**AUTOMOTIVE MANAGEMNET SYSTEM**



Session: 2022 – 2025

**Submitted by:**

Saba Shahdin 2022-CS-112

**Supervised by:**

Sir Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[**Introduction:** 3](#_Toc140576160)

[**Overview:** 3](#_Toc140576161)

[**Objective:** 3](#_Toc140576162)

[**Contribution in computer science field:** 3](#_Toc140576163)

[**Output:** 3](#_Toc140576164)

[**Users of Application** 3](#_Toc140576165)

[**Functional Requirements:** 3](#_Toc140576166)

[**OWNER:** 4](#_Toc140576167)

[**EMPLOYEE:** 4](#_Toc140576168)

[**MANAGER:** 5](#_Toc140576169)

[**Object Oriented Programming (OOP) Concept:** 5](#_Toc140576170)

[ **Inheritance:** 6](#_Toc140576171)

[**Polymorphism** 6](#_Toc140576172)

[**Association:** 6](#_Toc140576173)

[**Comparison with Procedural Programming:** 7](#_Toc140576174)

[**Design Pattern Implementation:** 7](#_Toc140576175)

[**Business Layer (BL):** 7](#_Toc140576176)

[**Data Layer (DL):** 10](#_Toc140576178)

[**User Interface:** 12](#_Toc140576179)

[**Key Classes:** 13](#_Toc140576180)

[**Wireframes:** 14](#_Toc140576181)

[**Resources:** 33](#_Toc140576218)

[**Class Diagram:** 34](#_Toc140576220)

[**Code:** 35](#_Toc140576221)

[**Business Logic:** 35](#_Toc140576222)

[ **Person:** 35](#_Toc140576223)

[ **Admin:** 37](#_Toc140576224)

[ **Staff:** 38](#_Toc140576225)

[ **Employee:** 40](#_Toc140576226)

[ **Manager:** 41](#_Toc140576227)

[ **Product:** 42](#_Toc140576228)

[ **Selling Product:** 43](#_Toc140576229)

[ **Repairing Product:** 44](#_Toc140576230)

[ **Money:** 45](#_Toc140576231)

[ **Money Sell:** 47](#_Toc140576232)

[ **Money Repair** 48](#_Toc140576233)

[ **Digital Layer:** 49](#_Toc140576234)

[ **Person DL:** 49](#_Toc140576235)

[ **Staff DL:** 51](#_Toc140576236)

[ **Employee DL:** 52](#_Toc140576237)

[ **Manager DL:** 55](#_Toc140576238)

[ **Selling DL:** 57](#_Toc140576239)

[ **Money Sell DL:** 60](#_Toc140576240)

[ **Money Repair DL:** 62](#_Toc140576241)

[ **User Interface:** 64](#_Toc140576242)

[ **Main Screen:** 64](#_Toc140576243)

[ **Main Role :** 64](#_Toc140576244)

[ **Sign IN:** 66](#_Toc140576245)

[ **ID:** 67](#_Toc140576246)

[ **Admin Screen:** 67](#_Toc140576247)

[ **Selling Products:** 68](#_Toc140576248)

[ **Repairing Products:** 78](#_Toc140576249)

[ **Employee UI :** 89](#_Toc140576250)

[ **Manager UI:** 99](#_Toc140576251)

[ **View UI:** 109](#_Toc140576252)

[ **Manager Interface:** 114](#_Toc140576253)

[ **Employee interface:** 115](#_Toc140576254)

[ **Manager Money:** 116](#_Toc140576255)

[ **Employee Money:** 118](#_Toc140576256)

[ **Salary UI:** 122](#_Toc140576257)

[**Conclusion:** 124](#_Toc140576259)

[**Achievements:** 124](#_Toc140576260)

[**Challenges:** 124](#_Toc140576261)

# **Visual Representation:**

For visual representation go to link:

https://youtu.be/oZq9oLESxRI

# **Introduction:**

## **Overview:**

The objective of “Automotive Management System “is to handle the system of small shops where owner concerns to keep daily record. This focuses more on the small scale business rather that large firms because in these days of technology some small shops still lack of digital system from which they can manage their records.

## **Objective:**

1. To manage the garage efficiently.
2. To make the system more manageable and organize it programmatically.
3. To help employees handle their own task properly.
4. As an owner handle all the affairs digitally in just one go.

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

1. Helps in digital management of the small business
2. Keep the previous record of firm.

## **Output:**

1. Number of products in the firm.
2. Number of products can be repaired.
3. Products prices
4. On manager and owner end, total money earned in one day.
5. Check salary on daily work hour

**Users of Application:**

There are three users of my **“AUTOMOTIVE MANAGEMNT SYSTEM”**

* Owner of Garage
* Manager
* Employee

## **Functional Requirements:**

The functional requirement of the automotive management system is to keep the check on working environment by controlling different modules.

## **OWNER:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***User Story ID*** | ***As a*** | ***I want to perform*** | ***So that I can*** |
| ***1*** | Owner | View sell products and their prices | **No of products sell in one day**  **No of products repair in one day**  **Money earned in one day by selling**  **Money earned in one day by repairing** |
| ***2*** | Owner | View repair products and their prices |
| ***3*** | Owner | Add selling products |
| ***4*** | Owner | Update selling products |
| ***5*** | Owner | Delete selling products |
| ***6*** | Owner | Add repairing products |
| ***7*** | Owner | Update repairing products |
| ***8*** | Owner | Delete repairing products |
| ***9*** | Owner | View employees record |
| ***10*** | Owner | View manager |
| ***11*** | Owner | Add manager |
| ***12*** | Owner | Remove manager |
| ***13*** | Owner | Update manager |
| ***14*** | Owner | Sort products with price |

## **EMPLOYEE:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***User Story ID*** | ***As a*** | ***I want to perform*** | ***So that I can*** |
| ***1*** | Employee | View products and their prices | **Add money earned in one day by selling**  **Add money earned in one day by repairing** |
| ***2*** | Employee | View repair products and their prices |
| ***3*** | Employee | Add number of products sell in one day |
| ***4*** | Employee | Add number of repair products in one day |

## **MANAGER:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***User Story ID*** | ***As a*** | ***I want to perform*** | ***So that I can*** |
| ***1*** | Manager | View record of employee | **View money**  **earned in one day**  **by selling**  **view money earned in one day by repairing**  **No of products repair in one day**  **No of products sell in one day** |
| ***2*** | Manager | View selling products and their prices |
| ***3*** | Manager | View repair products and their prices |
| ***4*** | Manager | Add employees |
| ***5*** | Manager | Remove employees |
| ***6*** | Manager | Update employees date |
| ***7*** | Manager | Check daily salary |

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

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

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

* Inheritance
* Polymorphism
* Association

## **Inheritance:**

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

In my project the concept of inheritance is used widely.

* In any system, there are different people with different jobs but mostly same attributes. In manual system, these attributes are name, age and city but if we talk about digital system or software’s these attributes are **username, password and role** so we make parent class of person which have these attributes and inherit it by admin and other staff members so that all of them use these attributes as their own.
* If we look into staff there are then many types of people which work as a staff but for a small shop there are mostly only two types of people **Employee, Managers**. When the admin hires these people, he assigns some **ID** to them which help them to login to their account and which is unique. So, Staff has an ID as a common attribute of employee and manager.
* In system there are some products on which the shop runs. If we talk about the automotive management system, these products are of **selling and repairing** type these products have some common attributes. So, we make one common class of **Product** and selling and repairing act as its child.
* If anyone wants to **calculate price** of products sell and repair, then he has to use the name and price of product so I inherit the money class with Product class and going further money is again calculated for repairing and selling product so they inherit **money** class as they use attributes of both money class and product class.

## **Polymorphism**

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

* I have polymorphised the check salary function in which employee or manager can check their monthly by adding work hours of day. The virtual function is created in Parent class and child classes override this function.

## **Association:**

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

# **Comparison with Procedural Programming:**

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

# **Design Pattern Implementation:**

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

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

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

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

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

**Parent Class:**

public class Person

{

protected string userName;

protected string password;

protected string role;

public bool setUsername(string username)

{

string pattern = "^[A-Za-z]+@[0-9]+$";

bool isMatch = Regex.IsMatch(username, pattern);

if ((string.IsNullOrEmpty(username) || !isMatch ))

{

return false;

}

else

{

this.userName = username;

return true;

}

}

public string getUsername()

{

return userName;

}

public bool setPassword(string password)

{

foreach(char c in password)

{

if ((password.Length <= 8 && password.Length >= 15) || char.IsLetter(c) || c == ',')

{

return false;

}

}

this.password = password;

return true;

}

public string getPassword()

{

return password;

}

public bool setRole(string role)

{

if (role == "Admin" || role == "Employee" || role == "Manager")

{

this.role = role;

return true;

}

return false;

}

public string getRole()

{

return role;

}

}

**Child Class:**

public class Selling: Product

{

private int Quantity;

public bool setQuantity (int Quantity)

{

string newProduct = Quantity.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || Quantity <= 0 || Quantity >= 70)

{

flag = false;

break;

}

else

{

this.Quantity = Quantity;

flag = true;

}

}

return flag;

}

public int getQunatity()

{

return Quantity;

}

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

public class Product

{

protected string productName;

protected int productPrice;

public Product()

{

}

public Product(string productName)

{

this.productName = productName;

}

public Product(string productName , int productPrice)

{

this.productName = productName;

this.productPrice = productPrice;

}

public bool setName(string productName)

{

bool flag = false;

foreach(char c in productName)

{

if(!(char.IsLetter(c)))

{

flag = false;

break;

}

else

{

this.productName = productName;

flag = true;

}

}

return flag;

}

public bool setPrice(int productPrice)

{

string newProduct = productPrice.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c))

{

flag = false;

break;

}

else

{

this.productPrice = productPrice;

flag = true;

}

}

return flag;

}

public string getProductName()

{

return productName;

}

public int getProductPrice()

{

return productPrice;

}

public virtual string toString()

{

return "ProductName" + "\t" + productName + "ProductPrice" + "\t" + productPrice;

}

The above snapshot of code show how we organize data into the Business layer class using constructor and getter setter method one thing in this class is toString() method which actually the well designated way to design a class.

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

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

**Sample Class**:

public class EmployeeDL

{

public static List<Employee> employee = new List<Employee>();

public static void AddIntoList(Employee e1)

{

employee.Add(e1);

}

public static bool checkID(string employeeID)

{

foreach(Employee e in employee)

{

if (e.getID() == employeeID)

{

return true;

}

}

return false;

}

public static void viewEmployee()

{

Console.WriteLine("UserName \t Password \t Employee ID “);

foreach (Employee store in employee)

{

Console.WriteLine(store.getUsername() + "\t" + store.getPassword() + "\t" + store. getID());

}

}

public static int searchEmployee (string newEmployee)

{

Employee e = new Employee();

for (int x = 0; x < employee.Count; x++)

{

if (employee == null)

{

return -1;

}

else if (newEmployee == employee[x].getUsername())

{

return x;

}

}

return -1;

} public static bool readFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string userName = splittedRecord[0];

string password = splittedRecord[1];

string employeeID = splittedRecord[2];

Employee e1 = new Employee(userName, password, employeeID);

AddIntoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeInFile(string path)

{

StreamWriter file = new StreamWriter(path,false);

foreach (Employee s in EmployeeDL.employee)

{

file.WriteLine(s.getUsername() + "," + s.getPassword() + "," + s.getID());}

file.Flush();

file.Close();

}}

## **User Interface:**

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

**Sample:**

public static void addEmployee()

{

Person p = new Person();

Employee e = new Employee();

string takeInput = "1";

while (takeInput == "1")

{

Console.WriteLine("Entre the username of employee");

string username = Console.ReadLine();

Console.WriteLine("Entre the password of employee");

string password = Console.ReadLine();

Console.WriteLine("Entre the ID of employee");

string employeeID = Console.ReadLine();

bool flag2 = EmployeeDL.Check(password , username , employeeID);

bool usernameFlag = p.setUsername(username);

bool passwordFlag = p.setPassword(password);

bool employeeFlag = e.setID(employeeID);

if (usernameFlag && passwordFlag && employeeFlag && flag2)

{

Person p1 = new Person(username, password, "Employee");

Employee e1 = new Employee(username, password, employeeID, "Employee");

EmployeeDL.AddIntoList(e1);

PersonDL.storeRoleInList(p1);

Console.WriteLine("Employee added successfully.");

}

else

{

Console.WriteLine("Wrong format of input ");

}

Console.WriteLine("Enter 1 to enter details for another employee, otherwise enter 0:");

takeInput = (Console.ReadLine());

}

}

This sample function take input from employees and employee ass to the software means it show user performed functions.

# **Key Classes:**

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

* + - * Person
      * Product

And all the other classes Behave as the child’s of these classes. When we see the Person class there are two types of person in any system own is admin and other is **Staff** and going forward Staff class consist of two main persons and that ae employees and manager. We see that Staff class is the child class of Person but it behaves as parent of employee and manager class.

The calculateSalary () function is an example of polymorphism in this code. There is virtual function in the Staff class and override method in Employee class.

