Table of Content

My YouTube video link

https://youtu.be/xRAPnzJKybU

Stationery Shop Management System

Session: 2022 – 2026

Submitted by:

Noor Fatima 2022-CS-12

Supervised by:

Maida Shahid

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Project Overview:

- I am trying to solve the problem of keeping record of all items in a stationary shop in separate files. It is a type of software that will make tasks easy and fast. The main purpose of developing this system is to help most stationary shops to manage their inventory in more efficient way. It will help both manager and customers as well as employees working in that particular shop.
- This software will help manager to keep record of all transaction of items.
- It will also help customers to see list of items and to calculate bill. It will save a lot of time wasted in arranging and sorting data.
- It will provide enough support to keep an eye on all activities and generate report after particular period of time.
- The system is too easy to use that can be installed in every stationery shop and user can edit data containing in it.

Users of the Application:

Followings are the users of this system:

- Customers or Client
- Admin or Employee

This application is both for admin and customers:

- Customers or Client: Customers will login as client, there they can see list of all items and price of each item.
- Manager or Employee: Manager will login as admin he will have access of changing data (adding or deleting items).

- **Intended Functionality**

As Admin:

1. List of items currently available in the store.
2. View price of all items.
3. Add a new item.
4. Delete or update an item.
5. View location of items.
6. View most priced items.
7. Check feedback.
8. Refill stock
9. Add new worker
10. View all workers
11. Update worker

12. Exit.

As customer:

1. View all the item.
2. Place order.
3. Calculate bill.
4. Show most priced items.
5. Show price of a specific item.
6. Place order.
7. View order.
8. Change password.
9. Send feedback.
10. Exit

Comparison between Object Oriented Programming and Procedural Programming:

Advantage	Object-Oriented Programming (OOP)	Procedural Programming
Code Organization	It focuses on creating classes and objects to represent entities in the problem domain. Code is organized into objects, and behavior is encapsulated within those objects using methods	It organizes code into procedures or functions. The program is typically divided into a sequence of steps, where each step represents a specific task.
Data Management	Data and methods (functions) are encapsulated within objects. Objects can hold both data (attributes) and behavior (methods), allowing for a more intuitive representation of real-world entities.	Data is typically managed through variables that can be accessed and modified by procedures or functions. Data and functions are not inherently tied together as in OPP.

Advantage	Object-Oriented Programming (OOP)	Procedural Programming
Modularity	It promotes modular design through the use of classes and objects.	Code can also be modularized, but the primary focus is on breaking down tasks into functions or procedures.
Encapsulation	Encapsulation is a core principal, ensuring data protection and controlled access.	Code does not emphasize abstraction explicitly.
Inheritance	Supports inheritance, enabling code reuse and hierarchy.	Code does not emphasize abstraction explicitly.
Polymorphism	Allows objects of different classes to be treated as a common type , providing flexibility.	Polymorphism is not inherent, making it harder to achieve dynamic behavior based on object types.
Maintainability and Scalability	Code emphasizes maintenance and scalability through modular and organized code structure.	Code become less maintainable and scalable as codebase grows.
Reusability	Objects can be instantiated from classes, and code can be reused by creating new instances or inheriting from existing classes.	Functions can be reused by calling them from different parts of the program.

Advantage	Object-Oriented Programming (OOP)	Procedural Programming
Abstraction	Abstraction is inherit, allowing for simplified representations of complex concepts.	Code does not emphasize abstraction explicitly.

Design Pattern Implementation:

1. **BL (Business Layer)**

This project contains the BL Design Pattern in such a way that all the BL classes contains the Business logic functions and all the attributes are declared in it which are kept private for security purposes and all getter( ) and setter( ) functions are also included in this layer.

2. **DL (Data Layer)**

This project utilizes the DL Design Pattern in such a way that the DL classes contain all the Lists, and all functions related to Lists and all the CRUD functions of classes.

Class Details:

1. User Class:

Attributes:

- username: The username chosen by the user.
- password: The unique password associated with the user account.
- role: Defines the role or type of user (e.g., Owner, manager, or customer).

Behaviors:

getname(): Returns the username of the user.

setname(username): Sets or updates the username for the user.

getPassword(): Retrieves the password associated with the user account.

setPassword(password): Sets or updates the password for the user account.

getRole(): Retrieves the role of the user.

setRole(role): Sets or updates the role of the user.

isUserValid(userName): Validates wheather user is present or not.

signIn(username , password) : Allow user to signIn if user present

2. Product

Attributes:

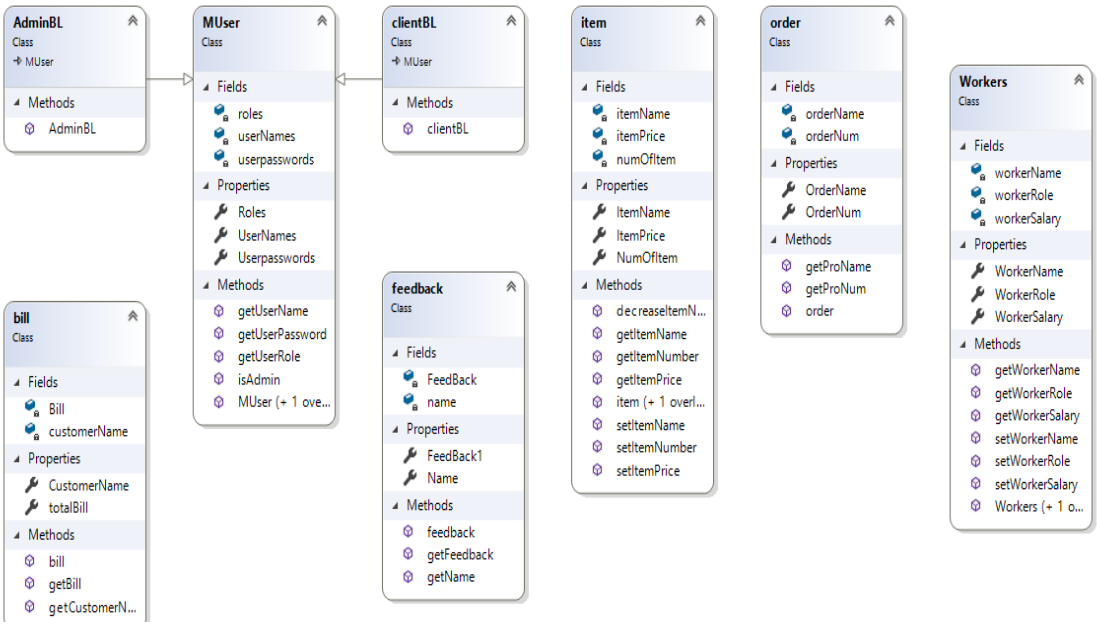
- itemName: It represent the type of product.
- numOfItem: It represent the quantity of product.
- itemPrices: It represent the prices of product.

