

General Shop Management System



Session: 2022 – 2026

Submitted by:

Abdul Rehman 2022-CS-79

Supervised by:

Ma'am Maida Shahid

Sir Laeeq Khan Niaz

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

1.1. Overview:.....	3
1.2. Objectives:	3
1.3. Intended Functionality:	4
2. OOP-CONCEPTS:	5
2.1. Association:.....	5
2.2. Encapsulation:.....	5
2.3. Inheritance:	5
2.4. Polymorphism:	6
3. COMPARE WITH PROCEDURAL PROGRAMING:.....	6
3.1. Reusability and Code Sharing:	6
3.2. Maintainability and Scalability:	6
3.3. Code Readability and Understandability:	6
3.4. Data Security and Abstraction:	7
4. DESIGN PATTERN IMPLEMENTATION:.....	7
4.1. Business Logic (BL) Design Pattern:.....	7
4.2. Data Access Layer (DL) Design Pattern:	7
4.3. User Interface (UI) Design Pattern.....	8
5. CLASSES DETAILS:	8
5.1. Business Logic (BL) Classes:.....	9
5.2. Enum classes:	11
5.3. Data Layer (DL) classes:	12
5.4. User-Interface (UI) classes:	14
6. WireFrames:.....	19

1. INTRODUCTION

1.1. Overview:

Problem Statement: In the fast-paced and competitive retail industry, managing a general shop efficiently and effectively can be a complex undertaking. Owners often face challenges in keeping track of inventory, sales, and overall business operations. These difficulties create a demand for a comprehensive General Shop Management System that streamlines various processes and offers a centralized platform for shop owners to optimize their operations.

Solution: Our General Shop Management System is an innovative and efficient software solution designed to simplify the complexities of running a retail shop. It seeks to address the challenges faced by shop owners and employees, providing a centralized platform that optimizes inventory management, sales, customer relationships, supplier interactions, and reporting.

Significance: The introduction and adoption of our General Shop Management System holds immense significance for the retail industry and business management. By harnessing the power of technology, this system brings forth a host of advantages that can revolutionize how general shops operate, impacting various stakeholders positively.

1.2. Objectives:

Efficient Inventory Control: The main objective of the General Shop Management System is to implement efficient inventory management practices. By providing real-time updates on stock levels, automating stock replenishment, and categorizing products, the system aims to optimize inventory control. This objective helps reduce stockouts, minimize excess inventory, and ensure that the shop maintains an adequate supply of products to meet customer demand.

Streamlined Sales and Billing Processes: The system's objective is to streamline sales and billing processes for a smoother customer checkout experience. Through automated billing procedures, accurate calculations, and detailed invoicing, the system minimizes errors in transactions. This objective enhances customer satisfaction and reduces delays at the point of sale.

Enhanced Customer Relationship Management: The General Shop Management System seeks to enhance customer relationship management by maintaining a centralized database of customer information. With access to purchase history, preferences, and loyalty rewards, the system enables personalized interactions and targeted marketing efforts. This objective aims to improve customer retention and loyalty.

Data-Driven Decision Making: Another objective of the system is to enable data-driven decision-making for shop owners and managers. The reporting and analytics module generates real-time sales reports, inventory status, and customer trends. This valuable data empowers users to make informed business decisions, identify trends, and optimize strategies for improved profitability and growth.

1.3. Intended Functionality:

User-Friendly Interface: The General Shop Management System will feature a user-friendly interface that is easy to navigate and operate. It will be designed with simplicity in mind, ensuring that both shop owners and employees can quickly adapt to the system without the need for extensive training.

Real-Time Inventory Tracking: The system will offer real-time tracking of inventory levels. Users receive alerts and current stock status of products, receive low-stock alerts, and check the availability of specific items instantly. This functionality ensures that shops can promptly restock products and avoid stockouts.

Sales and Billing Automation: The system will automate the sales and billing processes to streamline transactions. It will calculate accurate totals, generate detailed invoices, and process various payment methods seamlessly, reducing manual errors and facilitating faster checkout.

Customer Relationship Management: The system will enable efficient customer relationship management by maintaining a centralized database of customer information. Users can access customer profiles, view purchase history, and track customer preferences to offer personalized services and promotions.

Supplier Management: The system will assist in managing supplier relationships and orders. It will store supplier contact details, track purchase orders, and monitor delivery statuses. This functionality ensures timely product availability and fosters better supplier communication.

Reporting and Analytics: The system will provide comprehensive reporting and analytics tools. Shop owners can generate detailed sales reports, analyze inventory trends,

and gain insights into customer behavior. These data-driven insights will aid in making informed business decisions and optimizing shop operations.

2. OOP-CONCEPTS:

In the context of the General Shop Management System, object-oriented programming (OOP) principles and concepts can be applied to enhance the design, organization, and functionality of the system. Here are some examples of how OOP concepts can be associated with various aspects of the system:

2.1. Association:

Association is a fundamental OOP concept that represents relationships between classes. In the General Shop Management System, association can be observed between different classes that interact with each other. For instance, there can be an association between the Inventory class and the Sales class, where the Sales class accesses the Inventory class to update product quantities after each sale. Similarly, there can be an association between the Customer class and the Purchase class, where the Purchase class refers to the Customer class to store customer details for each transaction.

2.2. Encapsulation:

Encapsulation involves bundling data and methods within a class, shielding the internal implementation details from outside access. In the General Shop Management System, encapsulation can be utilized to protect sensitive data and provide controlled access to class members. For example, the Employee class might encapsulate certain methods and data related to employee management, ensuring that only authorized components can modify or access employee information.

2.3. Inheritance:

Inheritance is a key OOP concept that enables the creation of hierarchical relationships between classes. In the General Shop Management System, inheritance can be applied to establish relationships between various entities. For instance, there can be a base class named Product, which is inherited by more specific classes like Electronics, Clothing, or Groceries. The specific classes inherit the properties and methods of the Product class while adding their unique attributes and behaviors.

2.4. Polymorphism:

Polymorphism is a powerful OOP concept that allows objects of different classes to be treated interchangeably. In the General Shop Management System, polymorphism can be employed to handle various types of products uniformly. For example, a method for calculating the total price of items in the shopping cart can accept objects of different product types, such as Electronics or Clothing, as long as they are derived from a common base class or implement a common interface.

3. COMPARE WITH PROCEDURAL PROGRAMING:

Following are the various aspects in which OOP is better than procedural programming:

3.1. Reusability and Code Sharing:

OOP promotes code reusability through the concept of inheritance. Inheritance allows classes to inherit properties and behaviors from other classes, reducing the need to rewrite code. This not only saves development time but also makes the codebase more efficient and easier to maintain. In procedural programming, reusing code requires copying and pasting or creating separate functions, which can lead to code redundancy and maintenance issues.

3.2. Maintainability and Scalability:

OOP promotes code maintainability and scalability. With encapsulation and modularity, making changes to a specific functionality or fixing issues becomes easier because the affected code is localized within the relevant class. Additionally, OOP's ability to extend existing classes through inheritance allows for the addition of new features without modifying the existing codebase extensively. In procedural programming, making changes or adding features often involves modifying multiple functions, increasing the likelihood of introducing errors and making maintenance more challenging.

3.3. Code Readability and Understandability:

OOP encourages a more natural representation of real-world entities and their relationships. This makes the codebase more readable and understandable, as classes and objects closely resemble the entities and interactions they model. Procedural programming, on the other hand, can lack this intuitive representation, making it harder to grasp the overall system design and the relationships between various parts of the code.

3.4. Data Security and Abstraction:

OOP allows for the implementation of data security through the concept of encapsulation. By hiding the internal implementation details of a class, OOP ensures that data can only be accessed and modified through designated methods, reducing the risk of unintended modifications or data corruption. Procedural programming typically lacks built-in mechanisms for data security and abstraction, making it more prone to data integrity issues and unauthorized access.

4. DESIGN PATTERN IMPLEMENTATION:

In the general shop management system, the utilization of design patterns helps ensure modularity, separation of concerns, and maintainability. Here is how the project incorporates the Business Logic (BL), Data Access Layer (DL), and User Interface (UI) design patterns:

4.1. Business Logic (BL) Design Pattern:

The BL design pattern focuses on encapsulating the business rules and logic of the application. It ensures that the core functionality and operations of the system are independent of the underlying data storage or user interface.

To implement the **BL design** pattern in the **General Shop Management System**, we can utilize the **Model-View-Controller (MVC)** architectural pattern. The "Model" component represents the business logic and operations of the system. It encapsulates functionalities such as inventory management, order processing, and customer management. By adopting the **MVC** pattern, the **General Shop Management System** achieves a clear separation of concerns between the business logic and the user interface. This separation enhances code maintainability, modularity, and testability. Developers can modify or update specific components without affecting the overall functionality of the system, promoting code reusability and scalability.

4.2. Data Access Layer (DL) Design Pattern:

The DL design pattern focuses on managing the interaction between the application and the underlying data storage or database. It abstracts the data access operations, ensuring that the business logic is decoupled from the specifics of the data storage implementation.

To implement the **DL design** pattern in the **General Shop Management System**, we can adopt the **Repository** pattern. Repositories act as intermediaries between the business logic and the database. They handle data retrieval, storage, and update operations

while abstracting the complexities of data access. By employing the Repository pattern, the General Shop Management System gains several benefits. Firstly, it promotes a clear separation between the business logic and data storage, which enhances code maintainability and reduces dependencies. Secondly, repositories encapsulate the details of data access, such as querying the database, mapping data to objects, and ensuring data integrity and consistency. With this design pattern in place, the General Shop Management System can switch or upgrade the underlying database without affecting the overall functionality of the application. The separation of concerns achieved by the Repository pattern allows developers to work on data access and business logic independently, streamlining the development process and enabling the system to scale effectively with changing requirements.

4.3. User Interface (UI) Design Pattern

The UI design pattern focuses on structuring and organizing the user interface components to ensure a consistent and user-friendly experience. It separates the presentation layer from the underlying business logic and data access operations.

In the General Shop management system: The UI design pattern can be implemented using the **Model-View-ViewModel(MVVM)** architectural pattern. The model represents the data and business logic, while the view model acts as an intermediary between the model and the view. The view model exposes properties and commands that the view can bind to, facilitating data binding and handling user actions. The view represents the visual components of the user interface, displaying the data from the view model and capturing user input. By utilizing the UI design pattern, the blood donation management system separates the UI concerns from the underlying business logic and data access operations. This separation allows for easier modification, testing, and customization of the user interface, without impacting the core functionality of the system.

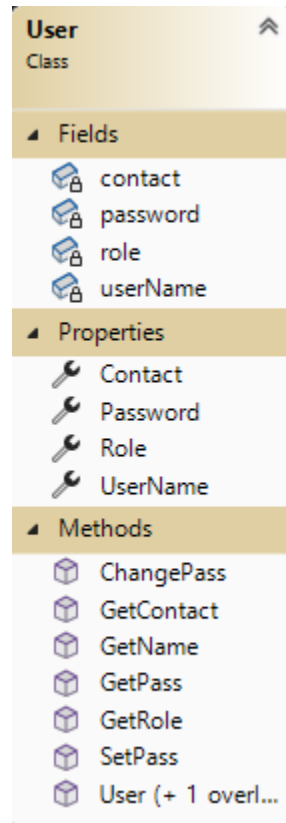
5. CLASSES DETAILS:

Explanations of classes in the general shop management system along with class diagram to illustrate their responsibilities:

5.1. Business Logic (BL) Classes:

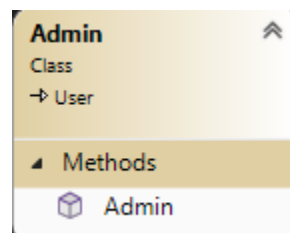
- **User:**

This is the main class for login to the application. It is the parent class for three users i.e., admin, employee, and customer.



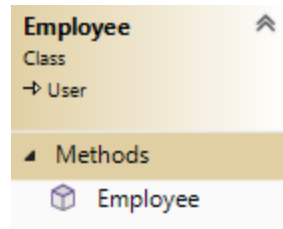
- **Admin:**

Admin is the child class of user, and it is responsible for employee's operation and for adding, removing products and sale.



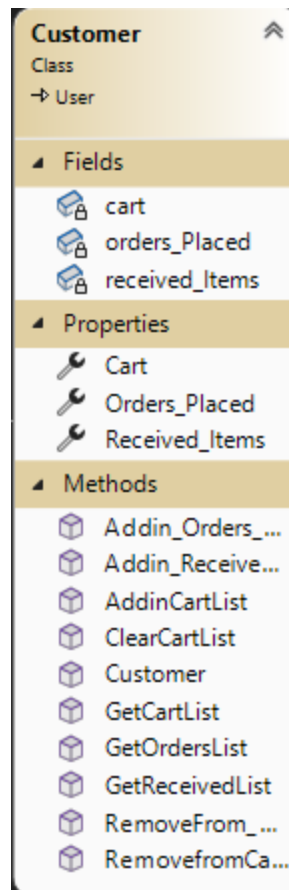
- **Employee:**

Employee is the child class of the User and is responsible for completing the orders and informing Admin about low stock and in demand products.



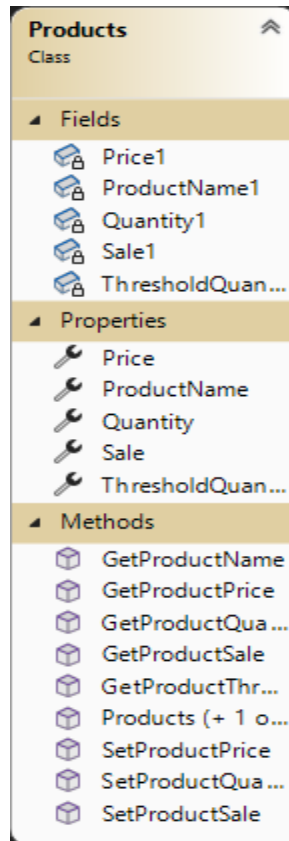
- **Customer:**

The customer is also the child class of the User and is responsible for purchasing items from the store.



- **Products:**

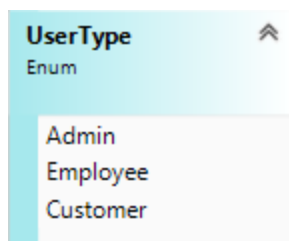
This class is responsible for keeping the record of products like their name, price, quantity, etc.



5.2. **Enum classes:**

- **UserType:**

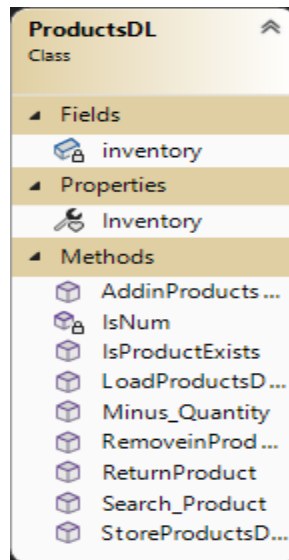
This class is responsible for describing the type of user like Admin, Employee and Customer.



5.3. Data Layer (DL) classes:

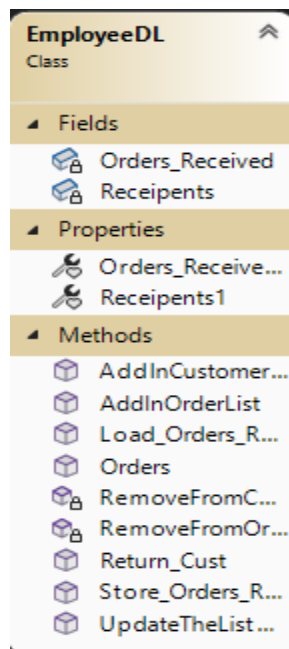
- **ProductsDL:**

This class is responsible for products data handling. It stores the data in a file whenever a product is added or removed from the management system.



