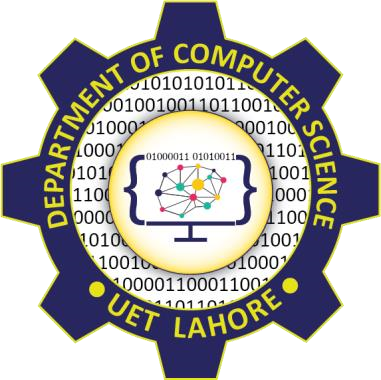
**Al Fajar Motors**



Session: 2022 – 2026

**Submitted by:**

Bisma Fajar 2022-CS-66

**Submitted To:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology** **Lahore Pakistan**

**Content**

[Youtube Video Link 6](#_Toc141559584)

[CRC 6](#_Toc141559585)

[BL Folder 7](#_Toc141559586)

[DL Folder 7](#_Toc141559587)

[UI Folder 8](#_Toc141559588)

[**Login Menu** 8](#_Toc141559589)

[**Admin Menu** 9](#_Toc141559590)

[**Customer Menu** 10](#_Toc141559591)

[Al- Fajar Motors 11](#_Toc141559592)

[Short Description of project 11](#_Toc141559593)

[Users of the application 11](#_Toc141559594)

[Functional Requirements 11](#_Toc141559595)

[OOP Concepts in the Project 14](#_Toc141559596)

[1. Association: 14](#_Toc141559597)

[2. Inheritance: 14](#_Toc141559598)

[3. Polymorphism: 14](#_Toc141559599)

[4. Encapsulation: 14](#_Toc141559600)

[Comparison with Traditional Procedural Programming 14](#_Toc141559601)

[1. Modularity and Reusability: 14](#_Toc141559602)

[2. Code Organization and Maintenance: 15](#_Toc141559603)

[3. Polymorphism and Flexibility: 15](#_Toc141559604)

[4. Data Abstraction and Security: 15](#_Toc141559605)

[Design Pattern Implementation: 15](#_Toc141559606)

[1. Business Logic (BL) Pattern: 15](#_Toc141559607)

[2. Data Access Layer (DL) Pattern: 15](#_Toc141559608)

[3. User Interface (UI) Pattern: 16](#_Toc141559609)

[Classes Details: 16](#_Toc141559610)

[1. `Person` (Base class): 16](#_Toc141559611)

[2. `Staff` (Derived from `Person`): 16](#_Toc141559612)

[3. `Customer` (Derived from `Person`): 16](#_Toc141559613)

[4. `Bus` (Base class): 17](#_Toc141559614)

[5. `Driver`(Derived from ‘Staff’): 17](#_Toc141559615)

[6. PersonDL: 17](#_Toc141559616)

[7.StaffDL: 17](#_Toc141559617)

[8.CustomerDL: 17](#_Toc141559618)

[9. BusDL: 17](#_Toc141559619)

[10. DriverDL: 17](#_Toc141559620)

[Complete Code: 17](#_Toc141559621)

[BL Folder 17](#_Toc141559622)

[**Person** 17](#_Toc141559623)

[**Driver** 19](#_Toc141559624)

[**Staff** 20](#_Toc141559625)

[**Customer** 21](#_Toc141559626)

[**Bus** 23](#_Toc141559627)

[DL Folder 24](#_Toc141559628)

[**PersonDL** 24](#_Toc141559629)

[**StaffDL** 26](#_Toc141559630)

[**DriverDL** 28](#_Toc141559631)

[**BusDL** 30](#_Toc141559632)

[UI Folder 32](#_Toc141559633)

[**Form1** 32](#_Toc141559634)

[**SignUp** 33](#_Toc141559635)

[**SignIn** 34](#_Toc141559636)

[**ForgotPassword** 35](#_Toc141559637)

[**AdminMenu** 35](#_Toc141559638)

[**AddBus** 38](#_Toc141559639)

[**AddStaff** 38](#_Toc141559640)

[**AddDriver** 39](#_Toc141559641)

[**RemoveBus** 40](#_Toc141559642)

[**RemoveStaff** 40](#_Toc141559643)

[**UpdateEmployee** 41](#_Toc141559644)

[**UpdateBus** 42](#_Toc141559645)

[**ViewDriver** 42](#_Toc141559646)

[**ViewStaff** 43](#_Toc141559647)

[**ViewBus** 44](#_Toc141559648)

[**CustomerMenu** 44](#_Toc141559649)

[**AddCustomerDetails** 46](#_Toc141559650)

[**CancelSeat** 49](#_Toc141559651)

[**UpdateSeat** 49](#_Toc141559652)

[Conclusion: 49](#_Toc141559653)

[Challenges: 50](#_Toc141559654)

[1. Designing a flexible and extensible architecture: 50](#_Toc141559655)

[2. Managing data persistence: 50](#_Toc141559656)

[3. User input validation: 50](#_Toc141559657)

[Achievements: 50](#_Toc141559658)

[1. Modularity and separation of concerns: 50](#_Toc141559659)

[2. Code reusability: 50](#_Toc141559660)

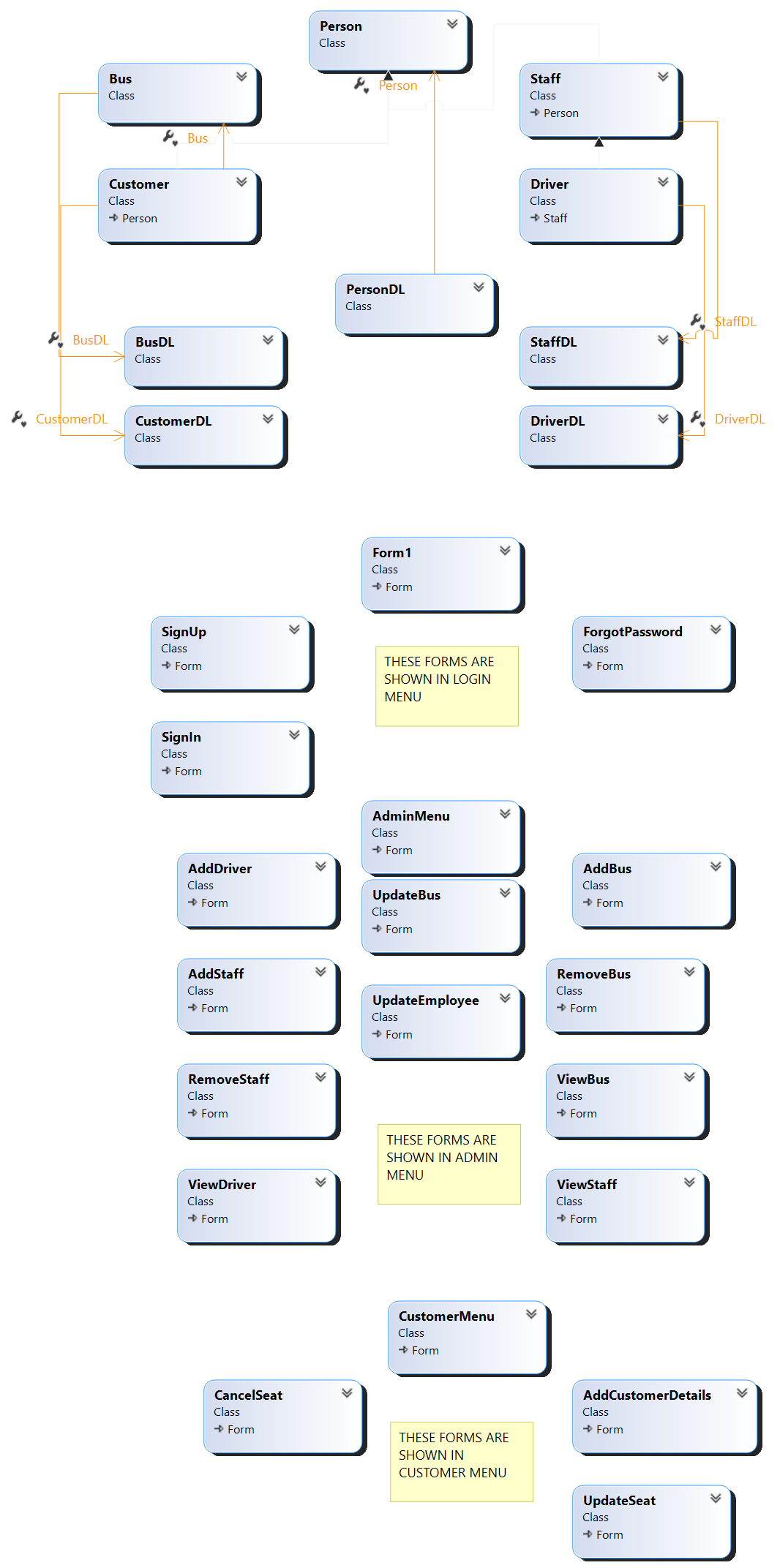
[3. Improved maintainability: 50](#_Toc141559661)

