**Restaurant Management System**

****

Session: 2022 – 2026

**Submitted by:**

Mohammad Salman 2022-CS-138

**Supervised by:**

Prof. Dr. Muhammad Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[Introduction 2](#_Toc141636700)

[Class Responsibility Collaboration Diagrams 3](#_Toc141636701)

[Object Oriented Programming 4](#_Toc141636702)

[Design Pattern Implementation 6](#_Toc141636703)

[Class Details 8](#_Toc141636704)

[Conclusion 9](#_Toc141636705)

[Wireframes 10](#_Toc141636706)

[MANAGER 11](#_Toc141636707)

[CASHIER 13](#_Toc141636708)

[CODE 15](#_Toc141636709)

# Introduction

* **Overview**

Restaurant Management System is a backend management application developed in C#, catering to two types of users: Manager and Cashier. It utilizes object-oriented programming concepts to organize and manage various tasks within a restaurant environment. The system provides efficient management and operational support, enabling seamless workflow and enhanced customer service.

* **Objectives**

The objectives of this system includes

1. **Efficient Backend Management**: The system aims to streamline backend operations by providing tools and functionalities to effectively manage products, cashiers, and sales data.
2. **Improved Customer Service**: By maintaining customer records, processing orders accurately, and offering support, the system aims to enhance customer satisfaction and provide personalized services.

* **Functionality**

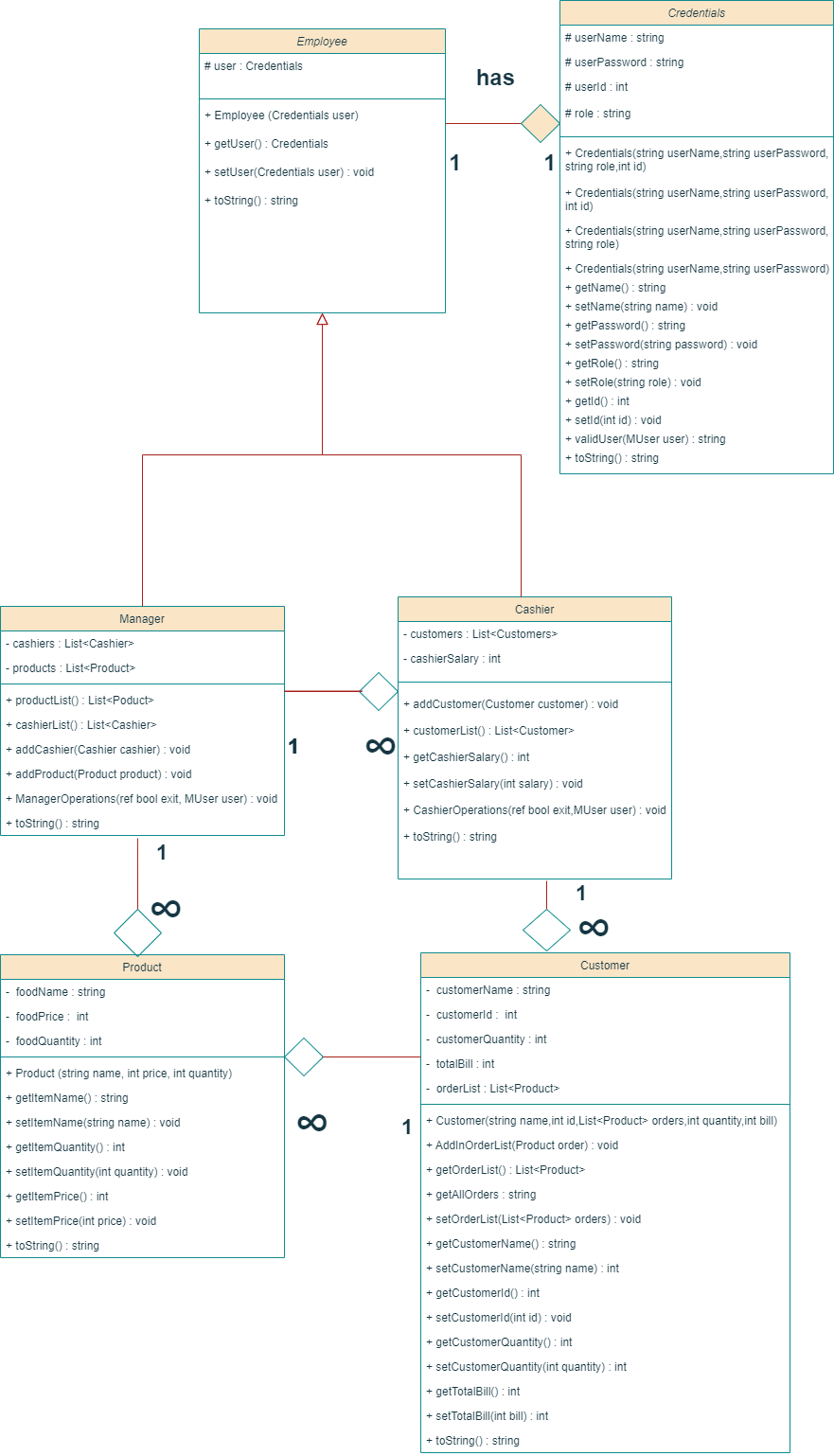
**Manager:**

1. **Product Management:** Add, edit, or remove food and beverage items, set prices, manage quantities, and update attributes.
2. **Employee Management:** Add, edit, or remove employees, assign roles, update personal information, and track performance.
3. **Inventory Management:** Track stock levels, monthly sale record, and reorder items as needed.

**Cashier:**

1. **Customer Management:** Create and manage customer accounts, record personal information, orders, and purchase history.
2. **Order Processing:** Take customer orders, add items, calculate total amount, and generate invoices or receipts.
3. **Payment Processing:** Accept payment, process transactions, and provide appropriate receipts.

# Class Responsibility Collaboration Diagrams

****

**­­­­**

# Object Oriented Programming

Object-oriented programming (OOP) is a programming paradigm based on the concepts of the “objects”, which contains data and code called as attributes and behavior of the class respectively. The main concepts of OOP includes association, inheritance and polymorphism. I have used this programming paradigm in my project.

* **Association**

In Object-oriented programming, Association is a relation between two separate classes which establishes through their objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many. There are two types of Association, **Aggregation** and **Composition**. I have used both Composition & Aggregation in my project. It is used in two places.

