**The Twirling Forks**



Session: 2022 – 2026

**Submitted by:**

Ayesha Khalid 2022-CS-153

**Supervised by:**

Sir Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[**Introduction:** 2](#_Toc141444069)

[**Overview:** 2](#_Toc141444070)

[**Objective:** 2](#_Toc141444071)

[**Contribution in computer science field:** 3](#_Toc141444072)

[**Output:** 3](#_Toc141444073)

[**Users of Application** 3](#_Toc141444074)

[**Functional Requirements:** 3](#_Toc141444075)

[**Object Oriented Programming (OOP) Concept:** 5](#_Toc141444076)

[ **Inheritance:** 5](#_Toc141444077)

[**Polymorphism** 5](#_Toc141444078)

[**Association:** 6](#_Toc141444079)

[**Comparison with Procedural Programming:** 6](#_Toc141444080)

[**Design Pattern Implementation:** 6](#_Toc141444081)

[**Code:** 7](#_Toc141444082)

[**Class Diagram:** 8](#_Toc141444083)

[**Business Layer (BL):** 9](#_Toc141444084)

[The BL pattern focuses on encapsulating the business rules and logic of the application**.** The BLlayer focus on security of data. It gives class the responsibility to protect its own data. My project also focus on separate Business layer to give security to data the security to data s given by access modifiers i.e. protected for parent class and public for child class and using getter and setter. 9](#_Toc141444085)

[**Data Layer (DL):** 16](#_Toc141444086)

[} 18](#_Toc141444087)

[**User Interface:** 25](#_Toc141444088)

[**Key Classes:** 45](#_Toc141444089)

[**Wireframes:** 45](#_Toc141444090)

[ **Main Screen:** 45](#_Toc141444091)

[ **Role Screen** 45](#_Toc141444092)

[ **Admin Menu:** 46](#_Toc141444093)

[ **Add Noodles:** 46](#_Toc141444094)

[ **View,Update,Delete and discounts:** 47](#_Toc141444095)

[ **View Sorted Data:** 47](#_Toc141444096)

[ **View Feedbacks:** 47](#_Toc141444097)

[ **View Customer’s Information:** 49](#_Toc141444098)

[ 49](#_Toc141444099)

[ **Customer’s Menu:** 49](#_Toc141444100)

[ **Add Customer:** 50](#_Toc141444101)

[ **View Services:** 50](#_Toc141444102)

[ **Noodles Menu:** 51](#_Toc141444103)

[51](#_Toc141444104)

[ **Expensive to cheapest ones:** 51](#_Toc141444105)

[ **Discounts:** 52](#_Toc141444106)

[ **Add To Cart:** 52](#_Toc141444107)

[ **View Bill:** 53](#_Toc141444108)

[ **Give Feedback:** 53](#_Toc141444109)

[**Resources:** 54](#_Toc141444110)

[54](#_Toc141444111)

[**Conclusion:** 54](#_Toc141444112)

[**Achievements:** 54](#_Toc141444113)

[**Challenges:** 55](#_Toc141444114)

# 

# 

# **Introduction:**

## **Overview:**

The Twirling Forks is a software application designed to streamline and enhance the operations of a noodles-based restaurant or food establishment. It provides a comprehensive platform to manage various aspects of the business, including order processing, customer information etc.

## **Objective:**

1. Improve Efficiency: The system aims to streamline restaurant operations by automating various processes, reducing manual tasks, and minimizing errors. This efficiency improvement leads to faster order processing and better service.To make the system more manageable and organize it programmatically.
2. Enhance Customer Experience: By providing a user-friendly interface for placing orders, managing reservations, and offering personalized service through customer databases, the system seeks to enhance the overall dining experience for customers.As an owner handle all the affairs digitally in just one go.

## **Contribution in computer science field:**

1. Helps in digital management of the small business
2. Security and privacy.
3. Networking and Communication.

## **Output:**

1. Noodles data can be created, updated, retrieve and deleted.
2. Custoemr’s feedback and reviews.

**Users of Application:**

There are two users of my **“The Twirling Forks”**

* Adnin
* Customer

## **Functional Requirements:**

The functional requirement of **“The Twirling Forks”**is to keep the check on working environment by controlling different modules.

***As a I want to perform So that I can***

|  |  |  |
| --- | --- | --- |
| Admin | The task of Adding the noodles | Add the noodles in stock |
| Admin | The task of viewing the noodles details | View the details |
| Admin | The task of updating the price | Update the price |
| Admin | The task of Deleting the noodles | Delete the noodles which is sold |
| Admin | The task of Adding discounts | Add the discounts to increase the sale of noodles. |
| Admin | The task of sorting the details | View most to least expensive products |
| Admin | The task of viewing feedbacks | View the feedbacks of the customers |
| Admin | The task of viewing the customers | See customer’s history |

## 

***As a I want to perform So that I can***

|  |  |  |
| --- | --- | --- |
| Customer | The task of viewing the service type | See the services of the noodles shop |
| Customer | The task of adding the details | Add myself to get access of the other functions |
| Customer | The task of looking discounts of noodles | Select the discounts |
| Customer | The task of looking for the menu | select my favorite noodles |
| Customer | The task of looking for cheapest noodles | Select the best noodles according to budget |
| Customer | The task of adding noodles in the cart | Place my order |
| Customer | The task of removing noodles from cart | Confirm my order |
| Customer | The task of giving the feedback | Give the Review on the noodles |
| Customer | The task of viewing the bill | Paymy bill |

# 

# 

# **Object Oriented Programming (OOP) Concept:**

I have implemented the major concept of OOP in my project as it helps to design my code efficiently and every function of code is manageable in an organized way.

The major concept of OOP is used efficiently in my code which are

* Inheritance
* Polymorphism
* Association

## **Inheritance:**

When attributes of different classes used some common attributes we make one class that contains the common attributes of these classes and named that **parent class or base class** and all the other classes inherit this parent class called **child classes or derived classes**.

In my project the concept of inheritance is used widely.

* Superclass:Food The superclass "Food" would define common properties and methods that are shared among noodles, such as "name," "price," "stock," and "discounts."
* Subclass: Noodle
* The subclass "Noodle" would be represented as a subclass of "Food."This subclass inherit the common properties and methods from the "Food" superclass. Subclasses: Ramen, Ho Fun, Tokoroten, etc.
* Subclasses of Subclass: Each specific type of noodle, like Ramen, Ho Fun, Tokoroten, etc., would be represented as a subclass of "Noodle." These subclasses inherit the common properties and methods from the "Noodle".
* Each subclass can also have its own unique properties and methods specific to that type of noodle. By using inheritance in this way, "The Twirling Forks" can achieve code reusability, as the common properties and methods are defined in the superclass and automatically inherited by all the subclasses. This approach also promotes a more organized and structured design, allowing easy maintenance and updates in the future.

## **Polymorphism**

When function decide on runtime that of which class it should be we use the concept of polymorphism

* I have polymorphised the role function in which admin or customer can sign them in application. The virtual function is created in Parent class “Person” and child classes “Admin and Custoemr” override this function.

## **Association:**

There is aggregation in all the project because there is no one in the project which lost if anyone of the class die so, there is minimal relation of Aggregation between Person class and Admin class ,Customer class and Food class and nNodles class .

# **Comparison with Procedural Programming:**

* In procedural programming, every part of code is in the main file. It is difficult to find and modify the specific part of code. But in OOP every part is well organized in different paths and make it manageable.
* In procedural programming, some parts of code are that it is repeated again and again, but in object oriented programming, using the concept of inheritance the common part of code is written once and is accessible to its child.
* Using the concept of encapsulation and access modifiers, the security of data is ensured by class itself but procedural programming does not provide this property.
* In object oriented programming, we can call the child class function by parent object.it makes our code manageable and even we don’t have to remind everything.
* In object oriented programming, the code is more manageable and well designed and by developing relations between classes we can not only make front end design better but also backend design.

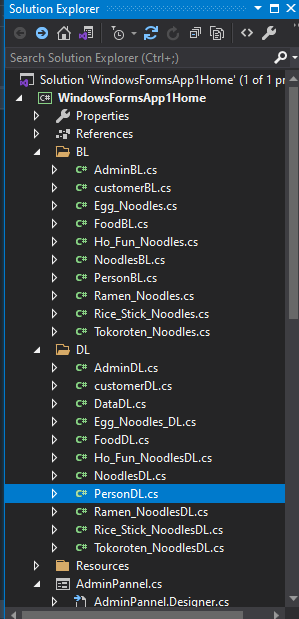
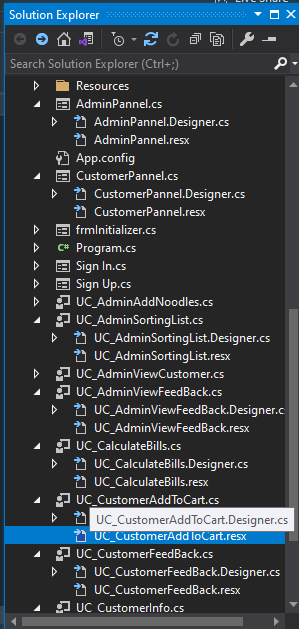
# **Design Pattern Implementation:**

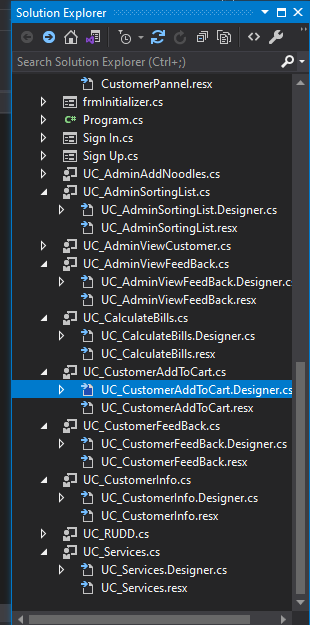
In the development of a software, the design of software is very important. It helps the programmer to manage the software components in a manageable and effective way. In designing a software, we usually divide the software into three layers to keep component manageable.