**Product:**

The another major class in my project is Product class. As we know in any automotive shop, we deal with two products selling and repairing and we have to calculate money in accordance. So all these things base on the Product class. So I make the parent class as Product and inherit it by selling repairing and money calculated. The money calculate is again base on the product is either selling and repairing.

**Selling class:**

The selling class consist of its personal attribute of number of products present in the stock to sell. These number are useful to check what number of stock is present in the system.

**Repairing Class:**

The repairing class has list of defects present in the object and that can be repaired. These list can be useful to check which defect is present and what number of defect can a shop remove.

**Money:**

Money is also the child of Product class because it has to calculate the money of product but we have two types of product selling and repairing so we have to calculate their money separately base on the type.

**Money Sell:**

As money is of two types, so money sell is the child of money class which can access all the properties of money as well as product class.

**Money Repair:**

Same goes for the money of repair to calculate money on the basis of defect removed in a product. All these classes make a hierarchy from product to money and money to money sell and repair.

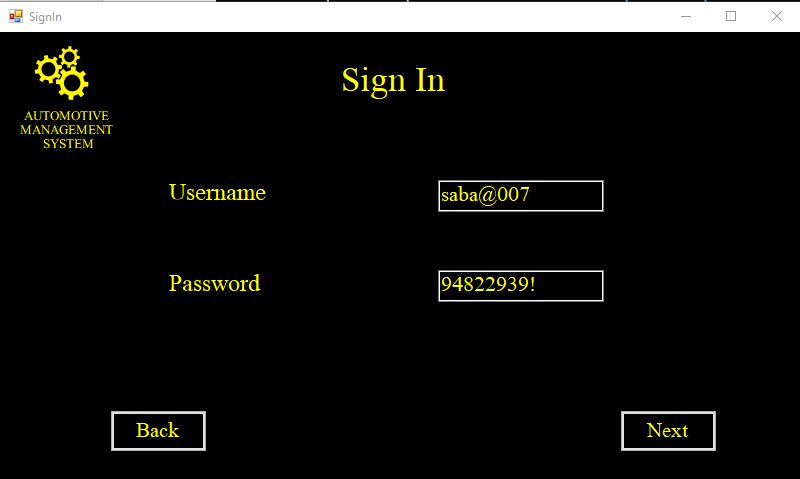
## **Wireframes:**

### **Main Screen:**



### **Role Screen**

### **Admin SignIn:**



### **ID Screen:**

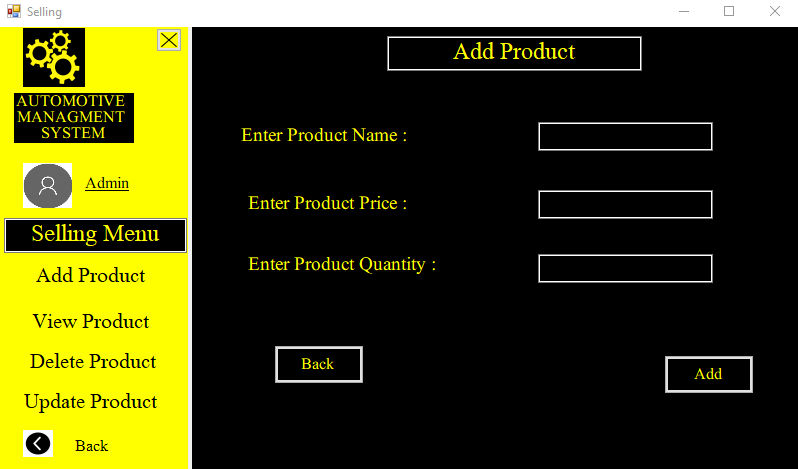
### **Admin Menu:**



### **Selling Products:**

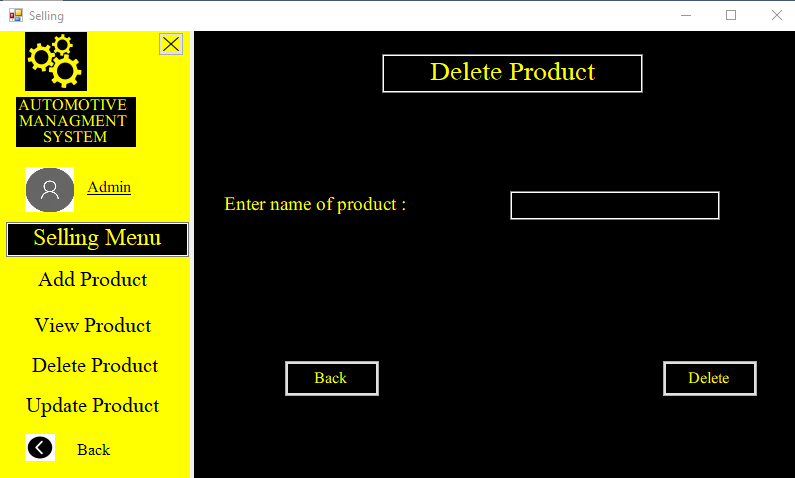


### **Add Selling Product:**



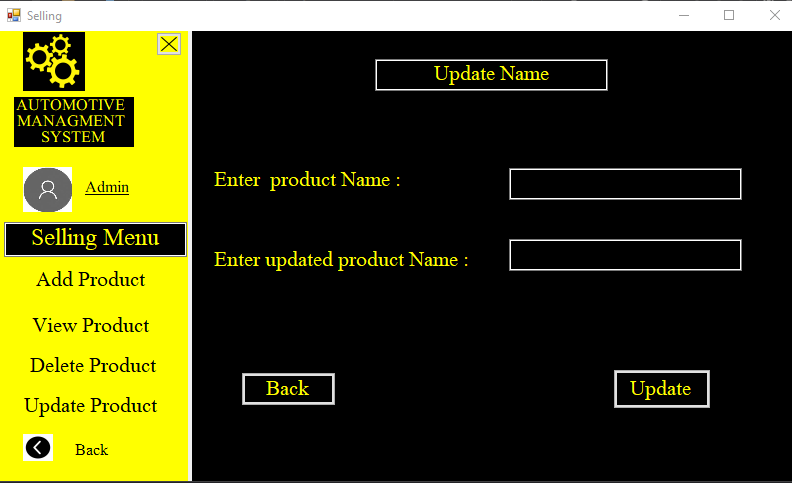
### **View Selling Product:**

### **Delete Selling Product:**

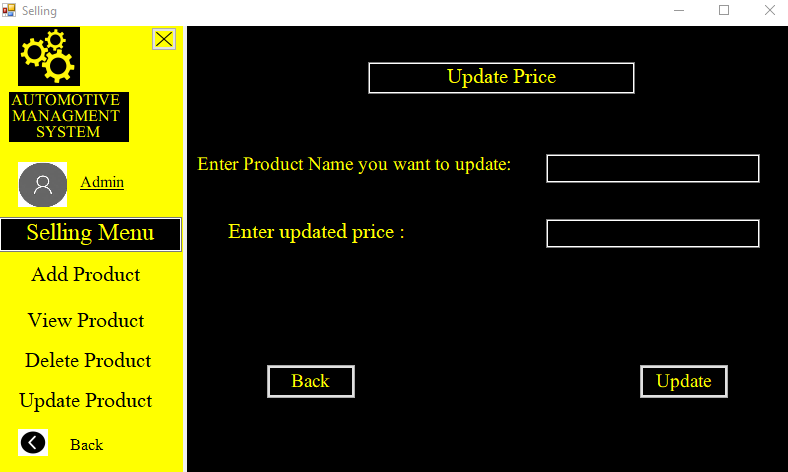


### **Update Selling Product:**

### **Update Selling Product:**



### **Update Selling Product:**



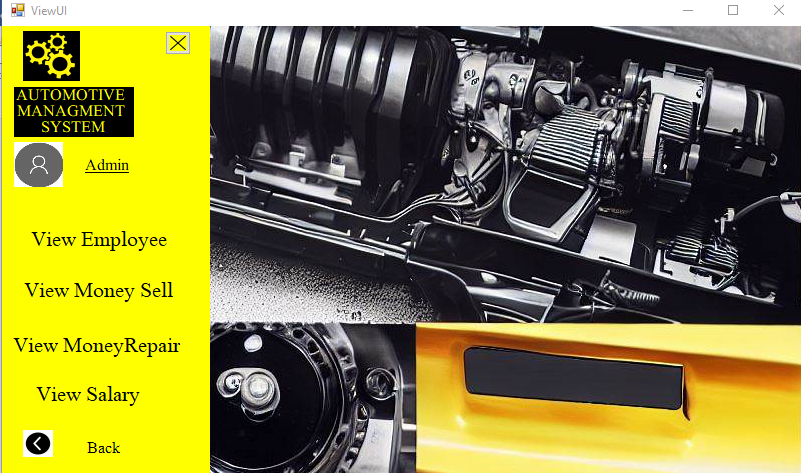
### **Update Selling Product:**

### **Repairing Product:**

### **Repairing Add Product:**

### **Repairing View Product**

### **Check Record:**



### **Check Record:**

### **Manager:**

### **Manager Add:**



### **Manager View:**

### **Update Manager:**

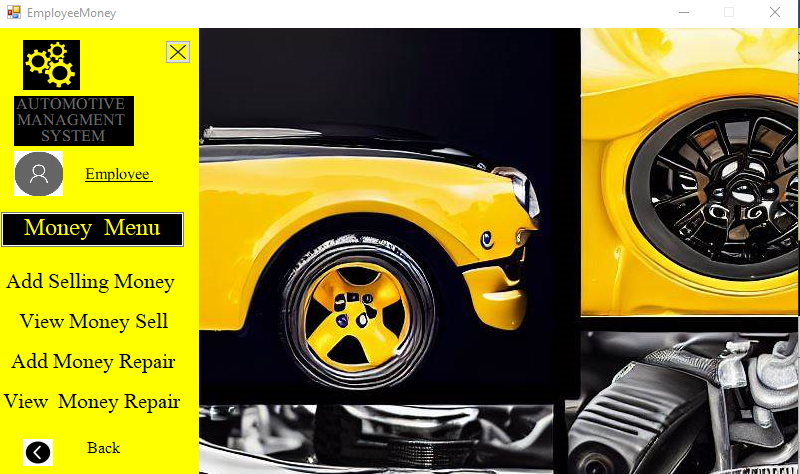
* **Employee Menu** 

### **Product Menu:**

### **View Selling Products:**

### **View Repairing Menu:**

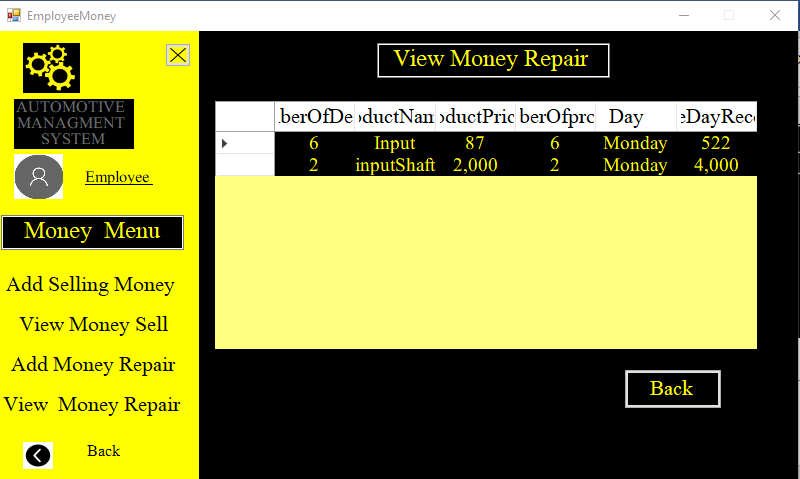
### **Money Menu:**



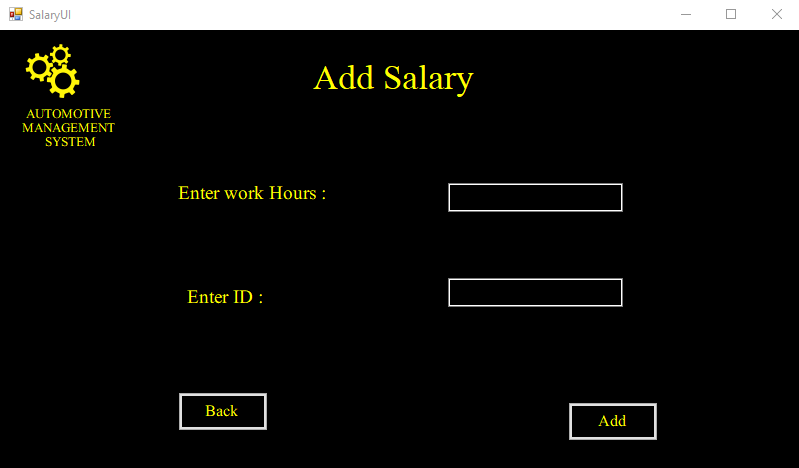
### **View Selling Money:**

### **Add Repairing Money:**

### **View Repairing Money:**

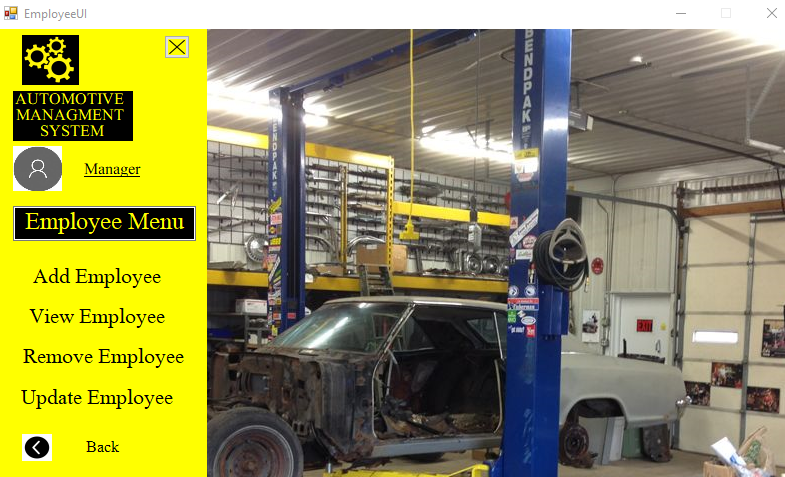


### **Add Salary:**



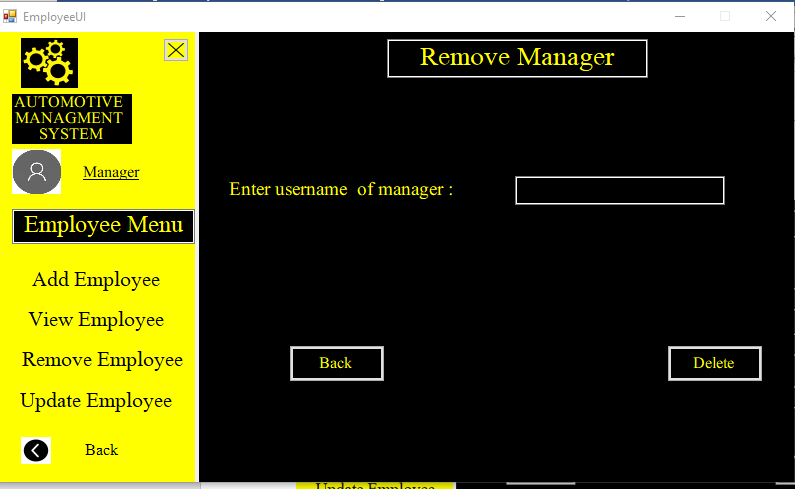
### **Manager Menu:**

### **Employee Menu:**



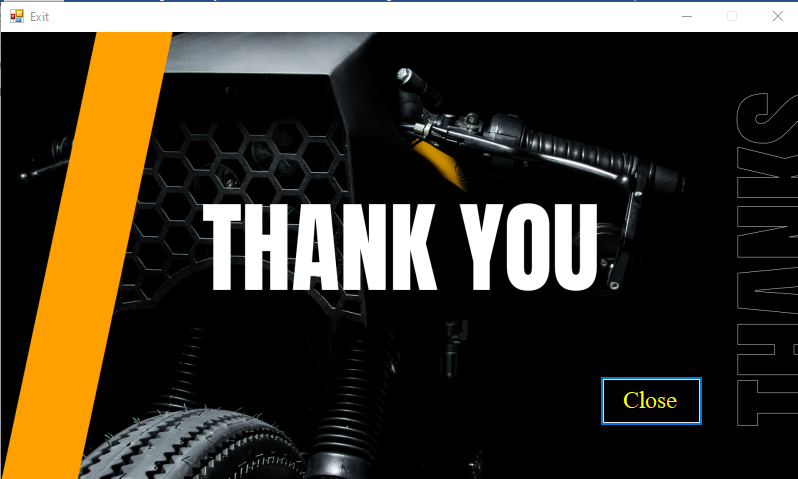
### **Add Employee:**

### **Remove Employee:**



### **Money Menu:**

### **End Screen:**



## **Resources:**

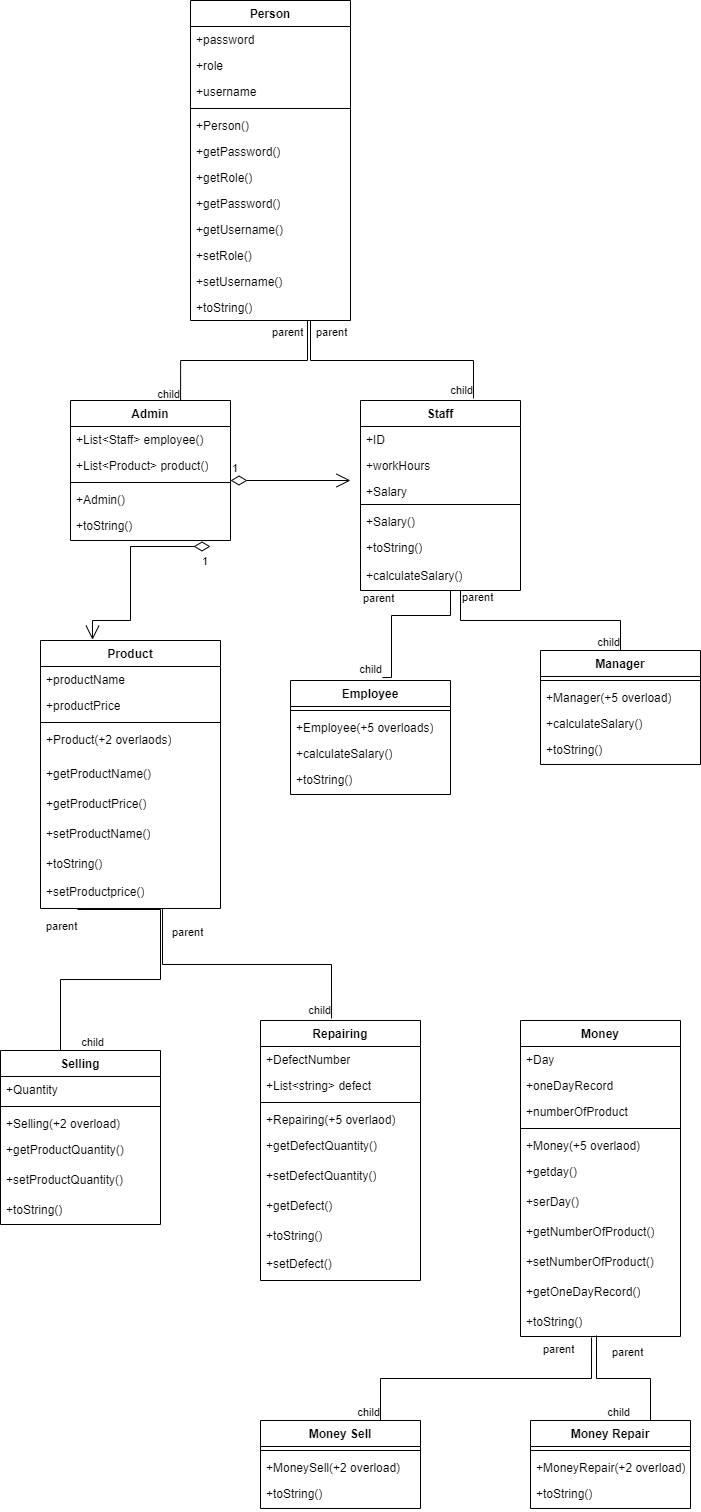




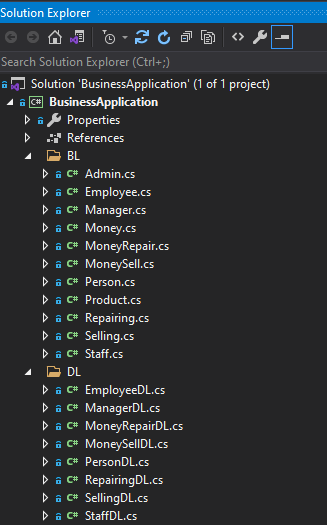
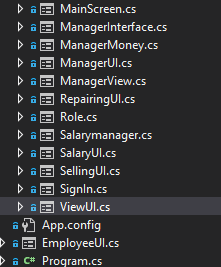
## 

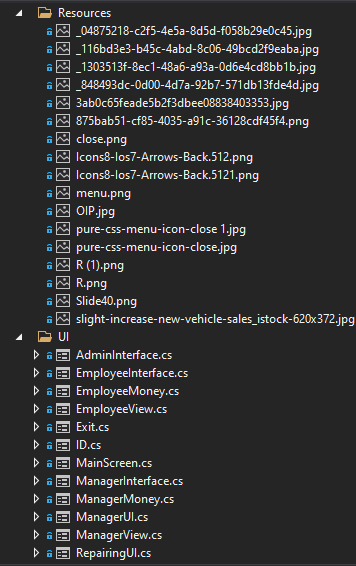


## **Class Diagram:**



## **Code:**





# **Business Logic:**

### **Person:**

namespace BusinessApplication.BL

{

public class Person

{

private string userName;

private string password;

private string role;

public string UserName { get => userName; set => userName = value; }

public string Password { get => password; set => password = value; }

public string Role { get => role; set => role = value; }

public Person()

{

}

public Person(string role)

{

this.Role = role;

}

public Person(string username, string password)

{

this.UserName = username;

this.Password = password;

}

public Person(string username, string password, string role)

{

this.UserName = username;

this.Password = password;

this.Role = role;

}

public bool setUsername(string username)