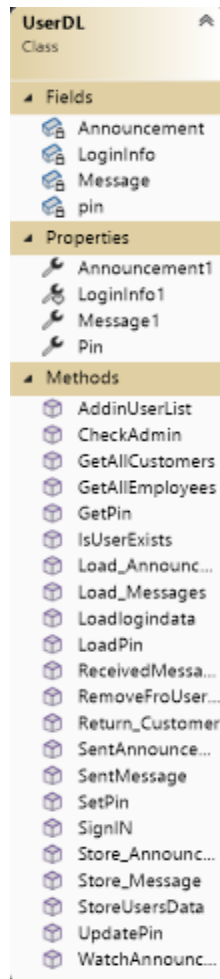
- **EmployeeDL;**

This class is responsible for keeping the orders data and products that are in low quantity in a file.



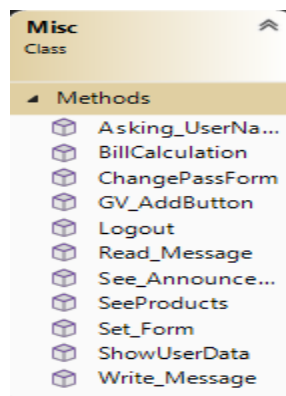
- **UserDL:**

This class is responsible for managing the data of the users. Adding new data and updating the previous data and keep all the data in the file.



- **Miscellaneous:**

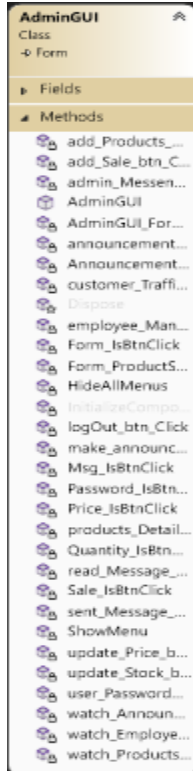
This class is used for general work.



5.4. User-Interface (UI) classes:

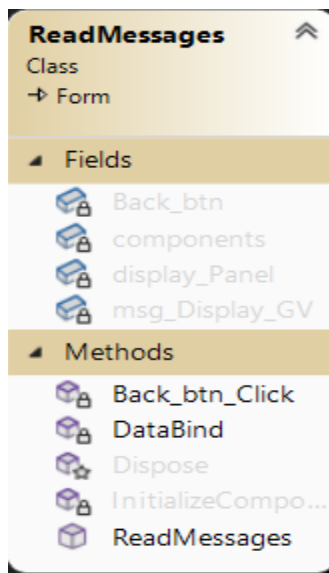
- **AdminGUI:**

This class is used to represent the user interface for Admin.



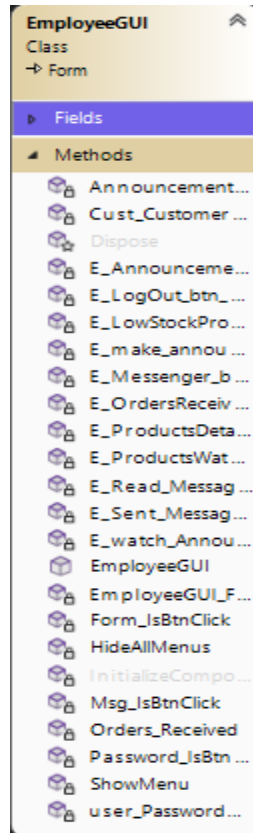
- **Read Message GUI:**

This class is used to show messages to the User.



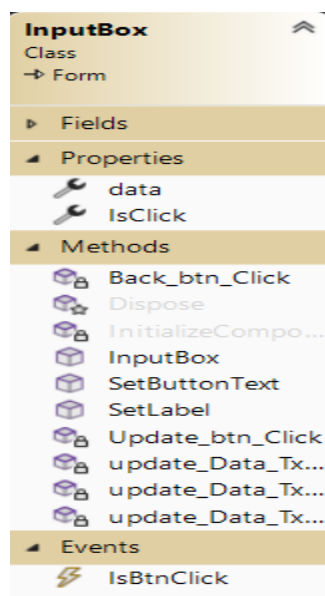
- **EmployeeGUI:**

This class is used to represent the employee interface.



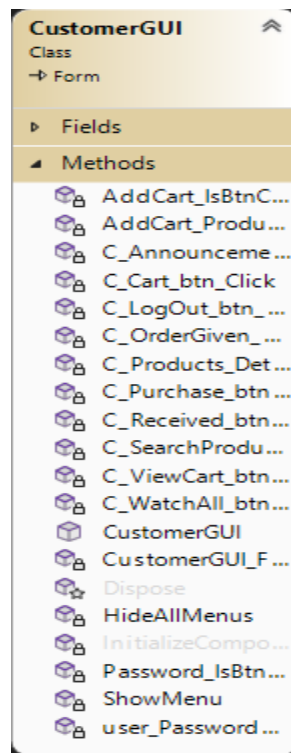
- **InputBox GUI:**

This class is used to take input from the user.



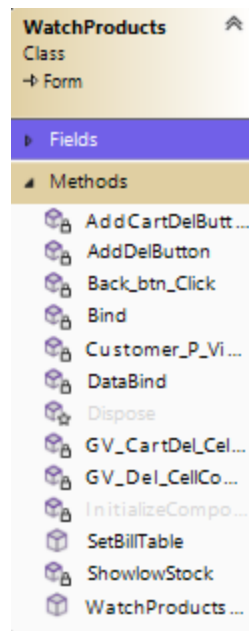
- **Customer GUI:**

This class is used to represent the customer interface.



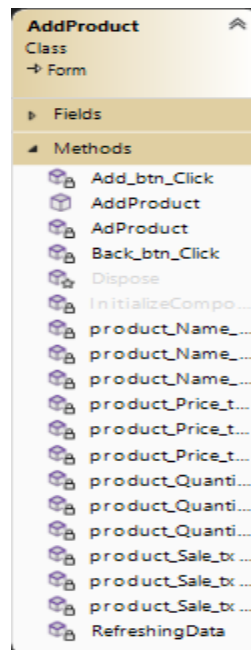
- **Watch Products GUI:**

This class is used to show the products data to the user.



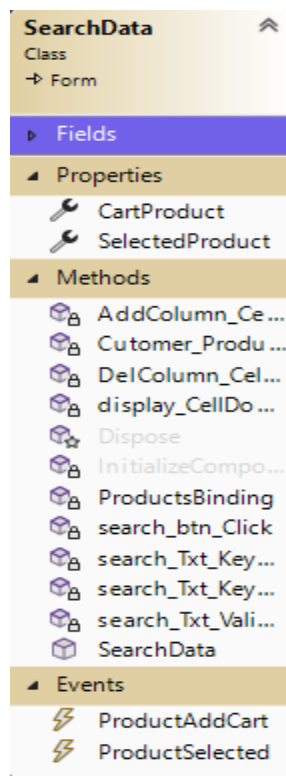
- **Add Products GUI:**

This class is used to add products in the system.



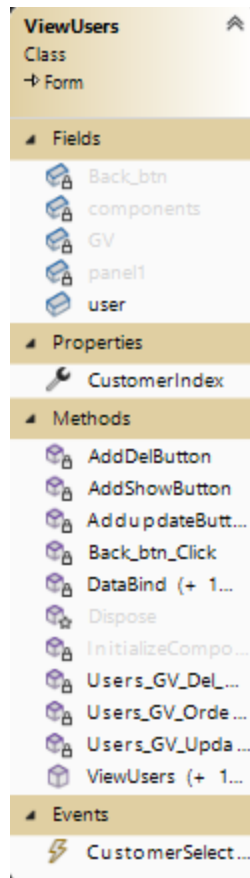
- **Search Data GUI:**

This class is used to search for specific data.

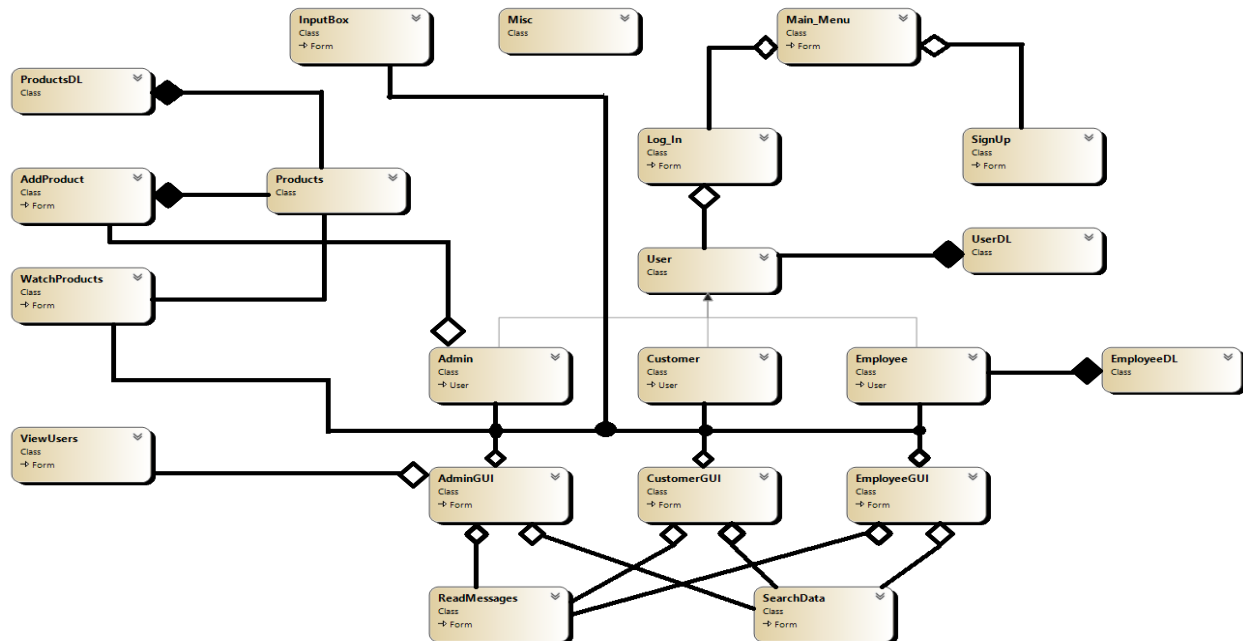


- **View Users GUI:**

This class is used to show users.



6. Class Diagram:



7. Wireframes of the Game:

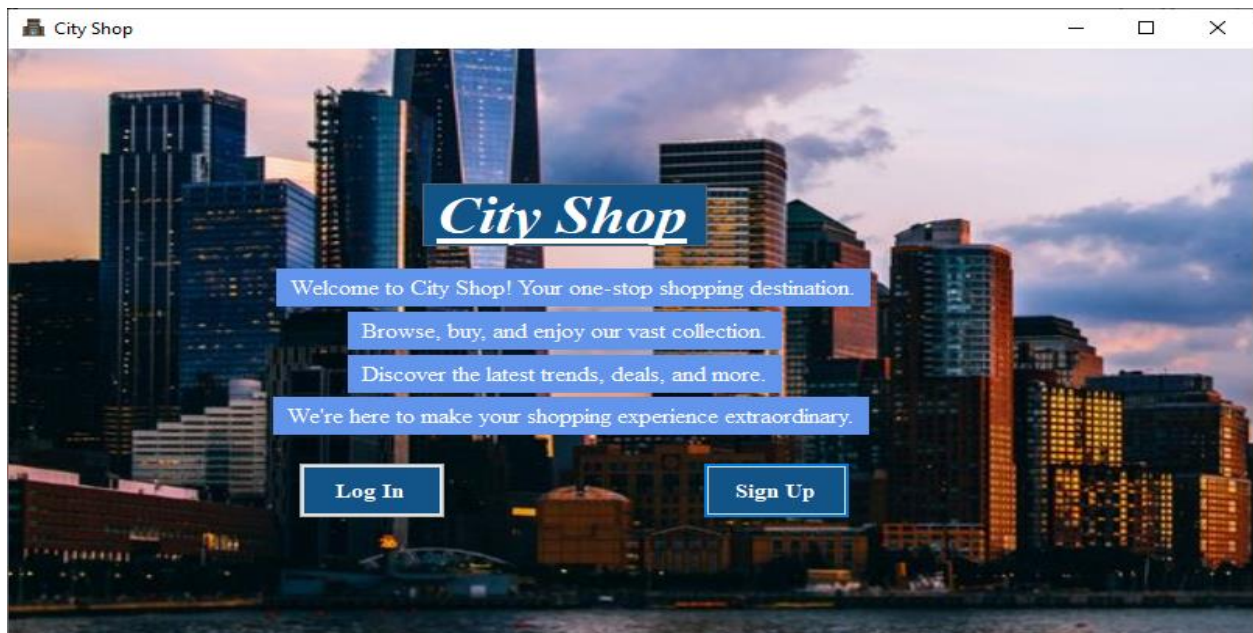
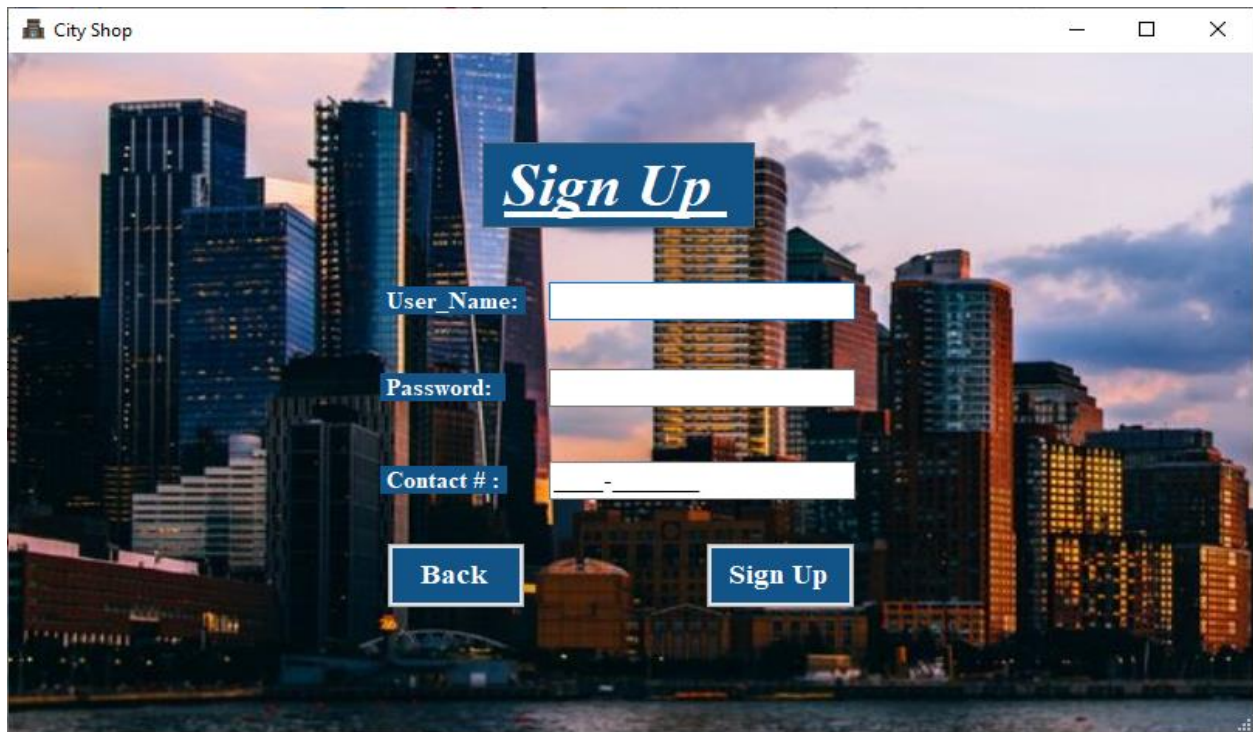
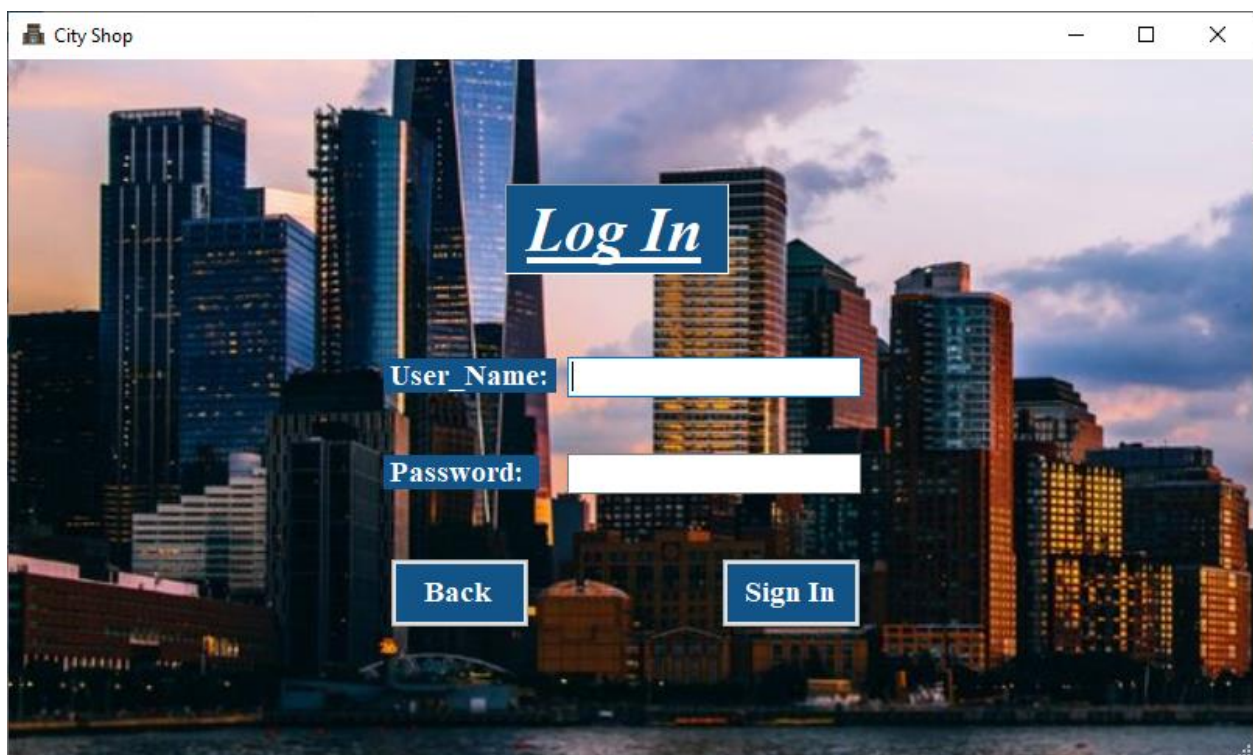