* + - * + Business Layer(BL)
        + Data Layer(DL)
        + User Interface Layer (UI)

Same goes with my project every layer handles its own functionality and manages its own concerns.

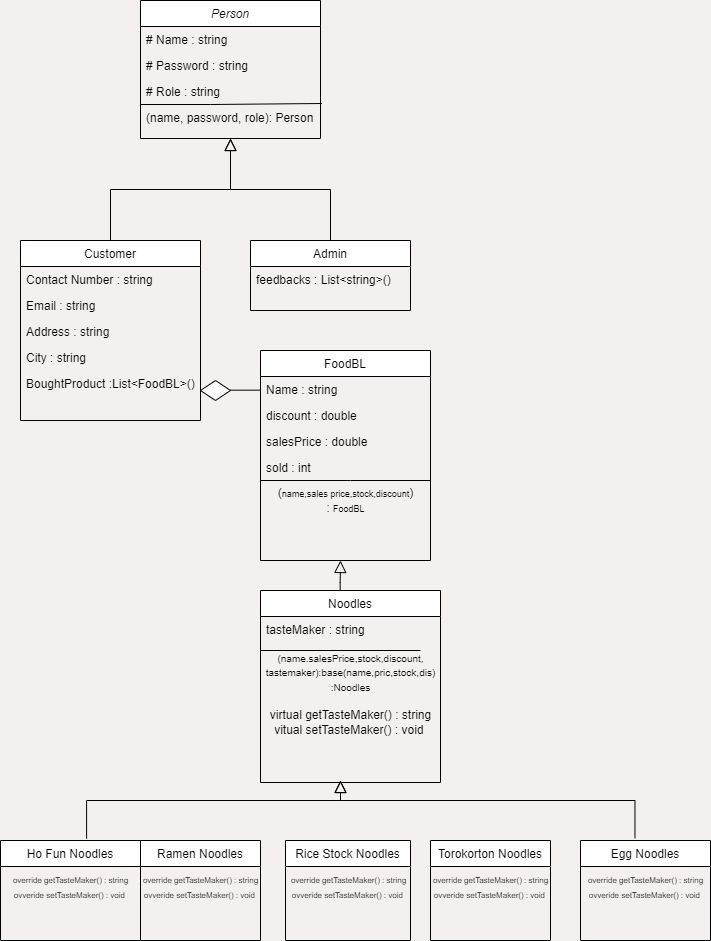
## **Code:**





## 

## **Class Diagram:**



## 

## **Business Layer (BL):**

## The BL pattern focuses on encapsulating the business rules and logic of the application**.** The BLlayer focus on security of data. It gives class the responsibility to protect its own data. My project also focus on separate Business layer to give security to data the security to data s given by access modifiers i.e. protected for parent class and public for child class and using getter and setter.

**Parent Class:**

public class FoodBL //parent class

{ //attributes

private string name;

private double discount;

private int stock;

private double salePrice;

private double actualPrice;

protected int sold = 0;

protected double quantity;

protected double bill;

public string Name { get => name; set => name = value; }

public double Discount { get => discount; set => discount = value; }

public int Stock { get => stock; set => stock = value; }

public double SalePrice { get => salePrice; set => salePrice = value; }

public FoodBL() { }//default constructor

public FoodBL(string name, double actualPrice, double salePrice, int stock, double discount)//constructor with five attributes

{

this.actualPrice = actualPrice;

this.Discount = discount;

this.Name = name;

this.SalePrice = salePrice;

this.Stock = stock;

}

public FoodBL(string name, double salePrice, int stock, double discount)//constructor with four attributes

{

this.Discount = discount;

this.Name = name;

this.SalePrice = salePrice;

this.Stock = stock;

}

public void setsold(int stock)//setter ftn for the stock which is sold

{

this.sold = this.sold + stock;

}

public int getsold()//getter ftn for the stock which is sold

{

return this.sold;

}

public void setName(string name)//setter ftn for the name of noodles

{

this.Name = name;

}

public string getName()//getter ftn for the name of the noodles

{

return Name;

}

public void setactualPrice(double actualPrice)//setter ftn for actual price of noodles

{

this.actualPrice = actualPrice;

}

public double getactualPrice()//getter ftn for actual price of noodles

{

return actualPrice;

}

public void setquantity(int quantity)//setter ftn for quantity of noodles

{

this.quantity = quantity;

}

public double getquantity()//getter ftn for the quantity of noodles

{

return quantity;

}

public void setsalePrice(double salePrice)//setter ftn for sales price of noodles

{

this.SalePrice = salePrice;

}

public double getsalePrice()//getter ftn for the sales price

{

return SalePrice;

}

public void setstock(int stock)//setter ftn for stock of noodles

{

this.Stock = stock;

}

public int getstock()//getter ftn for the stock

{

return Stock;

}

public void setsdiscount(double discount)//setter ftn for disvcount of noodles

{

this.Discount = discount;

}

public double getsdiscount()//getter ftn for the discounts

{

return Discount;

}

public void setbill(double bill)//setter ftn for bill of noodles

{

this.bill = bill;

}

public double getbill()//getter ftn for the bill

{

return bill;

}

}

**Child Class:**

public class NoodlesBL : FoodBL//NoodlesBL is child class of parent class FoodBL

{

protected string tasteMaker;//attribute

public NoodlesBL() { }

public NoodlesBL(string name, double salePrice) { }//default constructor

public NoodlesBL(string name, double actualPrice, double salePrice, int stock, double discount,string tasteMaker) : base (name,actualPrice,salePrice, stock, discount)//constructor with all the attributes

{

this.tasteMaker = tasteMaker;

}

public NoodlesBL(string name, double salePrice, double quantity, double discount)

{

this.Name = name;

this.SalePrice = salePrice;

this.quantity = quantity;

this.Discount = discount;

}

public NoodlesBL(string name, double salePrice, int quantity)

{

this.Name = name;

this.SalePrice = salePrice;

this.quantity = quantity;

}

public virtual void setTasteMaker(string tasteMaker)//setter virtual ftn of tastemaker of noodles

{

this.tasteMaker = tasteMaker;

}

public virtual string getTasteMaker()//getter virtual ftn of tastemaker of noodles

{

return tasteMaker;

}

}

**Grand Child Ho Fun:**

class Ho\_Fun\_Noodles : NoodlesBL//grand child of personBL and child of NoodlesBL

{

public override void setTasteMaker(string tasteMaker)//setter overrided ftn of tastemaker of noodles

{

this.tasteMaker = tasteMaker;

}

public override string getTasteMaker()//getter overrided ftn of tastemaker of noodles

{

return tasteMaker;

}

}

**Grand Child Rice Stick:**

class Rice\_Stick\_Noodles : NoodlesBL//grand child of personBL and child of NoodlesBL

{

public override void setTasteMaker(string tasteMaker)//setter overrided ftn of tastemaker of noodles

{

this.tasteMaker = tasteMaker;

}

public override string getTasteMaker()//getter overrided ftn of tastemaker of noodles

{

return this.tasteMaker;

}

}

**Parent Class:**

public class PersonBL //parent class

{

protected string name;

protected string password;

protected string role;

public PersonBL() { }//default constructor

public PersonBL(string name, string password) //constructor with two parameters

{

this.name = name;

this.password = password;

}

public PersonBL(string name, string password, string role) //constructor with three parameters

{

this.name = name;

this.password = password;

this.role = role;

}

public virtual void setRole(string role)//setter virtual function of role

{

this.role = role;

}

public virtual string getRole()//getter virtual function of the role

{

return role;

}

public void setname(string name)//setter function for name of the user

{

this.name = name;

}

public string getname()//getter function for the name

{

return name;

}

public void setpassword(string password)//setter function for the password

{

this.password = password;

}

public string getpassword()//getter function for the password

{

return password;

}

public bool isAdmin()

{

if (role == "Admin" || role == "admin")

{

return true;

}

return false;

}

public bool isCustomer()

{

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

{

return true;

}

return false;

}

}

**Child Class Admin:**

class AdminBL : PersonBL//admin is a child class having personBL as a parent

{

private List<FoodBL> products = new List<FoodBL>();//list to store product details

public AdminBL() { }//default constructors

public AdminBL(string name, string password,string role) : base(name,password,role)//constructor with three attributes

{

this.name = name;

this.password = password;

this.role = role;

}

public AdminBL(string name, string password) : base(name,password)//constructor with two attributes

{

this.name = name;

this.password = password;

}

public void addProducts(FoodBL F)//ftn to add product in list

{

products.Add(F);

}

public void setAllProducts(List<FoodBL> products)//ftn to set all he products

{

this.products = products;

}

public List<FoodBL> getAllProducts()//ftn to get all the products

{

return products;

}

public void addDiscount(FoodBL F)//ftn to add discount in list

{

products.Add(F);

}

public void setDiscount(List<FoodBL> products)//discount setter function

{

this.products = products;

}

public List<FoodBL> getDiscount()//discount getter ftn

{

return products;

}

public void makeBill(FoodBL F)//add bought products in list

{

products.Add(F);

}

public void setBill(List<FoodBL> products)//setter ftn of bill

{

this.products = products;

}

public List<FoodBL> getBill()//getter ftn of bill

{

return products;

}

public void deleteProducts(int index)//ftn todelete fproducts

{

products.RemoveAt(index);

}

public override void setRole(string role)//role setter ftn overrided from parent class personBL

{

this.role = "admin";

}

public override string getRole()//role getter ftn overrided from parent class personBL

{

return role;