Behaviors:

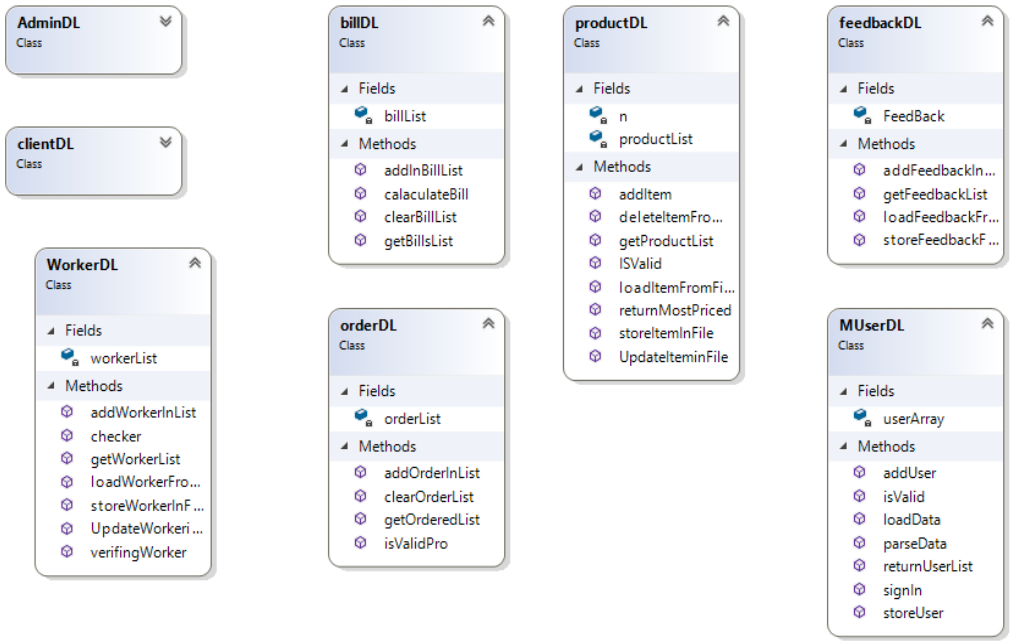
getItemPrice(): Returns the price of the product.
getItemName (): Returns the type of the product.
getItemNumber() Returns the quantitiy of the product.
setItemPrice():Retrieves the price of the product.
`setItemNumber()`: set number of the product.
`setItemPrice()`:set price of Item.

CRC Diagram:

