# 

**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

Sir. Laeeq Khan Niazi

# Course:

CSC-103 Object Oriented Programming

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

Table of Contents

[ Fashion Is U 3](#_Toc164965661)

[ Users Of Application 3](#_Toc164965662)

[ Features Lists 3](#_Toc164965663)

[ WireFrames (Winforms) 5](#_Toc164965664)

[ Class Diagram (CRC Model) 17](#_Toc164965665)

[ Complete Code 18](#_Toc164965666)

# 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.

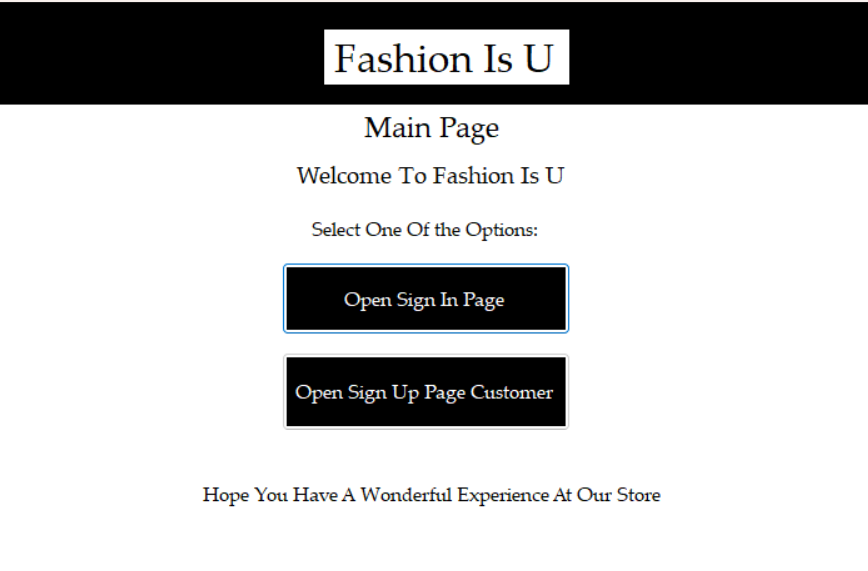
# Features Lists

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

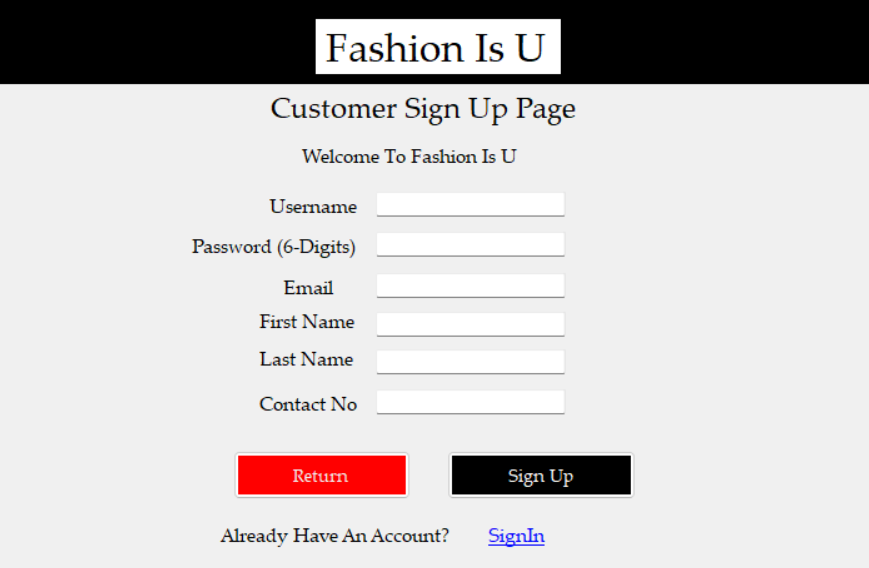
# WireFrames (Winforms)



