Dairy Delights

Session 2023 - 2027

Submitted by:

Abu Tayyab 2023-CS-54

Supervised by:

Prof. Maida Shahid

Course:

CSC-103 Object Oriented Programming

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Table of Contents

Table of Wire frames

1. Introduction:

Dairy Delights Business Application is designed to manage operations for a dairy products business. This application provides functionalities for four types of users: Admin, Employees, Delivery Boy and Customers. The application offers a user-friendly interface allowing efficient management of tasks related to inventory, sales, and customer interactions.

Design

The whole project consists of 3 sub-projects:

1. **Project Library:** It contains all the classes of the entities in the Business Layer and a Data Layer to manage all entities' data. The Data Base folder in the Data Layer handles all data using Database connection and SQL queries. Likewise, The File Handling folder handles only the User data. The Data Layer Interfaces folder contains all the Interfaces of the Data Layer Classes. The use of C# language is in full effect in the back-end. This Library makes the base of the Application.
2. **Window Forms Project:** It contains the Front-End of the Application. User-Friendly Window Forms are linked with the Project Library for smooth running of the Application.
3. **Console Application:** It contains a Console App in which the CRUD of the User entity has been performed.

OOP Pillars

1. **Encapsulation:** The Class Attributes have been made private and accessed through Getters & Setters. The Class Behaviors are public. The Parent Class Attributes are protected.
2. **Association:** The Customer Class has a Ticket Object to show Aggregation between Customer and Ticket classes so that when a customer buys a ticket, it is saved as an object in the customer class. The Ticket Class has an Object of Match class to show which match the ticket is of.

2. User Roles and Functionalities:

Dairy Delights supports four user roles: Admin, Employees, Delivery Boy and Customers. Each role has distinct functionalities tailored to their specific needs.

I) Admin:

1) Add new User to the system.

2) Remove User from the system.
3) View Orders
4) Add More Products
5) Remove Product
6) Update Products
7) Check Review about Products from Customers
8) See Complains/ Suggestions from Worker.
9) Update Details About Worker.
10) Change Password

ii) Worker

1) Add Products
2) Remove Products
3) Update Products
4) Check Reviews about Products
5) View Pending Orders
6) Write Complains / Suggestions to Admin
7) Update Details of Customer
8) View his details
9) Change Password

iii) Delivery Boy:

(1) View Pending Orders
(2) View Order Details
(3) Deliver Orders
(4) View Personal Details
(5) Change Password

iv) Customer:

(1) View Products
(2) View Details of Products
(3) View Reviews of Products
(4) Give Reviews
(5) View Own Cart
(6) Add Items in Cart
(7) Edit Cart
(8) Buy Cart

(9) View Orders.
(10) Change Password

As a	I want to perform	So that I can
Admin	Add new employee.	Add a new employee to the system.
	Remove employee.	Remove employee from the system.
	View Orders	See All The orders delivered or not
	Add Product	Add Product in the store
	Update Product	Update details quantity etc.
	Remove Products	Remove Products from Inventory
	Check Review	Check Reviews of Products
	See complains	See suggestion and problems of Worker
	Update Users	Update details of Users
	Change password.	Can change password of any username without even knowing old passwords.
Worker	Update Products	Update details of product
	Add products	Add new products for sale in inventory.
	Remove products	Remove products form the inventory
	View Orders	View Pending Orders
	Check Reviews About Products	Check Reviews form Customers.
	Remove discount.	Remove discount form the products.
	Update Customer	Change details of Customer
	Writes complain to Admin.	Writes the issues he is facing to the admin so that issues can be resolved.
	Change password.	He can change password but cannot change password of employee.
	View Orders	View Order from the Customers

Delivery Boy	Products Details	See products Details in Order
	Deliver Orders	Deliver orders to the Customer
	View His Own Details	See his details Bike number etc.
	Change Password	Change his own password
Customer	View products.	See products available for sale.
	View Reviews	See review of other Customers
	View Details	View the description and other details about products
	Give Feedback	Give review about products
	View Cart	View His Items
	Edit Cart	Change the cart add or remove things
	Buy	Buy the products
	Change password	Change login password
	Update Details	Update Address and phone number etc.