1. Employee class have Credential object. It is a relation of one-to-one because an employee class contains only one user object. Credentials cannot exist without Employee. So, I have used composition in this context.
2. Manager class have lists of Cashiers & Products. It is a relation of one-to-many because a manager class contains many cashiers and products. Both, Cashier & Product can exist without the object of Manager, so I used aggregation for this particular purpose.
3. Customer class contains a list of orders which is basically the list of products. This association also is one-to-many relation as a customer can contain multiple orders/products. As the object of Product can exist without Customer. So, the relation between them is aggregation.

**Advantage**

If I compare this with my procedural programming concepts, I can observe that there is a clear advantage of OOP. There was disjoint data of Product and Cashier in procedural programming which is rectified in OOP. Now the lists of Cashier and Product are within the classes of Manager and Customer respectively.

* **Inheritance**

Inheritance is one of the core concepts of Object-oriented programming approach. It is a feature that allows a new class to derive from an existing class. The new class inherits all the public or protected attributes and the member functions of the base class. I have used this OOP concept in one place in this management system.

1. Employee class is a parent class. Manager and Cashier class are derived from this class. Manager and Cashier are two users. So, that is the reason I have applied the concept of inheritance here as they are inheriting user object which contains name, password, role, and id form the MUser class.

**Advantage**

Inheritance gives various advantages over procedural programming. It promotes code-reusability and reduces redundancy. It helps in organizing the program’s structure. It allows flexibility in the code as you will adjust in one place and the rest of the code will work smoothly.

* **Polymorphism**

Polymorphism is also one of the core concepts of Object-oriented programming approach. This concept refers to the ability of a function to perform multiple operation under different circumstances. There are two types of Polymorphism. The type of polymorphism used to extend the functionality of common functions in parent and child classes is called Dynamic Polymorphism. I have used Dynamic Polymorphism in few places.

1. toString() functions in inheritance (Employee – Manager & Cashier).This function in parent class of Employee will return the name as chosen by the user. But when it is overridden in child classes, for instance in “Manager” class, it will return same name but in “Cashier”, it will also return id and salary.

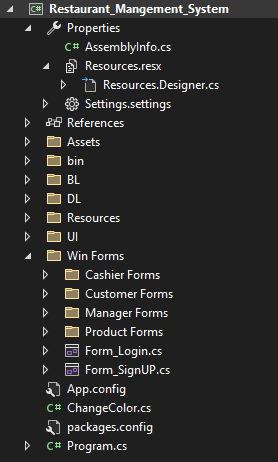
The other type of Polymorphism is Static Polymorphism. I have used this polymorphism only for the constructors.

**Advantage**

Polymorphism allowed us to extend the functionalities from the base class to use them for the child classes. Due to polymorphism, our code has become shorter because it didn’t required us to copy paste the whole code and then make changes to it. Dynamic Polymorphism has enabled the programmer to use the same function in different manner. We lacked this when we were making our projects in procedural programming.

# Design Pattern Implementation

The directory structure for the project is given below

****

* **Business Logic (BL)**

The Business Logic folder contains the main classes of the project. It includes the classes of Credentials, Employee, Manager, Cashier, Product, and Customer.

* **Data Layer (DL)**

The Data Layer folder contains the static Lists and functions of the project. It contains the list of products, list of users which contains the objects of Manager and Cashiers. There are static functions of each class as well such as storing and loading of data to and from files. It also includes the other static functions.

* **User Interface (UI)**

This folder contains all the code for printing and input of the data. This folder deals with the interaction of user with the application. It includes the menus and other functions which are used for printing and taking inputs from the user.

# Class Details

* **Employee**

An Employee class is used for generalizing the two classes of Manager and Cashier. It is a parent class of these two classes. It contains the common attributes of the both classes. It contains the user object of Credential class since every employee object created will have their own credentials.

* **Manager**

This class inherits the attributes and behavior of the parent class Employee. Its attributes includes the list of cashiers, products and function to add the objects of product and cashier to their respective list. All the attributes of the manager class are private and are accessed by the getter and setter functions.

* **Cashier**

Cashier class is a child class of Employee class. It inherits the attributes and behavior of the parent class. All the attributes of the Cashier class are also private. Its attributes include list of customers and function to add the object of customer to the defined list as cashier is able to take orders from customers. The cashier class has aggregation relation with manager class as the manager is able to add object of cashier to its list.

* **Credentials**

Credentials class contains strings of username, password and role of the user. Its attributes are also private. This class is created to facilitate in the sign-in and sign-up procedure. There are two constructors in this class. One constructor with three arguments are used when sign-up functionality is used. The other constructor with two arguments is used when we use sign-in function. There is another constructor which takes four arguments including id. So, that cashier can only login through his/her id once signing up.

* **Product**

Product class contains the attributes of food name, food price, and food quantity. This class has relation of aggregation with Manager as the manager class contains the list of products and is able to add product object to its list.

* **Customer**

The Customer class is a containing the attributes like customer name, customer id, bill, total quantity and list of orders which in fact is the list of products as the customer can order multiple food items. The customer class has many to one relation with cashier class and one to many relation with product class.

# Conclusion

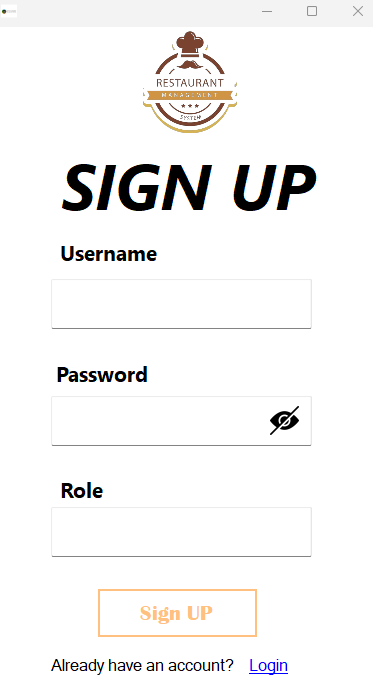
In conclusion, the Restaurant Management System is a robust software solution built using the principles of object-oriented programming. It incorporates key functionalities such as creating, retrieving, updating, and deleting data to effectively manage various aspects of a restaurant's operations. The system leverages important object-oriented concepts including association, inheritance, and polymorphism to create a modular and extensible design.