[Lesson Learned: 50](#_Toc141559662)

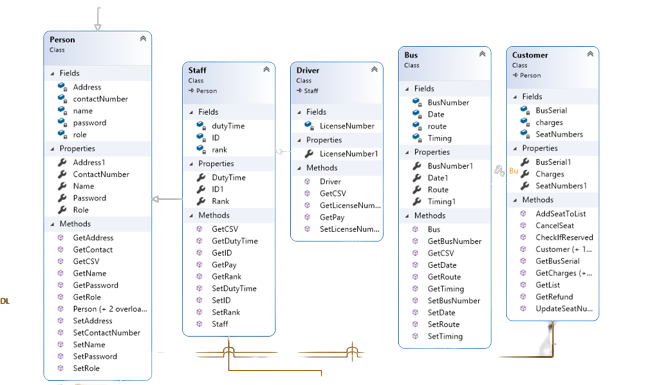
# Youtube Video Link

<https://youtu.be/N6l2E1dluE8>

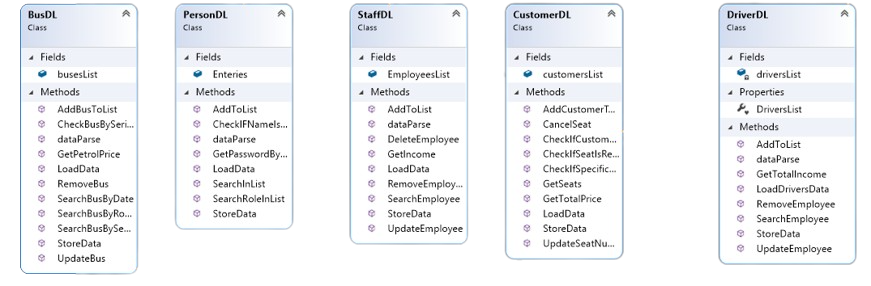
# CRC



## BL Folder

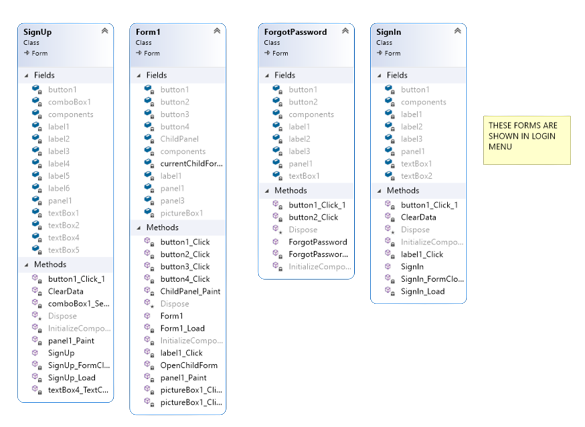


## DL Folder

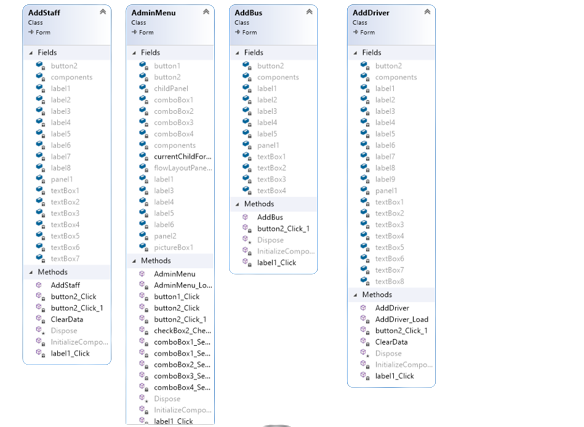
****

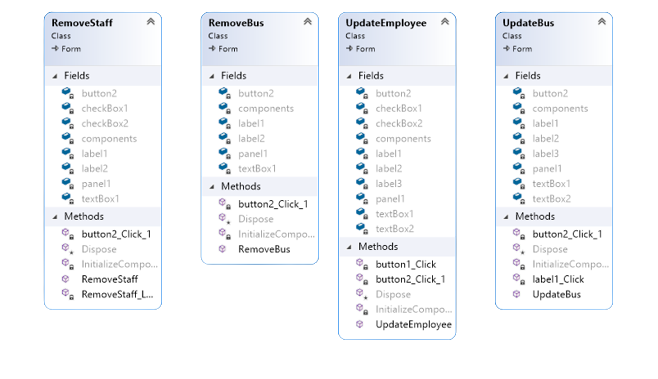
## UI Folder

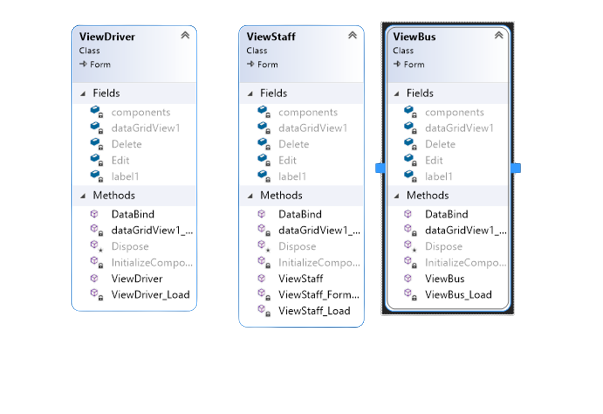
### **Login Menu**



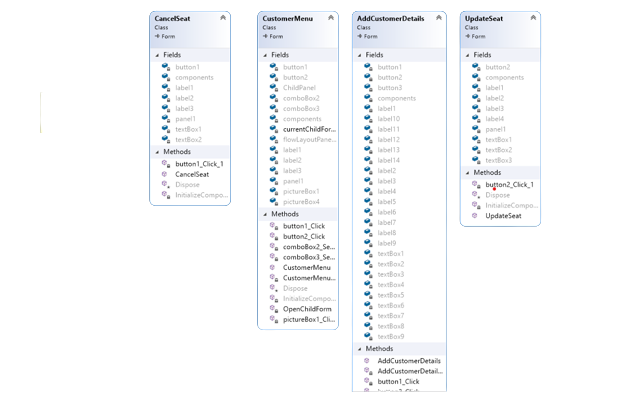
### **Admin Menu**







### **Customer Menu**



# Al- Fajar Motors

# Short Description of project

* + Bus reservation system will help a user to reserve seats online. This project covers two types of persons to access the application, first will be the admin and second will be the customer. At first a user will come and enter its user type if he is admin then he will be given access to money earned by selling tickets and loss if any, he can add, delete, update, search and view employees and buses. He will also be able to allot numbers to buses and he will set the timings of buses. And if the user type is customer then he can search the timings of buses, view all buses he will be able to reserve the seat in the bus, he can delete his ticket and can get refund according to policy of company, he can update ticket and view all tickets. This application will serve the computer user in a way that he/she has not to go bus station to check whether a bus is available or not and can reserve seats easily.