{

string pattern = "^[A-Za-z]+@[0-9]+$";

bool isMatch = Regex.IsMatch(username, pattern);

if ((string.IsNullOrEmpty(username) || !isMatch ))

{

return false;

}

else

{

this.UserName = username;

return true;

}

}

public string getUsername()

{

return UserName;

}

public bool setPassword(string password)

{

foreach(char c in password )

{

if ((password.Length <= 8 && password.Length >= 15) || char.IsLetter(c)|| c == ',')

{

return false;

}

}

this.Password = password;

return true;

}

public string getPassword()

{

return Password;

}

public bool setRole(string role)

{

if (role == "Admin" || role == "Employee" || role == "Manager")

{

this.Role = role;

return true;

}

return false;

}

public string getRole()

{

return Role;

}

public virtual string toString()

{

return "userName" + "\t" + UserName + "password" + "\t" + Password + "role" +"\t"+ Role;

}

}

### **Admin:**

public class Admin : Person

{

public static List<Employee> employees = new List<Employee>();

public static List<Manager> managers = new List<Manager>();

public static List<Product> products = new List<Product>();

public Admin()

{

}

public Admin( string role) : base( role)

{

this.Role = role;

}

public Admin(string username, string password) : base(username, password)

{

this.UserName = username;

this.Password = password;

}

public Admin(string username,string password,string role):base(username, password, role)

{

this.UserName = username;

this.Password = password;

this.Role = role;

}

public override string toString()

{

return base.toString();

}

}

### **Staff:**

public class Staff :Person

{

private string iD;

private int workHours;

private int salary;

public string ID { get => iD; set => iD = value; }

public int WorkHours { get => workHours; set => workHours = value; }

public int Salary { get => salary; set => salary = value; }

public Staff()

{

}

public Staff(string username, string password) : base(username, password)

{

this.UserName = username;

this.Password = password;

}

public Staff(string username, string password, string ID, string role) : base(username, password, role)

{

this.UserName = username;

this.Password = password;

this.ID = ID;

this.Role = role;

}

public Staff(string username, string password, string ID) : base(username, password , ID)

{

this.UserName = username;

this.Password = password;

this.ID = ID;

}

public Staff(string ID)

{

this.ID = ID;

}

public Staff(string ID, int workHour,string role)

{

this.ID = ID;

this.WorkHours = workHour;

this.Role = role;

}

public Staff(string ID, int workHour, string role , int salary)

{

this.ID = ID;

this.WorkHours = workHour;

this.Role = role;

this.Salary = salary;

}

public override string toString()

{

return base.toString() + "\t" + ID;

}

public bool setID(string ID)

{

string pattern = "^[A-Za-z]+@[0-9]+$";

bool isMatch = Regex.IsMatch(ID , pattern);

if ((string.IsNullOrEmpty(ID)) || !isMatch)

{

return false;

}

this.ID = ID;

return true;

}

public string getID()

{

return ID;

}

public bool setWorkHour(int workHour)

{

string newProduct = workHour.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || workHour < 0)

{

flag = false;

break;

}

else

{

this.WorkHours = workHour;

flag = true;

}

}

return flag;

}

public int getWorkHour()

{

return WorkHours;

}

public int getSalary()

{

return Salary;

}

public virtual int calculateSalary()

{

return 0;

}

}

### **Employee:**

public class Employee : Staff

{

public Employee()

{

}

public Employee(string username, string password) : base(username, password)

{

this.UserName = username;

this.Password = password;

}

public Employee(string username, string password , string ID , string role) : base(username, password , role , ID)

{

this.UserName = username;

this.Password = password;

this.ID = ID;

this.Role = role;