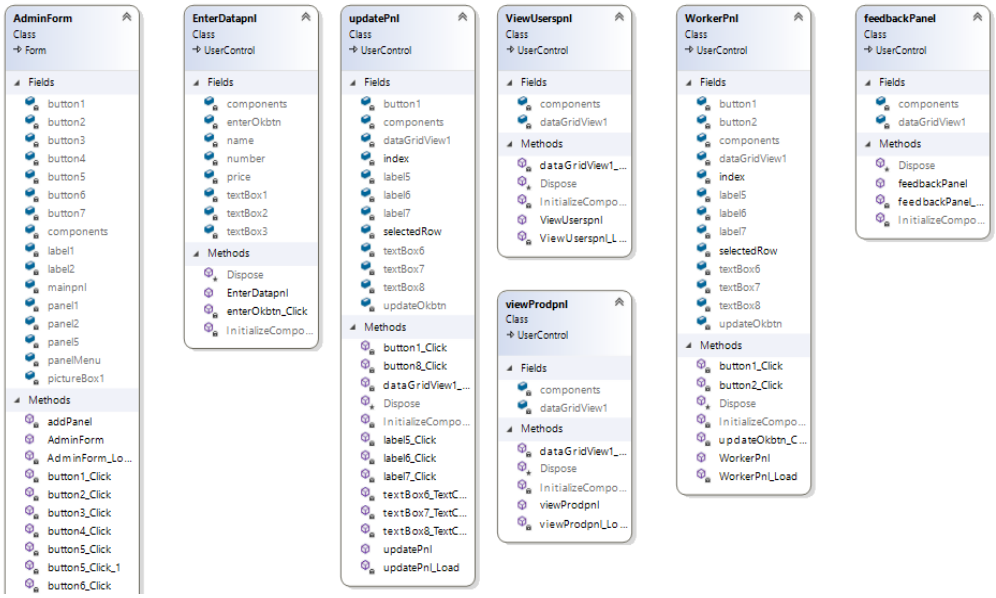
BL Folder



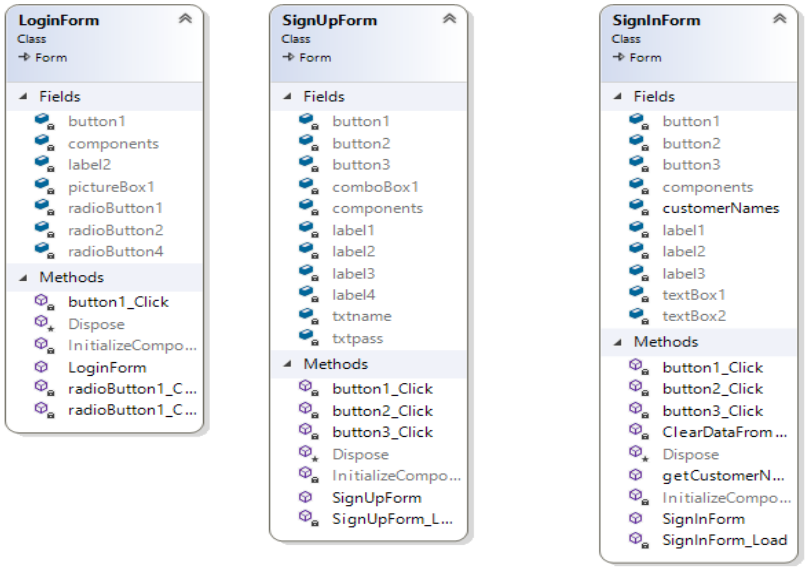
DL Folder



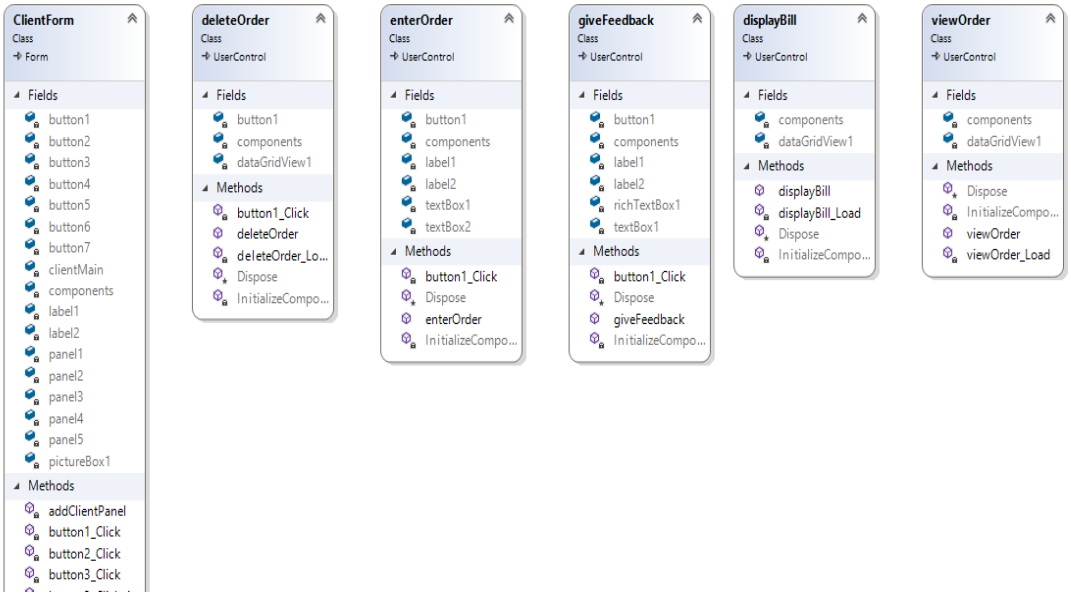
Admin Forms



Login Forms

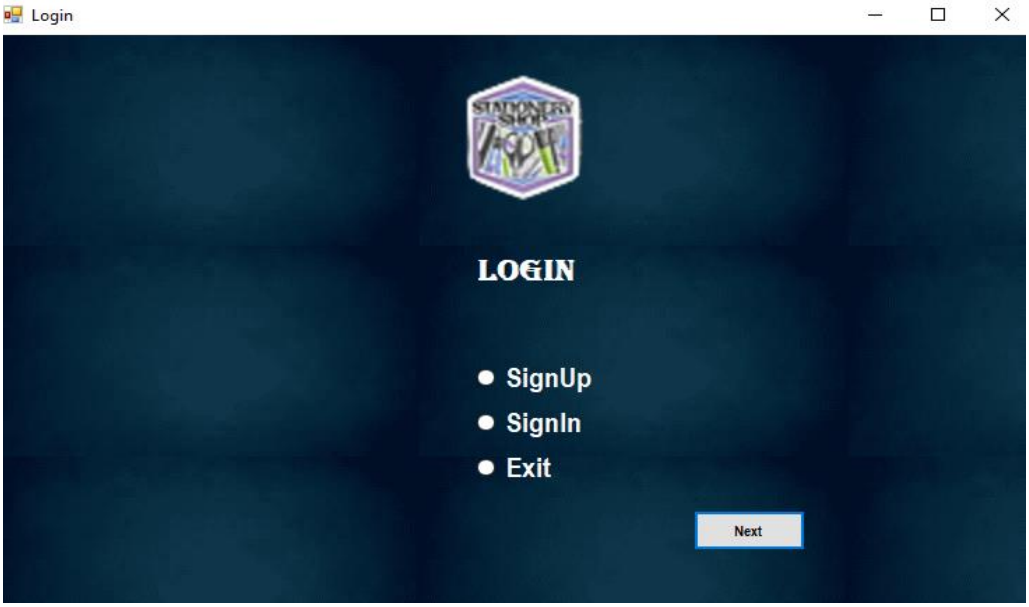


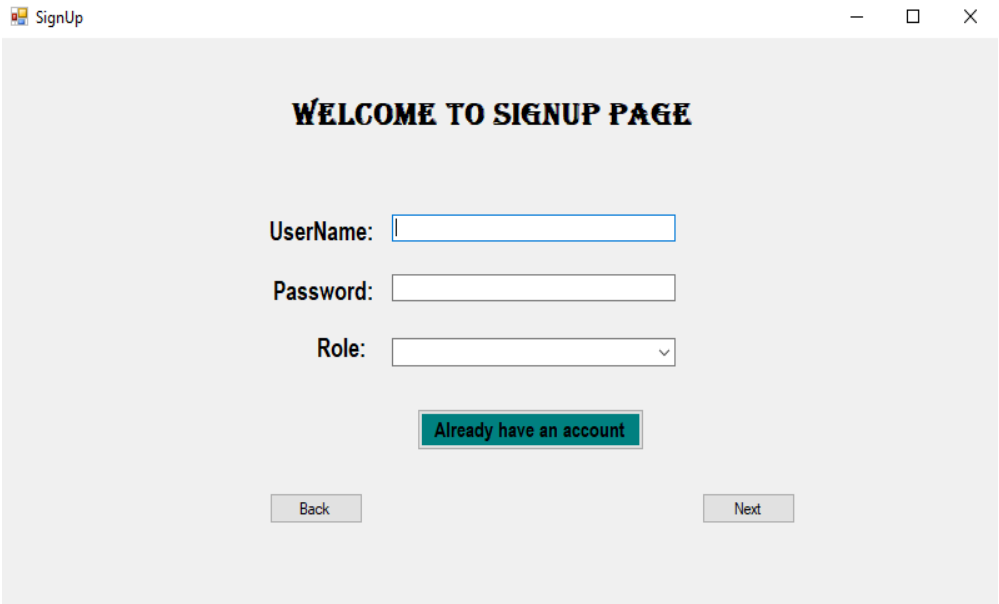
Client Forms



Wireframes:

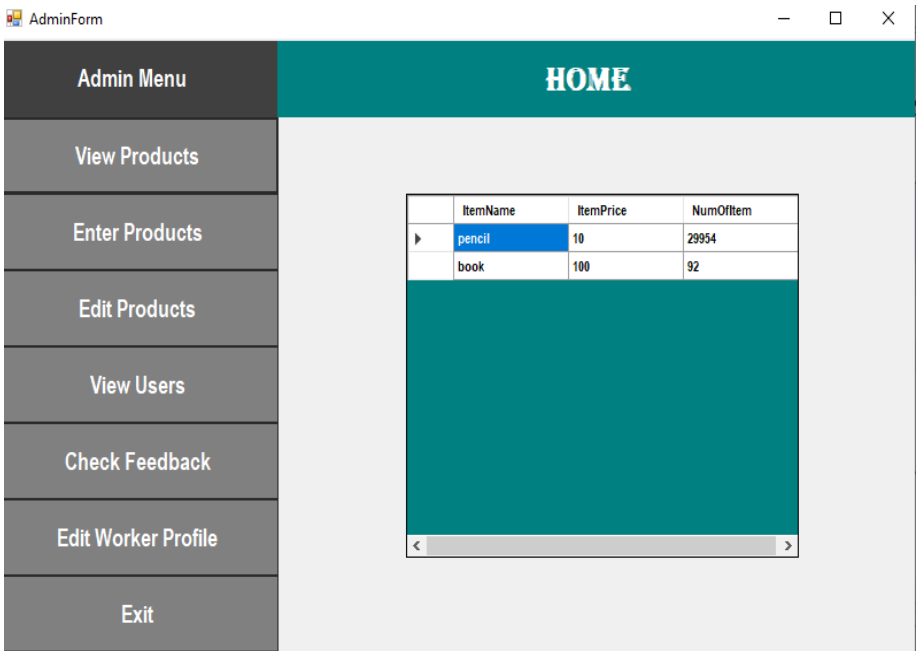
Login Forms:

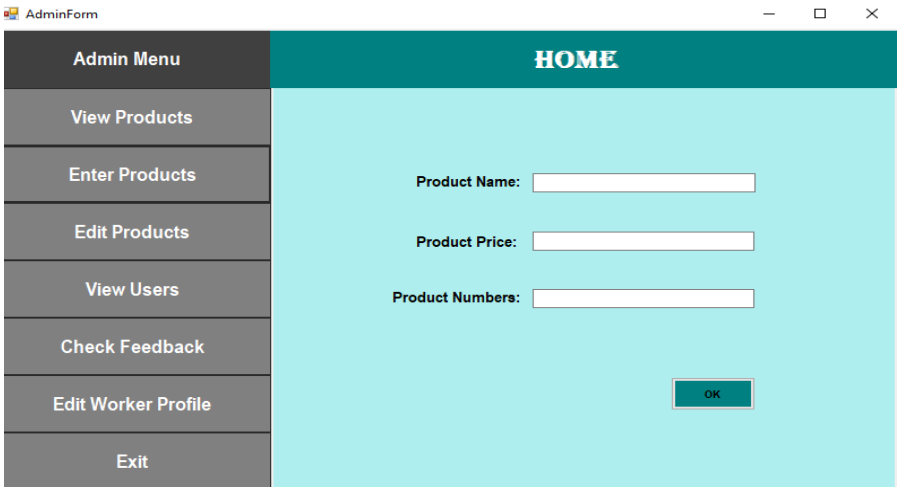


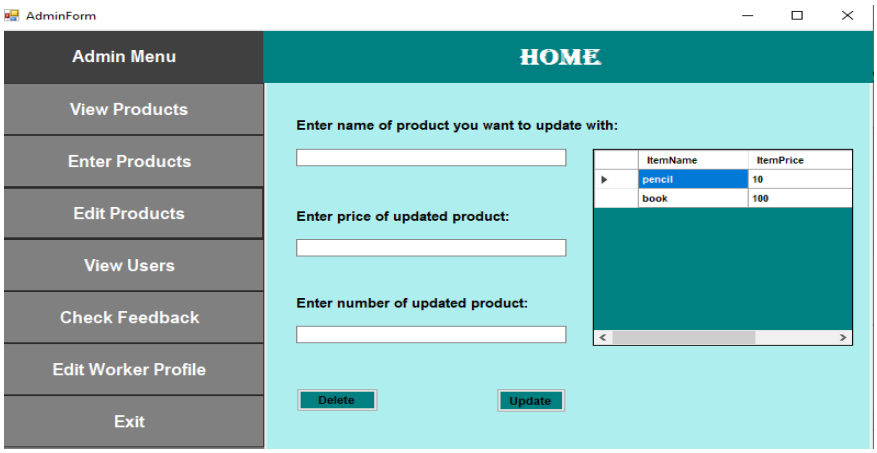


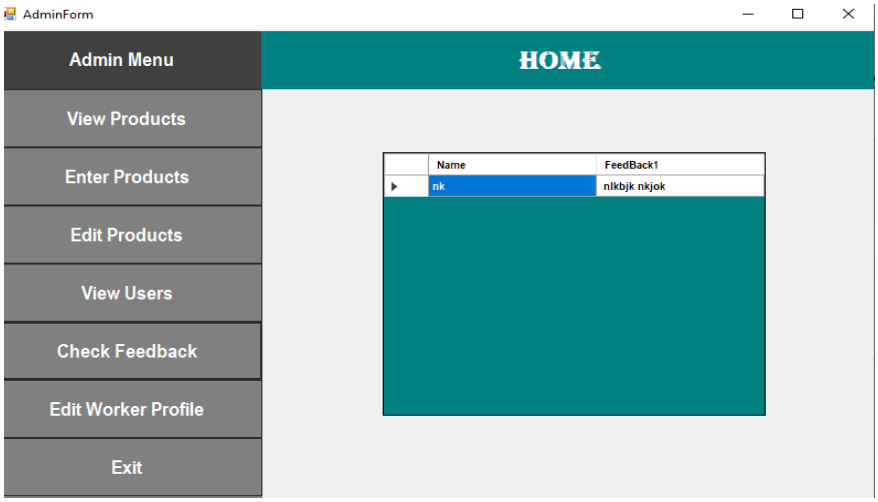

signin
WELCOME TO SIGNIN PAGE
UserName:
Password:
Create Account
Back
Next

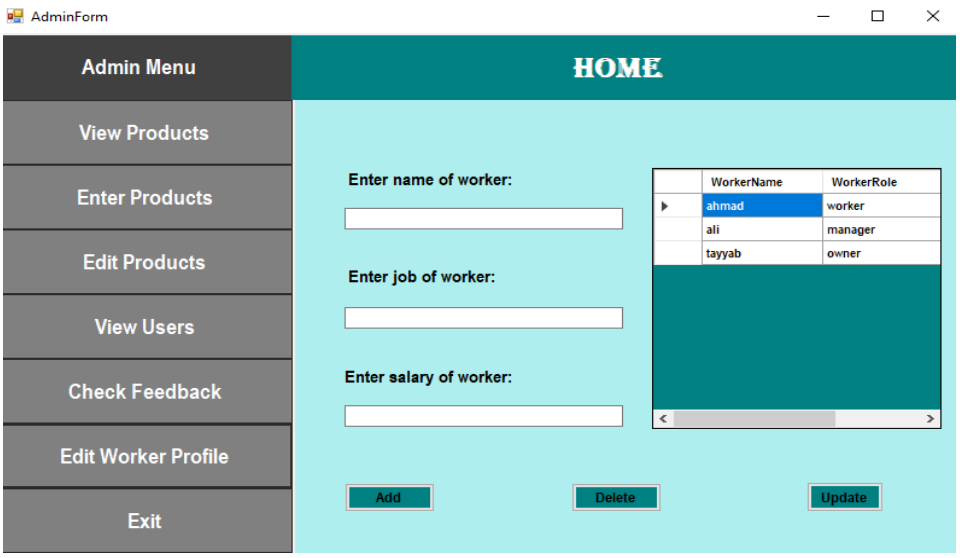
Admin Forms


AdminForm
Admin Menu
HOME
View Products
Enter Products
Edit Products
View Users
Check Feedback
Edit Worker Profile
Exit
ItemName
ItemPrice
NumOfItem
pencil
10
29954
book
100
92

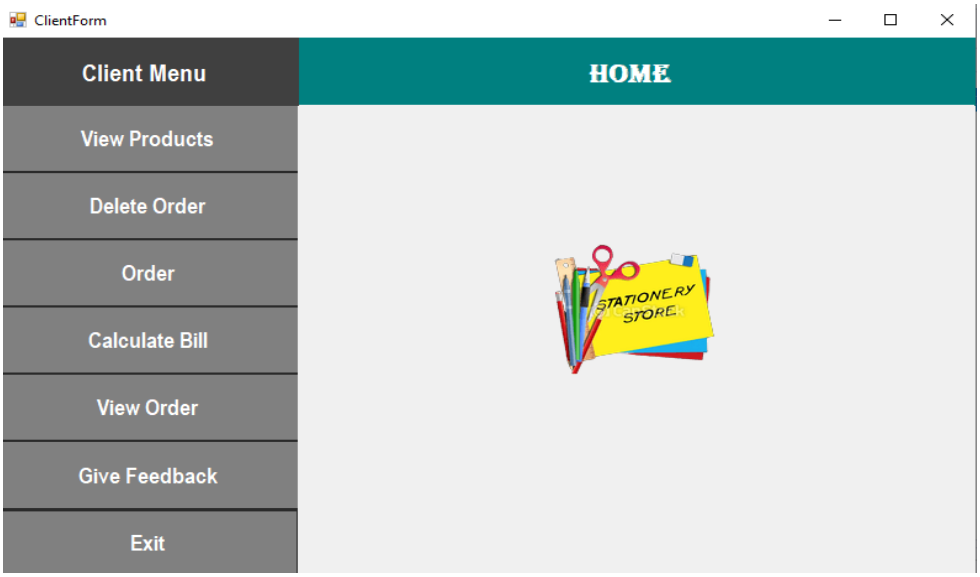

AdminForm
Admin Menu
HOME
View Products
Enter Products
Edit Products
View Users
Check Feedback
Edit Worker Profile
Exit
Product Name:
Product Price:
Product Numbers:
OK