}

}

**Child Class Customer:**

public class customerBL : PersonBL

{

//attributes of the child class customer

private string adress;

private string email;

private string contact;

public static List<FoodBL> ProductBuy = new List<FoodBL>() ;//list to store the bought noodles

public static List<FoodBL> getBuyProducts()//getter ftn of customer bought object list

{

return ProductBuy;

}

public string getEmail()//getter ftn of customer email

{

return email;

}

public void setemail(string email)//setter function of customer email

{

this.email = email;

}

public string getContact()//getter ftn of customer email

{

return contact;

}

public void setContact(string contact)//setter function of customer email

{

this.contact = contact;

}

public customerBL(string name, string password, string role) : base(name, password, role) //constructor of child class accessing the attribute from the (base) parent class with role

{

ProductBuy = new List<FoodBL>();

}

public customerBL(string name, string password, string adress, string contact, string email) : base(name, password)//constructor of child class accessing the attribute from the (base) parent class withlout role

{

this.adress = adress;

this.contact = contact;

this.email = email;

}

public customerBL() { }//default constructor

public customerBL(string name, string password) : base(name, password)//constructor with two parameters of child class

{

this.name = name;

this.password = password;

}

public override void setRole(string role)//setter function of role

{

this.role = "customer";

}

public override string getRole()//getter function of role

{

return role;

}

public static void addBuyProduct(FoodBL product)

{

ProductBuy.Add(product);

}

public void addbuyProduct(FoodBL F)//ftn to add bought noodles in list

{

ProductBuy.Add(F);

}

public List<FoodBL> viewAllProducts()//ftn to view all the noodles data stored in list

{

return ProductBuy;

}

public static void add\_item\_Info\_InList(FoodBL i)//ftn to add noodles data in list

{

ProductBuy.Add(i);

}

public double calculateBill()

{

double bill = 0;

foreach(var i in getBuyProducts())

{

bill += i.getstock() \* i.getsalePrice();

}

return bill;

}

}

The above code of classes is added to focus on how business provides security to its data by encapsulation and using getter and setter methods. We also used constructor in the class in business class

## **Data Layer (DL):**

Data layer is responsible for how data is store, organize and process, in other words we can say that data layer is responsible for the data of class. It has a static list or array to which data is store moreover it handles most of the operation that perform on the data such as search, view and save on some permanent storage such as files.

**PersonDL**:

class PersonDL

{

public static List<PersonBL> user = new List<PersonBL>(); //list to add users

public static PersonBL signIn(PersonBL u) //ftn to sign in the user

{

foreach (PersonBL storedUser in user)

{

if (u.getname() == storedUser.getname() && u.getpassword() == storedUser.getpassword())

{

return storedUser;

}

}

return null;

}

public static void storeDataInList(PersonBL users) //ftn to store the data of the noodles in the file

{

user.Add(users);

}

public static void storeDataInFile(string path,PersonBL p)//ftn to store data of the user in the file

{

StreamWriter file = new StreamWriter("Data.txt",true);

file.WriteLine(p.getname() + "," + p.getpassword() + "," + p.getRole());

file.Flush();

file.Close();

}

public static void readData(string path)//ftn to read the data of the noodles from file

{

if (File.Exists(path))

{

StreamReader fileVariable = new StreamReader("Data.txt");

string record;

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

{

string names = parseData(record, 1);

string password = parseData(record, 2);

string role = parseData(record, 3);

PersonBL o = new PersonBL(names, password, role);

storeDataInList(o);

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

{

customerDL.addCustomerInList(new customerBL(names, password, role));

}

user.Add(o);

}

fileVariable.Close();

}

else

{

Console.Write("Doesnot Exists");

}

}

public static void signUp(string path, string n, string p, ref int usersCount, ref int userArrSize) //sign up function for the usserq

{

StreamWriter myfile = new StreamWriter(path, true);

myfile.WriteLine(n + "," + p);

myfile.Flush();

myfile.Close();

if (usersCount < userArrSize)

{

user[usersCount] = addUserData(n, p);

usersCount++;

}

}

public static PersonBL addUserData(string n, string p)//ftn to as add user name and the password

{

PersonBL s1 = new PersonBL();

s1.setname(n);

s1.setpassword(p);

return s1;

}

public static string parseData(string record, int field)//ftn used in storing data of both products and the user

{

int comma = 1;

string item = "";

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

{

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

{

comma++;

}

else if (comma == field)

{

item = item + record[x];

}

}

return item;

}

public static void addObjectIntoList(PersonBL person)//Function to add User object into the list

{

user.Add(person);

}

public static void setObjectInList(PersonBL user)//ftn to add usuer object in list

{

addObjectIntoList(user);

}

## }

**customerDL**:

class customerDL

{ private static List<customerBL> customersList = new List<customerBL>(); //list to add the customers

public static customerBL m = new customerBL();

internal static List<customerBL> CustomersList { get => customersList; set => customersList = value; } //get and set function of customerList

public static List<customerBL> getaddCustomer() //get function of customerList

{

return customersList;

}

public static void addCustomerInList(customerBL temp) //function to add customer in the list

{

customersList.Add(temp);

}

public static customerBL returnCustomer(string password)//function to add the customer if the passwords are same

{

foreach (customerBL p in customersList)

{

if (p.getpassword() == password)

{

return p;

} } return null; }

public static void removeCustomer(string password) //function to remove customer from list

{

foreach (customerBL storedUser in customersList)

{

if (storedUser.getpassword() == password)

{

customersList.Remove(storedUser);

} } }

public static customerBL getCustomerByNameAndPassword(string name, string password)

{

foreach (customerBL customer in getaddCustomer())

{

if (customer.getname() == name && customer.getpassword() == password)

{ return customer; } }

return null;

} }

**FoodDL**:

class FoodDL

{

private static List<FoodBL> Products = new List<FoodBL>(); //list to add the details of the noodles

public static List<FoodBL> sortedList = new List<FoodBL>(); //list to sort the data of noodles

public static void addFood(FoodBL F) //function to add food object into the list

{

Products.Add(F);

}

public static void RemoveParticularFood(FoodBL noodle) //ftn to remove the particular index of the nooodles

{

Products.Remove(noodle);

}

public static void UpdateProduct(FoodBL p, FoodBL n)//Funtion to Remove products in the list

{

p.Name = n.Name ;

p.SalePrice = n.SalePrice;

p.Stock = n.Stock;

p.Discount = n.Discount;

}

public static List<FoodBL> getProducts() //get ftn of product list

{

return Products;

}

public static List<FoodBL> sortByPrice() //ftn to sort the data of noodles by price

{

List<FoodBL> temp = Products.OrderByDescending(o => o.getsalePrice()).ToList();

return temp;

}

public static List<FoodBL> sortBysold()//ftn to sort the data of noodles by sold

{

List<FoodBL> temp = Products.OrderByDescending(o => o.getsold()).ToList();

return temp;

}

public static int GetParticularIndex(string name) //ftn to get the particular index of noodles

{

for(int i = 0; i < Products.Count; i++ )

{

if(Products[i].getName().ToLower() == name)

{

return i;

}

}

return 5956;

}

public static void readNoodlesData(string path1)//filehandling to read data of noodles

{

if (File.Exists(path1))

{

StreamReader fileVariable = new StreamReader("Noodles.txt");

string record;

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

{

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

string Name = splittedRecord[0];

double Price = double.Parse(splittedRecord[1]);

int Stock = int.Parse(splittedRecord[2]);

double discount = double.Parse(splittedRecord[3]);

FoodBL nood = new FoodBL(Name, Price, Stock, discount);

addFood(nood);

}

fileVariable.Close();

}

else

{

//FoodUI.readNoodlesData1();

}

}

public static void sorting(List<FoodBL> products)//Function for view the Sorted List

{

List<FoodBL> sortedList = products.OrderByDescending(o => o.getsalePrice()).ToList();

Console.WriteLine("Name\t " + " : " + "Sales price " + " : " + "Stock" + " : " + "Discount");

foreach (var storedUser in sortedList)

{

Console.WriteLine(storedUser.getName() + "\t:\t" + storedUser.getsalePrice() + "\t:\t" + storedUser.getstock() + "\t:\t" + storedUser.getsdiscount());

}

}

public static List<FoodBL> SortByQuntity(List<FoodBL> Products) // ftn to sort the noodles by quantity

{

List<FoodBL> sortedList = getProducts().OrderByDescending(o => o.getquantity()).ToList();

return sortedList;

}

public static void writeItemDataInFile(string path1) //ftn to write the data of

{

StreamWriter myfile = new StreamWriter("Noodles.txt");

foreach (FoodBL storedUser in getProducts())

{

myfile.WriteLine(storedUser.getName() + "," + storedUser.getsalePrice() + "," + storedUser.getstock() + "," + storedUser.getsdiscount());

}

myfile.Flush();

myfile.Close();

}

public static void writeItemDataInFile() //ftn to write the data of

{

StreamWriter myfile = new StreamWriter("Noodles.txt");

foreach (FoodBL storedUser in getProducts())

{

myfile.WriteLine(storedUser.getName() + "," + storedUser.getsalePrice() + "," + storedUser.getstock() + "," + storedUser.getsdiscount());

}

myfile.Flush();

myfile.Close();

}

public static NoodlesBL getProductByName(string name)

{

foreach (NoodlesBL product in FoodDL.getProducts())

{

if (product.getName() == name)

{

return product;

}

}

return null; // If no matching product is found

}

}

**DataDL**:

class DataDL

{

private static string option;

private static string adminOption;

private static string customerOption;

private static string path;

private static string path1;

private static bool check;

private static bool check1;

private static PersonBL signUp;

private static PersonBL signIn;

private static bool flag;

private static bool flag1;

private static double total;

public static customerBL customer;

public static void SetOption(string option1)

{

option = option1;

}

public static string GetOption()

{

return option;

}

public static void SetAdminOption(string adminOption1)

{

adminOption = adminOption1;

