Pharmacy Management System

Session: 2022 – 2026

Submitted by:

Omar Baig 2022-CS-43

Submitted To:

Prof. Dr. Muhammad Awais Hassan

Department of Computer Science

University of Engineering and Technology Lahore Pakistan

Table Of Content

Project Documentation: Pharmacy Management System

Introduction

The Pharmacy Management System is a comprehensive software application developed to streamline the management of medicines, suppliers, customers, and workers in a pharmacy. The primary goal of this system is to automate and optimize the day-to-day operations, thus improving efficiency, accuracy, and productivity in the pharmacy.

Objectives

-Efficient Medicine Management: Enable pharmacists to maintain a digital inventory of medicines, update quantities, prices, and stocks, and track medicine availability.

- Supplier Data Management: Provide a centralized system to manage supplier information and the medicines supplied by each supplier.

- Customer Order Processing: Allow pharmacists to take and process customer orders, calculate billing amounts, and maintain order history.

- Worker Account Management: Provide an administrative interface to manage worker accounts and their roles in the system.

Intended Functionality

The Pharmacy Management System encompasses the following key functionalities:

1. Medicine Management:

- Pharmacists can add new medicines to the inventory, including details like medicine name, quantity, price, and stock availability.

- The system allows updating medicine information, such as price changes or stock adjustments, to reflect the current status accurately.

- Pharmacists can easily view and monitor the availability of medicines in stock.

2. Supplier Management:

- The system facilitates adding new suppliers and their associated medicine details to the database.

- Pharmacists can update supplier information and the medicines supplied by each supplier as needed.

- A user-friendly interface is provided to view and manage supplier data efficiently.

3. Customer Order Processing:

- Pharmacists can take and process customer orders for medicines, capturing details like medicine name, quantity, and customer information.

- The system automatically calculates the total billing amount for each customer order based on the ordered medicines' prices and quantities.

- A comprehensive order history is maintained to assist with future reference and billing accuracy.

4. Worker Account Management**:**

- Administrators can add new workers to the system and assign roles (administrator or worker) to each worker.

- Worker login authentication is implemented based on the provided credentials, ensuring secure access to the system.

2. OOP Concepts

Association

The Pharmacy Management System demonstrates the concept of association by establishing relationships between classes to represent real-world connections. For example, the `Medicine` class is associated with the `Supplier` class, reflecting the relationship between medicines and their respective suppliers. Similarly, the `Customer` class is associated with the `Medicine` class, representing the medicines included in a customer's order.

Inheritance

Inheritance is effectively used in the project to create specialized classes from a common base class. The `Person` class serves as the base class for `Admin` and `Worker`, providing a unified structure for both types of users. The derived classes inherit the properties and behaviors of the base class while allowing for specialized implementations as per the role.

Polymorphism

The project exhibits polymorphism through method overriding. The **`Admin`** and **`Worker`** classes override the **`SignIn`** method inherited from the base **`Person`** class, which demonstrates the ability to perform different actions based on the object's actual type. This flexibility allows the system to invoke the appropriate version of the method based on the user's role.

Advantages of OOP over Procedural Programming

- Code Reusability: OOP promotes code reusability through class inheritance, allowing developers to reuse existing code for new functionalities. This saves development time and reduces code duplication.

- Modularity: OOP encourages breaking down complex systems into smaller, manageable modules (classes). Each module handles specific tasks, making the code easier to understand and maintain.

- Encapsulation: OOP encapsulates data and methods within classes, protecting the data from unauthorized access. This enhances data security and reduces the chances of accidental data manipulation.

- Flexibility and Extensibility: OOP's polymorphism allows for flexible code that can adapt to different scenarios. It enables the addition of new functionalities without modifying existing code, promoting extensibility and scalability.

3. Design Pattern Implementation

Business Layer (BL) Pattern

The BL pattern is effectively utilized in the Pharmacy Management System to separate business logic from the user interface and data logic layer. The BL classes, such as **`MedicineCRUD`, `SupplierCRUD`, `CustomerCRUD`**, and **`PersonCRUD`**, encapsulate core business functionalities. This separation ensures that any changes to the business logic do not affect the UI layer and a bit to data logic layer.

Data Access Layer (DL) Pattern

The DL pattern is implemented to abstract the data storage and retrieval operations from the rest of the system and to perform logical operations. The DL classes use file-based storage to store and retrieve data, allowing for data

consistency and easy maintenance. Classes like **`MedicineCRUD`** and **`SupplierCRUD`** handle data persistence and ensure that the data is accessible from multiple locations.

User Interface (UI) Pattern

The UI pattern focuses on providing an intuitive and user-friendly interface for interacting with the pharmacy management system. The various Windows Forms, such as **`Form1`, `SignInMenu`, `AdminDashboard`, `WorkerDashboard`, `TakeOrderUI`, `AddMedicine`, `AddSupplier`,** and `**PreviewOrderUI**`, are designed to present relevant functionalities to the users in a clear and organized manner.

4. Class Details

Medicine Class

- **Responsibilities:**

- Stores and manages information related to medicines, such as name, quantity, price, and stock availability.

- Provides methods to access and modify medicine details.

- Supports association with the `Supplier` class to identify the supplier of each medicine.

Supplier Class

- Responsibilities:

- Represents supplier information, including name and the medicines supplied by the supplier.

- Associates with the `Medicine` class to provide a list of medicines supplied by the supplier.