3. Wire Frames:

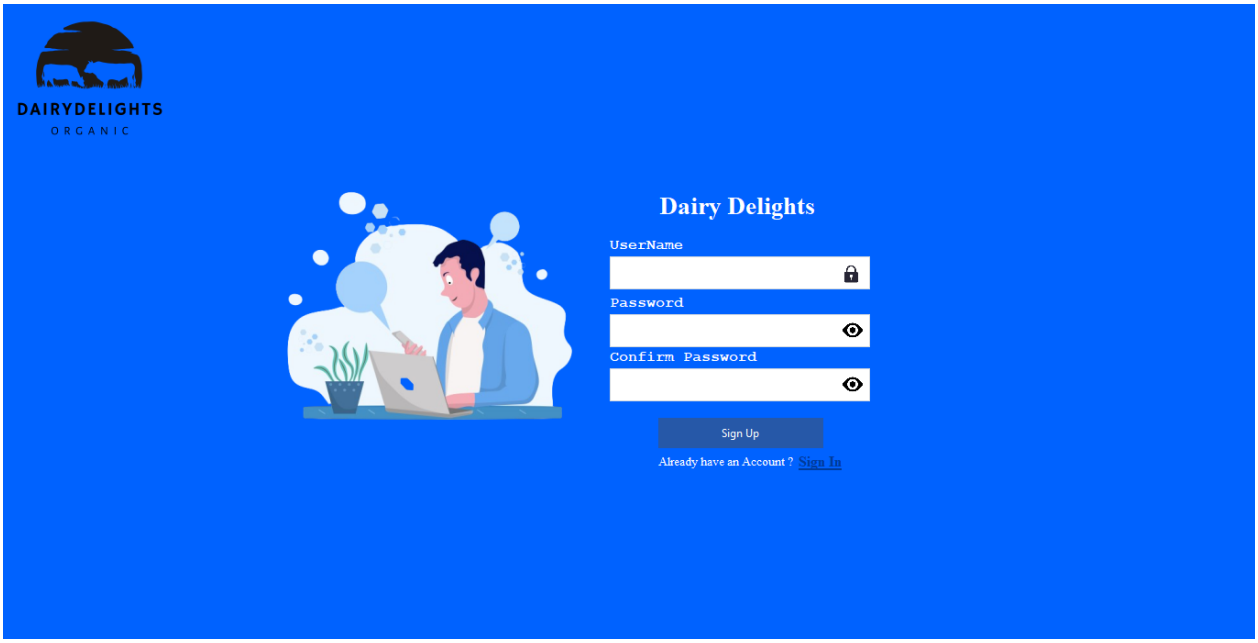


Figure 1Sign Up Page

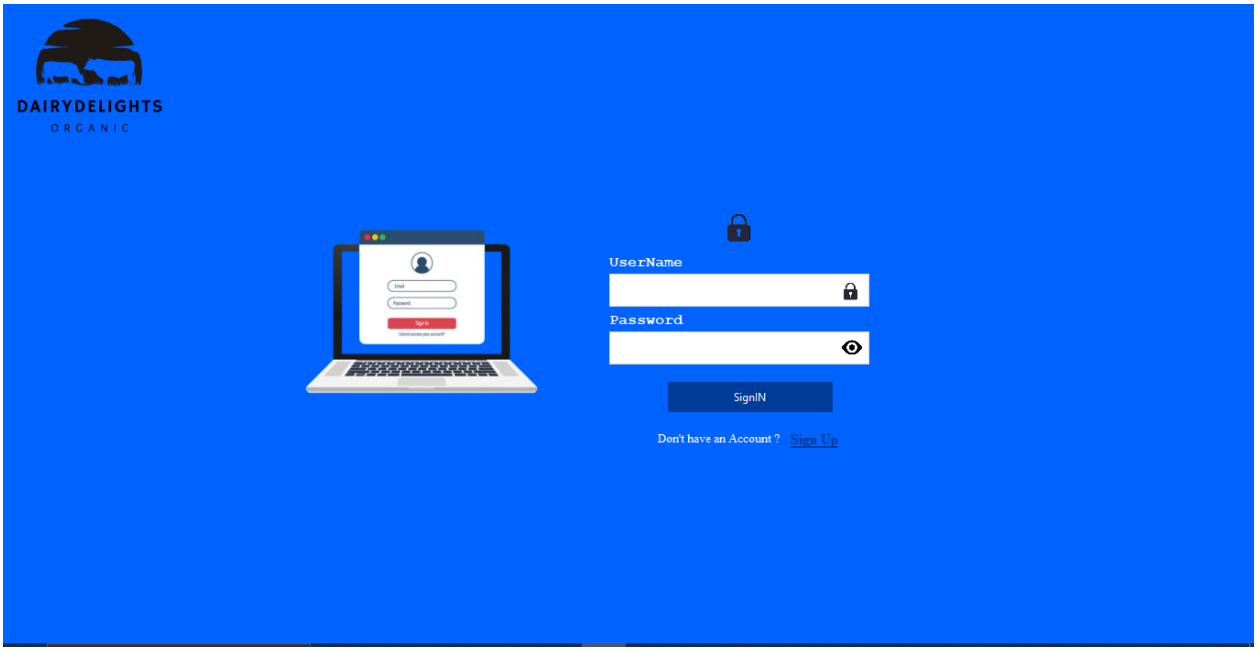


Figure 2Sign In page

Figure 3Admin Main Page

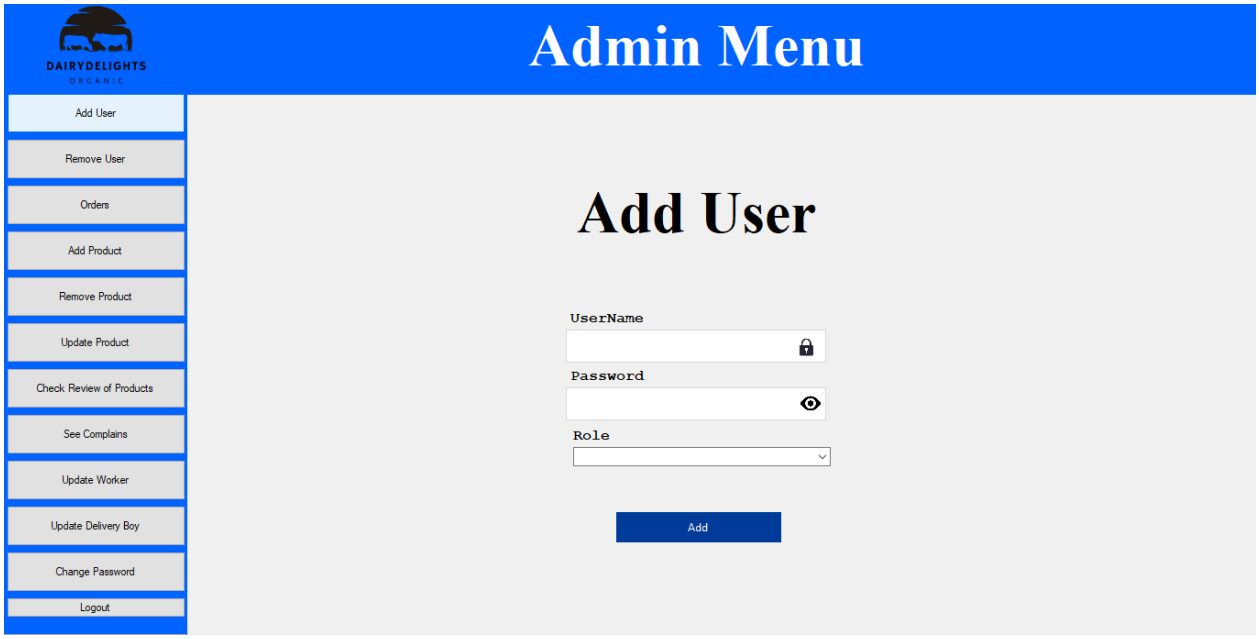


Figure 4Add User Admin

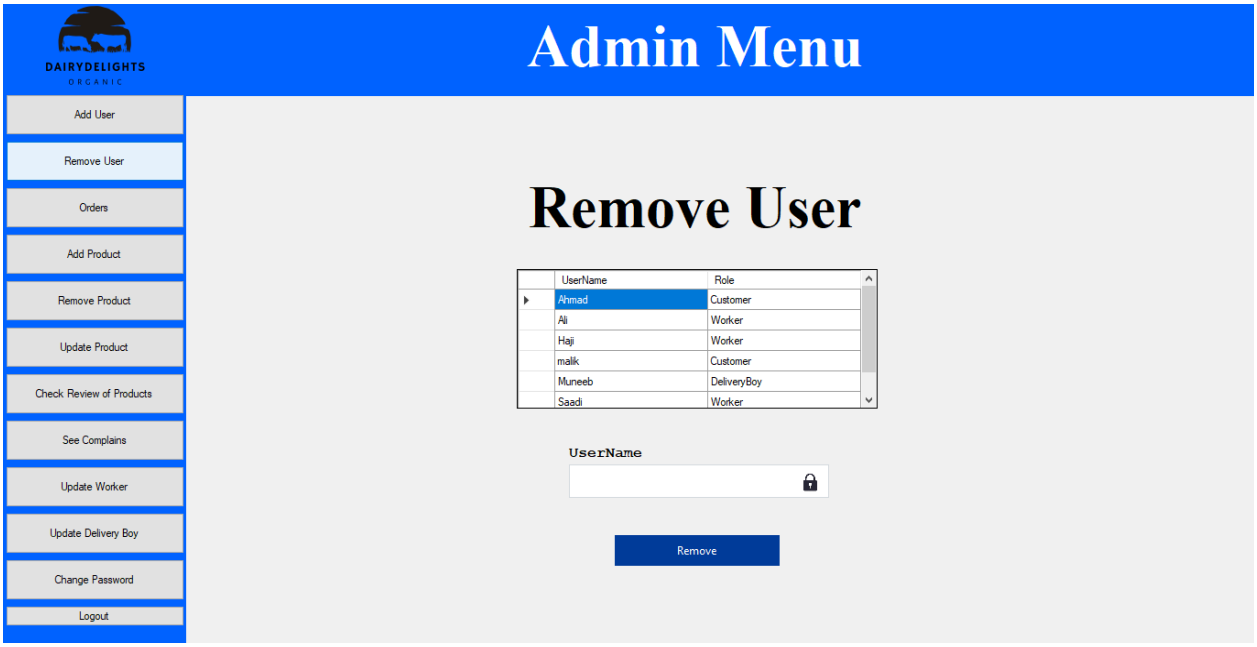


Figure 5Remove User

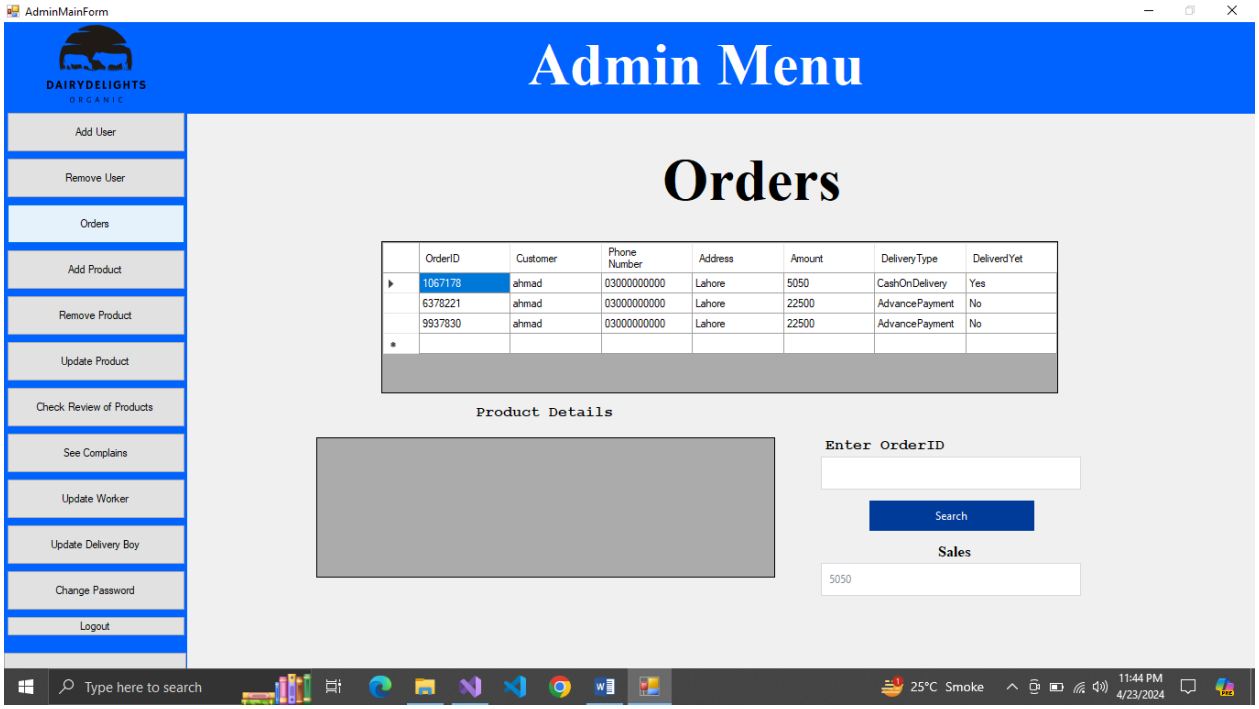


Figure 6orders

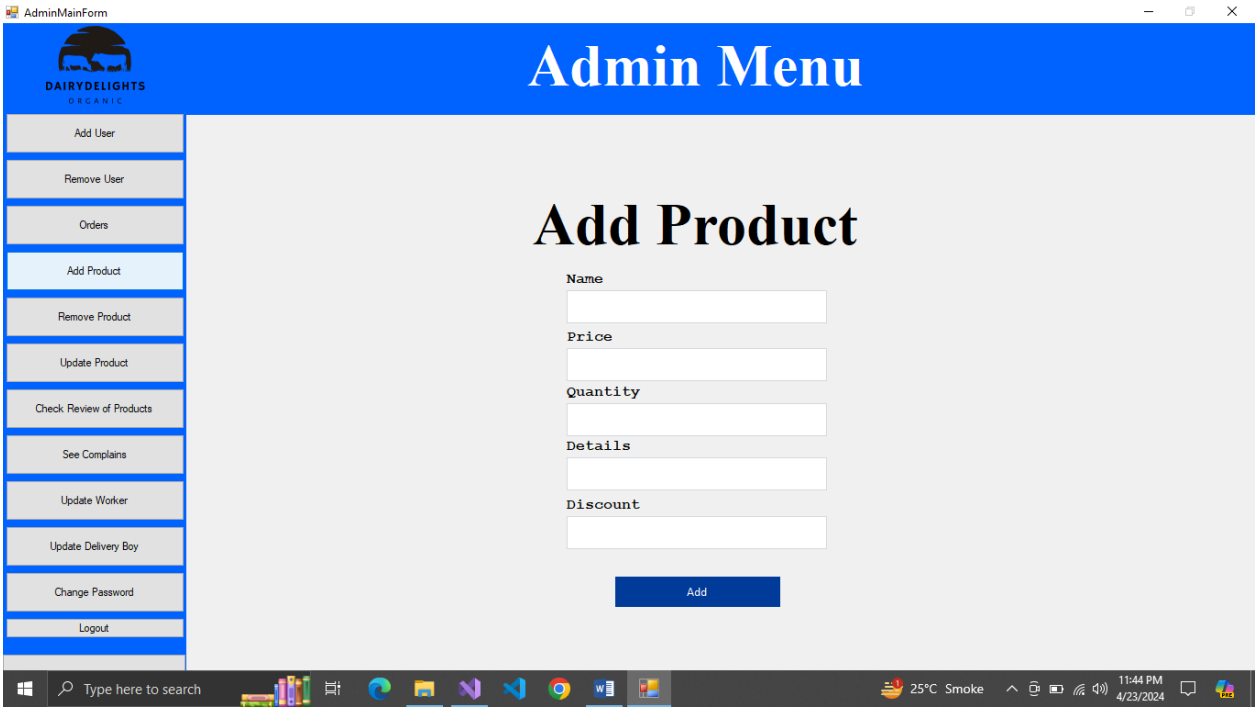


Figure 7Add Product

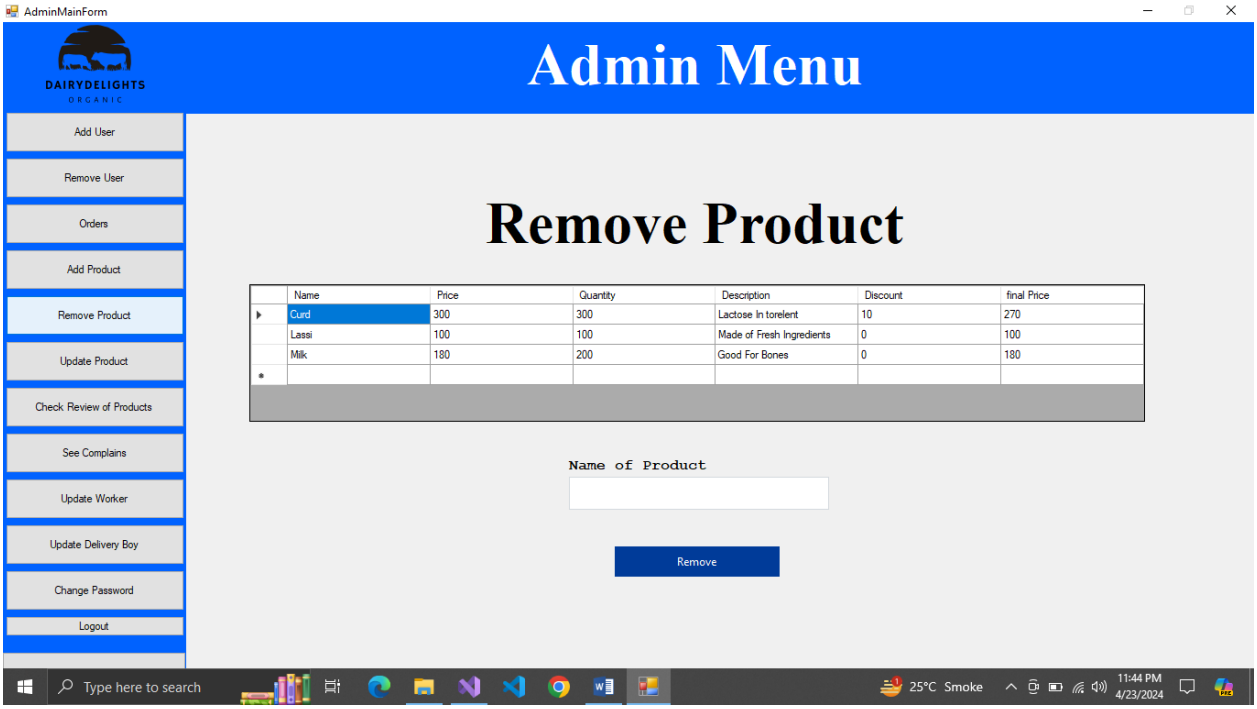


Figure 8Remove Products

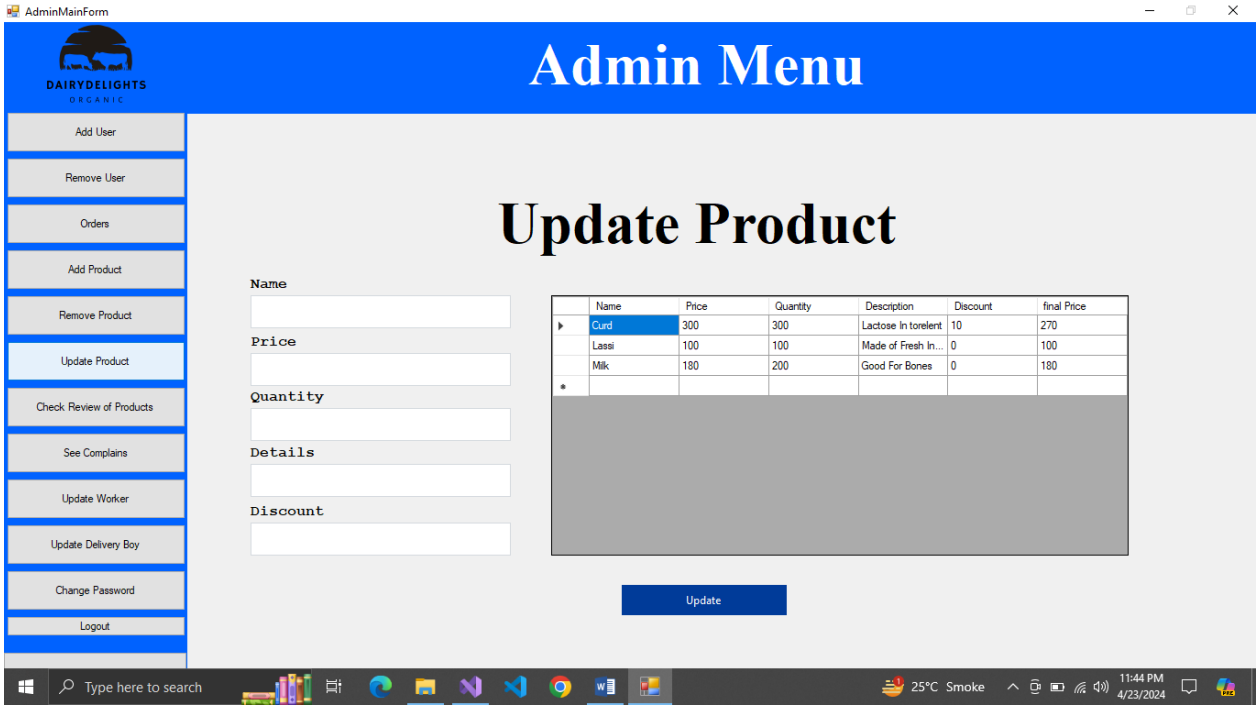
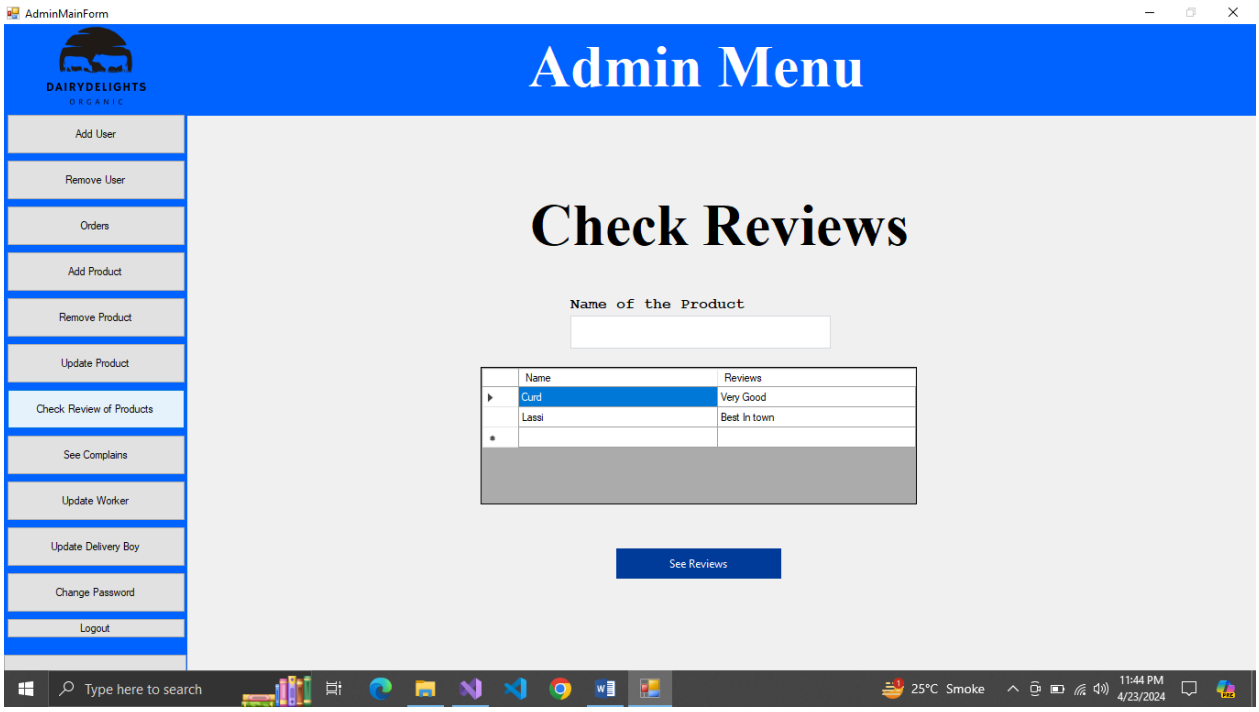