}

public Employee(string username, string password , string ID) : base(username, password , ID)

{

this.UserName = username;

this.Password = password;

this.ID = ID;

}

public Employee( string ID)

{

this.ID = ID;

}

public Employee(string ID , int workHour , string role , int salary) : base(ID , workHour , role, salary)

{

this.ID = ID;

this.WorkHours = workHour;

this.Role = role;

this.Salary = salary;

}

public override string toString()

{

return base.toString();

}

public override int calculateSalary()

{

int monthlySalary = 12000;

if(WorkHours > 9)

{

Salary = monthlySalary + 4000;

}

else

{

Salary = monthlySalary;

}

return Salary;

}

}

### **Manager:**

public class Manager : Staff

{

public Manager()

{

}

public Manager(string username, string password) : base(username, password)

{

this.UserName = username;

this.Password = password;

}

public Manager(string username, string password, string ManagerID, string role) : base(username, password, role)

{

this.UserName = username;

this.Password = password;

this.ID = ManagerID;

this.Role = role;

}

public Manager(string username, string password, string ManagerID) : base(username, password , ManagerID)

{

this.UserName = username;

this.Password = password;

this.ID = ManagerID;

}

public Manager(string ManagerID)

{

this.ID = ManagerID;

}

public Manager(string ID, int workHour, string role , int salary )

{

this.ID = ID;

this.WorkHours = workHour;

this.Role = role;

this.Salary = salary;

}

public override string toString()

{

return base.toString();

}

public override int calculateSalary()

{

int monthlySalary = 12000;

if (WorkHours > 9)

{

Salary = monthlySalary + 4000;

}

else

{

Salary = monthlySalary;

}

return Salary;

}

}

### **Product:**

public class Product

{

private string productName;

private int productPrice;

public string ProductName { get => productName; set => productName = value; }

public int ProductPrice { get => productPrice; set => productPrice = value; }

public Product()

{

}

public Product(string productName)

{

this.ProductName = productName;

}

public Product(string productName , int productPrice)

{

this.ProductName = productName;

this.ProductPrice = productPrice;

}

public bool setName(string productName)

{

bool flag = false;

foreach(char c in productName)

{

if(!(char.IsLetter(c)))

{

flag = false;

break;

}

else

{

this.ProductName = productName;

flag = true;

}

}

return flag;

}

public bool setPrice(int productPrice)

{

string newProduct = productPrice.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c))

{

flag = false;

break;

}

else

{

this.ProductPrice = productPrice;

flag = true;

}

}

return flag;

}

public string getProductName()

{

return ProductName;

}

public int getProductPrice()

{

return ProductPrice;

}

public virtual string toString()

{

return "ProductName" + "\t" + ProductName + "ProductPrice" + "\t" + ProductPrice;

}

}

### **Selling Product:**

public class Selling : Product

{

private int Quantity;

public int Quantity1 { get => Quantity; set => Quantity = value; }

public Selling()

{

}

public Selling(string productName , int productPrice) : base(productName , productPrice)

{

this.ProductName = productName;

this.ProductPrice = productPrice;

}

public Selling(string productName, int productPrice, int Quantity) : base(productName, productPrice)

{

this.ProductName = productName;

this.ProductPrice = productPrice;

this.Quantity1 = Quantity;

}

public bool setQuantity(int Quantity)

{

string newProduct = Quantity.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || Quantity <= 0 || Quantity >= 70)

{

flag = false;

break;

}

else

{

this.Quantity1 = Quantity;

flag = true;

}

}

return flag;

}

public int getQunatity()

{

return Quantity1;

}

public override string toString()

{

return base.toString();

}

}

### **Repairing Product:**

public class Repairing: Product

{

private int numberOfDefects;

public int NumberOfDefects { get => numberOfDefects; set => numberOfDefects = value; }

public Repairing()

{

}

public Repairing(string productName , int productPrice) : base(productName , productPrice)

{

this.ProductName = productName;

this.ProductPrice = productPrice;

}

public Repairing(string productName, int productPrice , int numberOfDefects) : base(productName, productPrice)

{

this.ProductName = productName;

this.ProductPrice = productPrice;

this.NumberOfDefects = numberOfDefects;

}

public bool setQuantity(int numberOfDefects)

{

string newProduct = numberOfDefects.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || numberOfDefects <= 0 || numberOfDefects > 10)

{

flag = false;

break;

}

else

{

this.NumberOfDefects = numberOfDefects;

flag = true;

}

}

return flag;

}

public int getNumber()

{

return NumberOfDefects;

}

public override string toString()

{

return base.toString();

}

}

### **Money:**

public class Money

{

private string ProductName;

private int ProductPrice;

private int numberOfproduct;

private string day;

private int oneDayRecord;

public string ProductName1 { get => ProductName; set => ProductName = value; }

public int ProductPrice1 { get => ProductPrice; set => ProductPrice = value; }

public int NumberOfproduct { get => numberOfproduct; set => numberOfproduct = value; }

public string Day { get => day; set => day = value; }

public int OneDayRecord { get => oneDayRecord; set => oneDayRecord = value; }

public Money()

{

}

public Money(int numberofProduct , string day )

{

this.NumberOfproduct = numberofProduct;

this.Day = day;

}

public Money(string productName , int productPrice , int numberofProduct, string day)

{

this.NumberOfproduct = numberofProduct;

this.Day = day;

this.ProductName1 = productName;

this.ProductPrice1 = productPrice;

}

public bool setDay(string day)

{

bool flag = false;

foreach (char c in day)

{

if (day == "Monday" || day == "Tuesday" || day == "Wednesday"|| day == "Thursday" || day == "Friday")

{

this.Day = day;

flag = true;

}

else

{

flag = false;

break;

}

}

return flag;

}

public bool setNumber(int numberOfProduct)

{

string newProduct = numberOfProduct.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || NumberOfproduct < 0 )

{

flag = false;

break;

}

else

{

this.NumberOfproduct = numberOfProduct;

flag = true;

}

}

return flag;

}

public string getDay()

{

return Day;

}

public int getNumber()

{

return NumberOfproduct;

}

public bool setName(string productName)

{

bool flag = false;

foreach (char c in productName)

{

if (!(char.IsLetter(c)))

{

flag = false;

break;

}

else

{

this.ProductName1 = productName;

flag = true;

}

}

return flag;

}

public bool setPrice(int productPrice)

{

string newProduct = productPrice.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c))

{

flag = false;

break;

}

else

{

this.ProductPrice1 = productPrice;

flag = true;

}

}

return flag;

}

public string getProductName()

{

return ProductName1;

}

public int getProductPrice()

{

return ProductPrice1;

}

public virtual string toString()

{

return "ProductName" + "\t" + ProductName1 + "ProductPrice" + "\t" + ProductPrice1 + "day" + "\t" + Day + "numberofProducts" + "\t" + NumberOfproduct;

}

}

### **Money Sell:**

public class MoneySell : Money

{

public MoneySell()

{

}

public MoneySell(string day , int numberOfProduct) : base (numberOfProduct , day)

{

this.Day = day;

this.NumberOfproduct = numberOfProduct;

}

public MoneySell( string productName , int productPrice , int numberofProduct, string day) : base( productName, productPrice , numberofProduct, day)

{

this.NumberOfproduct = numberofProduct;

this.Day = day;

this.ProductName1 = productName;

this.ProductPrice1 = productPrice;

}

public MoneySell(string productName, int productPrice , int numberOfProduct , string day , int oneDayRecord) : base ( productName, productPrice , numberOfProduct , day)

{

this.NumberOfproduct = numberOfProduct;

this.Day = day;

this.ProductName1 = productName;

this.ProductPrice1 = productPrice;

this.OneDayRecord = oneDayRecord;

}

public override string toString()

{

return base.toString();

}

}

### **Money Repair**

public class MoneyRepair : Money

{

private int numberOfDefects;

public MoneyRepair()

{

}

public MoneyRepair(string day, int numberOfProduct)

{

this.Day = day;

this.NumberOfDefects = numberOfProduct;

}

public MoneyRepair(string productName, int productPrice, int numberOfProduct , string day)

{

this.NumberOfDefects = numberOfProduct;

this.Day = day;

this.ProductName1 = productName;

this.ProductPrice1 = productPrice;

}

public MoneyRepair(string productName, int productPrice, int numberOfProduct, string day, int oneDayRecord) : base(productName, productPrice, numberOfProduct, day)

{

this.NumberOfDefects = numberOfProduct;

this.Day = day;

this.ProductName1 = productName;

this.ProductPrice1 = productPrice;

this.OneDayRecord = oneDayRecord;

}

public int NumberOfDefects { get => numberOfDefects; set => numberOfDefects = value; }

public int getNumberOfDefects()

{

return NumberOfDefects;

}

public bool setNumberOfDefects(int numberOfProduct)

{

string newProduct = numberOfProduct.ToString();

bool flag = false;

foreach (char c in newProduct)

{

if (char.IsLetter(c) || NumberOfproduct < 0)

{

flag = false;

break;

}

else

{

this.NumberOfDefects = numberOfProduct;

flag = true;

}

}

return flag;

}

public override string toString()

{

return base.toString();

}

}

# **Digital Layer:**

### **Person DL:**

public class PersonDL

{

public static List<Person> SignUp = new List<Person>();

public static void storeRoleInList(Person person)

{

SignUp.Add(person);

}

public static string signIn(Person person)

{

string role = null ;

foreach (Person storedUser in SignUp)

{

if (person.getUsername() == storedUser.getUsername() && person.getPassword() == storedUser.getPassword() )

{

role = storedUser.getRole();

}

}

return role;

}

public static bool readFromFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string userName = splittedRecord[0];

string password = splittedRecord[1];

string role = splittedRecord[2];

Person p1 = new Person(userName, password, role);

PersonDL.storeRoleInList(p1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeDataInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach(Person s in PersonDL.SignUp)

{

file.WriteLine(s.getUsername() + "," + s.getPassword() + "," + s.getRole());

}

file.Flush();

file.Close();

}

public static int searchPerson(string newPerson)

{

Person e = new Person();

for (int x = 0; x < SignUp.Count; x++)

{

if (SignUp == null)

{

return -1;

}

else if (newPerson == SignUp[x].getUsername())

{

return x;

}

}

return -1;

}

public static bool deletePerson(string newEmployee)

{

int delete = searchPerson(newEmployee);

bool flag = true;

if (delete != -1)

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

public static bool updatePerson(string newEmployee)

{

int update = searchPerson(newEmployee);

bool flag = true;

if (update != -1)

{

return flag;

}

else

{

flag = false;

}

return flag;

}

}

### **Staff DL:**

public class StaffDL

{

private static List<Staff> salary = new List<Staff>();

public static List<Staff> Salary { get => salary; set => salary = value; }

public static void storeSalaryInList(Staff person)

{

Salary.Add(person);

}

public static bool readSalaryFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string employeeID = splittedRecord[0];

int workHours = int.Parse(splittedRecord[1]);

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

string role = splittedRecord[3];

Staff e1 = new Staff(employeeID, workHours, role , salary);

storeSalaryInList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeSalaryInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (Staff s in Salary)

{

file.WriteLine(s.getID() + "," + s.getWorkHour() + "," + s.getSalary() + "," + s.getRole());

}

file.Flush();

file.Close();

}

}

### **Employee DL:**

public class EmployeeDL

{

private static List<Employee> employee = new List<Employee>();

public static List<Employee> Employee { get => employee; set => employee = value}

public static void AddIntoList(Employee e1)

{

Employee.Add(e1);

}

public static bool checkID(string employeeID)

{

foreach(Employee e in Employee)

{

if (e.getID() == employeeID)

{

return true;

}

}

return false;

}

public static void viewEmployee()

{

Console.WriteLine("UserName \t Password \t Employee ID ");

foreach (Employee store in Employee)

{

Console.WriteLine(store.getUsername() + "\t" + store.getPassword() + "\t" + store.getID())

}

}

public static int searchEmployee (string newEmployee)

{

Employee e = new Employee();

for (int x = 0; x < Employee.Count; x++)

{

if (Employee == null)

{

return -1;

}

else if (newEmployee == Employee[x].getUsername())

{

return x;

}

}

return -1;

}

public static bool deleteEmployee(string newEmployee)

{

int delete = searchEmployee( newEmployee);

bool flag = true;

if (delete != -1)

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

public static bool updateEmployee(string newEmployee)

{

int update = searchEmployee(newEmployee);

bool flag = true;

if (update != -1)

{

return flag;

}

else

{

flag = false;

}

return flag;

}

public static bool Check(string employeePassword , string employeeUserName , string employeeID)

{

bool flag = true;

foreach (Employee e in Employee)

{

if (e.getPassword() != employeePassword && e.getUsername() != employeeUserName && e.getID() != employeeID && employeeUserName != "saba@007" && employeePassword != "94822939!")