# Users of the application

There are two users of the application:

* + Admin
  + Customer

# Functional Requirements

***User***

***Story ID As a I want to perform So that I can***

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Admin | * Add employee. | First he/she will tell which type of employee he want to enter Driver or Other Staff . |
| 1 | Admin | * Remove employee. | He will check if the employee is present in the list then he will remove it from list and file. |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Admin | * Update employee. | He will check if the employee is present in the list then he will update it from list and file. |
| 1 | Admin | * Search employee. | He will check if the employee is present in the list. |
| 1 | Admin | * View all employees. | Can see the whole list. |
| 1 | Admin | * Add bus. | He can add bus. |
| 1 | Admin | * Remove bus. | He will check if the bus is present in the list then he will remove it from list and file. |
| 1 | Admin | * Update bus. | He will check if the bus is present in the list then he will update it from list and file. |
| 1 | Admin | * Search bus. | He will check if the bus is present in the list then he will return its object. |
| 1 | Admin | * View all buses. | Can see the whole list. |
| 1 | Admin | * View Income. | Can view profit and loss. |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Customer | * Search bus by route. | Customer can search the buses entered by admin. |
| 2 | Customer | * View all buses. | Can see the whole list of buses. |
| 2 | Customer | * Select bus. | Can select bus by entering serial number. |
| 2 | Customer | * Add personal details. | Add the personal details of passenger because sometimes the account name and passenger name are different. |
| 2 | Customer | * Reserve seat. | Enter seat number or numbers to reserve. |
| 2 | Customer | * View tickets. | Can view the reserved seats. |
| 2 | Customer | * Update ticket. | He will check if the seat is present in the list then he will update it. |
| 2 | Customer | * Pay for ticket. | Payment by cash or card. |
| 2 | Customer | * Cancel Seat. | Cancel the seat if he wants. |
| 2 | Customer | * Get refund. | Get Refund for cancelled seat. |

# OOP Concepts in the Project

The project incorporates the concepts of association, inheritance, and polymorphism to facilitate a well-structured and modular approach to software development. These concepts provide several advantages over traditional procedural programming, such as code reusability, modularity, flexibility, and enhanced code organization and maintenance.

## 1. Association:

The project demonstrates association by establishing relationships between classes and objects. For example, the `Customer` class is associated with the `Bus` class through the concept of a serial number of bus in which the customer is travelling.

## 2. Inheritance:

The project utilizes inheritance to create class hierarchies, where child classes inherit properties and behaviors from parent classes. The `Staff` and `Customer` classes inherit from the `Person` class, while ‘Driver’ class is associated with ‘Staff’ which provides common attributes and methods such as name and password. This inheritance relationship facilitates code reuse, encapsulation, and the organization of classes based on their similarities and differences.

## 3. Polymorphism:

Polymorphism is employed in the project through method overriding and method overloading. Method overriding allows child classes (`Admin` and `Customer`) to provide their own implementation of methods inherited from the parent class (`Person`). This enables the program to use the same method name while executing different behaviors based on the specific class instance.

For example there are functions for calculating pay and income in staff.

## 4. Encapsulation:

Encapsulation is employed through making attributes protected in parent classes whereas private in child classes. Methods are private in forms while static in DL folder.

# Comparison with Traditional Procedural Programming

In comparison to traditional procedural programming approaches, the adoption of object-oriented programming (OOP) brings several advantages:

## 1. Modularity and Reusability:

OOP promotes modular design by encapsulating data and behaviors within classes, facilitating code reuse. Classes can be easily reused in different parts of the program or in future projects. In procedural programming, code tends to be fragmented, making it challenging to reuse and maintain.

## 2. Code Organization and Maintenance:

OOP offers a structured and organized approach to code structure. Classes, objects, and their relationships mirror real-world entities and interactions, resulting in code that is easier to comprehend and maintain. In procedural programming, functions and data are often scattered throughout the code, making it more difficult to grasp the overall structure and flow.

## 3. Polymorphism and Flexibility:

Polymorphism in OOP allows for flexible and extensible code. Objects can be treated uniformly through their common interfaces or base classes, leveraging inheritance and polymorphic behavior. This simplifies code development, maintenance, and enhances scalability. Achieving similar flexibility and extensibility in procedural programming requires additional code and manual handling of different cases.

## 4. Data Abstraction and Security:

OOP enables data abstraction, where complex data structures and operations are hidden behind classes and interfaces. This abstraction enhances security by controlling data access and preventing unauthorized modifications. In procedural programming, data is often more exposed, increasing the risk of data manipulation or unauthorized access.

Overall, OOP provides a well-structured, modular, and flexible approach to software development compared to traditional procedural programming. It promotes code reuse, simplifies maintenance, and enhances scalability, making it the preferred choice for complex and large-scale projects.

# Design Pattern Implementation:

The project incorporates the Business Logic (BL), Data Access Layer (DL), and User Interface (UI) design patterns to ensure modularity and separation of concerns.

## 1. Business Logic (BL) Pattern:

The BL pattern is implemented to encapsulate the business rules and operations of the application. It separates the business logic from the UI and DL layers, promoting modularity and code reusability. The BL layer acts as an intermediary between the UI and DL layers, handling data validation, processing, and coordinating interactions between different components. By isolating the business logic, changes or updates to the UI or DL layers do not affect the core business rules, making the codebase more maintainable.

## 2. Data Access Layer (DL) Pattern:

The DL pattern is used to abstract the data storage and retrieval operations from the BL and UI layers. It provides a consistent interface for accessing and manipulating data, regardless of the underlying data storage technology or implementation details. The DL layer encapsulates database queries, file operations, or any other data access mechanisms. By separating data access logic from the BL layer, the codebase becomes more flexible and adaptable to changes in data sources or storage technologies. It also allows for better testability, as the BL layer can be tested independently without requiring a specific data source.

## 3. User Interface (UI) Pattern:

The UI pattern is employed to provide a structured and user-friendly interface for interacting with the application. It separates the presentation logic from the underlying business logic, enabling easier maintenance and updates to the UI without impacting the core functionality. The UI layer handles user input, displays relevant information, and interacts with the BL layer to perform actions and retrieve data. By decoupling the UI from the business logic, the application becomes more modular, allowing for different UI implementations or adaptations for various platforms or user experiences.

These design patterns ensure modularity and separation of concerns by dividing the application into distinct layers, each with its own specific responsibilities. The BL pattern isolates and manages the business logic, the DL pattern handles data access operations, and the UI pattern provides an intuitive interface for user interaction. This approach promotes code reusability, maintainability, and scalability, as changes or updates in one layer have minimal impact on the others. It also enhances testability, as each layer can be tested independently, leading to more robust and reliable software.

Overall, the implementation of these design patterns in the project ensures a well-structured and maintainable codebase, allowing for future enhancements and modifications with ease.

# Classes Details:

Here are the key classes in the project along with their responsibilities:

## 1. `Person` (Base class):

- Responsible for representing a person entity with common attributes like name, password, contact, address and role.

- Provides methods for setting and getting the person's name and password.

## 2. `Staff` (Derived from `Person`):

- Represents an admin user in the system.

- It has attributes of duty time, rank and Id.

- Inherits the properties and methods from the `Person` class and adds specific behavior for an admin user.

## 3. `Customer` (Derived from `Person`):

- Represents a customer user in the system.

- Inherits the properties and methods from the `Person` class and adds specific behavior for a customer user.

- Stores additional information such as the customer's address.

## 4. `Bus` (Base class):

- Represents the product it has attributes of bus serial, timing, route, date.

- Provides methods for setting and getting the product's attributes.

## 5. `Driver`(Derived from ‘Staff’):

- It has common attributes with staff and also license number of driver.

## 6. PersonDL:

Contains list of users of application and handles CURD operation on users.

## 7.StaffDL:

Contains method for loading and storing admin from file. Moreover, have function to load transaction history from file or store transactions in file.

## 8.CustomerDL:

Contains methods for loading and storing customers from file. Moreover, have function related to the cart of customers. To add or remove products from cart.