- Provides methods to manage supplier data.

Customer Class

- Responsibilities:

- Stores and manages customer details, including name, contact, address, and orders.

- Maintains a list of medicines ordered by the customer along with their quantities.

- Calculates the total billing amount for each order.

Admin Class (Inherits from Person)

- **Responsibilities:**

  - Represents administrators with specific privileges to manage the pharmacy system.

  - Implements the **`SignIn`** method to authenticate admin login.

  - Utilizes association with the `Medicine` class to manage medicine data.

Worker Class (Inherits from Person)

- Responsibilities:

  - Represents workers with limited privileges in the pharmacy system.

  - Implements the **`SignIn`** method to authenticate worker login.

  - Collaborates with the `Customer` and `Medicine` classes to process customer orders.

5. Conclusion

The Pharmacy Management System successfully achieves its objectives of automating the management of medicines, suppliers, customers, and workers in a pharmacy. By leveraging object-oriented programming concepts, such as association, inheritance, and polymorphism, the system promotes code reusability, modularity, and flexibility.

The implementation of BL, DL, and UI design patterns ensures that the system adheres to a structured and organized development approach. The separation of concerns allows for easy maintenance, testing, and extension of functionalities.

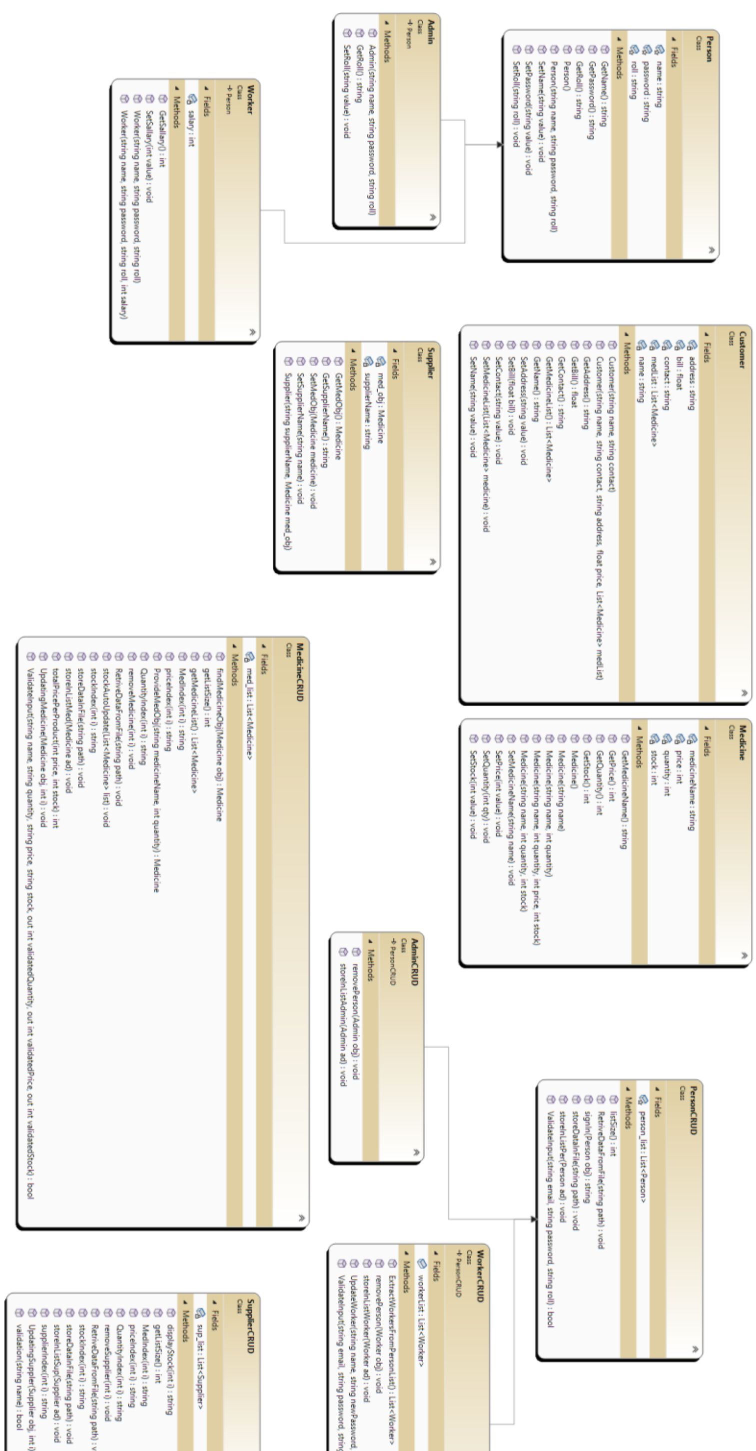
Throughout the development process, several challenges were encountered, such as data storage and retrieval from text files, validating user inputs, and handling various user interactions. However, by leveraging OOP principles and design patterns, I successfully addressed these challenges and delivered a robust pharmacy management system.

Although there can be made a lot of enhancements in it, but this design covers almost all the aspects of pharmacy management.

In conclusion, the Pharmacy Management System provides an efficient and user-friendly solution to streamline pharmacy operations and enhance overall productivity. It serves as a valuable tool for pharmacy administrators and workers to manage medicine inventory, process customer orders, and maintain supplier information effectively. The

system's adoption of OOP concepts and design patterns ensures scalability, maintainability, and adaptability to future enhancements.