{

flag = true;

}

else

{

flag = false;

break;

}

}

return flag;

}

public static bool readFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string userName = splittedRecord[0];

string password = splittedRecord[1];

string employeeID = splittedRecord[2];

Employee e1 = new Employee(userName , password , employeeID);

AddIntoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeInFile(string path)

{

StreamWriter file = new StreamWriter(path,false);

foreach (Employee s in EmployeeDL.Employee)

{

file.WriteLine(s.getUsername() + "," + s.getPassword() + "," + s.getID());

}

file.Flush();

file.Close();

}

}

### **Manager DL:**

public class ManagerDL

{

private static List<Manager> manager = new List<Manager>();

public static List<Manager> Manager { get => manager; set => manager = value; }

public static void AddIntoManangerList(Manager e1)

{

Manager.Add(e1);

}

public static bool checkManagerID(string ManagerID)

{

foreach (Manager e in Manager)

{

if (e.getID() == ManagerID)

{

return true;

}

}

return false ;

}

public static void viewManager()

{

Console.WriteLine("UserName \t Password \t Manager ID ");

foreach (Manager store in Manager)

{

Console.WriteLine(store.getUsername() + "\t" + store.getPassword() + "\t" + store.getID());

}

}

public static int searchManager(string newManager)

{

Manager e = new Manager();

for (int x = 0; x < Manager.Count; x++)

{

if (Manager == null)

{

return -1;

}

else if (newManager == Manager[x].getUsername())

{

return x;

}

}

return -1;

}

public static bool deleteManager(string newManager)

{

int delete = searchManager(newManager);

bool flag = true;

if (delete != -1)

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

public static bool updateManager(string newManager)

{

int update = searchManager(newManager);

bool flag = true;

if (update != -1)

{

return flag;

}

else

{

flag = false;

}

return flag;

}

public static bool CheckManager(string ManagerPassword, string ManagerUserName, string ManagerID)

{

bool flag = true;

foreach (Manager e in Manager)

{

if (e.getPassword() != ManagerPassword && e.getUsername() != ManagerUserName && e.getID() != ManagerID && ManagerUserName != "saba@007" && ManagerPassword != "94822939!")

{

flag = true;

}

else

{

flag = false;

break;

}

}

return flag;

}

public static bool readManagerFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string userName = splittedRecord[0];

string password = splittedRecord[1];

string ManagerID = splittedRecord[2];

Manager e1 = new Manager(userName, password, ManagerID);

ManagerDL.AddIntoManangerList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeManagerInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (Manager s in ManagerDL.Manager)

{

file.WriteLine(s.getUsername() + "," + s.getPassword() + "," + s.getID())

}

file.Flush();

file.Close();

}

}

### **Selling DL:**

public class SellingDL

{

private static List<Selling> sell = new List<Selling>();

public static List<Selling> Sell { get => sell; set => sell = value; }

public static void AddSellingintoList(Selling s1)

{

Sell.Add(s1);

}

public static int searchSelling(string newProduct)

{

Selling s = new Selling();

for (int x = 0; x < Sell.Count; x++)

{

if (Sell == null)

{

return -1;

}

else if (newProduct == Sell[x].getProductName())

{

return x;

}

}

return -1;

}

public static bool deleteSelling(string newProduct)

{

int delete = searchSelling(newProduct);

bool flag = true;

if (delete != -1)

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

public static int updateProduct(string newEmployee)

{

int update = searchSelling(newEmployee);

if (update != -1)

{

return update;

}

else

{

return -1;

}

return update;

}

public static bool readSellingFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string productName = splittedRecord[0];

int productPrice = int.Parse(splittedRecord[1]);

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

Selling e1 = new Selling(productName , productPrice , Quantity);

AddSellingintoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeSellInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (Selling s in Sell)

{

file.WriteLine(s.getProductName() + "," + s.getProductPrice() + "," + s.getQunatity());

}

file.Flush();

file.Close();

}

}

* **Repairing DL:**

public class RepairingDL

{

private static List<Repairing> repair = new List<Repairing>();

public static List<Repairing> Repair { get => repair; set => repair = value; }

public static void AddRepairingintoList(Repairing s1)

{

Repair.Add(s1);

}

public static int searchRepairing(string newProduct)

{

Repairing s = new Repairing();

for (int x = 0; x < Repair.Count; x++)

{

if (Repair == null)

{

return -1;

}

else if (newProduct == Repair[x].getProductName())

{

return x;

}

}

return -1;

}

public static bool deleteRepairing(string newProduct)

{

int delete = searchRepairing(newProduct);

bool flag = true;

if (delete != -1)

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

public static bool updateRepairingProduct(string newProduct)

{

int update = searchRepairing(newProduct);

bool flag = true;

if (update != -1)

{

return flag;

}

else

{

flag = false;

}

return flag;

}

public static bool readRepairingFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string productName = splittedRecord[0];

int productPrice = int.Parse(splittedRecord[1]);

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

Repairing e1 = new Repairing(productName, productPrice , NumberOfDefects);

AddRepairingintoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storerepairInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach(Repairing s in RepairingDL.Repair)

{

file.WriteLine(s.getProductName() + "," + s.getProductPrice() + "," + s.getNumber());

}

file.Flush();

file.Close();

}

}

### **Money Sell DL:**

public class MoneySellDL

{

private static List<MoneySell> money = new List<MoneySell>();

public static List<MoneySell> Money { get => money; set => money = value; }

public static void AddIntoList(MoneySell m1)

{

Money.Add(m1);

}

public static int searchPrice(string productName)

{

foreach(Selling s in SellingDL.Sell)

{

if (s == null)

{

return -1;

}

if (s.getProductName() == productName)

{

return s.getProductPrice();

}

}

return -1;

}

public static int searchQuantity(string productName)

{

foreach (Selling s in SellingDL.Sell)

{

if (s == null)

{

return -1;

}

if (productName == s.getProductName())

{

return s.getQunatity();

}

}

return -1;

}

public static void viewMoneySelling(string day)

{

int sum = 0;

foreach(MoneySell m in Money)

{

if (m.getDay() == day)

{

sum = sum + m.OneDayRecord;

}

}

Console.WriteLine("Money earned is ...." + sum);

}

public static bool readMoneySellFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string productName = splittedRecord[0];

int productPrice = int.Parse(splittedRecord[1]);

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

string day = splittedRecord[3];

int oneDayRecord = int.Parse(splittedRecord[4]);

MoneySell e1 = new MoneySell(productName, productPrice , numberOfProduct , day , oneDayRecord);

MoneySellDL.AddIntoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeMoneySellInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (MoneySell s in Money)

{

file.WriteLine(s.getProductName() + "," + s.getProductPrice() + "," + s.getNumber() + "," +s.getDay()+ "," + s.OneDayRecord );

}

file.Flush();

file.Close();

}

}

}

### **Money Repair DL:**

public class MoneyRepairDL

{

private static List<MoneyRepair> moneyRepair = new List<MoneyRepair>();

public static List<MoneyRepair> MoneyRepair { get => moneyRepair; set => moneyRepair = value; }

public static void AddIntoList(MoneyRepair m1)

{

MoneyRepair.Add(m1);

}

public static int searchPrice(string productName)

{

foreach (Repairing s in RepairingDL.Repair)

{

if (s == null)

{

return -1;

}

if (s.getProductName() == productName)

{

return s.getProductPrice();

}

}

return -1;

}

public static int searchQuantity(string productName)

{

foreach (Repairing s in RepairingDL.Repair)

{

if (s == null)

{

return -1;

}

if (productName == s.getProductName() )

{

return s.getNumber();

}

}

return -1;

}

public static void viewMoneyRepairing(string day)

{

int sum = 0;

foreach (MoneyRepair m in MoneyRepair)

{

if (m.getDay() == day)

{

sum = sum + m.OneDayRecord;

}

}

Console.WriteLine("Money earned is ...." + sum);

}

public static bool readMoneyRepairFile(string path)

{

StreamReader f = new StreamReader(path);

string record;

if (File.Exists(path))

{

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

{

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

string productName = splittedRecord[0];

int productPrice = int.Parse(splittedRecord[1]);

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

string day = splittedRecord[3];

int oneDayRecord = int.Parse(splittedRecord[4]);

MoneyRepair e1 = new MoneyRepair(productName, productPrice, numberOfProduct, day, oneDayRecord);

MoneyRepairDL.AddIntoList(e1);

}

f.Close();

return true;

}

else

{

return false;

}

}

public static void storeMoneyRepairInFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (MoneyRepair s in MoneyRepair)

{

file.WriteLine(s.getProductName() + "," + s.getProductPrice() + "," + s.getNumber() + "," + s.getDay() + "," + s.OneDayRecord);

}

file.Flush();

file.Close();

}

}

# **User Interface:**

### **Main Screen:**

public partial class MainScreen : Form

{

public MainScreen()

{

InitializeComponent();

}

private void chkLogin\_CheckedChanged(object sender, EventArgs e)

{

if(chkLogin.Checked)

{

Form roleForm = new Role();

roleForm.Show();

this.Hide();

chkLogin.Checked = false;

}

}

private void chkExit\_CheckedChanged(object sender, EventArgs e)

{

if (chkExit.Checked)

{

Form form = new Exit();

form.Show();

this.Hide();

}

}

}

### **Main Role :**

public partial class Role : Form

{

public static List<string> role = new List<string>();

public Role()

{

InitializeComponent();

role.Add("Admin");

role.Add("Employee");

role.Add("Manager");

string path = "Person.txt";

if(PersonDL.readFromFile(path))

{

}

}

public void Role\_Load(object sender, EventArgs e)

{

cmbRole.DataSource = role;

}

public void cmbRole\_SelectedIndexChanged(object sender, EventArgs e)

{

string Selectedrole = cmbRole.SelectedItem.ToString();

Selectedrole = cmbRole.Text;

}

public void btnRole\_Click(object sender, EventArgs e)

{

if(cmbRole.SelectedItem.ToString() != null)

{

if (cmbRole.SelectedItem.ToString() == "Admin")

{

Form signForm = new SignIn();

signForm.Show();

this.Hide();

role.Clear();

}

else

{

Form idForm = new ID();

idForm.Show();

this.Hide();

role.Clear();

}

}

else

{

MessageBox.Show("Please Select something");

}

}

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

{

Form mainForm = new MainScreen();

mainForm.Show();

this.Hide();

role.Clear();

}

### **Sign IN:**

public partial class SignIn : Form

{

public SignIn()

{

InitializeComponent();

}

private void ClearDataFromForm()

{

txtUsername.Text = "";

txtPassword.Text = "";

}

private void SignIn\_Load(object sender, EventArgs e)

{

}

private void btnMenu\_Click\_1(object sender, EventArgs e)

{

string username = txtUsername.Text;

string password = txtPassword.Text;

Person person = new Person(username, password);

bool flag1 = person.setUsername(username);

bool flag2 = person.setPassword(password);

string flag = PersonDL.signIn(person);

if (flag1 && flag2)

{

txtResult.Visible = true;

txtResult.Text = "Signed in successfully";

if (flag == "Admin")

{

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

if (flag == "Employee")

{

Form sign = new EmployeeInterface();

sign.Show();

this.Hide();

}

if (flag == "Manager")

{

Form sign = new ManagerInterface();

sign.Show();

this.Hide();

}

}

else if (flag1 || flag2 || flag == null)

{

txtResult.Visible = true;

txtResult.Text = "Wrong Input";

}

ClearDataFromForm();

}

private void btnBack\_Click\_1(object sender, EventArgs e)

{

Form mainForm = new Role();

mainForm.Show();

this.Hide();

}

### **ID:**

public partial class ID : Form

{

public ID()

{

InitializeComponent();

}

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

{

Form sign = new SignIn();

sign.Show();

this.Hide();

}

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

{

Form sign = new Role();

sign.Show();

this.Hide();

}

private void ID\_Load(object sender, EventArgs e)

{

}

}

### **Admin Screen:**

public partial class AdminInterface : Form

{

public AdminInterface()

{

InitializeComponent();

}

private void radbtnSelling\_CheckedChanged(object sender, EventArgs e)

{

if(radbtnSelling.Checked)

{

Form form = new SellingUI();

form.Show();

this.Hide();

}

}

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

{

Form form = new MainScreen();

form.Show();

this.Hide();

}

private void AdminInterface\_Load(object sender, EventArgs e)

{

}

private void radbtnRepairing\_CheckedChanged(object sender, EventArgs e)

{

if (radbtnRepairing.Checked)

{

Form form = new RepairingUI();

form.Show();

this.Hide();

}

}

private void radbtnManager\_CheckedChanged(object sender, EventArgs e)

{

if (radbtnManager.Checked)

{

Form form = new ManagerUI();

form.Show();

this.Hide();

}

}

private void radbtnMoney\_CheckedChanged(object sender, EventArgs e)

{

if (radbtnMoney.Checked)

{

Form form = new ViewUI();

form.Show();

this.Hide();

}

}

}

### **Selling Products:**

public partial class SellingUI : Form

{

public SellingUI()

{

InitializeComponent();

string path = "Selling.txt";

if (SellingDL.readSellingFile(path))

{

}

}

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

{

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

AdminPanel.Visible = true;

}

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

{

SellingDL.storeSellInFile("Selling.txt");

SellingDL.Sell.Clear();

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

SellingTab.Visible = true;

tabAdd.Visible = true;