Throughout the development process, I encountered various challenges, particularly in designing a comprehensive class diagram and effectively implementing object-oriented principles. However, these challenges provided valuable learning opportunities, allowing me to gain a deeper understanding of how to create efficient and maintainable software systems using object-oriented theory.

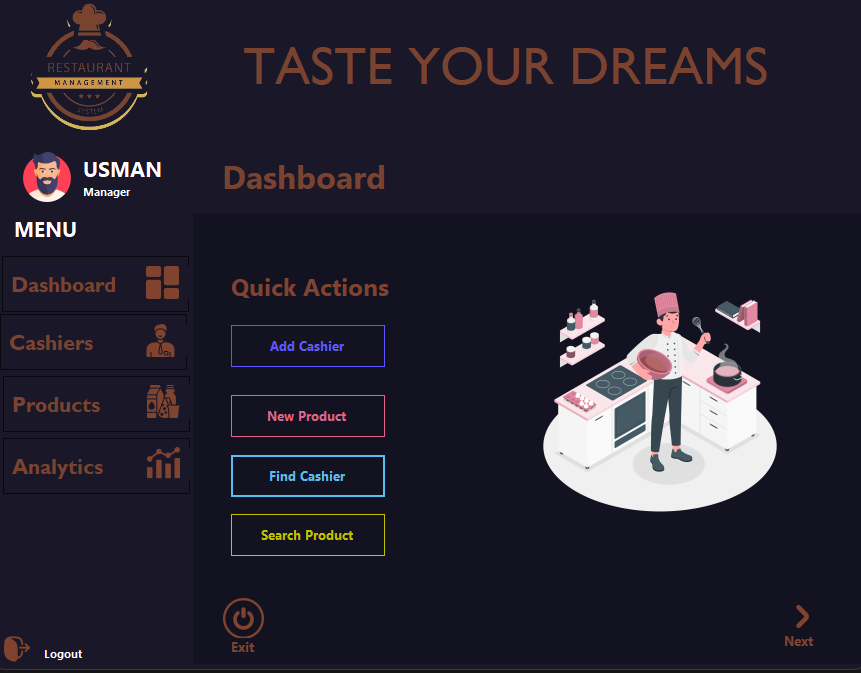
The adoption of the object-oriented approach brings numerous benefits to the Restaurant Management System. It enables scalability, making it easier to accommodate future enhancements and modifications. The modular nature of the system allows for easier maintenance, ensuring that updates and bug fixes can be implemented with minimal disruption.

Overall, the Restaurant Management System serves as a testament to the power and effectiveness of object-oriented programming in developing sophisticated software solutions. By leveraging the principles of object-oriented theory, the system offers a robust and flexible platform for efficiently managing restaurant operations.

# Wireframes

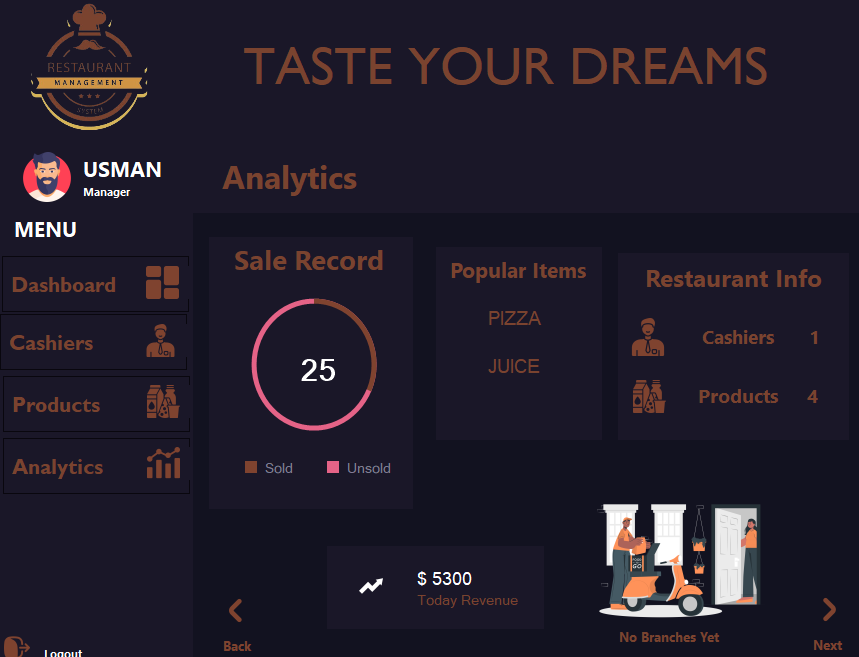
****

## MANAGER

****

****

****

****

## CASHIER

****

****

# CODE

1. **BL**

**Credentials**

using Restaurant\_Mangement\_System.DL;

namespace Restaurant\_Mangement\_System.BL

{

class Credentials

{

protected string userName = "";

protected string userPassword = "";

protected int userId = 0;

protected string role = "";

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

public string UserPassword { get => userPassword; set => userPassword = value; }

public int UserId { get => userId; set => userId = value; }

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

public Credentials()

{

}

public Credentials(string userName, int userId, string userPassword, string role)

{

this.UserName = userName;

this.UserId = userId;

this.UserPassword = userPassword;

this.Role = role;

}

public Credentials(string userName, string userPassword, string role)

{

this.UserName = userName;

this.UserPassword = userPassword;

this.Role = role;

}

public Credentials(string userName, int userId, string userPassword)

{

this.UserName = userName;

this.UserId = userId;

this.UserPassword = userPassword;

}

public Credentials(string userName, string userPassword)

{

this.UserName = userName;

this.UserPassword = userPassword;

}

public Credentials(int userId)

{

this.UserId = userId;

}

public static string validUser(Credentials existingUser)

{

string name = "";

foreach (Credentials user in CredentialsDL.usersList)

{

foreach (Employee cashier in Manager.Cashiers)

{

if (cashier.getUser().UserId == existingUser.UserId)

{

name = cashier.getUser().UserName;

}

}

if ((user.UserName == existingUser.UserName && user.UserPassword == existingUser.UserPassword) || user.UserName == name)

{

existingUser.UserName = user.UserName;

existingUser.UserPassword = user.UserPassword;

return user.Role;

}

}

return "User Not Found";

}

public virtual string toString()

{

return $"{UserName,-25}";

}

}

}

**Employee**

namespace Restaurant\_Mangement\_System.BL

{

class Employee

{

protected Credentials user = new Credentials();

public Employee() { }

public Employee(Credentials user)

