl;

cout << " " << box << box << box << box << box << box << box << " " << box << box << " " << box << " " << box << box << " " << box << box << box << box << box << box << box << box << box << endl;

cout << " " << box << " " << box << box << " " << box << " " << box << box << " " << box << " " << endl;

cout << " " << box << " " << box << box << " " << box << " " << box << box << " " << box << endl;

cout << " " << box << box << box << box << box << box << box << box << " " << box << box << " " << box << " " << box << box << " " << box << box << box << box << box << box << box << box << box << endl;

cout << " " << endl;

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

cout << " \*\*\*\*\*\*\*\* E-COMMERCE MANAGEMENT SYSTEM \*\*\*\*\*\*\*\*\* " << endl;

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

cout << " WELCOME TO E-COMMERCE SYSTEM " << endl;

cout << endl;

}

}

int loginMenu()

{

int option;

cout << "1. SignUp to get you ID" << endl;

cout << "2. SignIn with your ID" << endl;

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

cout << endl;

cout << "Enter the Option Number >> ";

cin >> option;

return option;

}

void subMenuBeforeMainMenu(string submenu)

{

string message = submenu + " Menu";

cout << message << endl;

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

<< endl;

}

void subMenu(string submenu)

{

string message = "Main Menu > " + submenu;

cout << message << endl;

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

<< endl;

}

// signup function to get your ID

bool signUp(string name, string password, string role)

{

if (usersCount < userArrSize)

{

users[usersCount] = name;

passwords[usersCount] = password;

roles[usersCount] = role;

usersCount++;

return true;

}

else

{

return false;

}

}

// signin function

string signIn(string name, string password)

{

for (int index = 0; index < usersCount; index++)

{

if (users[index] == name && passwords[index] == password)

{

return roles[index];

}

}

return "undefined";

}

void clearScreen()

{

cout << endl;

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

getch();

system("cls");

}

// Seller Interface function

void sellerInterface()

{

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

SetConsoleTextAttribute(hConsole, 1);

int sellerOption = 0;

while (sellerOption != 10)

{

topHeader();

subMenu("Seller");

sellerOption = sellerMenu();

if (sellerOption == 1)

{

// seller Menu1 = Product listing

system("cls");

topHeader();

subMenu("List the Products");

sellerMenu1();

StoreSellerMenu1();

}

else if (sellerOption == 2)

{

// seller Menu2 = Search a Product

system("cls");

topHeader();

subMenu("Search a Product");

sellerMenu2();

}

else if (sellerOption == 3)

{

// seller Menu3 = Order Fulfillment

system("cls");

topHeader();

subMenu("Order Fulfillment");

sellerMenu3();

}

else if (sellerOption == 4)

{

// seller Menu4 = Remove Product

system("cls");

topHeader();

subMenu("Remove Product");

sellerMenu4();

StoreSellerMenu4();

}

else if (sellerOption == 5)

{

// seller Menu5 = Update a Product

system("cls");

topHeader();

subMenu("Update a Product");

sellerMenu5();

StoreSellerMenu5();

}

else if (sellerOption == 6)

{

// seller Menu6 = Show All Products

system("cls");

topHeader();

subMenu("Show All Products");

sellerMenu6();

}

else if (sellerOption == 7)

{

// seller Menu7 = Payment and Shipping

system("cls");

topHeader();

subMenu("Payment and Shipping");

sellerMenu7();

StoreSellerMenu7();

}

else if (sellerOption == 8)

{

// seller Menu8 = Discount

system("cls");

topHeader();

subMenu("Discount");

sellerMenu8();

}

else if (sellerOption == 9)

{

// seller Menu8 = Customer Support

system("cls");

topHeader();

subMenu("Customer Support");

sellerMenu9();

}

clearScreen();

}

}

//-----------------product Listing / Menu 1

void sellerMenu1()

{

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

SetConsoleTextAttribute(hConsole, 3);

cout << "Enter the brand name : ";

cin.ignore();

getline(cin, productBrand[productCount]);

cout << "Enter the product name : ";

getline(cin, productName[productCount]);

cout << "Enter the product category : ";

getline(cin, productCategory[productCount]);

cout << "Enter the Price : ";

cin >> productPrice[productCount];

cout << "Enter the description : ";

cin.ignore();

getline(cin, productDetails[productCount]);

cout << "Enter the Review : ";

cin >> productReviews[productCount];

productCount++;