}

private void Selling\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

SellingTab.Visible = false;

SellingTab.Appearance = TabAppearance.FlatButtons;

SellingTab.ItemSize = new Size(0, 1);

tabView.Visible = false;

tabAdd.Visible = false;

}

private void ClearDataFromForm()

{

txtProductName.Text = "";

txtPrice.Text = "";

txtQuantity.Text = "";

}

private void btnAddProduct\_Click\_1(object sender, EventArgs e)

{

Selling s1 = new Selling();

string name = txtProductName.Text.ToLower();

string price = txtPrice.Text;

string Quantity = txtQuantity.Text;

if (int.TryParse(price, out int productPrice) && int.TryParse(Quantity, out int productQuantity))

{

bool usernameFlag = s1.setName(name);

bool passwordFlag = s1.setPrice(productPrice);

bool QuantityFlag = s1.setQuantity(productQuantity);

if (usernameFlag && passwordFlag && QuantityFlag)

{

Selling s2 = new Selling(name, productPrice, productQuantity);

SellingDL.AddSellingintoList(s2);

txtResult.Visible = true;

txtResult.Text = "Product Added successfully";

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

ClearDataFromForm();

}

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

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = true;

dataBind();

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = SellingDL.Sell;

ViewGrid.Refresh();

}

private void tabPage2\_Click(object sender, EventArgs e)

{

ViewGrid.DataSource = SellingDL.Sell;

}

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

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = true;

}

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

{

string name = txtAnswer.Text.ToLower();

int index = SellingDL.searchSelling(name);

if (SellingDL.deleteSelling(name))

{

txtResults.Visible = true;

txtResults.Text = "Product Found";

SellingDL.Sell.RemoveAt(index);

txtCon.Visible = true;

txtCon.Text = "Product removed successfully";

}

else

{

txtResults.Visible = false ;

txtResults.Text = "Product not Found";

}

txtAnswer.Text = "";

}

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

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = true;

}

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

{

Selling s = new Selling();

string name = txtEdit.Text.ToLower();

int index = SellingDL.searchSelling(name);

if (index != -1)

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Product Found";

string names = txtNames.Text;

bool usernameFlag = s.setName(names);

if (usernameFlag)

{

SellingDL.Sell[index].setName(names);

txtConclusion.Visible = true;

txtConclusion.Text = "Name updated successfully";

}

else

{

txtConclusion.Visible = true;

txtConclusion.Text = "Name can't be updated successfully";

}

}

else

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Product not Found";

}

txtEdit.Text = "";

txtNames.Text = "";

}

private void radNames\_CheckedChanged(object sender, EventArgs e)

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = true;

}

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

{

SellingTab.SelectedTab = tabUpdate;

}

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

{

Selling s = new Selling();

string name = txtOriginalName.Text.ToLower();

int index = SellingDL.searchSelling(name);

if (index != -1)

{

txtinfo.Visible = true;

txtinfo.Text = "Product Found";

string prices = txtUpdatedprices.Text;

if (int.TryParse(prices, out int productPrice))

{

bool usernameFlag = s.setPrice(productPrice);

if (usernameFlag)

{

SellingDL.Sell[index].setPrice(productPrice);

txtinfo1.Visible = true;

txtinfo1.Text = "Price updated successfully";

}

else

{

txtinfo1.Visible = true;

txtinfo1.Text = "Price can't be updated successfully";

}

}

else

{

txtinfo.Visible = true;

txtinfo.Text = "Product not Found";

}

}

txtOriginalName.Text = "";

txtUpdatePrice.Text = "";

}

private void radPrice\_CheckedChanged(object sender, EventArgs e)

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = true;

}

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

{

SellingTab.SelectedTab = tabUpdate;

}

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

{

Selling s = new Selling();

string name = txtOrginalName.Text.ToLower();

int index = SellingDL.searchSelling(name);

if (index != -1)

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Product Found";

string prices = txtUpdateQuantity.Text;

if (int.TryParse(prices, out int productPrice))

{

bool usernameFlag = s.setQuantity(productPrice);

if (usernameFlag)

{

SellingDL.Sell[index].setQuantity(productPrice);

txtInfos.Visible = true;

txtInfos.Text = "Quantity updated successfully";

}

else

{

txtInfos.Visible = true;

txtInfos.Text = "Quantity can't be updated ";

}

}

else

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Product not Found";

}

}

txtOrginalName.Text = "";

txtUpdateQuantity.Text = "";

}

private void radQuantity\_CheckedChanged(object sender, EventArgs e)

{

SellingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = true;

}

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

{

SellingTab.SelectedTab = tabUpdate;

}

public void Closer()

{

SellingTab.Visible = false;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = false;

}

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

{

Closer();

}

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

{

Closer();

}

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

{

Closer();

}

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

{

Closer();

}

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

{

AdminPanel.Visible = false;

}

### **Repairing Products:**

public partial class RepairingUI : Form

{

public RepairingUI()

{

InitializeComponent();

string path = "Repairing.txt";

if (RepairingDL.readRepairingFile(path))

{

}

}

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

{

AdminPanel.Hide();

}

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

{

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = true;

}

private void RepairingUI\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

RepairingTab.Visible = false;

RepairingTab.Appearance = TabAppearance.FlatButtons;

RepairingTab.ItemSize = new Size(0, 1);

tabView.Visible = false;

tabAdd.Visible = false;

}

private void ClearDataFromForm()

{

txtProductName.Text = "";

txtPrice.Text = "";

txtDefectNumber.Text = "";

}

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

{

Repairing s1 = new Repairing();

string name = txtProductName.Text.ToLower();

string price = txtPrice.Text;

string defects = txtDefectNumber.Text;

if (int.TryParse(price, out int productPrice) && int.TryParse(defects, out int numberOfdefect))

{

bool usernameFlag = s1.setName(name);

bool passwordFlag = s1.setPrice(productPrice);

bool QuantityFlag = s1.setQuantity(numberOfdefect);

if (usernameFlag && passwordFlag && QuantityFlag)

{

Repairing s2 = new Repairing(name, productPrice, numberOfdefect);

RepairingDL.AddRepairingintoList(s2);

txtResult.Visible = true;

txtResult.Text = "Product Added successfully";

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

ClearDataFromForm();

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

RepairingTab.SelectedTab = tabView;

dataBind();

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = RepairingDL.Repair;

ViewGrid.Refresh();

}

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

{

ViewGrid.DataSource = RepairingDL.Repair;

}

private void btnBack\_Click\_1(object sender, EventArgs e)

{

RepairingDL.storerepairInFile("Repairing.txt");

RepairingDL.Repair.Clear();

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

// tabDelete.Visible = true;

RepairingTab.SelectedTab = tabDelete;

}

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

{

string name = txtAnswer.Text.ToLower();

int index = RepairingDL.searchRepairing(name);

if (RepairingDL.deleteRepairing(name))

{

txtResults.Visible = true;

txtResults.Text = "Product Found";

RepairingDL.Repair.RemoveAt(index);

txtCon.Visible = true;

txtCon.Text = "Product removed successfully";

}

else

{

txtResults.Visible = false;

txtResults.Text = "Product not Found";

}

txtAnswer.Text = "";

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

RepairingTab.SelectedTab = tabUpdate;

// tabUpdate.Visible = true;

}

private void radNames\_CheckedChanged(object sender, EventArgs e)

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = true;

}

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

{

Repairing s = new Repairing();

string name = txtEdit.Text.ToLower();

int index = RepairingDL.searchRepairing(name);

if (index != -1)

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Product Found";

string names = txtNames.Text;

bool usernameFlag = s.setName(names);

if (usernameFlag)

{

RepairingDL.Repair[index].setName(names);

txtConclusion.Visible = true;

txtConclusion.Text = "Name updated successfully";

}

else

{

txtConclusion.Visible = true;

txtConclusion.Text = "Name can't be updated successfully";

}

}

else

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Product not Found";

}

txtEdit.Text = "";

txtNames.Text = "";

}

private void radPrice\_CheckedChanged(object sender, EventArgs e)

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = true;

}

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

{

Repairing s = new Repairing();

string name = txtOriginalName.Text.ToLower();

int index = RepairingDL.searchRepairing(name);

if (index != -1)

{

txtinfo.Visible = true;

txtinfo.Text = "Product Found";

string prices = txtUpdatedprices.Text;

if (int.TryParse(prices, out int productPrice))

{

bool usernameFlag = s.setPrice(productPrice);

if (usernameFlag)

{

RepairingDL.Repair[index].setPrice(productPrice);

txtinfo1.Visible = true;

txtinfo1.Text = "Price updated successfully";

}

else

{

txtinfo1.Visible = true;

txtinfo1.Text = "Price can't be updated successfully";

}

}

else

{

txtinfo.Visible = true;

txtinfo.Text = "Product not Found";

}

}

txtOriginalName.Text = "";

txtUpdatePrice.Text = "";

}

private void radQuantity\_CheckedChanged(object sender, EventArgs e)

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = true;

}

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

{

Repairing s = new Repairing();

string name = txtOrginalName.Text.ToLower();

int index = RepairingDL.searchRepairing(name);

if (index != -1)

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Product Found";

string prices = txtUpdateQuantity.Text;

if (int.TryParse(prices, out int productPrice))

{

bool usernameFlag = s.setQuantity(productPrice);

if (usernameFlag)

{

RepairingDL.Repair[index].setQuantity(productPrice);

txtInfos.Visible = true;

txtInfos.Text = "Quantity updated successfully";

}

else

{

txtInfos.Visible = true;

txtInfos.Text = "Quantity can't be updated ";

}

}

else

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Product not Found";

}

}

txtOrginalName.Text = "";

txtUpdateQuantity.Text = "";

}

public void Closer()

{

RepairingTab.Visible = false;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = false;

}

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

{

RepairingTab.SelectedTab = tabUpdate;

}

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

{

RepairingTab.SelectedTab = tabUpdate;

}

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

{

RepairingTab.SelectedTab = tabUpdate;

}

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

{

Closer();

}

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

{

Closer();

}

public void btnack\_Click(object sender, EventArgs e)

{

Closer();

}

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

{

Closer();

}

private void btnMenu\_Click\_1(object sender, EventArgs e)

{

AdminPanel.Show();

}

private void btnAddProduct\_Click\_1(object sender, EventArgs e)

{

}

}

### **Employee UI :**

public partial class EmployeeUI : Form

{

public EmployeeUI()

{

InitializeComponent();

string path = "Employee.txt";

if (EmployeeDL.readFile(path))

{

}

}

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

{

AdminPanel.Show();

}

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

{

AdminPanel.Hide();

}

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

{

EmployeeTab.Visible = true;

EmployeeTab.SelectedTab = tabAdd;

// tabAdd.Visible = true;

}

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

{

Person p = new Person();

Employee e2 = new Employee();

string username = txtUsernames.Text;

string password = txtpasswords.Text;

string ManagerID = txtID.Text;

bool flag2 = EmployeeDL.Check(password, username, ManagerID);

bool usernameFlag = p.setUsername(username);

bool passwordFlag = p.setPassword(password);

bool ManagerFlag = e2.setID(ManagerID);

if (usernameFlag && passwordFlag && ManagerFlag && flag2)

{

Person p1 = new Person(username, password, "Employee");

Employee e1 = new Employee(username, password, ManagerID, "Employee");

EmployeeDL.AddIntoList(e1);

PersonDL.storeRoleInList(p1);

txtResult.Visible = true;

txtResult.Text = "Employee Added successfully";

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

ClearDataFromForm();

}

private void ClearDataFromForm()

{

txtUsernames.Text = "";

txtpasswords.Text = "";

txtID.Text = "";

}

private void EmployeeUI\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

EmployeeTab.Visible = false;

EmployeeTab.Appearance = TabAppearance.FlatButtons;

EmployeeTab.ItemSize = new Size(0, 1);

tabView.Visible = false;

tabAdd.Visible = false;

}

public void btnView\_Click(object sender, EventArgs e)

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

EmployeeTab.SelectedTab = tabView;

dataBind();

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = EmployeeDL.Employee;

ConfigureViewGrid();

ViewGrid.Refresh();

}

public void ConfigureViewGrid()

{

ViewGrid.Columns.Remove("WorkHours");

ViewGrid.Columns.Remove("Salary");

ViewGrid.Columns.Remove("Role");

foreach (DataGridViewColumn column in ViewGrid.Columns)

{

if (column.Name != "ID" && column.Name != "UserName" && column.Name != "Password")

{

column.Visible = false;

}

}

ViewGrid.Columns["ID"].Visible = true;

ViewGrid.Columns["UserName"].Visible = true;

ViewGrid.Columns["Password"].Visible = true;

}

public void tabView\_Click(object sender, EventArgs e)

{

ConfigureViewGrid();

ViewGrid.DataSource = EmployeeDL.Employee;

}

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

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

EmployeeTab.SelectedTab = tabDelete;

}

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

{

string name = txtAnswer.Text;

int index = EmployeeDL.searchEmployee(name);

if (EmployeeDL.deleteEmployee(name))

{

txtResults.Visible = true;

txtResults.Text = "Employee Found";

