SKYLINES

Session 2023 - 2027

Submitted by:

Aamir Hashmi 2023-CS-11

Supervised by:

Prof. Dr. Muhammad Awais Hassan

& Sir. Laeeq Khan Niazi

Course:

CSC-102 Object Oriented Programming

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Content

1. Short Description:

i. Objective:

This app is an efficient and secure Airline Management System that caters to the functionalities required by both administrator and regular users.

ii. Output Expectations:

a. User-Friendly Interface

The interface for Console as well as GUI is designed to be easily understandable for users, they do not require any additional information before using the software.

b. Seamless Update Capability

This app is properly structured according to OOP principles, and I have also implemented Factory and Singleton patterns to facilitate easy updates in the future. It is versatile and can utilize either File Handling or Database for storing data.I

2. Users of Application:

There are two types of users in this application. An Admin who have the authority over the application and the Client who can access his/her account info and access services through application.

- **Admin**

  The admin can schedule flights, update flights schedule based on weather problems and manage the system by hiring and controlling staff. The admin can also expel staff or view revenue generated by different flights. He can also view the feedback of clients.

- **Client**

  The Client/traveler can book a seat in flight or cancel his reserved seat. He can search for flights, view special deals, submit feedback and track his reserved flight via single platform.

3. Functional Requirements

User Type	Functions	Action Performed
Admin/Manager	Add Flight	Add a flight in my system.
	Edit Flight Schedule	Edit the schedule of any flight if needed due to weather problems.
	View Flights	View all the flights available in system
	View Feedback	View the Feedback of clients/travelers.
	Hire Staff	Hire the staff for Airline system.
	Expel Staff	Expel staff of Airline System.
	View Staff	View staff of Airline System.
	Update Staff	Update any staff information, salary and designation.
	View Flights Revenue	View Revenue generated from different flights.
	Issue Discount	Issue Discount to Flights.
	View Members	View members of company.
Client/Traveler	Book Flight	Book/reserve seat in flight.
	Cancel Reserved Seat	Cancel the seat you reserved in any flight.
	View Reserved Flights	View the flights schedule in which you reserved your seat.
	Search Flights	Search and view flights of his/her interest.
	Submit Feedback	Submit feedback about our services.
	Get Discounts	Avail Discounts on flights.
	Get Member Ship Card	Get Member Ship Cards of Company.