}

// file handling SellerMenu1 for storing

void StoreSellerMenu1()

{

fstream file;

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

if (productCount != 0)

{

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

{

file << productBrand[i] << "\n";

file << productName[i] << "\n";

file << productCategory[i] << "\n";

file << productPrice[i] << "\n";

file << productDetails[i] << "\n";

file << productReviews[i] << "\n";

}

}

if (productCount == 0)

{

cout << "No Records yet.\n";

}

file.close();

}

// file handling SellerMenu1 for Loading

void LoadSellerMenu1()

{

fstream file;

string prodBrand;

string prodName;

string prodCategory;

string prodDetails;

string prodPrice;

string prodReviews;

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

while ((getline(file, prodBrand)) && (getline(file, prodName)) && (getline(file, prodCategory)) && (getline(file, prodPrice)) && (getline(file, prodDetails)) && (getline(file, prodReviews)))

{

productBrand[productCount] = prodBrand;

productName[productCount] = prodName;

productCategory[productCount] = prodCategory;

productDetails[productCount] = prodDetails;

productPrice[productCount] = stof(prodPrice);

productReviews[productCount] = stof(prodReviews);

productCount++;

}

file.close();

}

//-----------------Product Search / Menu 2

void sellerMenu2()

{

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

SetConsoleTextAttribute(hConsole, 4);

if (productCount == 0)

{

cout << "There are no Poducts Yet(^.^)\n\n";

}

else

{

int productOption;

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

{

cout << "Product " << i + 1 << "."

<< " " << productName[i] << endl;

}

cout << endl;

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

cin >> productOption;

cout << endl;

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

{

if (productOption == i + 1)

{

cout << "Brand Name : " << productBrand[i] << endl;

cout << "Product Name : " << productName[i] << endl;

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

cout << "Product Category : " << productCategory[i] << endl;

cout << "Product Reviews : " << productReviews[i] << "\n\n";

cout << "Product Description : " << productDetails[i] << endl;

}

}

}

}

//-----------------Order Fulfillment / Menu 3

void sellerMenu3()

{

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

SetConsoleTextAttribute(hConsole, 5);

int i = 0;

if (trackingNumber == 1298)

{

cout << "Order # " << i + 1 << " : \n";

cout << "Enter the current \*\*in Progress\*\* : ";

cin >> inProgress;

cout << "Enter the current \*\*Shipped\*\* : ";

cin >> shipped;

cout << "Enter the current \*\*Delivered\*\* : ";

cin >> delivered;

}

if (trackingNumber == 0)

{

cout << "You have not any Orders yet." << emoji << emoji << emoji << "\n\n";

}

}

//-----------------Remove Product / Menu 4

void sellerMenu4()

{

string removeProduct;

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

SetConsoleTextAttribute(hConsole, 6);

if (productCount == 0)

{

cout << "No Products Yet " << emoji << emoji << emoji << "\n\n";

}

else

{

cout << "Enter a product name : ";

cin.ignore();

getline(cin, removeProduct);

int index = 1000;

index = sellerMenu4subMenu(removeProduct);

while (index != 1000)

{

for (int i = index + 1; i < productCount; i++)

{

productBrand[i] = productBrand[i + 1];

productName[i] = productName[i + 1];

productCategory[i] = productCategory[i + 1];

productPrice[i] = productPrice[i + 1];

productDetails[i] = productDetails[i + 1];

productReviews[i] = productReviews[i + 1];

}

productCount--;

break;

}

if (index == 1000)

{

cout << "Invalid Product Name." << endl;

}

}

}

// SellMenu4 subMenu

int sellerMenu4subMenu(string removeProduct)

{

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

SetConsoleTextAttribute(hConsole, 2);

int removeIndex = 1000;

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

{

if (removeProduct == productName[i])

{

productBrand[i] = productBrand[i + 1];

productName[i] = productName[i + 1];

productCategory[i] = productCategory[i + 1];

productPrice[i] = productPrice[i + 1];

productDetails[i] = productDetails[i + 1];

productReviews[i] = productReviews[i + 1];

removeIndex = i;

Sleep(200);

cout << "\nWait until we are done...." << endl

<< endl;

Sleep(2000);

cout << "REMOVED SUCCESSFULLY "