Class Representative Diagram (CRC- Card)


Person
Admin
Worker
Customer
Supplier
MedicineCRUD
Medicine
AdminCRUD
PersonCRUD
WorkerCRUD
SupplierCRUD

Code:

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

namespace pharmacay_managemnt_system.BL
{
   internal class Customer
   {
      private string name;
      private string contact;
      private string address;
      private float bill;
      private List<Medicine> medList = new List<Medicine>();

      //Constructors
      public Customer(string name, string contact, string address, float price, List<Medicine> medList)
      {
         this.name = name;
         this.contact = contact;
         this.address = address;
         this.medList = medList;
         this.bill = price;
      }

      public Customer(string name, string contact)
```

```
    {
        this.name = name;
        this.contact = contact;
    }
    //getter and setters
    public string GetName()
    {
        return name;
    }

    public void SetName(string value)
    {
        name = value;
    }

    public string GetContact()
    {
        return contact;
    }

    public void SetContact(string value)
    {
        contact = value;
    }

    public string GetAddress()
    {
        return address;
    }
```

```
    public void SetAddress(string value)
    {
      address = value;
    }
    public float GetBill()
    {
      return bill;
    }

    public void SetBill(float bill)
    {
      this.bill = bill;
    }

    public List<Medicine> GetMedicineList()
    {
      return medList;
    }

    public void SetMedicineList(List<Medicine> medicine)
    {
      medList = medicine;
    }
  }
}

using System;
using System.Collections.Generic;
```

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

namespace pharmacay_managemnt_system.BL
{
   internal class Admin : Person
   {
      public Admin(string name, string password, string roll) : base(name, password, roll)
      {
      }

      public new string GetRoll()
      {
         return roll;
      }

      public new void SetRoll(string value)
      {
         roll = value;
      }
   }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace pharmacay_managemnt_system.BL
{
   internal class Person
   {
      protected string name;
      protected string password;
      protected string roll;

      public Person()
      {
         // Default constructor
      }

      public Person(string name, string password, string roll)
      {
         this.name = name;
         this.password = password;
         this.roll = roll;
      }

      public string GetName()
      {
         return name;
      }

      public void SetName(string value)
      {
         name = value;
      }
```

```
        public string GetRoll()
        {
            return roll;
        }

        public void SetRoll(string roll)
        {
            this.roll = roll;
        }

        public string GetPassword()
        {
            return password;
        }

        public void SetPassword(string value)
        {
            password = value;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pharmacay_managemnt_system.BL
{
```

```
    internal class Worker : Person
    {
        private int salary;



        public Worker(string name, string password, string roll ,int salary ) : base(name, password, roll)
        {
            this.salary = salary;
        }
        public Worker(string name, string password, string roll) : base(name, password, roll)
        {


        }
        public int GetSallary()
        {
            return this.salary;
        }


        public void SetSallary(int value)
        {
            salary = value;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace pharmacay_managemnt_system.BL
{
   internal class Customer
   {
      private string name;
      private string contact;
      private string address;
      private float bill;
      private List<Medicine> medList = new List<Medicine>();

      //Constructors
      public Customer(string name, string contact, string address, float price, List<Medicine> medList)
      {
         this.name = name;
         this.contact = contact;
         this.address = address;
         this.medList = medList;
         this.bill = price;
      }

      public Customer(string name, string contact)
      {
         this.name = name;
         this.contact = contact;
      }
      //getter and setters
      public string GetName()
      {
```

```
        return name;
    }

    public void SetName(string value)
    {
        name = value;
    }

    public string GetContact()
    {
        return contact;
    }

    public void SetContact(string value)
    {
        contact = value;
    }

    public string GetAddress()
    {
        return address;
    }

    public void SetAddress(string value)
    {
        address = value;
    }
    public float GetBill()
    {
```

```
        return bill;
    }

    public void SetBill(float bill)
    {
        this.bill = bill;
    }

    public List<Medicine> GetMedicineList()
    {
        return medList;
    }

    public void SetMedicineList(List<Medicine> medicine)
    {
        medList = medicine;
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pharmacay_managemnt_system.BL
{
  internal class Medicine
  {
```

```
    private string medicineName;
    private int quantity;
    private int price;
    private int stock;



    public Medicine(string name, int quantity, int price, int stock)
    {
       this.medicineName = name;
       this.quantity = quantity;
       this.price = price;
       this.stock = stock;
    }
    public Medicine(string name)
    {
       this.medicineName = name;
    }
    public Medicine()
    {
    }
    public Medicine(string name, int quantity)
    {
       this.medicineName = name;
       this.quantity = quantity;
    }
    public Medicine(string name, int quantity, int stock)
    {
       this.medicineName = name;
```

```
        this.quantity = quantity;
        this.stock = stock;
    }


    // \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ Getter and setters

    public string GetMedicineName()
    {
        return medicineName;
    }

    public void SetMedicineName(string name)
    {
        medicineName = name;
    }

    public int GetQuantity()
    {
        return quantity;
    }

    public void SetQuantity(int qty)
    {
        quantity = qty;
    }

    public int GetPrice()
    {
```

```
        return price;
    }

    public void SetPrice(int value)
    {
        price = value;
    }

    public int GetStock()
    {
        return stock;
    }

    public void SetStock(int value)
    {
        stock = value;
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pharmacay_managemnt_system.BL
{
  internal class Supplier
  {
```

```
    private string supplierName;
    private Medicine med_obj;

    public Supplier(string supplierName, Medicine med_obj)
    {
      this.supplierName = supplierName;
      this.med_obj = med_obj;
    }
    public string GetSupplierName()
    {
      return supplierName;
    }

    public void SetSupplierName(string name)
    {
      supplierName = name;
    }

    public Medicine GetMedObj()
    {
      return med_obj;
    }

    public void SetMedObj(Medicine medicine)
    {
      med_obj = medicine;
    }
  }
}
```