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



**Figure 2: Sign In Menu**



**Figure 3: Customer Sign Up Menu**



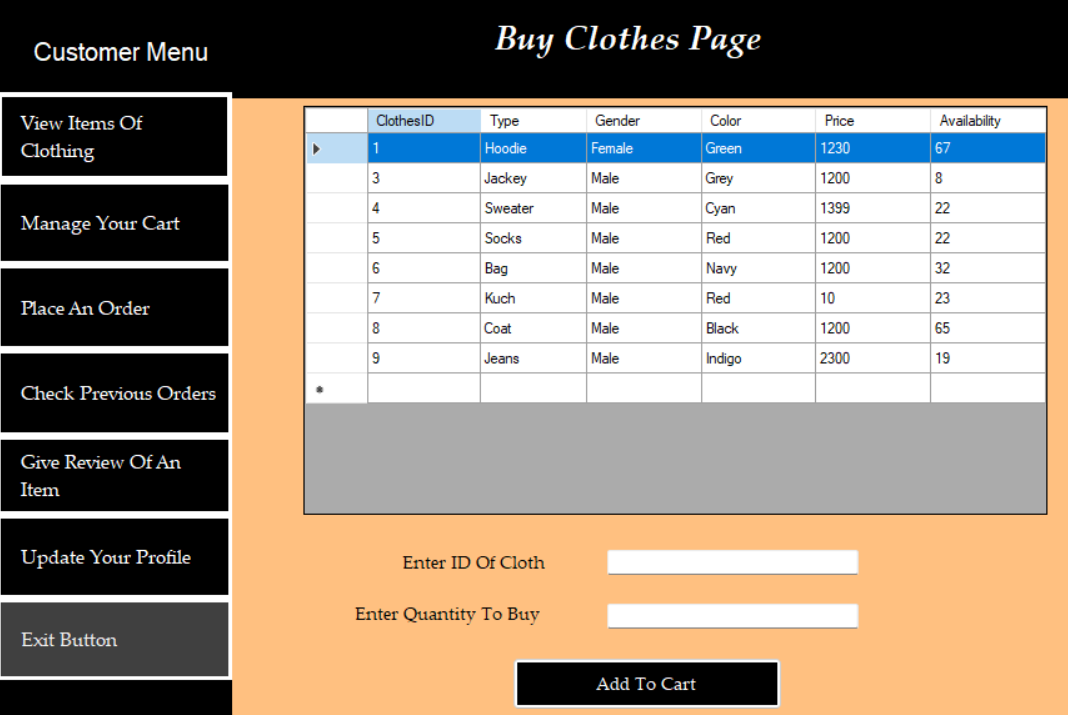
**Figure 4: Customer Menu**



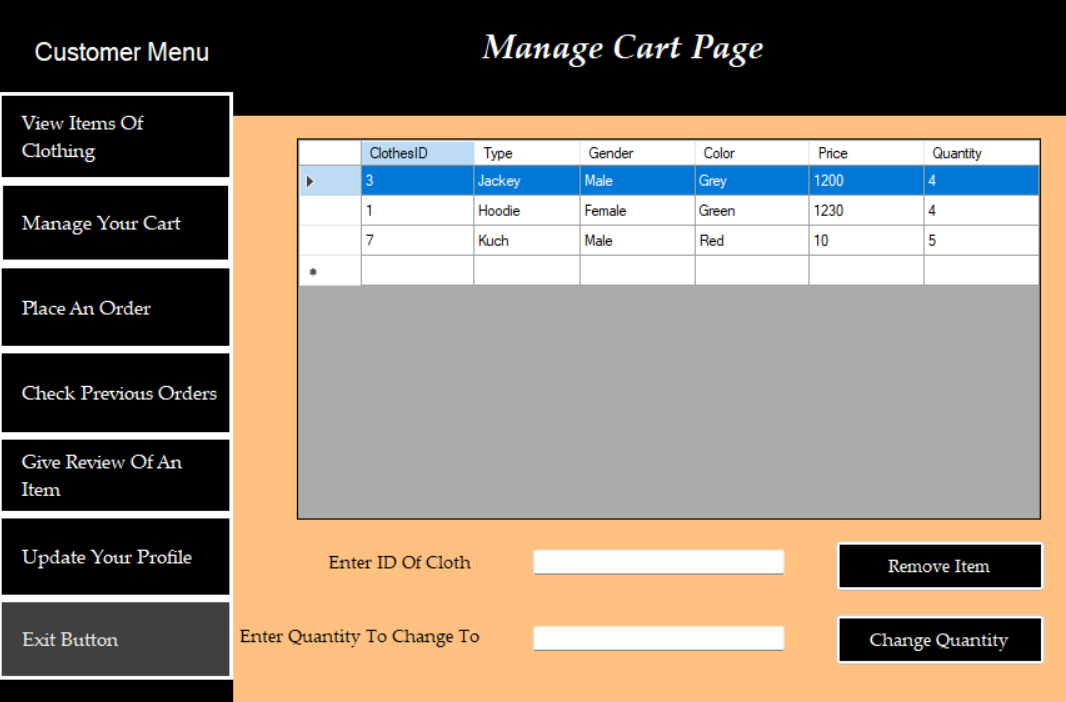