<< "\n\n";

break;

}

}

return removeIndex;

}

// file handling for seller Menu4 Storing

void StoreSellerMenu4()

{

fstream file;

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

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

{

file << productBrand[i] << "\n";

file << productName[i] << "\n";

file << productCategory[i] << "\n";

file << productPrice[i] << "\n";

file << productDetails[i] << "\n";

file << productReviews[i] << "\n";

}

}

//-----------------Update Product / Menu5

void sellerMenu5()

{

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

SetConsoleTextAttribute(hConsole, 7);

if (productCount == 0)

{

cout << "No Products Yet " << emoji << emoji << emoji << "\n";

}

else

{

int updateCount = 0;

string updatedBrand, searchProductName, updatedName, updatedCategory, updatedDetails;

int updatedPrice;

float updatedReviews;

cout << "Enter a product name : ";

cin.ignore();

getline(cin, searchProductName);

cout << endl;

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

{

if (searchProductName == productName[i])

{

cout << "Enter the new Product name : ";

getline(cin, updatedName);

productName[i] = updatedName;

cout << "Enter the new Brand name : ";

getline(cin, updatedBrand);

productBrand[i] = updatedBrand;

cout << "Enter the new Product Category : ";

getline(cin, updatedCategory);

productCategory[i] = updatedCategory;

cout << "Enter the Price : ";

cin >> updatedPrice;

productPrice[i] = updatedPrice;

cout << "Enter the new Description : ";

cin.ignore();

getline(cin, updatedDetails);

productDetails[i] = updatedDetails;

cout << "Enter the new Review : ";

cin >> updatedReviews;

productReviews[i] = updatedReviews;

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

SetConsoleTextAttribute(hConsole, 2);

Sleep(2000);

cout << endl

<< endl;

cout << "Done Successfully" << endl;

cout << "(^.^)" << endl;

Sleep(3000);

updateCount++;

break;

}

}

if (updateCount == 0)

{

cout << "Invaid Product Name." << endl;

}

}

}

// File Handling related to Seller Menu5 Storing

void StoreSellerMenu5()

{

fstream file;

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

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

{

file << productBrand[i] << "\n";

file << productName[i] << "\n";

file << productCategory[i] << "\n";

file << productPrice[i] << "\n";

file << productDetails[i] << "\n";

file << productReviews[i] << "\n";

}

}

//-------------SellerMenu6 / Show All Product

void sellerMenu6()

{

if (productCount == 0)

{

cout << "No Products Yet" << emoji << emoji << emoji << "\n";

}

else

{

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

{

cout << i + 1 << ".\n";

cout << "Brand Name : " << productBrand[i] << endl;

cout << "Product Name : " << productName[i] << endl;

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

cout << "Product Category : " << productCategory[i] << endl;

cout << "Product Reviews : " << productReviews[i] << "\n\n";

cout << "Product Description : " << productDetails[i] << endl

<< endl;

}

}

}

//-----------------Payment and Shipping / Menu 7

void sellerMenu7()

{

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

SetConsoleTextAttribute(hConsole, 9);

int shipCompany;

int payMeth;

cout << "How much Shipping Companies will you collaborate : ";

cin >> shipCompany;

cout << "How much Payment Methods will you accept : ";

cin >> payMeth;

cout << endl;

paymentMethods = payMeth;

shippingCompany = shipCompany;

cout << "Payment Details ==> " << endl;

cin.ignore();

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

{

cout << "Enter the " << i + 1 << " Payment Method : ";

getline(cin, payment[i]);

}

cout << "\nShipping Details ==> " << endl;

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

{

cout << "Enter the " << i + 1 << " shipping company name : ";

getline(cin, shipping[i]);

}

cout << "\nPayment and Shipping Details : " << endl

<< endl;

cout << "Payment Options : \n";

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

{

cout << i + 1 << ". " << payment[i] << "\n";

}

cout << "\n";

cout << "Shipping Options : \n";

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

{

cout << i + 1 << ". " << shipping[i] << "\n";

}

cout << "\n";

}

// File Handling for seller Menu7 for storing

void StoreSellerMenu7()

{

fstream file;

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

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

{

file << shipping[i] << "\n";

}

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

{

file << payment[i] << "\n";

}

file.close();

}

// File Handling for seller Menu7 for loading

void LoadSellerMenu7()