## 9. BusDL:

- Contains list of buses to perform CURD for admin and customer.

## 10. DriverDL:

- Contains list of buses to perform CURD for admin.

# Complete Code:

## BL Folder

### **Person**

class Person

{

private string name;

private string password;

private string role;

private string contactNumber;

private string Address;

public Person()

{

}

public string Name { get => name; set => name = value; }

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

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

public string ContactNumber { get => contactNumber; set => contactNumber = value; }

public string Address1 { get => Address; set => Address = value; }

public Person(string name, string password)

{

this.Name = name;

this.Password = password;

}

public Person(string name, string password, string role, string contactNumber, string Address)

{

SetName(name);

SetPassword(password);

SetRole(role);

SetAddress(Address);

SetContactNumber(contactNumber);

}

public void SetContactNumber(string contactNumber)

{

if (contactNumber != "")

{

this.ContactNumber = contactNumber;

}

}

public void SetAddress(string Address)

{

if (Address != "")

{

this.Address1 = Address;

}

}

public void SetName(string name)

{

if (name != "")

{

this.Name = name;

}

}

public void SetPassword(string password)

{

if (password != "")

{

this.Password = password;

}

}

public void SetRole(string role)

{

if (role == "admin" || role == "customer" || role == "driver" || role == "staff")

{

this.Role = role;

}

}

public string GetName()

{

return Name;

}

public string GetPassword()

{

return Password;

}

public string GetRole()

{

return Role;

}

public string GetContact()

{

return ContactNumber;

}

public string GetAddress()

{

return Address1;

}

public string GetCSV()

{

string line;

line = GetName() + "," + GetPassword() + "," + GetRole() + "," + GetContact() + "," + GetAddress();

return line;

}

}

### **Driver**

class Driver: Staff

{

private string LicenseNumber;

public string LicenseNumber1 { get => LicenseNumber; set => LicenseNumber = value; }

internal DL.DriverDL DriverDL

{

get => default;

set

{

}

}

public Driver(string name, string password, string role, string contactNumber, string Address, string ID, string rank, string dutyTime, string LicenseNumber) : base(name, password, role, contactNumber, Address, ID, rank, dutyTime)

{

this.LicenseNumber1 = LicenseNumber;

}

public string GetLicenseNumber()

{

return LicenseNumber1;

}

public void SetLicenseNumber(string LicenseNumber)

{

if (LicenseNumber != "")

{

this.LicenseNumber1 = LicenseNumber;

}

}

public override string GetCSV()

{

string line = GetName() + "," + GetPassword() + "," + GetRole() + "," + GetContact() + "," + GetAddress() + " , " + GetID() + " , " + GetRank() + " , " + GetDutyTime() + " , " + GetLicenseNumber();

return line;

}

public override double GetPay()

{

return 30000;

}

}

### **Staff**

class Staff: Person

{

private string rank;

private string dutyTime;

private string ID;

public string Rank { get => rank; set => rank = value; }

public string DutyTime { get => dutyTime; set => dutyTime = value; }

public string ID1 { get => ID; set => ID = value; }

internal DL.StaffDL StaffDL

{

get => default;

set

{

}

}

public Staff(string name, string password, string role, string contactNumber, string Address, string ID, string rank, string dutyTime) : base(name, password, role, contactNumber, Address)

{

this.Rank = rank;

this.DutyTime = dutyTime;

this.ID1 = ID;

}

public string GetRank()

{

return Rank;

}

public string GetDutyTime()

{

return DutyTime;

}

public string GetID()

{

return ID1;

}

public void SetRank(string rank)

{

if (rank != "")

{

this.Rank = rank;

}

}

public virtual double GetPay()

{

return 35000;

}

public void SetDutyTime(string dutyTime)

{

if (dutyTime == "night" || dutyTime == "day")

{

this.DutyTime = dutyTime;

}

}

public void SetID(string ID)

{

if (ID != "")

{

this.ID1 = ID;

}

}

new public virtual string GetCSV()

{

string line = GetName() + "," + GetPassword() + "," + GetRole() + "," + GetContact() + "," + GetAddress() + " , " + GetID() + " , " + GetRank() + " , " + GetDutyTime();

return line;

}

}

### **Customer**

class Customer: Person

{

private List<int> SeatNumbers = new List<int>();

private string BusSerial;

private double charges;

public Customer()

{

}

public List<int> SeatNumbers1 { get => SeatNumbers; set => SeatNumbers = value; }

public string BusSerial1 { get => BusSerial; set => BusSerial = value; }

public double Charges { get => charges; set => charges = value; }

public Customer(string name, string password, string role, string contactNumber, string Address,double charges, List<int> SeatNumbers, string reservedBus) : base(name, password, role, contactNumber, Address)

{

this.SeatNumbers1 = SeatNumbers;

this.BusSerial1 = reservedBus;

this.Charges = charges;

}

public void AddSeatToList(int seat)

{

SeatNumbers1.Add(seat);

}

public List<int> GetList()

{

List<int> temp = new List<int>();

temp = SeatNumbers1;

return temp;

}

public double GetCharges(string paymentType)

{

if (paymentType == "cash" || paymentType == "Cash")

{

Charges = SeatNumbers1.Count() \* 1500.0;

return Charges;

}

else if (paymentType == "card" || paymentType == "Card")

{

Charges = SeatNumbers1.Count() \* 1425.0;

return Charges;

}

return 0;

}

public double GetCharges()

{

return Charges;

}

public bool UpdateSeatNumber(int unChanged, int changed)

{

for (int x = 0; x < SeatNumbers1.Count(); x++)

{

if (SeatNumbers1[x] == unChanged)

{

SeatNumbers1[x] = changed;

return true;

}

}

return false;

}

public bool CheckIfReserved(List<int> seatNumber)

{

for (int x = 0; x < SeatNumbers1.Count(); x++)

{

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

{

if (SeatNumbers1[x] == seatNumber[y])

{

return false;

}

}

}

return true;

}

public bool CancelSeat(int seatNumber)

{

for (int x = 0; x < SeatNumbers1.Count(); x++)

{

if (SeatNumbers1[x] == seatNumber)

{

SeatNumbers1.RemoveAt(x);

return true;

}

}

return false;

}

public double GetRefund(bool flag)

{

if (flag == true)

{

Charges = Charges - 750;

return 750;

}

return 0;

}

public string GetBusSerial()

{

return BusSerial1;

}

internal Bus Bus

{

get => default;

set

{

}

}

internal DL.CustomerDL CustomerDL

{

get => default;

set

{

}

}

}

### **Bus**

class Bus

{

private string BusNumber;

private string Timing;

private string Date;

private string route;

public string BusNumber1 { get => BusNumber; set => BusNumber = value; }

public string Timing1 { get => Timing; set => Timing = value; }

public string Date1 { get => Date; set => Date = value; }

public string Route { get => route; set => route = value; }

internal DL.BusDL BusDL

{

get => default;

set

{

}

}

public Bus(string BusNumber, string Timing, string Date, string route)

{

SetBusNumber(BusNumber);

SetDate(Date);

SetRoute(route);

SetTiming(Timing);

}

public void SetBusNumber(string BusNumber)

{

if (BusNumber != "")

{

this.BusNumber1 = BusNumber;

}

}

public void SetTiming(string Timing)

{

if (Timing != "")

{

this.Timing1 = Timing;

}

}

public void SetDate(string Date)

{

if (Date != "")

{

this.Date1 = Date;

}

}

public void SetRoute(string route)

{

if (route != "")

{

this.Route = route;

}

}

public string GetBusNumber()

{

return BusNumber1;

}

public string GetDate()

{

return Date1;

}

public string GetTiming()

{

return Timing1;

}

public string GetRoute()

{

return Route;

}

public string GetCSV()

{

return GetBusNumber() + " , " + GetDate() + " , " + GetRoute() + " , " + GetTiming();

}

}

## DL Folder

### **PersonDL**

class PersonDL

{

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

internal Person Person

{

get => default;

set

{

}

}

public static bool CheckIFNameIsPresent(string name)