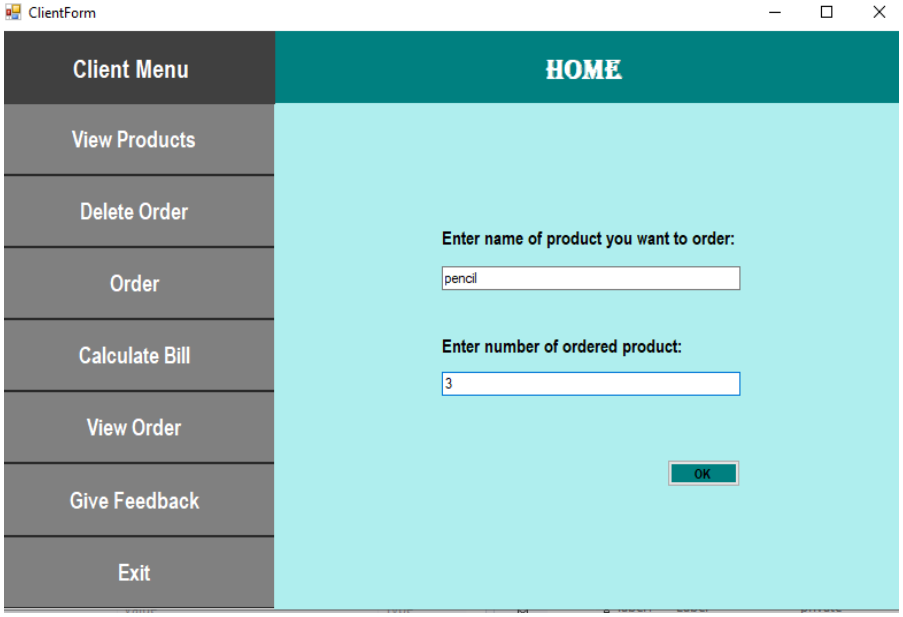

AdminForm
Admin Menu
HOME
View Products
Enter Products
Edit Products
View Users
Check Feedback
Edit Worker Profile
Exit
Enter name of product you want to update with:
Enter price of updated product:
Enter number of updated product:
ItemName
ItemPrice
pencil
10
book
100
Delete
Update

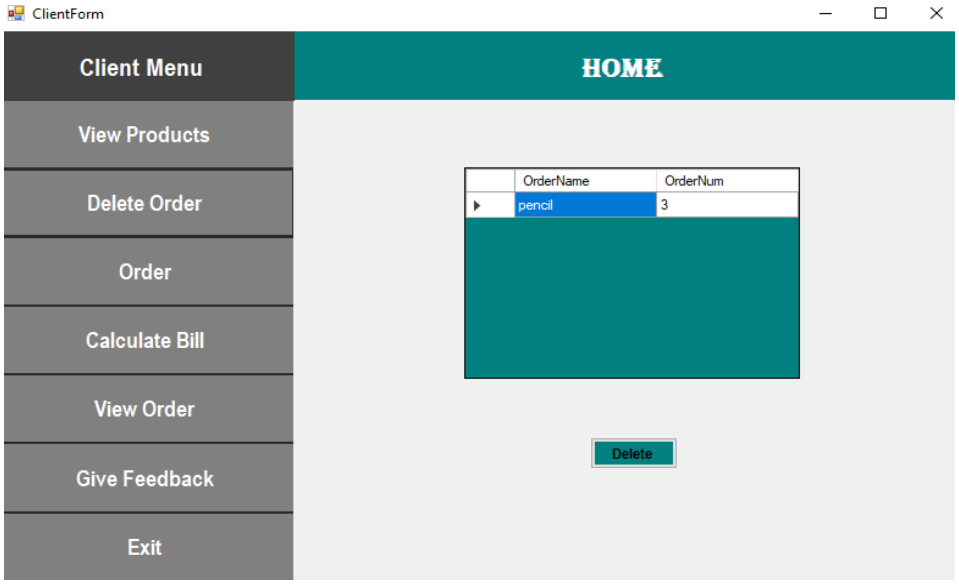

AdminForm
Admin Menu
HOME
View Products
Enter Products
Edit Products
View Users
Check Feedback
Edit Worker Profile
Exit
Name
FeedBack1
nk
nlkbjk nkjok