**Figure 5: Employee Menu**



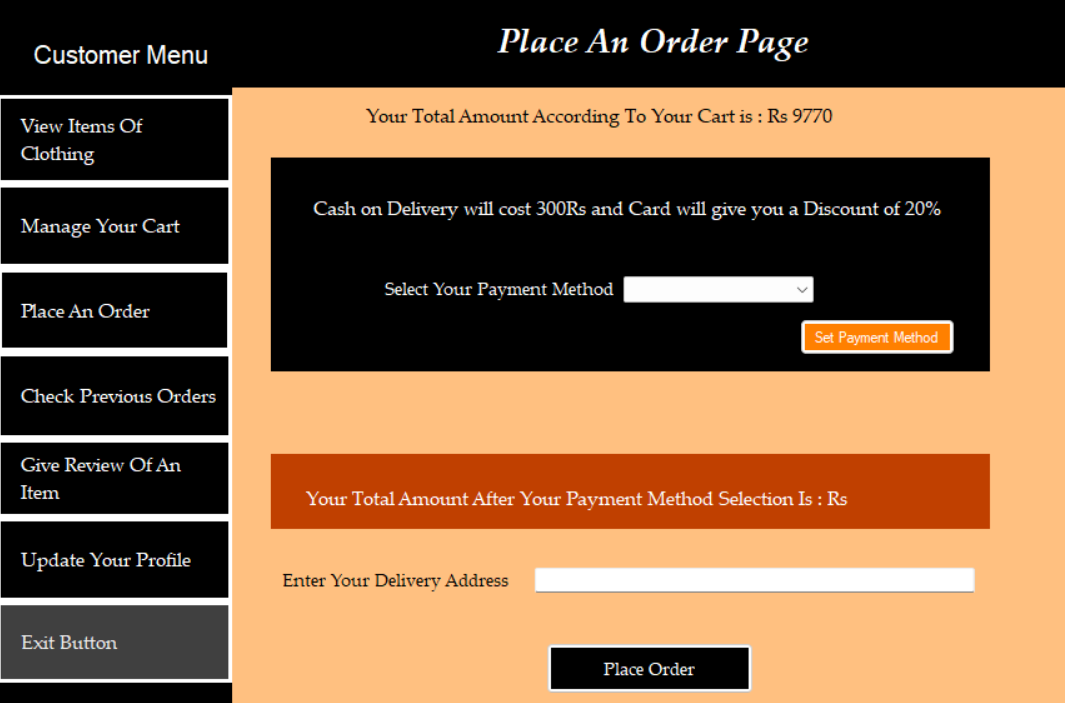
**Figure 6: Employee Menu**



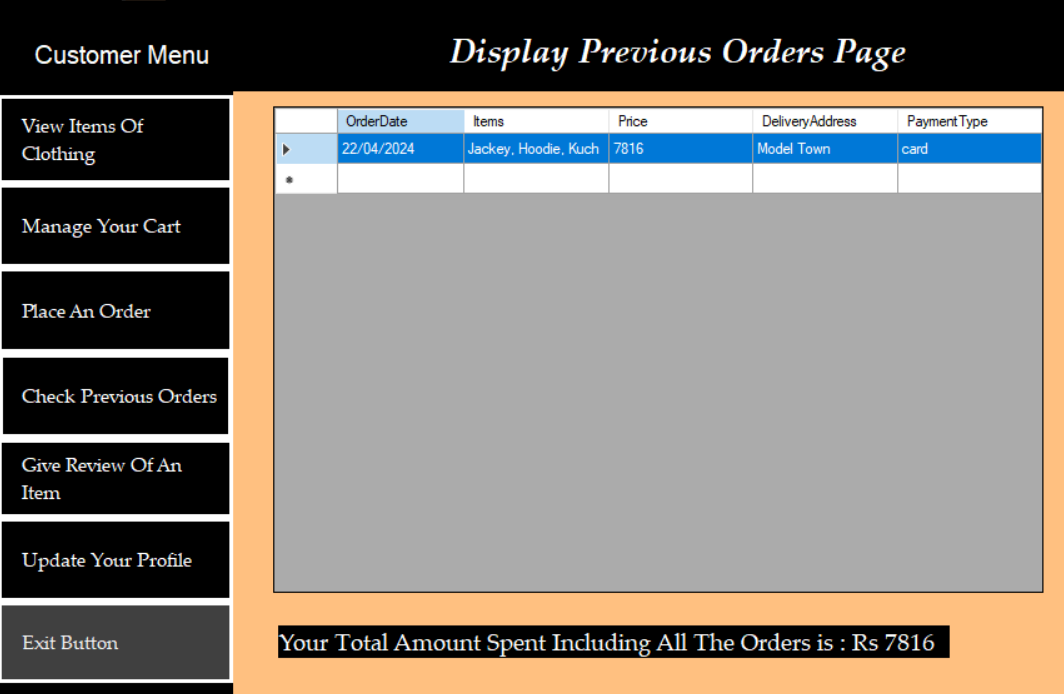
**Figure 7: Buy Clothes Page**



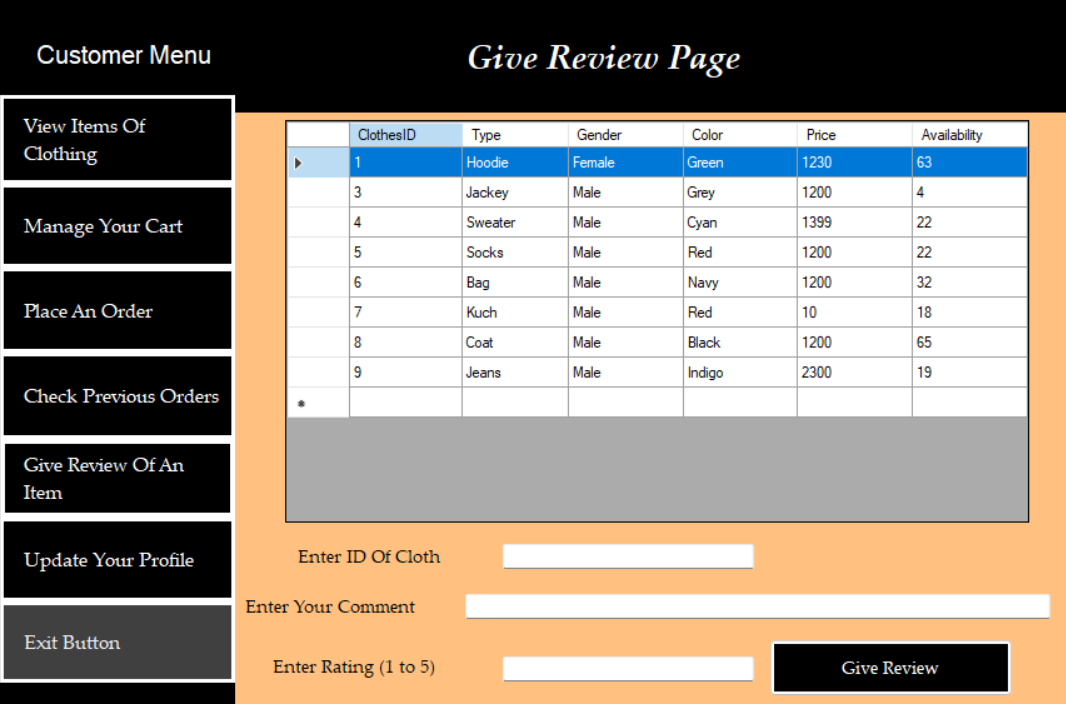
**Figure 8: Manage Cart Page**



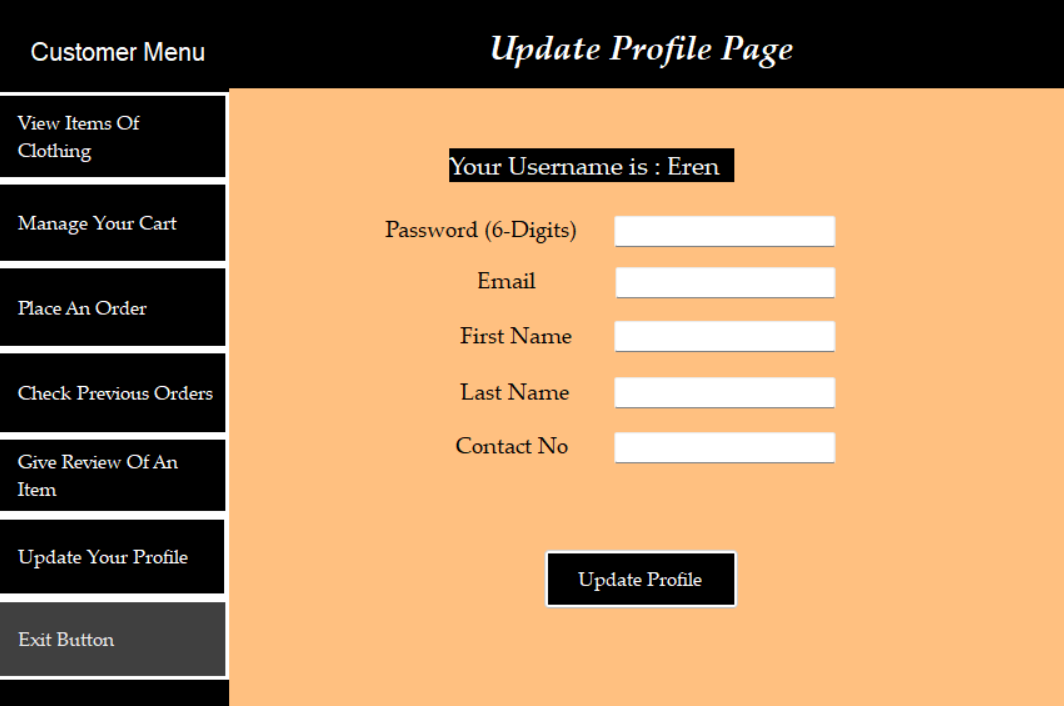