EmployeeDL.Employee.RemoveAt(index);

txtCon.Visible = true;

txtCon.Text = " Employee removed successfully";

}

else

{

txtResults.Visible = false;

txtResults.Text = "Manager not Found";

}

txtAnswer.Text = "";

}

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

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

EmployeeTab.SelectedTab = tabUpdate;

}

private void radNames\_CheckedChanged(object sender, EventArgs e)

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = true;

}

private void radPassword\_CheckedChanged(object sender, EventArgs e)

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = true;

}

private void radID\_CheckedChanged(object sender, EventArgs e)

{

EmployeeTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = true;

}

public void Closer()

{

EmployeeTab.Visible = false;

}

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

{

Closer();

}

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

{

Person s = new Person();

string name = txtEdit.Text;

int index = EmployeeDL.searchEmployee(name);

if (index != -1)

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Employee Found";

string names = txtNames.Text;

bool usernameFlag = s.setUsername(name);

if (usernameFlag)

{

EmployeeDL.Employee[index].setUsername(names);

txtConclusion.Visible = true;

txtConclusion.Text = "Useranme updated successfully";

}

else

{

txtConclusion.Visible = true;

txtConclusion.Text = "Username can't be updated successfully";

}

}

else

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Employee not Found";

}

txtEdit.Text = "";

txtNames.Text = "";

}

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

{

EmployeeTab.SelectedTab = tabUpdate;

}

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

{

Person s = new Person();

string name = txtOriginalName.Text.ToLower();

int index = EmployeeDL.searchEmployee(name);

if (index != -1)

{

txtinfo.Visible = true;

txtinfo.Text = "Employee Found";

string password = txtUpdatedpassword.Text;

bool usernameFlag = s.setPassword(password);

if (usernameFlag)

{

EmployeeDL.Employee[index].setPassword(password);

txtinfo1.Visible = true;

txtinfo1.Text = "Password updated successfully";

}

else

{

txtinfo1.Visible = true;

txtinfo1.Text = "Password can't be updated successfully";

}

}

else

{

txtinfo.Visible = true;

txtinfo.Text = "Employee not Found";

}

txtOriginalName.Text = "";

txtUpdatedpassword.Text = "";

}

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

{

EmployeeTab.SelectedTab = tabUpdate;

}

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

{

Employee s = new Employee();

string name = txtOrginalName.Text;

int index = EmployeeDL.searchEmployee(name);

if (index != -1)

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Employee Found";

string ID = txtUpdateIDS.Text;

bool usernameFlag = s.setID(ID);

if (usernameFlag)

{

EmployeeDL.Employee[index].setID(ID);

txtInfos.Visible = true;

txtInfos.Text = "ID updated successfully";

}

else

{

txtInfos.Visible = true;

txtInfos.Text = "ID can't be updated ";

}

}

else

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Employee not Found";

}

txtOrginalName.Text = "";

txtUpdateIDS.Text = "";

}

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

{

EmployeeTab.SelectedTab = tabUpdate;

}

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

{

Closer();

}

public void btnack\_Click(object sender, EventArgs e)

{

Closer();

}

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

{

Closer();

}

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

{

EmployeeDL.storeInFile("Employee.txt");

PersonDL.storeDataInFile("Person.txt");

EmployeeDL.Employee.Clear();

Form sign = new ManagerInterface();

sign.Show();

this.Hide();

}

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

{

EmployeeDL.storeInFile("Employee.txt");

PersonDL.storeDataInFile("Person.txt");

EmployeeDL.Employee.Clear();

Form sign = new ManagerInterface();

sign.Show();

this.Hide();

}

}

### **Manager UI:**

public partial class ManagerUI : Form

{

public ManagerUI()

{

InitializeComponent();

string path = "Manager.txt";

if (ManagerDL.readManagerFile(path))

{

}

}

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

{

AdminPanel.Show();

}

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

{

AdminPanel.Hide();

}

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

{

ManagerTab.Visible = true;

tabAdd.Visible = true;

}

private void ManagerUI\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

ManagerTab.Visible = false;

ManagerTab.Appearance = TabAppearance.FlatButtons;

ManagerTab.ItemSize = new Size(0, 1);

tabView.Visible = false;

tabAdd.Visible = false;

}

private void ClearDataFromForm()

{

txtUsernames.Text = "";

txtpasswords.Text = "";

txtID.Text = "";

}

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

{

Person p = new Person();

Manager e2 = new Manager();

string username = txtUsernames.Text;

string password = txtpasswords.Text;

string ManagerID = txtID.Text;

bool flag2 = ManagerDL.CheckManager(password, username, ManagerID);

bool usernameFlag = p.setUsername(username);

bool passwordFlag = p.setPassword(password);

bool ManagerFlag = e2.setID(ManagerID);

if (usernameFlag && passwordFlag && ManagerFlag && flag2)

{

Person p1 = new Person(username, password, "Manager");

Manager e1 = new Manager(username, password, ManagerID, "Manager");

ManagerDL.AddIntoManangerList(e1);

PersonDL.storeRoleInList(p1);

txtResult.Visible = true;

txtResult.Text = "Manager Added successfully";

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong form of input ";

}

ClearDataFromForm();

}

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

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

// tabView.Visible = true;

ManagerTab.SelectedTab = tabView;

dataBind();

}

public void ConfigureViewGrid()

{

ViewGrid.Columns.Remove("WorkHours");

ViewGrid.Columns.Remove("Salary");

ViewGrid.Columns.Remove("Role");

foreach (DataGridViewColumn column in ViewGrid.Columns)

{

if (column.Name != "ID" && column.Name != "UserName" && column.Name != "Password")

{

column.Visible = false;

}

}

ViewGrid.Columns["ID"].Visible = true;

ViewGrid.Columns["UserName"].Visible = true;

ViewGrid.Columns["Password"].Visible = true;

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = ManagerDL.Manager;

ViewGrid.Refresh();

ConfigureViewGrid();

}

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

{

ViewGrid.DataSource = ManagerDL.Manager;

ConfigureViewGrid();

}

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

{

ManagerDL.storeManagerInFile("Manager.txt");

PersonDL.storeDataInFile("Person.txt");

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

ManagerDL.storeManagerInFile("Manager.txt");

PersonDL.storeDataInFile("Person.txt");

ManagerDL.Manager.Clear();

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

string name = txtAnswer.Text;

int index = ManagerDL.searchManager(name);

if (ManagerDL.deleteManager(name))

{

txtResults.Visible = true;

txtResults.Text = "Manager Found";

ManagerDL.Manager.RemoveAt(index);

txtCon.Visible = true;

txtCon.Text = "Manager removed successfully";

}

else

{

txtResults.Visible = false;

txtResults.Text = "Manager not Found";

}

txtAnswer.Text = "";

}

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

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

ManagerTab.SelectedTab = tabDelete;

// tabDelete.Visible = true;

}

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

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

ManagerTab.SelectedTab = tabUpdate;

}

private void radNames\_CheckedChanged(object sender, EventArgs e)

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = true;

}

private void radPassword\_CheckedChanged(object sender, EventArgs e)

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = true;

}

private void radID\_CheckedChanged(object sender, EventArgs e)

{

ManagerTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = true;

}

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

{

Person s = new Person();

string name = txtEdit.Text;

int index = ManagerDL.searchManager(name);

if (index != -1)

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Product Found";

string names = txtNames.Text;

bool usernameFlag = s.setUsername(name);

if (usernameFlag)

{

ManagerDL.Manager[index].setUsername(names);

txtConclusion.Visible = true;

txtConclusion.Text = "Useranme updated successfully";

}

else

{

txtConclusion.Visible = true;

txtConclusion.Text = "Username can't be updated successfully";

}

}

else

{

txtUpdateAnswer.Visible = true;

txtUpdateAnswer.Text = "Manager not Found";

}

txtEdit.Text = "";

txtNames.Text = "";

}

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

{

Person s = new Person();

string name = txtOriginalName.Text.ToLower();

int index = ManagerDL.searchManager(name);

if (index != -1)

{

txtinfo.Visible = true;

txtinfo.Text = "Manager Found";

string password = txtUpdatedpassword.Text;

bool usernameFlag = s.setPassword(password);

if (usernameFlag)

{

ManagerDL.Manager[index].setPassword(password);

txtinfo1.Visible = true;

txtinfo1.Text = "Password updated successfully";

}

else

{

txtinfo1.Visible = true;

txtinfo1.Text = "Password can't be updated successfully";

}

}

else

{

txtinfo.Visible = true;

txtinfo.Text = "Manager not Found";

}

txtOriginalName.Text = "";

txtUpdatedpassword.Text = "";

}

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

{

Manager s = new Manager();

string name = txtOrginalName.Text;

int index = ManagerDL.searchManager(name);

if (index != -1)

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Manager Found";

string ID = txtUpdateIDS.Text;

bool usernameFlag = s.setID(ID);

if (usernameFlag)

{

ManagerDL.Manager[index].setID(ID);

txtInfos.Visible = true;

txtInfos.Text = "ID updated successfully";

}

else

{

txtInfos.Visible = true;

txtInfos.Text = "ID can't be updated ";

}

}

else

{

txtUpdateinfo.Visible = true;

txtUpdateinfo.Text = "Manager not Found";

}

txtOrginalName.Text = "";

txtUpdateIDS.Text = "";

}

public void Closer()

{

ManagerTab.Visible = false;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdate.Visible = false;

tabUpdateName.Visible = false;

tabUpdatePrice.Visible = false;

tabUpdateQuantity.Visible = false;

}

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

{

ManagerTab.SelectedTab = tabUpdate;

}

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

{

ManagerTab.SelectedTab = tabUpdate;

}

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

{

ManagerTab.SelectedTab = tabUpdate;

}

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

{

Closer();

}

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

{

Closer();

}

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

{

Closer();

}

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

{

Closer();

}

### **View UI:**

public partial class ViewUI : Form

{

public ViewUI()

{

InitializeComponent();

}

private void ViewUI\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

ViewTab.Visible = false;

ViewTab.Appearance = TabAppearance.FlatButtons;

ViewTab.ItemSize = new Size(0, 1);

}

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

{

string path = "Employee.txt";

if (EmployeeDL.readFile(path))

{

}

ViewTab.Visible = true;

ViewTab.SelectedTab = tabView;

tabDelete.Visible = false;

tabUpdateName.Visible = false;

tabSalary.Visible = false;

if(tabView.Visible == true)

{

Condition();

}

dataBind();

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = EmployeeDL.Employee;

ConfigureViewGrid();

ViewGrid.Refresh();

}

public void ConfigureViewGrid()

{

ViewGrid.Columns.Remove("WorkHours");

ViewGrid.Columns.Remove("Salary");

ViewGrid.Columns.Remove("Role");

foreach (DataGridViewColumn column in ViewGrid.Columns)

{

if (column.Name != "ID" && column.Name != "UserName" && column.Name != "Password")

{

column.Visible = false;

}

}

ViewGrid.Columns["ID"].Visible = true;

ViewGrid.Columns["UserName"].Visible = true;

ViewGrid.Columns["Password"].Visible = true;

}

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

{

ViewTab.Visible = false;

tabView.Visible = false;

EmployeeDL.Employee.Clear();

Conditions();

}

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

{

ConfigureViewGrid();

ViewGrid.DataSource = EmployeeDL.Employee;

}

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

{

ViewTab.Visible = false;

tabView.Visible = false;

MoneySellDL.Money.Clear();

Conditions();

}

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

{

string path = "MoneySell.txt";

if (MoneySellDL.readMoneySellFile(path))

{

}

ViewTab.Visible = true;

tabView.Visible = false;

ViewTab.SelectedTab = tabDelete;

tabUpdateName.Visible = false;

tabSalary.Visible = false;

if (tabDelete.Visible == true)

{

Condition();

}

dataBind1();

}

public void dataBind1()

{

GridV.DataSource = null;

GridV.DataSource = MoneySellDL.Money;

GridV.Refresh();

}

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

{

GridV.DataSource = MoneySellDL.Money;

}

public void Condition ()

{

btnView.Enabled = false;

btnDelete.Enabled = false;

btnViewMoneySell.Enabled = false;

btnUpdate.Enabled = false;

}

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

{

ViewGrids.DataSource = MoneyRepairDL.MoneyRepair;

}

public void dataBind2()

{

ViewGrids.DataSource = null;

ViewGrids.DataSource = MoneyRepairDL.MoneyRepair;

ViewGrids.Refresh();

}

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

{

ViewTab.Visible = false;

MoneyRepairDL.MoneyRepair.Clear();

Conditions();

}

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

{

string path = "MoneyRepair.txt";

if (MoneyRepairDL.readMoneyRepairFile(path))

{

}

ViewTab.Visible = true;

tabView.Visible = false;

tabDelete.Visible = false;

tabSalary.Visible = false;

ViewTab.SelectedTab = tabUpdateName;

if(tabUpdateName.Visible == true)

{

Condition();

}

dataBind2();

}

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

{

ConfigureSalaryGrid();

SalaryGrid.DataSource = StaffDL.Salary;

}

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

{

string path = "Salary.txt";

if (StaffDL.readSalaryFile(path))