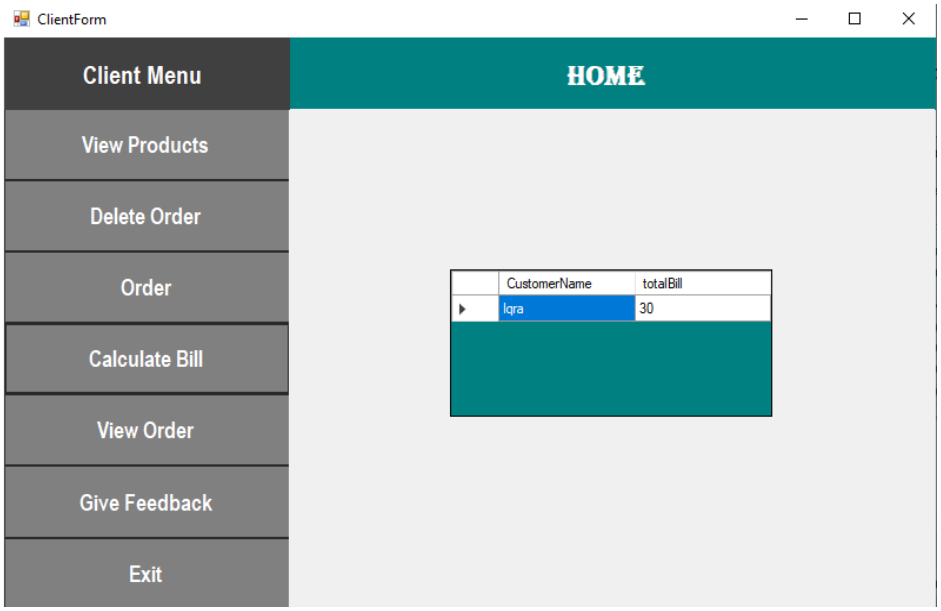


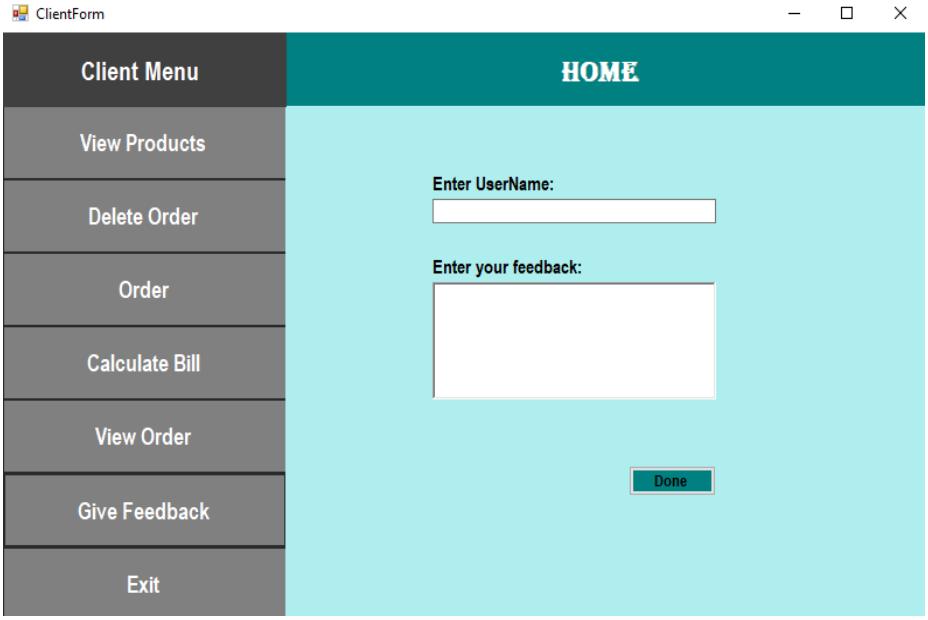
Client Forms




ClientForm
Client Menu
HOME
View Products
Delete Order
Order
Calculate Bill
View Order
Give Feedback
Exit
Enter name of product you want to order:
pencil
Enter number of ordered product:
3
OK


ClientForm
Client Menu
HOME
View Products
Delete Order
Order
Calculate Bill
View Order
Give Feedback
Exit
OrderName
OrderNum
pencil
3
Delete


ClientForm
Client Menu
HOME
View Products
Delete Order
Order
Calculate Bill
View Order
Give Feedback
Exit
CustomerName
totalBill
Iqra
30


ClientForm
Client Menu
HOME
View Products
Delete Order
Order
Calculate Bill
View Order
Give Feedback
Exit
Enter UserName:
Enter your feedback:
Done

Working Code:

Muser Class:

```
        class MUser
    {
        private string userNames;
        private string userpasswords;
        private string roles;

        public string UserNames { get => userNames; set => userNames = value; }
        public string Userpasswords { get => userpasswords; set => userpasswords = value;
}
        public string Roles { get => roles; set => roles = value; }

        public MUser(string userNames, string userpasswords)
        {
            this.userNames = userNames;
            this.userpasswords = userpasswords;
        }

        public MUser(string userNames, string userpasswords, string roles)
        {
            this.userNames = userNames;
            this.userpasswords = userpasswords;
            this.roles = roles;
        }
        public bool isAdmin()
        {
            if (roles == "Admin")
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public string getUserName()
        {
            return this.userNames;
        }
        public string getUserPassword()
        {
            return this.userpasswords;
        }
        public string getUserRole()
        {
            return this.roles;
        }
    }
```

Item Class:

```
class item
{

    private string itemName;
    private int itemPrice;
    private int numOfItem;

    public string ItemName { get => itemName; set => itemName = value; }
    public int ItemPrice { get => itemPrice; set => itemPrice = value; }
    public int NumOfItem { get => numOfItem; set => numOfItem = value; }

    public item()
    {

    }
    public item(string itemName, int itemPrice, int numOfItem)
    {
        this.itemName = itemName;
        this.itemPrice = itemPrice;
        this.numOfItem = numOfItem;
    }

    public void setItemName(string name)
    {

        this.itemName = name;

    }
    public string getItemName()
    {

        return this.itemName;

    }

    public void setItemPrice(int price)
    {
        if (price > 0)
        {
            this.itemPrice = price;
        }
    }
    public int getItemPrice()
    {

        return this.itemPrice;

    }
    public void setItemNumber(int num)
    {
        if (num > 0)
        {
            this.numOfItem = this.numOfItem - num;
        }
    }
    public int getItemNumber()
```

```
        {

            return this.numOfItem;

        }
        public bool decreaseItemNumber(int des)
        {
            if (des > 0 && this.numOfItem > 0)
            {
                this.numOfItem = this.numOfItem - des;
                return true;
            }
            else
                return false;
        }
    }
```

Worker Class:

```
            class Workers
    {
        private string workerName;
        private string workerRole;
        private float workerSalary;

        public string WorkerName { get => workerName; set => workerName = value; }
        public string WorkerRole { get => workerRole; set => workerRole = value; }
        public float WorkerSalary { get => workerSalary; set => workerSalary = value; }

        public Workers()
        {

        }
        public Workers(string workerName, string workerRole, float workerSalary)
        {
            this.workerName = workerName;
            this.workerRole = workerRole;
            this.workerSalary = workerSalary;
        }

        public void setWorkerName(string workerName)
        {

            this.workerName = workerName;

        }
        public string getWorkerName()
        {

            return this.workerName;

        }

        public void setWorkerRole(string workerRole)
        {


            this.workerRole = workerRole;
```

```
        }
        public string getWorkerRole()
        {

            return this.workerRole;

        }
        public void setWorkerSalary(float salary)
        {
            if (salary > 0)
            {
                this.workerSalary = salary;
            }
        }
        public float getWorkerSalary()
        {

            return this.workerSalary;

        }
    }
```

Order Class:

```
    class order
    {
      private string orderName;
        private int orderNum;

        public order(string orderName , int orderNum)
        {
            this.orderName = orderName;
            this.orderNum = orderNum;
        }

        public string OrderName { get => orderName; set => orderName = value; }
        public int OrderNum { get => orderNum; set => orderNum = value; }

        public string getProName()
        {
            return this.orderName;
        }
        public int getProNum()
        {
            return this.orderNum;
        }
    }
```

Feedback Class:

```
  class feedback
    {
        private string name;
        private string FeedBack;

        public feedback(string name , string FeedBack)
```

```
        {
            this.name = name;
            this.FeedBack = FeedBack;
        }

        public string getName()
         {
            return this.name;
        }
        public string getFeedback()
        {
            return this.FeedBack;
        }
        public string Name { get => name; set => name = value; }
        public string FeedBack1 { get => FeedBack; set => FeedBack = value; }
    }
```

Bill Class:

```
    class bill
    {
        string customerName;
        int Bill;

        public string CustomerName { get => customerName; set => customerName = value; }
        public int totalBill { get => Bill; set => Bill = value; }

        public bill(string customerName , int Bill)
        {
            this.customerName = customerName;
            this.Bill = Bill;
        }
        public string getCustomerName()
        {
            return this.customerName;
        }
        public int getBill()
        {
            return this.Bill;
        }
    }
```

DL Folder:

MUserDL Class:

```
    class MUserDL
    {
        private static List<MUser> userArray = new List<MUser>();

        public static void addUser(MUser input)
        {
            userArray.Add(input);
```

```
        }
        public static List<MUser> returnUserList()
        {
           return userArray;
        }

        public static string signIn(MUser User) // signIn
        {


            foreach (MUser storedUser in userArray)
            {
                if (User.getUserName() == storedUser.getUserName() &&
User.getUserPassword() == storedUser.getUserPassword())
                {
                    return storedUser.getUserRole();
                }
            }

            return null;
        }
        public static bool isValid(string name)
        {


            for (int i = 0; i < userArray.Count; i++)
            {
                if (userArray[i].getUserName() == name)
                {
                    return false;
                }

            }
            return true;
        }
        public static void storeUser(MUser input, string Path)
        {
            if (File.Exists(Path))
            {
                StreamWriter file = new StreamWriter(Path, true);

                file.WriteLine(input.getUserName() + ',' + input.getUserPassword()+ ',' +
input.getUserRole());

                file.Flush();
                file.Close();
            }
        }
        public static bool loadData(string Path)
        {


            StreamReader file = new StreamReader(Path);
            string record;

            if (File.Exists(Path))
            {
                while ((record = file.ReadLine()) != null)
```

```
                {

                    string userNames = parseData(record, 1);
                    string userpasswords = parseData(record, 2);
                    string roles = parseData(record, 3);
                    MUser u1 = new MUser(userNames, userpasswords, roles);
                    userArray.Add(u1);


                }

                file.Close();
                return true;
            }
            else
            {
                return false;
            }


        }

        public static 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;
        }
    }
```