**Figure 9: Place Order Page**



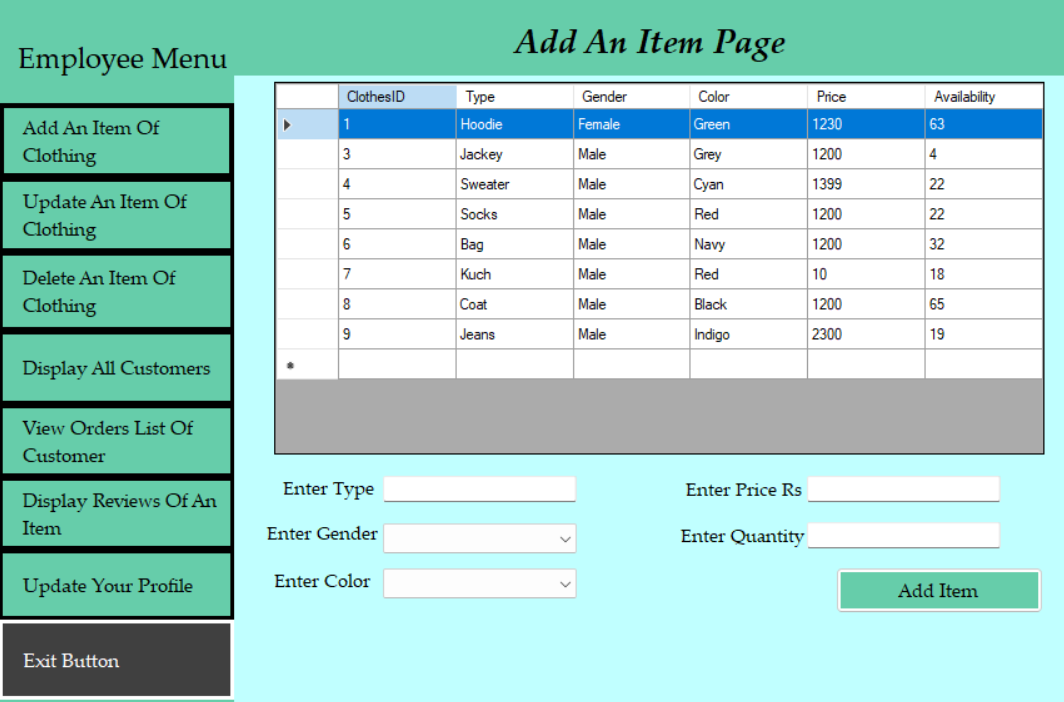
**Figure 10: Display Orders Page**

******

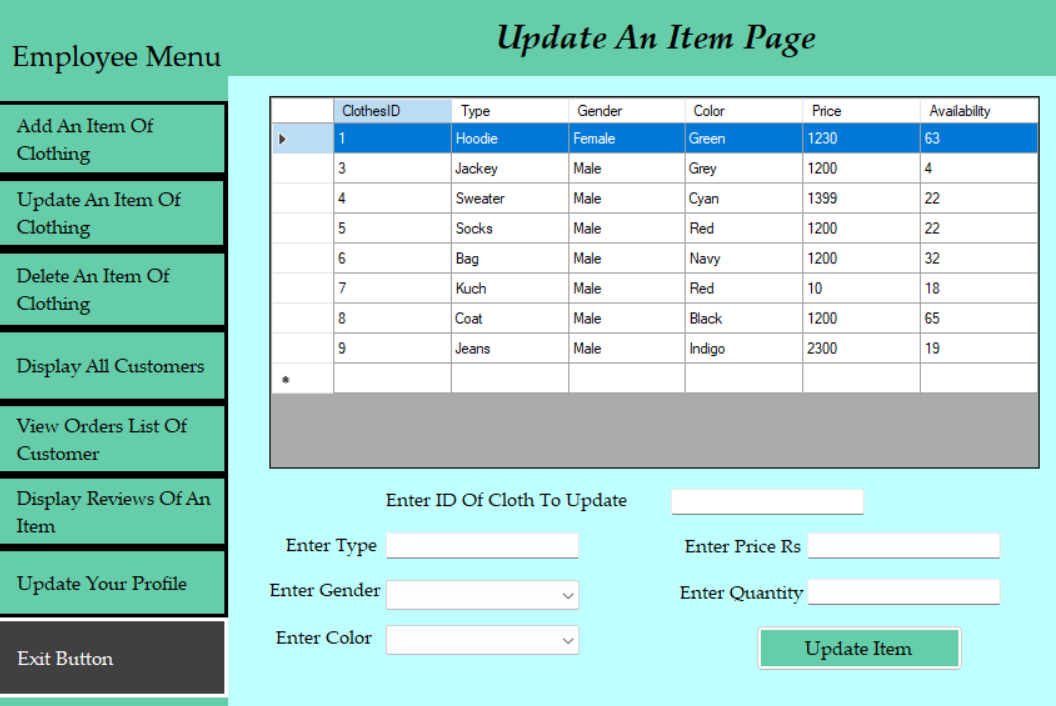
**Figure 11: Give Review Page**



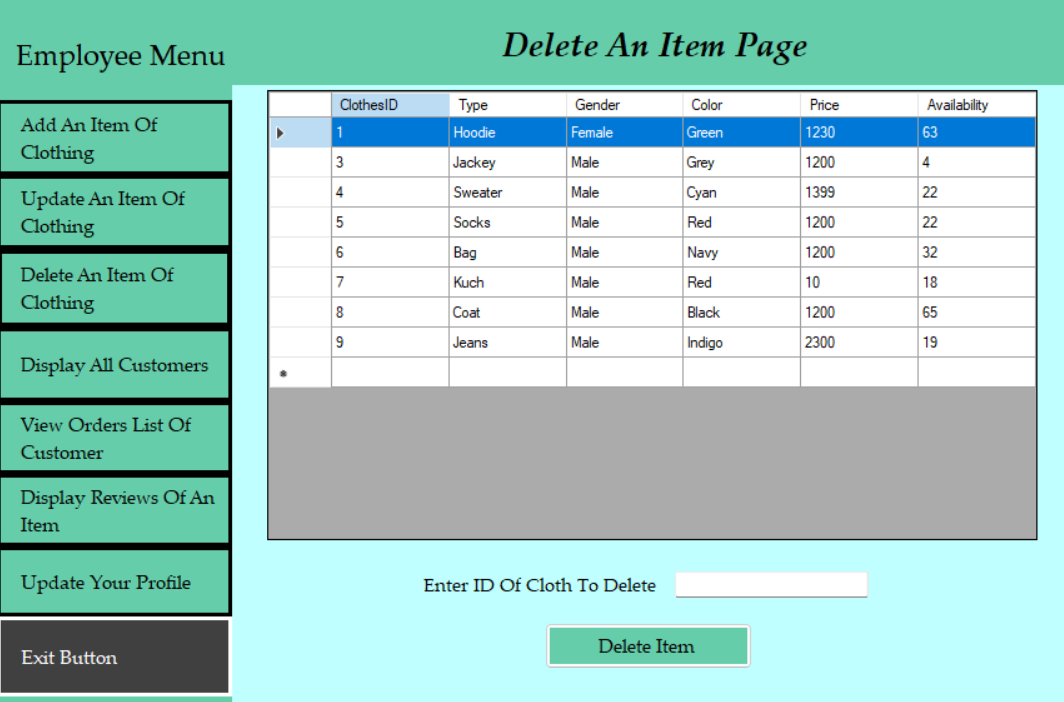
**Figure 12: Update Profile Page (Customer)**