{

int count = 0;

int counter = 0;

fstream file;

string ship;

string pay;

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

while (getline(file, ship) && count != 3)

{

shipping[shippingCompany] = ship;

count++;

}

while (getline(file, pay) && counter != 3)

{

payment[paymentMethods] = pay;

counter++;

}

file.close();

}

//-----------------Discounts / Menu 8

void sellerMenu8()

{

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

SetConsoleTextAttribute(hConsole, 10);

cout << "1. Enter the number of items to give discount to customers : ";

cin >> numberOfItemsDiscount;

cout << " Enter the discount percentage for Items : ";

cin >> discountItemPercentage;

cout << endl

<< endl;

cout << "2. Enter the total Checkout to give Discount : ";

cin >> checkOutDiscount;

cout << " Enter the discount percentage for Checkout : ";

cin >> discountCheckOutPercentage;

cout << endl

<< endl;

cout << "3. Enter the Coupon Code to give Discounts : ";

cin >> couponDiscount;

cout << " Enter the discount percentage for Coupons : ";

cin >> discountCouponPercentage;

cout << endl

<< endl;

cout << "DISCOUNT Details : \n";

cout << "1.Buy " << numberOfItemsDiscount << " Items to AVAIL " << discountItemPercentage << "% DISCOUNT.\n";

cout << "2.CheckOut a Total of " << checkOutDiscount << " to AVAIL " << discountCheckOutPercentage << "% DISCOUNT.\n";

cout << "3.Apply " << couponDiscount << " as a Coupon at the checkout to AVAIL " << discountCouponPercentage << "% DISCOUNT.\n";

}

// File Handling for seller Menu8 for storing

void StoreSellerMenu8()

{

fstream file;

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

file << numberOfItemsDiscount << "\n";

file << discountItemPercentage << "\n";

file << checkOutDiscount << "\n";

file << discountCheckOutPercentage << "\n";

file << couponDiscount << "\n";

file << discountCouponPercentage << "\n";

file.close();

}

// File Handling for seller Menu7 for Loading

void LoadSellerMenu8()

{

fstream file;

string itemDis;

string itemDisPercent;

string checkoutDis;

string checkoutDisPercent;

string couponDis;

string couponDisPercent;

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

while (getline(file, itemDis) && getline(file, itemDisPercent) && getline(file, checkoutDis) && getline(file, checkoutDisPercent) && getline(file, couponDis) && getline(file, couponDisPercent))

{

numberOfItemsDiscount = stoi(itemDis);

discountItemPercentage = stof(itemDisPercent);

checkoutDis = stoi(checkoutDis);

discountCheckOutPercentage = stof(checkoutDisPercent);

couponDiscount = couponDis;

discountCouponPercentage = stof(couponDisPercent);

}

file.close();

}

//-----------------Customer Care / Menu 9

void sellerMenu9()

{

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

SetConsoleTextAttribute(hConsole, 1);

cout << "Q 1: How to remove a product ? \n";

cout << "Q 2: How to accept the order of customer? \n";

cout << "Q 3: How to give coupons to customers ? \n";

cout << "Q 4: How to increase the products ? \n\n";

cout << "Enter your query : ";

int option;

cin >> option;

cout << "\n\n";

sellerMenu9SubMenu1(option);

sellerMenu9SubMenu2();

}

// SellerMenu9 Sub Menu

void sellerMenu9SubMenu1(int option)

{

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

SetConsoleTextAttribute(hConsole, 2);

if (option == 1)

{

cout << "Select Option 4 from the seller menu and enter the product name and the product will be removed\n\n";

}

else if (option == 2)

{

cout << "Select Option 3 from the seller menu and accept the order of customer\n\n";

}

else if (option == 3)

{

cout << "Select Option 7 from the seller menu and enter the coupon name.\n\n";

}

else if (option == 4)

{

cout << "Select Option 1 from the seller menu and list your new products.\n\n";

}

else

{

cout << "Wait for the answer.. (^.^)\n\n";

}

}

// SellerMenu9 Sub Menu

void sellerMenu9SubMenu2()

{

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

SetConsoleTextAttribute(hConsole, 2);

char answer;

cout << "Are you Satisfied with the answer ?\n (Press Y for yes or Press N for no)";

cin >> answer;

cout << "\n\n";

if (answer == 'Y' || answer == 'y')