Fig.: Main Screen



The image shows a web application window titled "City Shop" with a city skyline background. The "Sign Up" screen features a large blue button with the text "Sign Up" in white, underlined, and italicized. Below this are three input fields: "User_Name:", "Password:", and "Contact # :". At the bottom, there are two blue buttons: "Back" and "Sign Up".

Fig.: Sign-Up Screen



The image shows a web application window titled "City Shop" with a city skyline background. The "Log In" screen features a large blue button with the text "Log In" in white, underlined, and italicized. Below this are two input fields: "User_Name:" and "Password:". At the bottom, there are two blue buttons: "Back" and "Sign In".

Fig.: Log In Screen

General Store Management System

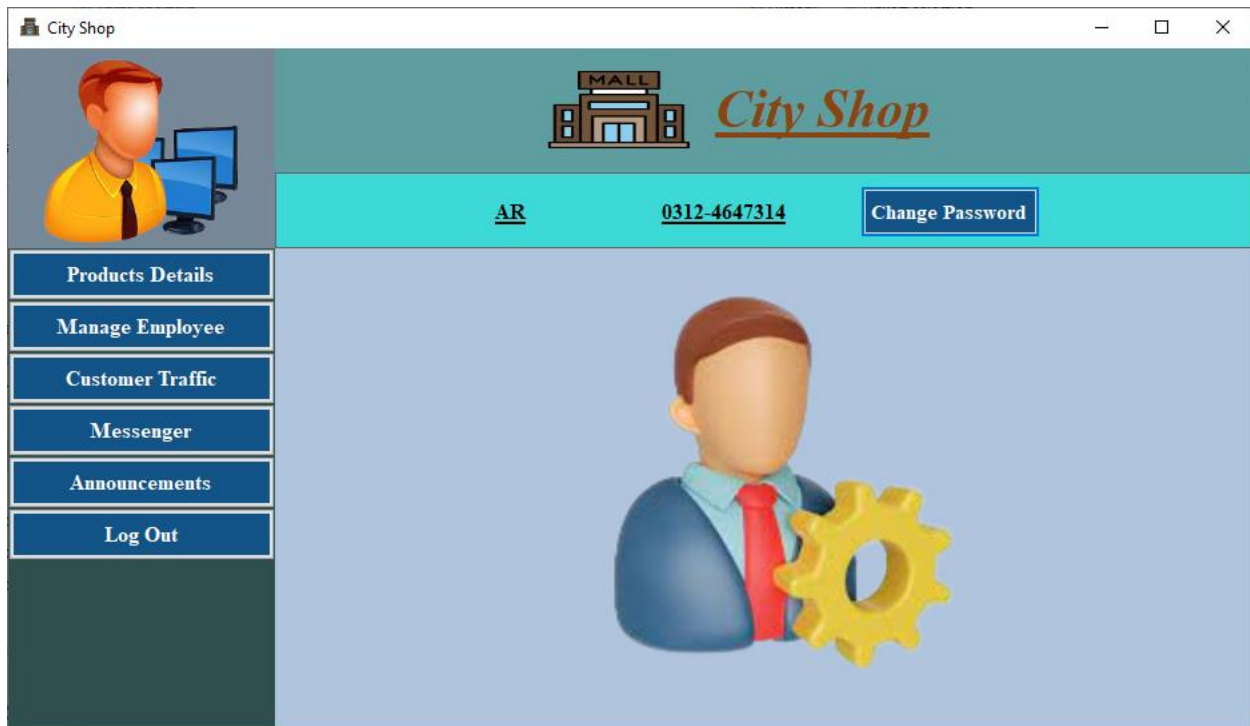


Fig.: Admin Interface

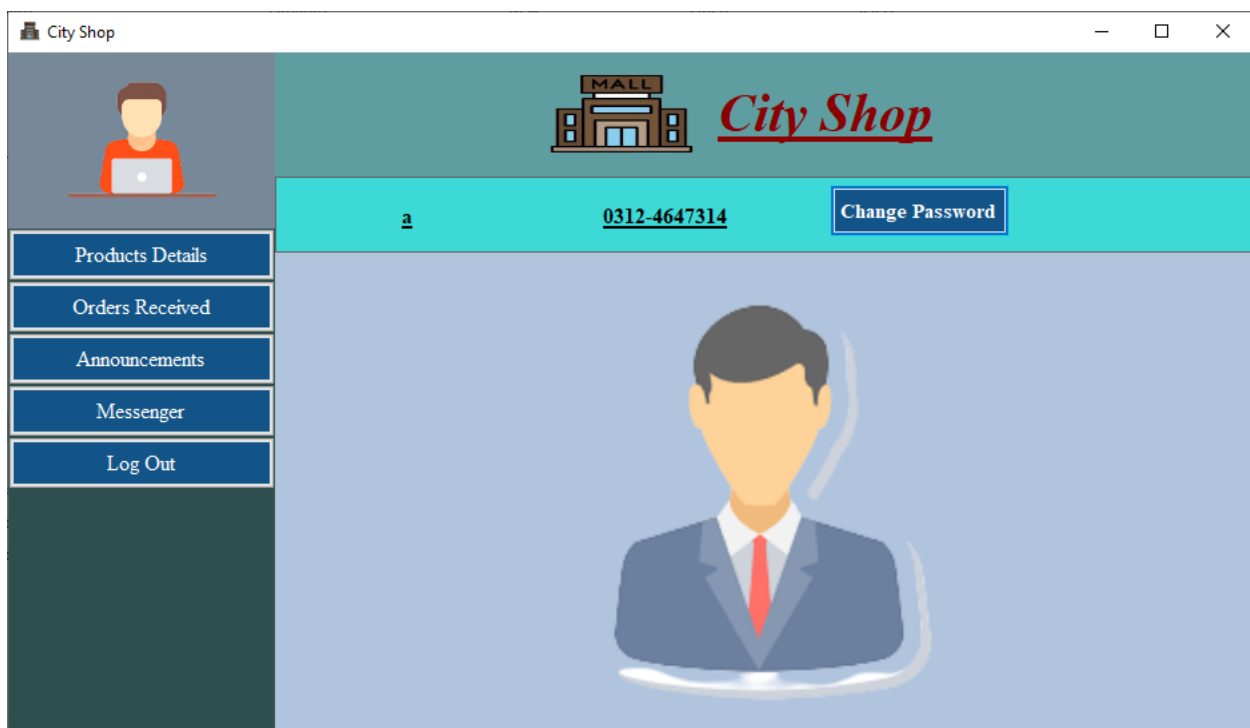


Fig.: Employee Interface

General Store Management System

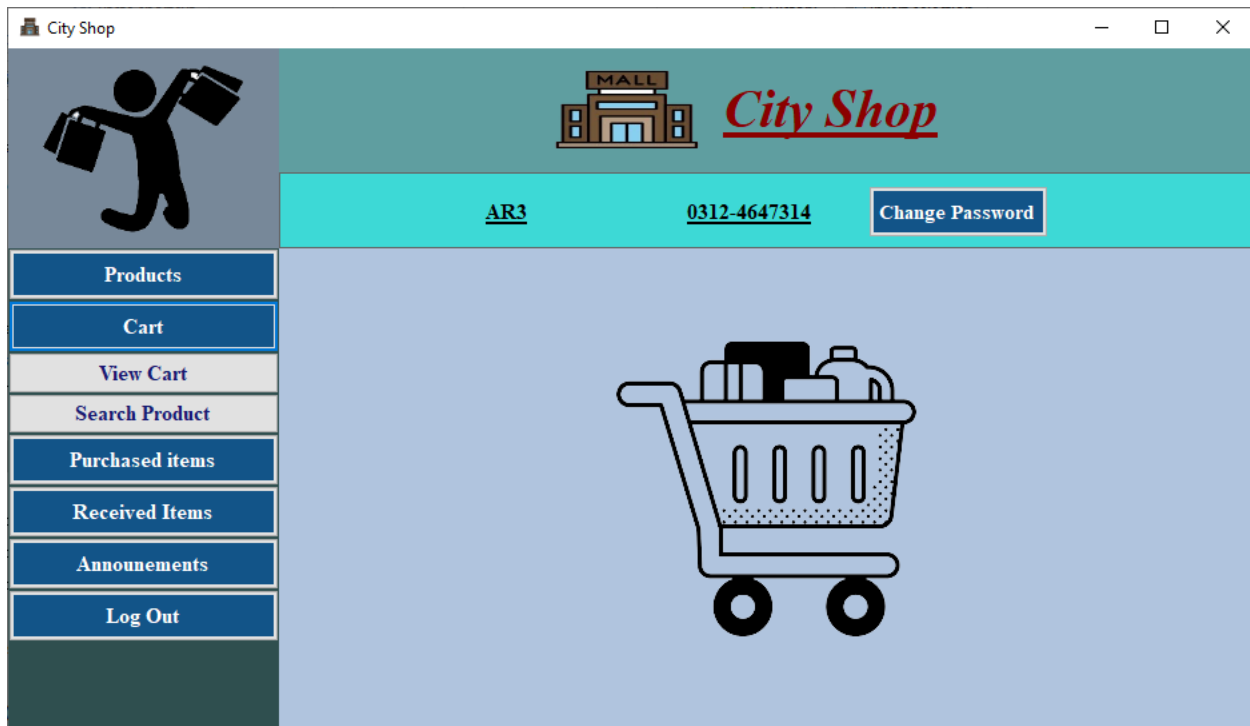


Fig.: Customer Interface

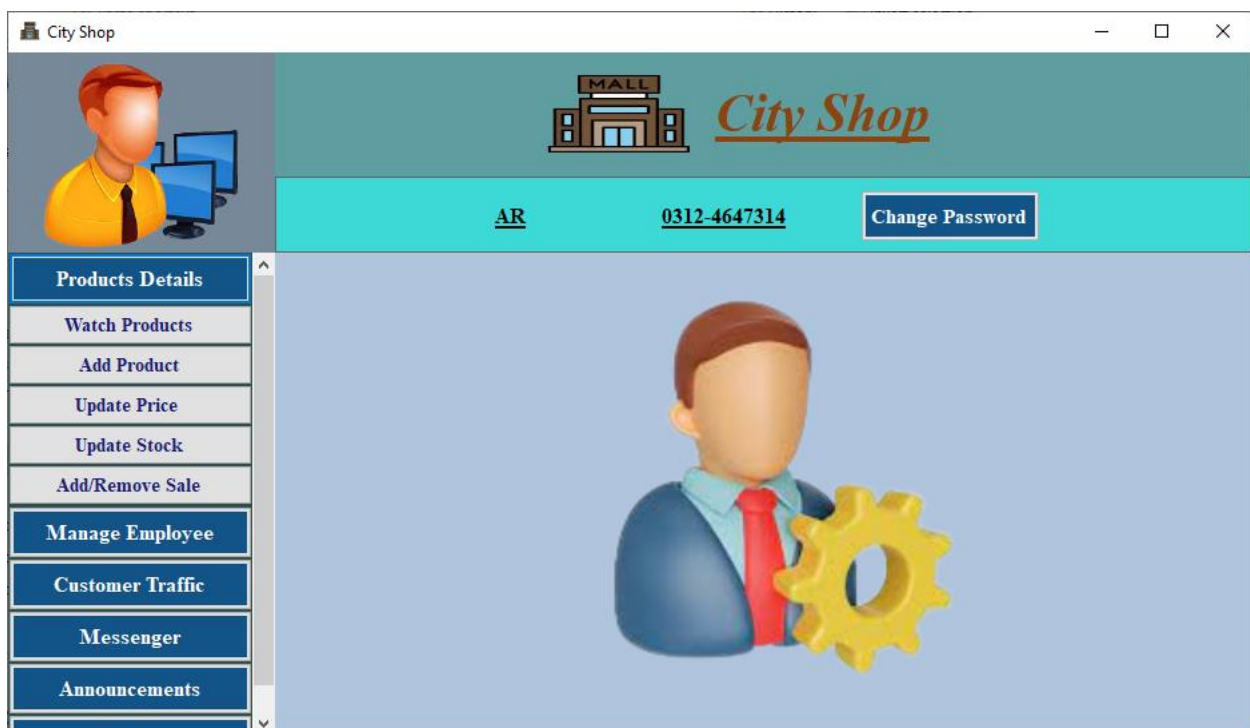





Fig.: Users Options

General Store Management System



City Shop



**City Shop**

AR 0312-4647314 [Change Password](#)

[Products Details](#)
[Manage Employee](#)
[Customer Traffic](#)
[Messenger](#)
[Announcements](#)
[Log Out](#)

Product Name:


Product Price:

Product Quantity:


Product Sale:


[Back](#) [Add](#)

Fig.: Add Product Option



City Shop



**City Shop**

AR 0312-4647314 [Change Password](#)

[Products Details](#)
[Manage Employee](#)
[Customer Traffic](#)
[Messenger](#)
[Announcements](#)
[Log Out](#)

	ProductName	Price	Quantity	Sale	Delete
▶	Cream	30	1000	15	Delete
	Tiger	20	1000	25	Delete

[Back](#)