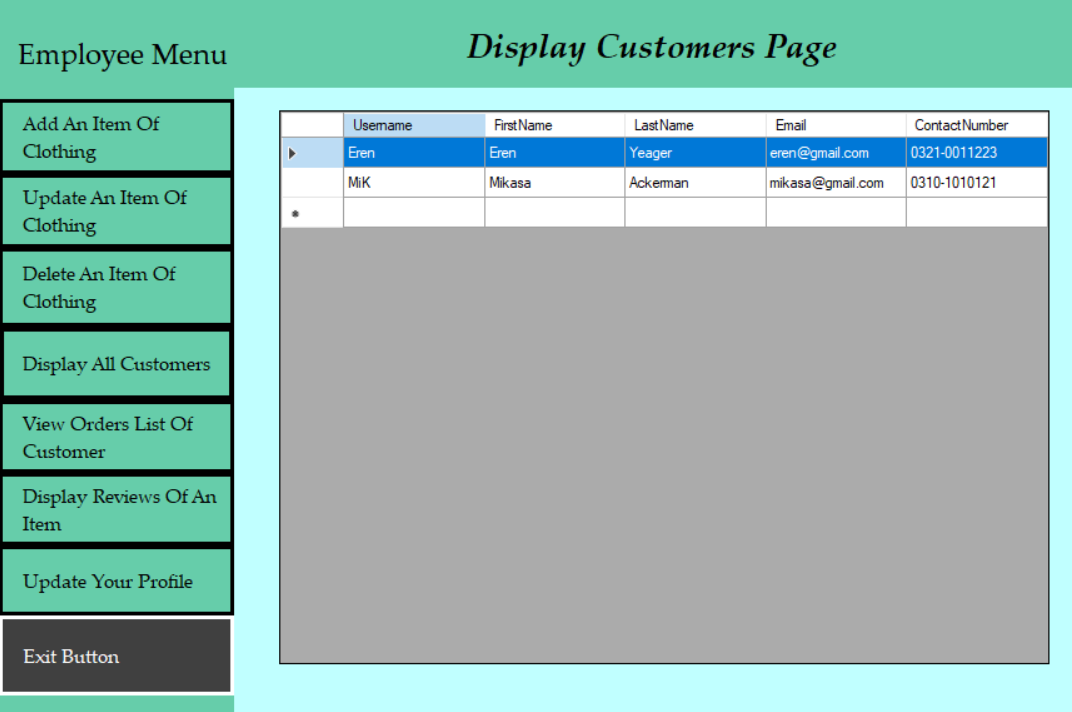
**Figure 13: Add an Item Page**



**Figure 14: Update an Item Page**



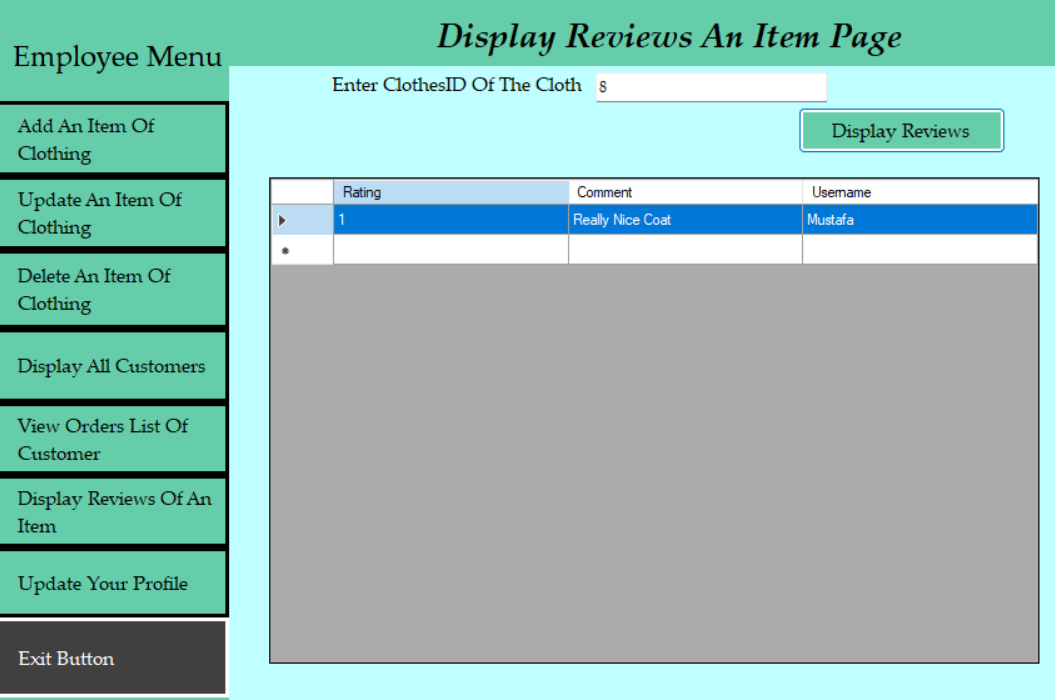
**Figure 15: Delete an Item Page**



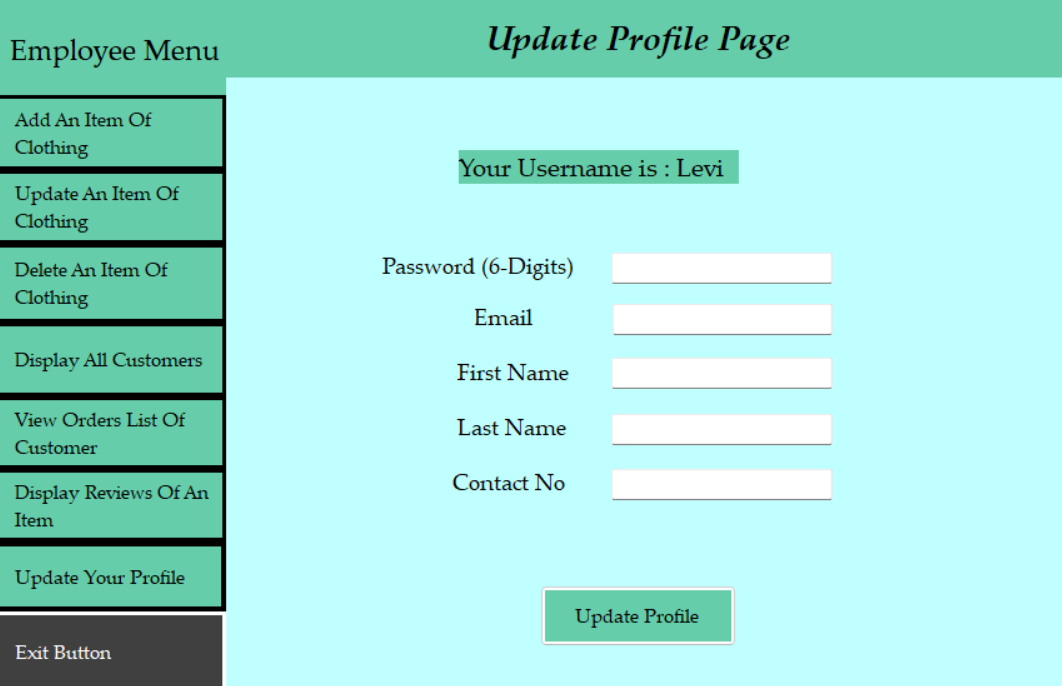
**Figure 16: Display All Customers Page**