{

foreach (var x in Enteries)

{

if (x.GetName() == name)

{

return true;

}

}

return false;

}

public static bool LoadData(string path)

{

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

string line;

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

{

string name = dataParse(line, 1);

string password = dataParse(line, 2);

string Role = dataParse(line, 3);

string contact = dataParse(line, 4);

string address = dataParse(line, 5);

Person person = new Person(name, password, Role, contact, address);

PersonDL.AddToList(person);

}

file.Close();

return true;

}

else

{

Console.WriteLine("File does not exists");

}

return false;

}

public static string dataParse(string line, int field)

{

string item = "";

int commacount = 1;

for (int i = 0; i < line.Length; i++)

{

if (line[i] == ',')

{

commacount++;

}

else if (commacount == field)

{

item = item + line[i];

}

}

return item;

}

public static void StoreData(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (Person x in Enteries)

{

file.WriteLine(x.GetName() + "," + x.GetPassword() + "," + x.GetRole() + "," + x.GetContact() + "," + x.GetAddress());

/\* if(x is Driver) Type Casting

{

Driver driver = (Driver)x;

file.WriteLine(x.GetName() + "," + x.GetPassword() + "," + x.GetRole() + "," + x.GetContact() + "," + x.GetAddress() + "," + driver.GetID() + "," + driver.GetRank() + "," + driver.GetDutyTime() + "," + driver.GetLicenseNumber());

}\*/

}

file.Flush();

file.Close();

}

public static void AddToList(Person p)

{

if (p.GetName() != null && p.GetPassword() != null && p.GetRole() != null)

{

Enteries.Add(p);

}

}

public static string SearchRoleInList(string name, string password)

{

foreach (var x in Enteries)

{

if (x.GetName() == name && x.GetPassword() == password)

{

return x.GetRole();

}

}

return null;

}

public static string GetPasswordByName(string name)

{

foreach (var x in Enteries)

{

if (x.GetName() == name)

{

return x.GetPassword();

}

}

return null;

}

}

### **StaffDL**

class StaffDL

{

public static List<Staff> EmployeesList = new List<Staff>();

public static void AddToList(Staff staff)

{

EmployeesList.Add(staff);

}

public static Staff SearchEmployee(string name)

{

foreach (var x in EmployeesList)

{

if (x.GetName() == name)

{

return x;

}

}

return null;

}

public static void DeleteEmployee(Staff s)

{

EmployeesList.Remove(s);

}

public static void RemoveEmployee(string name, string rank)

{

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

{

if (EmployeesList[x].GetName() == name && EmployeesList[x].GetRank() == rank)

{

EmployeesList.RemoveAt(x);

}

}

}

public static void UpdateEmployee(string name, string updatedName, string rank, string ID)

{

foreach (var x in EmployeesList)

{

if (x.GetName() == name && x.GetRank() == rank && x.GetID() == ID)

{

x.SetName(updatedName);

}

}

}

public static void StoreData(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (var x in EmployeesList)

{

file.WriteLine(x.GetName() + "," + x.GetPassword() + "," + x.GetRole() + "," + x.GetContact() + "," + x.GetAddress() + "," + x.GetID() + "," + x.GetRank() + "," + x.GetDutyTime());

}

file.Flush();

file.Close();

}

public static bool LoadData(string path)

{

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

string line;

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

{

string name = dataParse(line, 1);

string password = dataParse(line, 2);

string Role = dataParse(line, 3);

string contact = dataParse(line, 4);

string address = dataParse(line, 5);

string ID = dataParse(line, 6);

string Rank = dataParse(line, 7);

string DutyTime = dataParse(line, 8);

Staff person = new Staff(name, password, Role, contact, address, ID, Rank, DutyTime);

StaffDL.AddToList(person);

}

file.Close();

return true;

}

else

{

Console.WriteLine("File does not exists");

}

return false;

}

public static string dataParse(string line, int field)

{

string item = "";

int commacount = 1;

for (int i = 0; i < line.Length; i++)

{

if (line[i] == ',')

{

commacount++;

}

else if (commacount == field)

{

item = item + line[i];

}

}

return item;

}

public static double GetIncome()

{

double payAble = DriverDL.GetTotalIncome() + BusDL.GetPetrolPrice();

double earned = CustomerDL.GetTotalPrice();

double Total = earned - payAble;

return Total;

}

}

### **DriverDL**

class DriverDL

{

private static List<Driver> driversList = new List<Driver>();

internal static List<Driver> DriversList { get => driversList; set => driversList = value; }

public static bool LoadDriversData(string path)

{

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

string line;

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

{

string name = dataParse(line, 1);

string password = dataParse(line, 2);

string Role = dataParse(line, 3);

string contact = dataParse(line, 4);

string address = dataParse(line, 5);

string ID = dataParse(line, 6);

string Rank = dataParse(line, 7);

string DutyTime = dataParse(line, 8);

string License = dataParse(line, 9);

Driver person = new Driver(name, password, Role, contact, address, ID, Rank, DutyTime, License);

DriverDL.AddToList(person);

}

file.Close();

return true;

}

else

{

Console.WriteLine("File does not exists");

}

return false;

}

public static string dataParse(string line, int field)

{

string item = "";

int commacount = 1;

for (int i = 0; i < line.Length; i++)

{

if (line[i] == ',')

{

commacount++;

}

else if (commacount == field)

{

item = item + line[i];

}

}

return item;

}

public static void StoreData(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (var x in DriversList)

{

file.WriteLine(x.GetName() + "," + x.GetPassword() + "," + x.GetRole() + "," + x.GetContact() + "," + x.GetAddress() + "," + x.GetID() + "," + x.GetRank() + "," + x.GetDutyTime() + "," + x.GetLicenseNumber());

}

file.Flush();

file.Close();

}

public static void AddToList(Driver p)

{

if (p.GetName() != "" && p.GetPassword() != "" && p.GetRole() != "")

{

DriversList.Add(p);

}

}

public static Driver SearchEmployee(string name)

{

foreach (var x in DriversList)

{

if (x.GetName() == name)

{

return x;

}

}

return null;

}

public static void RemoveEmployee(string name, string rank)

{

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

{

if (DriversList[x].GetName() == name && DriversList[x].GetRank() == rank)

{

DriversList.RemoveAt(x);

}

}

}

public static void UpdateEmployee(string name, string updatedName, string rank, string ID)

{

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

{

if (DriversList[x].GetName() == name && DriversList[x].GetRank() == rank && DriversList[x].GetID() == ID)

{

DriversList[x].SetName(updatedName);

}

}

}

public static double GetTotalIncome()

{

double charges = 0;

foreach (var x in DriversList)

{

charges = x.GetPay();

}

foreach (var x in StaffDL.EmployeesList)

{

charges = x.GetPay();

}

return charges;

}

}

### **BusDL**

class BusDL

{

public static List<Bus> busesList = new List<Bus>();

public static void AddBusToList(Bus bus)

{

busesList.Add(bus);

}

public static bool CheckBusBySerial(string serial)

{

foreach (var x in busesList)

{

if (x.GetBusNumber() == serial)

{

return true;

}

}

return false;

}

public static Bus SearchBusBySerial(string serial)

{

foreach (var x in busesList)

{

if (x.GetBusNumber() == serial)

{

return x;

}

}

return null;

}

public static Bus SearchBusByRoute(string route)

{

foreach (var x in busesList)

{

if (x.GetRoute() == route)

{

return x;

}

}

return null;

}

public static Bus SearchBusByDate(string date)

{

foreach (var x in busesList)

{

if (x.GetDate() == date)

{

return x;

}

}

return null;

}

public static bool RemoveBus(string BusNumber)

{

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

{

if (busesList[x].GetBusNumber() == BusNumber)

{

busesList.RemoveAt(x);

return true;

}

}

return false;

}

public static void UpdateBus(string busNumber, string route, string UpdatedRoute)

{

foreach (var x in busesList)