Fig.: Watch Products Option

General Store Management System

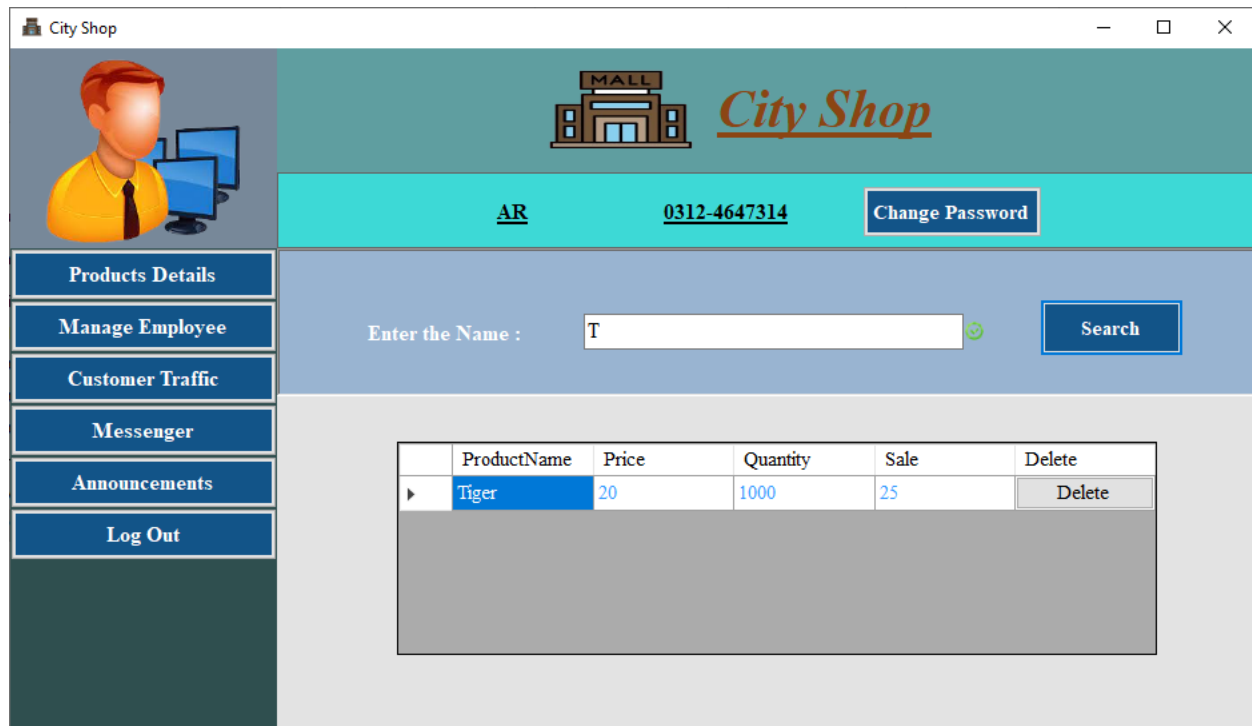


Fig.: Products Search Option

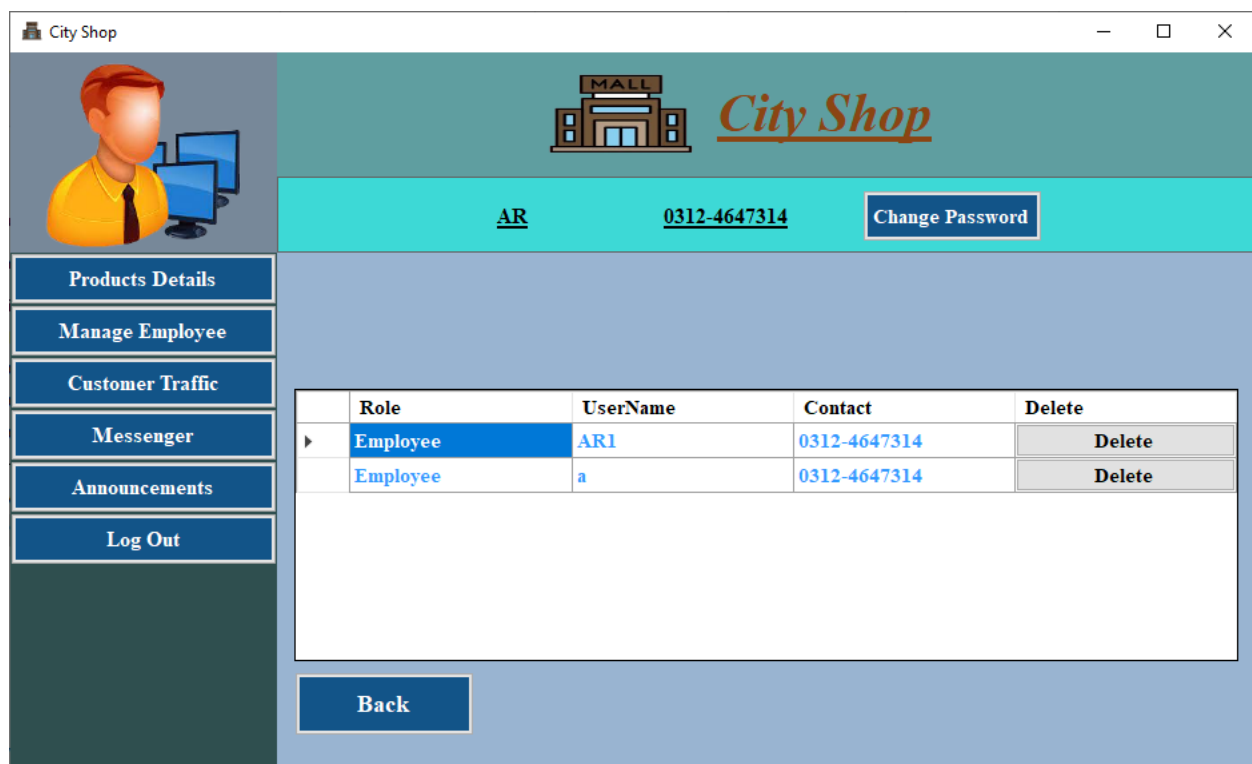


Fig.: Watch Employee Option

General Store Management System

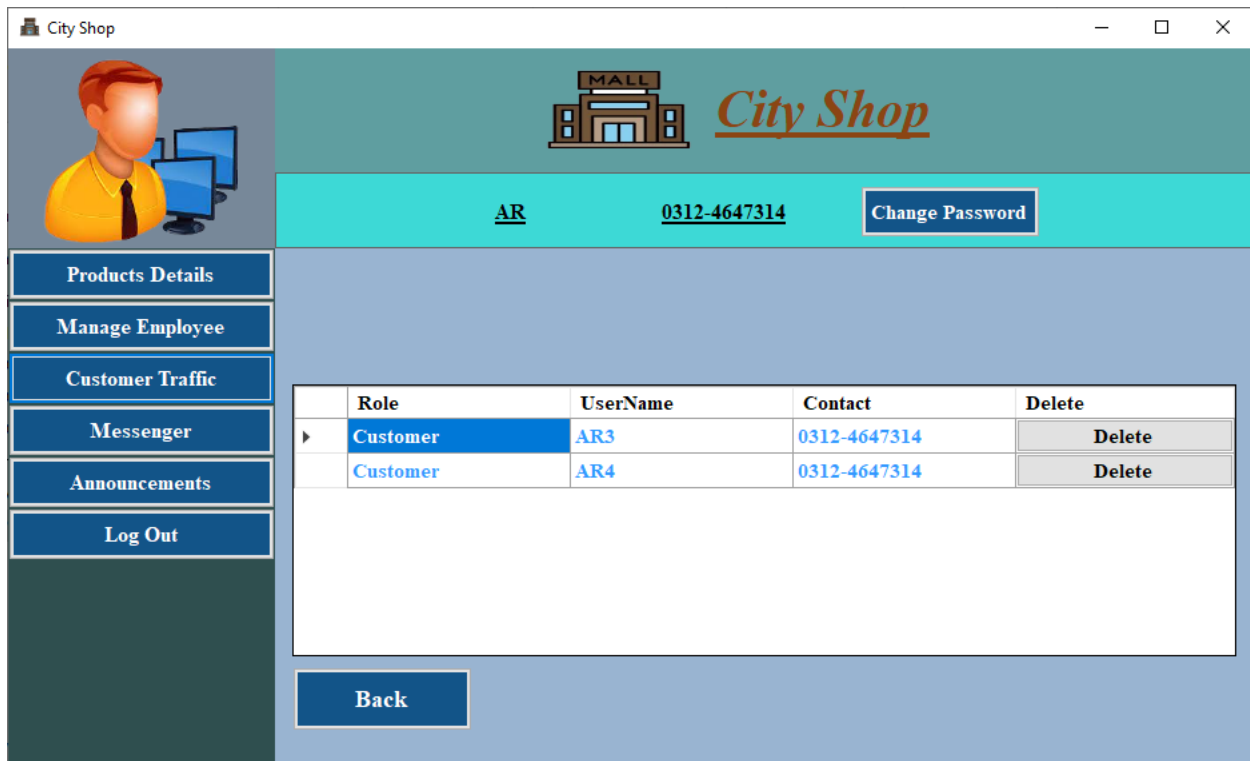


Fig.: Watch Customers Option

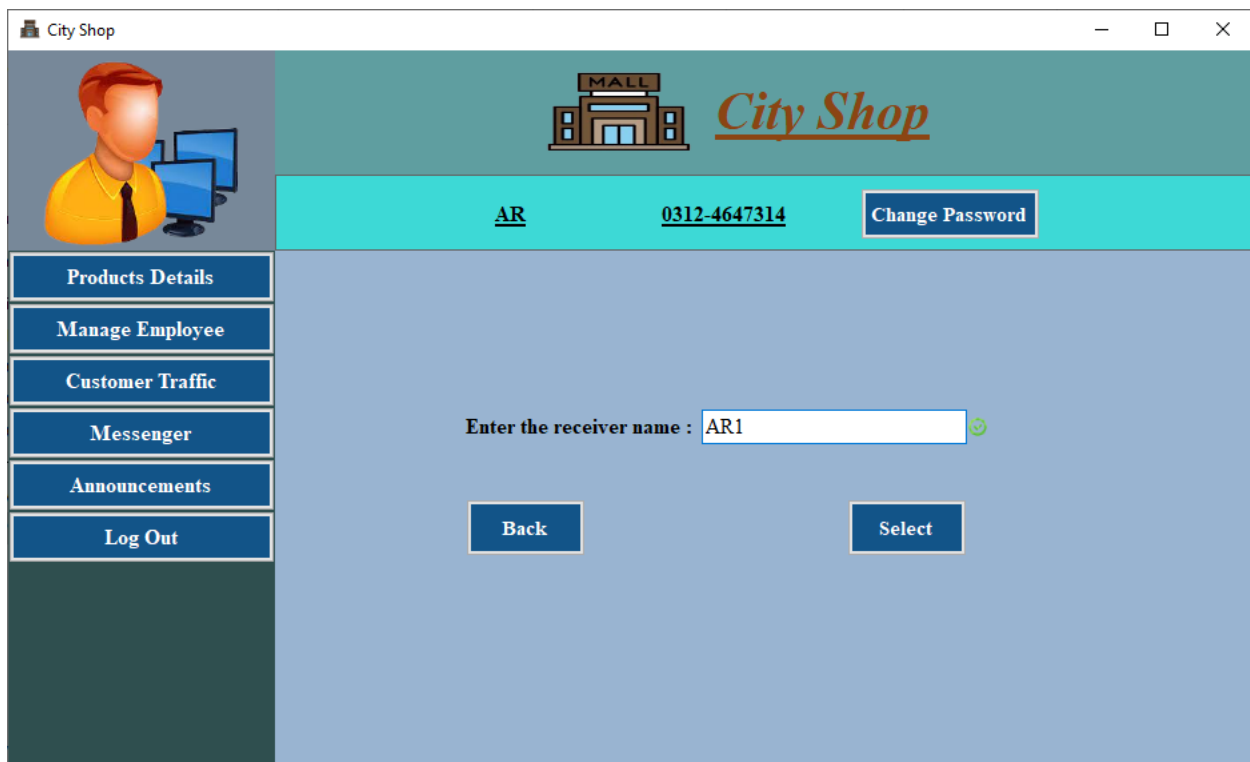


Fig.: Messenger asking name Screen

General Store Management System

The screenshot shows the 'City Shop' application window. On the left is a sidebar with a user profile icon and a menu containing: Products Details, Manage Employee, Customer Traffic, Messenger, Announcements, and Log Out. The main header area displays a mall icon, the 'City Shop' logo, and a cyan bar with the text 'AR' and '0312-4647314', along with a 'Change Password' button. The main content area is titled 'Enter your message :' and features a text input field. Below the input field are two buttons: 'Back' and 'Send'.

Fig.: Sent Message/Announcement Screen

This screenshot displays the 'City Shop' application with the same sidebar and header as the previous screen. The main content area shows a list of messages. The first message is from 'AR1' with the text '==>>Hi ar1'. The second message is from 'a' with the text '==>>HI ab13'. Below the message list is a 'Back' button.

Fig.: Watch Message Screen

8. Complete Code:

8.1. Business Logic (BL) classes:

- Users:

```
public class User
{
    private UserType role;
    private string userName;
    private string password;
    private string contact;

    public UserType Role { get => role; set => role = value; }
    public string UserName { get => userName; set => userName = value; }
    public string Password { get => password; set => password = value; }
    public string Contact { get => contact; set => contact = value; }

    public User(string userName, string password)
    {
        this.UserName = userName;
        this.Password = password;
    }

    public User(UserType role, string userName, string password, string contact)
    {
        this.Role = role;
        this.UserName = userName;
        this.Password = password;
        this.Contact = contact;
    }

    public string GetName()
    {
        return this.UserName;
    }

    public string GetPass()
    {
        return this.Password;
    }

    public UserType GetRole()
    {
        return this.Role;
    }

    public void SetPass(string password)
    {
        this.Password = password;
    }
}
```

```
public string GetContact()
{
    return this.Contact;
}

public void ChangePass(string pass)
{
    SetPass(pass);
    UserDL.StoreUsersData();
}
}
```

- **Admin:**

```
public class Admin : User
{
    public Admin(UserType role, string userName, string password, string
contact) : base(role, userName, password, contact)
    {
    }
}
}
```

- **Employee:**

```
public class Employee : User
{
    public Employee(UserType role, string userName, string password, string
contact) : base(role, userName, password, contact)
    {
    }
}
}
```

- **Customer:**

```
public class Customer : User
{
    private List<Products> cart = new List<Products>();
    private List<Products> orders_Placed = new List<Products>();
    private List<Products> received_Items = new List<Products>();

    public List<Products> Cart { get => cart; set => cart = value; }
    public List<Products> Orders_Placed { get => orders_Placed; set =>
orders_Placed = value; }
    public List<Products> Received_Items { get => received_Items; set =>
received_Items = value; }

    public Customer(UserType role, string userName, string password, string
contact) : base(role, userName, password, contact)
}
```

General Store Management System

```
{  
  
    public void AddinCartList(Products p)  
    {  
        cart.Add(p);  
    }  
  
    public void ClearCartList()  
    {  
        cart.Clear();  
    }  
  
    public void RemovefromCartList(Products p)  
    {  
        cart.Remove(p);  
    }  
  
    public void RemoveFrom_Orders_Placed_List()  
    {  
        orders_Placed.Clear();  
    }  
  
    public void Addin_Orders_placed_List(List<Products> list)  
    {  
        orders_Placed = list;  
    }  
  
    public void Addin_Received_List(List<Products> list)  
    {  
        foreach (var item in list)  
        {  
            received_Items.Add(item);  
        }  
    }  
  
    public List<Products> GetCartList()  
    {  
        return cart;  
    }  
  
    public List<Products> GetOrdersList()  
    {  
        return orders_Placed;  
    }  
  
    public List<Products> GetReceivedList()  
    {  
        return received_Items;  
    }  
}
```

- **Products:**

```
public class Products  
{
```

General Store Management System

```
private string ProductName1;
private float Price1;
private int Quantity1;
private int ThresholdQuantity1;
private int Sale1;

public string ProductName { get => ProductName1; set => ProductName1 =
value; }
public float Price { get => Price1; set => Price1 = value; }
public int Quantity { get => Quantity1; set => Quantity1 = value; }
public int ThresholdQuantity { get => ThresholdQuantity1; set =>
ThresholdQuantity1 = value; }
public int Sale { get => Sale1; set => Sale1 = value; }

public Products(string ProductName, float Price, int Quantity, int Sale)
{
    this.ProductName1 = ProductName;
    this.Price1 = Price;
    this.Quantity1 = Quantity;
    this.Sale1 = Sale;
}

public Products(string ProductName, float Price, int Quantity, int
ThresholdQuantity, int Sale)
{
    this.ProductName1 = ProductName;
    this.Price1 = Price;
    this.Quantity1 = Quantity;
    this.ThresholdQuantity1 = ThresholdQuantity;
    this.Sale1 = Sale;
}

public string GetProductName()
{
    return this.ProductName1;
}

public float GetProductPrice()
{
    return this.Price1;
}

public void SetProductPrice(float Price)
{
    this.Price1 = Price;
    ProductsDL.StoreProductsData();
}

public int GetProductQuantity()
{
    return this.Quantity1;
}

public void SetProductQuantity(int Quantity)
{
    this.Quantity1 = Quantity;
}
```

```
        ProductsDL.StoreProductsData();
    }

    public int GetProductThreshold()
    {
        return this.ThresholdQuantity1;
    }

    public int GetProductSale()
    {
        return this.Sale1;
    }

    public void SetProductSale(int Sale)
    {
        this.Sale1 = Sale;
        ProductsDL.StoreProductsData();
    }
}
```