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

namespace pharmacay_managemnt_system.DL
{
   internal class AdminCRUD : PersonCRUD
   {
      public void storeInListAdmin(Admin ad)
      {
         if (!person_list.Contains(ad))
         {
            storeInListPer(ad);
         }
      }
      public void removePerson(Admin obj)
      {
         for (int i = 0; i < person_list.Count; i++)
         {
            if (person_list[i].GetName() == obj.GetName() && person_list[i].GetPassword() == obj.GetPassword() &&
person_list[i].GetRoll() == obj.GetRoll())
            {
               person_list.RemoveAt(i);
               break;
            }
         }
      }
```

```
    }
}
using pharmacay_managemnt_system.BL;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system.DL
{
    internal class PersonCRUD
    {
        protected static List<Person> person_list = new List<Person>();
        public static void storeInListPer(Person ad)
        {
            if (!person_list.Contains(ad))
            {
                person_list.Add(ad);
            }
        }
        public static string signIn(Person obj)
        {
            for (int i = 0; i < person_list.Count; i++)
            {
                if (person_list[i].GetName() == obj.GetName() && person_list[i].GetPassword() == obj.GetPassword() &&
person_list[i].GetRoll() == obj.GetRoll())
                {
```

```
            return person_list[i].GetRoll();
        }
    }
    return null;
}
public static int listSize()
{
    return person_list.Count;
}
public static void storeDataInFile(string path)
{
    StreamWriter file = new StreamWriter(path, true);
    for (int i = 0; i < person_list.Count(); i++)
    {
        file.WriteLine(person_list[i].GetName() + "," + person_list[i].GetPassword() + "," + person_list[i].GetRoll());
    }
    file.Flush();
    file.Close();
}
public static void RetriveDataFromFile(string path)
{
    StreamReader filevariable = new StreamReader(path);
    string record;
    while ((record = filevariable.ReadLine()) != null)
    {
        string[] resultarray = record.Split(',');
        string name = resultarray[0];
        string password = resultarray[1];
        string roll = resultarray[2];
```

```
            Person ad = new Person(name, password, roll);
            storeInListPer(ad);
        }
        filevariable.Close();
    }
    public static bool ValidateInput(string email, string password, string roll)
    {
        if (string.IsNullOrEmpty(email))
        {
            MessageBox.Show("Email cannot be empty.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
        }

        if (string.IsNullOrEmpty(password))
        {
            MessageBox.Show("Password cannot be empty.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
        }

        if (string.IsNullOrEmpty(roll))
        {
            MessageBox.Show("Roll cannot be empty.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
        }
        return true;
    }
}
```

```
}
using pharmacay_managemnt_system.BL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace pharmacay_managemnt_system.DL
{
   internal class WorkerCRUD : PersonCRUD
   {
      public static List<Worker> workerList = new List<Worker>();
      public static void storeInListWorker(Worker ad)
      {
         if (!person_list.Contains(ad))
         {
            storeInListPer(ad);
         }
      }
      public static void removePerson(Worker obj)
      {
         for (int i = 0; i < person_list.Count; i++)
         {
            if (person_list[i] is Worker worker && worker.GetName() == obj.GetName() && worker.GetPassword() ==
obj.GetPassword())
            {
               person_list.RemoveAt(i);
```

```
            break;
        }
    }
}
public static List<Worker> ExtractWorkersFromPersonList()
{
    List<Worker> workerList = new List<Worker>();

    for (int i = 0; i < person_list.Count; i++)
    {
        if (person_list[i] is Worker worker)
        {
            workerList.Add(worker);
        }
    }

    return workerList;
}
public static void UpdateWorker(string name, string newPassword, int newSalary)
{
    for (int i = 0; i < person_list.Count; i++)
    {
        if (person_list[i] is Worker worker && worker.GetName() == name)
        {
            worker.SetPassword(newPassword);
            worker.SetSallary(newSalary);
            break;
        }
    }
```

```
    }
    public static bool ValidateInput(string email, string password, string salary,out int validSalary)
    {
      if (string.IsNullOrEmpty(email))
      {
        MessageBox.Show("Email cannot be empty.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        validSalary = 0;
        return false;
      }

      if (string.IsNullOrEmpty(password))
      {
        MessageBox.Show("Password cannot be empty.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        validSalary = 0;
        return false;
      }