{

cout << " " << emoji << emoji << emoji << " Thanks for your Response " << emoji << emoji << emoji << endl;

cout << " We are HAPPY to see your Response\n";

}

else if (answer == 'N' || answer == 'n')

{

cout << " " << emoji << emoji << emoji << " Thanks for your Response " << emoji << emoji << emoji << endl;

cout << " We will try better next time\n";

}

}

// All Seller Options

int sellerMenu()

{

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

SetConsoleTextAttribute(hConsole, 2);

int option;

cout << "Select one of the following options number..." << endl

<< endl;

cout << "1\tList the Products" << endl; // done

cout << "2.\tSearch a Product" << endl; // done

cout << "3.\tOrder Fulfillment" << endl;

cout << "4.\tRemove a Product" << endl;

cout << "5.\tUpdate a Product" << endl;

cout << "6.\tShow All Products" << endl;

cout << "7.\tPayment and Shipping" << endl;

cout << "8.\tDiscounts" << endl;

cout << "9.\tCustomer Care" << endl;

cout << "10.\tExit" << endl

<< endl;

cout << "Your Option..";

cin >> option;

return option;

}

//-----------------------CUSTOMER OPTION----------------------//

void customerInterface()

{

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

SetConsoleTextAttribute(hConsole, 4);

int customerOption = 0;

while (customerOption != 10)

{

topHeader();

subMenu("Customer");

customerOption = customerMenu();

//-----------Search a Product

if (customerOption == 1)

{

system("cls");

topHeader();

subMenu("Search Product");

customerMenu1();

}

//----------Product Under Budget

if (customerOption == 2)

{

system("cls");

topHeader();

subMenu("Product Under Budget");

customerMenu2();

}

//----------Add to cart

if (customerOption == 3)

{

system("cls");

topHeader();

subMenu("Add to Cart");

customerMenu3();

StoreCustomerMenu3();

}

//----------Show Cart and Request Order

if (customerOption == 4)

{

system("cls");

topHeader();

subMenu("Show Cart and Request Order");

customerMenu4();

}

//----------Shipping Details

if (customerOption == 5)

{

system("cls");

topHeader();

subMenu("Shipping Details");

customerMenu5();

StoreCustomerMenu5();

}

//----------Generate Bill

if (customerOption == 6)

{

system("cls");

topHeader();

subMenu("Generate Bill");

customerMenu6();

}

//----------Track Order

if (customerOption == 7)

{

system("cls");

topHeader();

subMenu("Tack Order");

customerMenu7();

StoreCustomerMenu7();

}

//----------Give Review

if (customerOption == 8)

{

system("cls");

topHeader();

subMenu("Give Review");

customerMenu8();

}

//----------Customer Care

if (customerOption == 9)

{

system("cls");

topHeader();

subMenu("Customer Care");

customerMenu9();

}

clearScreen();

}

}

//---------------Search Product-------------//

void customerMenu1()

{

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

SetConsoleTextAttribute(hConsole, 4);

if (productCount == 0)

{

cout << "No Product Found" << emoji;

}

else

{

string searchProductName;

cout << "Enter the Product Name : ";

cin.ignore();

getline(cin, searchProductName);

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

{

if (searchProductName == productName[i])

{

cout << i + 1 << ".\n";

cout << "Brand Name : " << productBrand[i] << endl;

cout << "Product Name : " << productName[i] << endl;

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

cout << "Product Category : " << productCategory[i] << endl;

cout << "Product Reviews : " << productReviews[i] << "\n";

cout << "Product Description : " << productDetails[i] << "\n\n";

}

}

cout << "\n\n";

}

}

//---------------Product Under Budget-------------//

void customerMenu2()

{

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

SetConsoleTextAttribute(hConsole, 5);

int searchProductBudget;

string productUnderBudgetName;

cout << "Enter your Budget : ";

cin >> searchProductBudget;

cout << "\n";

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

{

if (searchProductBudget >= productPrice[i])

{

cout << "Product " << i + 1 << ". " << productName[i] << "\n";

}

}

cout << "\n";

cout << "Enter the product name to find its details : ";

cin.ignore();

getline(cin, productUnderBudgetName);

cout << "\n\n";

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

{

if (productUnderBudgetName == productName[i])