8.2. Enums Classes

- **UserType:**

```
public enum UserType
{
    Admin,
    Employee,
    Customer
}
```

8.3. Data Layer (DL) Classes:

- **UserDL:**

```
class UserDL
{
    private static string pin = "";
    private static List<User> LoginInfo = new List<User>();
    private static List<List<string>> Announcement = new List<List<string>>();
    private static List<List<string>> Message = new List<List<string>>();

    public static string Pin { get => pin; set => pin = value; }
    internal static List<User> LoginInfo1 { get => LoginInfo; set => LoginInfo = value; }
    public static List<List<string>> Announcement1 { get => Announcement; set => Announcement = value; }
}
```

General Store Management System

```
public static List<List<string>> Message1 { get => Message; set => Message =
value; }

public static List<Employee> GetAllEmployees()
{
    List<Employee> employees = new List<Employee>();
    foreach (var x in LoginInfo)
    {
        if (x is Employee person)
        {
            employees.Add(person);
        }
    }
    return employees;
}

public static List<Customer> GetAllCustomers()
{
    List<Customer> customers = new List<Customer>();
    foreach (var x in LoginInfo)
    {
        if (x is Customer person)
        {
            customers.Add(person);
        }
    }
    return customers;
}

public static void AddinUserList(User a)
{
    LoginInfo.Add(a);
}

public static void RemoveFroUserList(int idx)
{
    LoginInfo.RemoveAt(idx);
}

public static string GetPin()
{
    return Pin;
}

public static void SetPin(string p)
{
    Pin = p;
    UpdatePin();
}

public static bool CheckAdmin()
```


General Store Management System

```
{
    bool flag = false;
    foreach (var x in LoginInfo)
    {
        if (x is Admin)
        {
            flag = true;
            break;
        }
    }
    return flag;
}

public static int IsUserExists(string cname)
{
    int num = -1;
    for (int x = 0; x < LoginInfo.Count; x++)
    {
        if (LoginInfo1[x].GetName() == cname)
        {
            num = x;
            break;
        }
    }
    return num;
}

public static Customer Return_Customer(int idx)
{
    return (Customer)LoginInfo[idx];
}

public static User SignIN(User user)
{
    User person = null;

    foreach (var x in LoginInfo1)
    {
        if (x.GetName() == user.GetName() && x.GetPass() == user.GetPass())
        {
            person = x;
            break;
        }
    }

    return person;
}

public static void SentAnnouncement(string loginName, string message)
{
    List<string> list = null;
    int idx = 0;
    foreach (var x in Announcement1)
    {
```

General Store Management System

```
        if (x[0] == loginName)
        {
            list = x;
            break;
        }
        idx++;
    }
    if (idx >= Announcement1.Count)
    {
        list = new List<string>();
        list.Add(loginName);
    }

    list.Add(message);

    if (idx >= Announcement1.Count)
    {
        Announcement1.Add(list);
    }
    Store_Announcement();
}

public static List<string> WatchAnnouncement(int count)
{
    List<string> list = null;
    for (int i = 0; i < Announcement?.Count; i++)
    {
        if (i == count)
        {
            list = Announcement[count];
        }
    }
    return list;
}

public static bool SentMessage(string loginName, string receiver, string
msg)
{
    int idx;
    bool result = false;

    idx = IsUserExists(receiver);
    if (idx != -1)
    {
        List<string> list = null;
        int count = 1;
        foreach (var x in Message)
        {
            if (x[0] == receiver && x[1] == loginName)
            {
                list = x;
                break;
            }
            count++;
        }

        if (count > Message.Count){
```

General Store Management System

```
        list = new List<string>();
        list.Add(receiver);
        list.Add(loginName);
    }

    string msg1 = "==>" + msg;
    list.Add(msg1);

    if (count > Message.Count){
        Message.Add(list);
    }

    result = true;
    Store_Message();
}
return result;
}

public static List<string> ReceivedMessage(string loginName, int count)
{
    int idx = 0;
    List<string> list = null;
    foreach (var innerlist in Message1)
    {
        if (innerlist[0] == loginName && count == idx)
        {
            list = innerlist;
        }
        idx++;
    }
    return list;
}

/* UserS Data */

public static void StoreUsersData()
{
    StreamWriter myFile = new StreamWriter("UserData.txt");
    foreach (var x in LoginInfo1)
    {
        myFile.WriteLine(x.GetRole() + "," + x.GetName() + "," + x.GetPass()
+ "," + x.GetContact());
    }
    myFile.Flush();
    myFile.Close();
}

public static bool Loadlogindata()
{
    bool result = false;
    string line;
    if (File.Exists("UserData.txt"))
    {
        LoginInfo.Clear();
        StreamReader myFile = new StreamReader("UserData.txt");
```

```
while (!(myFile.EndOfStream))
{
    line = myFile.ReadLine();
    if (line != "")
    {
        string[] splitarray = line.Split(',');

        string role = splitarray[0];
        string name = splitarray[1];
        string password = splitarray[2];
        string contact = splitarray[3];

        if (name != null && password != null && contact != null)
        {
            if (role == (UserType.Admin).ToString())
            {
                AddinUserList(new Admin(UserType.Admin, name,
password, contact));
            }
            else if (role == (UserType.Employee).ToString())
            {
                AddinUserList(new Employee(UserType.Employee, name,
password, contact));
            }
            else if (role == (UserType.Customer).ToString())
            {
                AddinUserList(new Customer(UserType.Customer, name,
password, contact));
            }
        }
    }
}
result = true;
myFile.Close();
}
return result;
}

/* Admin Pin */
public static bool LoadPin()
{
    bool result = false;
    if (File.Exists("AdminPin.txt"))
    {
        pin = null;
        StreamReader File = new StreamReader("AdminPin.txt");
        Pin = File.ReadLine();
        File.Close();
        result = true;
    }
    return result;
}

public static void UpdatePin()
{
    StreamWriter File = new StreamWriter("AdminPin.txt");
    File.WriteLine(Pin);
}
```

```
        File.Flush();
        File.Close();
    }

    /* Announcements */

    public static void Store_Announcement()
    {
        StreamWriter File = new StreamWriter("Announcements.txt");
        foreach (var innerList in Announcement1)
        {
            foreach (var x in innerList)
            {
                File.Write(x + ",");
            }
            File.WriteLine();
        }
        File.Close();
    }

    public static bool Load_Announcements()
    {
        bool result = false;
        string line;
        if (File.Exists("Announcements.txt"))
        {
            Announcement.Clear();
            StreamReader File = new StreamReader("Announcements.txt");

            while (!(File.EndOfStream))
            {
                line = File.ReadLine();

                if (line != "")
                {
                    string[] array = line.Split(',');

                    List<string> list = new List<string>();
                    list.Add(array[0]);

                    for (int i = 1; i < array.Length - 1; i++)
                    {
                        if (array[i] != "")
                        {
                            list.Add(array[i]);
                        }
                    }

                    Announcement1.Add(list);
                }
            }
            result = true;
            File.Close();
        }
        return result;
    }
}
```

```
/* Messenger */

public static void Store_Message()
{
    StreamWriter File = new StreamWriter("Messenger.txt");
    foreach (var innerList in Message1)
    {
        foreach (var x in innerList)
        {
            File.Write(x + ",");
        }
        File.WriteLine();
    }
    File.Close();
}

public static bool Load_Messages()
{
    bool result = false;
    string line;
    if (File.Exists("Messenger.txt"))
    {
        Message.Clear();
        StreamReader File = new StreamReader("Messenger.txt");

        while (!(File.EndOfStream))
        {
            line = File.ReadLine();

            if (line != "")
            {
                string[] array = line.Split(',');

                List<string> list = new List<string>();
                list.Add(array[0]);
                list.Add(array[1]);

                for (int i = 2; i < array.Length - 1; i++)
                {
                    if (array[i] != "")
                    {
                        list.Add(array[i]);
                    }
                }

                Message1.Add(list);
            }
        }
        result = true;
        File.Close();
    }
    return result;
}

}
```

- **EmployeeDL:**

```
class EmployeeDL
{
    private static List<List<Products>> Orders_Received = new
List<List<Products>>();
    private static List<Customer> Receipents = new List<Customer>();

    internal static List<List<Products>> Orders_Received1 { get =>
Orders_Received; set => Orders_Received = value; }
    internal static List<Customer> Receipents1 { get => Receipents; set =>
Receipents = value; }

    public static void AddInOrderList(List<Products> list)
    {
        Orders_Received.Add(list);
    }

    private static void RemoveFromOrderList(int idx)
    {
        Receipents[idx].Addin_Received_List(Orders_Received[idx]);
        Receipents[idx].RemoveFrom_Orders_Placed_List();
        Orders_Received.RemoveAt(idx);
    }

    public static void AddInCustomerList(Customer c)
    {
        Receipents.Add(c);
        Store_Orders_Received_Data();
    }

    private static void RemoveFromCustomerList(int idx)
    {
        Receipents.RemoveAt(idx);
        Store_Orders_Received_Data();
    }

    public static Customer Return_Cust(int idx)
    {
        return Receipents[idx];
    }

    public static List<Products> Orders(Customer c)
    {
        List<Products> list = null;

        for (int x = 0; x < Receipents1.Count; x++)
        {
            if (c == Receipents[x])
            {
                list = Orders_Received1[x];
                break;
            }
        }
    }
}
```

General Store Management System

```
    }
    return list;
}

public static void UpdateTheListData(int idx)
{
    RemoveFromOrderList(idx);
    RemoveFromCustomerList(idx);
}

public static void Store_Orders_Received_Data()
{
    StreamWriter myFile = new StreamWriter("Orders_Received_Data.txt");
    for (int x = 0; x < Receipents1.Count; x++)
    {
        myFile.Write(Receipents1[x].GetName() + ",");

        foreach (var item in Orders_Received1[x])
        {
            myFile.Write(item.GetProductName() + ";" +
item.GetProductPrice() + ";" + item.GetProductQuantity() + ";" +
item.GetProductSale() + "|");
        }

        myFile.WriteLine();
    }
    myFile.Flush();
    myFile.Close();
}

public static bool Load_Orders_Received_Data()
{
    bool result = false;
    string line;
    if (File.Exists("Orders_Received_Data.txt"))
    {
        Receipents.Clear();
        Orders_Received.Clear();
        StreamReader myFile = new StreamReader("Orders_Received_Data.txt");

        while (!(myFile.EndOfStream))
        {
            line = myFile.ReadLine();

            if (line != "")
            {
                string[] splitarray = line.Split(',');

                string custName = splitarray[0];
                Customer cust =
UserDL.Return_Customer(UserDL.IsUserExists(custName));
                Receipents1.Add(cust);

                string[] pro = splitarray[1].Split('|');
                List<Products> list = new List<Products>();
            }
        }
    }
}
```



```
        for (int i = 0; i < pro.Length; i++)
        {
            if (pro[i] != "")
            {
                string[] data = pro[i].Split(';');
                list.Add(new Products(data[0], float.Parse(data[1]),
int.Parse(data[2]), int.Parse(data[3])));
            }
        }

        cust.Addin_Orders_placed_List(list);
        AddInOrderList(list);
    }
}
result = true;
myFile.Close();
}
return result;
}
}
```

- **ProductsDL:**

```
class ProductsDL
{
    private static List<Products> inventory = new List<Products>();

    internal static List<Products> Inventory { get => inventory; set =>
inventory = value; }

    public static void AddinProductsList(Products P)
    {
        inventory.Add(P);
    }

    public static void RemoveinProductsList(int idx)
    {
        inventory.RemoveAt(idx);
    }

    public static int IsProductExists(string pname)
    {
        int num = -1;
        for (int i = 0; i < inventory.Count; i++)
        {
            if (inventory[i].GetProductName() == pname)
            {
                num = i;
                break;
            }
        }
        return num;
    }
}
```

General Store Management System

```
}

public static Products ReturnProduct(string name)
{
    Products item = null;
    foreach (var x in inventory)
    {
        if(x.GetProductName() == name)
        {
            item = x; break;
        }
    }
    return item;
}

public static void Minus_Quantity(Products p)
{
    Products a = ReturnProduct(p.ProductName);

    if ((a.Quantity - p.Quantity) < 0)
    {
        p.SetProductQuantity(a.Quantity);
    }

    a.SetProductQuantity( a.Quantity - p.Quantity );
}

public static List<Products> Search_Product(string s)
{
    List<Products> product = inventory.FindAll(item =>
item.GetProductName().StartsWith(s));
    return product;
}

public static void StoreProductsData()
{
    StreamWriter myFile = new StreamWriter("ProductsData.txt");
    foreach (var x in inventory)
    {
        myFile.WriteLine(x.GetProductName() + "," + x.GetProductPrice() +
", " + x.GetProductQuantity() + "," + x.GetProductThreshold() + "," +
x.GetProductSale());
    }
    myFile.Flush();
    myFile.Close();
}

public static bool LoadProductsData()
{
    bool result = false;
    string line;
    if (File.Exists("ProductsData.txt"))
```

General Store Management System

```
{
    inventory.Clear();
    StreamReader myFile = new StreamReader("ProductsData.txt");
    while (!(myFile.EndOfStream))
    {
        line = myFile.ReadLine();
        if (line != "")
        {
            string[] splittedarray = line.Split(',');

            string productName = splittedarray[0];
            float price = float.Parse(splittedarray[1]);
            int quantity = int.Parse(splittedarray[2]);
            int sale = int.Parse(splittedarray[4]);

            if (productName != null && IsNum(price.ToString()) &&
                IsNum(quantity.ToString()) && IsNum(sale.ToString()))
            {
                int threshold = (40 * quantity) / 100;
                AddinProductsList(new Products(productName, price,
                    quantity, threshold, sale));
            }
        }
        result = true;
        myFile.Close();
    }
    return result;
}

private static bool IsNum(string str)
{
    bool result = false;
    if (str != null)
    {
        foreach (var i in str)
        {
            if((int)i >= 48 && (int)i <= 58)
            {
                result = true;
            }
            else
            {
                break;
            }
        }
    }
    return result;
}
}
```

- **Misc:**

```
internal class Misc
{
```

```
public static void ShowUserData(Label pname1, Label pname2, User user)
{
    pname1.Text = user.GetName();
    pname2.Text = user.GetContact();
}

public static void Logout(Form form)
{
    form.Close();
    form.Hide();
    Form nform = new Main_Menu();
    nform.ShowDialog();
}

public static void Set_Form(Form name, Panel p_name)
{
    name.TopLevel = false;
    name.FormBorderStyle = FormBorderStyle.None;
    name.Dock = DockStyle.Fill;
    p_name.Controls.Add(name);
    p_name.Tag = name;
    name.BringToFront();
    name.Show();
}

public static InputBox Asking_UserName(Panel pname)
{
    InputBox form = new InputBox();
    form.SetLabel("Enter the receiver name :");
    form.SetButtonText("Select");
    Set_Form(form, pname);
    return form;
}

public static InputBox Write_Message(Panel pname)
{
    InputBox form = new InputBox();
    form.SetLabel("Enter your message :");
    form.SetButtonText("Send");
    Set_Form(form, pname);
    return form;
}

public static InputBox ChangePassForm(Panel pname)
{
    InputBox nform = new InputBox();
    nform.SetLabel("Enter the new password :");
    Set_Form(nform, pname);
    return nform;
}

public static bool SeeProducts(User user, Panel pname)
{
    bool result = false;
    if (ProductsDL.Inventory?.Count > 0)
    {