{

if (x.GetBusNumber() == busNumber && x.GetRoute() == route)

{

x.SetRoute(UpdatedRoute);

}

}

}

public static void StoreData(string path)

{

StreamWriter file = new StreamWriter(path, false);

foreach (var x in busesList)

{

file.WriteLine(x.GetBusNumber() + "," + x.GetTiming() + "," + x.GetDate() + "," + x.GetRoute());

}

file.Flush();

file.Close();

}

public static bool LoadData(string path)

{

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

string line;

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

{

string serial = dataParse(line, 1);

string timing = dataParse(line, 2);

string date = dataParse(line, 3);

string route = dataParse(line, 4);

Bus bus = new Bus(serial, timing, date, route);

AddBusToList(bus);

}

file.Close();

return true;

}

else

{

Console.WriteLine("File does not exists");

}

return false;

}

public static string dataParse(string line, int field)

{

string item = "";

int commacount = 1;

for (int i = 0; i < line.Length; i++)

{

if (line[i] == ',')

{

commacount++;

}

else if (commacount == field)

{

item = item + line[i];

}

}

return item;

}

public static double GetPetrolPrice()

{

return busesList.Count() \* 5000.0;

}

}

## UI Folder

### **Form1**

public partial class Form1 : Form

{

Form currentChildForm;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{ }

private void label1\_Click(object sender, EventArgs e)

{ }

private void OpenChildForm(Form childForm)

{

if(currentChildForm != null)

{

currentChildForm.Close();

}

currentChildForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Left;

ChildPanel.Controls.Add(childForm);

ChildPanel.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

}

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

{ }

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void pictureBox1\_Click\_1(object sender, EventArgs e)

{

}

private void ChildPanel\_Paint(object sender, PaintEventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

SignUp sign = new SignUp();

OpenChildForm(sign);

}

private void button3\_Click(object sender, EventArgs e)

{

SignIn signIn = new SignIn();

OpenChildForm(signIn);

}

private void button4\_Click(object sender, EventArgs e)

{

ForgotPassword password = new ForgotPassword();

OpenChildForm(password);

}

}

### **SignUp**

public partial class SignUp : Form

{

public SignUp()

{

InitializeComponent();

}

private void ClearData()

{

textBox1.Text = "";

textBox2.Text = "";

comboBox1.Text = "";

textBox4.Text = "";

textBox5.Text = "";

}

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

{ }

private void SignUp\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{ }

private void textBox4\_TextChanged(object sender, EventArgs e)

{ }

private void panel1\_Paint(object sender, PaintEventArgs e)

{ }

private void button1\_Click\_1(object sender, EventArgs e)

{

string name = textBox1.Text;

string password = textBox2.Text;

string role = comboBox1.Text;

role = role.ToLower();

string path = "Data.txt";

string contact = textBox4.Text;

string address = textBox5.Text;

Person person = new Person(name, password, role, contact, address);

if (PersonDL.CheckIFNameIsPresent(name))

{

MessageBox.Show("Already Present User");

this.Hide();

}

else

{

if (name != "")

{

PersonDL.AddToList(person);

MessageBox.Show("User added");

if(role == "admin")

{

AdminMenu menu = new AdminMenu();

this.Hide();

menu.Show();

}

else if(role == "customer")

{

CustomerMenu menu = new CustomerMenu();

this.Hide();

menu.Show();

}

PersonDL.StoreData(path);

ClearData();

this.Hide();

}

}

this.Hide();

}

}

### **SignIn**

public partial class SignIn : Form

{

public SignIn()

{

InitializeComponent();

}

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

{ }

private void label1\_Click(object sender, EventArgs e)

{ }

private void ClearData()

{

textBox1.Text = "";

textBox2.Text = "";

}

private void SignIn\_FormClosed(object sender, FormClosedEventArgs e)

{ }

private void button1\_Click\_1(object sender, EventArgs e)

{

string name = textBox1.Text;

string password = textBox2.Text;

string role = PersonDL.SearchRoleInList(name, password);

if (role != null)

{

role = role.ToLower();

if (PersonDL.CheckIFNameIsPresent(name))

{

MessageBox.Show("User Present as " + role);

ClearData();

if (role == "admin")

{

AdminMenu admin = new AdminMenu();

admin.Show();

}

else if (role == "customer")

{

CustomerMenu customer = new CustomerMenu();

customer.Show();

}

this.Hide();

}

}

else

{

MessageBox.Show("User Not Present");

this.Hide();

}

}

}

### **ForgotPassword**

public partial class ForgotPassword : Form

{

public ForgotPassword()

{

InitializeComponent();

}

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

{ }

private void button1\_Click\_1(object sender, EventArgs e)

{ }

private void button2\_Click(object sender, EventArgs e)

{

string name = textBox1.Text;

string password = PersonDL.GetPasswordByName(name);

label3.Text = password;

MessageBox.Show("You can copy your password and Sign In again");

}

}

### **AdminMenu**

public partial class AdminMenu : Form

{

Form currentChildForm;

public AdminMenu()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

SignIn signIn = new SignIn();

this.Hide();

signIn.Show();

}

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

{ }

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{ }

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

{ }

private void button2\_Click(object sender, EventArgs e)

{ }

private void OpenChildForm(Form childForm)

{

if (currentChildForm != null)

{

currentChildForm.Close();

}

currentChildForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Left;

childPanel.Controls.Add(childForm);

childPanel.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

private void comboBox1\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

string type = comboBox1.Text.ToLower();

comboBox1.ResetText();

if (type == "driver")

{

comboBox1.Text = " ";

AddDriver add = new AddDriver();

OpenChildForm(add);

}

else if(type == "employee")

{

comboBox1.Text = " ";

AddStaff add = new AddStaff();

OpenChildForm(add);

}

else

{

comboBox1.Text = " ";

AddBus add = new AddBus();

OpenChildForm(add);

}

comboBox1.Text = " ";

}

private void label3\_Click(object sender, EventArgs e)

{ }

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

string type = comboBox2.Text.ToLower();

if (type == "driver")

{

RemoveStaff staff = new RemoveStaff();

OpenChildForm(staff);

comboBox2.Text = " ";

}

else if (type == "employee")

{

RemoveStaff staff = new RemoveStaff();

OpenChildForm(staff);

comboBox2.Text = " ";

}

else

{

RemoveBus bus = new RemoveBus();

OpenChildForm(bus);

comboBox2.Text = " ";

}

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

string type = comboBox3.Text.ToLower();

if (type == "driver")

{

UpdateEmployee staff = new UpdateEmployee();

OpenChildForm(staff);

comboBox3.Text = " ";

}

else if (type == "employee")

{

UpdateEmployee staff = new UpdateEmployee();

OpenChildForm(staff);

comboBox3.Text = " ";

}

else

{

UpdateEmployee staff = new UpdateEmployee();

OpenChildForm(staff);

comboBox3.Text = " ";

}

}

private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)

{

string type = comboBox4.Text.ToLower();

if (type == "employee")

{

ViewStaff view = new ViewStaff();

OpenChildForm(view);

}

else if(type == "driver")

{

ViewDriver view = new ViewDriver();

OpenChildForm(view);

}

else if(type == "bus")

{

ViewBus bus = new ViewBus();

OpenChildForm(bus);

}

else

{

double money = StaffDL.GetIncome();

MessageBox.Show("Total income after subtracting staff pay and bus charges is : "+money.ToString());

}

}

private void label1\_Click(object sender, EventArgs e)

{ }

private void button2\_Click\_1(object sender, EventArgs e)

{

Form1 form = new Form1();

this.Hide();

form.Show();

}

private void label4\_Click(object sender, EventArgs e)

{ }

}

### **AddBus**

public partial class AddBus : Form

{

public AddBus()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{ }

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Bus.txt";

string serial = textBox1.Text;

string date = textBox2.Text;

string route = textBox3.Text;

string timing = textBox4.Text;

Bus bus = new Bus(serial, timing, date, route);