ProductDL Class:

```
    class productDL
    {
        static int n;
        private static List<item> productList = new List<item>();
        public static List<item> getProductList()
        {
            return productList;
        }
        public static void addItem(item i)
        {
            productList.Add(i);
        }

        public static bool ISValid(string check)
```

```
        {
            for (int i = 0; i < productList.Count; i++)
            {
                if (productList[i].getItemName() == check)
                {
                    n = i;
                    return true;
                }
            }
            return false;
        }


        public static void storeItemInFile(item input, string Path)
        {

            StreamWriter file1 = new StreamWriter(Path, true);
            if (File.Exists(Path))
            {


                file1.WriteLine(input.getItemName() + ',' + input.getItemPrice() + ',' +
input.getItemNumber());

                file1.Flush();
                file1.Close();
            }

        }
        public static bool loadItemFromFile(string path)
        {


            StreamReader file2 = new StreamReader(path);
            string record;

            if (File.Exists(path))
            {
                while ((record = file2.ReadLine()) != null)
                {
                    if (string.IsNullOrWhiteSpace(record))
                        continue;
                    string[] splittedData = record.Split(',');
                    string itemName = splittedData[0];
                    int itemPrice = int.Parse(splittedData[1]);
                    int numOfItem = int.Parse(splittedData[2]);
                    item i2 = new item(itemName, itemPrice, numOfItem);
                    productList.Add(i2);


                }

                file2.Close();
                return true;
            }
            else
            {
                return false;
```

```
            }

        }


        public static bool deleteItemFromFile(string Path)
        {

            if (File.Exists(Path))
            {
                StreamWriter file3 = new StreamWriter(Path);


                for (int i = 0; i < productList.Count; i++)
                {
                    file3.WriteLine(productList[i].getItemName() + ',' +
productList[i].getItemPrice() + ',' + productList[i].getItemNumber());

                }
                file3.Flush();
                file3.Close();
                return true;
            }
            else
                return false;

        }



        public static bool UpdateIteminFile(string path)
        {
            if (File.Exists(path))
            {
                StreamWriter file4 = new StreamWriter(path);


                for (int i = 0; i < productList.Count; i++)
                {
                    file4.WriteLine(productList[i].getItemName() + ',' +
productList[i].getItemPrice() + ',' + productList[i].getItemNumber());
                }
                file4.Flush();
                file4.Close();
                return true;
            }
            else
            return false;
        }

        public static string returnMostPriced(int idx)
        {
            return productList[idx].getItemName();

        }
    }
```

WorkerDL Class:

```
class WorkerDL
    {
        private static List<Workers> workerList = new List<Workers>();
        public static List<Workers> getWorkerList()
        {
            return workerList;
        }
        public static void addWorkerInList(Workers w)
        {
            workerList.Add(w);
        }



        public static bool checker(string workerNames)
        {

            foreach (Workers work in workerList)
            {
                if (work.getWorkerName() == workerNames)
                {
                    return false;
                }
            }
            return true;
        }

        public static bool storeWorkerInFile(Workers inputs, string Path)
        {

            StreamWriter file1 = new StreamWriter(Path, true);
            if (File.Exists(Path))
            {


                file1.WriteLine(inputs.getWorkerName() + ',' + inputs.getWorkerRole() +
',' + inputs.getWorkerSalary());

                file1.Flush();
                file1.Close();
                return true;
            }
            else
            {
                return false;
            }
        }

        public static bool loadWorkerFromFile(string path)
        {


            StreamReader file2 = new StreamReader(path);
            string record;

            if (File.Exists(path))
```

```
            {
                while ((record = file2.ReadLine()) != null)
                {

                    string[] splittedData = record.Split(',');
                    string workerName = splittedData[0];
                    string workerRole = (splittedData[1]);
                    float salary = float.Parse(splittedData[2]);
                    Workers w2 = new Workers(workerName, workerRole, salary);
                    workerList.Add(w2);


                }

                file2.Close();
                return true;
            }
            else
            {
                return false;
            }

        }
        public static bool verifingWorker(string Name)
        {
            bool check = false;
            for (int i = 0; i < workerList.Count; i++)
            {
                if (workerList[i].getWorkerName() == Name)
                {

                    check = true;

                }
            }
            return check;
        }

        public static bool UpdateWorkerinFile(string path)
        {
            if (File.Exists(path))
            {
                StreamWriter file4 = new StreamWriter(path);


                for (int i = 0; i < workerList.Count; i++)
                {
                    file4.WriteLine(workerList[i].getWorkerName() + ',' +
workerList[i].getWorkerRole() + ',' + workerList[i].getWorkerSalary());
                }
                file4.Flush();
                file4.Close();
                return true;
            }
            else
                return false;
        }
    }
```

OrderDL Class:

```
class orderDL
    {
        private static List<order> orderList = new List<order>();

        public static List<order> getOrderedList()
        {
            return orderList;
        }

        public static void addOrderInList(order odr)
        {
            orderList.Add(odr);
        }

        public static int isValidPro(string check)
        {
            int idx = -1;
            List<item> proList = productDL.getProductList();
            for (int i = 0; i < proList.Count; i++)
            {
                if (proList[i].getItemName() == check)
                {
                    idx = i;
                    return idx;
                }
            }
            return idx;
        }

        public static void clearOrderList()
        {
            orderList.Clear();
        }

}
```

FeedbackDL Class:

```
class feedbackDL
    {
        private static List<feedback> FeedBack = new List<feedback>();

        public static List<feedback> getFeedbackList()
        {
            return FeedBack;
        }

        public static void addFeedbackInList(feedback inp)
        {
            FeedBack.Add(inp);
        }
        public static void storeFeedbackFromFile(feedback response,  string Path1)
        {
```

```
            StreamWriter file2 = new StreamWriter(Path1);


            if (File.Exists(Path1))
            {

                 file2.WriteLine(response.getName() + ',' + response.getFeedback() );


                file2.Close();

            }


        }
        public static bool loadFeedbackFromFile(string Path1)
        {


            StreamReader file2 = new StreamReader(Path1);
            string record;

            if (File.Exists(Path1))
            {
                while ((record = file2.ReadLine()) != null)
                {


                    string[] splittedData = record.Split(',');
                    string Name = splittedData[0];
                    string feedBack = splittedData[1];
                    feedback fb = new feedback(Name,feedBack);
                    FeedBack.Add(fb);

                }

                file2.Close();
                return true;
            }
            else
            {
                return false;
            }

        }
    }
```