Figure 9Update Products



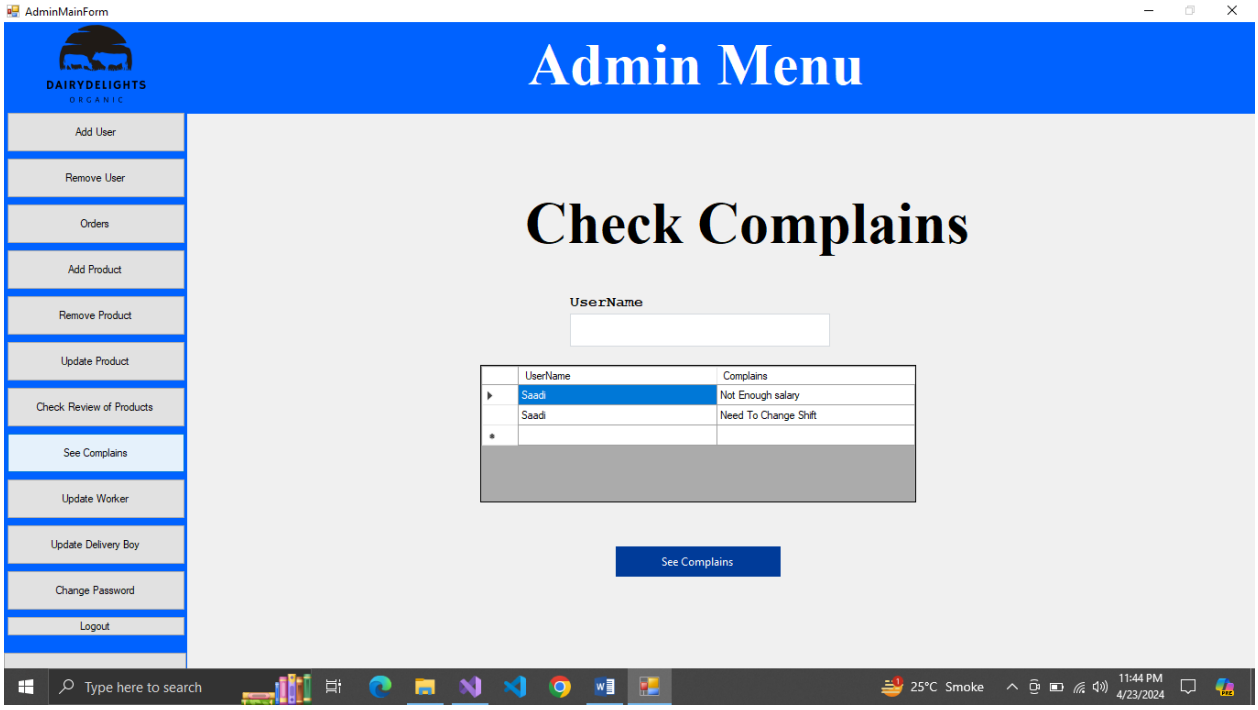
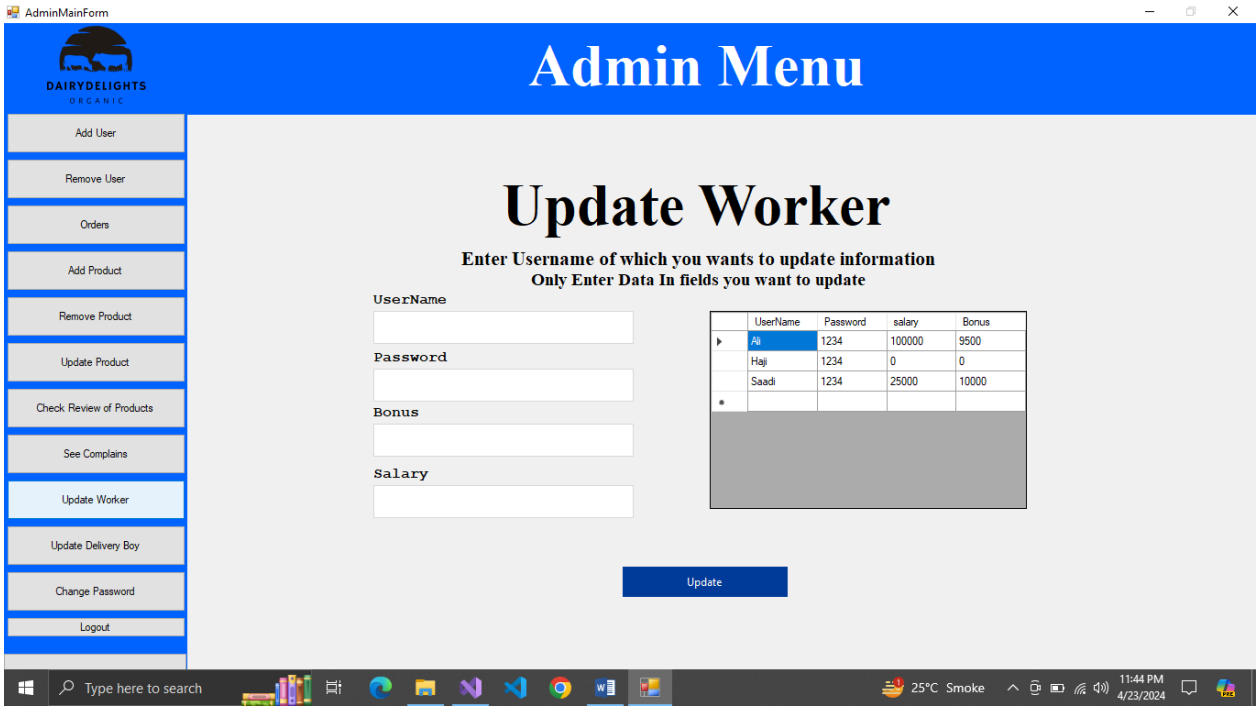


Figure 11Check complains



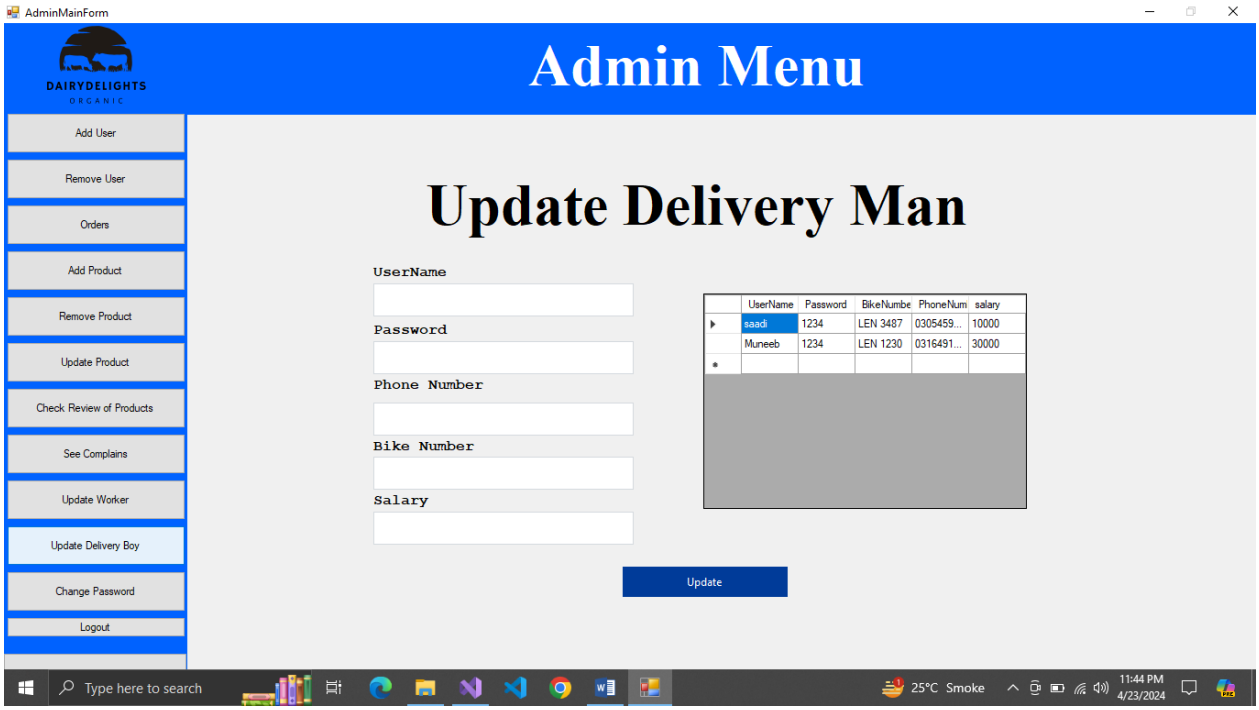


Figure 13Update Delivery Boy

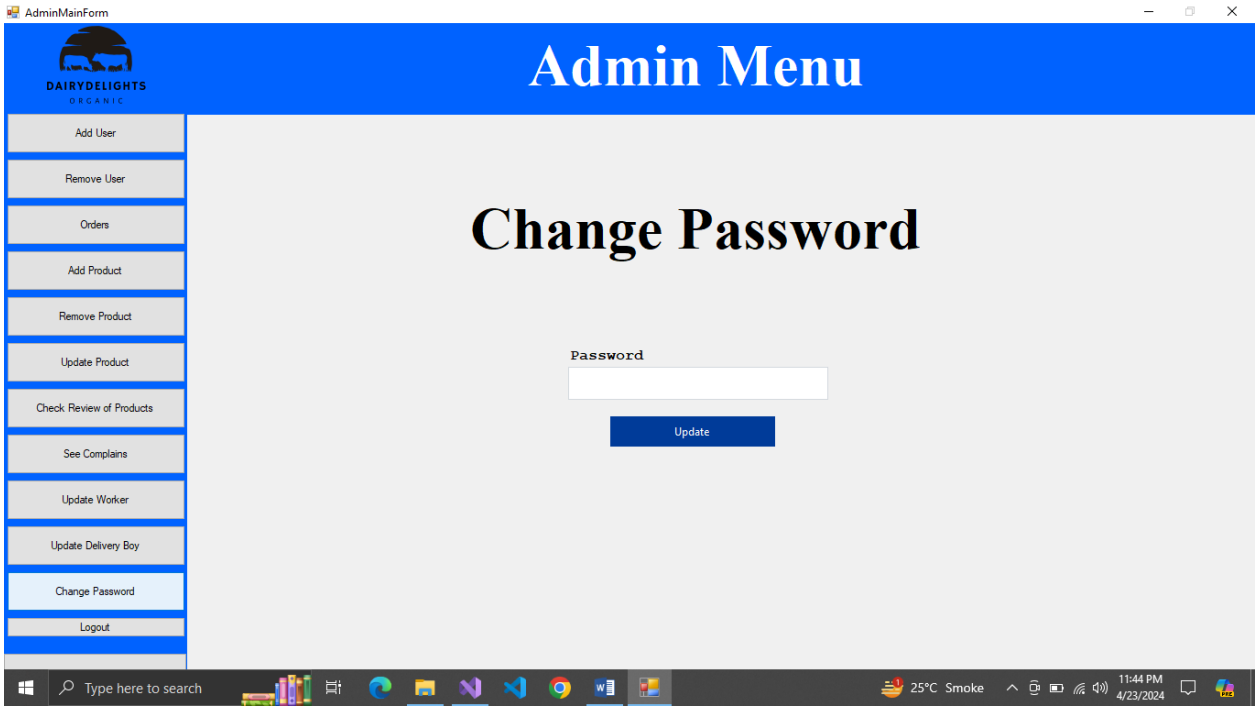
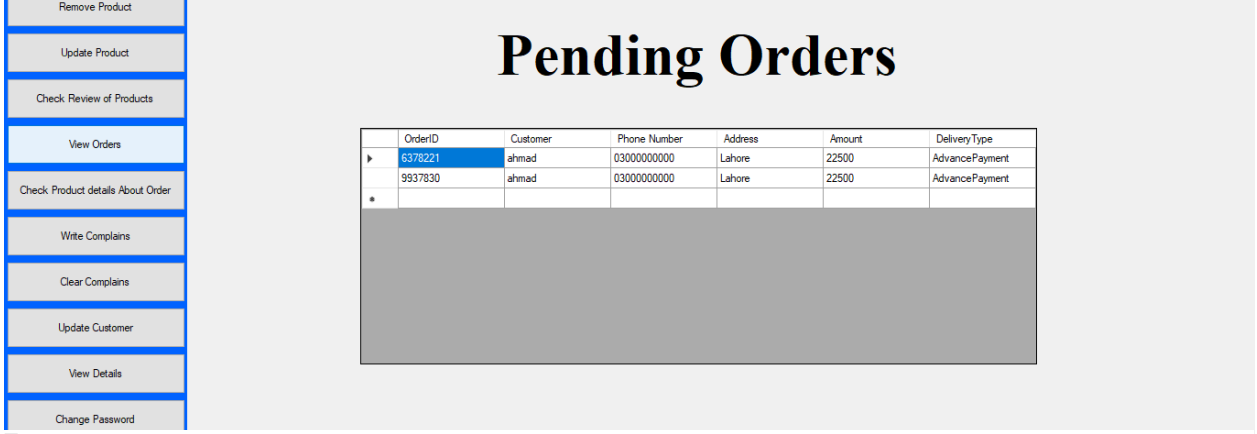