}

public static string GetAdminOption()

{

return adminOption;

}

public static void SetCustomerOption(string customerOption1)

{

customerOption = customerOption1;

}

public static string GetCustomerOption()

{

return customerOption;

}

public static void SetUsersPath(string path1)

{

path = path1;

}

public static string GetUsersPath()

{

return path;

}

public static void SetProductsPath(string path2)

{

path1 = path2;

}

public static string GetProductsPath()

{

return path1;

}

public static void SetCheck(bool check1)

{

check = check1;

}

public static bool GetCheck()

{

return check;

}

public static void SetCheck1(bool check2)

{

check1 = check2;

}

public static bool GetCheck1()

{

return check1;

}

public static void SetSignUpObject(PersonBL u)

{

signUp = u;

}

public static PersonBL GetSignUpObject()

{

return signUp;

}

public static void SetSignInObject(PersonBL s)

{

signIn = s;

}

public static PersonBL GetSignInObject()

{

return signIn;

}

public static void SetFlag(bool flag1)

{

flag = flag1;

}

public static bool GetFlag()

{

return flag;

}

public static void SetFlag1(bool flag2)

{

flag1 = flag2;

}

public static bool GetFlag1()

{

return flag1;

}

public static void SetTotal(double total1)

{

total = total1;

}

public static double GetTotal()

{

return total;

}

public static void AddCustomerObject(customerBL c)

{

customer = c;

}

public static PersonBL GetCustomerObject()

{

return customer;

}

}

## **User Interface:**

As name show user interface layer focus on how software is shown to the users, how user communicate to the user and how menus and sets shown to the user. It consists of all the function which user performs and all the menus and functions which show to the user.

**frmSignIn**:

public partial class frmSignIn : Form

{

public frmSignIn()

{

InitializeComponent();

PersonDL.readData("Data.txt");

}

private void btnSignIn\_Click(object sender, EventArgs e)//ftn to sign in the user

{

DataDL.SetSignInObject(takeInputWithOutRoleSignIn());

if (DataDL.GetSignInObject() != null)

{

DataDL.SetSignInObject(PersonDL.signIn(DataDL.GetSignInObject()));

if (DataDL.GetSignInObject() == null)

{

MessageBox.Show("InValid Input");

}

else if (DataDL.GetSignInObject().isAdmin())

{

AdminPannel dis = new AdminPannel();

dis.Show();

}

else if(DataDL.GetSignInObject().isCustomer())

{

frmCustomerPannel dis = new frmCustomerPannel();

dis.Show();

}

}

}

private PersonBL takeInputWithOutRoleSignIn()//ftn to take input of user without role

{

string name = txtSignInUserName.Text;

string password = txtSignInPassword.Text;

if (name != null && password != null)

{

PersonBL user = new PersonBL(name, password);

return user;

}

return null;

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)//ftn to open signup form

{

this.Hide();

frmSignUp obj = new frmSignUp();

obj.ShowDialog();

}

private void frmSignIn\_Load(object sender, EventArgs e)//ftn to read dara of noodles from file

{

FoodDL.readNoodlesData("Noodles.txt");

}

}

**frmSignUp**:

public partial class frmSignUp : Form

{

public frmSignUp()

{

InitializeComponent();

}

private void btnSignUp\_Click(object sender, EventArgs e)//ftn to sign up user

{

new frmSignIn();

}

private void lblSignIn\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)//ftn to sign in the user

{

new frmSignIn();

}

private void btnSignUpp\_Click(object sender, EventArgs e)//ftn to signup the user

{

DataDL.SetSignUpObject(TakeInputWithRoleSignUp());

if (DataDL.GetSignUpObject() != null)

{

PersonBL signUp\_User = DataDL.GetSignUpObject();

PersonDL.addObjectIntoList(signUp\_User);

if (signUp\_User.getRole() == "customer")

{

customerDL.addCustomerInList(new customerBL(signUp\_User.getname(), signUp\_User.getpassword(), signUp\_User.getRole()));

}

PersonDL.storeDataInFile("Data.txt",DataDL.GetSignUpObject());

frmSignIn form = new frmSignIn();

form.Show();

this.Hide();

}

}

private PersonBL TakeInputWithRoleSignUp()//ftn to take input with role

{

string name = txtSignUpUserName.Text;

string password = txtPasswordSignUp.Text;

string role = txtRoleSignUp.Text;

if (string.IsNullOrWhiteSpace(name) || string.IsNullOrWhiteSpace(password) || string.IsNullOrWhiteSpace(role))

{

MessageBox.Show("Please fill in all required fields.", "Invalid Information", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return null;

}

// Validation for the password

if (!IsValidPassword(password))

{

MessageBox.Show("Password must contain at least one alphabet, one numeric digit, and one special character (@#$%^&\*!) and be at least 8 characters long.", "Invalid Password", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return null;

}

if (role.Equals("Customer", StringComparison.OrdinalIgnoreCase))

{

customerBL user = new customerBL(name, password, role);

MessageBox.Show("SignedUp Successfully");

return user;

}

else if (role.Equals("Admin", StringComparison.OrdinalIgnoreCase))

{

AdminBL user = new AdminBL(name, password, role);

MessageBox.Show("SignedUp Successfully");

return user;

}

return null;

}

private void lnklblSignUp\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Hide();

frmSignIn obj = new frmSignIn();

obj.ShowDialog();

}

private bool IsValidPassword(string password)

{

// Password must contain at least one alphabet, one numeric digit, and one special character (@#$%^&\*!) and be at least 8 characters long.

string pattern = @"^(?=.\*[a-zA-Z])(?=.\*\d)(?=.\*[@#$%^&\*!]).{8,}$";

return Regex.IsMatch(password, pattern);

}

}

**frmAdminPannel**:

public partial class AdminPannel : Form

{

public AdminPannel()

{

InitializeComponent();

}

private void loadforms(UC\_AdminAddNoodles frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms(UC\_AdminSortingList frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms(UC\_RUDD frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms(UC\_AdminViewCustomer frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms1(UC\_AdminSortingList frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms1(UC\_AdminViewFeedBack frm)

{

try

{

this.tblAdminChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblAdminChangable.Controls.Add(f);

this.tblAdminChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void addControl(UserControl c)

{

c.Dock = DockStyle.Fill;

AdminPannel obj = new AdminPannel();

obj.Controls.Add(c);

c.BringToFront();

}

private void lblAddNoodles\_Click(object sender, EventArgs e)//ftn to load user control of admin add noodles

{

loadforms(new UC\_AdminAddNoodles());

}

private void lblUpdateNoodles\_Click(object sender, EventArgs e)//ftn to load user control of admin updating data according to price

{

loadforms(new UC\_AdminSortingList());

}

private void lblViewNoodles\_Click(object sender, EventArgs e)//ftn to load user control of update,retrieve,delete and discounts

{

loadforms(new UC\_RUDD());

}

private void lblSortedData\_Click(object sender, EventArgs e)//ftn to load user control of admin sorted data according to price high to low

{

loadforms1(new UC\_AdminSortingList());

}

private void lblExit\_Click(object sender, EventArgs e)//ftn to load user control of exiting admin pannel

{

Hide();

frmSignIn obj = new frmSignIn();

obj.ShowDialog();

}

private void lblFeedBack\_Click(object sender, EventArgs e)//ftn to load user control of viewing feedback of customers

{

loadforms1(new UC\_AdminViewFeedBack());

}

private void lblAdminViewCustomer\_Click\_1(object sender, EventArgs e)//ftn to load user control of viewing customemrs list

{

loadforms(new UC\_AdminViewCustomer());

}

}

**UC\_AddNoodles**:

public partial class UC\_AdminAddNoodles : UserControl

{

public UC\_AdminAddNoodles()

{

InitializeComponent();

}

private void cmdAdminAddNoodlesAdd\_Click(object sender, EventArgs e)//add noodlesin list and store in file

{

// Validate input fields before adding the product

if (string.IsNullOrWhiteSpace(cmbAdminAddNoodlesName.Text) ||

!double.TryParse(txtAdminAddNoodlesPrice.Text, out double price) || price < 0 ||

!int.TryParse(txtAdminAddNoodlesStock.Text, out int stock) || stock <= 0 ||

!double.TryParse(txtAdminAddNoodlesDiscount.Text, out double discount) || discount < 0 || discount > 100)

{

MessageBox.Show("Please enter valid data in all fields.\nDiscount should be between 0 and 100.", "Invalid Input", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Check if the item already exists in the list

string itemName = cmbAdminAddNoodlesName.Text;

bool itemExists = FoodDL.getProducts().Any(p => p.Name.Equals(itemName, StringComparison.OrdinalIgnoreCase));

if (itemExists)

{

MessageBox.Show($"The item '{itemName}' already exists in the file. Please update it instead.", "Item Already Exists", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// If input is valid and the item doesn't exist, add the product to the list

FoodBL newProduct = new FoodBL(itemName, double.Parse(txtAdminAddNoodlesPrice.Text), int.Parse(txtAdminAddNoodlesStock.Text), double.Parse(txtAdminAddNoodlesDiscount.Text));

FoodDL.getProducts().Add(newProduct);

FoodDL.writeItemDataInFile("Noodles.txt");

MessageBox.Show("Product added successfully to the file.");

}

private void cmdAdminAddNoodlesBack\_Click(object sender, EventArgs e)//open admin pannel

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

}

**UC\_RDD(Update,Retrieve,Delete,View and discounts)**:

public partial class UC\_RUDD : UserControl

{

FoodBL selectedProduct;

public UC\_RUDD()

{

InitializeComponent();

}

private void loadDataInTextBoxes(FoodBL p)//storing data in object

{

cmbCrudName.Text = p.Name;

txtCrudPrice.Text = p.SalePrice.ToString();

txtCrudStock.Text = p.Stock.ToString();

txtCrudDiscount.Text = p.Discount.ToString();