{

this.user = user;

}

public Credentials getUser()

{

return this.user;

}

public void setUser(Credentials user)

{

this.user = user;

}

public virtual string toString()

{

return $"{user.UserName,-25}";

}

}

}

**Manager**

using System.Collections.Generic;

namespace Restaurant\_Mangement\_System.BL

{

class Manager : Employee

{

private static List<Cashier> cashiers = new List<Cashier>();

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

public static List<Cashier> Cashiers { get => cashiers; set => cashiers = value; }

public static List<Product> Products { get => products; set => products = value; }

public static void addCashier(Cashier emp)

{

if (emp != null)

{

Cashiers.Add(emp);

}

else

{

return;

}

}

public static void addProduct(Product product)

{

if (product != null)

{

Products.Add(product);

}

else

{

return;

}

}

public Manager()

{

}

public Manager(Credentials user) : base(user)

{

this.user = user;

}

public override string toString()

{

return base.toString();

}

}

}

**Cashier**

using System.Collections.Generic;

namespace Restaurant\_Mangement\_System.BL

{

class Cashier : Employee

{

private int cashierSalary = 0;

public Cashier()

{

}

public Cashier(int cashierSalary, Credentials user) : base(user)

{

this.user = user;

this.CashierSalary = cashierSalary;

}

private static List<Customer> customers = new List<Customer>();

public int CashierSalary { get => cashierSalary; set => cashierSalary = value; }

public static List<Customer> Customers { get => customers; set => customers = value; }

public static void addCustomer(Customer customer)

{

if (customer != null)

{

Customers.Add(customer);

}

else

{

return;

}

}

public override string toString()

{

return $"{base.toString()}{getUser().UserId,-20}{CashierSalary,-20}";

}

}

}

**Product**

namespace Restaurant\_Mangement\_System.BL

{

class Product

{

private string foodName = "";

private int foodPrice;

private int foodQuantity;

private int initialQuantity = 0;

public Product()

{

FoodName = "No Name";

FoodPrice = 0;

FoodQuantity = 0;

}

public Product(string name, int price, int quantity)

{

FoodName = name;

FoodPrice = price;

FoodQuantity = quantity;

InitialQuantity = quantity;

}

public string FoodName { get => foodName; set => foodName = value; }

public int FoodPrice { get => foodPrice; set => foodPrice = value; }

public int FoodQuantity { get => foodQuantity; set => foodQuantity = value; }

public int InitialQuantity { get => initialQuantity; set => initialQuantity = value; }

public static string getAllNames()

{

string names = "";

foreach (Product item in Manager.Products)

{

names += item.foodName + "\n";

}

return names;

}

public string toString()

{

return ($"{FoodName,-20}{FoodPrice,-15}" +

$"{FoodQuantity,-20}");

}

public static int totalSoldQuantity()

{

int total = 0;

foreach (Product item in Manager.Products)

{

total += item.InitialQuantity - item.FoodQuantity;

}

return total;

}

public static int totalQuantity()

{

int total = 0;

foreach (Product item in Manager.Products)

{

total += item.InitialQuantity;

}

return total;

}

}

}

1. **DL**

**Cashier**

using Restaurant\_Mangement\_System.BL;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class CashierDL

{

public static Cashier FindCashier(int id)

{

Cashier employee = Manager.Cashiers.FirstOrDefault(e => e.getUser().UserId == id);

return employee;

}

public static void FireCashier(int id)

{

Cashier employee = FindCashier(id);

if (employee != null)

{

Manager.Cashiers.Remove(employee);

}

}

public static void UpdateInfo(Cashier editedCashier)

{

int id = 0;

Cashier employee = FindCashier(id);

if (employee != null)

{

int index = Manager.Cashiers.IndexOf(employee);

Cashier updateCashier = editedCashier;

if (updateCashier != null)

{

string name = employee.getUser().UserName;

updateCashier.getUser().UserName = (name);

Manager.Cashiers.RemoveAt(index);

Manager.Cashiers.Insert(index, updateCashier);

}

}

else

{

return;

}

}

public static void storeCashier(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Cashier emp in Manager.Cashiers)

{

file.WriteLine(emp.getUser().UserName + "," + (emp.getUser().UserId) + "," + (emp.CashierSalary) + "," + emp.getUser().UserPassword);

}

file.Close();

}

public static void loadCashier(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

string name = userFields[0];

int id = int.Parse(userFields[1]);

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

string password = userFields[3];

Cashier employee = new Cashier(salary, new Credentials(name, id, password));

Manager.addCashier(employee);

}

file.Close();

}

else

{

MessageBox.Show("File Not Exists");

}

}

}

}

**CredentialsDL**

using Restaurant\_Mangement\_System.BL;

using System.Collections.Generic;

using System.IO;

namespace Restaurant\_Mangement\_System.DL

{

class CredentialsDL

{

public static List<Credentials> usersList = new List<Credentials>();

public static bool addUser(Credentials user, string path)

{

bool flag = false;

foreach (Credentials u in usersList)

{

if (user.Role.ToUpper() == "MANAGER")

{

if (u.UserPassword == user.UserPassword && u.UserName == user.UserName)

{

flag = false;

return flag;

}

else

{

flag = true;

}

}

}

if (user.Role.ToUpper() == "CASHIER")

{

foreach (Cashier employee in Manager.Cashiers)

{

if (employee.getUser().UserName == user.UserName && employee.getUser().UserPassword == user.UserPassword)

{

flag = true;

break;

}

else

{

flag = false;

}

}

}

if (flag == true)

{

usersList.Add(user);

storeUser(path, user);

return flag;

}

return flag;

}

public static void storeUser(string path, Credentials newUser)

{

StreamWriter writer = new StreamWriter(path);

foreach (Credentials user in usersList)

{

writer.WriteLine(user.UserName + "," + user.UserPassword + "," + user.Role);

}

writer.Close();

}

public static void loadUser(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

Credentials newUser = new Credentials();

newUser.UserName = (userFields[0]);

newUser.UserPassword = (userFields[1]);

newUser.Role = (userFields[2]);

usersList.Add(newUser);

}

file.Close();

}

}

}

}

**CustomerDL**

using Restaurant\_Mangement\_System.BL;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class CustomerDL

{

private static List<Customer> sortedCustomersList = new List<Customer>();

internal static List<Customer> SortedCustomersList { get => sortedCustomersList; set => sortedCustomersList = value; }

public static Customer FindCustomer(int id)