Figure 16Worker Main Menu

Figure 15pending Orders Worker



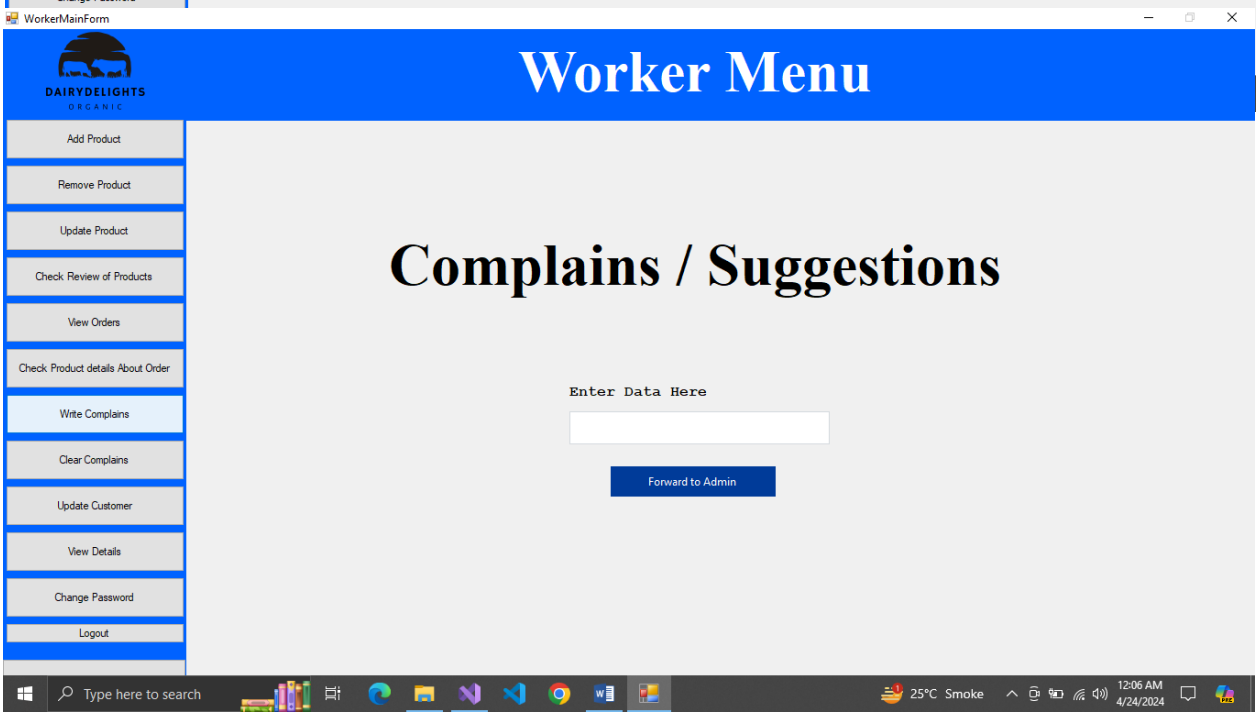


Figure 17Add Complains

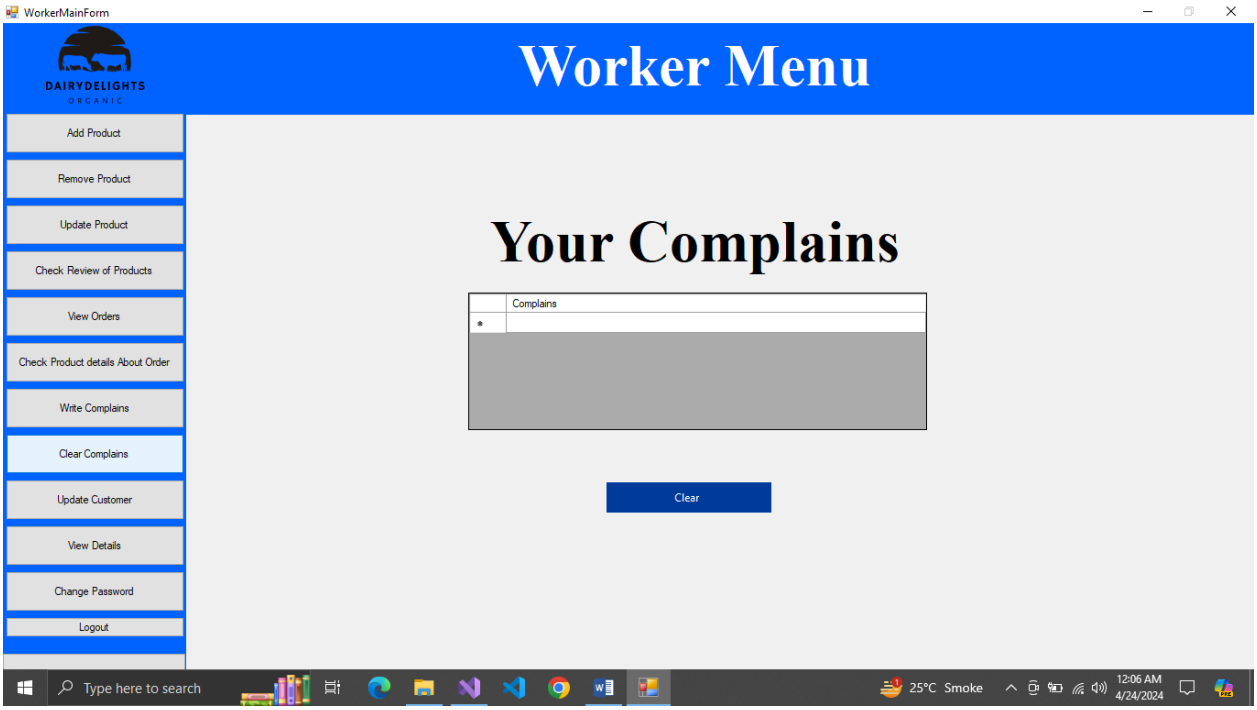


Figure 18view and Delete Complains

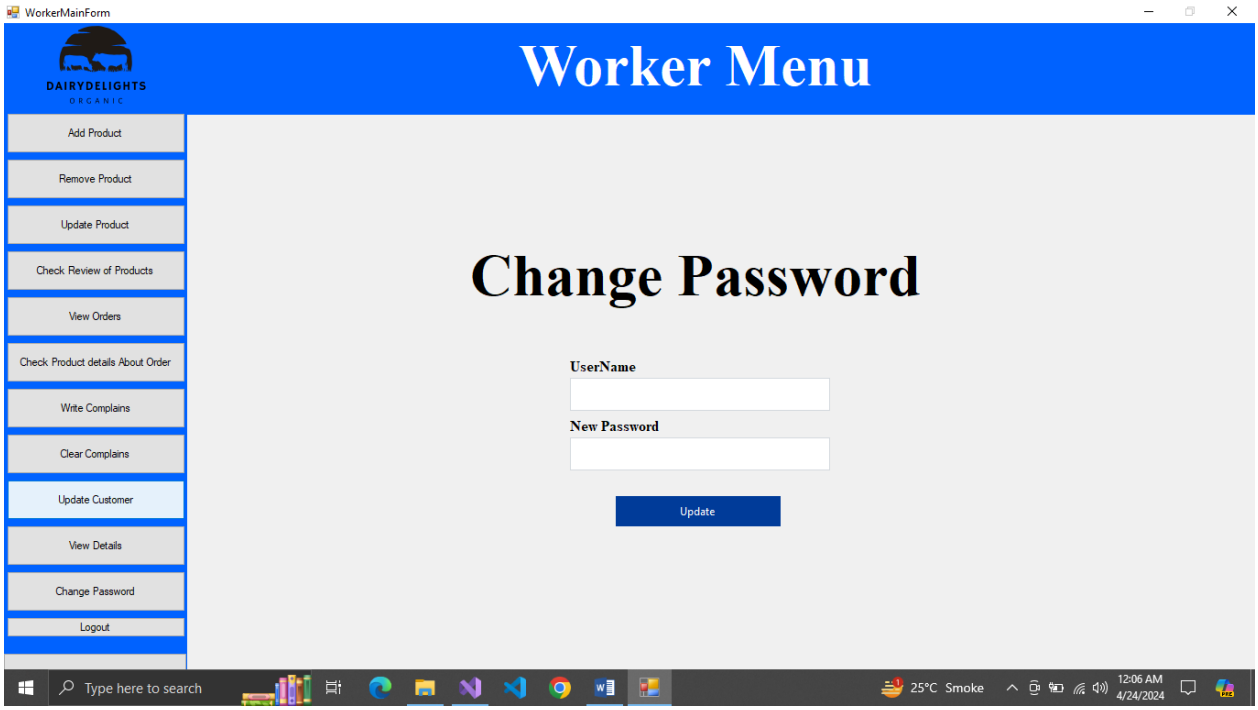


Figure 19Update Customer

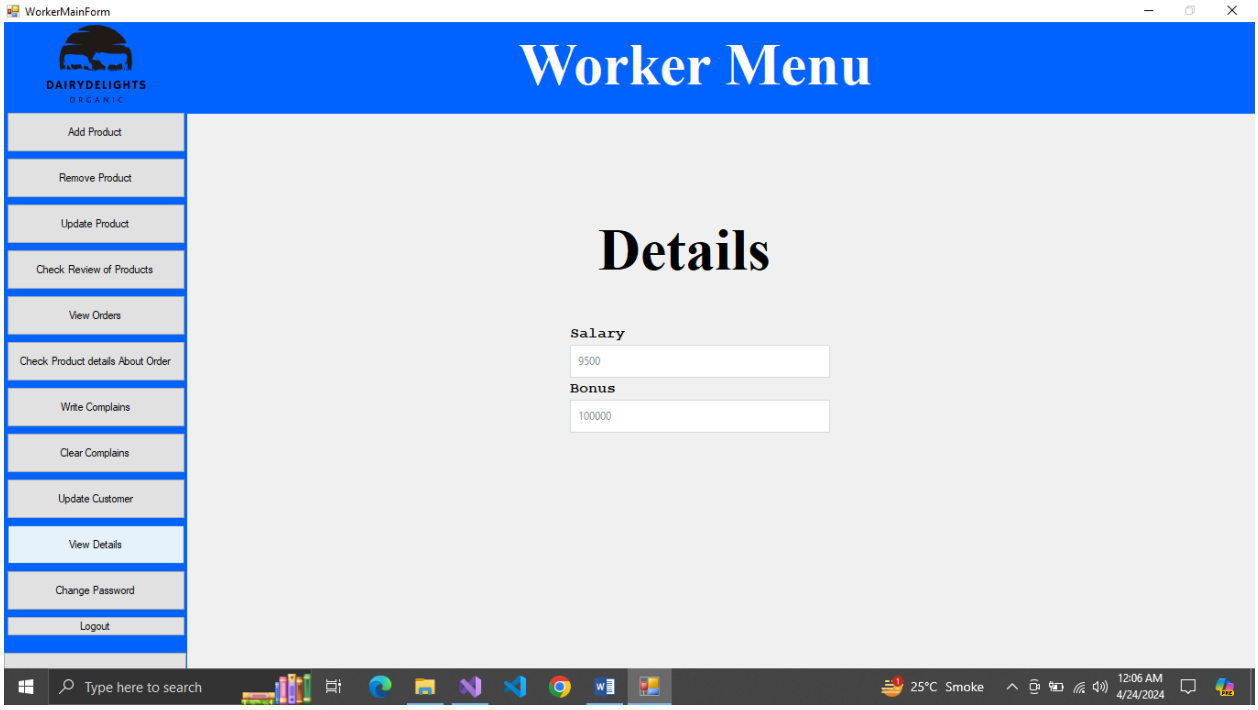


Figure 20View details Worker

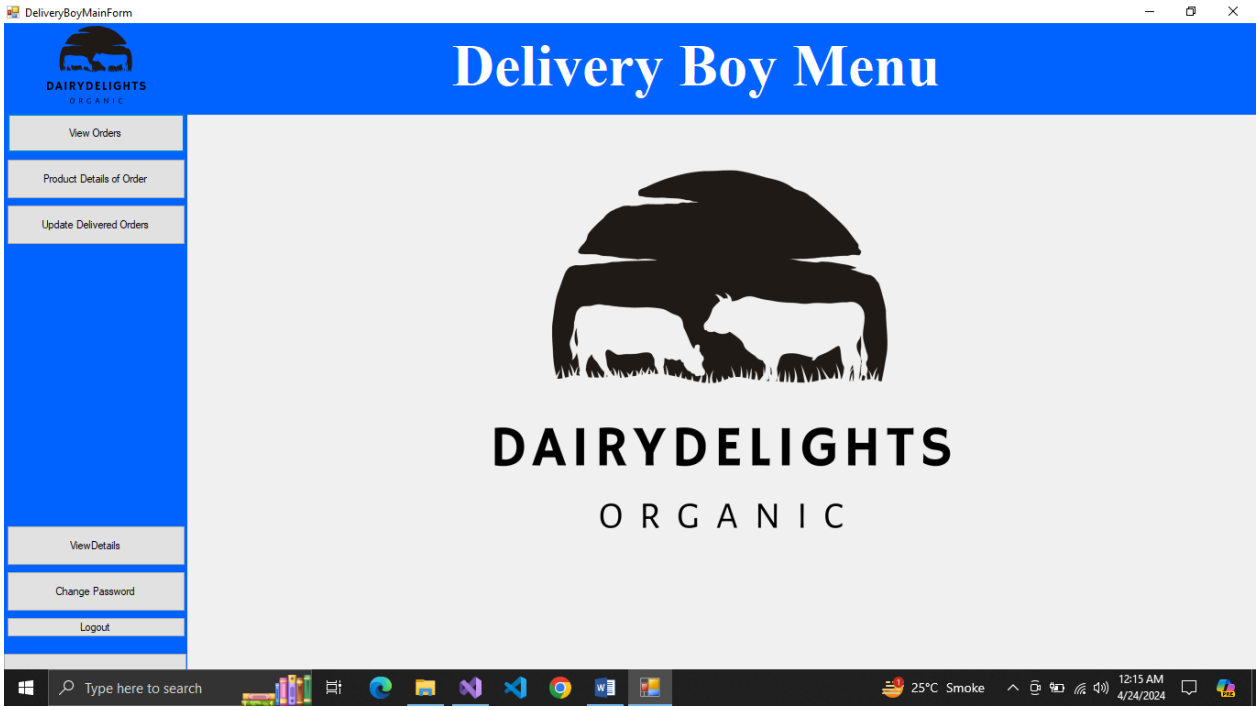


Figure 21Delivery Boy Main Menu

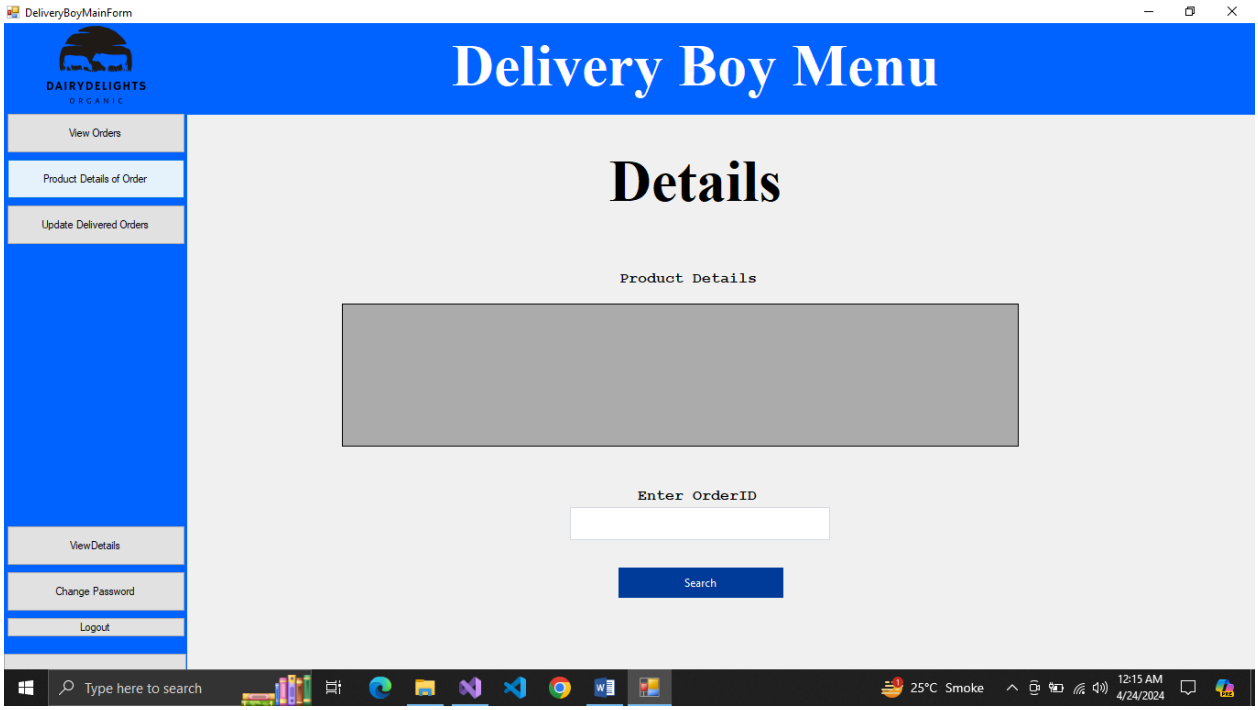


Figure 22View Product details in Order

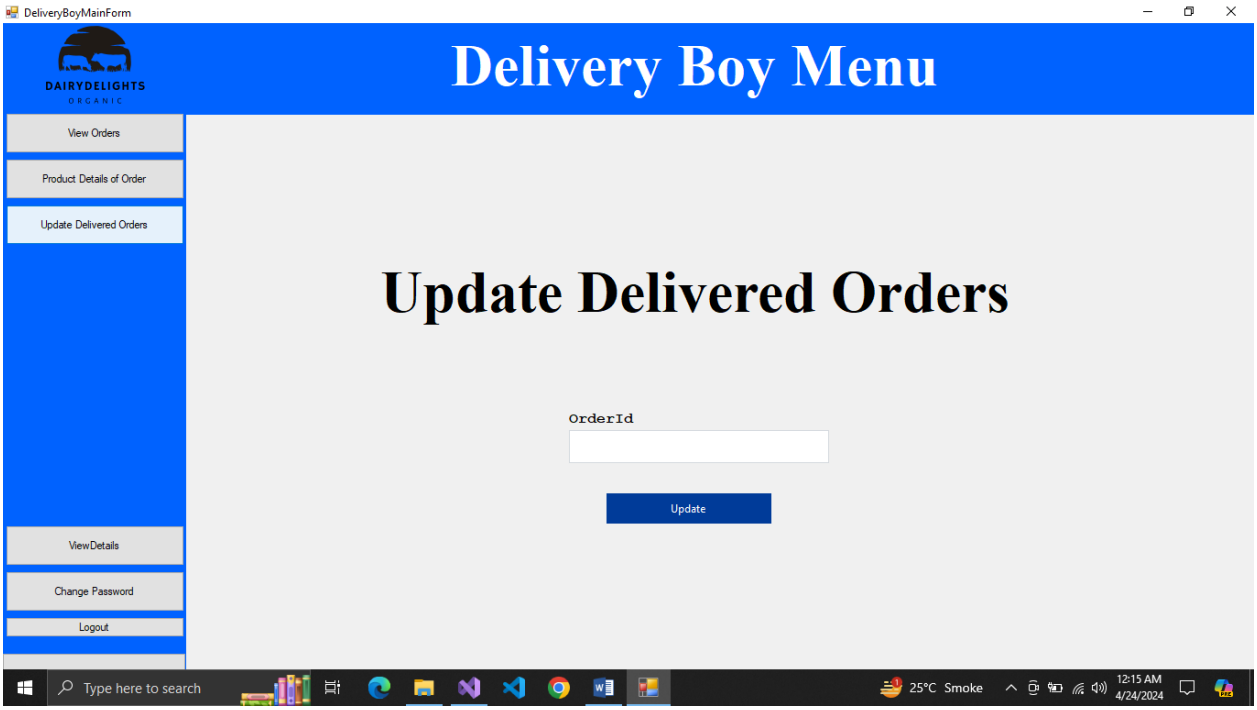


Figure 23Deliver Order

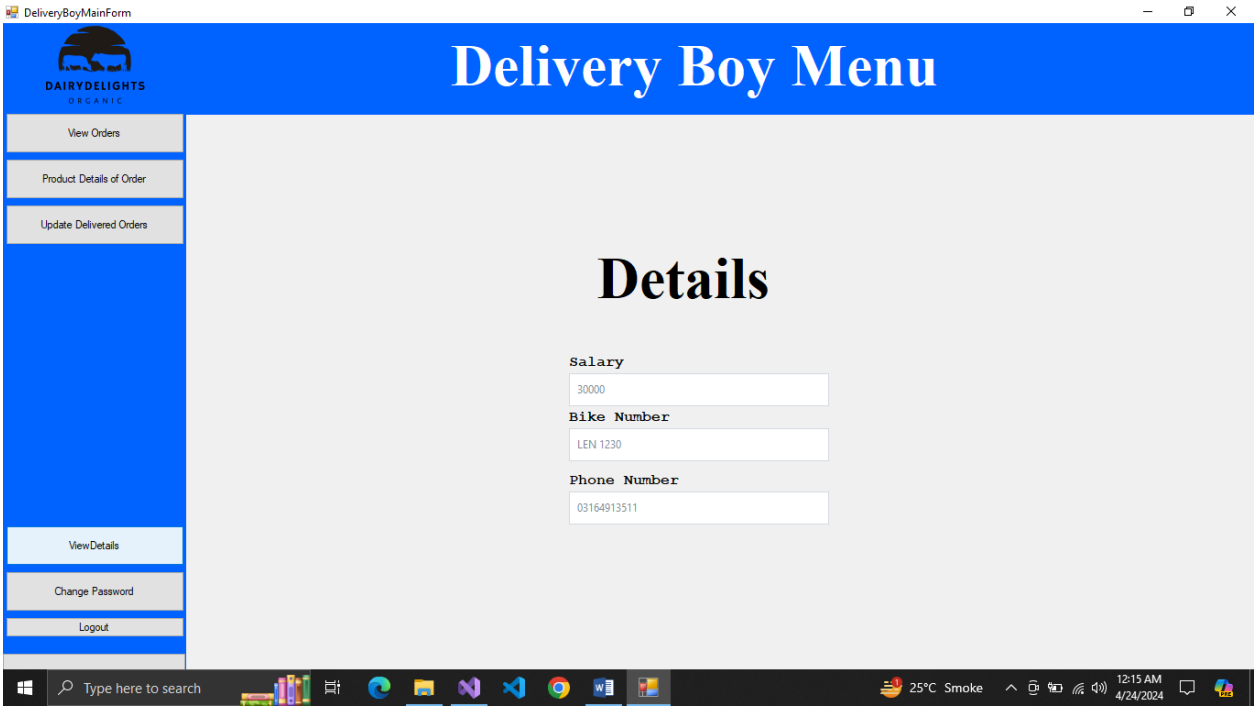


Figure 24View Details Delivery Boy

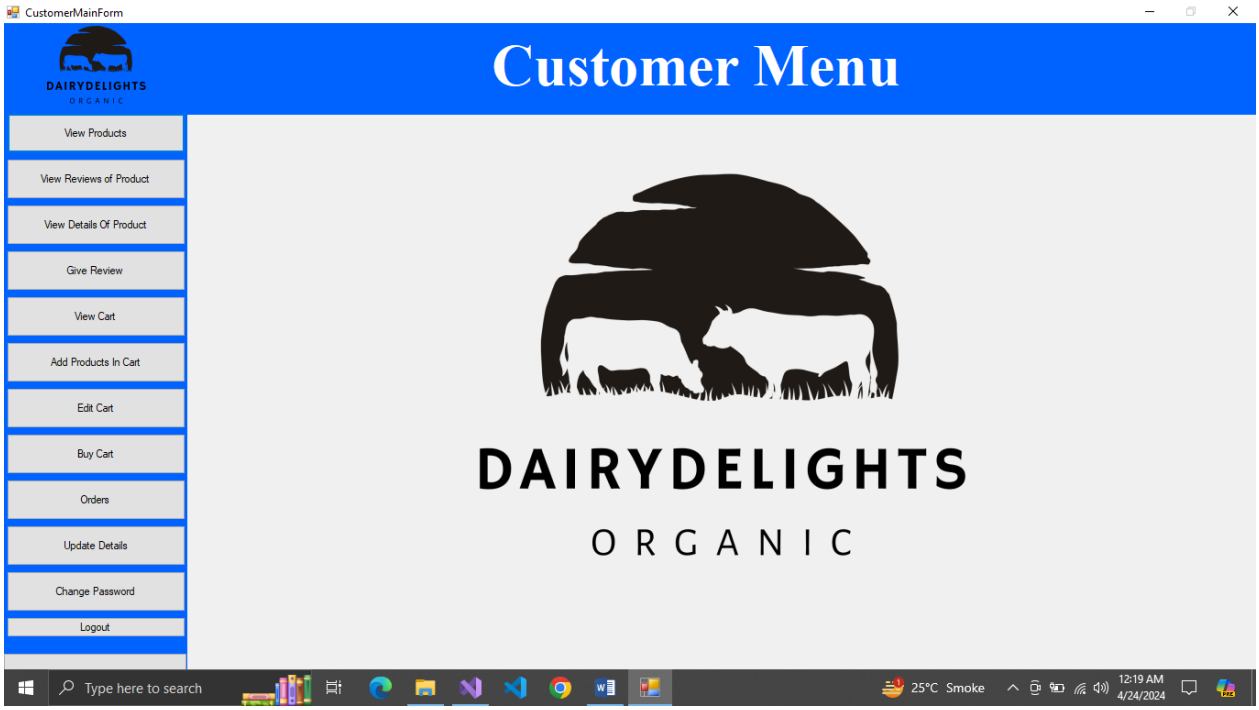


Figure 25Customer Main Page

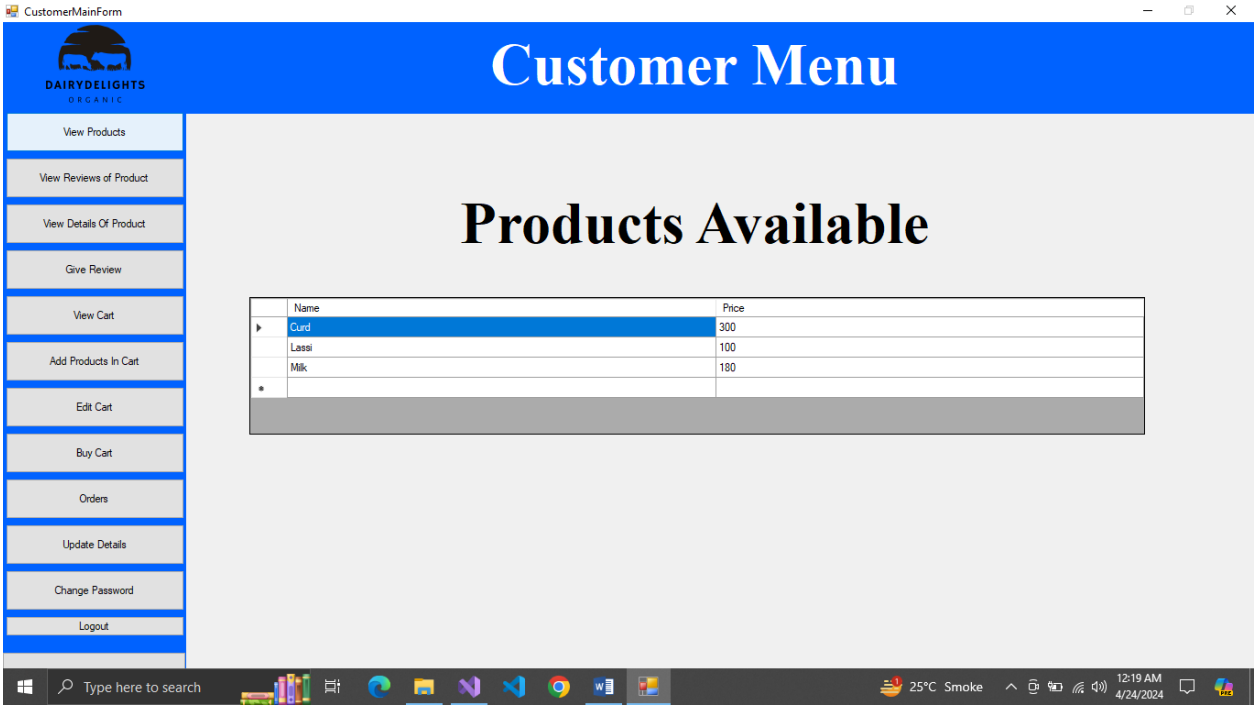


Figure 26View Products Available for Sale

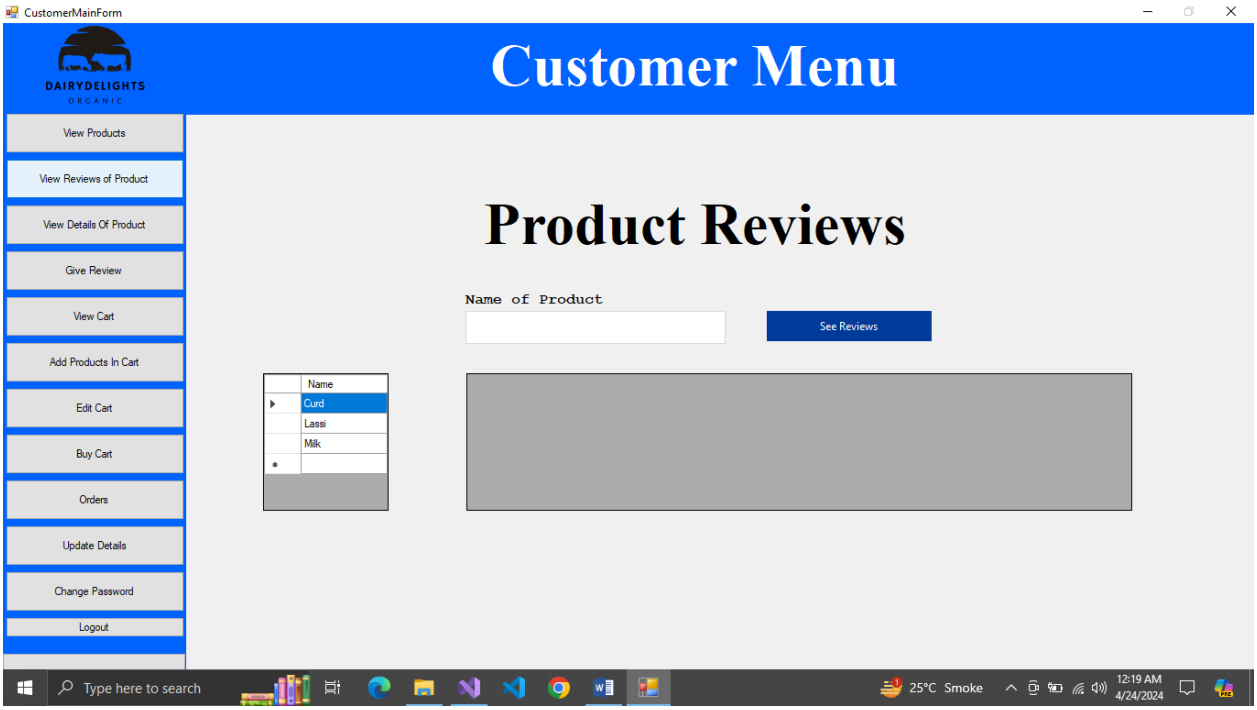


Figure 27View Reviews about Product

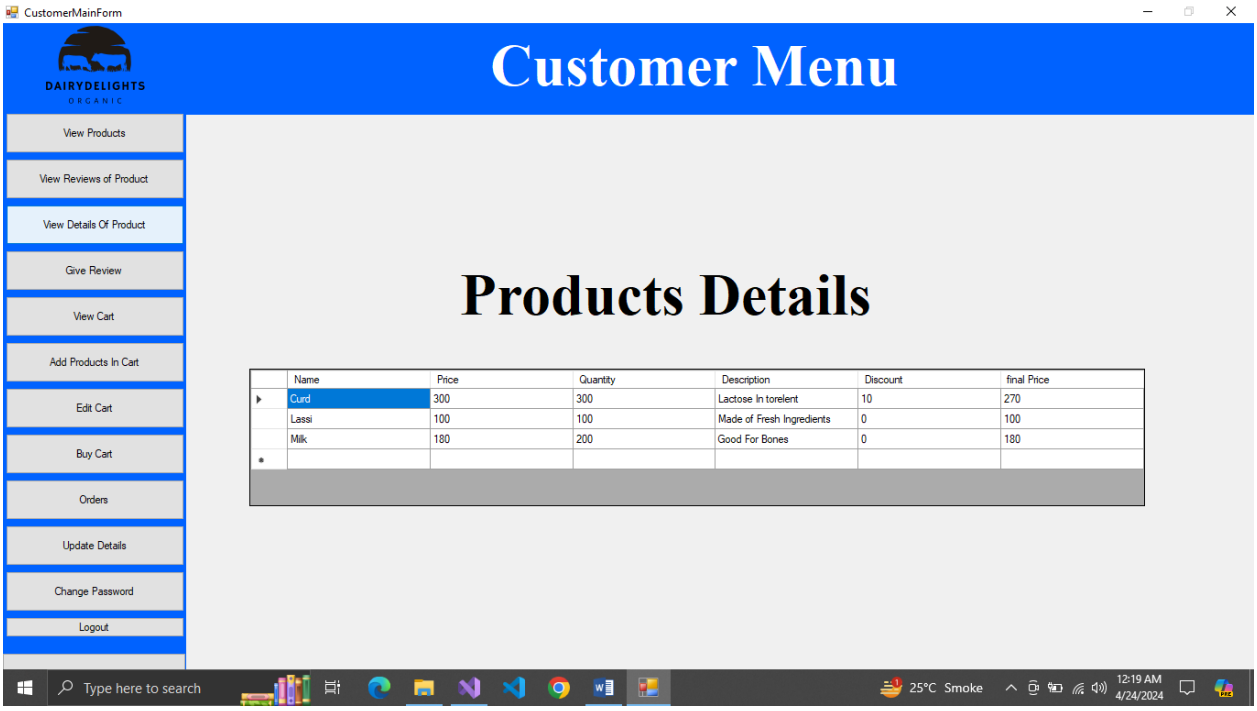


Figure 28View Product Details

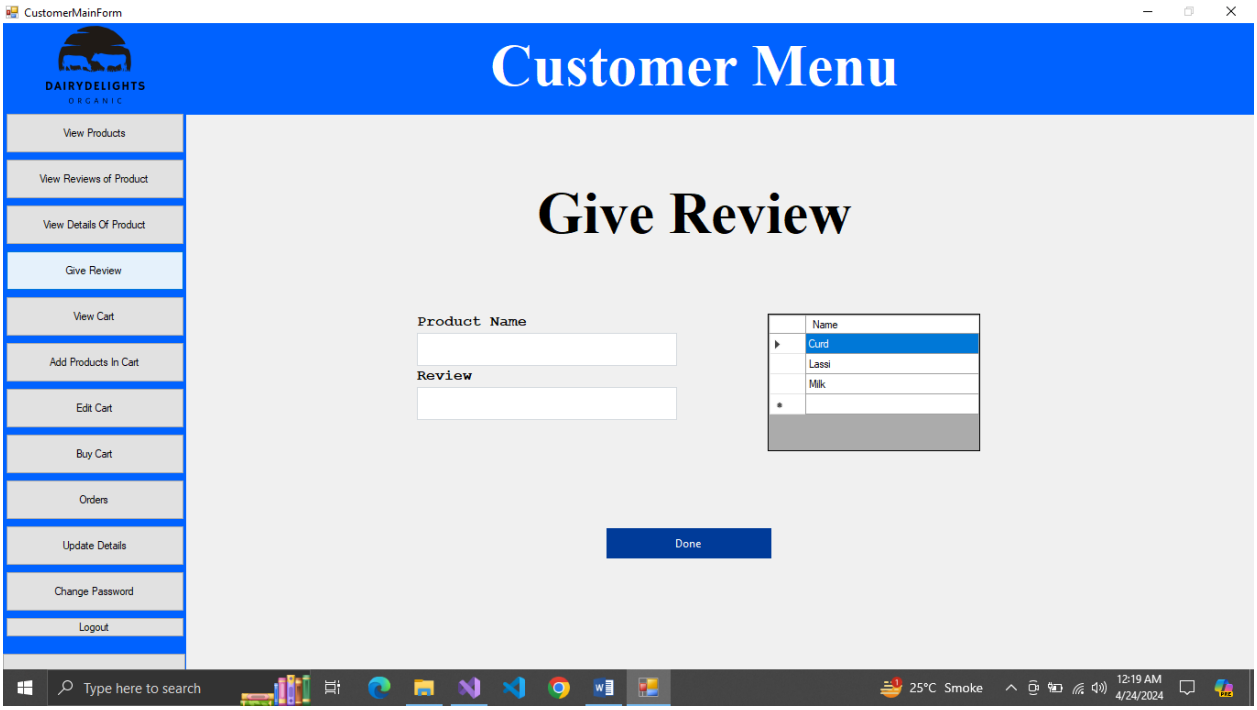


Figure 29Give Review

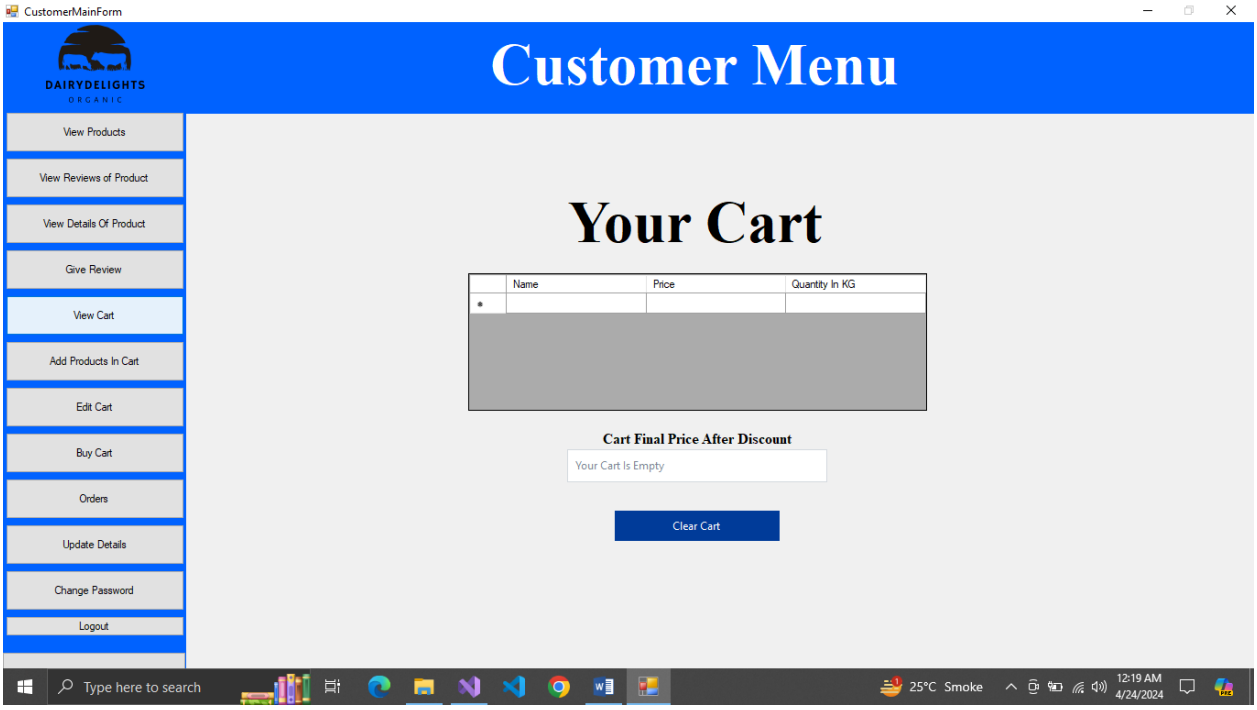


Figure 30View Cart

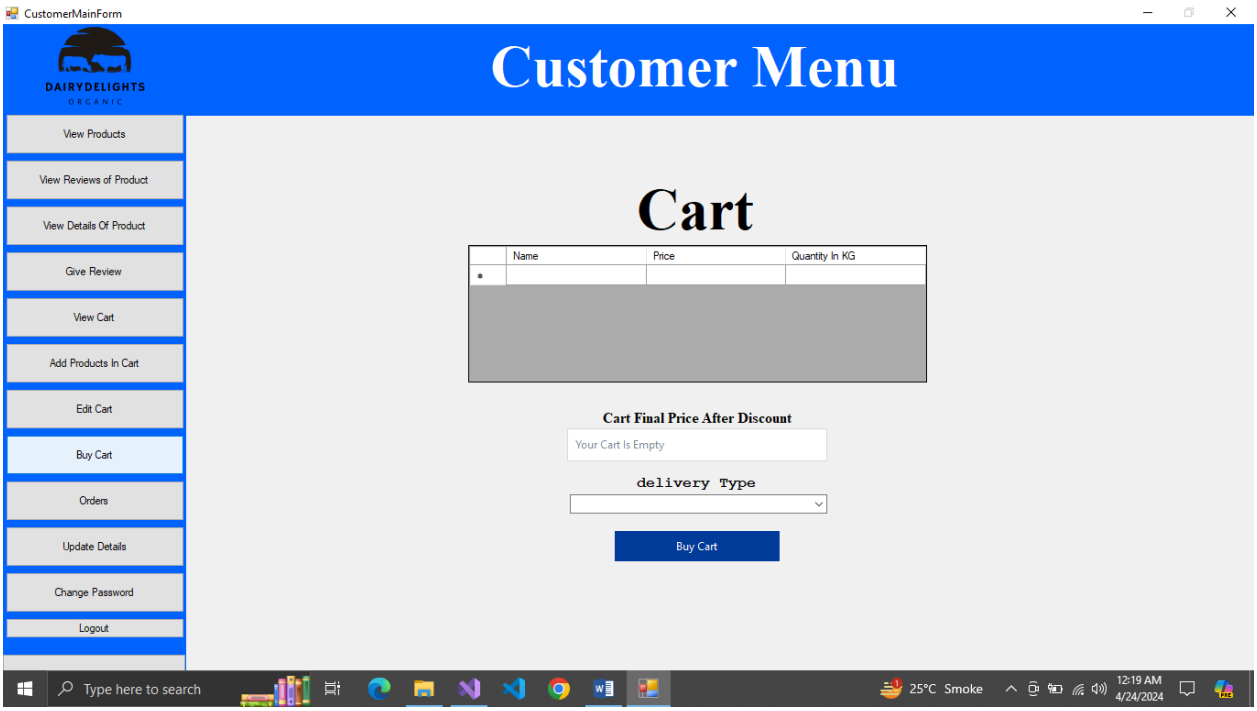


Figure 31Buy Cart

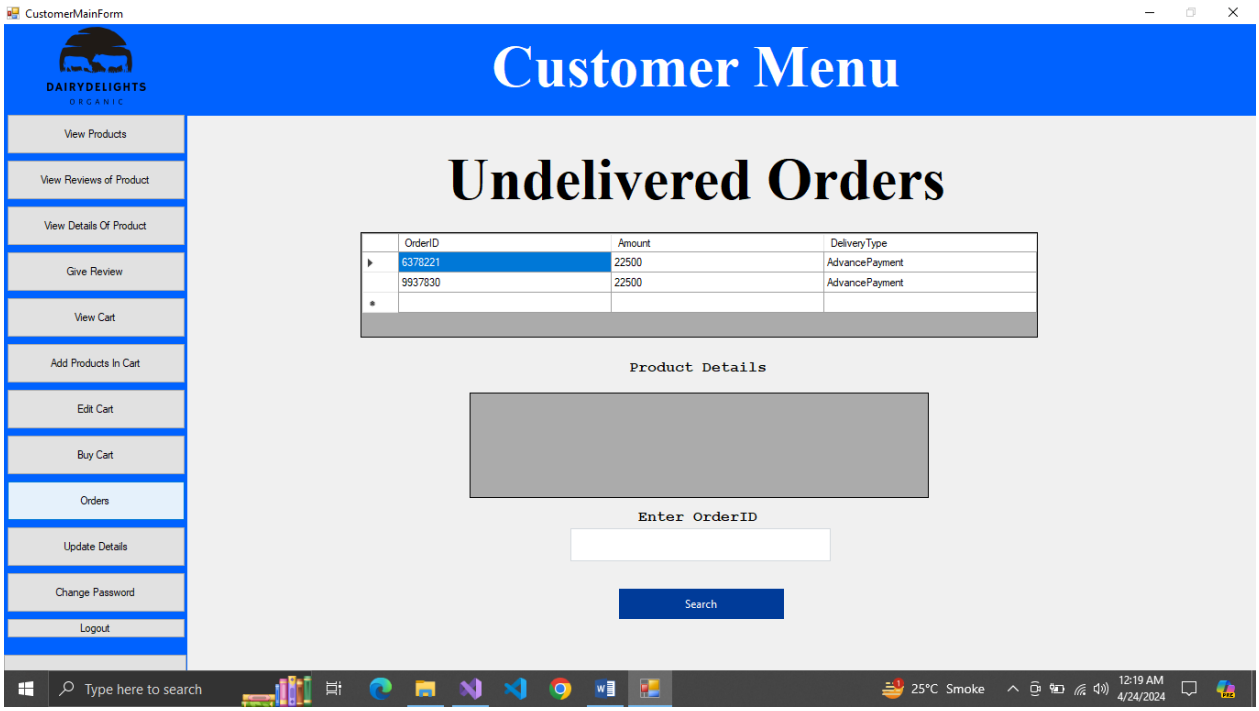


Figure 32View undelivered Orders

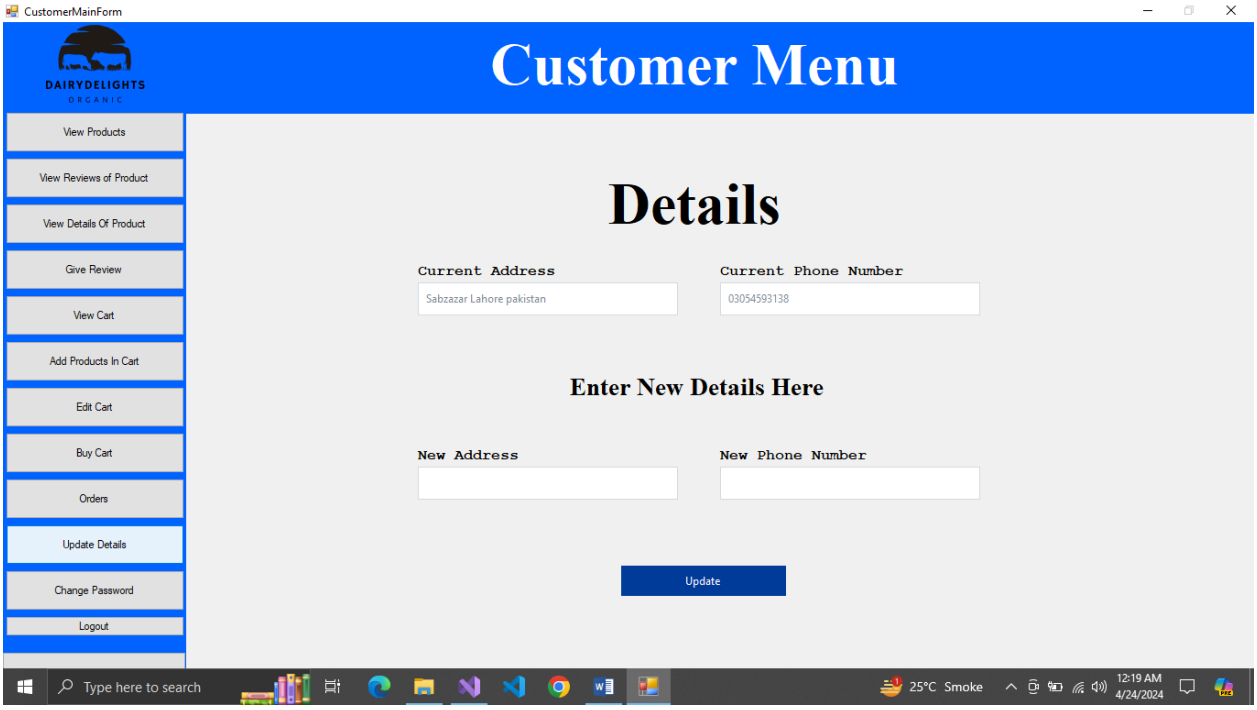


Figure 33Update Customer Details

4. CRC

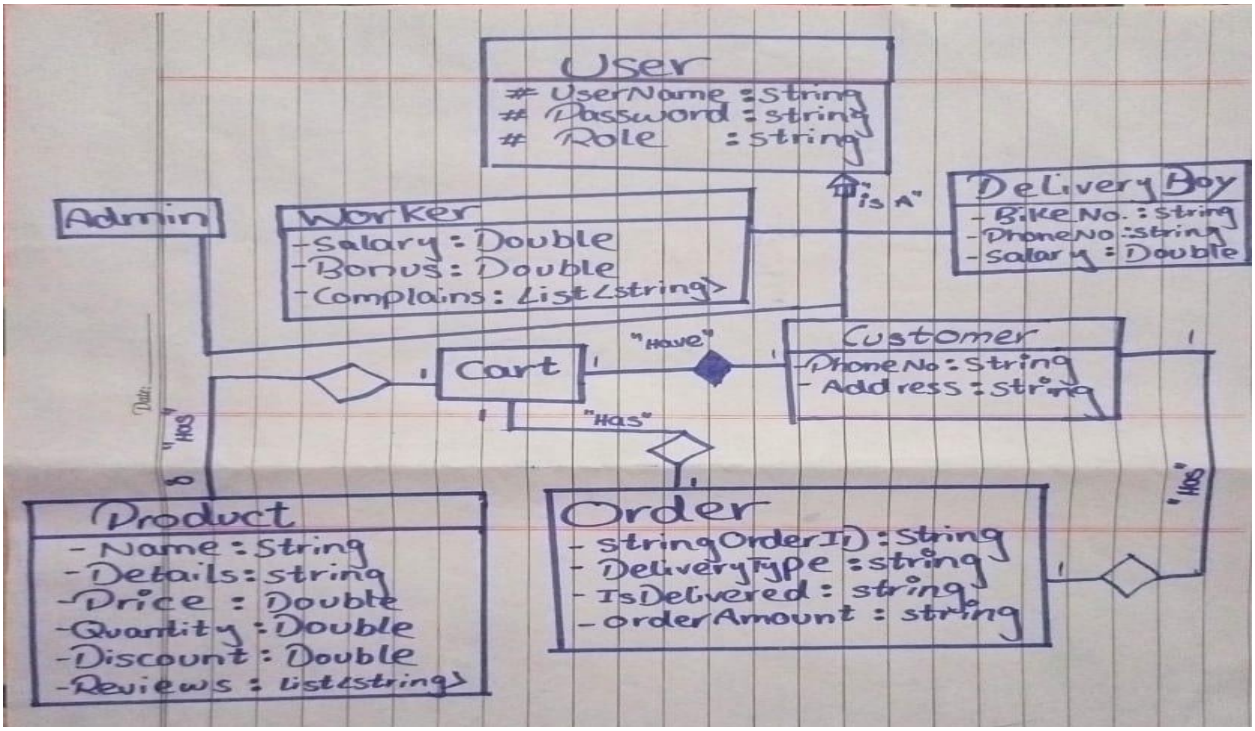


Figure 34CRC

5. Complete Code:

UserBL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DairyDelightsLibrary.BL
{
    public class User
    {
        protected string UserName, Password, Role;   //attribute of the User
        public User()
        { }

        public User(string username)
        {
            this.UserName = username;
        }
        public User(User u)      //copy constructor
        {
            UserName = u.GetUserName();
            Password = u.GetPassword();
            Role = u.GetRole();

        }
        public User(string UserName, string Password)         //parameterized constructor
        {
            this.UserName = UserName;
            this.Password = Password;
            this.Role = "NA";
        }

        public User(string UserName, string Password, string Role)         //parameterized constructor
        {
            this.UserName = UserName;
            this.Password = Password;
            this.Role = Role;
        }

        public string GetRole()
```

```
        {
            return this.Role;
        }
        public bool SetRole(string Role)
        {
            this.Role = Role;
            return true;
        }
        public string GetUserName()
        {
            return this.UserName;
        }
        public bool SetUserName(string UserName)
        {
            this.UserName = UserName;
            return true;
        }
        public string GetPassword()
        {
            return this.Password;
        }
        public bool SetPassword(string Password)