4. Wireframes for GUI Application

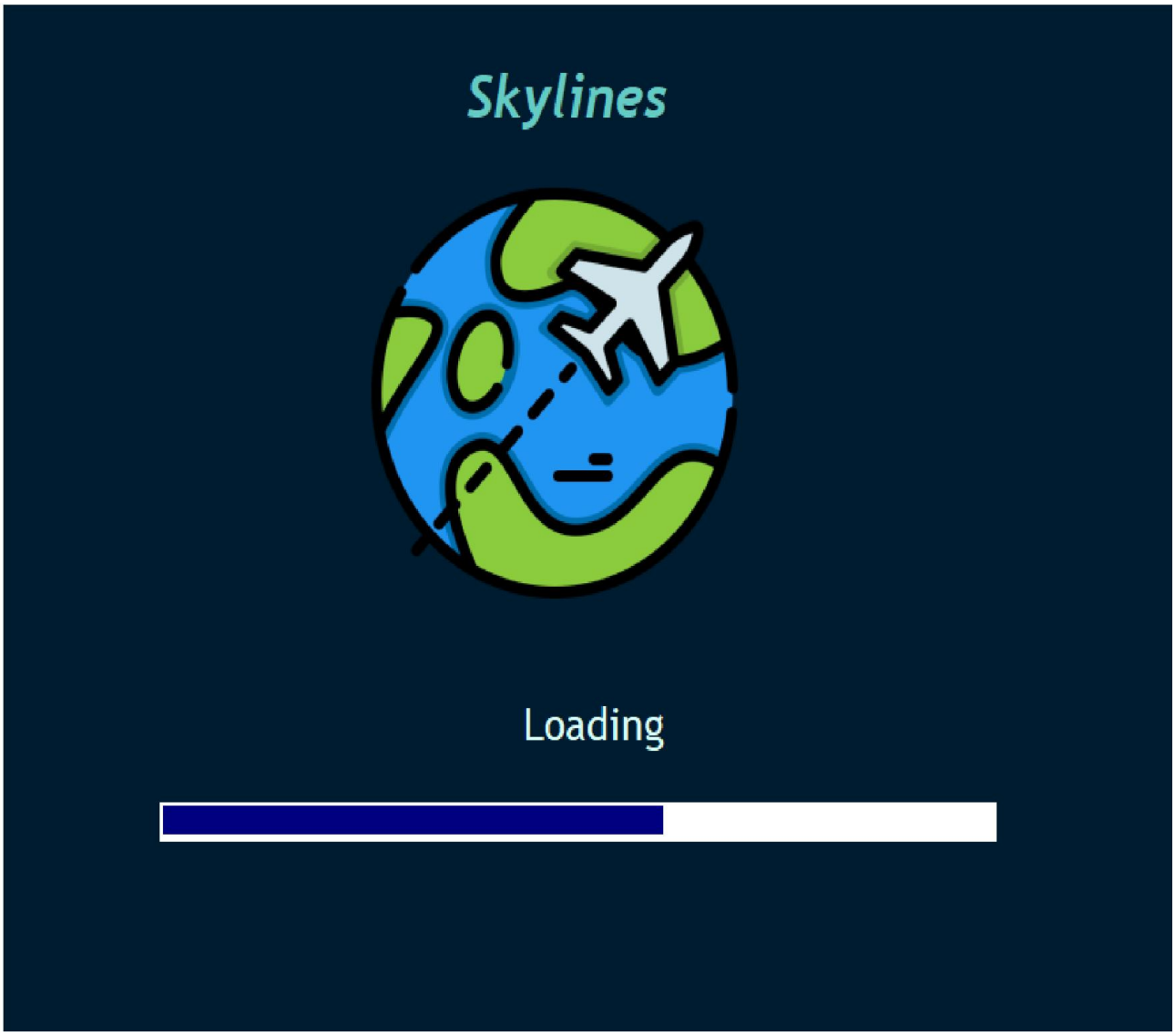


Figure 01 GUI - Loading Page

Figure 02 GUI - Login Page

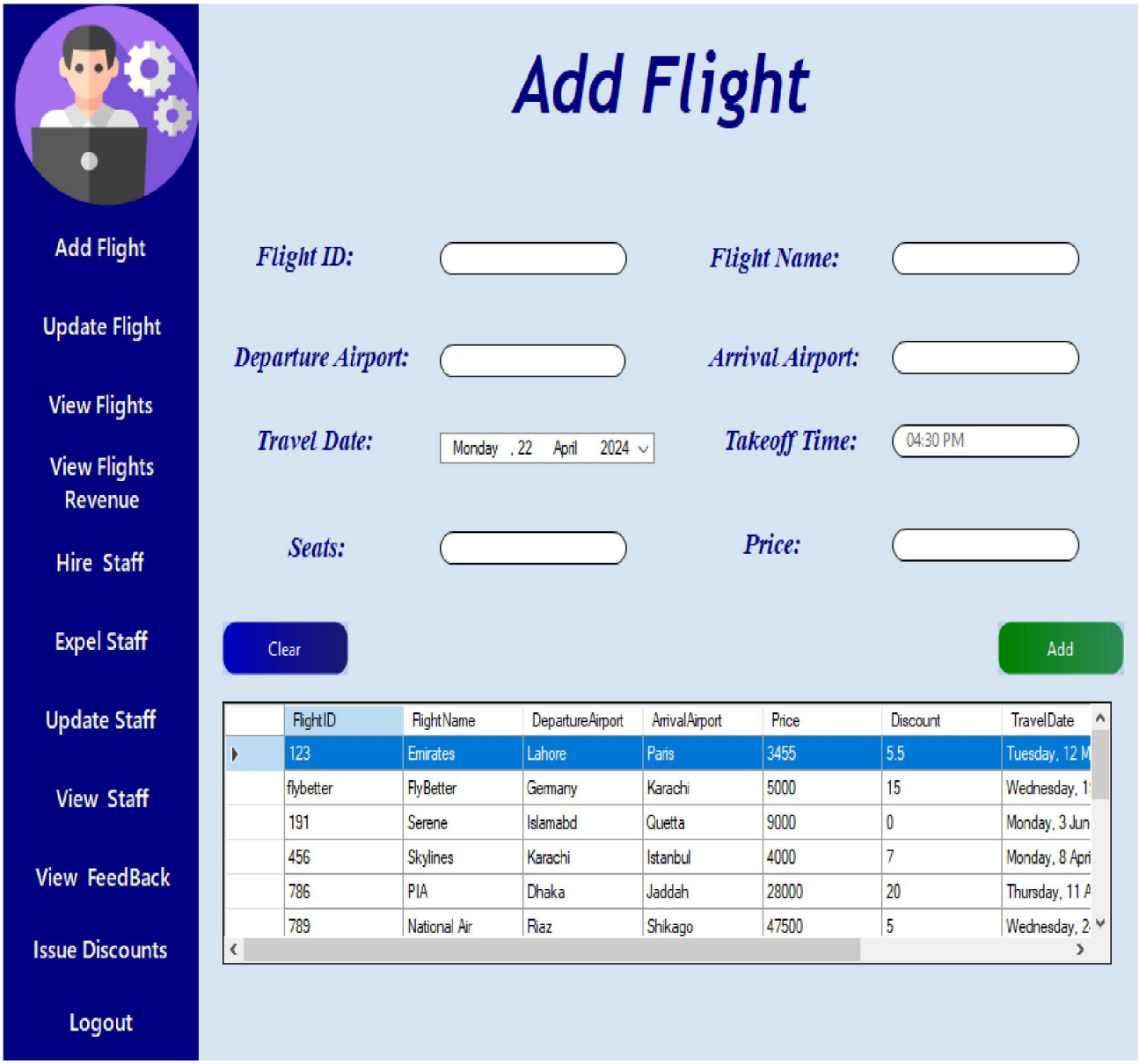


Figure 03 GUI - Admin Panel

Figure 04 GUI - User Panel

FlightID	FlightName	Price	SeatsBooked	RevenueGenerated
123	Emirates	3455	2	6910
flybetter	FlyBetter	5000	1	5000
191	Serene	9000	0	0
456	Skylines	4000	2	8000
786	PIA	28000	2	56000
789	National Air	47500	1	47500
491111	Bombars	1395000	2	2790000
1122	Rescue	530750	1	530750
112233	Tosham	53200	0	0
9977	ComfortLines	45000	1	45000
9922	Paksitan Aires	45000	1	45000