{

cout << "Brand Name : " << productBrand[i] << endl;

cout << "Product Name : " << productName[i] << endl;

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

cout << "Product Category : " << productCategory[i] << endl;

cout << "Product Reviews : " << productReviews[i] << endl;

cout << "Product Description : " << productDetails[i] << endl;

}

}

}

//---------------Add to Cart-------------//

void customerMenu3()

{

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

SetConsoleTextAttribute(hConsole, 6);

char placeCartOption;

string searchCartProductName;

cout << "Enter the Product Name : ";

cin.ignore();

getline(cin, searchCartProductName);

cout << "\n\n";

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

{

if (searchCartProductName == productName[i])

{

cout << i + 1 << ".\n";

cout << "Brand Name : " << productBrand[i] << endl;

cout << "Product Name : " << productName[i] << endl;

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

cout << "Product Category : " << productCategory[i] << endl;

cout << "Product Reviews : " << productReviews[i] << "\n";

cout << "Product Description : " << productDetails[i] << "\n\n";

cout << "Enter A to ADD TO CART or any other to continue......";

cin >> placeCartOption;

if (placeCartOption == 'a' || placeCartOption == 'A')

{

cout << "Enter the product number to add to cart : ";

cin >> productCartNumber;

addToCartProducts[cartCount] = i;

cartCount++;

}

}

}

}

//--------------File Handling for Customer Menu3 Storing

void StoreCustomerMenu3()

{

fstream file;

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

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

{

int index = addToCartProducts[i];

file << index << "\n";

file << productBrand[index] << "\n";

file << productName[index] << "\n";

file << productCategory[index] << "\n";

file << productPrice[index] << "\n";

file << productDetails[index] << "\n";

file << productReviews[index] << "\n";

}

file.close();

}

//--------------File Handling for Customer Menu3 Loading

void LoadCustomerMenu3()

{

fstream file;

string index;

string proBrand;

string proName;

string proCategory;

string proDetails;

string proPrice;

string proReviews;

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

while ((getline(file, index)) && (getline(file, proBrand)) && (getline(file, proName)) && (getline(file, proCategory)) && (getline(file, proPrice)) && (getline(file, proDetails)) && (getline(file, proReviews)))

{

addToCartProducts[cartCount] = stoi(index);

productBrand[productCount] = proBrand;

productName[productCount] = proName;

productCategory[productCount] = proCategory;

productDetails[productCount] = proDetails;

productPrice[productCount] = stof(proPrice);

productReviews[productCount] = stof(proReviews);

productCount++;

cartCount++;

}

file.close();

}

//---------------Show Cart and request a order-------------//

void customerMenu4()

{

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

SetConsoleTextAttribute(hConsole, 7);

int showCart;

int indexNumber;

char placeOrderOption;

// for loop to show the cart

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

{

showCart = addToCartProducts[i];

cout << i + 1 << ".\n";

cout << "Brand Name : " << productBrand[showCart] << endl;

cout << "Product Name : " << productName[showCart] << endl;

cout << "Product Price : " << productPrice[showCart] << endl;

cout << "Product Category : " << productCategory[showCart] << endl;

cout << "Product Reviews : " << productReviews[showCart] << "\n";

cout << "Product Description : " << productDetails[showCart] << "\n\n";

}

cout << "Press P to PLACE THE ORDER or press any key to continue : ";

cin >> placeOrderOption;

if (placeOrderOption == 'P' || placeOrderOption == 'p')

{

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

cin >> numberOfProductsOrdered;

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

{

cout << "\nEnter the product number : ";

cin >> indexNumber;

orderPlacedArray[i] = addToCartProducts[i];

placeOrderedindex = addToCartProducts[indexNumber - 1];

cout << "\nProduct Details ==> \n";

cout << "Brand Name : " << productBrand[placeOrderedindex] << endl;

cout << "Product Name : " << productName[placeOrderedindex] << endl;

cout << "Product Price : " << productPrice[placeOrderedindex] << endl

<< endl;

totalCheckout = totalCheckout + productPrice[placeOrderedindex];

}

cout << "Now Add the Shipping Information and Generate the bill to CONFIRM THE ORDER.\n";

cout << endl;

}

}

//---------------Shipping Details-------------//

void customerMenu5()