{

}

ViewTab.Visible = true;

tabView.Visible = false;

tabDelete.Visible = false;

tabUpdateName.Visible = false;

ViewTab.SelectedTab = tabSalary;

if(tabSalary.Visible == true)

{

Condition();

}

// tabSalary.Visible = true;

dataBind3();

}

public void Conditions()

{

btnView.Enabled = true;

btnDelete.Enabled = true;

btnViewMoneySell.Enabled = true;

btnUpdate.Enabled = true;

}

public void dataBind3()

{

SalaryGrid.DataSource = null;

SalaryGrid.DataSource = StaffDL.Salary;

ConfigureSalaryGrid();

SalaryGrid.Refresh();

}

public void ConfigureSalaryGrid()

{

SalaryGrid.Columns.Remove("UserName");

SalaryGrid.Columns.Remove("Password");

foreach (DataGridViewColumn column in SalaryGrid.Columns)

{

if (column.Name != "WorkHours" && column.Name != "Salary" && column.Name != "Role" && column.Name != "ID")

{

column.Visible = false;

}

}

SalaryGrid.Columns["WorkHours"].Visible = true;

SalaryGrid.Columns["Salary"].Visible = true;

SalaryGrid.Columns["Role"].Visible = true;

SalaryGrid.Columns["ID"].Visible = true;

}

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

{

ViewTab.Visible = false;

StaffDL.Salary.Clear();

Conditions();

}

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

{

AdminPanel.Hide();

}

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

{

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

StaffDL.Salary.Clear();

MoneyRepairDL.MoneyRepair.Clear();

EmployeeDL.Employee.Clear();

MoneySellDL.Money.Clear();

Form sign = new AdminInterface();

sign.Show();

this.Hide();

}

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

{

AdminPanel.Show();

}

### **Manager Interface:**

public partial class ManagerInterface : Form

{

public ManagerInterface()

{

InitializeComponent();

}

private void radProducts\_CheckedChanged(object sender, EventArgs e)

{

if(radProducts.Checked == true)

{

Form form = new ManagerView();

form.Show();

this.Hide();

}

}

private void radEmployee\_CheckedChanged(object sender, EventArgs e)

{

if(radEmployee.Checked == true)

{

Form form = new EmployeeUI();

form.Show();

this.Hide();

}

}

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

{

Form form = new MainScreen();

form.Show();

this.Hide();

}

private void radSalary\_CheckedChanged(object sender, EventArgs e)

{

if(radSalary.Checked == true)

{

Form form = new Salarymanager();

form.Show();

this.Hide();

}

}

private void radMoney\_CheckedChanged(object sender, EventArgs e)

{

if (radMoney.Checked == true)

{

Form form = new ManagerMoney();

form.Show();

this.Hide();

}

}

}

### **Employee interface:**

public partial class EmployeeInterface : Form

{

public EmployeeInterface()

{

InitializeComponent();

}

private void radProducts\_CheckedChanged(object sender, EventArgs e)

{

if (radProducts.Checked == true)

{

Form form = new EmployeeView();

form.Show();

this.Hide();

}

}

private void radMoney\_CheckedChanged(object sender, EventArgs e)

{

if (radMoney.Checked == true)

{

Form form = new EmployeeMoney();

form.Show();

this.Hide();

}

}

private void radSalary\_CheckedChanged(object sender, EventArgs e)

{

if (radSalary.Checked == true)

{

Form form = new SalaryUI();

form.Show();

this.Hide();

}

}

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

{

Form form = new MainScreen();

form.Show();

this.Hide();

}

}

### **Manager Money:**

public partial class ManagerMoney : Form

{

public ManagerMoney()

{

InitializeComponent();

}

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

{

string path = "MoneySell.txt";

if (MoneySellDL.readMoneySellFile(path))

{

}

ViewTab.Visible = true;

ViewTab.SelectedTab = tabDelete;

tabUpdateName.Visible = false;

if (tabDelete.Visible == true)

{

btnViewMoneySell.Enabled = false;

btnDelete.Enabled = false;

}

dataBind1();

}

public void dataBind1()

{

GridV.DataSource = null;

GridV.DataSource = MoneySellDL.Money;

GridV.Refresh();

}

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

{

ViewTab.Visible = false;

MoneySellDL.Money.Clear();

btnViewMoneySell.Enabled = true;

btnDelete.Enabled = true;

}

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

{

GridV.DataSource = MoneySellDL.Money;

}

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

{

string path = "MoneyRepair.txt";

if (MoneyRepairDL.readMoneyRepairFile(path))

{

}

ViewTab.Visible = true;

tabDelete.Visible = false;

ViewTab.SelectedTab = tabUpdateName;

if (tabUpdateName.Visible == true)

{

btnViewMoneySell.Enabled = false;

btnDelete.Enabled = false;

}

dataBind2();

}

public void dataBind2()

{

ViewGrids.DataSource = null;

ViewGrids.DataSource = MoneyRepairDL.MoneyRepair;

ViewGrids.Refresh();

}

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

{

ViewTab.Visible = false;

MoneyRepairDL.MoneyRepair.Clear();

btnViewMoneySell.Enabled = true;

btnDelete.Enabled = true;

}

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

{

AdminPanel.Hide();

}

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

{

Form sign = new ManagerInterface();

sign.Show();

this.Hide();

}

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

{

SellingDL.Sell.Clear();

RepairingDL.Repair.Clear();

Form sign = new EmployeeInterface();

sign.Show();

this.Hide();

}

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

{

AdminPanel.Show();

}

private void ManagerMoney\_Load(object sender, EventArgs e)

{

}

### **Employee Money:**

public partial class EmployeeMoney : Form

{

public EmployeeMoney()

{

InitializeComponent();

string path = "MoneySell.txt";

if (MoneySellDL.readMoneySellFile(path))

{

}

string path2 = "Selling.txt";

if (SellingDL.readSellingFile(path2))

{

}

string path1 = "Repairing.txt";

if (RepairingDL.readRepairingFile(path1))

{

}

string path3 = "MoneyRepair.txt";

if (MoneyRepairDL.readMoneyRepairFile(path3))

{

}

}

private void ClearDataFromForm()

{

txtProductName.Text = "";

txtPrice.Text = "";

txtDefectNumber.Text = "";

}

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

{

RepairingTab.Visible = true;

RepairingTab.SelectedTab = tabAdd;

}

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

{

string day = txtProductName.Text;

string productName = txtPrice.Text;

string numberOfProduct = txtDefectNumber.Text;

int price = MoneySellDL.searchPrice(productName);

if(price != -1)

{

if (int.TryParse(numberOfProduct, out int numberOfProducts))

{

int quantity = MoneySellDL.searchQuantity(productName);

if (quantity > numberOfProducts)

{

Money m = new Money();

int oneDayRecord = numberOfProducts \* price;

bool flag = m.setDay(day);

bool flag1 = m.setNumber(numberOfProducts);

bool flag2 = m.setPrice(price);

bool flag3 = m.setName(productName);

if (flag && flag1 && flag2 && flag3)

{

MoneySell m1 = new MoneySell(productName, price, numberOfProducts, day, oneDayRecord);

MoneySellDL.AddIntoList(m1);

txtAnswers.Visible = true;

txtAnswers.Text = "Money added ";

}

else

{

txtAnswers.Visible = true;

txtAnswers.Text = "Money cant be added";

}

}

else

{

txtResult.Visible = true;

txtResult.Text = "Quantity exceed stock";

}

}

else

{

txtAnswers.Visible = true;

txtAnswers.Text = "Wrong format of input";

}

}

else

{

txtAnswers.Visible = true;

txtAnswers.Text = "Product name don't find";

}

ClearDataFromForm();

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

RepairingTab.SelectedTab = tabView;

dataBind();

}

public void dataBind()

{

ViewGrid.DataSource = null;

ViewGrid.DataSource = MoneySellDL.Money;

ViewGrid.Refresh();

}

private void ViewGrid\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

ViewGrid.DataSource = MoneySellDL.Money;

}

private void EmployeeMoney\_Load(object sender, EventArgs e)

{

AdminPanel.Visible = true;

RepairingTab.Visible = false;

RepairingTab.Appearance = TabAppearance.FlatButtons;

RepairingTab.ItemSize = new Size(0, 1);

tabView.Visible = false;

tabAdd.Visible = false;

}

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

{

AdminPanel.Hide();

}

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

{

AdminPanel.Show();

}

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

{

MoneySellDL.storeMoneySellInFile("MoneySell.txt");

MoneySellDL.Money.Clear();

Form sign = new EmployeeInterface();

sign.Show();

this.Hide();

}

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

{

Form sign = new EmployeeInterface();

sign.Show();

this.Hide();

}

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

{

RepairingTab.Visible = false;

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

RepairingTab.SelectedTab = tabDelete;

}

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

{

string day = txtday.Text;

string productName = txtnamess.Text;

string numberOfProduct = txtdefect.Text;

int price = MoneyRepairDL.searchPrice(productName);

if (price != -1)

{

if (int.TryParse(numberOfProduct, out int numberOfProducts))

{

int quantity = MoneyRepairDL.searchQuantity(productName);

if (quantity > numberOfProducts)

{

Money m = new Money();

MoneyRepair m2 = new MoneyRepair();

int oneDayRecord = numberOfProducts \* price;

bool flag = m.setDay(day);

bool flag1 = m2.setNumberOfDefects(numberOfProducts);

bool flag2 = m.setPrice(price);

bool flag3 = m.setName(productName);

if (flag && flag1 && flag2 && flag3)

{

MoneyRepair m1 = new MoneyRepair(productName, price, numberOfProducts, day, oneDayRecord);

MoneyRepairDL.AddIntoList(m1);

txtCon.Visible = true;

txtCon.Text = "Money added ";

}

else

{

txtCon.Visible = true;

txtCon.Text = "Money cant be added";

}

}

else

{

txtResults.Visible = true;

txtResults.Text = "Quantity exceed stock";

}

}

else

{

txtCon.Visible = true;

txtCon.Text = "Wrong format of input";

}

}

else

{

txtCon.Visible = true;

txtCon.Text = "Wrong format of input";

}

txtday.Text = "";

txtnamess.Text = "";

txtdefect.Text = "";

}

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

{

RepairingTab.Visible = true;

tabAdd.Visible = false;

tabView.Visible = false;

tabDelete.Visible = false;

RepairingTab.SelectedTab = tabUpdateName;

dataBind1();

}

public void dataBind1()

{

ViewGrids.DataSource = null;

ViewGrids.DataSource = MoneyRepairDL.MoneyRepair;

ViewGrids.Refresh();

}

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

{

ViewGrids.DataSource = MoneyRepairDL.MoneyRepair;

}

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

{

RepairingTab.Visible = false;

}

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

{

RepairingTab.Visible = false;

}

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

{

RepairingTab.Visible = false;

}

}

### **Salary UI:**

public partial class SalaryUI : Form

{

public SalaryUI()

{

InitializeComponent();

string path = "Employee.txt";

if (EmployeeDL.readFile(path))

{

}

}

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

{

Employee e1 = new Employee();

string workHours = txtProductName.Text;

string ID = tXTIDS.Text;

if (int.TryParse(workHours, out int workHour) && EmployeeDL.checkID(ID))

{

if (e1.setWorkHour(workHour))

{

int salary = e1.calculateSalary();

Staff e2 = new Staff(ID, workHour, "Employee", salary);

txtResult.Visible = true;

string s = salary.ToString();

txtResult.Text = "Salary on the basis of Hour you work are " + s;

StaffDL.storeSalaryInList(e2);

}

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong format of input OR Employee Doesnot exist";

}

}

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

{

Form form = new EmployeeInterface();

form.Show();

this.Hide();

}

}

### **Salary Manager:**

public partial class Salarymanager : Form

{

public Salarymanager()

{

InitializeComponent();

string path = "Manager.txt";

if (ManagerDL.readManagerFile(path))

{

}

}

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

{

Manager e1 = new Manager();

string workHours = txtProductName.Text;

string ID = tXTIDS.Text;

if (int.TryParse(workHours, out int workHour) && ManagerDL.checkManagerID(ID))

{

if (e1.setWorkHour(workHour))

{

int salary = e1.calculateSalary();

Staff e2 = new Staff(ID, workHour, "Manager", salary);

string s = salary.ToString();

txtResult.Visible = true;

txtResult.Text = "Salary on the basis of Hour you work are " +salary;

StaffDL.storeSalaryInList(e2);

}

}

else

{

txtResult.Visible = true;

txtResult.Text = "Wrong format of input OR Manager Doesnot exist";

}

}

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

{

Form form = new ManagerInterface();

form.Show();

this.Hide();

}

# **Conclusion:**

Automotive management system is a well-established software for the business running as a small shop in the markets where work related to cars are done. In this project, every single module of such business are established and achieved.

## **Achievements:**

This system is capable of managing the staff and stock, keeping record of sales and money earned at the end and checking the salary and giving bonus if desired requirement is achieved. This system in short can handle all the workable requirement of small business on side roads on digital media rather than keeping on hand record.

## **Challenges:**

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