{

Customer customer = Cashier.Customers.FirstOrDefault(e => e.CustomerId == id);

return customer;

}

public static void RemoveCustomer(int id)

{

Customer customer = FindCustomer(id);

if (customer != null)

{

foreach (Product item in customer.OrdersList)

{

foreach (Product item1 in Manager.Products)

{

if (item.FoodName == item1.FoodName)

{

MessageBox.Show((item.FoodQuantity).ToString());

MessageBox.Show((item1.FoodQuantity).ToString());

int quantity = item.FoodQuantity + item1.FoodQuantity;

item1.FoodQuantity = quantity;

ProductsDL.storeStock(Program.path1);

}

}

}

MessageBox.Show("\n\tYOU HAVE REMOVED " + customer.CustomerName);

Cashier.Customers.Remove(customer);

}

else

{

MessageBox.Show("Customer Not Found");

}

}

public static void UpdateCustomerInfo(Product editedProduct, List<Product> orderList, int newQuantity, string text, int totalQuantity)

{

if (editedProduct != null)

{

if (newQuantity <= editedProduct.FoodQuantity)

{

Product exist = orderList.FirstOrDefault(f => f.FoodName == editedProduct.FoodName);

editedProduct.FoodQuantity = (editedProduct.FoodQuantity - newQuantity);

//refreshGrid();

totalQuantity += newQuantity;

if (exist != null)

{

exist.FoodName = editedProduct.FoodName;

exist.FoodPrice = newQuantity \* editedProduct.FoodPrice;

exist.FoodQuantity += newQuantity;

}

else

{

Product newProduct = new Product(editedProduct.FoodName, editedProduct.FoodPrice \* newQuantity, newQuantity);

orderList.Add(newProduct);

text = "Product Added Successfully";

}

}

else

{

text = editedProduct.FoodName + " not enough in stock";

}

}

else

{

text = "Product Does Not Exists";

}

}

public static List<Customer> sortedCustomers()

{

SortedCustomersList = Cashier.Customers.OrderBy(o => o.CustomerId).ToList();

return SortedCustomersList;

}

public static void LoadCustomers(string path)

{

StreamReader file = new StreamReader(path);

string line;

while ((line = file.ReadLine()) != null)

{

string[] usersField = line.Split(',');

string name = usersField[0];

int id = int.Parse(usersField[1]);

string[] orderDetails = usersField[2].Split(';');

List<Product> orders = new List<Product>();

foreach (string orderDetail in orderDetails)

{

string[] orderData = orderDetail.Split(':');

string itemName = orderData[0];

int itemQuantity = int.Parse(orderData[2]);

int itemPrice = int.Parse(orderData[1]);

Product product = new Product(itemName, itemPrice, itemQuantity);

orders.Add(product);

}

int quantity = int.Parse(usersField[3]);

int bill = int.Parse(usersField[4]);

Customer customer = new Customer(name, id, orders, quantity, bill);

Cashier.Customers.Add(customer);

}

file.Close();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STORE CUSTOMERS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void StoreCustomers(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Customer customer in Cashier.Customers)

{

string orderNames = "";

if (customer.OrdersList != null)

{

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

{

if (x == customer.OrdersList.Count - 1)

{

orderNames += customer.OrdersList[x].FoodName + ":" + customer.OrdersList[x].FoodQuantity + ":" + customer.OrdersList[x].FoodPrice;

}

else

{

orderNames += customer.OrdersList[x].FoodName + ":" + customer.OrdersList[x].FoodQuantity + ":" + customer.OrdersList[x].FoodPrice + ";";

}

}

file.Write(customer.CustomerName + ",");

file.Write(customer.CustomerId + ",");

file.Write(orderNames + ",");

file.Write(customer.CustomerQuantity + ",");

file.Write(customer.TotalBill + "\n");

}

}

file.AutoFlush = true;

file.Close();

}

}

}

**ProductsDL**

using Restaurant\_Mangement\_System.BL;

using Restaurant\_Mangement\_System.UI;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class ProductsDL

{

private static List<Product> sortedList = new List<Product>();

internal static List<Product> SortedList { get => sortedList; set => sortedList = value; }

public static Product SearchItem(string name)

{

Product product = Manager.Products.Find(e => e.FoodName == name);

return product;

}

public static void RemoveItem(string name)

{

Product product = SearchItem(name);

if (product != null)

{

MessageBox.Show($"\n\tYOU HAVE REMOVED {product.FoodName}");

Manager.Products.Remove(product);

}

else

{

MessageBox.Show("Product Not Found");

}

}

public static List<Product> sortProducts()

{

MessageBox.Show("1. PRICE LOW TO HIGH");

MessageBox.Show("2. PRICE HIGH TO LOW");

Console.Write(" YOUR OPTION: ");

int option = MiscUI.ValidateInteger();

if (option == 1)

{

SortedList = Manager.Products.OrderBy(o => o.FoodPrice).ToList();

}

else if (option == 2)

{

SortedList = Manager.Products.OrderByDescending(o => o.FoodPrice).ToList();

}

else

{

SortedList = Manager.Products;

}

return SortedList;

}

public static void UpdateStock(string name, int price, int quanity)

{

foreach (Product item in Manager.Products)

{

if (name == item.FoodName)

{

item.FoodPrice = price;

item.FoodQuantity = quanity;

item.InitialQuantity = quanity;

storeStock(Program.path1);

break;

}

}

}

public static void loadStock(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

string name = userFields[0];

int price = int.Parse(userFields[1]);

int quantity = int.Parse(userFields[2]);

Product product = new Product(name, price, quantity);

product.InitialQuantity = int.Parse(userFields[3]);

Manager.Products.Add(product);

}

file.Close();

}

else

{

MessageBox.Show("Path Not Found");

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STORE STOCK \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static void storeStock(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Product product in Manager.Products)

{

file.WriteLine(product.FoodName + "," + product.FoodPrice + "," + product.FoodQuantity + "," + product.InitialQuantity);

}

file.Flush();

file.Close();

}

}

}

1. **DL**

**Cashier**

using Restaurant\_Mangement\_System.BL;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class CashierDL

{

public static Cashier FindCashier(int id)

{

Cashier employee = Manager.Cashiers.FirstOrDefault(e => e.getUser().UserId == id);

return employee;

}

public static void FireCashier(int id)

{

Cashier employee = FindCashier(id);

if (employee != null)