        {
            this.Password = Password;
            return true;
        }
    }
}
```

UserDL With Data Base

```
using DairyDelightsLibrary.BL;
using DairyDelightsLibrary.Utility;
using DairyDelightsLibrary.Interface;

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DairyDelightsLibrary.DL.DataBase
{
    public class UserDL :IUser
    {
```

```
        private string ActiveUser;
        static UserDL Instance;
        private UserDL(string Connection)
        {
            Utility.Connection.SetConnectionString(Connection);

        }
        public static UserDL GetInstance(string Connection)
        {
            if(Instance == null)
            {
                Instance = new UserDL(Connection);
            }
            return Instance;
        }
        public string GetActiveUser()
        {
            return this.ActiveUser;
        }

        public  bool SignUP(User user)
        {
            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();

            string query = string.Format("Insert into Credentials(Username,Password,Role) Values('{0}',
'{1}', '{2}')", user.GetUserName(), user.GetPassword(), user.GetRole());
            SqlCommand cmd = new SqlCommand(query, connection);
            int rows = cmd.ExecuteNonQuery();
            connection.Close();
            ActiveUser = user.GetUserName();

            if (user.GetRole() == "Customer")
            {
                connection.Open();
                string QueryCustomer = string.Format("Insert into Customer(Username) Values('{0}')",
user.GetUserName());
                SqlCommand cmdCustomer = new SqlCommand(QueryCustomer, connection); // Corrected
variable name
                int Rows = cmdCustomer.ExecuteNonQuery(); // Corrected variable name
                connection.Close();
            }
            else if (user.GetRole() == "DeliveryBoy")
            {
                connection.Open();
```

```
            string QueryDeliveryBoy = string.Format("Insert into DeliveryBoy(Username)
Values('{0}')", user.GetUserName());
            SqlCommand cmdDeliveryBoy = new SqlCommand(QueryDeliveryBoy, connection);
            int Rows = cmdDeliveryBoy.ExecuteNonQuery();
            connection.Close();
        }
        else if (user.GetRole() == "Worker")
        {
            connection.Open();
            string QueryWorker = string.Format("Insert into Worker(Username) Values('{0}')",
user.GetUserName());
            SqlCommand cmdWorker = new SqlCommand(QueryWorker, connection);
            int Rows = cmdWorker.ExecuteNonQuery();
            connection.Close();
        }

        return true;
    }
    public  bool RemoveUser(string UserName)
    {
        bool check = false;
        string connectiongString = Connection.GetConnectionString();
        SqlConnection connection = new SqlConnection(connectiongString);