{

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

SetConsoleTextAttribute(hConsole, 9);

cout << "Enter your Name : ";

cin.ignore();

getline(cin, nameShipping);

cout << "Enter your Phone Number : ";

getline(cin, numberShipping);

cout << "Enter your city : ";

getline(cin, cityShipping);

cout << "Enter your provice : ";

getline(cin, provinceShipping);

cout << "Enter your E-mail address : ";

getline(cin, emailShipping);

cout << "Enter your Billing Information : ";

getline(cin, billingInfo);

cout << "Enter your Shipping Details here : ";

getline(cin, shippingInfo);

}

//---------------------File Handling for Customer Menu5 Storing

void StoreCustomerMenu5()

{

fstream file;

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

file << nameShipping << endl;

file << numberShipping << endl;

file << cityShipping << endl;

file << provinceShipping << endl;

file << emailShipping << endl;

file << billingInfo << endl;

file << shippingInfo << endl;

file.close();

}

//---------------------File Handling for Customer Menu5 Load

void LoadCustomerMenu5()

{

fstream file;

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

string nameShip, numberShip, cityShip, provinceShip, emailShip, billInfo, shipInfo;

while ((getline(file, nameShip)) && (getline(file, numberShip)) && (getline(file, cityShip)) && (getline(file, provinceShip)) && (getline(file, emailShip)) && (getline(file, billInfo)) && (getline(file, shipInfo)))

{

nameShipping = nameShip;

numberShipping = numberShip;

cityShipping = cityShip;

provinceShipping = provinceShip;

emailShipping = emailShip;

billingInfo = billInfo;

shippingInfo = shippingInfo;

}

file.close();

}

//---------------Generate Bill-------------//

void customerMenu6()

{

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

SetConsoleTextAttribute(hConsole, 10);

string couponCode = "No";

float totalCheckoutAfterDiscount;

string pay;

if (numberOfProductsOrdered == 0)

{

cout << "You have not Order any Product Yet (^.^)\n";

}

else

{

trackingNumber = 1298;

cout << "Your Order Details : \n";

cout << "Your Details ==> \n\n";

cout << "Your Shipping Fee : " << shippingFees << endl

<< endl;

cout << "Your Name : " << nameShipping << endl;

cout << "Your Phone Number : " << numberShipping << endl;

cout << "Your Email Address : " << emailShipping << endl;

cout << endl;

cout << "Bill to : " << billingInfo << endl

<< endl;

cout << "Ship to : " << shippingInfo << endl

<< endl;

cout << nameShipping << " , " << numberShipping << " , " << cityShipping << " , " << provinceShipping << " , " << emailShipping << endl

<< endl;

float bill = totalCheckout + (shippingFees \* numberOfProductsOrdered);

cout << "Your GENERATED BILL : " << bill << endl

<< endl;

cout << "Enter the COUPON CODE Otherwise Enter No : ";

cin.ignore();

getline(cin, couponCode);

if (couponCode == couponDiscount)

{

float percent = ((discountCouponPercentage \* bill) / 100);

totalCheckoutAfterDiscount = bill - percent;

}

else if (numberOfProductsOrdered >= numberOfItemsDiscount)

{

float percent = ((discountItemPercentage \* bill) / 100);

totalCheckoutAfterDiscount = bill - percent;

}

else if (bill >= checkOutDiscount)

{

float percent = ((discountCheckOutPercentage \* bill) / 100);

totalCheckoutAfterDiscount = bill - percent;

}

else

{

totalCheckoutAfterDiscount = bill;

}

cout << "\n\nTOTAL CHECKOUT AFTER DISCOUNT : " << totalCheckoutAfterDiscount << endl

<< endl;

cout << "Enter the Payment Method : ";

cin >> pay;

cout << endl

<< endl

<< endl;

Sleep(500);

cout << " Wait Until Your ORDER is Placed.\n";

Sleep(3000);

cout << " " << emoji << emoji << emoji << " Your ORDER has been placed SUCCESSFULLY " << emoji << emoji << emoji;

cout << "\n\n\nYour Tracking number is : " << trackingNumber << endl;

}

}

//---------------Track Order-------------//

void customerMenu7()

{

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

SetConsoleTextAttribute(hConsole, 1);

int track;

cout << "Enter your Tracking Number : ";

cin >> track;

if (track == 1298)