{

Manager.Cashiers.Remove(employee);

}

}

public static void UpdateInfo(Cashier editedCashier)

{

int id = 0;

Cashier employee = FindCashier(id);

if (employee != null)

{

int index = Manager.Cashiers.IndexOf(employee);

Cashier updateCashier = editedCashier;

if (updateCashier != null)

{

string name = employee.getUser().UserName;

updateCashier.getUser().UserName = (name);

Manager.Cashiers.RemoveAt(index);

Manager.Cashiers.Insert(index, updateCashier);

}

}

else

{

return;

}

}

public static void storeCashier(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Cashier emp in Manager.Cashiers)

{

file.WriteLine(emp.getUser().UserName + "," + (emp.getUser().UserId) + "," + (emp.CashierSalary) + "," + emp.getUser().UserPassword);

}

file.Close();

}

public static void loadCashier(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

string name = userFields[0];

int id = int.Parse(userFields[1]);

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

string password = userFields[3];

Cashier employee = new Cashier(salary, new Credentials(name, id, password));

Manager.addCashier(employee);

}

file.Close();

}

else

{

MessageBox.Show("File Not Exists");

}

}

}

}

**CredentialsDL**

using Restaurant\_Mangement\_System.BL;

using System.Collections.Generic;

using System.IO;

namespace Restaurant\_Mangement\_System.DL

{

class CredentialsDL

{

public static List<Credentials> usersList = new List<Credentials>();

public static bool addUser(Credentials user, string path)

{

bool flag = false;

foreach (Credentials u in usersList)

{

if (user.Role.ToUpper() == "MANAGER")

{

if (u.UserPassword == user.UserPassword && u.UserName == user.UserName)

{

flag = false;

return flag;

}

else

{

flag = true;

}

}

}

if (user.Role.ToUpper() == "CASHIER")

{

foreach (Cashier employee in Manager.Cashiers)

{

if (employee.getUser().UserName == user.UserName && employee.getUser().UserPassword == user.UserPassword)

{

flag = true;

break;

}

else

{

flag = false;

}

}

}

if (flag == true)

{

usersList.Add(user);

storeUser(path, user);

return flag;

}

return flag;

}

public static void storeUser(string path, Credentials newUser)

{

StreamWriter writer = new StreamWriter(path);

foreach (Credentials user in usersList)

{

writer.WriteLine(user.UserName + "," + user.UserPassword + "," + user.Role);

}

writer.Close();

}

public static void loadUser(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

Credentials newUser = new Credentials();

newUser.UserName = (userFields[0]);

newUser.UserPassword = (userFields[1]);

newUser.Role = (userFields[2]);

usersList.Add(newUser);

}

file.Close();

}

}

}

}

**CustomerDL**

using Restaurant\_Mangement\_System.BL;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class CustomerDL

{

private static List<Customer> sortedCustomersList = new List<Customer>();

internal static List<Customer> SortedCustomersList { get => sortedCustomersList; set => sortedCustomersList = value; }

public static Customer FindCustomer(int id)

{

Customer customer = Cashier.Customers.FirstOrDefault(e => e.CustomerId == id);

return customer;

}

public static void RemoveCustomer(int id)

{

Customer customer = FindCustomer(id);

if (customer != null)

{

foreach (Product item in customer.OrdersList)

{

foreach (Product item1 in Manager.Products)

{

if (item.FoodName == item1.FoodName)

{

MessageBox.Show((item.FoodQuantity).ToString());

MessageBox.Show((item1.FoodQuantity).ToString());

int quantity = item.FoodQuantity + item1.FoodQuantity;

item1.FoodQuantity = quantity;

ProductsDL.storeStock(Program.path1);

}

}

}

MessageBox.Show("\n\tYOU HAVE REMOVED " + customer.CustomerName);

Cashier.Customers.Remove(customer);

}

else

{

MessageBox.Show("Customer Not Found");

}

}

public static void UpdateCustomerInfo(Product editedProduct, List<Product> orderList, int newQuantity, string text, int totalQuantity)

{

if (editedProduct != null)

{

if (newQuantity <= editedProduct.FoodQuantity)

{

Product exist = orderList.FirstOrDefault(f => f.FoodName == editedProduct.FoodName);

editedProduct.FoodQuantity = (editedProduct.FoodQuantity - newQuantity);

//refreshGrid();

totalQuantity += newQuantity;

if (exist != null)

{

exist.FoodName = editedProduct.FoodName;

exist.FoodPrice = newQuantity \* editedProduct.FoodPrice;

exist.FoodQuantity += newQuantity;

}

else

{

Product newProduct = new Product(editedProduct.FoodName, editedProduct.FoodPrice \* newQuantity, newQuantity);

orderList.Add(newProduct);

text = "Product Added Successfully";

}

}

else

{

text = editedProduct.FoodName + " not enough in stock";

}

}

else

{

text = "Product Does Not Exists";

}

}

public static List<Customer> sortedCustomers()

{

SortedCustomersList = Cashier.Customers.OrderBy(o => o.CustomerId).ToList();

return SortedCustomersList;

}

public static void LoadCustomers(string path)

{

StreamReader file = new StreamReader(path);

string line;

while ((line = file.ReadLine()) != null)

{

string[] usersField = line.Split(',');

string name = usersField[0];

int id = int.Parse(usersField[1]);

string[] orderDetails = usersField[2].Split(';');

List<Product> orders = new List<Product>();

foreach (string orderDetail in orderDetails)

{

string[] orderData = orderDetail.Split(':');

string itemName = orderData[0];

int itemQuantity = int.Parse(orderData[2]);

int itemPrice = int.Parse(orderData[1]);

Product product = new Product(itemName, itemPrice, itemQuantity);

orders.Add(product);

}

int quantity = int.Parse(usersField[3]);

int bill = int.Parse(usersField[4]);

Customer customer = new Customer(name, id, orders, quantity, bill);

Cashier.Customers.Add(customer);

}

file.Close();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STORE CUSTOMERS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void StoreCustomers(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Customer customer in Cashier.Customers)

{

string orderNames = "";

if (customer.OrdersList != null)

{

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

{

if (x == customer.OrdersList.Count - 1)

{

orderNames += customer.OrdersList[x].FoodName + ":" + customer.OrdersList[x].FoodQuantity + ":" + customer.OrdersList[x].FoodPrice;