Figure 05 GUI – View Flights Revenue Page

5. Wireframes for Console Application

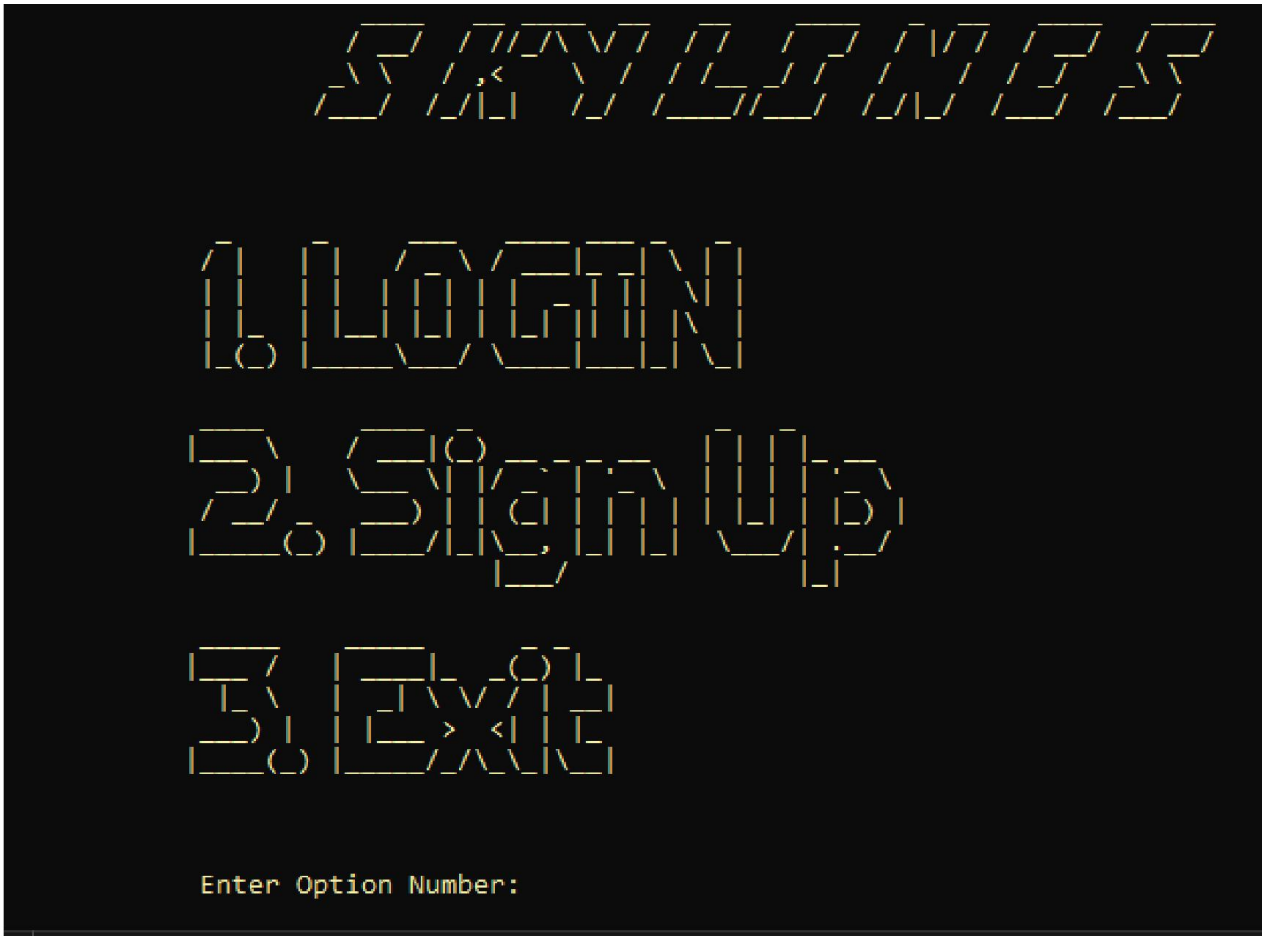
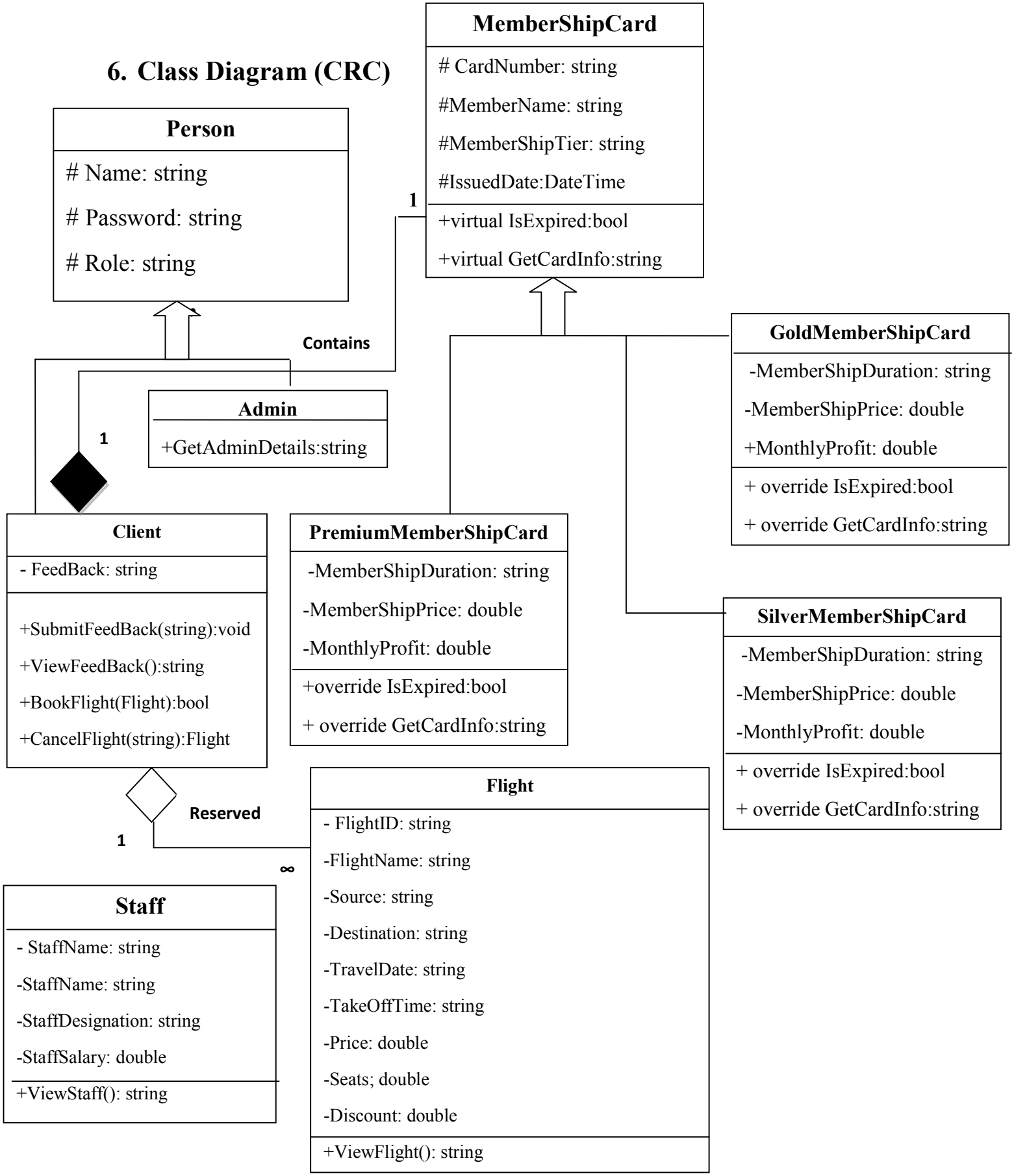