      if (!int.TryParse(salary,out validSalary) || validSalary <= 0)
      {
        MessageBox.Show("Invalid salary input.", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
      }
      return true;
    }
  }
}
using pharmacay_managemnt_system.BL;
using System;
```

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system.DL
{
   internal class CustomerCRUD
   {
      private static List<Customer> cust_list = new List<Customer>();

      public static void storeInListCust(Customer ad)
      {
         cust_list.Add(ad);
      }

      public static Customer getObj(Customer obj)
      {
         for (int i = 0; i < cust_list.Count; i++)
         {
            if (cust_list[i].GetName() == obj.GetName() && cust_list[i].GetContact() == obj.GetContact())
            {
               return cust_list[i];
            }
         }
         return null;
      }
```

```
public static int listCounts()
{
   return cust_list.Count;
}

public static void storeDataInFile(string path)
{
   using (StreamWriter file = new StreamWriter(path, true))
   {
      foreach (Customer customer in cust_list)
      {
         file.Write(customer.GetName() + "," + customer.GetContact() + "," + customer.GetAddress() + "," +
customer.GetBill() + ",");
         foreach (Medicine medicine in customer.GetMedicineList())
         {
            file.Write( medicine.GetMedicineName() + "/" + medicine.GetQuantity() + "/" + medicine.GetPrice() + "/" +
medicine.GetStock() + "/" + "#");
         }
         file.WriteLine("#,");
      }
   }
}

public static void RetriveDataFromFile(string path)
{
   if (!File.Exists(path))
   {
      return;
```

```
        }

        cust_list.Clear();

        using (StreamReader filevariable = new StreamReader(path))
        {
            string record;
            while ((record = filevariable.ReadLine()) != null)
            {
                string[] splittedRecord = record.Split(',');

                if (splittedRecord.Length >= 5)
                {
                    string name = splittedRecord[0];
                    string contact = splittedRecord[1];
                    string address = splittedRecord[2];
                    float bill = float.Parse(splittedRecord[3]);

                    List<Medicine> med = new List<Medicine>();

                    // Check if there is data for medicines
                    if (splittedRecord.Length >= 6)
                    {
                        string[] medObj = splittedRecord[4].Split('#');
                        foreach (string medData in medObj)
                        {
                            string[] objData = medData.Split('/');
                            if (objData.Length >= 4)
                            {
```

```
                    string medName = objData[0];
                    int quantity = int.Parse(objData[1]);
                    int price = int.Parse(objData[2]);
                    int stock = int.Parse(objData[3]);
                    Medicine m = new Medicine(medName, quantity, price, stock);
                    med.Add(m);
                }
            }
        }

        Customer c = new Customer(name, contact, address, bill, med);
        cust_list.Add(c);
    }
  }
 }

}

using pharmacay_managemnt_system.BL;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace pharmacay_managemnt_system.DL
{
   internal class MedicineCRUD
   {
      private static List<Medicine> med_list = new List<Medicine>();
      public static void storeInListMed(Medicine ad)
      {
         med_list.Add(ad);
      }

      public static List<Medicine> getMedicineList()
      {
         return med_list;
      }
      public static Medicine findMedicineObj(Medicine obj)
      {
         for (int i = 0; i < med_list.Count(); i++)
         {
            if (med_list[i].GetMedicineName() == obj.GetMedicineName() && med_list[i].GetQuantity() ==
obj.GetQuantity() && med_list[i].GetPrice() == obj.GetPrice() && med_list[i].GetStock() == obj.GetStock())
            {
               Medicine med = med_list[i];
               return med;
            }
         }
         return null;
      }

      public static void UpdatingMedicine(Medicine obj , int i)
```

```
    {
      med_list[i].SetPrice(obj.GetPrice());
      med_list[i].SetStock(obj.GetStock());
      med_list[i].SetMedicineName(obj.GetMedicineName());
      med_list[i].SetQuantity(obj.GetQuantity());
    }

    public static void removeMedicine(int i)
    {
       med_list.RemoveAt(i);
    }

    public static Medicine ProvideMedObj(string medicineName, int quantity)
    {
       for (int i = 0; i < med_list.Count; i++)
       {
          if (med_list[i].GetMedicineName() == medicineName && med_list[i].GetQuantity() == quantity)
          {
             return med_list[i];
          }
       }
       return null;
    }
    public static string MedIndex(int i)
    {
       return med_list[i].GetMedicineName();
    }
    public static string QuantityIndex(int i)
    {
```

```
        return med_list[i].GetQuantity().ToString();
    }
    public static string priceIndex(int i)
    {
       return med_list[i].GetPrice().ToString();
    }
    public static string stockIndex(int i)
    {
       return med_list[i].GetStock().ToString();
    }

    public static int getListSize()
    {
       return med_list.Count();
    }
    public static void stockAutoUpdate(List<Medicine> list)
    {
       for (int i = 0; i < med_list.Count; i++)
       {
          for(int j = 0; j < list.Count; j++ )
          if (med_list[i].GetMedicineName() == list[j].GetMedicineName() && med_list[i].GetQuantity() ==
list[j].GetQuantity())
          {
             int x = med_list[i].GetStock();
             int v = list[j].GetStock();
             x = x - v;
          }
       }
    }
    public static int totalPricePerProduct(int price, int stock)
```

```
    {
        price = price * stock;
        return price;
    }