}

else

{

orderNames += customer.OrdersList[x].FoodName + ":" + customer.OrdersList[x].FoodQuantity + ":" + customer.OrdersList[x].FoodPrice + ";";

}

}

file.Write(customer.CustomerName + ",");

file.Write(customer.CustomerId + ",");

file.Write(orderNames + ",");

file.Write(customer.CustomerQuantity + ",");

file.Write(customer.TotalBill + "\n");

}

}

file.AutoFlush = true;

file.Close();

}

}

}

**ProductsDL**

using Restaurant\_Mangement\_System.BL;

using Restaurant\_Mangement\_System.UI;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.DL

{

class ProductsDL

{

private static List<Product> sortedList = new List<Product>();

internal static List<Product> SortedList { get => sortedList; set => sortedList = value; }

public static Product SearchItem(string name)

{

Product product = Manager.Products.Find(e => e.FoodName == name);

return product;

}

public static void RemoveItem(string name)

{

Product product = SearchItem(name);

if (product != null)

{

MessageBox.Show($"\n\tYOU HAVE REMOVED {product.FoodName}");

Manager.Products.Remove(product);

}

else

{

MessageBox.Show("Product Not Found");

}

}

public static List<Product> sortProducts()

{

MessageBox.Show("1. PRICE LOW TO HIGH");

MessageBox.Show("2. PRICE HIGH TO LOW");

Console.Write(" YOUR OPTION: ");

int option = MiscUI.ValidateInteger();

if (option == 1)

{

SortedList = Manager.Products.OrderBy(o => o.FoodPrice).ToList();

}

else if (option == 2)

{

SortedList = Manager.Products.OrderByDescending(o => o.FoodPrice).ToList();

}

else

{

SortedList = Manager.Products;

}

return SortedList;

}

public static void UpdateStock(string name, int price, int quanity)

{

foreach (Product item in Manager.Products)

{

if (name == item.FoodName)

{

item.FoodPrice = price;

item.FoodQuantity = quanity;

item.InitialQuantity = quanity;

storeStock(Program.path1);

break;

}

}

}

public static void loadStock(string path)

{

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while ((line = file.ReadLine()) != null)

{

string[] userFields = line.Split(',');

string name = userFields[0];

int price = int.Parse(userFields[1]);

int quantity = int.Parse(userFields[2]);

Product product = new Product(name, price, quantity);

product.InitialQuantity = int.Parse(userFields[3]);

Manager.Products.Add(product);

}

file.Close();

}

else

{

MessageBox.Show("Path Not Found");

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STORE STOCK \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static void storeStock(string path)

{

StreamWriter file = new StreamWriter(path);

foreach (Product product in Manager.Products)

{

file.WriteLine(product.FoodName + "," + product.FoodPrice + "," + product.FoodQuantity + "," + product.InitialQuantity);

}

file.Flush();

file.Close();

}

}

}

1. **Forms Code**

**Form\_ Manager**

using Restaurant\_Mangement\_System.Win\_Forms;

using System;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System

{

public partial class frm\_Manager : Form

{

frm\_Employees employees;

frm\_Products products;

frm\_Analytics analytics;

frm\_Dashboard dashbiard;

private static bool LoadImage = true;

public frm\_Manager(string name)

{

InitializeComponent();

lbl\_personName.Text = name;

lbl\_personName.Visible = true;

dashbiard = new frm\_Dashboard(this, employees, products) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

analytics = new frm\_Analytics(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

products = new frm\_Products(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

employees = new frm\_Employees(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

}

private void Form\_Manager\_Load(object sender, EventArgs e)

{

ShowDashboard();

}

private void btn\_Cashiers\_Click(object sender, EventArgs e)

{

ShowCashiersForm();

}

private void btn\_Product\_Click(object sender, EventArgs e)

{

ShowProductsForm();

}

private void btn\_Analytics\_Click(object sender, EventArgs e)

{

ShowAnalyticsForm();

}

private void btn\_Dashboard\_Click(object sender, EventArgs e)

{

ShowDashboard();

}

private void img\_Logout\_Click(object sender, EventArgs e)

{

this.Hide();

frm\_login login = new frm\_login();

login.FormClosed += (s, args) => this.Close();

login.Show();

}

public void ShowDashboard()

{

this.lbl\_pnlHeader.Text = "Dashboard";

this.pnl\_Info.Controls.Clear();

this.pnl\_Info.Controls.Add(dashbiard);

dashbiard.Show();

}

public void ShowCashiersForm()

{

this.lbl\_pnlHeader.Text = "Cashiers";

this.pnl\_Info.Controls.Clear();

this.pnl\_Info.Controls.Add(employees);

employees.Show();

}

public void ShowProductsForm()

{

this.lbl\_pnlHeader.Text = "Products";

this.pnl\_Info.Controls.Clear();

this.pnl\_Info.Controls.Add(products);

products.Show();

}

public void ShowAnalyticsForm()

{

this.lbl\_pnlHeader.Text = "Analytics";

this.pnl\_Info.Controls.Clear();

this.pnl\_Info.Controls.Add(analytics);

analytics.Show();

}

}

}

**Form\_Cashier**

using System;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.Win\_Forms.Cashier\_Forms

{

public partial class frm\_Cashier : Form

{

frm\_Customers customers;

frm\_CashierDashboard dashboard;

public frm\_Cashier(string name)

{

InitializeComponent();

lbl\_personName.Text = name;

lbl\_personName.Visible = true;

customers = new frm\_Customers(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

}

private void img\_Customers\_Click(object sender, EventArgs e)

{

}

private void btn\_Customers\_Click(object sender, EventArgs e)

{

ShowCustomerForm();

}

private void img\_Logout\_Click(object sender, EventArgs e)

{

this.Hide();

frm\_login login = new frm\_login();

login.FormClosed += (s, args) => this.Close();

login.Show();

}

private void btn\_Analytics\_Click(object sender, EventArgs e)

{

ShowAnalytics();

}

private void btn\_List\_Click(object sender, EventArgs e)

{

ShowMenu();

}

private void frm\_Cashier\_Load(object sender, EventArgs e)

{

ShowDashboard();

}

private void btn\_Dashboard\_Click(object sender, EventArgs e)

{

ShowDashboard();

}

public void ShowDashboard()

{

this.lbl\_pnlHeader.Text = "Dashboard";

this.pnl\_Info.Controls.Clear();

dashboard = new frm\_CashierDashboard(this, customers) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

this.pnl\_Info.Controls.Add(dashboard);

dashboard.Show();