BillDL Class:

```
    class billDL
    {
        private static List<bill> billList = new List<bill>();
        public static List<bill> getBillsList()
        {
          return  billList;
        }
        public static void addInBillList(bill inp)
```

```
        {
            if(billList.Count > 0)
            {
                billList.RemoveAt(0);
                billList.Insert(0, inp);
            }
           else
                billList.Insert(0, inp);
        }

        public static void clearBillList()
        {
            billList.Clear();
        }
         public static int calaculateBill()
        {
            int sum = 0;
            int total=0;
            int price=0;
            string cusProNames;
            List<order> cusOrder = orderDL.getOrderedList();
            List<item> Product = productDL.getProductList();
            for (int i = 0; i < cusOrder.Count; i++)
            {
                cusProNames = cusOrder[i].getProName();
                for(int j =0; j < Product.Count; j++)
                {
                    if(Product[i].getItemName() == cusProNames)
                    {
                        price = Product[i].getItemPrice();
                    }
                }
                sum = price * cusOrder[i].getProNum();
                total = total + sum;



            }
            return total;

        }


    }
```

Forms:

Login Form:

```
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
            string path = "user.txt";
            string proPath = "items.txt";
            string feebackPath = "Feedback.txt";
```

```
            string workerPath = "worker.txt";
            // load data from files
            if (MUserDL.loadData(path) && productDL.loadItemFromFile(proPath)
               && feedbackDL.loadFeedbackFromFile(feebackPath) &&
WorkerDL.loadWorkerFromFile(workerPath))
            {
                MessageBox.Show("Data loaded from file");
            }
            else
            {
                MessageBox.Show("Data not loaded from file");
            }

        }


        private void button1_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                Form moreForm = new SignInForm();
                moreForm.Show();
                radioButton1.Checked = false;
            }
            if (radioButton2.Checked)
            {
                Form moreForm = new SignUpForm();
                moreForm.Show();
                radioButton2.Checked = false;
            }
            if (radioButton4.Checked)
            {
                this.Close();
            }
        }

}
```

SignUp Form:

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



        private void button2_Click(object sender, EventArgs e) // signIn code
        {
            string username = txtname.Text;
            bool check = MUserDL.isValid(username);
            if (check)
            {
                MUser user = new MUser(txtname.Text, txtpass.Text, comboBox1.Text);
                MUserDL.addUser(user);
```

```
                MUserDL.storeUser(user, "user.txt");
                if (user != null)
                {
                    MessageBox.Show("added successfully");
                }
            }
            else
            {
                MessageBox.Show("Inavlid userName");
            }
            txtname.Text = string.Empty;
            txtpass.Text = string.Empty;
            comboBox1.Text = string.Empty;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form moreForm = new SignInForm();
            moreForm.Show();
        }
}
```

SignIn Form:

```
public partial class SignInForm : Form
    {
       static string customerNames;
        public SignInForm()
        {
            InitializeComponent();
            orderDL.clearOrderList();
            billDL.clearBillList();
        }
        private void ClearDataFromForm()
        {
            textBox1.Text = "";
            textBox1.Text = "";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string username = textBox1.Text;
            string password = textBox2.Text;
            customerNames = textBox1.Text;
            MUser user = new MUser(username, password);
            string validUser = MUserDL.signIn(user); // check user valid
            if(validUser != null)
            {

                if (validUser == "Admin")
                {
```

```
                    this.Hide();
                    Form moreForm = new AdminForm();// open admin form
                    moreForm.Show();
                }
                else
                {
                    this.Hide();
                    Form moreForm = new ClientForm(); // open client form
                    moreForm.Show();
                }
            }
            else
            {
                MessageBox.Show("Invalid Input");
            }
            textBox1.Text= string.Empty;
            textBox2.Text = string.Empty;
        }
        public static string getCustomerName()
        {
            return customerNames;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form moreForm = new SignUpForm();
            moreForm.Show();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

}
```

Admin Form:

```
public partial class AdminForm : Form
    {

        public AdminForm()
        {
            InitializeComponent();




        }


        private void addPanel(UserControl panel) // main panel
        {
            panel.Dock = DockStyle.Fill;
            mainpnl.Controls.Clear();
            mainpnl.Controls.Add(panel);
            panel.BringToFront();
        }
```

```
        private void button1_Click(object sender, EventArgs e) // add view product panel
        {
            viewProdpnl viewProdpnl = new viewProdpnl();
            addPanel(viewProdpnl);
        }
        private void button2_Click(object sender, EventArgs e) // add enter panel
        {
            EnterDatapnl enterDatapnl = new EnterDatapnl();
            addPanel(enterDatapnl);


        }
        private void button3_Click(object sender, EventArgs e) // add edit panel
        {
            updatePnl open = new updatePnl();
            addPanel(open);
        }
        private void button4_Click(object sender, EventArgs e) //add view user panel
        {

            ViewUserspnl userPanel = new ViewUserspnl();
            addPanel(userPanel);


        }
        private void button5_Click_1(object sender, EventArgs e) // add feedback panel
        {
            feedbackPanel showFeedback = new feedbackPanel();
            addPanel(showFeedback);
        }

        private void button6_Click_1(object sender, EventArgs e) // add worker panel
        {
            WorkerPnl showworker = new WorkerPnl();
            addPanel(showworker);
        }

        private void button7_Click_1(object sender, EventArgs e) // exit
        {
            this.Hide();
            Form moreForm = new SignInForm();
            moreForm.Show();
        }

}
```

Client Form:

```
public partial class ClientForm : Form
    {
        public ClientForm()
        {
            InitializeComponent();
        }
        private void addClientPanel(UserControl panel) // add client panel to main panel
        {
            panel.Dock = DockStyle.Fill;
```

```
            clientMain.Controls.Clear();
            clientMain.Controls.Add(panel);
            panel.BringToFront();
        }
        private void button1_Click(object sender, EventArgs e)// add view panel
        {
            viewProdpnl viewProdpnl = new viewProdpnl();
            addClientPanel(viewProdpnl);
        }

        private void button2_Click(object sender, EventArgs e)// add place order panel
        {
            placeOrderpnl order = new placeOrderpnl();
            addClientPanel(order);
        }



        private void button3_Click_1(object sender, EventArgs e) // add view order panel
        {
            viewOrder vieworder = new viewOrder();
            addClientPanel(vieworder);
        }



        private void button4_Click_1(object sender, EventArgs e) // add delete order
panel
        {
            deleteOrder deleteorder = new deleteOrder();
            addClientPanel(deleteorder);
        }

        private void button5_Click(object sender, EventArgs e)// add display bill panel
        {

            displayBill display = new displayBill();
            addClientPanel(display);
        }

        private void button6_Click(object sender, EventArgs e) // add give feedback panel
        {
            giveFeedback feedbacks = new giveFeedback();
            addClientPanel(feedbacks);
        }

        private void button7_Click(object sender, EventArgs e)// exit
        {
            this.Hide();
            Form moreForm = new SignInForm();
            moreForm.Show();
        }

}
```

Conclusion:

The stationery shop management system project successfully applied the principles of encapsulation, inheritance, and polymorphism in the context of object-oriented programming. It achieved modular and reusable code through encapsulating data and functionality within classes, established relationships between classes through inheritance, and allowed objects of different classes to be treated uniformly through polymorphism. Challenges included designing a flexible class hierarchy and ensuring proper encapsulation, while lessons learned revolved around creating reusable and extensible class structures and effectively utilizing inheritance and polymorphism.