}

private void dataBind()//show the list

{

DataGridCrud.DataSource = null;

DataGridCrud.DataSource = FoodDL.getProducts();

DataGridCrud.Refresh();

}

private void removeDataFromTextBoxes()//remove data from the text boxes

{

cmbCrudName.Text = string.Empty;

txtCrudPrice.Text = string.Empty;

txtCrudStock.Text = string.Empty;

txtCrudDiscount.Text = string.Empty;

}

private void lblCrudUpdate\_Click(object sender, EventArgs e)//update the noodles data

{

if (selectedProduct != null)

{

if (ValidateInput())

{

FoodBL p = new FoodBL(cmbCrudName.Text, double.Parse(txtCrudPrice.Text), int.Parse(txtCrudStock.Text), double.Parse(txtCrudDiscount.Text));

FoodDL.UpdateProduct(selectedProduct, p);

MessageBox.Show("Updated Successfully");

removeDataFromTextBoxes();

dataBind();

selectedProduct = null;

FoodDL.writeItemDataInFile();

}

}

else

{

MessageBox.Show("Please Select Something");

}

}

private void cmdCrudDelete\_Click(object sender, EventArgs e)//delete noodles data

{

if (selectedProduct != null)

{

FoodDL.RemoveParticularFood(selectedProduct);

MessageBox.Show("Deleted Successfully");

removeDataFromTextBoxes();

dataBind();

selectedProduct = null;

FoodDL.writeItemDataInFile("Noodles.txt");

}

else

{

MessageBox.Show("Please Select Something");

}

}

private void UC\_AdminViewNoodles\_Load(object sender, EventArgs e)

{

dataBind();

}

private void DataGridCrud\_CellClick(object sender, DataGridViewCellEventArgs e)//load the data in text boxes while deleting and updating

{

selectedProduct = (FoodBL)DataGridCrud.CurrentRow.DataBoundItem;

loadDataInTextBoxes(selectedProduct);

}

private bool ValidateInput()//validations

{

if (string.IsNullOrWhiteSpace(cmbCrudName.Text) ||

!double.TryParse(txtCrudPrice.Text, out double price) || price < 0 ||

!int.TryParse(txtCrudStock.Text, out int stock) || stock <= 0 ||

!double.TryParse(txtCrudDiscount.Text, out double discount) || discount < 0 || discount > 100)

{

MessageBox.Show("Please enter valid data in all fields.\n price and stock must be > 0 \nDiscount should be between 0 and 100.", "Invalid Input", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

return true;

}

}

**UC\_ViewSortedData**:

public partial class UC\_AdminSortingList : UserControl

{

public UC\_AdminSortingList()

{

InitializeComponent();

}

private void UC\_AdminSortingList\_Load(object sender, EventArgs e)

{

dataGridSortingList.DataSource = null;

dataGridSortingList.DataSource = FoodDL.sortByPrice();

dataGridSortingList.Refresh();

}

private void cmdAdminSortedListBack\_Click(object sender, EventArgs e)

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

}

**UC\_ViewFeedBack**:

public partial class UC\_AdminViewFeedBack : UserControl

{

public UC\_AdminViewFeedBack()

{

InitializeComponent();

}

private void UC\_AdminViewFeedBack\_Load(object sender, EventArgs e)

{

AdminDL.Feedback = ReadFeedbackFromFile();//call here to read the feedback from file

DisplayFeedbackInDataGridView();//call this method here to show the feedback

}

private void DisplayFeedbackInDataGridView()

{

// Clear the existing rows and columns in the DataGridView

dataGridViewFeedback.Rows.Clear();

dataGridViewFeedback.Columns.Clear();

// Add a single column to display the feedback

DataGridViewTextBoxColumn feedbackColumn = new DataGridViewTextBoxColumn

{

Name = "colFeedback",

HeaderText = "Feedback",

AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill

};

dataGridViewFeedback.Columns.Add(feedbackColumn);

// Add the feedback data to the DataGridView

foreach (string feedback in AdminDL.Feedback)

{

int rowIndex = dataGridViewFeedback.Rows.Add(feedback);

}

}

private List<string> ReadFeedbackFromFile()

{

// Read the feedback from the "feedback.txt" file

if (File.Exists("feedback.txt"))

{

using (StreamReader sr = new StreamReader("feedback.txt"))

{

string line;

while ((line = sr.ReadLine()) != null)

{

AdminDL.Feedback.Add(line);

}

}

}

return AdminDL.Feedback;

}

private void DisplayFeedbackInDataGridView(List<string> feedbackList)

{

// Assuming you have a DataGridView named "dataGridViewFeedback" in the UserControl

dataGridViewFeedback.Rows.Clear();

foreach (string feedback in feedbackList)

{

int rowIndex = dataGridViewFeedback.Rows.Add();

dataGridViewFeedback.Rows[rowIndex].Cells[0].Value = feedback;

}

}

private void cmdAdminViewFeedbackBack\_Click(object sender, EventArgs e)//open admin pannel

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

}

**UC\_ViewCustomerInfo**:

public partial class UC\_AdminViewCustomer : UserControl

{

public UC\_AdminViewCustomer()

{

InitializeComponent();

InitializeDataGridView(); // Calling the method here to set up the DataGridView columns

LoadCustomerDataFromFile();//calling ftn here to load the data of the customer from the dile

}

private void UC\_AdminViewCustomer\_Load(object sender, EventArgs e)

{

LoadCustomerDataFromFile();

}

private void InitializeDataGridView()

{

// Clear the existing data in the DataGridView and set AutoGenerateColumns to false

dataGridViewCustomer.Rows.Clear();

dataGridViewCustomer.AutoGenerateColumns = false;

// Create columns for password and contact information

DataGridViewTextBoxColumn passwordColumn = new DataGridViewTextBoxColumn();

passwordColumn.Name = "PasswordColumn";

passwordColumn.HeaderText = "Password";

dataGridViewCustomer.Columns.Add(passwordColumn);

DataGridViewTextBoxColumn contactColumn = new DataGridViewTextBoxColumn();

contactColumn.Name = "ContactColumn";

contactColumn.HeaderText = "Contact";

dataGridViewCustomer.Columns.Add(contactColumn);

}

private void LoadCustomerDataFromFile()

{

try

{

// Read all lines from the "customerInfo.txt" file

string[] lines = File.ReadAllLines("customerInfo.txt");

foreach (string line in lines)

{

// Split the line to extract the password and contact information

string[] parts = line.Split(',');

if (parts.Length == 2)

{

string password = parts[0].Substring(parts[0].IndexOf(":") + 1).Trim(); // Extract the password value

string contact = parts[1].Substring(parts[1].IndexOf(":") + 1).Trim(); // Extract the contact value

// Add the data to the DataGridView

int rowIndex = dataGridViewCustomer.Rows.Add();

dataGridViewCustomer.Rows[rowIndex].Cells["PasswordColumn"].Value = password;

dataGridViewCustomer.Rows[rowIndex].Cells["ContactColumn"].Value = contact;

}

}

}

catch (Exception ex)

{

// Handle any exceptions that may occur while reading the file

MessageBox.Show("Error occurred while reading customer information from file: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void cmdAdminViewCustomer\_Click(object sender, EventArgs e)

{

}

}

**frmCustomerPannel**:

public partial class frmCustomerPannel : Form

{

customerBL customer = new customerBL();

public frmCustomerPannel()

{

InitializeComponent();

}

private void lblCustomerDiscounts\_Click(object sender, EventArgs e) //ftn to load user control of customer sorting list

{

loadforms1(new UC\_AdminSortingList());

}

private void lblCustomerFeedBack\_Click(object sender, EventArgs e)//ftn to load user control of customer give feedback

{

loadforms(new UC\_CustomerFeedBack());

}

private void lblExit\_Click(object sender, EventArgs e)//ftn to exit customer pannel

{

Hide();

frmSignIn obj = new frmSignIn();

obj.ShowDialog();

}

private void lblCustomerAddInfo\_Click(object sender, EventArgs e)//ftn to load user control of customer information

{

loadform2(new UC\_CustomerInfo(customer));

}

private void lblCustomerAddCart\_Click(object sender, EventArgs e)//ftn to load user control of customer add to cart

{

loadforms(new UC\_CustomerAddToCart(customer));

}

private void lblCustomerViewBills\_Click(object sender, EventArgs e)//ftn to load user control of customer view bill

{

loadforms(new UC\_CalculateBills(customer));

}

private void lblCustomerMenu\_Click(object sender, EventArgs e)//ftn to load user control of sorting list

{

loadforms1(new UC\_AdminSortingList());

}

private void lblCustomerCheapestNoodles\_Click(object sender, EventArgs e)//ftn to load user control of viewing cheapest noodles

{

loadforms1(new UC\_AdminSortingList());

}

private void loadforms(UC\_CustomerFeedBack frm) //load ftn of customer feedback

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadform2(UC\_CustomerInfo frm)//load ftn of customer information

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms(UC\_CustomerAddToCart frm)//load ftn of customer add to cart

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms(UC\_CalculateBills frm)//load ftn of customer calculate bills

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void lblCustomerServices\_Click(object sender, EventArgs e)//ftn to load customers services

{

loadforms(new UC\_Services());

}

private void loadforms(UC\_Services frm)//load ftn of customer services

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadforms1(UC\_AdminSortingList frm)//load ftn of customer viewing sorting noodles according to high to low price of noodles

{

try

{

this.tblCustomerChangable.Controls.Clear();

UserControl f = frm as UserControl;

f.Dock = DockStyle.Fill;

this.tblCustomerChangable.Controls.Add(f);

this.tblCustomerChangable.Tag = f;

f.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

**UC\_AddCustomerInfo**:

public partial class UC\_CustomerInfo : UserControl

{

customerBL customer;

public UC\_CustomerInfo(customerBL customer)