}

public void ShowMenu()

{

this.lbl\_pnlHeader.Text = "Menu List";

this.pnl\_Info.Controls.Clear();

frm\_CashierMenu menu = new frm\_CashierMenu(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

this.pnl\_Info.Controls.Add(menu);

menu.Show();

}

public void ShowAnalytics()

{

this.lbl\_pnlHeader.Text = "Analytics";

this.pnl\_Info.Controls.Clear();

frm\_CashierAnalytics analytics = new frm\_CashierAnalytics(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

this.pnl\_Info.Controls.Add(analytics);

analytics.Show();

}

public void ShowCustomerForm()

{

this.lbl\_pnlHeader.Text = "Customers";

this.pnl\_Info.Controls.Clear();

this.pnl\_Info.Controls.Add(customers);

customers.Show();

}

}

}

**Form\_Products**

using Restaurant\_Mangement\_System.BL;

using Restaurant\_Mangement\_System.DL;

using Restaurant\_Mangement\_System.Win\_Forms.Product\_Forms;

using System;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System

{

public partial class frm\_Products : Form

{

private frm\_Manager managerForm;

public frm\_Products(frm\_Manager managerForm)

{

InitializeComponent();

this.managerForm = managerForm;

}

private void frm\_Products\_Load(object sender, EventArgs e)

{

refreshGrid();

}

public void refreshGrid()

{

grid\_Products.DataSource = null;

grid\_Products.DataSource = Manager.Products;

grid\_Products.Columns["InitialQuantity"].Visible = false;

Program.resizeGrid(grid\_Products);

grid\_Products.Refresh();

}

public DataGridView GetGrid()

{

return grid\_Products;

}

private void pB\_Add\_Click(object sender, EventArgs e)

{

frm\_ProductAdd add = new frm\_ProductAdd(grid\_Products);

add.Show();

}

private void pB\_Delete\_Click(object sender, EventArgs e)

{

Product product = grid\_Products.CurrentRow.DataBoundItem as Product;

Manager.Products.Remove(product);

refreshGrid();

ProductsDL.storeStock(Program.path1);

}

private void pB\_Edit\_Click(object sender, EventArgs e)

{

Product product = grid\_Products.CurrentRow.DataBoundItem as Product;

frm\_ProductEdit edit = new frm\_ProductEdit(product.FoodName, grid\_Products);

edit.Show();

}

private void pB\_Find\_Click(object sender, EventArgs e)

{

frm\_ProductSearch search = new frm\_ProductSearch();

search.Show();

}

private void pB\_Next\_Click(object sender, EventArgs e)

{

managerForm.ShowAnalyticsForm();

}

private void pB\_Back\_Click(object sender, EventArgs e)

{

managerForm.ShowCashiersForm();

}

}

}

**Form\_Customers**

using Restaurant\_Mangement\_System.BL;

using Restaurant\_Mangement\_System.DL;

using Restaurant\_Mangement\_System.Win\_Forms.Cashier\_Forms;

using Restaurant\_Mangement\_System.Win\_Forms.Customer\_Forms;

using System;

using System.Linq;

using System.Windows.Forms;

namespace Restaurant\_Mangement\_System.Win\_Forms

{

public partial class frm\_Customers : Form

{

private frm\_Cashier cashierForm;

public frm\_Customers(frm\_Cashier cashierForm)

{

InitializeComponent();

this.cashierForm = cashierForm;

}

private void Form\_Customers\_Load(object sender, EventArgs e)

{

refreshGrid();

}

private void grid\_Customers\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void btn\_Delete\_Click(object sender, EventArgs e)

{

int id = int.Parse(grid\_Customers.CurrentRow.Cells[1].Value?.ToString());

CustomerDL.RemoveCustomer(id);

CustomerDL.StoreCustomers(Program.path3);

refreshGrid();

}

public void refreshGrid()

{

grid\_Customers.DataSource = null;

grid\_Customers.DataSource = Cashier.Customers.Select(c => new

{

Name = c.CustomerName,

Id = c.CustomerId,

Order = c.getAllOrders(),

Quantity = c.CustomerQuantity,

Bill = c.TotalBill

}).ToList();

Program.resizeGrid(grid\_Customers);

grid\_Customers.Refresh();

}

private void btn\_Search\_Click(object sender, EventArgs e)

{

frm\_CustomerSearch search = new frm\_CustomerSearch();

search.Show();

}

private void btn\_Edit\_Click(object sender, EventArgs e)

{

int id = int.Parse(grid\_Customers.CurrentRow.Cells[1].Value?.ToString());

Customer customer = CustomerDL.FindCustomer(id);

if (customer != null)

{

frm\_CustomerEdit edit = new frm\_CustomerEdit(customer, this);

edit.Show();

}

}

private void btn\_Bill\_Click(object sender, EventArgs e)

{

}

public DataGridView GetCustomerGrid()

{

return grid\_Customers;

}

private void pB\_Add\_Click(object sender, EventArgs e)

{

frm\_CustomerAdd add = new frm\_CustomerAdd(this);

add.Show();

}

private void pB\_Edit\_Click(object sender, EventArgs e)

{

int id = int.Parse(grid\_Customers.CurrentRow.Cells[1].Value?.ToString());

Customer customer = CustomerDL.FindCustomer(id);

if (customer != null)

{

frm\_CustomerEdit edit = new frm\_CustomerEdit(customer, this);

edit.Show();

}

}

private void pB\_Find\_Click(object sender, EventArgs e)

{

frm\_CustomerSearch search = new frm\_CustomerSearch();

search.Show();

}

private void pB\_Delete\_Click(object sender, EventArgs e)

{

int id = int.Parse(grid\_Customers.CurrentRow.Cells[1].Value?.ToString());

CustomerDL.RemoveCustomer(id);

CustomerDL.StoreCustomers(Program.path3);

refreshGrid();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

int id = int.Parse(grid\_Customers.CurrentRow.Cells[1].Value?.ToString());

Customer customer = CustomerDL.FindCustomer(id);

if (customer != null)

{

frm\_CustomerBill bill = new frm\_CustomerBill(customer);

bill.Show();

}

}

private void pB\_Next\_Click(object sender, EventArgs e)

{

cashierForm.ShowAnalytics();

}

private void pB\_Back\_Click(object sender, EventArgs e)

{

cashierForm.ShowMenu();

}

}

}