```

```
        WatchProducts emp = new WatchProducts(user, ProductsDL.Inventory);
        Set_Form(emp, pname);
        result = true;
    }
    return result;
}

public static float BillCalculation(List<Products> list)
{
    float money, discount, amount = 0;

    foreach (var item in list)
    {
        money = item.GetProductPrice() * item.GetProductQuantity();

        if (item.GetProductSale() > 0)
        {
            discount = (item.GetProductPrice() * item.GetProductSale()) /
100;
            money = money - (discount * item.GetProductQuantity());
        }
        amount += money;
    }

    return amount;
}

public static DataGridViewButtonColumn GV_AddButton(string headingname,
string name)
{
    DataGridViewButtonColumn buttonColumn = new DataGridViewButtonColumn();

    buttonColumn.HeaderText = headingname;
    buttonColumn.Text = headingname;
    buttonColumn.UseColumnTextForButtonValue = true;

    buttonColumn.Name = name;

    return buttonColumn;
}

public static void Read_Message(string name, Panel pname)
{
    int count = 0;
    List<string> list1;
    List<string> list = new List<string>();
    do
    {
        list1 = UserDL.ReceivedMessage(name, count);

        if (list1 != null)
        {
            count++;
        }
    }
}
```

```
        for (int i = 1; i < list1.Count; i++)
        {
            list.Add(list1[i]);
        }

        list.Add("-----");
    }
}
while (list1 != null);

if (list?.Count > 0)
{
    ReadMessages form = new ReadMessages(list);
    Set_Form(form, pname);
}
else
{
    MessageBox.Show("No message Revceived");
}
}

public static void See_Announcements(Pane1 pname)
{
    int count = 0;
    List<string> list1;
    List<string> list = new List<string>();
    do
    {
        list1 = UserDL.WatchAnnouncement(count);

        if (list1 != null)
        {
            count++;
            list.AddRange(list1);
            list.Add("-----");
        }
    }
    while (list1 != null);

    if (list?.Count > 0)
    {
        ReadMessages form = new ReadMessages(list);
        Set_Form(form, pname);
    }
    else
    {
        MessageBox.Show("No new Announcements");
    }
}

}
```

8.4. Graphical User Interface (GUI) forms:

- **Main Menu:**

```
public partial class Main_Menu : Form
{
    private int count = 0;

    public Main_Menu()
    {
        InitializeComponent();
        Load_Data();
        HideAllMenus();
    }

    private void HideAllMenus()
    {
        signUp_Role_Panel.Visible = false;
    }

    private void ShowMenu()
    {
        signUp_Role_Panel.Visible = true;
    }

    private void Load_Data()
    {
        if (UserDL.LoadPin() == false)
        {
            MessageBox.Show("Unable to load Pin data !!!");
        }
        if (UserDL.Load_Messages() == false)
        {
            MessageBox.Show("Unable to load Message data !!!");
        }
        if (UserDL.Loadlogindata() == false)
        {
            MessageBox.Show("Unable to load Users data !!!");
        }
        if (UserDL.Load_Announcements() == false)
        {
            MessageBox.Show("Unable to load Announcements data !!!");
        }
        if (ProductsDL.LoadProductsData() == false)
        {
            MessageBox.Show("Unable to load Products data !!!");
        }
        if (EmployeeDL.Load_Orders_Received_Data() == false)
        {
            MessageBox.Show("Unable to load Orders Received data !!!");
        }
    }

    private void signUp_btn_Click(object sender, EventArgs e)
    {
        ShowMenu();
    }
}
```

```
}

private void logIn_btn_Click(object sender, EventArgs e)
{
    this.Hide();
    Form nform = new Log_In();
    nform.ShowDialog();
}

private void OpenSignUpForm(UserType user)
{
    this.Hide();
    Form nform = new SignUp(user);
    nform.ShowDialog();
}

private void admin_SignUp_btn_Click(object sender, EventArgs e)
{
    count = 1;

    if (UserDL.CheckAdmin())
    {
        HideAllMenus();
        EnterSignUpPin();
    }
    else
    {
        OpenSignUpForm(UserType.Admin);
    }
}

private void employee_SignUp_btn_Click(object sender, EventArgs e)
{
    count = 2;
    HideAllMenus();
    EnterSignUpPin();
}

private void customer_SignUp_btn_Click(object sender, EventArgs e)
{
    OpenSignUpForm(UserType.Customer);
}

private void EnterSignUpPin()
{
    InputBox form = new InputBox();
    form.SetLabel("Enter Sign-Up Pin:");
    form.SetButtonText("Sign-Up");
    form.IsBtnClick += Form_IsBtnClick;
    form.ShowDialog();
}

private void Form_IsBtnClick(object sender, EventArgs e)
{
    if (((InputBox)sender).IsClick && ((InputBox)sender).data ==
UserDL.GetPin())
    {

```



```
        this.Hide();

        if (count == 1)
        {
            OpenSignUpForm(UserType.Admin);
        }
        else if (count == 2)
        {
            OpenSignUpForm(UserType.Employee);
        }

        this.Close();
    }
    else
    {
        MessageBox.Show("You can't Sign Up without Pin!!!");
    }
}

private void Main_Menu_FormClosed(object sender, FormClosedEventArgs e)
{
    Environment.Exit(0);
}

}
```

- **Sign Up;**

```
public partial class SignUp : Form
{
    private UserType _userType;

    public SignUp(UserType user)
    {
        InitializeComponent();
        _userType = user;
    }

    private void create_Account_btn_Click(object sender, EventArgs e)
    {
        if (signUp_username_txt.Text != "" && signUp_password_txt.Text != "" &&
signUp_contact_txt.Text != "")
        {
            if (UserDL.CheckAdmin())
            {
                if (Create_Account())
                {
                    MessageBox.Show("Account Created Successfully !!!");
                }
                else
                {
                    MessageBox.Show("User Already Exists");
                }
            }
        }
    }
}
```

```
        else if (Create_Account())
        {
            InputBox update = new InputBox();
            update.SetLabel("Enter the Sign-Up Pin:");
            update.ShowDialog();

            if (((InputBox)sender).data != "")
            {
                update.IsBtnClick += Form_IsBtnClick;
            }
        }
        else
        {
            MessageBox.Show("User Already Exists");
        }
    }
}

private void Form_IsBtnClick(object sender, EventArgs e)
{
    if (((InputBox)sender).IsClick)
    {
        string pin = ((InputBox)sender).data;

        if (pin != null && pin.Length == 5)
        {
            UserDL.SetPin(pin);

            MessageBox.Show("Account Created Successfully !!!");
            RefreshData();
        }
    }
}

private bool Create_Account()
{
    bool result = false;
    string userName = signUp_userName_txt.Text;
    if (UserDL.IsUserExists(userName) != -1)
    {
        result = false;
    }
    else
    {
        string password = signUp_Password_txt.Text;
        string contact = signUp_Contact_txt.Text;

        if (_userType == UserType.Admin)
        {
            Admin info = new Admin(UserType.Admin, userName, password,
contact);
            UserDL.AddinUserList(info);
        }
        else if (_userType == UserType.Employee)
        {
            Employee info = new Employee(UserType.Employee, userName,
password, contact);
```

General Store Management System

```
        UserDL.AddinUserList(info);
    }
    else if (_userType == UserType.Employee)
    {
        Customer info = new Customer(UserType.Customer, userName,
password, contact);
        UserDL.AddinUserList(info);
    }
    result = true;
    UserDL.StoreUsersData();
}
return result;
}

private void RefreshData()
{
    signUp_userName_txt.Text = null;
    signUp_Password_txt.Text = null;
    signUp_Contact_txt.Text = null;
}

private void Back_btn_Click(object sender, EventArgs e)
{
    Main_Menu form = new Main_Menu();
    form.Show();
}

// Validating Functions

// UserName
private void signUp_userName_txt_KeyPress(object sender, KeyPressEventArgs
e)
{
    e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
}

private void signUp_userName_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(signUp_userName_txt, string.Empty);
    if (!string.IsNullOrEmpty(signUp_userName_txt.Text.Trim()))
    {
        epSuccess.SetError(signUp_userName_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(signUp_userName_txt, string.Empty);
    }
}

private void signUp_userName_txt_Validating(object sender, CancelEventArgs
e)
{
    if (!string.IsNullOrEmpty(signUp_userName_txt.Text.Trim()))
    {
        epError.SetError(signUp_userName_txt, string.Empty);
    }
}
```

```
    }
    else
    {
        epError.SetError(signUp_userName_txt, "Name is required.");
    }
}

// Password
private void signUp_Password_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(signUp_Password_txt, string.Empty);
    if (!string.IsNullOrEmpty(signUp_Password_txt.Text.Trim()))
    {
        epSuccess.SetError(signUp_Password_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(signUp_Password_txt, string.Empty);
    }
}

private void signUp_Password_txt_Validating(object sender, CancelEventArgs
e)
{
    if (!string.IsNullOrEmpty(signUp_Password_txt.Text.Trim()))
    {
        epError.SetError(signUp_Password_txt, string.Empty);
    }
    else
    {
        epError.SetError(signUp_Password_txt, "Name is required.");
    }
}

// Contact
private void signUp_Contact_txt_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = !(char.IsNumber(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
}

private void signUp_Contact_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(signUp_Contact_txt, string.Empty);
    if (!string.IsNullOrEmpty(signUp_Contact_txt.Text.Trim()))
    {
        epSuccess.SetError(signUp_Contact_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(signUp_Contact_txt, string.Empty);
    }
}

private void signUp_Contact_txt_Validating(object sender, CancelEventArgs e)
```

```
{
    if (!string.IsNullOrEmpty(signUp_Contact_txt.Text.Trim()))
    {
        epError.SetError(signUp_Contact_txt, string.Empty);
    }
    else
    {
        epError.SetError(signUp_Contact_txt, "Name is required.");
    }
}

}
```

- **Log In:**

```
public partial class Log_In : Form
{
    public Log_In()
    {
        InitializeComponent();
    }

    private void Log_In_FormClosed(object sender, FormClosedEventArgs e)
    {
        Environment.Exit(0);
    }

    private void Back_btn_Click(object sender, EventArgs e)
    {
        this.Close();
        this.Hide();
        Form nform = new Main_Menu();
        nform.ShowDialog();
    }

    private void signIn_btn_Click(object sender, EventArgs e)
    {
        if (logIn_UserName_txt.Text != "" && logIn_Password_txt.Text != "")
        {
            User loginUser = UserDL.SignIN(SignIn_User());

            if (loginUser != null)
            {
                this.Close();
                this.Hide();

                if (loginUser is Admin)
                {
                    Admin person = (Admin)loginUser;
                    Form nform = new AdminGUI(person);
                    nform.ShowDialog();
                }
                else if (loginUser is Employee)
                {
                    Employee person = (Employee)loginUser;
                    Form nform = new EmployeeGUI(person);
                }
            }
        }
    }
}
```

```
        nform.ShowDialog();
    }
    else if (loginUser is Customer)
    {
        Customer person = (Customer)loginUser;
        Form nform = new CustomerGUI(person);
        nform.ShowDialog();
    }
}
else
{
    MessageBox.Show("No User Found");
}
}
}

private User SignIn_User()
{
    User user;
    string name, pass;
    do
    {
        name = logIn_UserName_txt.Text;
        pass = logIn_Password_txt.Text;
    }
    while (name == null && pass == null);

    return user = new User(name, pass);
}

// Validating Functions

private void logIn_UserName_txt_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = !(char.IsLetterOrDigit(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
}

private void logIn_UserName_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(logIn_UserName_txt, string.Empty);
    if (!string.IsNullOrEmpty(logIn_UserName_txt.Text.Trim()))
    {
        epSuccess.SetError(logIn_UserName_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(logIn_UserName_txt, string.Empty);
    }
}

private void logIn_UserName_txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(logIn_UserName_txt.Text.Trim()))
    {
        epError.SetError(logIn_UserName_txt, string.Empty);
    }
}
```

```
    }
    else
    {
        epError.SetError(logIn_UserName_txt, "Name is required.");
    }
}

private void logIn_Password_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(logIn_Password_txt, string.Empty);
    if (!string.IsNullOrEmpty(logIn_Password_txt.Text.Trim()))
    {
        epSuccess.SetError(logIn_Password_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(logIn_Password_txt, string.Empty);
    }
}

private void logIn_Password_txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(logIn_Password_txt.Text.Trim()))
    {
        epError.SetError(logIn_Password_txt, string.Empty);
    }
    else
    {
        epError.SetError(logIn_Password_txt, "Name is required.");
    }
}
}
```

- **Admin GUI:**

```
public partial class AdminGUI : Form
{
    private Admin _user;
    private object idx = null;
    private string receiver = null;

    public AdminGUI(Admin user)
    {
        this._user = user;
        this.InitializeComponent();
        Main_image.BackColor = Color.Transparent;
        this.HideAllMenus();
        Misc.ShowUserData(userName_lbl, userContact_lbl, _user);
    }

    private void HideAllMenus()
    {
        admin_Products_Options_Panel.Visible = false;
    }
}
```

```
        manage_Employee_Options_Panel.Visible = false;
        messenger_Options_Panel.Visible = false;
        announcement_Options_Panel.Visible = false;

        menu_Display_Panel.Controls.Clear();
    }

    private void ShowMenu(Panel p)
    {
        p.Visible = true;
    }

    private void AdminGUI_FormClosed(object sender, FormClosedEventArgs e)
    {
        Environment.Exit(0);
    }

    // Products Details

    private void products_Details_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        this.ShowMenu(admin_Products_Options_Panel);
    }

    // Watch Products

    private void watch_Products_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        if (!Misc.SeeProducts(_user, menu_Display_Panel))
        {
            MessageBox.Show("No Product Found");
        }
    }

    // Add Products

    private void add_Products_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        AddProduct form = new AddProduct(_user);
        Misc.Set_Form(form, menu_Display_Panel);
    }

    // Product Selected Event Form

    private void Form_ProductSelected(object sender, EventArgs e)
    {
        SearchData searchData = ((SearchData)sender);
        idx = searchData.SelectedProduct;
```


General Store Management System

```
        InputBox nform = searchData.Tag as InputBox;
        Misc.Set_Form(nform, menu_Display_Panel);
    }

    // Update Price

    private void update_Price_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        SearchData form = new SearchData(_user);
        InputBox nform = new InputBox();
        nform.IsBtnClick += Price_IsBtnClick;
        form.Tag = nform;
        Misc.Set_Form(form, menu_Display_Panel);
        form.ProductSelected += Form_ProductSelected;
    }
    private void Price_IsBtnClick(object sender, EventArgs e)
    {
        if (((InputBox)sender).IsClick)
        {
            ProductsDL.ReturnProduct(idx.ToString()) .
SetProductPrice(float.Parse(((InputBox)sender).data));
            MessageBox.Show("Data Updated Successfully !");
        }
    }

    // Update Quantity

    private void update_Stock_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        SearchData form = new SearchData(_user);
        InputBox nform = new InputBox();
        nform.IsBtnClick += Quantity_IsBtnClick;
        form.Tag = nform;
        Misc.Set_Form(form, menu_Display_Panel);
        form.ProductSelected += Form_ProductSelected;
    }
    private void Quantity_IsBtnClick(object sender, EventArgs e)
    {
        if (((InputBox)sender).IsClick)
        {
            ProductsDL.ReturnProduct(idx.ToString()) .
SetProductQuantity(int.Parse(((InputBox)sender).data));
            MessageBox.Show("Data Updated Successfully !");
        }
    }

    // Add / Remove Sale

    private void add_Sale_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
    }
```