{

InitializeComponent();

this.customer = customer;

txtCustomerInfoContact.KeyPress += txtCustomerInfoContact\_KeyPress;

}

private void cmdCustomerInfoAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtCustomerInfoPassword.Text) || txtCustomerInfoContact.Text.Length != 11)

{

MessageBox.Show("Please fill in all required fields with valid data.", "Invalid Information", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

string password = txtCustomerInfoPassword.Text;

string contact = txtCustomerInfoContact.Text;

SaveCustomerInfoToFile(password, contact);

MessageBox.Show("Customer Added", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void cmdCustomerInfoBack\_Click(object sender, EventArgs e)//open customer pannel

{

Hide();

frmCustomerPannel obj = new frmCustomerPannel();

obj.ShowDialog();

}

private void txtCustomerInfoContact\_TextChanged(object sender, EventArgs e)

{

string input = new string(txtCustomerInfoContact.Text.Where(char.IsDigit).ToArray());

// Limit the total number of characters to 11

if (input.Length > 11)

{

input = input.Substring(0, 11);

}

// Update the TextBox text with the validated input

txtCustomerInfoContact.Text = input;

txtCustomerInfoContact.SelectionStart = input.Length;

}

private void SaveCustomerInfoToFile(string password, string contact)

{

// Construct the customer information string to be saved in the file

string customerInfo = $"Password: {password}, Contact: {contact}";

try

{

// Open the "customerInfo.txt" file for appending (if the file doesn't exist, it will be created)

using (StreamWriter sw = File.AppendText("customerInfo.txt"))

{

// Write the customer information to the file

sw.WriteLine(customerInfo);

}

}

catch (Exception ex)

{

// Handle any exceptions that may occur while saving to the file

MessageBox.Show("Error occurred while saving customer information to file: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void txtCustomerInfoContact\_KeyPress(object sender, KeyPressEventArgs e) { }

}

**UC\_Services**:

public partial class UC\_Services : UserControl

{

public UC\_Services()

{

InitializeComponent();

}

private void cmdCustomerrServicesBack\_Click(object sender, EventArgs e)

{

Hide();

frmCustomerPannel obj = new frmCustomerPannel();

obj.ShowDialog();

}

}

**UC\_Menu**:

public UC\_Menu()

{

InitializeComponent();

}

private void UC\_AdminSortingList\_Load(object sender, EventArgs e)

{

dataGridSortingList.DataSource = null;

dataGridSortingList.DataSource = FoodDL.sortByPrice();

dataGridSortingList.Refresh();

}

private void cmdAdminSortedListBack\_Click(object sender, EventArgs e)

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

**UC\_CheapestNoodles**:

public UC\_cheapest()

{

InitializeComponent();

}

private void UC\_AdminSortingList\_Load(object sender, EventArgs e)

{

dataGridSortingList.DataSource = null;

dataGridSortingList.DataSource = FoodDL.sortByPrice();

dataGridSortingList.Refresh();

}

private void cmdAdminSortedListBack\_Click(object sender, EventArgs e)

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

**UC\_Discounts**:

public UC\_Discounts()

{

InitializeComponent();

}

private void UC\_AdminSortingList\_Load(object sender, EventArgs e)

{

dataGridSortingList.DataSource = null;

dataGridSortingList.DataSource = FoodDL.sortByPrice();

dataGridSortingList.Refresh();

}

private void cmdAdminSortedListBack\_Click(object sender, EventArgs e)

{

Hide();

AdminPannel obj = new AdminPannel();

obj.ShowDialog();

}

**UC\_AddToCart**:

public partial class UC\_CustomerAddToCart : UserControl

{

customerBL customer;

public UC\_CustomerAddToCart(customerBL customer)

{

InitializeComponent();

this.customer = customer;

}

private void UC\_CustomerAddToCart\_Load(object sender, EventArgs e)

{

cmbAddToCartName.DataSource = null;

cmbAddToCartName.DataSource = FoodDL.getProducts();

cmbAddToCartName.DisplayMember = "Name";

cmbAddToCartName.Refresh();

}

private void btnAddToCart\_Click(object sender, EventArgs e)

{

FoodBL f = (FoodBL)cmbAddToCartName.SelectedItem;

FoodBL fnew = new FoodBL(f.Name, f.SalePrice, int.Parse(txtAddToCardQuantity.Text), f.getsdiscount());

f.setstock(f.getstock() - fnew.getstock());

customerBL.addBuyProduct(fnew);

FoodDL.writeItemDataInFile();

dataBind();

}

private void dataBind()

{

DataGridAddToCart.DataSource = null;

DataGridAddToCart.DataSource = customerBL.getBuyProducts();

DataGridAddToCart.Refresh();

}

private void cmdAddToCartBack\_Click(object sender, EventArgs e)

{

Hide();

frmCustomerPannel obj = new frmCustomerPannel();

obj.ShowDialog();

}

}

**UC\_ViewBill**:

public partial class UC\_CalculateBills : UserControl

{

customerBL customer;

public UC\_CalculateBills(customerBL customer)

{

InitializeComponent();

this.customer = customer;

lblBillsPrice.Text = customer.calculateBill().ToString();

}

private void cmdCustomerCalculateBillBack\_Click(object sender, EventArgs e)

{

Hide();

frmCustomerPannel obj = new frmCustomerPannel();

obj.ShowDialog();

}

}

**UC\_GiveFeedback**:

public partial class UC\_CustomerFeedBack : UserControl

{

public UC\_CustomerFeedBack()

{

InitializeComponent();

}

private void cmdCustomerAddFeedback\_Click(object sender, EventArgs e)//add feeedback in file and list

{

string feedbackText = txtCustomerAddFeedback.Text.Trim();

if (string.IsNullOrWhiteSpace(feedbackText))

{

MessageBox.Show("Please enter your feedback.", "Empty Feedback", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Save the customer feedback to the "feedback.txt" file

SaveFeedbackToFile(feedbackText);

MessageBox.Show("Feedback added successfully.");

}

private void cmdAdminAddNoodlesBack\_Click(object sender, EventArgs e)//open customer pannel

{

Hide();

frmCustomerPannel obj = new frmCustomerPannel();

obj.ShowDialog();

}

private void SaveFeedbackToFile(string feedbackText)

{

// Write the feedback to the "feedback.txt" file

using (StreamWriter sw = new StreamWriter("feedback.txt", true))

{

sw.WriteLine(feedbackText);

}

}

}

# **Key Classes:**

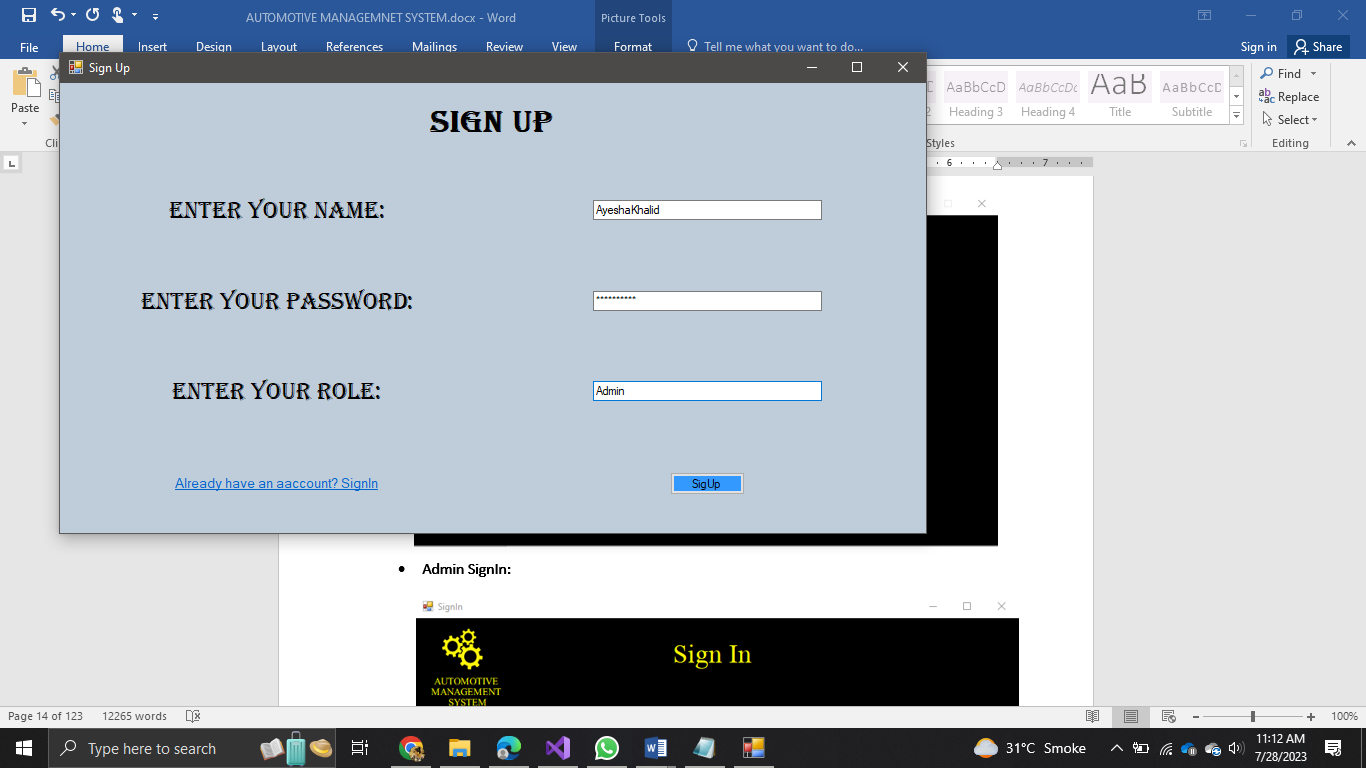
The project consists of two major things and all the operation perform on these things and that’s why they are named as parent classes