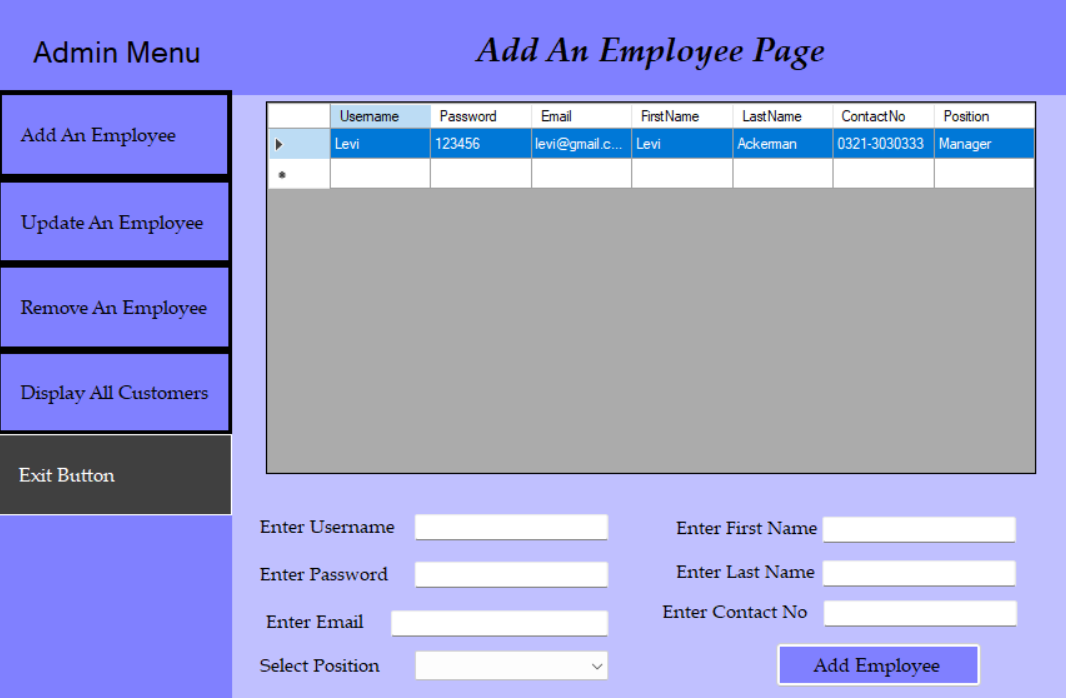
**Figure 17: Display Orders of Customer Page**



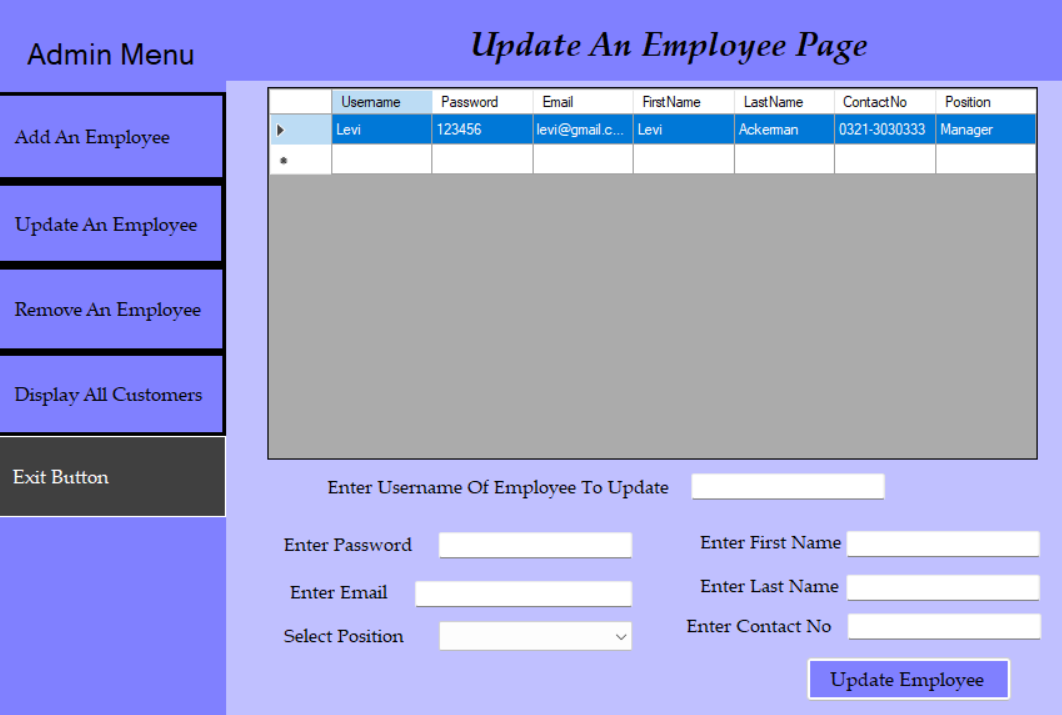
**Figure 18: Display Reviews Page**



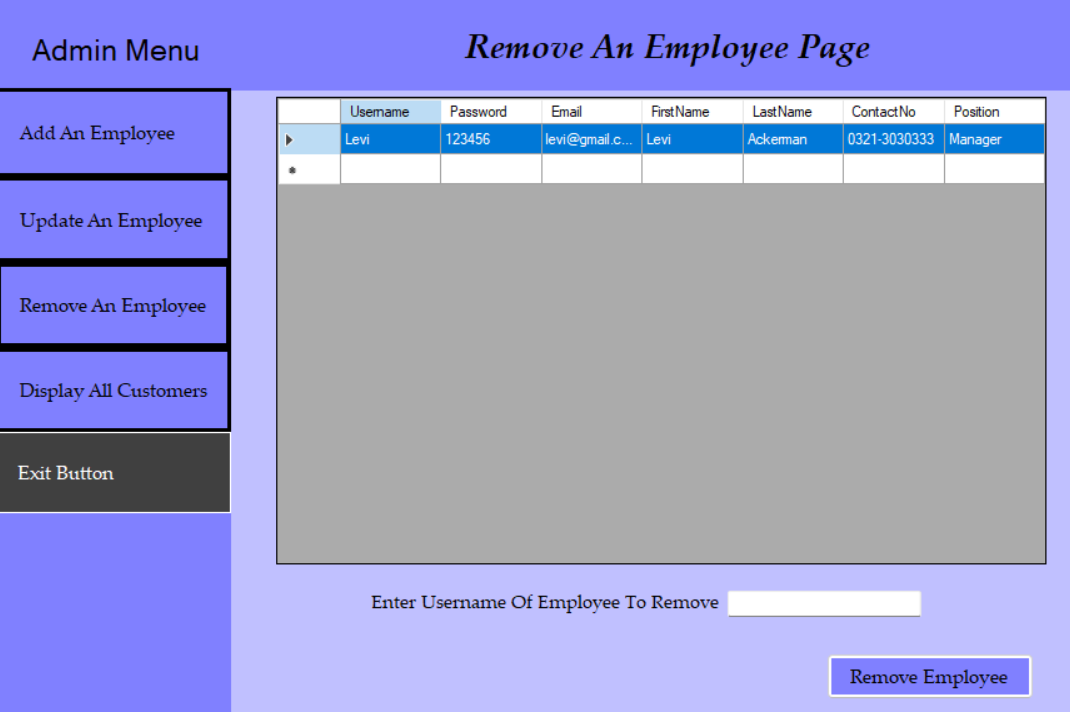
**Figure 19: Update Profile Page (Employee)**