        connection.Open();

        string query = string.Format("Delete from Credentials where Username = '{0}'", UserName);
        SqlCommand cmd = new SqlCommand(query, connection);
        int rows = cmd.ExecuteNonQuery();
        connection.Close();

        string Role = FindRoleByUserName(UserName);

        if (Role == "Customer")
        {
            connection.Open();
            string QueryCustomer = string.Format("Delete from Customer where Username = '{0}'",
UserName);
            SqlCommand cmdCustmer = new SqlCommand(query, connection);
            int Rows = cmd.ExecuteNonQuery();
            connection.Close();
        }
        else if (Role == "DeliveryBoy")
        {
            connection.Open();
            string QueryCustomer = string.Format("Delete from DeliveryBoy where Username = '{0}'",
UserName);
```

```
                SqlCommand cmdCustmer = new SqlCommand(query, connection);
                int Rows = cmd.ExecuteNonQuery();
                connection.Close();
            }
            else if (Role == "Worker")
            {
                connection.Open();
                string QueryCustomer = string.Format("Delete from Worker where Username = '{0}'",
UserName);
                SqlCommand cmdCustmer = new SqlCommand(query, connection);
                int Rows = cmd.ExecuteNonQuery();
                connection.Close();
            }
            if (rows > 0)
            {
                check = true;
            }
                return check;
        }
        public  string SignIN(string UserName, string Password)
        {
            string result = "UserNotFound";

            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();

            string query = string.Format("select Role from Credentials where Username = '{0}' and
Password = '{1}' ",UserName,Password);
            SqlCommand cmd = new SqlCommand(query, connection);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                result = Convert.ToString(reader["Role"]);
                ActiveUser = UserName;
            }

            return result;
        }
        public  string FindRoleByUserName(string UserName)
        {
            string result = "";

            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();
```

```
            string query = string.Format("select Role from Credentials where Username =
'{0}'",UserName);
            SqlCommand cmd = new SqlCommand(query, connection);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                result = Convert.ToString(reader["Role"]);
            }
            return result;
        }
        public string FindPasswordByUserName(string UserName)
        {
            string result = "";

            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();

            string query = string.Format("select Password from Credentials where Username = '{0}'",
UserName);