* + - * Person
      * Food

## **Wireframes:**

### **Main Screen:**

### **Role Screen**



### **Admin Menu:**



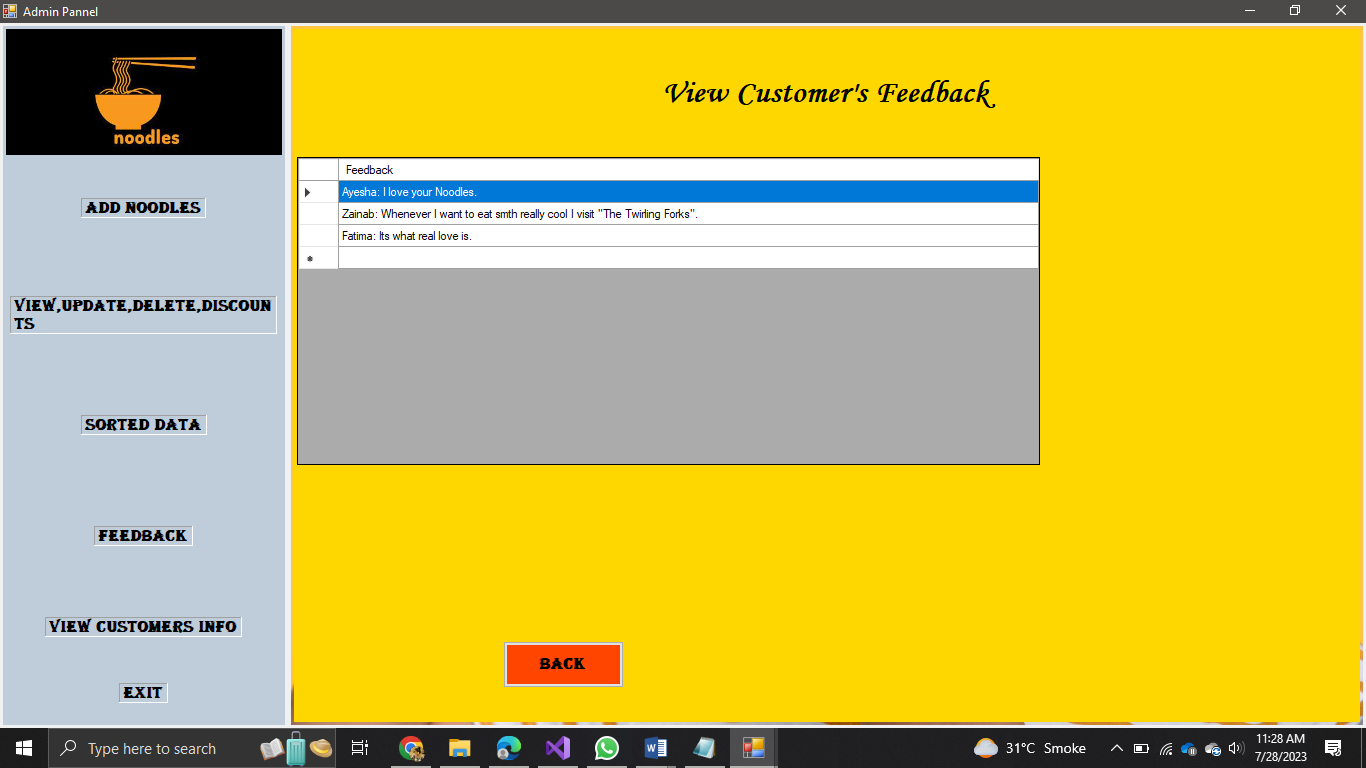
### **Add Noodles:**



### **View,Update,Delete and discounts:**

### **View Sorted Data:**

### **View Feedbacks:**



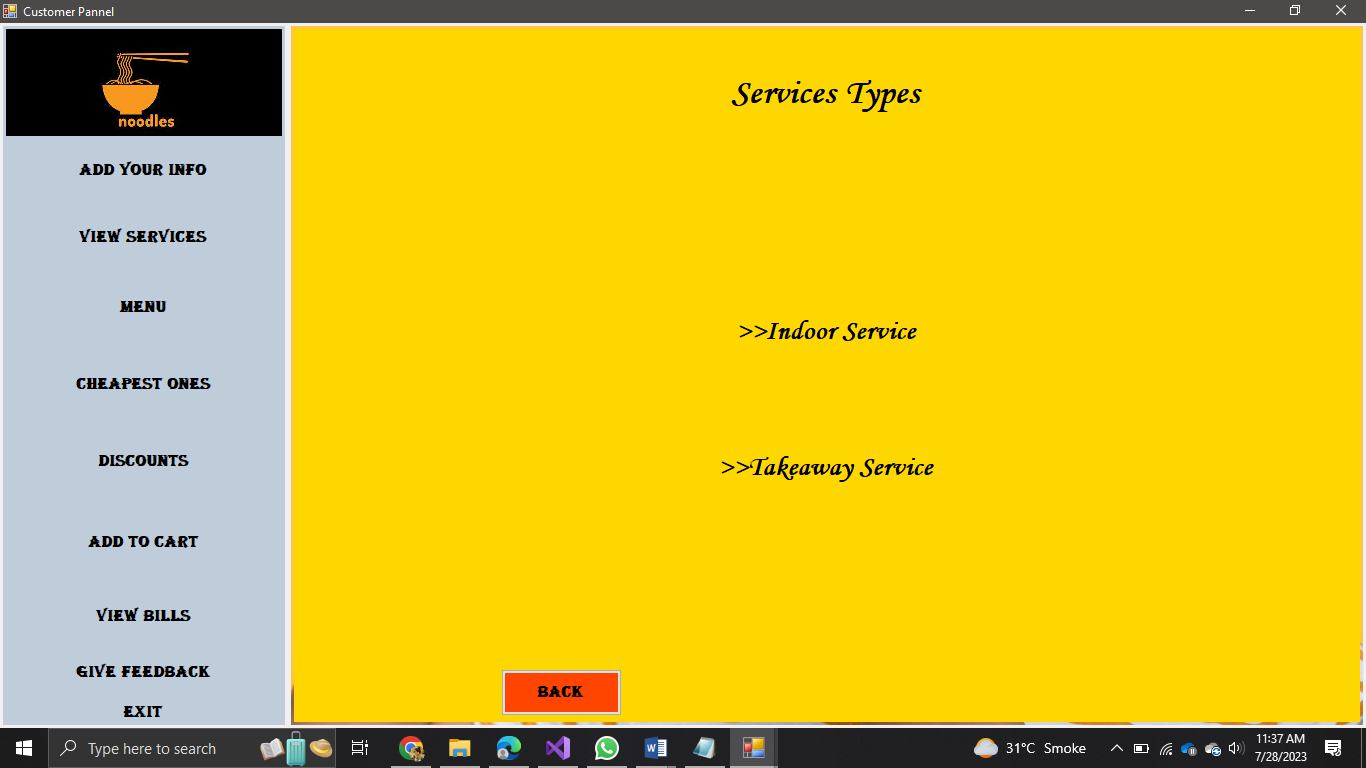
### **View Customer’s Information:**

### 

### **Customer’s Menu:**

### **Add Customer:**

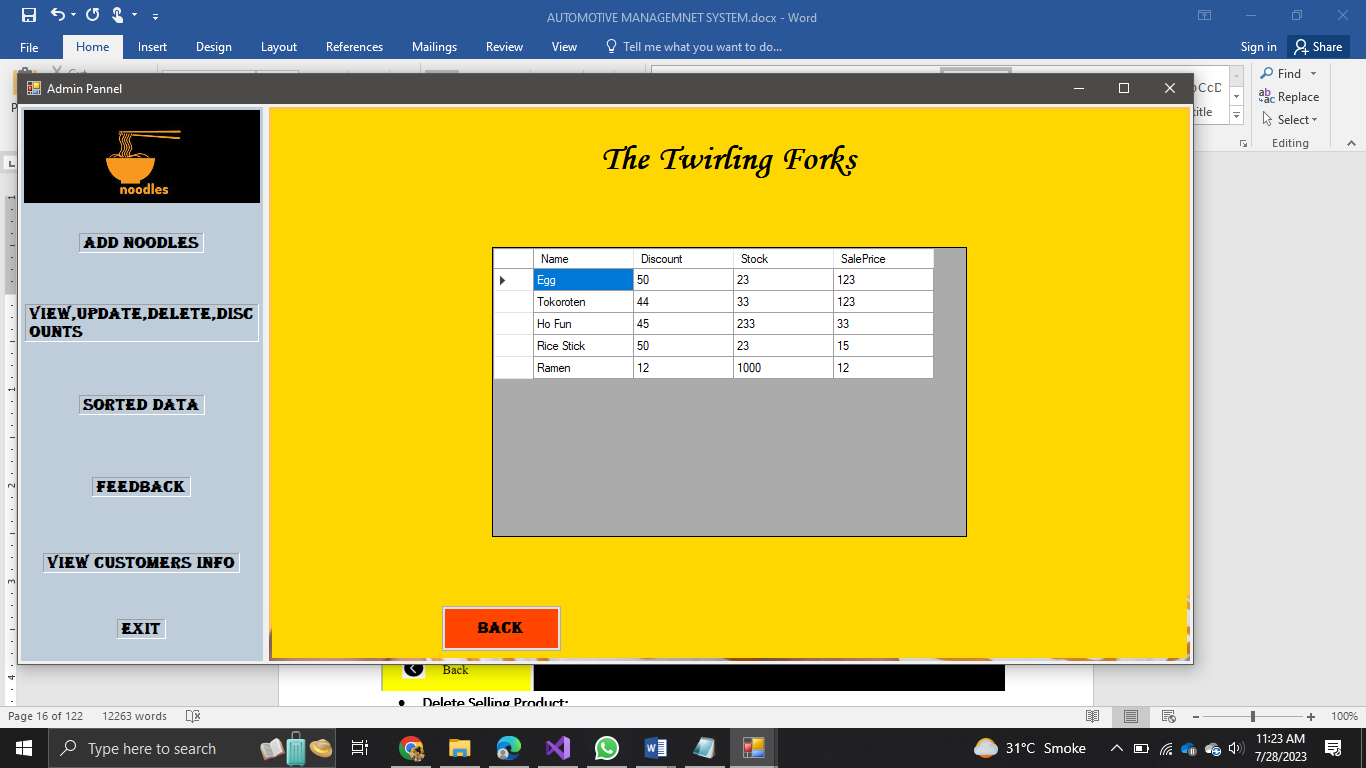
### **View Services:**



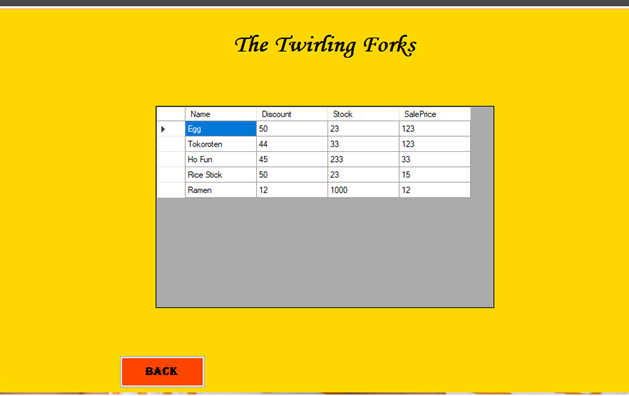
### **Noodles Menu:**

### 

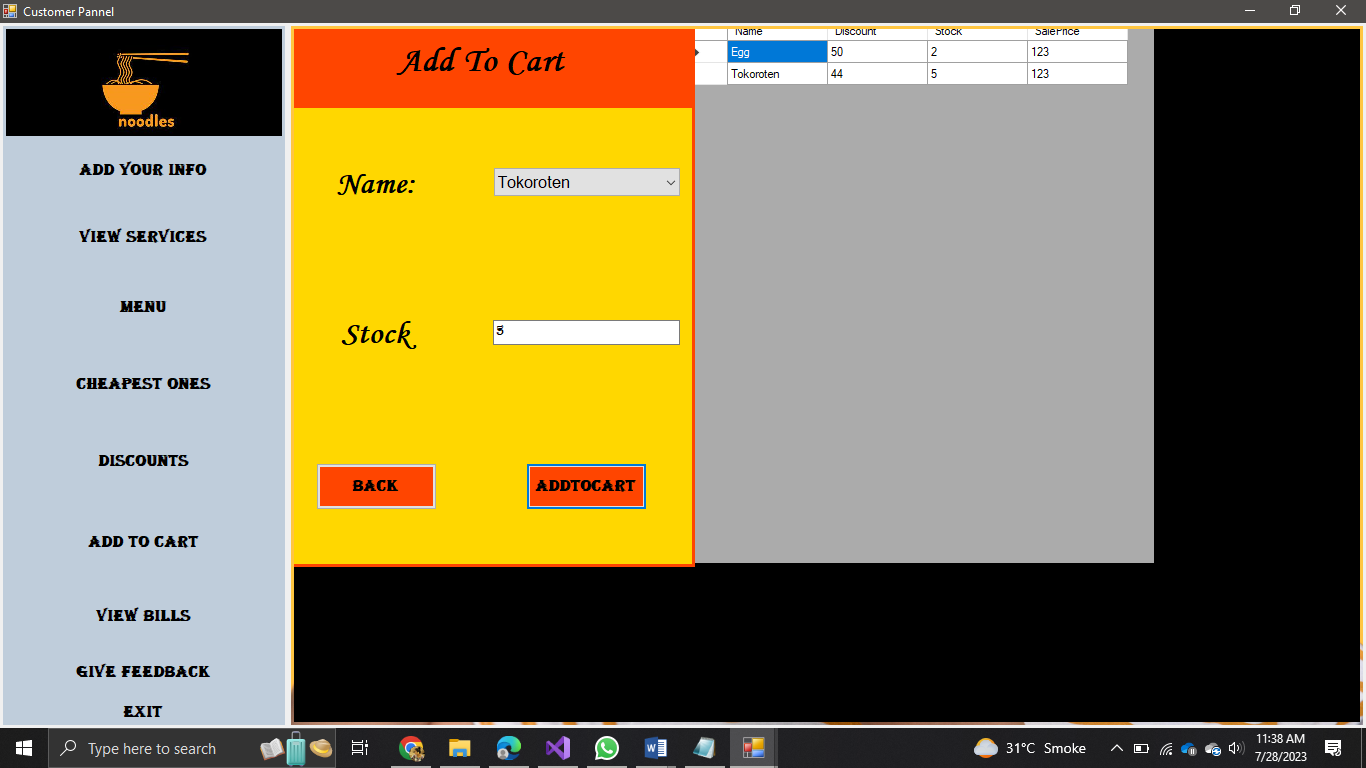
### **Expensive to cheapest ones:**

****

### **Discounts:**



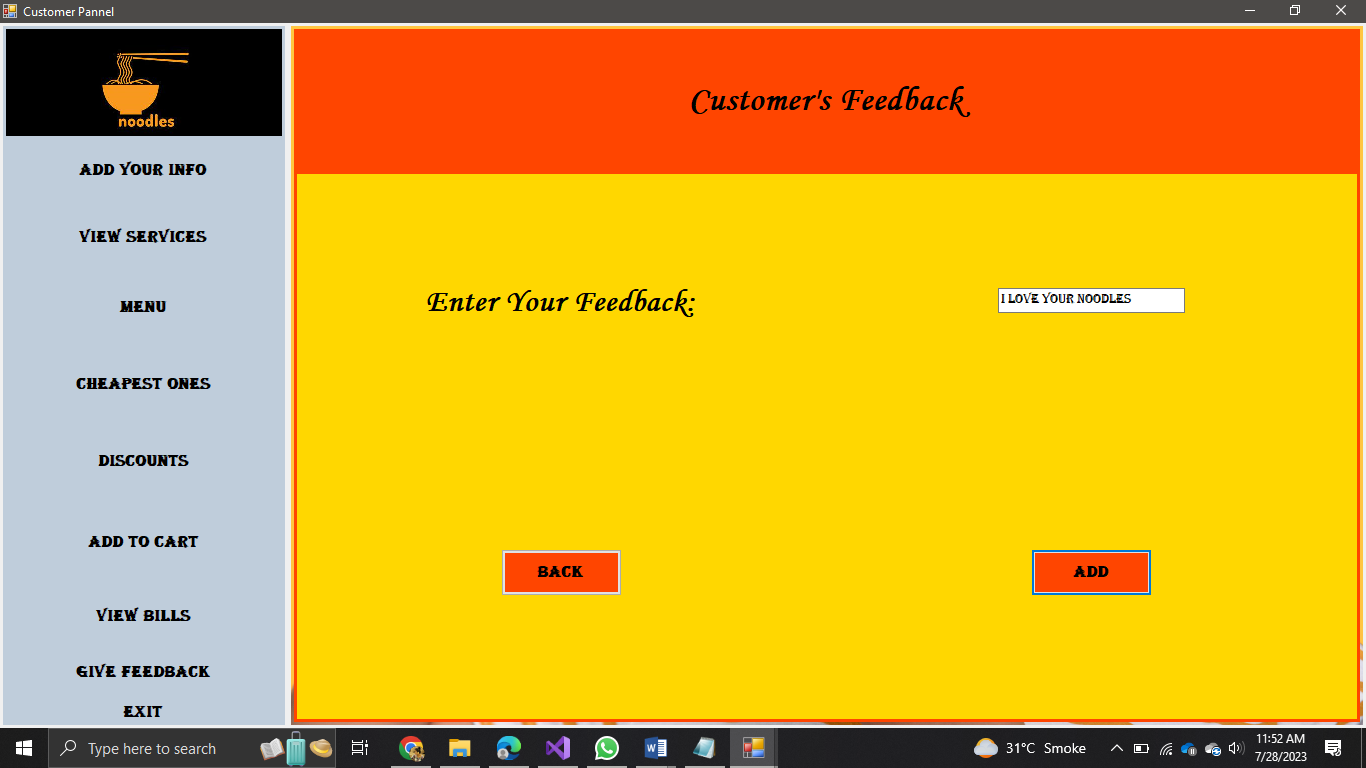
### **Add To Cart:**



### **View Bill:**

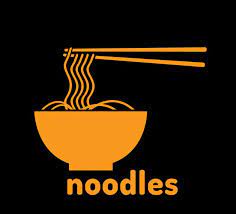


### **Give Feedback:**



## **Resources:**





## 



# **Conclusion:**

In conclusion, “The Twirlilng Forks” has proven to be a robust and efficient solution for streamlining noodle-related operations in our restaurant. Through its user-friendly interface and comprehensive features, the system has significantly improved the overall management of noodle inventory, orders, and sales.

## **Achievements:**

"The Twirling Forks" has achieved its objectives of enhancing operational efficiency, improving customer service, and providing valuable insights for better decision-making. Its implementation of object-oriented programming principles has contributed to its robustness and adaptability. The system's achievements have had a positive impact on the restaurant's overall performance and competitiveness in the market.

## **Challenges:**

In developing this software, I faced difficulty in handling the static list in data layer but I overcome this challenge by using my problem solving skill. The lesson I learned is that you have to boost your mind and focus on the main problem rather than thinking about other stuff.