**Figure 20: Add an Employee Page**

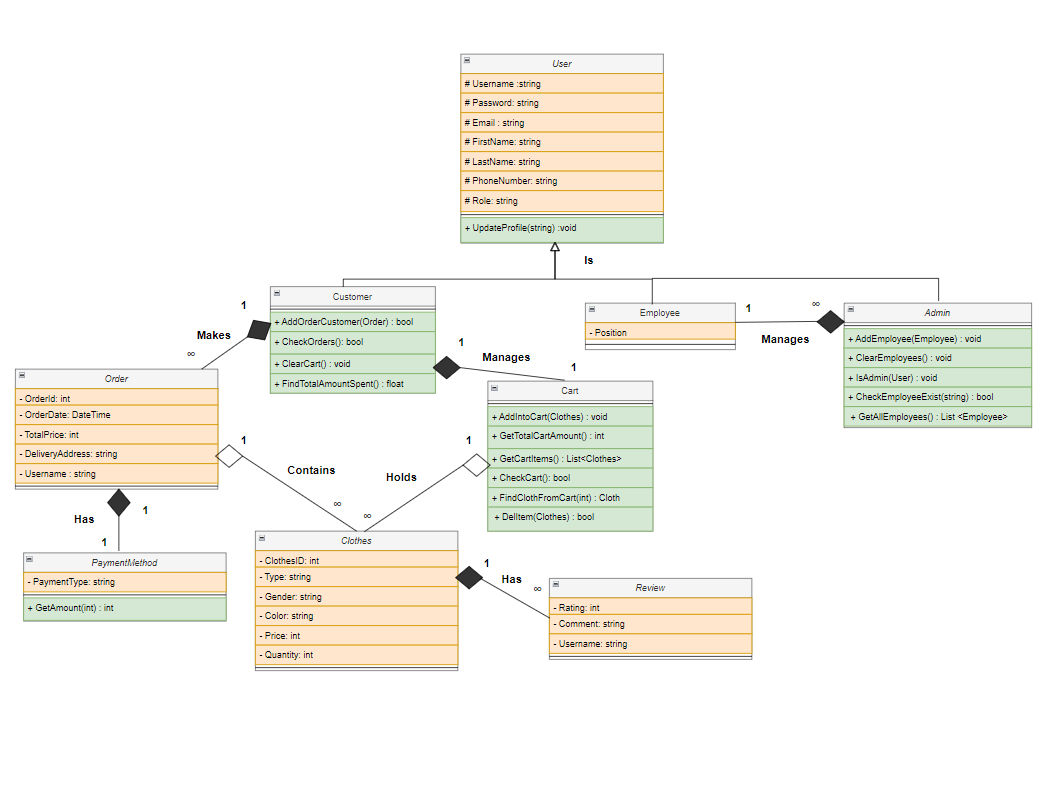


**Figure 21: Update an Employee Page**



**Figure 22: Remove an Employee Page**

# Class Diagram (CRC Model)



# Complete Code

## // ClothesBL

public class ClothesBL

{

// attributes of clothes

private int ClothesID;

private string Type;

private string Gender;

private string Color;

private int Price;

private int Quantity;

private List<ReviewBL> Reviews;

// Parameterized Constructors

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;

}

// Getter and Setters

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) // Compares the quantity to tell if an item is available to buy

{

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

{ return true; }

return false;

}

public void DropQuantity(int quantity) // Reduces the quantity of Cloth after being bought

{

Quantity -= quantity;

}

public void AddQuantity(int quantity) // Increases the quantity of Cloth

{ Quantity += quantity; }

public void AddReview(ReviewBL rev) // Adds the review In the List Of Reviews

{

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

}

public List<ReviewBL> GetReviews() // Gives the list of reviews

{

return Reviews;

}

public void SetReviews(List<ReviewBL> reviews)

{

this.Reviews = reviews;

}

public void ClearReviews() //Clears the list of reviews

{ Reviews.Clear(); }

public void UpdateCloth(ClothesBL cloth) // Updates the details of the cloth

{

SetGender(cloth.GetGender());

SetColor(cloth.GetColor());

SetType(cloth.GetType());

SetPrice(cloth.GetPrice());

SetQuantity(cloth.GetQuantity());

}

## // ClothesDL (Database)

public class ClothesDB:IClothesDL

{

// this is for the singleton functionality

private static ClothesDB ClothesDBInstance; // instance of clothes db

private string ConnectionString = "";

private ClothesDB(string ConnectionString)

{

this.ConnectionString = ConnectionString;

}

public static ClothesDB GetClothesDB(string connectionString) // returns the instance

{

if (ClothesDBInstance == null)

{

ClothesDBInstance = new ClothesDB(connectionString);

}

return ClothesDBInstance;

}

// adds an item of clothing in the database

public bool AddClothes(ClothesBL c)

{

SqlConnection connection = new SqlConnection(ConnectionString);

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() // checks the count of cloths in the database

{

SqlConnection connection = new SqlConnection(ConnectionString);

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) // finds an item of clothing from the database from their id

{

ClothesBL cloth = null;

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) // checks if an items exists or not

{

bool clothExists = false;

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) // checks if an items already exist including their quantity

{

bool clothExists = false;

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) //changes the quantity of an item

{

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) // updates an item entirely

{

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() // returns the list of all clothes

{

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

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 bool DeleteCloth(ClothesBL c) // deletes an item of clothing

{

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;

}

}

}

}

## //ClothesDL (File Handling)

public class ClothesFH : IClothesDL

{

// this is for the single functionality

private static int ClothesId = 0; // this is for auto increment functionality in file

private static ClothesFH ClothesFHInstance; // instance of clothes

private string FilePath = "";

private ClothesFH(string FilePath)

{

this.FilePath = FilePath;

}

public static ClothesFH GetClothesFH(string filePath) // get the instance

{

if (ClothesFHInstance == null)

{

ClothesFHInstance = new ClothesFH(filePath);

}

return ClothesFHInstance;