            SqlCommand cmd = new SqlCommand(query, connection);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                result = Convert.ToString(reader["Password"]);
            }
            return result;
        }

        public bool CheckIfUserNameAlreadyExist(string UserName)
        {
            string result = "";
            bool check = true;

            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();

            string query = string.Format("select Role from Credentials where Username = '{0}'",
UserName);
            SqlCommand cmd = new SqlCommand(query, connection);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                result = Convert.ToString(reader["Role"]);
```

```
            }
            if (result == "")
            {
                check = false;
            }
            return check;
        }
        public bool ChangePassword(string Password)
        {
            bool Result = false;
            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();
            string query = string.Format("UPDATE Credentials SET Password = '{0}' WHERE Username
= '{1}'", Password, ActiveUser);
            SqlCommand cmd = new SqlCommand(query, connection);
            int rows = cmd.ExecuteNonQuery();
            connection.Close();
            if(rows > 0)
            {
                Result = true;
            }
            return Result;

        }
        public bool ChangePasswordByUserName(string Username,string Password)
        {
            bool Result = false;
            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();
            string query = string.Format("UPDATE Credentials SET Password = '{0}' WHERE Username
= '{1}'", Password, Username);
            SqlCommand cmd = new SqlCommand(query, connection);
            int rows = cmd.ExecuteNonQuery();
            connection.Close();
            if (rows > 0)
            {
                Result = true;
            }
            return Result;

        }
        public List<User> GetUsersList()
        {
            List<User> users = new List<User>();
```

```
            string connectiongString = Connection.GetConnectionString();
            SqlConnection connection = new SqlConnection(connectiongString);
            connection.Open();
            string Query = string.Format("select * from Credentials");
            SqlCommand Command = new SqlCommand(Query, connection);
            SqlDataReader reader = Command.ExecuteReader();
            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);
                string password = Convert.ToString(reader["Password"]);
                string role = Convert.ToString(reader["Role"]);

                User user = new User(username, password, role); //object is made
                users.Add(user);
            }
            connection.Close();
            return users;