    public static bool ValidateInput(string name, string quantity, string price, string stock, out int validatedQuantity, out int validatedPrice, out int validatedStock)
    {
        if (string.IsNullOrWhiteSpace(name))
        {
            MessageBox.Show("Please enter a valid name.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            validatedQuantity = validatedPrice = validatedStock = 0;
            return false;
        }

        if (!int.TryParse(quantity, out validatedQuantity) || validatedQuantity <= 0)
        {
            MessageBox.Show("Please enter a valid quantity.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            validatedPrice = validatedStock = 0;
            return false;
        }

        if (!int.TryParse(price, out validatedPrice) || validatedPrice <= 0)
        {
            MessageBox.Show("Please enter a valid price.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            validatedStock = 0;
            return false;
        }

        if (!int.TryParse(stock, out validatedStock) || validatedStock < 0)
```

```
        {
            MessageBox.Show("Please enter a valid stock value.", "Input Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
        }

        return true;
    }
    public static void storeDataInFile(string path)
    {
        StreamWriter file = new StreamWriter(path, true);
        for (int i = 0; i < med_list.Count(); i++)
        {
            file.WriteLine(med_list[i].GetMedicineName() + ";" + med_list[i].GetQuantity() + ";" + med_list[i].GetPrice() +
";" + med_list[i].GetStock());
        }
        file.Flush();
        file.Close();
    }

    public static void RetriveDataFromFile(string path)
    {
        StreamReader filevariable = new StreamReader(path);
        string record;
        while ((record = filevariable.ReadLine()) != null)
        {
            string[] objString = record.Split(';');
            string name = objString[0];
            int quantity = int.Parse(objString[1]);
            int price = int.Parse(objString[2]);
```

```
            int stock = int.Parse(objString[3]);
            Medicine med = new Medicine(name, quantity, price, stock);
            storeInListMed(med);
        }
        filevariable.Close();
    }
  }
}
using pharmacay_managemnt_system.BL;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system.DL
{
  internal class SupplierCRUD
  {
    private static List<Supplier> sup_list = new List<Supplier>();

    public static void storeInListSup(Supplier ad)
    {
      sup_list.Add(ad);
    }
    public static void removeSupplier(int i)
    {
```

```
        sup_list.RemoveAt(i);


    }
    public static void UpdatingSuppler(Supplier obj, int i)
    {
        sup_list[i].SetSupplierName(obj.GetSupplierName());
        sup_list[i].GetMedObj().SetMedicineName(obj.GetMedObj().GetMedicineName());
        sup_list[i].GetMedObj().SetQuantity(obj.GetMedObj().GetQuantity());
        sup_list[i].GetMedObj().SetPrice(obj.GetMedObj().GetPrice());
        sup_list[i].GetMedObj().SetStock(obj.GetMedObj().GetStock());
    }
    public static string displayStock(int i)
    {
        if (i >= 0 && i < sup_list.Count)
        {
            return (sup_list[i].GetSupplierName() + "\t\t\t" + sup_list[i].GetMedObj().GetMedicineName() + "\t\t\t" +
sup_list[i].GetMedObj().GetQuantity() + "\t\t " + sup_list[i].GetMedObj().GetPrice().ToString() + "\t\t " +
sup_list[i].GetMedObj().GetStock().ToString());
        }
        return null;
    }
    public static int getListSize()
    {
        return sup_list.Count();
    }
    public static bool validation(string name)
    {
        if (String.IsNullOrEmpty(name))
        {
```

```
            MessageBox.Show("Supplier section cannot be empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
          }
          return true;
      }




      public static string supplierIndex(int i)
      {
          return sup_list[i].GetSupplierName();
      }
      public static string MedIndex(int i)
      {
          return sup_list[i].GetMedObj().GetMedicineName();
      }
      public static string QuantityIndex(int i)
      {
          return sup_list[i].GetMedObj().GetQuantity().ToString();
      }
      public static string priceIndex(int i)
      {
          return sup_list[i].GetMedObj().GetPrice().ToString();
      }
      public static string stockIndex(int i)
      {
          return sup_list[i].GetMedObj().GetStock().ToString();
      }
```

```
public static void storeDataInFile(string path)
{
   StreamWriter file = new StreamWriter(path, true);
   for (int i = 0; i < sup_list.Count(); i++)
   {
      file.WriteLine(sup_list[i].GetSupplierName() + "," + sup_list[i].GetMedObj().GetMedicineName().ToString() + ";"
+ sup_list[i].GetMedObj().GetQuantity().ToString() + ";" + sup_list[i].GetMedObj().GetPrice().ToString() + ";" +
sup_list[i].GetMedObj().GetStock().ToString());
   }
   file.Flush();
   file.Close();
}
public static void RetriveDataFromFile(string path)
{
   StreamReader filevariable = new StreamReader(path);
   string record;
   while ((record = filevariable.ReadLine()) != null)
   {
      string[] splittedRecord = record.Split(',');
      string supplierName = splittedRecord[0];
      string[] objString = splittedRecord[1].Split(';');
      string name = objString[0];
      int quantity =  int.Parse(objString[1]);
      int price = int.Parse(objString[2]);
      int stock = int.Parse(objString[3]);
      Medicine med = new Medicine(name, quantity, price, stock);
```

```
            Supplier ad = new Supplier(supplierName, med);
            storeInListSup(ad);
        }
        filevariable.Close();
    }
  }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using pharmacay_managemnt_system.DL;
using pharmacay_managemnt_system.WorkerUI;

namespace pharmacay_managemnt_system
{
  public partial class Form1 : Form
  {
    string mPath = "medicineData.txt";
    string sPath = "supplierData.txt";
    string cPath = "customerData.txt";