General Store Management System

```
        SearchData form = new SearchData(_user);
        InputBox nform = new InputBox();
        nform.IsBtnClick += Sale_IsBtnClick;
        form.Tag = nform;
        Misc.Set_Form(form, menu_Display_Panel);
        form.ProductSelected += Form_ProductSelected;
    }
    private void Sale_IsBtnClick(object sender, EventArgs e)
    {
        if (((InputBox)sender).IsClick)
        {
            ProductsDL.ReturnProduct(idx.ToString()) .
SetProductSale(int.Parse(((InputBox)sender).data));
            MessageBox.Show("Data Updated Successfully !");
        }
    }

    // Watch Employee Button

    private void employee_Manage_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        this.ShowMenu(manage_Employee_Options_Panel);
    }

    // Watch Employee

    private void watch_Employee_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        List<Employee> list = UserDL.GetAllEmployees();
        if (list?.Count > 0)
        {
            Form cust = new ViewUsers(list);
            Misc.Set_Form(cust, menu_Display_Panel);
        }
        else
        {
            MessageBox.Show("Not Found !!! ");
        }
    }

    // Customer Traffic

    private void customer_Traffic_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        List<Customer> list = UserDL.GetAllCustomers();
        if (list?.Count > 0)
        {
            Form cust = new ViewUsers(list, _user);
            Misc.Set_Form(cust, menu_Display_Panel);
        }
    }
}
```

```
        else
        {
            MessageBox.Show("Not Found !!! ");
        }
    }

    // Messenger Button

    private void admin_Messenger_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        this.ShowMenu(messenger_Options_Panel);
    }

    // Send Message

    private void sent_Message_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        Misc.Asking_UserName(menu_Display_Panel).IsBtnClick += Form_IsBtnClick;
    }
    private void Form_IsBtnClick(object sender, EventArgs e)
    {
        receiver = ((InputBox)sender).data;
        if (UserDL.IsUserExists(receiver) != -1)
        {
            Misc.Write_Message(menu_Display_Panel).IsBtnClick += Msg_IsBtnClick;
        }
        else
        {
            MessageBox.Show("No User Exists !!");
        }
    }
    private void Msg_IsBtnClick(object sender, EventArgs e)
    {
        string message = ((InputBox)sender).data.ToString();
        if (((InputBox)sender).IsClick && message != null)
        {
            if (UserDL.SendMessage(_user.GetName(), receiver, message))
            {
                MessageBox.Show("Message has been sent ");
            }
        }
    }

    // Read Messages

    private void read_Message_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        Misc.Read_Message(_user.GetName(), menu_Display_Panel);
    }
```

```
// Announcement Button

private void announcement_btn_Click(object sender, EventArgs e)
{
    this.HideAllMenus();
    this.ShowMenu(announcement_Options_Panel);
}

// Make Announcement

private void make_announcement_btn_Click(object sender, EventArgs e)
{
    Misc.Write_Message(menu_Display_Panel).IsBtnClick +=
Announcement_IsBtnClick;
}
private void Announcement_IsBtnClick(object sender, EventArgs e)
{
    string message = ((InputBox)sender).data.ToString();
    if (((InputBox)sender).IsClick && message != null)
    {
        UserDL.SentAnnouncement(_user.GetName(), message);
        MessageBox.Show("Announcement is Posted !!! ");
    }
}

// Watch Announcement

private void watch_Announcements_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    Misc.See_Announcements(menu_Display_Panel);
}

// Log Out

private void logOut_btn_Click(object sender, EventArgs e)
{
    Misc.Logout(this);
}

// Password Change

private void user_PasswordChange_btn_Click(object sender, EventArgs e)
{
    this.HideAllMenus();
    Misc.ChangePassForm(menu_Display_Panel).IsBtnClick +=
Password_IsBtnClick;
}
private void Password_IsBtnClick(object sender, EventArgs e)
{
    if (((InputBox)sender).IsClick)
```

```
        {
            _user.ChangePass(((InputBox)sender).data);
            MessageBox.Show("Data Updated Successfully !");
        }
    }
}
```

- **Employee GUI:**

```
public partial class EmployeeGUI : Form
{
    private Employee _user;
    private int customerIndex = -1;
    private string receiver = null;

    public EmployeeGUI(Employee user)
    {
        this._user = user;
        this.InitializeComponent();
        this.HideAllMenus();
        Misc.ShowUserData(userName_lbl, userContact_lbl, _user);
    }

    private void HideAllMenus()
    {
        E_Products_Panel.Visible = false;
        E_Announcement_Panel.Visible = false;
        E_Messenger_Panel.Visible = false;
        menu_Display_Panel.Controls.Clear();
    }

    private void ShowMenu(Panel p)
    {
        p.Visible = true;
    }

    private void EmployeeGUI_FormClosed(object sender, FormClosedEventArgs e)
    {
        Environment.Exit(0);
    }

    // Products Details

    private void E_ProductsDetails_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        this.ShowMenu(E_Products_Panel);
    }

    // Watch Products

    private void E_ProductsWatch_btn_Click(object sender, EventArgs e)
    {

```

General Store Management System

```
        HideAllMenus();
        if (!Misc.SeeProducts(_user, menu_Display_Panel))
        {
            MessageBox.Show("No Product Found");
        }
    }

    // Watch Low Quantity Products

    private void E_LowStockProduct_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        if (ProductsDL.Inventory.Count > 0)
        {
            Form emp = new WatchProducts(ProductsDL.Inventory);
            Misc.Set_Form(emp, menu_Display_Panel);
        }
        else
        {
            MessageBox.Show("No Product Found");
        }
    }

    // Orders Received Button

    private void E_OrdersReceive_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        if (EmployeeDL.Receipents1.Count > 0)
        {
            ViewUsers cust = new ViewUsers(EmployeeDL.Receipents1, _user);
            cust.CustomerSelected += Cust_CustomerSelected;
            Misc.Set_Form(cust, menu_Display_Panel);
        }
        else
        {
            MessageBox.Show("No Orders Received !!!");
        }
    }

    private void Cust_CustomerSelected(object sender, EventArgs e)
    {
        this.customerIndex = ((ViewUsers)sender).CustomerIndex;
        Orders_Received();
    }

    private void Orders_Received()
    {
        HideAllMenus();
        Form orders = new WatchProducts(_user,
EmployeeDL.Orders(EmployeeDL.Return_Cust(customerIndex)));
        Misc.Set_Form(orders, menu_Display_Panel);
    }

    // Announcement Button

    private void E_Announcement_btn_Click(object sender, EventArgs e)
```

General Store Management System

```
{
    this.HideAllMenus();
    this.ShowMenu(E_Announcement_Panel);
}

// Make Announcement

private void E_make_announcement_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    Misc.Write_Message(menu_Display_Panel).IsBtnClick +=
Announcement_IsBtnClick;
}
private void Announcement_IsBtnClick(object sender, EventArgs e)
{
    string message = ((InputBox)sender).data.ToString();
    if (((InputBox)sender).IsClick && message != null)
    {
        UserDL.SentAnnouncement(_user.GetName(), message);
        MessageBox.Show("Announcement is Posted !!! ");
    }
}

// Watch Announcement

private void E_watch_Announcements_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    Misc.See_Announcements(menu_Display_Panel);
}

// Messenger Button

private void E_Messenger_btn_Click(object sender, EventArgs e)
{
    this.HideAllMenus();
    this.ShowMenu(E_Messenger_Panel);
}

// Send Message

private void E_Sent_Message_btn_Click(object sender, EventArgs e)
{
    this.HideAllMenus();
    Misc.Asking_UserName(menu_Display_Panel).IsBtnClick += Form_IsBtnClick;
}
private void Form_IsBtnClick(object sender, EventArgs e)
{
    receiver = ((InputBox)sender).data;
    if (UserDL.IsUserExists(receiver) != -1)
    {
        Misc.Write_Message(menu_Display_Panel).IsBtnClick += Msg_IsBtnClick;
    }
    else
    {
        MessageBox.Show("No User Exists !!");
    }
}
```

```
    }
}
private void Msg_IsBtnClick(object sender, EventArgs e)
{
    string message = ((InputBox)sender).data.ToString();
    if (((InputBox)sender).IsClick && message != null)
    {
        if (UserDL.SendMessage(_user.GetName(), receiver, message))
        {
            MessageBox.Show("Message has been sent ");
        }
    }
}

// Watch Message
private void E_Read_Message_btn_Click(object sender, EventArgs e)
{
    Misc.Read_Message(_user.GetName(), menu_Display_Panel);
}

// Log Out
private void E_LogOut_btn_Click(object sender, EventArgs e)
{
    Misc.Logout(this);
}

// Password Change
private void user_PasswordChange_btn_Click(object sender, EventArgs e)
{
    this.HideAllMenus();
    Misc.ChangePassForm(menu_Display_Panel).IsBtnClick +=
Password_IsBtnClick;
}
private void Password_IsBtnClick(object sender, EventArgs e)
{
    if (((InputBox)sender).IsClick)
    {
        _user.ChangePass(((InputBox)sender).data);
        MessageBox.Show("Data Updated Successfully !");
    }
}
}
```

- **Customer GUI:**

```
public partial class CustomerGUI : Form
{
    private Customer _user;
    private Products item;
    public CustomerGUI(Customer user)
```



```
{
    this._user = user;
    InitializeComponent();
    this.HideAllMenus();
    Misc.ShowUserData(userName_lbl, userContact_lbl, _user);
}

private void CustomerGUI_FormClosed(object sender, FormClosedEventArgs e)
{
    Environment.Exit(0);
}

private void HideAllMenus()
{
    C_Products_Panel.Visible = false;
    C_Cart_Panel.Visible = false;
    C_Purchase_Panel.Visible = false;

    menu_Display_Panel.Controls.Clear();
}

private void ShowMenu(Pane1 p)
{
    p.Visible = true;
}

private void C_Logout_btn_Click(object sender, EventArgs e)
{
    Misc.Logout(this);
}

// Products Details Button

private void C_Products_Details_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    ShowMenu(C_Products_Panel);
}

// Watch Products

private void C_WatchAll_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    if (ProductsDL.Inventory?.Count > 0)
    {
        WatchProducts pro = new WatchProducts(_user, ProductsDL.Inventory);
        Misc.Set_Form(pro, menu_Display_Panel);
    }
    else
    {
        MessageBox.Show("No Product Found");
    }
}
```

```
// Cart Button

private void C_Cart_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    ShowMenu(C_Cart_Panel);
}

// Search Products
private void C_SearchProduct_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    SearchData form = new SearchData(_user);
    Misc.Set_Form(form, menu_Display_Panel);
    form.ProductAddCart += AddCart_ProductAddCart;
}
private void AddCart_ProductAddCart(object sender, EventArgs e)
{
    item =
ProductsDL.ReturnProduct(((SearchData)sender).CartProduct.ToString());

    InputBox nform = new InputBox();
    nform.SetLabel("Enter the Quantity :");
    nform.SetButtonText("Add");
    nform.IsBtnClick += AddCart_IsBtnClick;
    nform.ShowDialog();
}
private void AddCart_IsBtnClick(object sender, EventArgs e)
{
    if (((InputBox)sender).IsClick)
    {
        string name = item.GetProductName();
        int idx = ProductsDL.IsProductExists(name);
        if (idx != -1)
        {
            float price = ProductsDL.ReturnProduct(name).GetProductPrice();
            int quantity = int.Parse(((InputBox)sender).data);
            int sale = ProductsDL.ReturnProduct(name).GetProductSale();

            if (quantity <
ProductsDL.ReturnProduct(name).GetProductQuantity())
            {
                _user.AddinCartList(new Products(name, price, quantity,
sale));

                MessageBox.Show("Item Added Successfully !!!");
            }
            else
            {
                MessageBox.Show("Enter valid number !");
            }
        }
        else
    }
}
```

```
        {
            MessageBox.Show("Product doesn't Exists!!!");
        }
    }
}

// View Cart

private void C_ViewCart_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    List<Products> list = _user.GetCartList();
    if (list?.Count > 0)
    {
        WatchProducts form = new WatchProducts(_user, list, 1);
        Misc.Set_Form(form, menu_Display_Panel);
        form.SetBillTable(Misc.BillCalculation(list).ToString());
    }
    else
    {
        MessageBox.Show("No Product added in Cart!!!");
    }
}

// Purchase Button

private void C_Purchase_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    ShowMenu(C_Purchase_Panel);
}

// Watch Orders Given

private void C_OrderGiven_btn_Click(object sender, EventArgs e)
{
    HideAllMenus();
    List<Products> list = _user.GetOrdersList();
    if (list?.Count > 0)
    {
        WatchProducts form = new WatchProducts(_user, list, 2);
        form.SetBillTable(Misc.BillCalculation(list).ToString());
        Misc.Set_Form(form, menu_Display_Panel);
    }
    else
    {
        MessageBox.Show("No items Purchased!!!");
    }
}

// Items Received

private void C_Received_btn_Click(object sender, EventArgs e)
{

```

```
        HideAllMenus();
        List<Products> list = _user.GetReceivedList();
        if (list?.Count > 0)
        {
            WatchProducts form = new WatchProducts(_user, list, 2);
            Misc.Set_Form(form, menu_Display_Panel);
        }
        else
        {
            MessageBox.Show("No items Received!!!");
        }
    }

    // Watch Announcements

    private void C_Announcement_btn_Click(object sender, EventArgs e)
    {
        HideAllMenus();
        Misc.See_Announcements(menu_Display_Panel);
    }

    // Change Password

    private void user_PasswordChange_btn_Click(object sender, EventArgs e)
    {
        this.HideAllMenus();
        Misc.ChangePassForm(menu_Display_Panel).IsBtnClick +=
        Password_IsBtnClick;
    }
    private void Password_IsBtnClick(object sender, EventArgs e)
    {
        if (((InputBox)sender).IsClick)
        {
            _user.ChangePass(((InputBox)sender).data);
            MessageBox.Show("Data Updated Successfully !");
        }
    }
}
```

• Add Products:

```
public partial class AddProduct : Form
{
    private User user = null;
    public AddProduct(User user)
    {
        this.user = user;
        InitializeComponent();
    }

    private void Add_btn_Click(object sender, EventArgs e)
    {

```

General Store Management System

```
        if (product_Name_txt.Text != "" && product_Price_txt.Text != "" &&
product_Quantity_txt.Text != "" && product_Sale_txt.Text != "")
        {
            AdProduct();
        }
    }

    private void AdProduct()
    {
        string name = product_Name_txt.Text;
        if (ProductsDL.IsProductExists(name) == -1 && name != "")
        {
            float price = float.Parse(product_Price_txt.Text);
            int quantity = int.Parse(product_Quantity_txt.Text);
            int threshold = (40 * quantity) / 100;
            int sale = int.Parse(product_Sale_txt.Text);
            Products items = new Products(name, price, quantity, threshold,
sale);

            ProductsDL.AddinProductsList(items);
            ProductsDL.StoreProductsData();
            MessageBox.Show("Product Added!!!");
            RefreshingData();
        }
        else
        {
            MessageBox.Show("Product Exists!!!");
            RefreshingData();
        }
    }

    private void RefreshingData()
    {
        product_Name_txt.Text = null;
        product_Price_txt.Text = null;
        product_Quantity_txt.Text = null;
        product_Sale_txt.Text = null;
    }

    private void Back_btn_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    // Validation Functions

    // Name Validation
    private void product_Name_txt_KeyPress(object sender, KeyPressEventArgs e)
    {
        e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
    }

    private void product_Name_txt_KeyUp(object sender, KeyEventArgs e)
```

```
{
    epError.SetError(product_Name_txt, string.Empty);
    if (!string.IsNullOrEmpty(product_Name_txt.Text.Trim()))
    {
        epSuccess.SetError(product_Name_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(product_Name_txt, string.Empty);
    }
}

private void product_Name_txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(product_Name_txt.Text.Trim()))
    {
        epError.SetError(product_Name_txt, string.Empty);
    }
    else
    {
        epError.SetError(product_Name_txt, "Name is required.");
    }
}

// Price Validation
private void product_Price_txt_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = !(char.IsNumber(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
}

private void product_Price_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(product_Price_txt, string.Empty);
    if (!string.IsNullOrEmpty(product_Price_txt.Text.Trim()))
    {
        epSuccess.SetError(product_Price_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(product_Price_txt, string.Empty);
    }
}

private void product_Price_txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(product_Price_txt.Text.Trim()))
    {
        epError.SetError(product_Price_txt, string.Empty);
    }
    else
    {
        epError.SetError(product_Price_txt, "Name is required.");
    }
}
```

```
// Quantity Validation
private void product_Quantity_txt_KeyPress(object sender, KeyPressEventArgs
e)
{
    e.Handled = !(char.IsNumber(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
}

private void product_Quantity_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(product_Quantity_txt, string.Empty);
    if (!string.IsNullOrEmpty(product_Quantity_txt.Text.Trim()))
    {
        epSuccess.SetError(product_Quantity_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(product_Quantity_txt, string.Empty);
    }
}

private void product_Quantity_txt_Validating(object sender, CancelEventArgs
e)
{
    if (!string.IsNullOrEmpty(product_Quantity_txt.Text.Trim()))
    {
        epError.SetError(product_Quantity_txt, string.Empty);
    }
    else
    {
        epError.SetError(product_Quantity_txt, "Name is required.");
    }
}

// Sale Validation
private void product_Sale_txt_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = !(char.IsNumber(e.KeyChar) || e.KeyChar == (char)Keys.Back
|| e.KeyChar == (char)Keys.Space);
}

private void product_Sale_txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(product_Sale_txt, string.Empty);
    if (!string.IsNullOrEmpty(product_Sale_txt.Text.Trim()))
    {
        epSuccess.SetError(product_Sale_txt, "Valid");
    }
    else
    {
        epSuccess.SetError(product_Sale_txt, string.Empty);
    }
}

private void product_Sale_txt_Validating(object sender, CancelEventArgs e)
```

```
{
    if (!string.IsNullOrEmpty(product_Sale_txt.Text.Trim()))
    {
        epError.SetError(product_Sale_txt, string.Empty);
    }
    else
    {
        epError.SetError(product_Sale_txt, "Name is required.");
    }
}
}
```