BusDL.AddBusToList(bus);

BusDL.StoreData(path);

MessageBox.Show("Bus Added");

this.Hide();

}

}

### **AddStaff**

public partial class AddStaff : Form

{

public AddStaff()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{ }

private void button2\_Click(object sender, EventArgs e)

{ }

private void ClearData()

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

}

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Staff.txt";

string name = textBox1.Text;

string password = textBox2.Text;

string role = "driver";

string contact = textBox3.Text;

string address = textBox4.Text;

string ID = textBox5.Text;

string rank = textBox6.Text;

string dutyTime = textBox7.Text;

Staff staff = new Staff(name, password, role, contact, address, ID, rank, dutyTime);

if (name != "")

{

StaffDL.AddToList(staff);

MessageBox.Show("Employee added");

StaffDL.StoreData(path);

ClearData();

this.Hide();

}

this.Hide();

}

}

}

### **AddDriver**

public partial class AddDriver : Form

{

public AddDriver()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{ }

private void ClearData()

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

}

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

{ }

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Driver.txt";

string name = textBox1.Text;

string password = textBox2.Text;

string role = "driver";

string contact = textBox3.Text;

string address = textBox4.Text;

string ID = textBox5.Text;

string rank = textBox6.Text;

string dutyTime = textBox7.Text;

string licenseNumber = textBox8.Text;

Driver driver = new Driver(name, password, role, contact, address, ID, rank, dutyTime, licenseNumber);

if (name != "")

{

DriverDL.AddToList(driver);

MessageBox.Show("Employee added");

DriverDL.StoreData(path);

ClearData();

this.Hide();

}

this.Hide();

}

}

### **RemoveBus**

public partial class RemoveBus : Form

{

public RemoveBus()

{

InitializeComponent();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

string serial = textBox1.Text;

string path = "Bus.txt";

bool flag = BusDL.RemoveBus(serial);

if (flag)

{

MessageBox.Show("Bus Removed");

}

else

{

MessageBox.Show("Bus Not present");

}

BusDL.StoreData(path);

this.Hide();

}

}

### **RemoveStaff**

public partial class RemoveStaff : Form

{

public RemoveStaff()

{

InitializeComponent();

}

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

{ }

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Driver.txt";

string path2 = "Staff.txt";

string name = textBox1.Text;

if (checkBox1.Checked)

{

Driver driver = DriverDL.SearchEmployee(name);

if (driver != null)

{

DriverDL.RemoveEmployee(driver.GetName(), driver.GetRank());

DriverDL.StoreData(path);

MessageBox.Show("Driver Removed");

}

else

{

MessageBox.Show("User Not Present");

}

}

else

{

Staff staff = StaffDL.SearchEmployee(name);

if (staff != null)

{

StaffDL.RemoveEmployee(staff.GetName(), staff.GetRank());

StaffDL.StoreData(path2);

MessageBox.Show("Employee Removed");

}

else

{

MessageBox.Show("User Not Present");

}

}

this.Hide();

}

}

### **UpdateEmployee**

public partial class UpdateEmployee : Form

{

public UpdateEmployee()

{

InitializeComponent();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Driver.txt";

string path2 = "Staff.txt";

string name = textBox1.Text;

string Updated = textBox2.Text;

if (checkBox1.Checked)

{

Driver driver = DriverDL.SearchEmployee(name);

if (driver != null)

{

DriverDL.UpdateEmployee(name, Updated, driver.GetRank(), driver.GetID());

DriverDL.StoreData(path);

MessageBox.Show("Driver Updated");

}

else

{

MessageBox.Show("User Not Present");

}

}

else

{

Staff staff = StaffDL.SearchEmployee(name);

if (staff != null)

{

StaffDL.UpdateEmployee(name, Updated, staff.GetRank(), staff.GetID());

StaffDL.StoreData(path2);

MessageBox.Show("Employee Updated");

}

else

{

MessageBox.Show("User Not Present");

}

}

this.Hide();

}

}

### **UpdateBus**

public partial class UpdateBus : Form

{

public UpdateBus()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{ }

private void button2\_Click\_1(object sender, EventArgs e)

{

string path = "Bus.txt";

string serial = textBox1.Text;

string route = textBox2.Text;

Bus bus = BusDL.SearchBusBySerial(serial);

if (bus != null)

{

BusDL.UpdateBus(serial, bus.GetRoute(), route);

BusDL.StoreData(path);

MessageBox.Show("Bus route Updated");

}

else

{

MessageBox.Show("Bus Not Present");

}

this.Hide();

}

}

### **ViewDriver**

public partial class ViewDriver : Form

{

public ViewDriver()

{

InitializeComponent();

}

public void DataBind()

{

dataGridView1.DataSource = null;

//dataGridView1.DataSource = StaffDL.EmployeesList;

foreach (Driver s in DriverDL.DriversList)

{

dataGridView1.DataSource = DriverDL.DriversList.Select(c => new { c.Name, c.Rank, c.ID1, c.DutyTime,c.LicenseNumber1 }).ToList();

}

dataGridView1.Refresh();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex == dataGridView1.NewRowIndex || e.RowIndex < 0)

return;

if (e.ColumnIndex == dataGridView1.Columns["Delete"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

Driver d = DriverDL.SearchEmployee(name);

DriverDL.RemoveEmployee(d.Name,d.Rank);

DataBind();

this.Hide();

}

if (e.ColumnIndex == dataGridView1.Columns["Edit"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

UpdateEmployee employee = new UpdateEmployee();

employee.ShowDialog();

DriverDL.StoreData("Driver.txt");

DataBind();

this.Hide();

}

}

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

{

DataBind();

}

}

### **ViewStaff**

public partial class ViewStaff : Form

{

public ViewStaff()

{

InitializeComponent();

}

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

{

DataBind();

}

public void DataBind()

{

dataGridView1.DataSource = null;

foreach (Staff s in StaffDL.EmployeesList)

{

dataGridView1.DataSource = StaffDL.EmployeesList.Select(c => new { c.Name, c.Rank, c.ID1, c.DutyTime }).ToList();

}

dataGridView1.Refresh();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex == dataGridView1.NewRowIndex || e.RowIndex < 0)

return;

if (e.ColumnIndex == dataGridView1.Columns["Delete"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

StaffDL.DeleteEmployee(StaffDL.SearchEmployee(name));

StaffDL.StoreData("Staff.txt");

DataBind();

this.Hide();

}

if (e.ColumnIndex == dataGridView1.Columns["Edit"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

UpdateEmployee employee = new UpdateEmployee();

employee.ShowDialog();

StaffDL.StoreData("Staff.txt");

DataBind();

this.Hide();

}

}

private void ViewStaff\_FormClosed(object sender, FormClosedEventArgs e)

{

/\* Application.Exit();\*/

}

}

### **ViewBus**

public partial class ViewBus : Form

{

public ViewBus()

{

InitializeComponent();

}

public void DataBind()

{

dataGridView1.DataSource = null;

//dataGridView1.DataSource = StaffDL.EmployeesList;

foreach (Bus s in BusDL.busesList)

{

dataGridView1.DataSource = BusDL.busesList.Select(c => new { c.BusNumber1,c.Date1,c.Route,c.Timing1 }).ToList();

}

dataGridView1.Refresh();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex == dataGridView1.NewRowIndex || e.RowIndex < 0)

return;

if (e.ColumnIndex == dataGridView1.Columns["Delete"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

BusDL.RemoveBus(name);

BusDL.StoreData("Bus.txt");

DataBind();

this.Hide();

}

if (e.ColumnIndex == dataGridView1.Columns["Edit"].Index)

{

int index = e.RowIndex;

string name = dataGridView1.Rows[index].Cells[2].Value.ToString();

UpdateBus bus = new UpdateBus();

bus.ShowDialog();

BusDL.StoreData("Bus.txt");

DataBind();

this.Hide();

}

}

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

{

DataBind();

}

}

### **CustomerMenu**

public partial class CustomerMenu : Form

{

Form currentChildForm;

public CustomerMenu()