    string pPath = "personData.txt";
```

```
    private static bool checkDataisLoaded = false;
    public Form1()
    {
       InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
       if(!checkDataisLoaded)
       {
          MedicineCRUD.RetriveDataFromFile(mPath);
          SupplierCRUD.RetriveDataFromFile(sPath);
          PersonCRUD.RetriveDataFromFile(pPath);
          CustomerCRUD.RetriveDataFromFile(cPath);
          checkDataisLoaded = true;
       }
    }

    private void SignInBtn_Click_1(object sender, EventArgs e)
    {
       SignInMenu fo = new SignInMenu();
       fo.Show();
       this.Hide();
    }
    private void SignUpBtn_Click_1(object sender, EventArgs e)
    {
       ProtectedSignUp a = new ProtectedSignUp();
       a.Show();
       this.Hide();
```

```
        }

        private void StoreDataBtn_Click(object sender, EventArgs e)
        {
            MedicineCRUD.storeDataInFile(mPath);
            SupplierCRUD.storeDataInFile(sPath);
            CustomerCRUD.storeDataInFile(cPath);
            PersonCRUD.storeDataInFile(pPath);
            StoreDataBtn.BackColor = Color.Green;
            MessageBox.Show("Data Stored", "Success", MessageBoxButtons.OK);
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace pharmacay_managemnt_system
{
   public partial class ProtectedSignUp : Form
   {
      public ProtectedSignUp()
      {
         InitializeComponent();
      }

      private void passwordBox_TextChanged(object sender, EventArgs e)
      {

      }

      private void pictureBox1_Click(object sender, EventArgs e)
      {
         Form1 form = new Form1();
         form.Show();
         this.Close();
      }

      private void Submit_Click(object sender, EventArgs e)
      {
         if (passwordBox.Text == "1234")
         {
            SignUPMenu signUPMenu = new SignUPMenu();
            signUPMenu.Show();
            this.Close();
         }
```

```
        else
        {
            MessageBox.Show("invalid Pin.", "Validity Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void ProtectedSignUp_Load(object sender, EventArgs e)
    {

    }
  }
}
using pharmacay_managemnt_system.BL;
using pharmacay_managemnt_system.DL;
using pharmacay_managemnt_system.WorkerUI;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system
{
  public partial class SignInMenu : Form
  {
```

```
public SignInMenu()
{
   InitializeComponent();
}

private void pictureBox1_Click(object sender, EventArgs e)
{
   Form1 f = new Form1();
   f.Show();
   this.Close();
   this.Hide();
}

private void Submit_Click(object sender, EventArgs e)
{
   string email = emailBox.Text.Trim();
   string password = passwordBox.Text.Trim();
   string roll = rollComboBox.Text;

   if (!(PersonCRUD.ValidateInput(email, password, roll)))
   {
     return;
   }

   Person p = new Admin(email, password, roll);
   string flag = PersonCRUD.signIn(p);
   if(flag == "admin")
   {
     ShowAdminDashboard();
```

```
        }
        else if (flag == "worker")
        {
          ShowWorkerDashboard();
        }
        else
        {
          MessageBox.Show("User Not Found.", "Recognition error. ", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
        }

    }
    private void ShowAdminDashboard()
    {
      AdminDashboard a = new AdminDashboard();
      a.Show();
      this.Hide();
    }
    private void ShowWorkerDashboard()
    {
      WorkerDashboard a = new WorkerDashboard();
      a.Show();
      this.Hide();
    }
    private void guna2Button1_Click(object sender, EventArgs e)
    {
      clearData();
    }
    private void clearData()
    {
```

```
            emailBox.Clear();
            passwordBox.Clear();
            rollComboBox.Items.Clear();
        }

        private void SignInMenu_Load(object sender, EventArgs e)
        {

        }
    }
}
using pharmacay_managemnt_system.BL;
using pharmacay_managemnt_system.DL;
using pharmacay_managemnt_system.WorkerUI;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system
{
    public partial class SignInMenu : Form
    {
        public SignInMenu()
```

```
    {
        InitializeComponent();
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        Form1 f = new Form1();
        f.Show();
        this.Close();
        this.Hide();
    }

    private void Submit_Click(object sender, EventArgs e)
    {
        string email = emailBox.Text.Trim();
        string password = passwordBox.Text.Trim();
        string roll = rollComboBox.Text;

        if (!(PersonCRUD.ValidateInput(email, password, roll)))
        {
          return;
        }

        Person p = new Admin(email, password, roll);
        string flag = PersonCRUD.signIn(p);
        if(flag == "admin")
        {
          ShowAdminDashboard();
        }
```

```
        else if (flag == "worker")
        {
          ShowWorkerDashboard();
        }
        else
        {
          MessageBox.Show("User Not Found.", "Recognition error. ", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
        }

    }
    private void ShowAdminDashboard()
    {
        AdminDashboard a = new AdminDashboard();
        a.Show();
        this.Hide();
    }
    private void ShowWorkerDashboard()
    {
        WorkerDashboard a = new WorkerDashboard();
        a.Show();
        this.Hide();
    }
    private void guna2Button1_Click(object sender, EventArgs e)
    {
        clearData();
    }
    private void clearData()
    {
        emailBox.Clear();
```

```
            passwordBox.Clear();
            rollComboBox.Items.Clear();
        }

        private void SignInMenu_Load(object sender, EventArgs e)
        {

        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Management;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Guna.UI2.WinForms;
using pharmacay_managemnt_system.BL;
using pharmacay_managemnt_system.DL;

namespace pharmacay_managemnt_system
{
    public partial class SignUPMenu : Form
    {