        }


    }
}
```

User DL with file FileHandling

```
using DairyDelightsLibrary.BL;
using DairyDelightsLibrary.Interface;
using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Threading;

namespace DairyDelightsLibrary.DL.FileHandling
{
    public class UserDL : IUser
    {
        public static string ActiveUser;
        static UserDL Instance;
        public List<User> UsersList = new List<User>();
        private UserDL(string path)
        {
            Utility.Path.SetUserFilePath(path);
            UsersList = GetDataFromUserFile(path);
        }
        public static UserDL GetInstance(string path)
```

```
{
    if (Instance == null)
    {
        Instance = new UserDL(path);
    }
    return Instance;
}

public string SignIN(string UserName, string Password)
{
    foreach (User user in UsersList)
    {
        if (UserName == user.GetUserName() && Password == user.GetPassword())
        {
            ActiveUser = UserName;
            return user.GetRole();
        }
    }
    return "UserNotFound";
}

public bool SignUP(User user)
{
    UsersList.Add(user);
    ActiveUser = user.GetUserName();
    StoreDetailsOfUser();
    return true;
}

public string GetActiveUser()
{
    return ActiveUser;
}

public bool ChangePassword(string Password)
{
    foreach (User user in UsersList)
    {
        if (ActiveUser == user.GetUserName())
        {
            bool check = user.SetRole(Password);
            StoreDetailsOfUser();
            return check;
        }
    }
    return false;
```

```
}

public bool ChangePasswordByUserName(string Username, string Password)
{
    foreach (User user in UsersList)
    {
        if (Username == user.GetUserName())
        {
            bool check = user.SetRole(Password);
            return check;
        }
    }
    return false;
}

public bool CheckIfUserNameAlreadyExist(string UserName)
{
    if(UsersList == null)
    {
        return false;
    }
    foreach (User user in UsersList)
    {
        if (UserName == user.GetUserName())
        {
            return true;
        }
    }
    return false;
}

public string FindPasswordByUserName(string UserName)
{
    foreach (User user in UsersList)
    {
        if (UserName == user.GetUserName())
        {
            return user.GetPassword();
        }
    }
    return "";
}

public string FindRoleByUserName(string UserName)
{
    foreach (User user in UsersList)
```

```
        {
            if (UserName == user.GetUserName())
            {
                return user.GetRole();
            }
        }
        return "";
    }

    public List<User> GetUsersList()
    {
        return this.UsersList;
    }

    public bool RemoveUser(string UserName)
    {
        foreach (User user in UsersList)
        {
            if (UserName == user.GetUserName())
            {
                UsersList.Remove(user);
                StoreDetailsOfUser();
                return true;
            }
        }
        return false;
    }


    private string parseData(string record, int field)
    {
        int comma = 1;
        string item = "";
        for (int x = 0; x < record.Length; x++)
        {
            if (record[x] == '%')
            {
                comma++;
            }
            else if (comma == field)
            {
                item = item + record[x];
            }
        }
        return item;
    }
```

```
        private List<User> GetDataFromUserFile(string path)
        {
            List<User> users = new List<User>();
            if (File.Exists(path))
            {
                StreamReader fileVariable = new StreamReader(path);
                string record;
                while ((record = fileVariable.ReadLine()) != null)
                {
                    string userName = parseData(record, 1);
                    string userPassword = parseData(record, 2);
                    string userRole = parseData(record, 3);

                    User user = new User(userName, userPassword, userRole);
                    users.Add(user);

                }
                fileVariable.Close();
            }
            return users;
        }
        private void StoreDetailsOfUser()
        {
            string path = Utility.Path.GetUserFilePath();

            StreamWriter file = new StreamWriter(path, append :false);

            for (int i = 0; i < UsersList.Count; i++)
            {
                User user1 = UsersList[i];
                file.WriteLine($"{user1.GetUserName()}%{user1.GetPassword()}%{user1.GetRole()}%");
            }
            file.Flush();
            file.Close();

        }


    }
}
```

UserUI

```
using DairyDelightsLibrary.BL;
using DairyDelightsLibrary.Validation;
using System;
using System.Data;
```

```
using System.Security.Cryptography;

namespace ConsoleAppProject.UI.Menu
{
    internal class GetInformationMenus
    {
        public static string MainMenu()
        {
            Console.Clear();
            string option = "5";
            Console.WriteLine("Welcome");
            Console.WriteLine("1. Sign In");
            Console.WriteLine("2.sign Up");
            Console.WriteLine("3. Exit");
            Console.Write("Enter Your Option : ");
            option = Console.ReadLine();

            return option;
        }
        public static User SignIN()
        {
            Console.Clear();
            Console.Write("Enter UserName : ");
            string username = Console.ReadLine();
            Console.Write("Enter Password : ");
            string password = Console.ReadLine();

            User user = new User(username, password);
            return user;
        }
        public static User SignUp()
        {

            Console.Clear();
            Console.Write("Enter UserName : ");
            string username = Console.ReadLine();
            Console.Write("Enter Password : ");
            string password = Console.ReadLine();
            string role = "Customer";
            if (username != "" && password != "")
            {
                if (UserValidation.IsStringValid(username) && UserValidation.IsStringValid(password))
                {
                    User user = new User(username, password, role);
                    return user;
                }
```

```
            }
            return null;
        }
        public static string AdminMenu()
        {
            Console.Clear();
            string option;
            Console.WriteLine("
********************************************************** Admin Menu
**********************************************************");
            Console.WriteLine();
            Console.WriteLine("1. Add New Employee");
            Console.WriteLine("2. Remove Employee");
            Console.WriteLine("3. Change Others Password");
            Console.WriteLine("4. Update Password");
            Console.WriteLine("5. Logout");
            Console.Write("Enter Your Option Here: ");
            option = Console.ReadLine();

            return option;
        }
        public static User AddUser()
        {
            Console.Clear();
            Console.Write("Enter UserName : ");
            string username = Console.ReadLine();
            Console.Write("Enter Password : ");
            string password = Console.ReadLine();
            Console.Write("Enter Role (Worker or DeliveryBoy): ");
            string role = Console.ReadLine();

            if (username != "" && password != "" && role != "")
            {
                if (UserValidation.IsStringValid(username) && UserValidation.IsStringValid(password))
                {
                    if(role == "Worker" ||role == "DeliveryBoy")
                    {
                        User user = new User(username, password, role);
                        return user;
                    }
                }
            }
            return null;
        }
        public static void DisplayAllUsersOnScreen(List<User> users)
        {
```

```
            foreach(User user in users)
            {
                Console.WriteLine("UserName = " + user.GetUserName() + ", Password = " +
user.GetPassword() + ", Role = " + user.GetRole());
            }
        }
        public static string EnterUserName()
        {
            Console.Write("Enter UserName: ");
            string id = Console.ReadLine();
            return id;
        }
        public static string EnterPassword()
        {
            Console.Write("Enter Password: ");
            string Password = Console.ReadLine();
            return Password;
        }
        public static string EnterNewPassword()
        {
            Console.Write("Enter New Password: ");
            string Password = Console.ReadLine();
            return Password;
        }
        public static string EnterRole()
        {
            Console.Write("Enter Role (Worker or DeliveryBoy): ");
            string Role = Console.ReadLine();
            return Role;
        }
        public static void ClearScreen()
        {
            Console.Clear();
        }
        public static void Successfull()
        {
            Console.WriteLine("Operation Successfull :)");
            Console.ReadKey();
        }
        public static void Unsuccessfull()
        {
            Console.WriteLine("Operation Unsuccessfull :)");
            Console.ReadKey();
        }
    }
}
```

ObjectHandler

```
using DairyDelightsLibrary.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DairyDelightsLibrary.BL;
using DairyDelightsLibrary.Validation;
using System.IO;
using DairyDelightsLibrary.Utility;
using DairyDelightsLibrary.DL.DataBase;
//using DairyDelightsLibrary.DL.FileHandling;

namespace ConsoleAppProject
{
    public class ObjectHandler
    {
        static string path = "UserDetails.txt";
        static string Connection = "Data Source=MALIK;Initial Catalog=Final_Project;Integrated
Security=True;";
        public static IUser GetUserInstance()
        {
            //IUser user = UserDL.GetInstance(path);
            IUser user = UserDL.GetInstance(Connection);

            return user;
        }
    }
}
```

Driver Program

```
using DairyDelightsLibrary.BL;
using DairyDelightsLibrary.Interface;
using DairyDelightsLibrary.Validation;
using System;


namespace ConsoleAppProject
{
    internal class Program
    {
        static void Main(string[] args)
        {
            IUser user = ObjectHandler.GetUserInstance();
```

```
while (true)
{
    string option = UI.Menu.GetInformationMenus.MainMenu();
    if (option == "1")
    {
        User OldUser = UI.Menu.GetInformationMenus.SignIN();
        string Role = user.SignIN(OldUser.GetUserName(), OldUser.GetPassword());
        if (Role == "Admin")
        {
            while (true)
            {
                string AdminOption = UI.Menu.AdminMenuUI.AdminMenu();
                if (AdminOption == "1")
                {
                    User NewEmployee = UI.Menu.AdminMenuUI.AddUser();
                    if(NewEmployee != null)
                    {
                        bool check = user.SignUP(NewEmployee);
                        if (check)
                        {
                            UI.Menu.GetInformationMenus.Successfull();
                            continue;
                        }
                    }
                    else
                    {
                        UI.Menu.GetInformationMenus.Unsuccessfull();
                        continue;
                    }

                }
                else if (AdminOption == "2")
                {
                    UI.Menu.AdminMenuUI.DisplayAllUsersOnScreen(user.GetUsersList());

                    string UserName = UI.Menu.GetInformationMenus.EnterUserName();
                    if(user.CheckIfUserNameAlreadyExist(UserName))
                    {
                        bool check = user.RemoveUser(UserName);
                        if (check)
                        {
                            UI.Menu.GetInformationMenus.Successfull();
                            continue;
                        }
                    }
                    else
```

```
        {
            UI.Menu.GetInformationMenus.Unsuccessfull();
            continue;
        }

    }
    else if (AdminOption == "3")
    {
        UI.Menu.AdminMenuUI.DisplayAllUsersOnScreen(user.GetUsersList());
        string UserName = UI.Menu.GetInformationMenus.EnterUserName();
        string NewPassword = UI.Menu.GetInformationMenus.EnterNewPassword();
        if (user.CheckIfUserNameAlreadyExist(UserName))
        {
            bool check = user.ChangePasswordByUserName(UserName, NewPassword);
            if (check)
            {
                UI.Menu.GetInformationMenus.Successfull();
                continue;
            }
        }
        else
        {
            UI.Menu.GetInformationMenus.Unsuccessfull();
            continue;
        }


    }
    else if (AdminOption == "4")
    {
        string NewPassword = UI.Menu.GetInformationMenus.EnterNewPassword();
        if(UserValidation.IsStringValid(NewPassword))
        {
            bool check = user.ChangePassword(NewPassword);
            if (check)
            {
                UI.Menu.GetInformationMenus.Successfull();
                continue;
            }
        }
        else
        {
            UI.Menu.GetInformationMenus.Unsuccessfull();
            continue;
        }
```

```
            }
            else if (AdminOption == "5")
            {
                break;
            }
            else
            {
                continue;
            }
        }
    }
    else if (Role == "Worker")
    {
        while (true)
        {
            string optionworker = UI.Menu.WorkerMenuUI.WorkerMenu();
            if(optionworker == "1")
            {
                string NewPassword = UI.Menu.GetInformationMenus.EnterNewPassword();
                if (UserValidation.IsStringValid(NewPassword))
                {
                    bool check = user.ChangePassword(NewPassword);
                    if (check)
                    {
                        UI.Menu.GetInformationMenus.Successfull();
                        continue;
                    }
                }
                else
                {
                    UI.Menu.GetInformationMenus.Unsuccessfull();
                    continue;
                }

            }
            else if(optionworker == "2")
            {
                break;
            }
            else
            {
                continue;
            }


        }
```

```
}
else if (Role == "Delivery Boy")
{
    while (true)
    {
        string optionWorker = UI.Menu.DeliveryBoyMenuUI.DeliveryBoyMenu();
        if (optionWorker == "1")
        {
            string NewPassword = UI.Menu.GetInformationMenus.EnterNewPassword();
            if (UserValidation.IsStringValid(NewPassword))
            {
                bool check = user.ChangePassword(NewPassword);
                if (check)
                {
                    UI.Menu.GetInformationMenus.Successfull();
                    continue;
                }
            }
            else
            {
                UI.Menu.GetInformationMenus.Unsuccessfull();
                continue;
            }

        }
        else if (optionWorker == "2")
        {
            break;
        }
        else
        {
            continue;
        }

    }
}
else if (Role == "Customer")
{
    while (true)
    {
        string optionCustomer = UI.Menu.CustomerMenuUI.CustomerMenu();
        if (optionCustomer == "1")
        {
            string NewPassword = UI.Menu.GetInformationMenus.EnterNewPassword();
            if (UserValidation.IsStringValid(NewPassword))
```

```
                {
                    bool check = user.ChangePassword(NewPassword);
                    if (check)
                    {
                        UI.Menu.GetInformationMenus.Successfull();
                        continue;
                    }
                }
                else
                {
                    UI.Menu.GetInformationMenus.Unsuccessfull();
                    continue;
                }

            }
            else if (optionCustomer == "2")
            {
                break;
            }
            else
            {
                continue;
            }

        }
    }
    else
    {
        UI.Menu.GetInformationMenus.Unsuccessfull();
        continue;
    }
}
else if (option == "2")
{
    User NewUser = UI.Menu.GetInformationMenus.SignUp();
    if (NewUser != null)
    {
        bool check = user.SignUP(NewUser);
        if (check)
        {
            UI.Menu.GetInformationMenus.Successfull();
        }
    }
    else
    {
        UI.Menu.GetInformationMenus.Unsuccessfull();
```

```
                continue;
            }

        }
        else if (option == "3")
        {
            break;
        }
        else
        {
            continue;
        }

    }
  }
 }
}
```

6. Weakness in Project:

a. The Delivery system is not Good the Customer do not know when will his order arrive and do not know anyone to contact for order
b. There is no module of Refund if wrong order delivered
c. Console App project is for just User not for other entities
d. Window Forms are not Responsive
e. Data base is not Normalize and one cell has more than one data

7. Future Directions

a. There will be a lot imprudent in which user will know who will deliver his order and when and he can also call him to comfit
b. There will be Refund method in which you will get refund of order
c. Console App and Windows Form Complete
d. Data Layer complete with Data Base and file handling so Front-End Developer can use what he like
e. Normalization of Data Base