{

InitializeComponent();

}

private void OpenChildForm(Form childForm)

{

if (currentChildForm != null)

{

currentChildForm.Close();

}

currentChildForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Left;

ChildPanel.Controls.Add(childForm);

ChildPanel.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

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

{ }

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

string type = comboBox2.Text.ToLower();

if(type == "add")

{

AddCustomerDetails customer = new AddCustomerDetails();

OpenChildForm(customer);

}

else if(type == "update")

{

UpdateSeat seat = new UpdateSeat();

OpenChildForm(seat);

}

else if(type == "cancel")

{

CancelSeat seat = new CancelSeat();

OpenChildForm(seat);

}

else if(type == "refund")

{

CancelSeat seat = new CancelSeat();

OpenChildForm(seat);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Form1 form = new Form1();

this.Hide();

form.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

SignUp sign = new SignUp();

this.Hide();

sign.Show();

}

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

{

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

string type = comboBox3.Text.ToLower();

if (type == "bus")

{

ViewBusForCustomer view = new ViewBusForCustomer();

OpenChildForm(view);

}

}

}

### **AddCustomerDetails**

public partial class AddCustomerDetails : Form

{

public AddCustomerDetails()

{

InitializeComponent();

}

private void ClearData()

{

textBox1.Text = " ";

textBox2.Text = " ";

textBox3.Text = " ";

textBox4.Text = " ";

textBox5.Text = " ";

textBox6.Text = " ";

textBox7.Text = " ";

textBox8.Text = " ";

textBox9.Text = " ";

}

private void button2\_Click(object sender, EventArgs e)

{

string path = "Customer.txt";

string serial = textBox7.Text;

string name = textBox1.Text;

string password = textBox2.Text;

string contact = textBox3.Text;

string address = textBox4.Text;

int seats = int.Parse(textBox5.Text);

int number = 0;

List<int> seatList = new List<int>();

if (seats == 1)

{

if (textBox6.Text != " ")

{

number = int.Parse(textBox6.Text);

if (CustomerDL.CheckIfSeatIsReserved(number, serial))

{

MessageBox.Show("Cannot select this seat");

textBox6.Text = " ";

}

else

{

seatList.Add(number);

textBox6.Text = " ";

}

}

if (number == 0)

{

int number2 = int.Parse(textBox9.Text);

if (CustomerDL.CheckIfSeatIsReserved(number2, serial))

{

MessageBox.Show("Cannot select this seat");

textBox9.Text = " ";

}

else

{

seatList.Add(number2);

textBox9.Text = " ";

}

}

}

else if (seats == 2)

{

number = int.Parse(textBox6.Text);

if (CustomerDL.CheckIfSeatIsReserved(number, serial))

{

textBox6.Text = " ";

MessageBox.Show("Cannot select this seat");

}

else

{

textBox6.Text = " ";

seatList.Add(number);

}

int number2 = int.Parse(textBox9.Text);

if (CustomerDL.CheckIfSeatIsReserved(number2, serial))

{

MessageBox.Show("Cannot select this seat");

textBox9.Text = " ";

}

else

{

textBox9.Text = " ";

seatList.Add(number2);

}

}

else

{

MessageBox.Show("Cannot select this number of seats");

}

string payment = textBox8.Text.ToLower();

double charges = GetPayment(payment, seats);

if (seatList.Count != 0)

{

if (charges != 0)

{

Customer customer = new Customer(name, password, "customer", contact, address, charges, seatList, serial);

CustomerDL.AddCustomerToList(customer);

CustomerDL.StoreData(path);

MessageBox.Show("Seat reserved");

ClearData();

}

else

{

MessageBox.Show("Invalid Input ");

ClearData();

}

}

ClearText();

this.Hide();

}

private double GetPayment(string type, int number)

{

if (type == "cash")

{

MessageBox.Show("Your payable amount is rupees : " + 1500 \* number);

return 1500 \* number;

}

else if (type == "card")

{

MessageBox.Show("Your payable amount is rupees: " + 1425 \* number);

return 1425 \* number;

}

else

{

MessageBox.Show("Invalid method");

}

return 0;

}

private void label1\_Click(object sender, EventArgs e)

{ }

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

{ }

private void textBox7\_TextChanged(object sender, EventArgs e)

{ }

private void label9\_Click(object sender, EventArgs e)

{ }

private void ClearText()

{

textBox1.Text = "";

}

private void button3\_Click(object sender, EventArgs e)

{

string serial = textBox7.Text;

bool bus = BusDL.CheckBusBySerial(serial);

if (bus)

{

MessageBox.Show("Bus Is Selected");

}

else

{

MessageBox.Show(" Bus Cannot Be Selected");

}

}

private void button1\_Click(object sender, EventArgs e)

{

CustomerMenu menu = new CustomerMenu();

menu.Show();

ClearText();

this.Hide();

}

private void button4\_Click(object sender, EventArgs e)

{ }

}

### **CancelSeat**

public partial class CancelSeat : Form

{

public CancelSeat()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string name = textBox1.Text;

int seta = int.Parse(textBox2.Text);

if (CustomerDL.CancelSeat(name, seta))

{

MessageBox.Show("Cancelled & 50% refund will be transferred to you ");

CustomerDL.StoreData("Customer.txt");

}

else

{

MessageBox.Show("Invalid Data");

}

this.Hide();

}

}

### **UpdateSeat**

public partial class UpdateSeat : Form

{

public UpdateSeat()

{

InitializeComponent();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

string name = textBox1.Text;

int num = int.Parse(textBox2.Text);

int updated = int.Parse(textBox3.Text);

bool flag = CustomerDL.CheckIfCustomerIsPresent(name);

bool flag2 = CustomerDL.CheckIfSpecificSeatIsREserved(name, num);

if (flag2 == true && flag == true)

{

bool flag3 = CustomerDL.UpdateSeatNumber(num, updated);

if (flag3)

{

MessageBox.Show("Seat Updated");

CustomerDL.StoreData("Customer.txt");

}

}

else

{

MessageBox.Show("Invalid data we cannot find it in our record");

}

this.Hide();

}

}

# Conclusion:

The project aimed to develop a business application with a focus on modularity, separation of concerns, and utilizing object-oriented design principles and design patterns. The project successfully implemented the BL (Business Logic), DL (Data Access Layer), and UI (User Interface) layers to achieve these goals.

By utilizing the BL, DL, and UI design patterns, the project ensured modularity and separation of concerns. The BL layer encapsulated the business logic and rules, the DL layer handled data access and manipulation, and the UI layer provided the user interface for interaction. This design approach allowed for easy maintenance, scalability, and code reusability.

# Challenges:

Throughout the development process, some challenges were faced. These included:

1. Designing a flexible and extensible architecture: Ensuring that the application's architecture could accommodate future changes and enhancements required thoughtful design decisions and planning.

2. Managing data persistence: Implementing an efficient and reliable data storage mechanism and integrating it with the DL layer posed challenges. Selecting the appropriate database technology and ensuring proper data retrieval and manipulation were key considerations.

3. User input validation: Validating user input to maintain data integrity and security required careful handling and validation mechanisms in the BL and DL layers.

# Achievements:

Despite these challenges, the project achieved the following:

1. Modularity and separation of concerns: The project successfully implemented a layered architecture with clear separation between the BL, DL, and UI layers. This ensured maintainability, scalability, and ease of testing.

2. Code reusability: By utilizing object-oriented design principles, design patterns, and modular architecture, the project achieved code reusability. This allowed for efficient development and reduced duplication of code.

3. Improved maintainability: The separation of concerns and modular design made it easier to identify and modify specific components of the application without impacting other parts. This facilitated maintenance and future enhancements.

# Lesson Learned:

Lessons learned from the project include the importance of proper architectural planning, adherence to design principles, and the effective use of design patterns to achieve modular and maintainable code. Additionally, careful consideration should be given to data persistence and user input validation to ensure data integrity and security.