}

public bool AddClothes(ClothesBL c) // adds an item of clothing

{

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

using (StreamWriter f = new StreamWriter(FilePath, 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() // checks the amount of clothes or its existence

{

int count = 0;

if (File.Exists(FilePath))

{

using (StreamReader f = new StreamReader(FilePath))

{

string record;

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

{

if (!string.IsNullOrEmpty(record))

{

count++;

}

}

}

}

return count > 0;

}

public List<ClothesBL> GetAllClothes() // gives the lst of all clothes

{

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

if (File.Exists(FilePath))

{

using (StreamReader f = new StreamReader(FilePath))

{

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) // find a cloth by their id

{

if (File.Exists(FilePath))

{

using (StreamReader f = new StreamReader(FilePath))

{

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) // checks the existence of a cloth in database

{

if (File.Exists(FilePath))

{

using (StreamReader f = new StreamReader(FilePath))

{

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) // cheks the exitence inclduing their quantity

{

if (File.Exists(FilePath))

{

using (StreamReader f = new StreamReader(FilePath))

{

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) // changes the quantity of an items

{

List<ClothesBL> AllClothes = GetAllClothes();

foreach (ClothesBL stored in AllClothes)

{

if (stored.GetId() == id)

{

stored.SetQuantity(quantity);

}

}

File.WriteAllText(FilePath, "");

foreach (ClothesBL stored in AllClothes)

{

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

{

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

f.Flush();

}

}

}

public void UpdateCloth(ClothesBL cloth) // updates the quantity of an item

{

List<ClothesBL> AllClothes = GetAllClothes();

foreach (ClothesBL stored in AllClothes)

{

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

{

stored.UpdateCloth(cloth);

}

}

File.WriteAllText(FilePath, "");

foreach (ClothesBL stored in AllClothes)

{

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

{

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

f.Flush();

}

}

}

public bool DeleteCloth(ClothesBL c) // deletes an item of clothing

{

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(FilePath, "");

foreach (ClothesBL stored in AllClothes)

{

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

{

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

f.Flush();

}

}

}

return clothDeleted;

}

}

## // ClothesUI

internal class ClothesUI

{

public static void DisplayAllClothes(List <ClothesBL> AllClothes) // display all the list of clothes

{

Console.WriteLine();

Console.WriteLine("--------------------------------DISPLAY ALL CLOTHES---------------------------------------------");

Console.WriteLine("------------------------------------------------------------------------------------------------");

Console.WriteLine("|{0,-15}|{1,-15}|{2,-15}|{3,-15}|{4,-15}|{5,-15}|", "ClothesID","Type", "Gender", "Color", "Price", "Availability");

Console.WriteLine("------------------------------------------------------------------------------------------------");

foreach (ClothesBL c in AllClothes)

{

Console.WriteLine("|{0,-15}|{1,-15}|{2,-15}|{3,-15}|{4,-15}|{5,-15}|", c.GetId(), c.GetType(), c.GetGender(), c.GetColor(), "Rs"+c.GetPrice(), c.GetQuantity());

}

Console.WriteLine("------------------------------------------------------------------------------------------------");

}

public static void NoClothesFound() // displays that clothes are not found

{

Console.WriteLine("No Clothes Found ....");

Thread.Sleep(500);

}

public static void IncorrectId() // shows that id is incorrect

{

Console.WriteLine("No Cloth found against that ID ....");

Thread.Sleep(500);

}

public static ClothesBL TakeInputForClothes() // takes the input for adding a cloth

{

int Id = 0;

int price =0, available = 0;

string temp;

Console.WriteLine();

Console.WriteLine("------------------------------ADD AN ITEM OF CLOTHING----------------------------");

Console.WriteLine();

Console.WriteLine("Enter clothing details:");

Console.WriteLine();

Console.Write("Enter the Type: ");

string type = Console.ReadLine();

type = ConsoleValidationUI.ValidateWordsWithInt(type);

Console.Write("Enter the Gender: ");

string gender = Console.ReadLine();

gender = ConsoleValidationUI.ValidateGender(gender);

Console.Write("Enter the Color: ");

string color = Console.ReadLine();

color = ConsoleValidationUI.ValidateWordsWithInt(color);

Console.Write("Enter the Price: Rs ");

temp = Console.ReadLine();

price = ConsoleValidationUI.ValidateInt(temp, price);

Console.Write("Enter the Quantity: ");

temp = Console.ReadLine();

available = ConsoleValidationUI.ValidateInt(temp, available);

ClothesBL c = new ClothesBL(type, gender, color, price, available);

return c;

}

public static ClothesBL TakeInputForUpdateClothe(ClothesBL cloth) // takes the input for updating cloth

{

int price = 0, available = 0;

string temp;

Console.WriteLine();

Console.WriteLine("------------------------------UPDATE AN ITEM OF CLOTHING----------------------------");

Console.WriteLine();

Console.WriteLine("Enter clothing details:");

Console.WriteLine();

Console.Write("Enter the Type: ");

string type = Console.ReadLine();

type = ConsoleValidationUI.ValidateWordsWithInt(type);

Console.Write("Enter the Gender: ");

string gender = Console.ReadLine();

gender = ConsoleValidationUI.ValidateGender(gender);

Console.Write("Enter the Color: ");

string color = Console.ReadLine();

color = ConsoleValidationUI.ValidateWordsWithInt(color);

Console.Write("Enter the Price: Rs ");

temp = Console.ReadLine();

price= ConsoleValidationUI.ValidateInt(temp, price);

Console.Write("Enter the Quantity: ");

temp = Console.ReadLine();

available = ConsoleValidationUI.ValidateInt(temp, available);

return new ClothesBL(type, gender, color, price, available);

}

public static int TakeId() // takes the id of cloth

{

int id = 0;

Console.WriteLine();

Console.Write("Enter the ID of Clothing : ");

string temp = Console.ReadLine() ;

id = ConsoleValidationUI.ValidateInt(temp, id);

return id;

}

public static int TakeQuantity() // takes the quantity of cloth

{

int quantity = 0;

Console.Write("Enter the Quantity: ");

string temp = Console.ReadLine();

quantity = ConsoleValidationUI.ValidateInt(temp, quantity);

return quantity;

}

public static void NotPossible() // shows that it is not possible

{

Console.WriteLine("Not Possible....");

Thread.Sleep(500);

}

public static void ClothUpdatedSuccessfully() // show that cloth is updated successfully

{

Console.WriteLine("Cloth Updated Successfully....");

Thread.Sleep(500);

}

public static void ClothDeletedSuccessfully() // shows that cloth is deleted successfully

{

Console.WriteLine("Cloth Deleted Successfully....");

Thread.Sleep(500);

}

public static void ClothAlreadyExist() // shows that cloth already exist

{

Console.WriteLine("Cloth Already Exist....");

Thread.Sleep(500);

}

}