- **Watch Products:**

```
public partial class WatchProducts : Form
{
    private User user = null;
    private List<Products> list = null;
    private int isCartItem = 0;

    public WatchProducts(List<Products> list)
    {
        this.list = list;
        InitializeComponent();
        Bill_TablePanel.Visible = false;
        ShowlowStock();
    }

    public WatchProducts(User user, List<Products> list)
    {
        this.user = user;
        this.list = list;
        InitializeComponent();
        Bill_TablePanel.Visible = false;
        DataBind();
    }

    public WatchProducts(User user, List<Products> list, int isCartItem)
    {
        this.user = user;
        this.list = list;
        this.isCartItem = isCartItem;

        InitializeComponent();

        if (isCartItem == 1)
        {
            Bill_TablePanel.Visible = false;
            Back_btn.Text = "Purchase";
        }

        DataBind();
    }
}
```



```
private void DataBind()
{
    GV.DataSource = null;

    if (user is Admin)
    {
        Bind();
        AddDelButton();
    }
    else if (user is Employee)
    {
        Bind();
    }
    else
    {
        if (isCartItem == 1)
        {
            Bind();
            AddCartDelButton();
        }
        else if (isCartItem == 2)
        {
            Bind();
        }
        else
        {
            Customer_P_View();
        }
    }

    GV.Refresh();
}

private void Bind()
{
    GV.DataSource = list.Select(c => new { c.ProductName, c.Price,
c.Quantity, c.Sale }).ToList();
}
private void Customer_P_View()
{
    foreach (var o in list)
    {
        string Status = "In-Stock";

        if (o.GetProductQuantity() < o.GetProductThreshold())
        {
            Status = "Out of Stock";
        }

        GV.DataSource = list.Select(c => new { c.ProductName, c.Price,
c.Sale, Status }).ToList();
    }
}

private void AddDelButton()
```

General Store Management System

```
{
    DataGridViewButtonColumn button = Misc.GV_AddButton("Delete",
"Gv_Del_Btn");
    GV.Columns.Add(button);
    GV.CellContentClick += GV_Del_CellContentClick;
}
private void GV_Del_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (GV.SelectedCells[0].ColumnIndex == GV.Columns["Gv_Del_Btn"].Index)
    {
        int idx = GV.SelectedCells[0].RowIndex;
        object cellValue = GV.Rows[idx].Cells["ProductName"].Value;

ProductsDL.RemoveinProductsList(ProductsDL.IsProductExists(cellValue.ToString()));
        ProductsDL.StoreProductsData();

        MessageBox.Show("Item Deleted !!!");
    }
}

private void AddCartDelButton()
{
    DataGridViewButtonColumn button = Misc.GV_AddButton("Delete",
"Gv_Del_Btn");
    GV.Columns.Add(button);
    GV.CellContentClick += GV_CartDel_CellContentClick;
}
private void GV_CartDel_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (GV.SelectedCells[0].ColumnIndex == GV.Columns["Gv_Del_Btn"].Index)
    {
        int idx = GV.SelectedCells[0].RowIndex;

        Customer cust = (Customer)user;

cust.RemovefromCartList(ProductsDL.ReturnProduct((GV.Rows[idx].Cells["ProductName"].
Value).ToString()));

        MessageBox.Show("Item Deleted !!!");
    }
}

private void ShowlowStock()
{
    GV.DataSource = null;
    if (list.Count > 0)
    {
        foreach (var o in list)
        {
```

General Store Management System

```
        if (o.GetProductQuantity() < o.GetProductThreshold())
        {
            GV.DataSource = list.Select(c => new { c.ProductName,
c.Price, c.Quantity, c.Sale }).ToList();
        }
        GV.Refresh();
    }
    else
    {
        MessageBox.Show("No Product Found");
        this.Close();
    }
}

public void SetBillTable(string line)
{
    Bill_TablePanel.Visible = true;
    total_Bill_lbl.Text = line;
}
private void Back_btn_Click(object sender, EventArgs e)
{
    if (isCartItem == 1)
    {
        Customer cust = user as Customer;

        foreach (var x in list)
        {
            ProductsDL.Minus_Quantity(x);
        }

        cust.Addin_Orders_placed_List(list);
        EmployeeDL.AddInOrderList(list);

EmployeeDL.AddInCustomerList(UserDL.Return_Customer(UserDL.IsUserExists(user.GetName
())));

        cust.ClearCartList();

        MessageBox.Show("Items are Purchased. You will shortly received the
items !!!");
    }
    this.Hide();
}
}
```

- **Search Data:**

```
public partial class SearchData : Form
{
    public event EventHandler ProductSelected;
```

General Store Management System

```
public event EventHandler ProductAddCart;

public object SelectedProduct { get =>
display_GV.Rows[display_GV.SelectedCells[0].RowIndex].Cells["ProductName"].Value; }
public object CartProduct { get => ITEM; }

public object ITEM;

private User user;

public SearchData(User user)
{
    InitializeComponent();
    display_GV.Visible = false;
    this.user = user;
}

private void search_btn_Click(object sender, EventArgs e)
{
    if (search_Txt.Text != null)
    {
        List<Products> list = ProductsDL.Search_Product(search_Txt.Text);

        if (list?.Count > 0)
        {
            display_GV.DataSource = null;

            if (user is Customer)
            {
                Cutomer_ProductsBinding(list);
                DataGridViewButtonColumn button = Misc.GV_AddButton("Add",
"Gv_Add_Btn");
                display_GV.Columns.Add(button);
                display_GV.CellContentClick += AddColumn_CellContentClick;
            }
            else if (user is Admin)
            {
                ProductsBinding(list);
                DataGridViewButtonColumn button =
Misc.GV_AddButton("Delete", "Gv_Del_Btn");
                display_GV.Columns.Add(button);
                display_GV.CellContentClick += DelColumn_CellContentClick;
            }

            display_GV.Visible = true;
            display_GV.Refresh();
        }
        else
        {
            MessageBox.Show("No Data Found !!!");
        }
    }
}

private void DelColumn_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{

```

General Store Management System

```
        if (display_GV.SelectedCells[0].ColumnIndex ==
display_GV.Columns["Gv_Del_btn"].Index)
        {
            int idx = display_GV.SelectedCells[0].RowIndex;

            object cellValue = display_GV.Rows[idx].Cells["ProductName"].Value;

ProductsDL.RemoveinProductsList(ProductsDL.IsProductExists(cellValue.ToString()));
            UserDL.StoreUsersData();

            MessageBox.Show("Item Deleted !!!");
        }
    }

    private void AddColumn_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        if (display_GV.SelectedCells[0].ColumnIndex ==
display_GV.Columns["Gv_Add_btn"].Index)
        {
            int idx = display_GV.SelectedCells[0].RowIndex;
            ITEM = display_GV.Rows[idx].Cells["ProductName"].Value;
            ProductAddCart?.Invoke(this, e);
        }
    }

    private void display_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        ProductSelected?.Invoke(this, e);
    }

    private void ProductsBinding(List<Products> list)
    {
        display_GV.DataSource = list.Select(c => new { c.ProductName, c.Price,
c.Quantity, c.Sale }).ToList();
    }

    private void Cutomer_ProductsBinding(List<Products> list)
    {
        display_GV.DataSource = list.Select(c => new { c.ProductName, c.Price,
c.Sale }).ToList();
    }

    // Validating Function

    private void search_Txt_KeyPress(object sender, KeyPressEventArgs e)
    {
        e.Handled = !(char.IsLetterOrDigit(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
    }
}
```

```
}

private void search_Txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(search_Txt, string.Empty);
    if (!string.IsNullOrEmpty(search_Txt.Text.Trim()))
    {
        epSuccess.SetError(search_Txt, "Valid");
    }
    else
    {
        epSuccess.SetError(search_Txt, string.Empty);
    }
}

private void search_Txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(search_Txt.Text.Trim()))
    {
        epError.SetError(search_Txt, string.Empty);
    }
    else
    {
        epError.SetError(search_Txt, "Name is required.");
    }
}
}
```

- **View Users:**

```
public partial class ViewUsers : Form
{
    public User user;

    public event EventHandler CustomerSelected;
    public int CustomerIndex { get => GV.SelectedCells[0].RowIndex; }

    public ViewUsers(List<Employee> list)
    {
        InitializeComponent();
        DataBind(list);
    }
    public ViewUsers(List<Customer> list, User user)
    {
        this.user = user;
        InitializeComponent();
        DataBind(list);
    }

    private void DataBind(List<Employee> list)
    {
        GV.DataSource = null;
```

General Store Management System

```
        GV.DataSource = list.Select(c => new { c.Role, c.UserName, c.Contact
    }).ToList();
    GV.Refresh();
    AddDelButton();
}

private void DataBind(List<Customer> list)
{
    GV.DataSource = null;
    GV.DataSource = list.Select(c => new { c.Role, c.UserName, c.Contact
}).ToList();
    GV.Refresh();
    if (user is Admin)
    {
        AddDelButton();
    }
    else if (user is Employee)
    {
        AddShowButton();
        AddupdateButton();
    }
}

private void Back_btn_Click(object sender, EventArgs e)
{
    this.Close();
}

private void AddShowButton()
{
    DataGridViewButtonColumn button = Misc.GV_AddButton("Orders",
"Gv_Orders_Btn");
    GV.Columns.Add(button);
    GV.CellContentClick += Users_GV_Orders_CellContentClick;
}
private void Users_GV_Orders_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if(GV.SelectedCells[0].ColumnIndex == GV.Columns["Gv_Orders_Btn"].Index)
    {
        CustomerSelected?.Invoke(this, new EventArgs());
        this.Close();
    }
}

private void AddDelButton()
{
    DataGridViewButtonColumn button = Misc.GV_AddButton("Delete",
"Gv_Del_Btn");
    GV.Columns.Add(button);
    GV.CellContentClick += Users_GV_Del_CellContentClick;
}
```

General Store Management System

```
private void Users_GV_Del_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (GV.SelectedCells[0].ColumnIndex == GV.Columns["Gv_Del_Btn"].Index)
    {
        int idx = GV.SelectedCells[0].RowIndex;
        object cellValue = GV.Rows[idx].Cells["UserName"].Value;

        UserDL.RemoveFroUserList(UserDL.IsUserExists(cellValue.ToString()));
        UserDL.StoreUsersData();

        MessageBox.Show("User Deleted !!!");
    }
}

private void AddupdateButton()
{
    DataGridViewButtonColumn button = Misc.GV_AddButton("Update",
"Gv_Update_Btn");
    GV.Columns.Add(button);
    GV.CellContentDoubleClick += Users_GV_Update_CellContentDoubleClick;
}

private void Users_GV_Update_CellContentDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    if (GV.SelectedCells[0].ColumnIndex ==
GV.Columns["Gv_Update_Btn"].Index)
    {
        EmployeeDL.UpdateTheListData(e.RowIndex);
        MessageBox.Show("Order is fulfilled !!!");
    }
}
}
```

- **Input Box:**

```
public partial class InputBox : Form
{
    public event EventHandler IsBtnClick;

    public bool IsClick
    {
        get => Is_Click;
    }

    public string data
    {
        get => update_Data_Txt.Text;
    }

    private bool Is_Click;
    public InputBox()
    {
        InitializeComponent();
    }
}
```


General Store Management System

```
{
    Is_Click = false;
    InitializeComponent();
}

private void Back_btn_Click(object sender, EventArgs e)
{
    this.Close();
}

private void Update_btn_Click(object sender, EventArgs e)
{
    if (update_Data_Txt.Text != "")
    {
        Is_Click = true;
        IsBtnClick?.Invoke(this, e);
        this.Close();
        this.Hide();
    }
}

public void SetLabel(string line)
{
    updated_lbl.Text = line;
}

public void SetButtonText(string line)
{
    update_btn.Text = line;
}

// Validation Functions

private void update_Data_Txt_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = !(char.IsLetterOrDigit(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
}

private void update_Data_Txt_KeyUp(object sender, KeyEventArgs e)
{
    epError.SetError(update_Data_Txt, string.Empty);
    if (!string.IsNullOrEmpty(update_Data_Txt.Text.Trim()))
    {
        epSuccess.SetError(update_Data_Txt, "Valid");
    }
    else
    {
        epSuccess.SetError(update_Data_Txt, string.Empty);
    }
}

private void update_Data_Txt_Validating(object sender, CancelEventArgs e)
{
    if (!string.IsNullOrEmpty(update_Data_Txt.Text.Trim()))
```

```
        {
            epError.SetError(update_Data_Txt, string.Empty);
        }
        else
        {
            epError.SetError(update_Data_Txt, "Name is required.");
        }
    }
}
```

- **Read Message:**

```
public partial class ReadMessages : Form
{
    public ReadMessages(List<string> list)
    {
        InitializeComponent();
        DataBind(list);
    }

    private void Back_btn_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void DataBind(List<string> list)
    {
        msg_Display_GV.DataSource = null;
        msg_Display_GV.DataSource = list.Select(s => new { value = s
    }).ToList();
        msg_Display_GV.Refresh();
    }
}
```

9.CONCLUSION:

In conclusion, I will summarize the General Shop Management System project, its achievements, the challenges faced during development, and the insights gained from this endeavor.

9.1. Summarization and Achievements:

The General Shop Management System is a modern, technology-driven solution designed to streamline and optimize the operations of a retail shop. By applying the principles of efficient data management and user-friendly interfaces, the system aims to enhance the overall shop management process, leading to improved customer service and increased productivity.

Throughout the development of the project, several noteworthy achievements have been realized. The system offers a user-friendly interface that enables easy navigation and quick access to essential shop data. Real-time inventory tracking ensures that shop owners can efficiently manage stock levels, minimizing stockouts and maximizing profits. The system also incorporates features such as sales and purchase management, employee management, and data security measures, bolstering the system's overall functionality and reliability.

9.2. Challenges:

However, the project did encounter some challenges during its development journey. One significant challenge was ensuring seamless communication and integration between different modules within the system. Managing a vast amount of product data, sales records, and employee information while maintaining data accuracy and consistency also posted its own set of challenges. Additionally, implementing Object-Relational Mapping (ORM) techniques for efficient database management required careful consideration and attention to detail.

8.3 Lessons Learned:

Throughout the project, several valuable lessons were learned that have contributed to a better understanding of software development and management systems. Effective planning and thorough requirement gathering were vital for the successful development of a complex system like this. Emphasizing modularity and separation of concerns played a crucial role in achieving maintainability and adaptability. Leveraging design patterns, such as Model-View-Controller (MVC) and Data Access Object (DAO), helped in creating a structured and easily maintainable codebase. Additionally, continuous testing and quality assurance were indispensable for identifying and rectifying any issues early in the development process.

In conclusion, the General Shop Management System represents a significant step towards transforming traditional retail shop operations into a more efficient and customer-centric process. By overcoming challenges and incorporating valuable lessons, the project has laid a strong foundation for further improvements and future advancements in shop management technology.