        public SignUPMenu()
```

```
    {
        InitializeComponent();
    }

    private void Submit_Click(object sender, EventArgs e)
    {
        string email = emailBox.Text.Trim();
        string password = passwordBox.Text.Trim();
        string roll = "admin";

        if (!(PersonCRUD.ValidateInput(email, password, roll)))
        {
            return;
        }
        Person p = new Admin(email, password, roll);
        PersonCRUD.storeInListPer(p);
        closeForm();
    }

    private void guna2Button1_Click(object sender, EventArgs e)
    {
        clearData();
    }

    private void clearData()
    {
        emailBox.Clear();
        passwordBox.Clear();
    }
```

```
        private void SignUPMenu_Load(object sender, EventArgs e)
        {

        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            closeForm();
        }

        private void closeForm()
        {
            Form1 form = new Form1();
            form.Show();
            this.Close();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system
{
   public partial class AdminDashboard : Form
   {
      public AdminDashboard()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         AddMedicine f = new AddMedicine();
         f.Show();
         this.Hide();
      }

      private void AdminDashboard_Load(object sender, EventArgs e)
      {

      }

      private void btnLogOut_Click(object sender, EventArgs e)
      {
         Form1 a = new Form1();
```

```
            a.Show();
            this.Hide();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            AddSupplier addSupplier = new AddSupplier();
            addSupplier.Show();
            this.Close();
        }



        private void button8_Click(object sender, EventArgs e)
        {
            AddWorker addWorker = new AddWorker();
            addWorker.Show();
            this.Close();
        }

    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace pharmacay_managemnt_system.WorkerUI
{
   public partial class WorkerDashboard : Form
   {
      public WorkerDashboard()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         TakeOrderUI f = new TakeOrderUI();
         f.Show();
         this.Close();
      }

      private void button5_Click(object sender, EventArgs e)
      {
         PreviewOrderUI f = new PreviewOrderUI();
         f.Show();
         this.Close();
      }


      private void btnLogOut_Click(object sender, EventArgs e)
      {
         Form1 f = new Form1();
         f.Show();
```

```
            this.Close();
        }


    }
}
using pharmacay_managemnt_system.DL;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system.WorkerUI
{
    public partial class StockDisplay : Form
    {
        DataTable table = new DataTable("table");
        public StockDisplay()
        {
            InitializeComponent();
            InitializeDataTable();
        }

        private void StockDisplay_Load(object sender, EventArgs e)
        {
```

```
        displayFromList();
    }
    private void InitializeDataTable()
    {
        table.Columns.Add("Medicine", typeof(string));
        table.Columns.Add("Quantity_mg", typeof(int));
        table.Columns.Add("Price", typeof(int));
        table.Columns.Add("Stock", typeof(int));

        dataGridView1.DataSource = table;
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        WorkerDashboard f = new WorkerDashboard();
        f.Show();
        this.Close();
    }
    public void displayFromList()
    {
        int x = MedicineCRUD.getListSize();
        for (int i = 0; i < x; i++)
        {
            displayInGrid(MedicineCRUD.MedIndex(i), int.Parse(MedicineCRUD.QuantityIndex(i)),
int.Parse(MedicineCRUD.priceIndex(i)), int.Parse(MedicineCRUD.stockIndex(i)));
        }
    }
    public void displayInGrid(string name, int quantity, int price, int stock)
    {
        table.Rows.Add(name, quantity, price, stock);
```

```
        }
    }
}
using pharmacay_managemnt_system.BL;
using pharmacay_managemnt_system.DL;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace pharmacay_managemnt_system.WorkerUI
{
    public partial class PreviewOrderUI : Form
    {

        DataTable table  = new DataTable();
        public PreviewOrderUI()
        {
            InitializeComponent();
            initializeTable();
        }

        private void PreviewOrderUI_Load(object sender, EventArgs e)
        {
```

```
    }

    public void checkForCustomer()
    {
        string name = CustomerBox.Text;
        string contact = ContactBox.Text;

        if (int.TryParse(contact, out int value))
        {
            Customer c = new Customer(name, contact);
            c = CustomerCRUD.getObj(c);
            if (c != null)
            {
                textBox1.Text = c.GetBill().ToString();
                List<Medicine> list = c.GetMedicineList();
                for (int i = 0; i < list.Count; i++)
                {
                    displayInGrid(list[i].GetMedicineName(), list[i].GetQuantity(), list[i].GetPrice(), list[i].GetStock());
                }
            }
            else
            {
                MessageBox.Show("Customer Not Found", "Invalid Input", MessageBoxButtons.OK);
            }
        }
    }
    public void displayInGrid(string name, int quantity, int price, int amount)
    {
```

```
        table.Rows.Add(name, quantity, price, amount);
    }
    public void initializeTable()
    {
        table.Columns.Add("Medicine", typeof(string));
        table.Columns.Add("Quantity(mg)", typeof(string));
        table.Columns.Add("Price", typeof(int));
        table.Columns.Add("Amount", typeof(int));

        dataGridView1.DataSource = table;
    }

    private void deleteBtn_Click(object sender, EventArgs e)
    {
        checkForCustomer();
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        WorkerDashboard f = new WorkerDashboard();
        f.Show();
        this.Close();
    }
```