{

cout << "Your Tracking Information : \n\n";

cout << "Your Tracking Number : " << track << "\n\n";

cout << "In Progress : " << inProgress << "\n";

cout << "Shipped : " << shipped << "\n";

cout << "Out for Delivery : " << delivered << "\n";

}

else

{

cout << "There are no records yet.Please enter a valid tracking number.\n";

}

}

//--------------------File Handling for Customer Menu 7 Storing

void StoreCustomerMenu7()

{

fstream file;

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

file << inProgress << "\n";

file << shipped << "\n";

file << delivered << "\n";

file.close();

}

void LoadCustomerMenu7()

{

fstream file;

string inProg, shipped, deli;

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

while (getline(file, inProg) && getline(file, shipped) && getline(file, delivered))

{

inProgress = inProg;

shipped = shipped;

delivered = delivered;

}

file.close();

}

//---------------Give Review-------------//

void customerMenu8()

{

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

SetConsoleTextAttribute(hConsole, 2);

if (delivered == "yes")

{

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

{

int index = orderPlacedArray[i];

cout << "\nProduct Name : " << productName[index] << endl;

cout << "Give rating out of 5.0 : ";

cin >> productReviews[index];

cout << endl;

}

cout << "\nThnks Alot for your response " << emoji << emoji << emoji << "\n\n";

}

else

{

cout << "There are no records yet.\n";

}

}

//---------------Customer care-------------//

void customerMenu9()

{

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

SetConsoleTextAttribute(hConsole, 3);

cout << "Q 1: How to place the order ? \n";

cout << "Q 2: How to have checkout discount ? \n";

cout << "Q 3: How to apply Coupon Code ? \n";

cout << "Q 4: how to apply price filter ? \n";

cout << "Enter your query : ";

int option;

cin >> option;

cout << "\n\n";

customerMenu9SubMenu1(option);

customerMenu9SubMenu2();

}

// customerMenu9 sub menu1

void customerMenu9SubMenu1(int option)

{

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

SetConsoleTextAttribute(hConsole, 4);

if (option == 1)

{

cout << "Select Option 6 from the customer menu at the time of checkout\n\n";

}

else if (option == 2)

{

cout << "At Option 6,while generating the bill the discount will automatiocally be calculated.\n\n";

}

else if (option == 3)

{

cout << "Select Option 6,at the time of check out,enter the coupon code to avail discount.\n\n";

}

else if (option == 4)

{

cout << "Select Option 2 and appy your desired price filter.\n\n";

}

else

{

cout << "Wait for the answer.." << emoji << endl

<< endl;

}

}

// customerMenu9 sub menu2

void customerMenu9SubMenu2()

{

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

SetConsoleTextAttribute(hConsole, 2);

char answer;

cout << "Are you Satisfied with the answer ?\n (Press Y for yes or Press N for no)";

cin >> answer;

cout << "\n\n";

if (answer == 'Y' || answer == 'y')

{

cout << " " << emoji << emoji << emoji << " Thanks for your Response " << emoji << emoji << emoji << endl;

cout << " We are HAPPY to see your Response\n";

}

else if (answer == 'N' || answer == 'n')

{

cout << " " << emoji << emoji << emoji << " Thanks for your Response " << emoji << emoji << emoji << endl;

cout << " We will try better next time\n";

}

}

// All customer Options

int customerMenu()

{

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

SetConsoleTextAttribute(hConsole, 2);

int option;

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

cout << "1. \tSearch Product" << endl;

cout << "2. \tProduct Under Budget" << endl;

cout << "3. \tAdd to Cart" << endl;

cout << "4. \tShow Cart and Request a Order" << endl;

cout << "5. \tShipping Details" << endl;

cout << "6. \tGenerate Bill" << endl;

cout << "7. \tTrack Order" << endl;

cout << "8. \tGive Review" << endl;

cout << "9. \tCustomer Care" << endl;

cout << "10. \tExit" << endl;

cout << endl;

cout << "Your Option...";

cin >> option;

cout << endl;

return option;

}

* **Weakness in the Business Application**
  + Two or more customers cannot buy products at same time.
* **Future Directions**
  + I will apply filter option by any category and I will checkout the payment through bank account at instant and if COD selected then checkout payment after some time.

**Student Reg. No. :**  2022 R/2021-CS-199  **Student Name.**  Muhammad Furqan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |