# 

**Project: 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-103 Object Oriented Programming

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

Table of Contents

[ Fashion Is U 3](#_Toc164713522)

[ Users Of Application 3](#_Toc164713523)

[ Features Lists 3](#_Toc164713524)

[ WireFrames 5](#_Toc164713525)

[ CRC Model 17](#_Toc164713526)

[ Complete Code 18](#_Toc164713527)

# 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 three users:

* Customer: This user will have access to all the clothing items that are available also maintenance of their Cart.
* Employee: This user manages the Clothes and check Orders.
* Admin: This user manages the Employees.

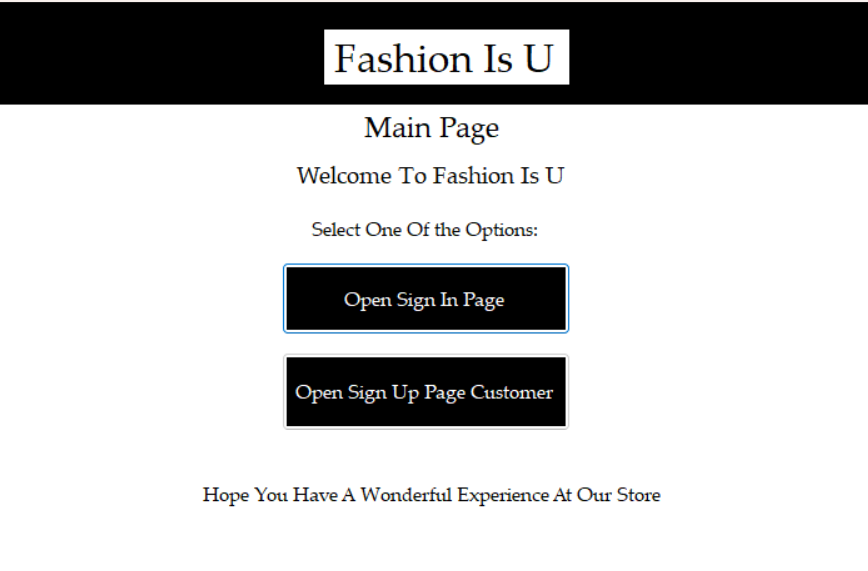
# Features Lists

|  |  |  |  |
| --- | --- | --- | --- |
| 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. Update an Item | Update any of the Clothes |
| 4. Delete an Item | Deletion in items of clothing menu |
| 5. View List of Customers | Display the list of all Customers |
| 6. View List of Orders of a Customer | Display the All The Orders of a Selected Customer |
| 7. Check Reviews | Display the Reviews Of Selected Item |
| 8. Update Your Profile | Make changes In Your Profile |
| 9. 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.Remove an Item From  Cart | Remove An Item from their Cart |
| 4.Change the Quantity | Change the Quantity Of Cart Items |
| 5.Place Order | This will allow the customer to place the order on their cart. |
| 6. Check Previous Orders | This Display all orders of the Customer |
| 7. Leave A Review | This allows the customer to leave review on Item |
| 8. Find Total Amount Spent | The customer is able to check the Total Amount they have spent on the store. |
| 9. Update Profile | The customer can give update their profile. |
| 10. Log Out | It logs the user out. |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | User Type | Required Function | Result of Action Performed |
| 3 | Admin | 1. Add an Employee | Allows Admin to add Employee |
| 1. Display Employee | Display List of All Employees |
| 3. Update Employee | Update Details of an Employee |
| 4. Remove Employee | Removes An Employee |
| 5. View List of Customers | Display the list of all Customers |

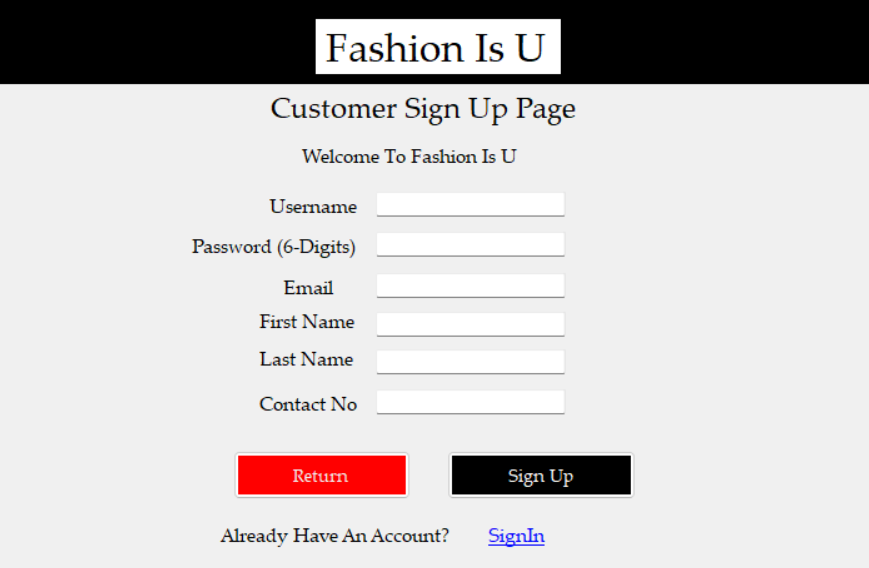
# WireFrames



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



**Figure 2: Sign In Menu**



**Figure 3: Customer Sign Up Menu**



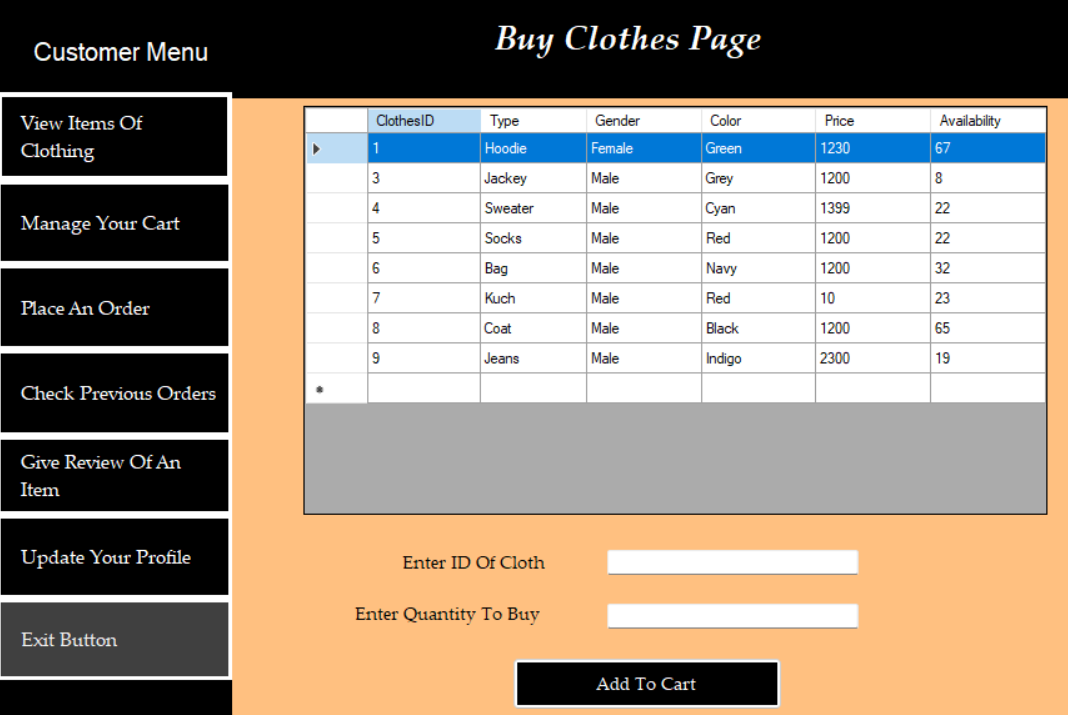
**Figure 4: Customer Menu**



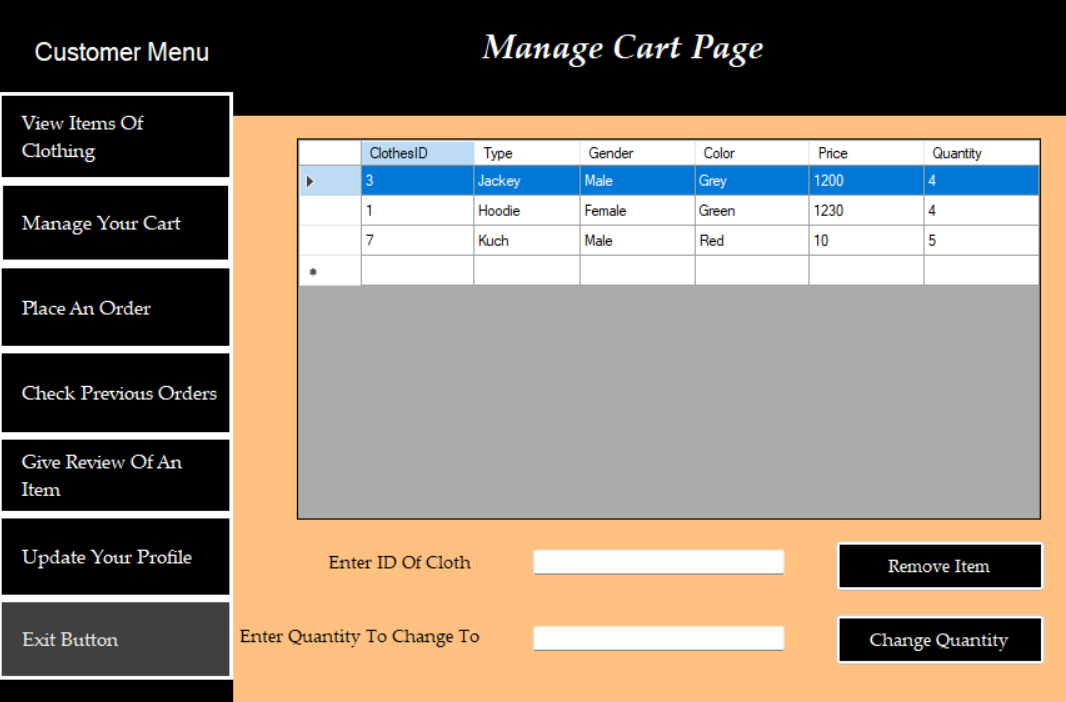
**Figure 5: Employee Menu**



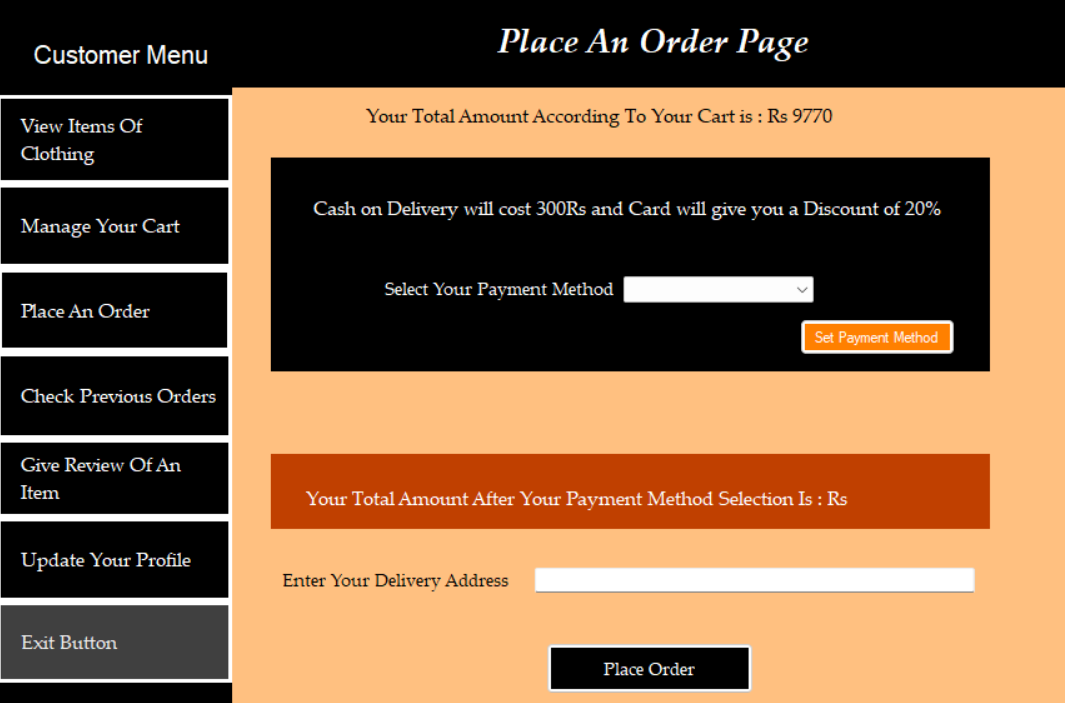
**Figure 6: Employee Menu**



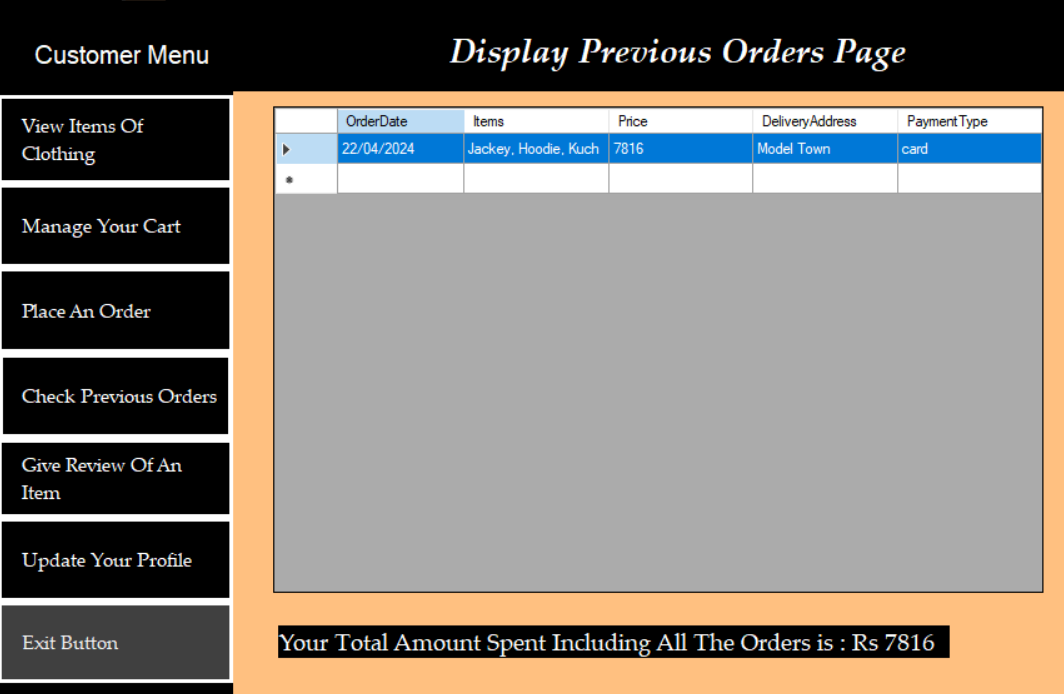
**Figure 7: Buy Clothes Page**



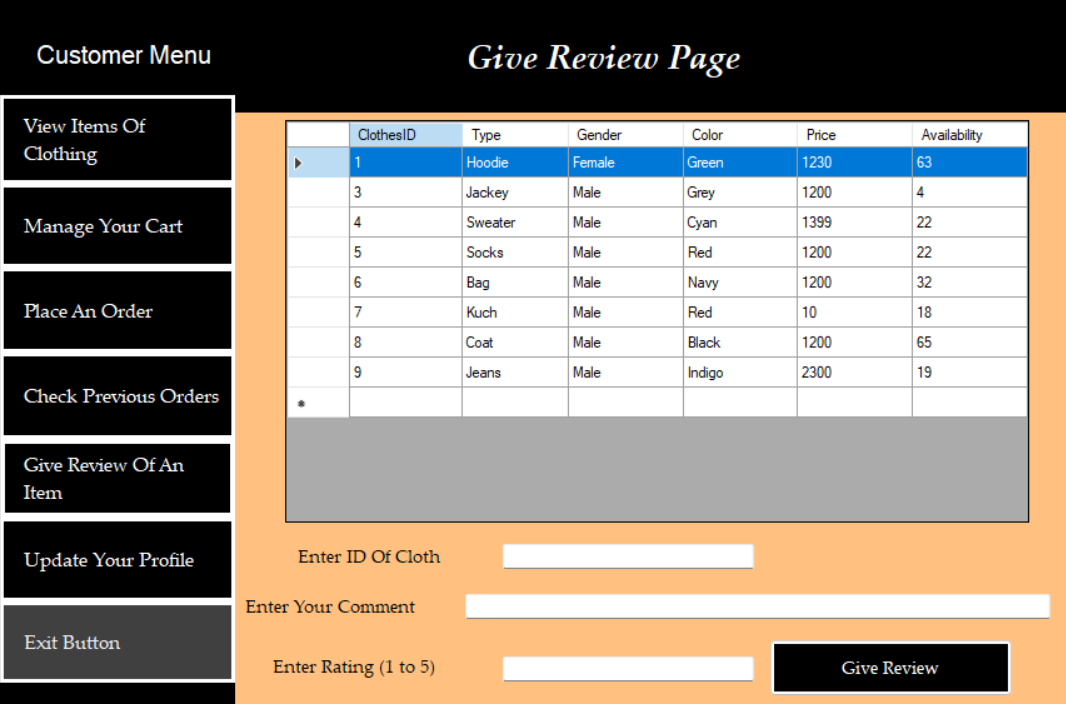
**Figure 8: Manage Cart Page**



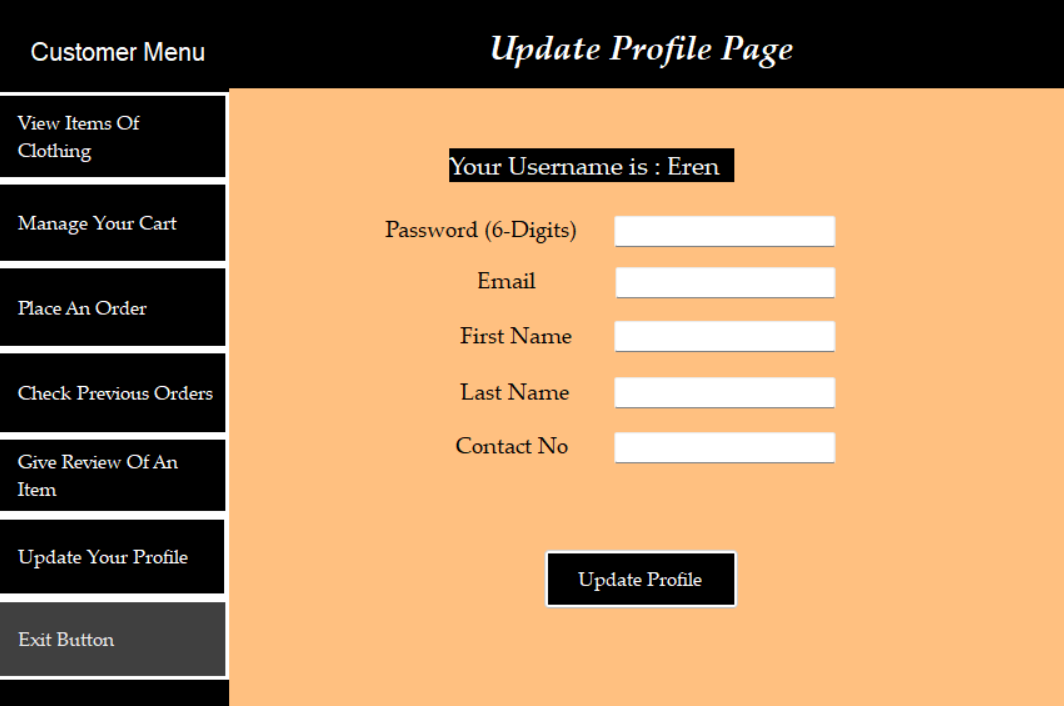
**Figure 9: Place Order Page**



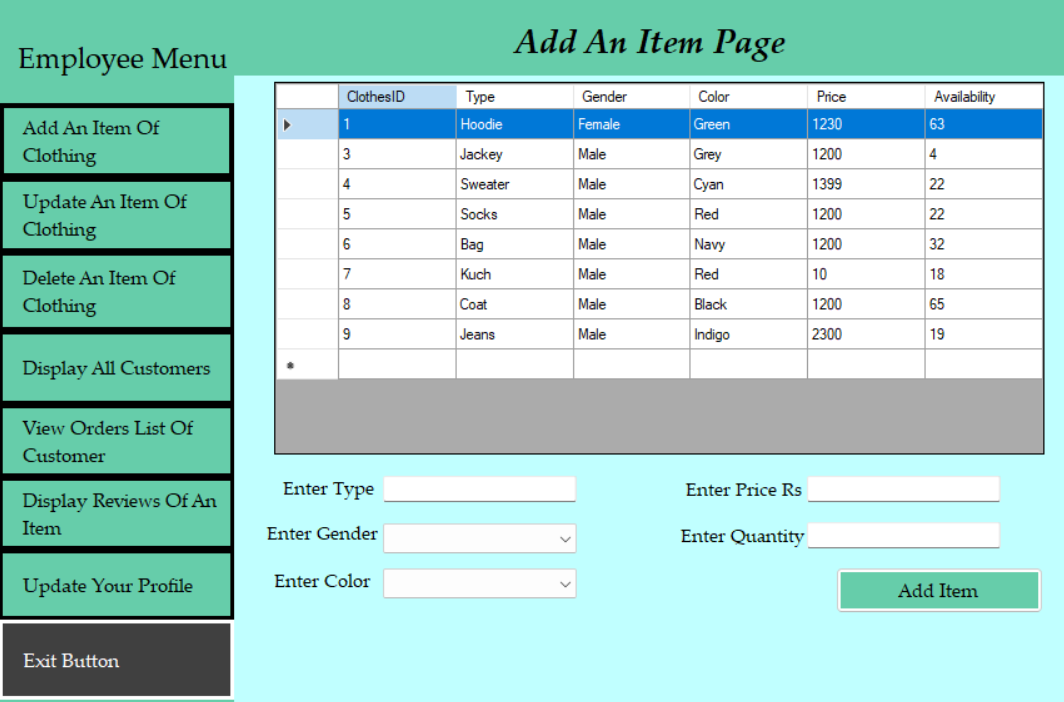
**Figure 10: Display Orders Page**

******

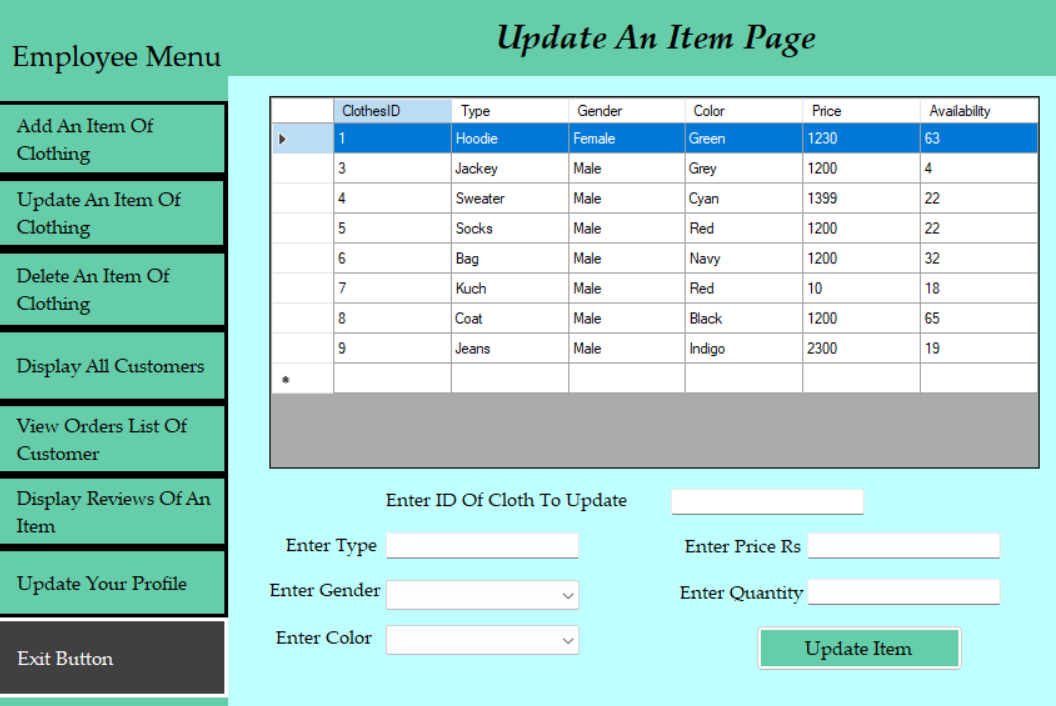
**Figure 11: Give Review Page**



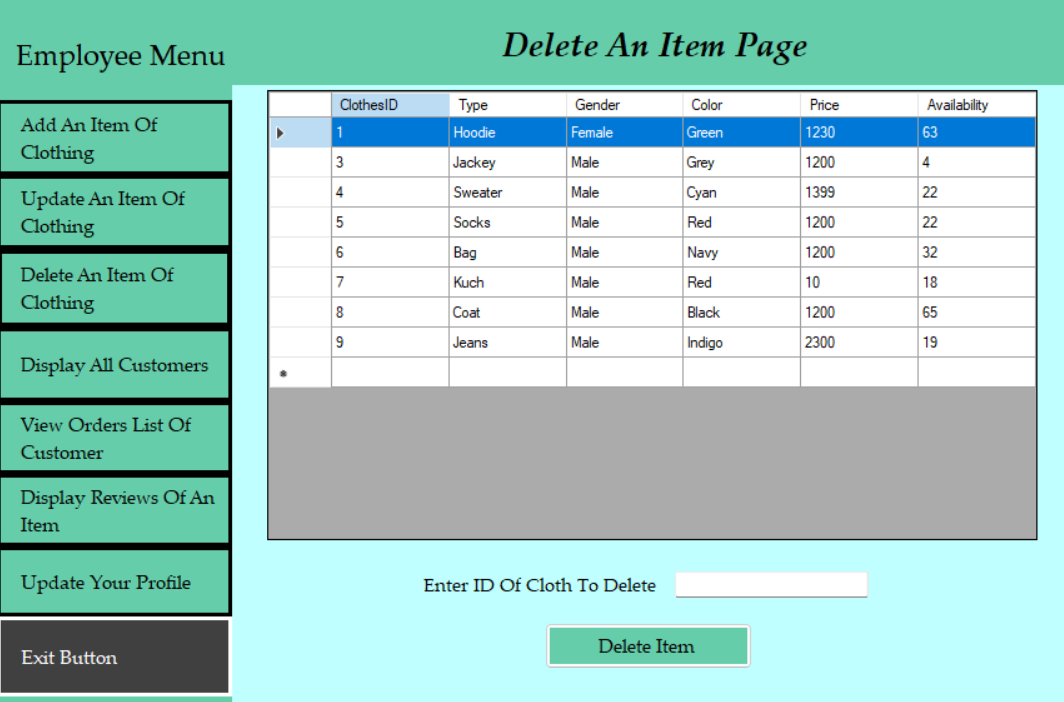
**Figure 12: Update Profile Page (Customer)**



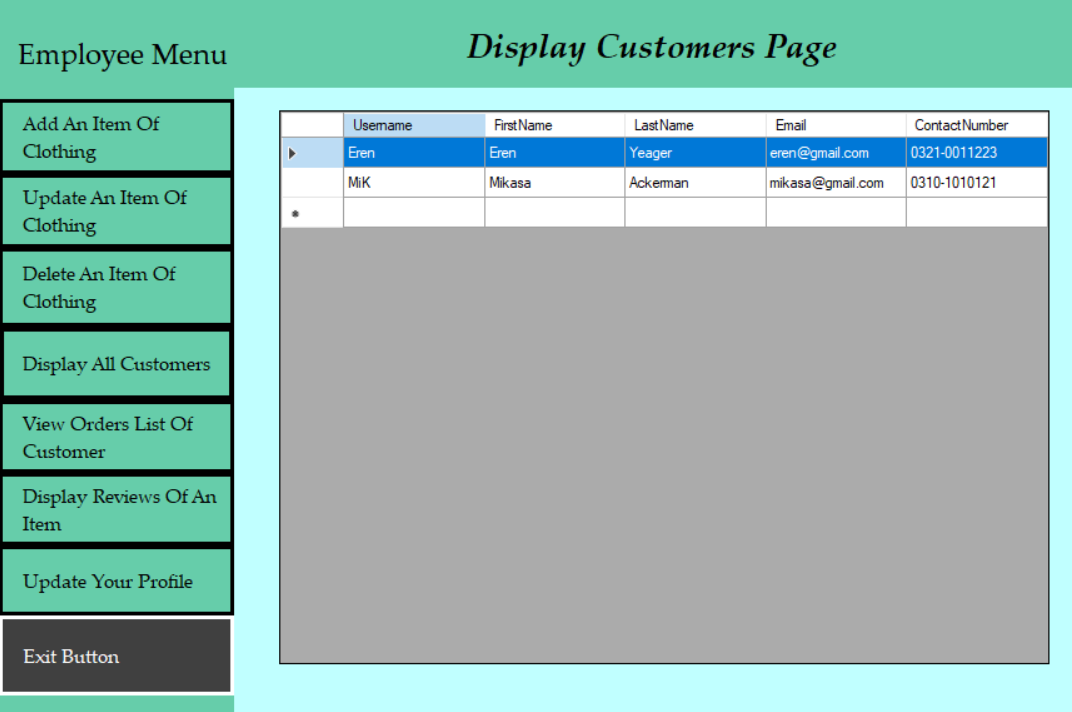
**Figure 13: Add an Item Page**



**Figure 14: Update an Item Page**



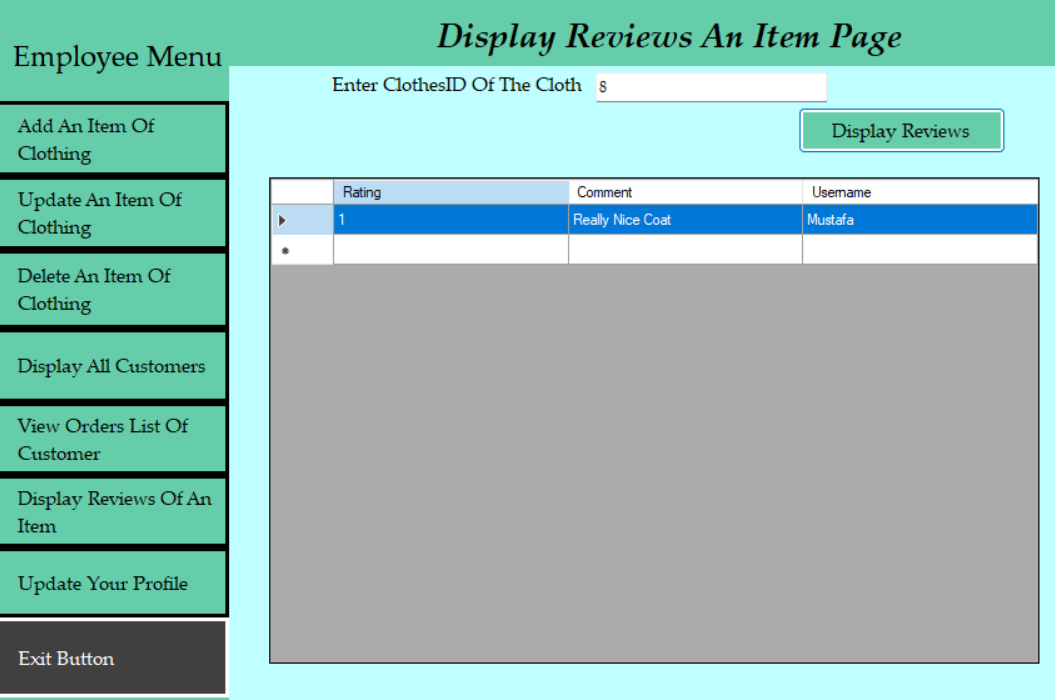
**Figure 15: Delete an Item Page**



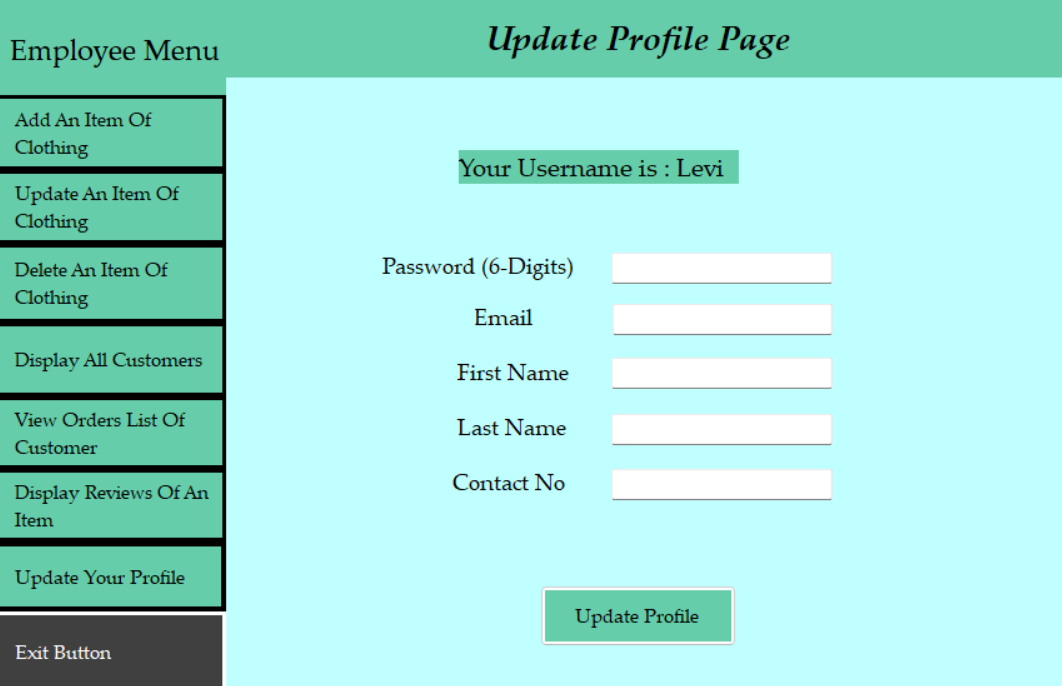
**Figure 16: Display All Customers Page**



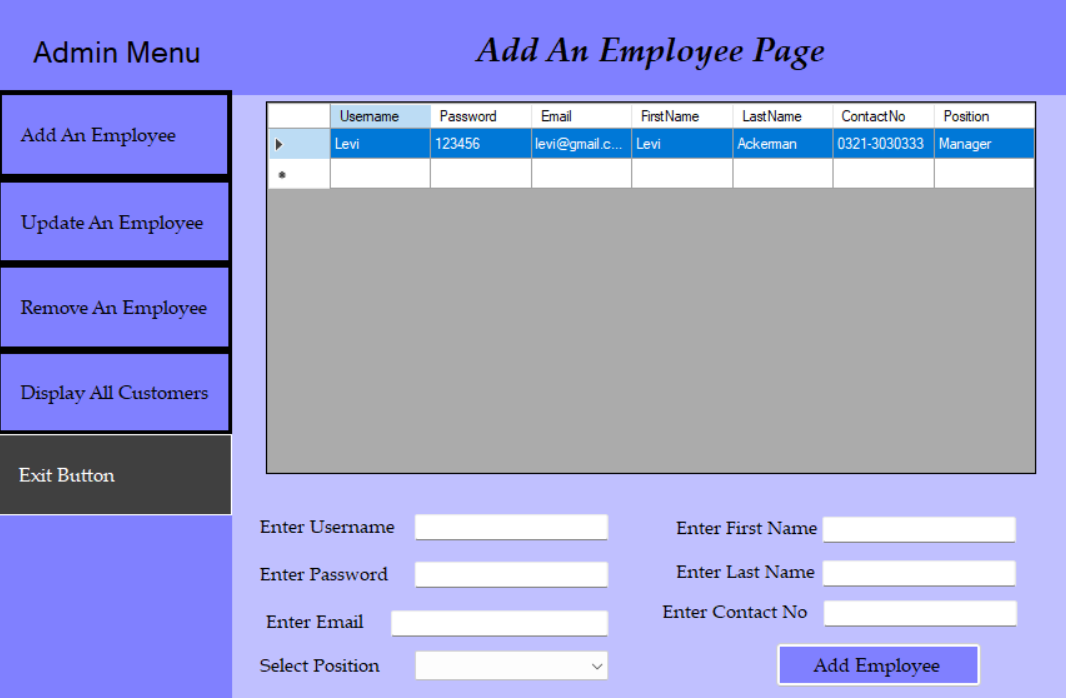
**Figure 17: Display Orders of Customer Page**



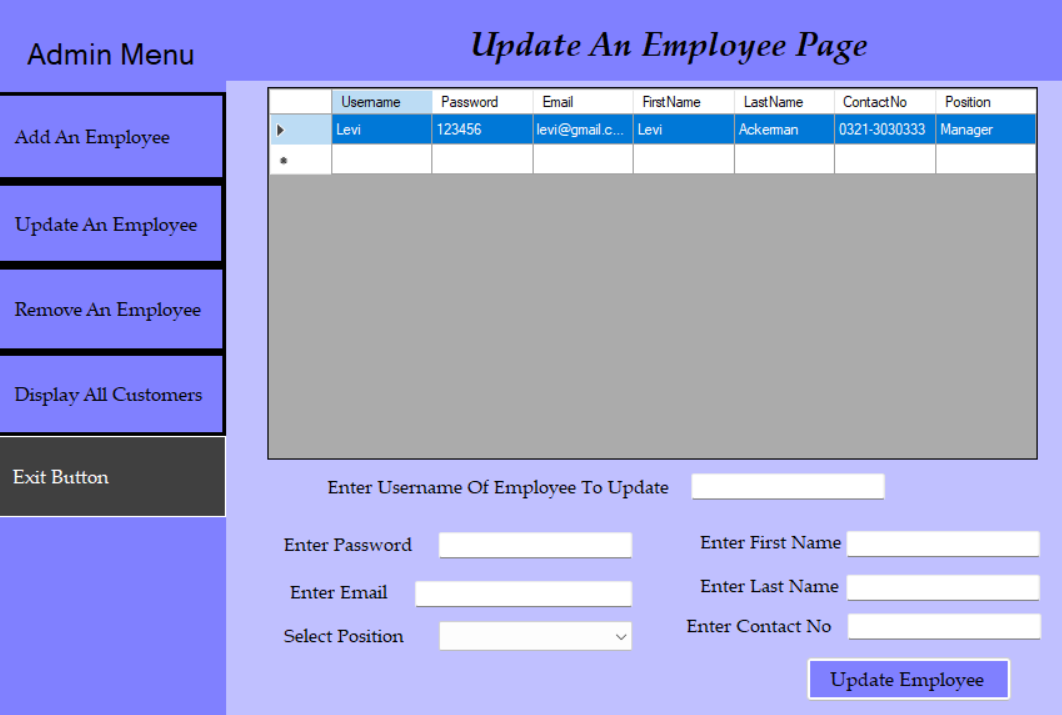
**Figure 18: Display Reviews Page**



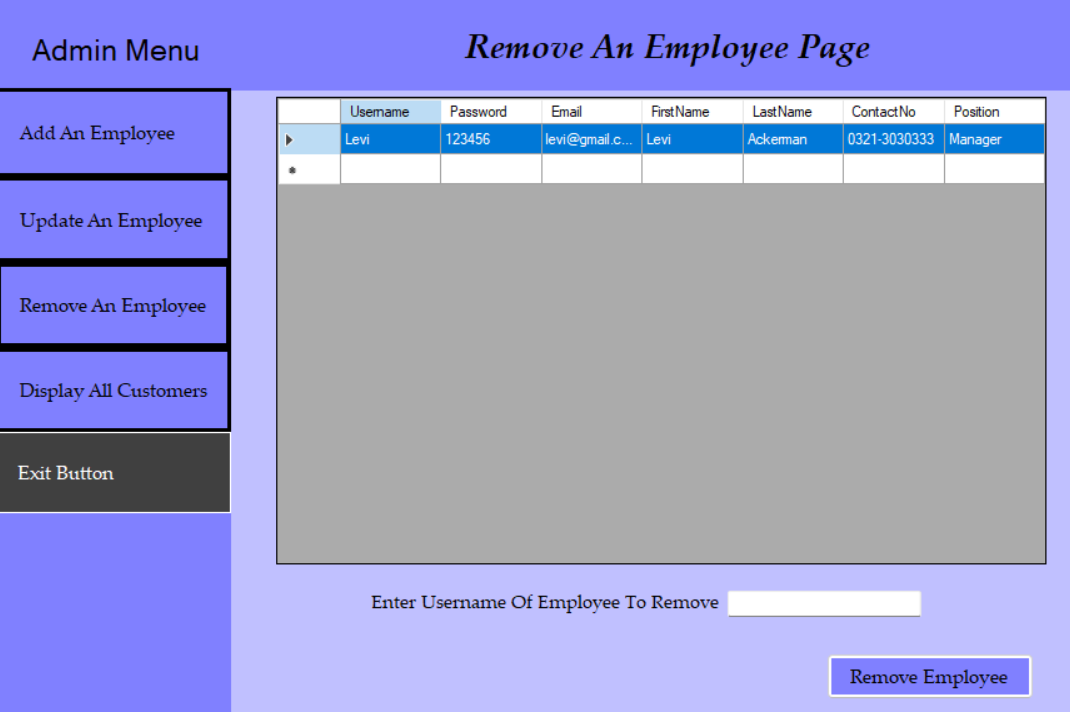
**Figure 19: Update Profile Page (Employee)**



**Figure 20: Add an Employee Page**

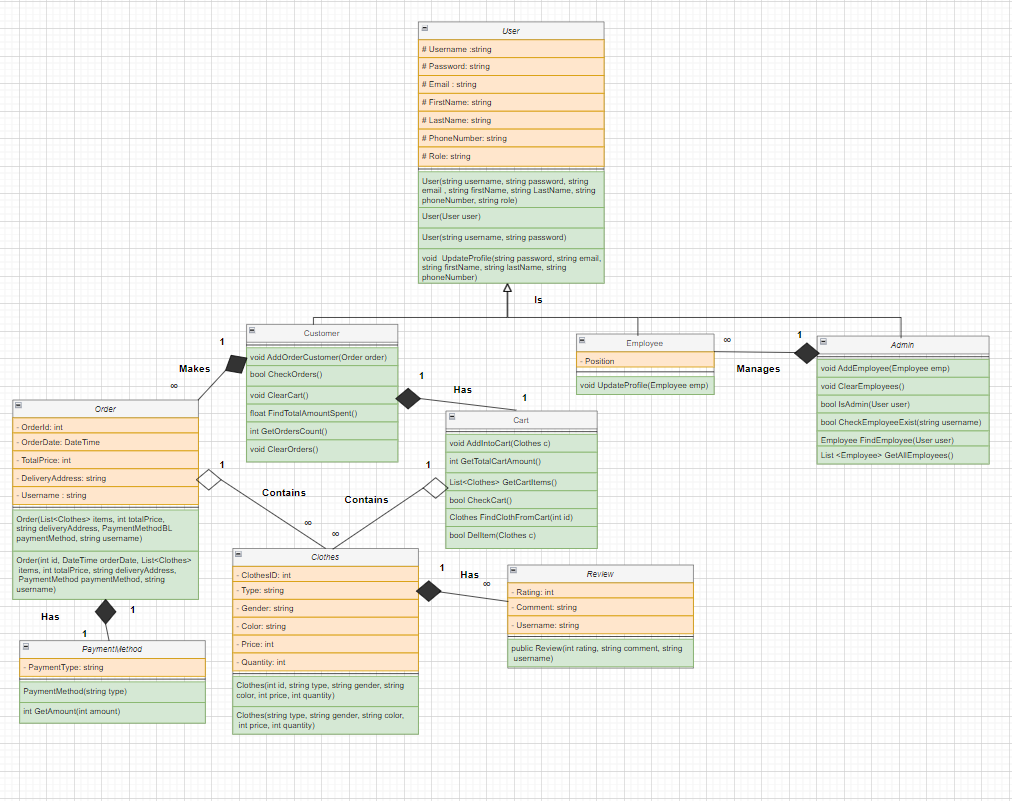


**Figure 21: Update an Employee Page**



**Figure 22: Remove an Employee Page**

# CRC Model



# Complete Code

public abstract class UserBL

{

protected string Username;

protected string Password;

protected string Email;

protected string FirstName;

protected string LastName;

protected string PhoneNumber;

protected string Role;

public UserBL(string username, string password, string email, string firstName, string lastName, string phoneNumber, string role)

{

Username = username;

Password = password;

Email = email;

FirstName = firstName;

LastName = lastName;

PhoneNumber = phoneNumber;

this.Role = role;

}

public UserBL(UserBL user)

{

this.Username = user.GetUsername();

this.Password = user.GetPassword();

this.Email = user.GetEmail();

this.FirstName = user.GetFirstName();

this.LastName = user.GetLastName();

this.PhoneNumber = user.GetPhoneNumber();

this.Role = user.GetRole();

}

public UserBL(string username, string password)

{

Username = username;

Password = password;

}

public string GetRole()

{

return Role;

}

public string GetUsername()

{

return Username;

}

public void SetUsername(string value)

{

Username = value;

}

// Getter and setter methods for Password

public string GetPassword()

{

return Password;

}

public void SetPassword(string value)

{

Password = value;

}

// Getter and setter methods for Email

public string GetEmail()

{

return Email;

}

public void SetEmail(string value)

{

Email = value;

}

// Getter and setter methods for FirstName

public string GetFirstName()

{

return FirstName;

}

public void SetFirstName(string value)

{

FirstName = value;

}

// Getter and setter methods for LastName

public string GetLastName()

{

return LastName;

}

public void SetLastName(string value)

{

LastName = value;

}

// Getter and setter methods for PhoneNumber

public string GetPhoneNumber()

{

return PhoneNumber;

}

public void SetPhoneNumber(string value)

{

PhoneNumber = value;

}

public void UpdateProfile(string password, string email, string firstName, string lastName, string phoneNumber)

{

SetPassword(password);

SetEmail(email);

SetFirstName(firstName);

SetLastName(lastName);

SetPhoneNumber(phoneNumber);

}

}

public class CustomerBL : UserBL

{

private List<OrderBL> Orders;

private CartBL Cart;

public CustomerBL(string username,string password, string email, string firstName,string lastName, string phone,string role) : base(username, password, email, firstName, lastName, phone, role)

{

Orders = new List<OrderBL>();

Cart = new CartBL();

}

public CustomerBL(string username, string password) : base(username, password) { }

public CustomerBL(UserBL u) : base(u)

{

Orders = new List<OrderBL>();

Cart = new CartBL();

}

public CartBL GetCart()

{

return Cart;

}

public List<OrderBL> GetOrderList()

{

return Orders;

}

public void SetOrderList(List <OrderBL> Orders)

{

this.Orders = Orders;

}

public void AddOrderCustomer(OrderBL order)

{

Orders.Add(order);

}

public bool CheckOrders()

{

return Orders.Count > 0;

}

public void ClearCart()

{

Cart.GetCartItems().Clear();

}

public float FindTotalAmountSpent()

{

float totalAmount = 0;

foreach(OrderBL order in Orders)

{

totalAmount += order.GetTotalPrice();

}

return totalAmount;

}

public int GetOrdersCount()

{

return Orders.Count;

}

public void ClearOrders()

{ Orders.Clear(); }

}

public class EmployeeBL : UserBL

{

public EmployeeBL(string username, string password, string email, string firstName, string lastName , string phoneNumber, string role) : base(username, password, email, firstName, lastName, phoneNumber, role)

{

}

public EmployeeBL(string username, string password) : base(username, password) { }

public EmployeeBL(UserBL user) : base(user)

{

}

}

public class ClothesBL

{

private int ClothesID;

private string Type;

private string Gender;

private string Color;

private int Price;

private int Quantity;

private List<ReviewBL> Reviews;

public ClothesBL(int id, string type, string gender, string color, int price, int availability)

{

ClothesID = id;

Type = type;

Gender = gender;

Color = color;

Price = price;

Quantity = availability;

Reviews = new List<ReviewBL>();

}

public ClothesBL(string type, string gender, string color, int price, int availability)

{

Type = type;

Gender = gender;

Color = color;

Price = price;

Quantity = availability;

Reviews = new List<ReviewBL>();

}

public ClothesBL(ClothesBL c)

{

ClothesID = c.GetId();

Type = c.GetType();

Gender = c.Gender;

Color = c.Color;

Price = c.Price;

Quantity = c.Quantity;

}

public int GetId()

{

return ClothesID;

}

public void SetId(int id)

{

ClothesID = id;

}

public string GetType()

{

return Type;

}

public void SetType(string type)

{

Type = type;

}

public string GetGender()

{

return Gender;

}

public void SetGender(string gender)

{

Gender = gender;

}

public string GetColor()

{

return Color;

}

public void SetColor(string color)

{

Color = color;

}

public int GetPrice()

{

return Price;

}

public void SetPrice(int price)

{

Price = price;

}

public int GetQuantity()

{

return Quantity;

}

public void SetQuantity(int availability)

{

Quantity = availability;

}

public bool IsAvailableToBuy(int Quantity)

{

if (Quantity > 0 && Quantity <= this.Quantity)

{ return true; }

return false;

}

public void DropQuantity(int quantity)

{

Quantity -= quantity;

}

public void AddQuantity(int quantity)

{ Quantity += quantity; }

public void AddReview(ReviewBL rev)

{

Reviews.Add(new ReviewBL(rev.GetRating(), rev.GetComment(), rev.GetUsername()));

}

public List<ReviewBL> GetReviews()

{

return Reviews;

}

public void SetReviews(List<ReviewBL> reviews)

{

this.Reviews = reviews;

}

public void ClearReviews()

{ Reviews.Clear(); }

public void UpdateCloth(ClothesBL cloth)

{

SetGender(cloth.GetGender());

SetColor(cloth.GetColor());

SetType(cloth.GetType());

SetPrice(cloth.GetPrice());

SetQuantity(cloth.GetQuantity());

}

}

public class CartBL

{

private List<ClothesBL> Cart;

public CartBL()

{

Cart = new List<ClothesBL>();

}

public void AddIntoCart(ClothesBL c)

{

if(!Cart.Contains(c))

{

Cart.Add(c);

}

}

public int GetTotalCartAmount()

{

int total = 0;

foreach (ClothesBL c in Cart)

{

total += c.GetPrice() \* c.GetQuantity();

}

return total;

}

public List<ClothesBL> GetCartItems()

{

return Cart;

}

public bool CheckCart()

{

return Cart.Count > 0;

}

public ClothesBL FindClothFromCart(int id)

{

foreach(ClothesBL c in Cart)

{

if(c.GetId() == id)

{

return c;

}

}

return null;

}

public bool DelItem(ClothesBL c)

{

foreach(ClothesBL cloth in Cart)

{

if(c == cloth)

{

Cart.Remove(cloth);

return true;

}

}

return false;

}

}

public class OrderBL

{

private int OrderId;

private DateTime OrderDate;

private List<ClothesBL> Items;

private int TotalPrice;

private string DeliveryAddress;

private PaymentMethodBL PaymentMethod;

private string Username;

public OrderBL(List<ClothesBL> items, int totalPrice, string deliveryAddress, PaymentMethodBL paymentMethod, string username)

{

OrderDate = DateTime.Now;

Items = items;

TotalPrice = totalPrice;

DeliveryAddress = deliveryAddress;

this.PaymentMethod = paymentMethod;

this.Username = username;

}

public OrderBL(int id, DateTime orderDate, List<ClothesBL> items, int totalPrice, string deliveryAddress, PaymentMethodBL paymentMethod, string username)

{

OrderId = id;

OrderDate = orderDate;

Items = items;

TotalPrice = totalPrice;

DeliveryAddress = deliveryAddress;

this.PaymentMethod = paymentMethod;

this.Username = username;

}

public int GetId()

{

return OrderId;

}

public string GetUsername()

{

return Username;

}

public void SetUsername(string username) { }

public DateTime GetOrderDate()

{

return OrderDate;

}

public void SetOrderDate(DateTime orderDate)

{

OrderDate = orderDate;

}

public List<ClothesBL> GetItems()

{

return Items;

}

public void SetItems(List<ClothesBL> items)

{

Items = items;

}

public int GetTotalPrice()

{

return TotalPrice;

}

public void SetTotalPrice(int totalPrice)

{

TotalPrice = totalPrice;

}

public string GetDeliveryAddress()

{

return DeliveryAddress;

}

public void SetDeliveryAddress(string deliveryAddress)

{

DeliveryAddress = deliveryAddress;

}

public PaymentMethodBL GetPaymentMethod()

{

return PaymentMethod;

}

}

public class PaymentMethodBL

{

private string PaymentType;

public PaymentMethodBL(string type)

{

PaymentType = type;

}

public int GetAmount(int amount)

{

if (PaymentType.ToLower() == "cash")

{

amount = amount + 300;

}

else

{

amount = (int)(amount - amount \* 0.20);

}

return amount;

}

public void SetPaymentType(string paymentType)

{

PaymentType = paymentType;

}

public string GetPaymentType()

{

return PaymentType;

}

}

public class ReviewBL

{

private int Rating;

private string Comment;

private string Username;

public ReviewBL(int rating, string comment, string username)

{

Rating = rating;

Comment = comment;

Username = username;

}

public int GetRating()

{

return Rating;

}

public void SetRating(int value)

{

Rating = value;

}

public string GetComment()

{

return Comment;

}

public void SetComment(string value)

{

Comment = value;

}

public string GetUsername()

{

return Username;

}

public void SetUsername(string value)

{

Username = value;

}

}

public class CartDB:ICartDL

{

public void SaveItemInCart(ClothesBL cloth, CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

if(!CheckItemInCart(cloth.GetId(), customer))

{

string query = "INSERT INTO Carts (Username, ClothesId, Type, Gender, Color, Price, Quantity) VALUES (@Username, @ClothId, @Type, @Gender, @Color, @Price, @Quantity);";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@Username", customer.GetUsername());

cmd.Parameters.AddWithValue("@ClothId", cloth.GetId()); // Assuming you have a property for ClothId in your OrderItem class

cmd.Parameters.AddWithValue("@Type", cloth.GetType()); // Assuming you have a property for Quantity in your OrderItem class

cmd.Parameters.AddWithValue("@Gender", cloth.GetGender());

cmd.Parameters.AddWithValue("@Color", cloth.GetColor());

cmd.Parameters.AddWithValue("@Price", cloth.GetPrice());

cmd.Parameters.AddWithValue("@Quantity", cloth.GetQuantity());

cmd.ExecuteNonQuery();

}

}

connection.Close();

}

}

public bool CheckItemInCart(int id, CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Carts WHERE ClothesID = @clothID and Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", id);

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

}

public void RetrieveCart(CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Carts WHERE Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int clothesID = Convert.ToInt32(reader["ClothesID"]);

string type = Convert.ToString(reader["Type"]);

string gender = Convert.ToString(reader["Gender"]);

string color = Convert.ToString(reader["Color"]);

int price = Convert.ToInt32(reader["Price"]);

int quantity = Convert.ToInt32(reader["Quantity"]);

ClothesBL cloth = new ClothesBL(clothesID, type, gender, color, price, quantity);

customer.GetCart().AddIntoCart(cloth);

}

connection.Close();

}

}

}

}

public void DeleteAnItem(int id, CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Delete FROM Carts WHERE ClothesID = @clothID and Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", id);

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public void UpdateQuantity(int id, CustomerBL customer, int quantity)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Update Carts Set Quantity = @quantity WHERE ClothesID = @clothID and Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", id);

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

cmd.Parameters.AddWithValue("@quantity", quantity);

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public void EmptyCart(CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Delete from Carts WHERE Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

}

public class CartFH : ICartDL

{

public void SaveItemInCart(ClothesBL cloth, CustomerBL customer)

{

string path = UtilityClass.GetCartsFilePath();

if (!CheckItemInCart(cloth.GetId(), customer))

{

using (StreamWriter f = new StreamWriter(path, true))

{

f.WriteLine(customer.GetUsername() + "," + cloth.GetId() + "," + cloth.GetType() + "," + cloth.GetGender() + "," + cloth.GetColor() + "," + cloth.GetPrice() + "," + cloth.GetQuantity());

f.Flush();

}

}

}

public bool CheckItemInCart(int id, CustomerBL customer)

{

string path = UtilityClass.GetCartsFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

int clothId = Convert.ToInt32(splittedRecord[1]);

if (username == customer.GetUsername() && clothId == id)

{

return true;

}

}

}

}

}

return false;

}

public void RetrieveCart(CustomerBL customer)

{

string path = UtilityClass.GetCartsFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

if (username == customer.GetUsername())

{

int clothesID = Convert.ToInt32(splittedRecord[1]);

string type = splittedRecord[2];

string gender = splittedRecord[3];

string color = splittedRecord[4];

int price = Convert.ToInt32(splittedRecord[5]);

int quantity = Convert.ToInt32(splittedRecord[6]);

ClothesBL cloth = new ClothesBL(clothesID, type, gender, color, price, quantity);

customer.GetCart().AddIntoCart(cloth);

}

}

}

}

}

}

public List<ClothesBL> GetAllClothesInCart(CustomerBL customer)

{

List<ClothesBL> AllClothes = new List<ClothesBL>();

string path = UtilityClass.GetCartsFilePath();

if (File.Exists(path))

{

List<string> linesToRemove = new List<string>();

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

if (username == customer.GetUsername())

{

int clothesID = Convert.ToInt32(splittedRecord[1]);

string type = splittedRecord[2];

string gender = splittedRecord[3];

string color = splittedRecord[4];

int price = Convert.ToInt32(splittedRecord[5]);

int quantity = Convert.ToInt32(splittedRecord[6]);

ClothesBL cloth = new ClothesBL(clothesID, type, gender, color, price, quantity);

AllClothes.Add(cloth);

linesToRemove.Add(record);

}

}

}

}

if (linesToRemove.Count > 0)

{

string[] allLines = File.ReadAllLines(path);

List<string> remainingLines = new List<string>();

foreach (string line in allLines)

{

if (!linesToRemove.Contains(line))

{

remainingLines.Add(line);

}

}

File.WriteAllLines(path, remainingLines);

}

}

return AllClothes;

}

public void DeleteAnItem(int id, CustomerBL customer)

{

string path = UtilityClass.GetCartsFilePath();

List<ClothesBL> AllClothes = GetAllClothesInCart(customer);

for (int i = AllClothes.Count - 1; i >= 0; i--)

{

if (AllClothes[i].GetId() == id)

{

AllClothes.RemoveAt(i);

}

}

foreach (ClothesBL stored in AllClothes)

{

SaveItemInCart(stored, customer);

}

}

public void UpdateQuantity(int id, CustomerBL customer, int quantity)

{

string path = UtilityClass.GetCartsFilePath();

List<ClothesBL> AllClothes = GetAllClothesInCart(customer);

foreach (ClothesBL stored in AllClothes)

{

if (stored.GetId() == id)

{

stored.SetQuantity(quantity);

}

}

foreach (ClothesBL stored in AllClothes)

{

SaveItemInCart(stored, customer);

}

}

public void EmptyCart(CustomerBL customer)

{

string path = UtilityClass.GetCartsFilePath();

if (File.Exists(path))

{

List<string> linesToRemove = new List<string>();

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

if (username == customer.GetUsername())

{

linesToRemove.Add(record);

}

}

}

if (linesToRemove.Count > 0)

{

string[] allLines = File.ReadAllLines(path);

List<string> remainingLines = new List<string>();

foreach (string line in allLines)

{

if (!linesToRemove.Contains(line))

{

remainingLines.Add(line);

}

}

File.WriteAllLines(path, remainingLines);

}

}

}

}

public class ClothesDB:IClothesDL

{

//private List<ClothesBL> TotalClothes = new List<ClothesBL> { };

/\*

public void AddClothes(ClothesBL c)

{

TotalClothes.Add(c);

}

\*/

private static ClothesDB Instance;

private ClothesDB() { }

public static ClothesDB GetClothesDB()

{

if (Instance == null)

{

Instance = new ClothesDB();

}

return Instance;

}

public bool AddClothes(ClothesBL c)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Insert into Clothes (Type, Gender, Color, Price, Quantity) Values('{0}', '{1}', '{2}', {3}, {4})", c.GetType(), c.GetGender(), c.GetColor(), c.GetPrice(), c.GetQuantity());

SqlCommand cmd = new SqlCommand(query, connection);

int rows = cmd.ExecuteNonQuery();

connection.Close();

if (rows > 0)

{

return true;

}

else

{

return false;

}

}

/\*

public bool CheckClothes()

{

return TotalClothes.Count > 0;

}

\*/

public bool CheckClothes()

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Select count(\*) from Clothes");

SqlCommand cmd = new SqlCommand(query, connection);

int count = (int)cmd.ExecuteScalar();

connection.Close();

return count > 0;

}

/\*

public ClothesBL FindClothByID(int id)

{

foreach(ClothesBL c in TotalClothes)

{

if(c.GetId() == id)

{

return c;

}

}

return null;

}

\*/

public ClothesBL FindClothByID(int id)

{

ClothesBL cloth = null;

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Clothes where ClothesId = @ClothesId";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@ClothesId", id);

using (SqlDataReader reader = cmd.ExecuteReader())

{

if(reader.Read())

{

int cID= Convert.ToInt32(reader["ClothesId"]);

string type = Convert.ToString(reader["Type"]);

string gender = Convert.ToString(reader["Gender"]);

string color = Convert.ToString(reader["Color"]);

int price = Convert.ToInt32(reader["Price"]);

int quantity = Convert.ToInt32(reader["Quantity"]);

cloth = new ClothesBL(cID, type, gender, color, price, quantity);

}

connection.Close();

}

}

}

return cloth;

}

public bool CheckClothExistence(ClothesBL c)

{

bool clothExists = false;

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Clothes WHERE LOWER(Type) = LOWER(@Type) AND LOWER(Gender) = LOWER(@Gender) AND LOWER(Color) = LOWER(@Color)";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@Type", c.GetType());

cmd.Parameters.AddWithValue("@Gender", c.GetGender());

cmd.Parameters.AddWithValue("@Color", c.GetColor());

int count = (int)cmd.ExecuteScalar();

if (count > 0)

{

clothExists = true;

}

}

}

return clothExists;

}

public bool CheckClothExistenceByQuantity(ClothesBL c)

{

bool clothExists = false;

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Clothes WHERE LOWER(Type) = LOWER(@Type) AND LOWER(Gender) = LOWER(@Gender) AND LOWER(Color) = LOWER(@Color) AND Quantity = @Quantity";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@Type", c.GetType());

cmd.Parameters.AddWithValue("@Gender", c.GetGender());

cmd.Parameters.AddWithValue("@Color", c.GetColor());

cmd.Parameters.AddWithValue("@Quantity", c.GetQuantity());

int count = (int)cmd.ExecuteScalar();

if (count > 0)

{

clothExists = true;

}

}

}

return clothExists;

}

public void ChangeQuantity(int id, int quantity)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Update Clothes Set Quantity = @quantity WHERE ClothesID = @clothID";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", id);

cmd.Parameters.AddWithValue("@quantity", quantity);

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public void UpdateCloth(ClothesBL cloth)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Update Clothes Set Type = @type, Gender = @gender, Color = @color, Price = @price, Quantity = @quantity WHERE ClothesID = @clothID";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", cloth.GetId());

cmd.Parameters.AddWithValue("@quantity", cloth.GetQuantity());

cmd.Parameters.AddWithValue("@type", cloth.GetType());

cmd.Parameters.AddWithValue("@gender", cloth.GetGender());

cmd.Parameters.AddWithValue("@color", cloth.GetColor());

cmd.Parameters.AddWithValue("@price", cloth.GetPrice());

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

/\*

public List <ClothesBL> GetAllClothes()

{

return TotalClothes;

}

\*/

public List <ClothesBL> GetAllClothes()

{

List <ClothesBL> clothes = new List <ClothesBL>();

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Clothes";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

while(reader.Read())

{

int cID = Convert.ToInt32(reader["ClothesId"]);

string type = Convert.ToString(reader["Type"]);

string gender = Convert.ToString(reader["Gender"]);

string color = Convert.ToString(reader["Color"]);

int price = Convert.ToInt32(reader["Price"]);

int quantity = Convert.ToInt32(reader["Quantity"]);

ClothesBL cloth = new ClothesBL(cID, type, gender, color, price, quantity);

clothes.Add(cloth);

}

connection.Close();

}

}

}

return clothes;

}

/\*

public void DeleteCloth(ClothesBL c)

{

TotalClothes.Remove(c);

}

\*/

public bool DeleteCloth(ClothesBL c)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Clothes WHERE ClothesId = @ClothesId";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@ClothesId", c.GetId());

// Execute the DELETE query

int rowsAffected = cmd.ExecuteNonQuery();

connection.Close();

if (rowsAffected > 0)

{

return true;

}

else

{

return false;

}

}

}

}

}

public class ClothesFH : IClothesDL

{

private static int ClothesId = 0;

public bool AddClothes(ClothesBL c)

{

string path = UtilityClass.GetClothesFilePath();

ClothesId = GetAllClothes().Max(cloth => cloth.GetId())+1;

using (StreamWriter f = new StreamWriter(path, true))

{

if (f != null)

{

f.WriteLine(ClothesId + "," + c.GetType() + "," + c.GetGender() + "," + c.GetColor() + "," + c.GetPrice() + "," + c.GetQuantity());

f.Flush();

return true;

}

}

return false;

}

public bool CheckClothes()

{

int count = 0;

string path = UtilityClass.GetClothesFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

count++;

}

}

}

}

return count > 0;

}

public List<ClothesBL> GetAllClothes()

{

List<ClothesBL> Clothes = new List<ClothesBL>();

string path = UtilityClass.GetClothesFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

int cID = Convert.ToInt32(splittedRecord[0]);

string type = splittedRecord[1];

string gender = splittedRecord[2];

string color = splittedRecord[3];

int price = Convert.ToInt32(splittedRecord[4]);

int quantity = Convert.ToInt32(splittedRecord[5]);

ClothesBL cloth = new ClothesBL(cID, type, gender, color, price, quantity);

Clothes.Add(cloth);

}

}

}

}

return Clothes;

}

public ClothesBL FindClothByID(int id)

{

string path = UtilityClass.GetClothesFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

int cID = Convert.ToInt32(splittedRecord[0]);

string type = splittedRecord[1];

string gender = splittedRecord[2];

string color = splittedRecord[3];

int price = Convert.ToInt32(splittedRecord[4]);

int quantity = Convert.ToInt32(splittedRecord[5]);

if (id == cID)

{

ClothesBL cloth = new ClothesBL(cID, type, gender, color, price, quantity);

return cloth;

}

}

}

}

}

return null;

}

public bool CheckClothExistence(ClothesBL c)

{

string path = UtilityClass.GetClothesFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

string[] splittedRecord = record.Split(',');

string type = splittedRecord[1];

string gender = splittedRecord[2];

string color = splittedRecord[3];

if (type.ToLower() == c.GetType().ToLower() && gender.ToLower() == c.GetGender().ToLower() && color.ToLower() == c.GetColor().ToLower())

{

return true;

}

}

}

}

return false;

}

public bool CheckClothExistenceByQuantity(ClothesBL c)

{

string path = UtilityClass.GetClothesFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string type = splittedRecord[1];

string gender = splittedRecord[2];

string color = splittedRecord[3];

int quantity = Convert.ToInt32(splittedRecord[5]);

if (type.ToLower() == c.GetType().ToLower() && gender.ToLower() == c.GetGender().ToLower() && color.ToLower() == c.GetColor().ToLower() && quantity == c.GetQuantity())

{

return true;

}

}

}

}

}

return false;

}

public void ChangeQuantity(int id, int quantity)

{

string path = UtilityClass.GetClothesFilePath();

List<ClothesBL> AllClothes = GetAllClothes();

foreach (ClothesBL stored in AllClothes)

{

if (stored.GetId() == id)

{

stored.SetQuantity(quantity);

}

}

File.WriteAllText(path, "");

foreach (ClothesBL stored in AllClothes)

{

using (StreamWriter f = new StreamWriter(path, true))

{

f.WriteLine(stored.GetId() + "," + stored.GetType() + "," + stored.GetGender() + "," + stored.GetColor() + "," + stored.GetPrice() + "," + stored.GetQuantity());

f.Flush();

}

}

}

public void UpdateCloth(ClothesBL cloth)

{

string path = UtilityClass.GetClothesFilePath();

List<ClothesBL> AllClothes = GetAllClothes();

foreach (ClothesBL stored in AllClothes)

{

if (stored.GetId() == cloth.GetId())

{

stored.UpdateCloth(cloth);

}

}

File.WriteAllText(path, "");

foreach (ClothesBL stored in AllClothes)

{

using (StreamWriter f = new StreamWriter(path, true))

{

f.WriteLine(stored.GetId() + "," + stored.GetType() + "," + stored.GetGender() + "," + stored.GetColor() + "," + stored.GetPrice() + "," + stored.GetQuantity());

f.Flush();

}

}

}

public bool DeleteCloth(ClothesBL c)

{

string path = UtilityClass.GetClothesFilePath();

List<ClothesBL> AllClothes = GetAllClothes();

bool clothDeleted = false;

foreach (ClothesBL stored in AllClothes)

{

if (stored.GetId() == c.GetId())

{

AllClothes.Remove(stored);

clothDeleted = true;

break;

}

}

if (clothDeleted)

{

File.WriteAllText(path, "");

foreach (ClothesBL stored in AllClothes)

{

using (StreamWriter f = new StreamWriter(path, true))

{

f.WriteLine(stored.GetId() + "," + stored.GetType() + "," + stored.GetGender() + "," + stored.GetColor() + "," + stored.GetPrice() + "," + stored.GetQuantity());

f.Flush();

}

}

}

return clothDeleted;

}

}

public class OrderDB:IOrderDL

{

// private List <OrderBL> TotalOrders = new List <OrderBL> ();

/\*

public void AddOrder(OrderBL order)

{

TotalOrders.Add (order);

}

\*/

public bool AddOrder(OrderBL order)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Insert into Orders (OrderDate, TotalPrice, DeliveryAddress, PaymentType, Username) OUTPUT INSERTED.OrderID Values('{0}', '{1}', '{2}', '{3}', '{4}')", order.GetOrderDate(), order.GetTotalPrice(), order.GetDeliveryAddress(), order.GetPaymentMethod().GetPaymentType(), order.GetUsername());

SqlCommand cmd = new SqlCommand(query, connection);

int orderId = (int)cmd.ExecuteScalar();

connection.Close();

SaveOrderItems(orderId, order);

if(orderId != 0)

{

return true;

}

return false;

}

public void SaveOrderItems(int OrderId, OrderBL order)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

foreach (ClothesBL cloth in order.GetItems())

{

string query = "INSERT INTO OrderItems (OrderId, ClothesId, Type, Gender, Color, Price, Quantity) VALUES (@OrderId, @ClothId, @Type, @Gender, @Color, @Price, @Quantity);";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@OrderId", OrderId);

cmd.Parameters.AddWithValue("@ClothId", cloth.GetId()); // Assuming you have a property for ClothId in your OrderItem class

cmd.Parameters.AddWithValue("@Type", cloth.GetType()); // Assuming you have a property for Quantity in your OrderItem class

cmd.Parameters.AddWithValue("@Gender", cloth.GetGender());

cmd.Parameters.AddWithValue("@Color", cloth.GetColor());

cmd.Parameters.AddWithValue("@Price", cloth.GetPrice());

cmd.Parameters.AddWithValue("@Quantity", cloth.GetQuantity());

cmd.ExecuteNonQuery();

}

}

connection.Close();

}

}

public void RetrieveOrdersOfCustomer(CustomerBL customer)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Orders WHERE Username = @username";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@username", customer.GetUsername());

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int orderID = Convert.ToInt32(reader["OrderID"]);

DateTime orderDate = Convert.ToDateTime(reader["OrderDate"]);

int price = Convert.ToInt32(reader["TotalPrice"]);

string address = Convert.ToString(reader["DeliveryAddress"]);

string paymentType = Convert.ToString(reader["PaymentType"]);

List<ClothesBL> Clothes = GetListOfClothesInOrder(orderID);

OrderBL order = new OrderBL(orderID, orderDate, Clothes, price, address, new PaymentMethodBL(paymentType), customer.GetUsername());

customer.AddOrderCustomer(order);

}

connection.Close();

}

}

}

}

public List <ClothesBL> GetListOfClothesInOrder(int orderID)

{

List<ClothesBL> Clothes = new List<ClothesBL>();

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM OrderItems WHERE OrderID = @OrderID";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@OrderID", orderID);

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int clothesID = Convert.ToInt32(reader["ClothesID"]);

string type = Convert.ToString(reader["Type"]);

string gender = Convert.ToString(reader["Gender"]);

string color = Convert.ToString(reader["Color"]);

int price = Convert.ToInt32(reader["Price"]);

int quantity = Convert.ToInt32(reader["Quantity"]);

ClothesBL cloth = new ClothesBL(clothesID, type, gender, color, price, quantity);

Clothes.Add(cloth);

}

connection.Close();

}

}

}

return Clothes;

}

/\*

public List <OrderBL> GetAllOrder()

{

return TotalOrders;

}

\*/

}

public class OrderFH:IOrderDL

{

public bool AddOrder(OrderBL order)

{

string path = UtilityClass.GetOrdersFilePath();

int orderId = GetAllOrdersCount()+1;

using (StreamWriter f = new StreamWriter(path, true))

{

if (f != null)

{

f.WriteLine(orderId+","+order.GetOrderDate() + "," + order.GetTotalPrice() + "," + order.GetDeliveryAddress() + "," + order.GetPaymentMethod().GetPaymentType() + "," + order.GetUsername());

f.Flush();

SaveOrderItems(orderId, order);

return true;

}

}

return false;

}

public void SaveOrderItems(int OrderId, OrderBL order)

{

string path = UtilityClass.GetOrderItemsFilePath();

using (StreamWriter f = new StreamWriter(path, true))

{

foreach(ClothesBL cloth in order.GetItems())

{

f.WriteLine(OrderId + "," + cloth.GetId() + "," + cloth.GetType() + "," + cloth.GetGender() + "," + cloth.GetColor() + "," + cloth.GetPrice() + "," + cloth.GetQuantity());

f.Flush();

}

}

}

public int GetAllOrdersCount()

{

int count = 0;

string path = UtilityClass.GetOrdersFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

count++;

}

}

}

}

return count;

}

public List<ClothesBL> GetListOfClothesInOrder(int orderID)

{

List<ClothesBL> Clothes = new List<ClothesBL>();

string path = UtilityClass.GetOrderItemsFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

int oID = Convert.ToInt32(splittedRecord[0]);

if (orderID == oID)

{

int clothesID = Convert.ToInt32(splittedRecord[1]);

string type = splittedRecord[2];

string gender = splittedRecord[3];

string color = splittedRecord[4];

int price = Convert.ToInt32(splittedRecord[5]);

int quantity = Convert.ToInt32(splittedRecord[6]);

ClothesBL cloth = new ClothesBL(clothesID, type, gender, color, price, quantity);

Clothes.Add(cloth);

}

}

}

}

}

return Clothes;

}

public void RetrieveOrdersOfCustomer(CustomerBL customer)

{

string path = UtilityClass.GetOrdersFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[5];

if (username == customer.GetUsername())

{

int orderId = Convert.ToInt32(splittedRecord[0]);

DateTime date = Convert.ToDateTime(splittedRecord[1]);

int price = Convert.ToInt32(splittedRecord[2]);

string address = splittedRecord[3];

string paymentType = splittedRecord[4];

List<ClothesBL> Clothes = GetListOfClothesInOrder(orderId);

OrderBL order = new OrderBL(orderId, date, Clothes, price, address, new PaymentMethodBL(paymentType), customer.GetUsername());

customer.AddOrderCustomer(order);

}

}

}

}

}

}

}

public class ReviewDB:IReviewDL

{

// private List<ReviewBL> AllReviews = new List<ReviewBL>();

/\*

public void AddReviews(ReviewBL rev)

{

AllReviews.Add(rev);

}

\*/

public void DeleteReview(ClothesBL c)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "Delete FROM Reviews WHERE ClothesID = @clothID";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@clothID", c.GetId());

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public bool AddReviews(ReviewBL rev, ClothesBL cloth)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Insert into Reviews (Rating, Comment, Username, ClothID) Values('{0}', '{1}', '{2}', {3})", rev.GetRating(), rev.GetComment(), rev.GetUsername(), cloth.GetId());

SqlCommand cmd = new SqlCommand(query, connection);

int rows = cmd.ExecuteNonQuery();

connection.Close();

if (rows > 0)

{

return true;

}

else

{

return false;

}

}

public void RetrieveReviews(ClothesBL c)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Reviews where clothid = @id";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("id", c.GetId());

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

int rating = Convert.ToInt32(reader["Rating"]);

string comment = Convert.ToString(reader["Comment"]);

string username = Convert.ToString(reader["Username"]);

ReviewBL rev = new ReviewBL(rating, comment, username);

c.AddReview(new ReviewBL(rev.GetRating(), rev.GetComment(), rev.GetUsername()));

}

connection.Close();

}

}

}

}

}

public class ReviewFH:IReviewDL

{

public bool AddReviews(ReviewBL rev, ClothesBL cloth)

{

string path = UtilityClass.GetReviewsFilePath();

using (StreamWriter f = new StreamWriter(path, true))

{

if (f != null)

{

f.WriteLine(rev.GetRating() + "," + rev.GetComment() + "," + rev.GetUsername() + "," + cloth.GetId());

f.Flush();

return true;

}

}

return false;

}

public void RetrieveReviews(ClothesBL c)

{

string path = UtilityClass.GetReviewsFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

int clothId = Convert.ToInt32(splittedRecord[3]);

if (clothId == c.GetId())

{

int rating = Convert.ToInt32(splittedRecord[0]);

string comment = splittedRecord[1];

string username = splittedRecord[2];

ReviewBL rev = new ReviewBL(rating, comment, username);

c.AddReview(new ReviewBL(rev.GetRating(), rev.GetComment(), rev.GetUsername()));

}

}

}

}

}

}

public void DeleteReview(ClothesBL c)

{

string path = UtilityClass.GetReviewsFilePath();

if (File.Exists(path))

{

List<string> remainingLines = new List<string>();

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

int currentClothId = Convert.ToInt32(splittedRecord[3]);

// If the clothId matches the specified clothId, skip adding the review

if (currentClothId != c.GetId())

{

remainingLines.Add(record);

}

}

}

}

// Write the remaining reviews back to the file

File.WriteAllLines(path, remainingLines);

}

}

}

public class UserDB: IUserDL

{

/\*

public void AddUser(UserBL u)

{

Users.Add(u);

}

\*/

public bool AddUser(UserBL user)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Insert into Users (Username, Password, Email, FirstName, LastName, PhoneNumber, Role) Values('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}')", user.GetUsername(), user.GetPassword(), user.GetEmail(), user.GetFirstName(), user.GetLastName(), user.GetPhoneNumber(), user.GetRole());

SqlCommand cmd = new SqlCommand(query, connection);

int rows = cmd.ExecuteNonQuery();

connection.Close();

if (rows > 0)

{

return true;

}

else

{

return false;

}

}

public void UpdateProfile(UserBL user)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("UPDATE Users SET Password = '{0}', Email = '{1}', FirstName = '{2}', LastName = '{3}', PhoneNumber = '{4}' WHERE username = '{5}'", user.GetPassword(), user.GetEmail(), user.GetFirstName(), user.GetLastName(), user.GetPhoneNumber(), user.GetUsername());

SqlCommand cmd = new SqlCommand(query, connection);

int rows = cmd.ExecuteNonQuery();

connection.Close();

}

public bool IsUserExists(UserBL cUser)

{

string connectiongString = UtilityClass.GetConnectionString();

SqlConnection connection = new SqlConnection(connectiongString);

connection.Open();

string query = string.Format("Select count(\*) from Users where Username = '{0}'", cUser.GetUsername());

SqlCommand cmd = new SqlCommand(query, connection);

int count = (int)cmd.ExecuteScalar();

connection.Close();

return count > 0;

}

/\*

public bool IsUserExists(UserBL cUser)

{

foreach (UserBL stored in Users)

{

if (cUser.GetUsername() == stored.GetUsername())

{

return true;

}

}

return false;

}

\*/

/\*

public UserBL FindUser(UserBL u)

{

foreach (UserBL stored in Users)

{

if(u.GetUsername() == stored.GetUsername() && u.GetPassword() == stored.GetPassword())

{ return stored; }

}

return null;

}

\*/

public UserBL FindUser(UserBL u)

{

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Users WHERE Username = @Username AND Password = @Password";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@Username", u.GetUsername());

cmd.Parameters.AddWithValue("@Password", u.GetPassword());

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

string username = Convert.ToString(reader["Username"]);

string password = Convert.ToString(reader["Password"]);

string email = Convert.ToString(reader["Email"]);

string fname = Convert.ToString(reader["FirstName"]);

string lname = Convert.ToString(reader["LastName"]);

string phone = Convert.ToString(reader["PhoneNumber"]);

string role = Convert.ToString(reader["Role"]);

connection.Close();

if(role == "customer")

{

CustomerBL customer = new CustomerBL(username, password, email, fname, lname, phone, role);

return customer;

}

else

{

EmployeeBL employee = new EmployeeBL(username, password, email, fname, lname, phone, role);

return employee;

}

}

}

}

}

return null; // User not found

}

public List<CustomerBL> GetAllCustomers()

{

List<CustomerBL> customers = new List<CustomerBL>();

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Users WHERE Role = 'customer'";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

string username = Convert.ToString(reader["Username"]);

string password = Convert.ToString(reader["Password"]);

string email = Convert.ToString(reader["Email"]);

string fname = Convert.ToString(reader["FirstName"]);

string lname = Convert.ToString(reader["LastName"]);

string phone = Convert.ToString(reader["PhoneNumber"]);

string role = Convert.ToString(reader["Role"]);

CustomerBL customer = new CustomerBL(username, password, email, fname, lname, phone, role);

customers.Add(customer);

}

connection.Close();

}

}

}

return customers;

}

public bool CheckCustomersCount()

{

int count = 0;

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Users WHERE Role = 'customer'";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

count = (int)cmd.ExecuteScalar();

}

connection.Close() ;

}

return count>0;

}

public CustomerBL FindCustomerByUsername(string username)

{

CustomerBL customer = null;

string connectionString = UtilityClass.GetConnectionString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Users WHERE Username = @Username AND Role = 'customer'";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@Username", username);

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

string foundUsername = reader["Username"].ToString();

string password = reader["Password"].ToString();

string email = reader["Email"].ToString();

string fname = reader["FirstName"].ToString();

string lname = reader["LastName"].ToString();

string phone = reader["PhoneNumber"].ToString();

string role = reader["Role"].ToString();

customer = new CustomerBL(foundUsername, password, email, fname, lname, phone, role);

}

connection.Close();

}

}

}

return customer;

}

public class UserFH:IUserDL

{

public bool AddUser(UserBL user)

{

string path = UtilityClass.GetUserFilePath();

using (StreamWriter f = new StreamWriter(path,true))

{

if (f != null)

{

f.WriteLine(user.GetUsername() + "," + user.GetPassword() + "," + user.GetEmail() + "," + user.GetFirstName() + "," + user.GetLastName() + "," + user.GetPhoneNumber() + "," + user.GetRole());

f.Flush();

return true;

}

}

return false;

}

public UserBL FindUser(UserBL u)

{

string path = UtilityClass.GetUserFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

string password = splittedRecord[1];

string email = splittedRecord[2];

string fname = splittedRecord[3];

string lname = splittedRecord[4];

string phone = splittedRecord[5];

string role = splittedRecord[6];

if (role == "customer")

{

if (username == u.GetUsername() && password == u.GetPassword())

{

CustomerBL customer = new CustomerBL(username, password, email, fname, lname, phone, role);

return customer;

}

}

else

{

if (username == u.GetUsername() && password == u.GetPassword())

{

EmployeeBL employee = new EmployeeBL(username, password, email, fname, lname, phone, role);

return employee;

}

}

}

}

}

}

return null; // User not found

}

public List<CustomerBL> GetAllCustomers()

{

List <CustomerBL> customers = new List<CustomerBL>();

string path = UtilityClass.GetUserFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

string password = splittedRecord[1];

string email = splittedRecord[2];

string fname = splittedRecord[3];

string lname = splittedRecord[4];

string phone = splittedRecord[5];

string role = splittedRecord[6];

if (role == "customer")

{

CustomerBL customer = new CustomerBL(username, password, email, fname, lname, phone, role);

customers.Add(customer);

}

}

}

}

}

return customers;

}

public bool CheckCustomersCount()

{

int count = 0;

string path = UtilityClass.GetUserFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

string[] splittedRecord = record.Split(',');

string role = splittedRecord[6];

if (role == "customer")

{

count++;

}

}

}

}

return count > 0;

}

public bool IsUserExists(UserBL cUser)

{

string path = UtilityClass.GetUserFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

if (username == cUser.GetUsername())

{

return true;

}

}

}

}

}

return false;

}

public CustomerBL FindCustomerByUsername(string username)

{

string path = UtilityClass.GetUserFilePath();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string name = splittedRecord[0];

string role = splittedRecord[6];

if (name == username && role == "customer")

{

string password = splittedRecord[1];

string email = splittedRecord[2];

string fname = splittedRecord[3];

string lname = splittedRecord[4];

string phone = splittedRecord[5];

return new CustomerBL(name, password, email, fname, lname, phone, role);

}

}

}

}

}

return null;

}

public void UpdateProfile(UserBL user)

{

string path = UtilityClass.GetUserFilePath();

List <UserBL> AllUsers = new List <UserBL>();

if (File.Exists(path))

{

using (StreamReader f = new StreamReader(path))

{

string record;

while ((record = f.ReadLine()) != null)

{

if (!string.IsNullOrEmpty(record))

{

string[] splittedRecord = record.Split(',');

string username = splittedRecord[0];

string password = splittedRecord[1];

string email = splittedRecord[2];

string fname = splittedRecord[3];

string lname = splittedRecord[4];

string phone = splittedRecord[5];

string role = splittedRecord[6];

if (role == "customer")

{

CustomerBL customer = new CustomerBL(username, password, email, fname, lname, phone, role);

AllUsers.Add(customer);

}

else

{

EmployeeBL employee = new EmployeeBL(username, password, email, fname, lname, phone, role);

AllUsers.Add(employee);

}

}

}

}

}

foreach(UserBL stored in AllUsers)

{

if(stored.GetUsername() == user.GetUsername())

{

stored.UpdateProfile(user.GetPassword(), user.GetEmail(), user.GetFirstName(), user.GetLastName(), user.GetPhoneNumber());

}

}

File.WriteAllText(path, "");

foreach(UserBL stored in AllUsers)

{

AddUser(stored);

}

}

}