Figure 6-Console-MainPage

Figure 7-Console-AdminPanel

Figure 8-Console-UserPanel

6. Class Diagram (CRC)



7. Complete Code

//FlightBL

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

namespace SkyLinesLibrary
{
   public class Flight
   {
      private string FlightID;
      private string FlightName;
      private string Source;
      private string Destination;
      private double Price;
      private double Discount;
      private string TravelDate;
      private string TakeoffTime;
      private double Seats;

      // Constructor for initializing a new instance of the 'Flight' class with provided details
      public Flight(string FlightID, string FlightName, string Source, string Destination, string
TravelDate, string TakeoffTime, double Price, double Seats)
      {
         this.FlightID = FlightID;
```

```
        this.FlightName = FlightName;
        this.Source = Source;
        this.Destination = Destination;
        this.TravelDate = TravelDate;
        this.TakeoffTime = TakeoffTime;
        this.Price = Price;
        this.Seats = Seats;
        this.Discount = 0;
    }

    // Method to view flight details
    public string ViewFlight()
    {
        return ($" {FlightID}\t\t\t\t {FlightName}\t\t\t {Source}\t\t\t\t {Destination}\t\t\t\t
{TravelDate}\t\t\t {TakeoffTime}\t\t\t {Price}\t\t{Seats}");
    }

    // Method to get the flight name
    public string GetFlightName()
    {
        return FlightName;
    }

    // Method to set the flight name
    public void SetFlightname(string name)
    {
        this.FlightName = name;
    }
```

```
// Method to get the flight ID
public string GetFlightID()
{
    return FlightID;
}

// Method to set the flight ID
public void SetFlightID(string ID)
{
    this.FlightID = ID;
}

// Method to get the source of the flight
public string GetSource()
{
    return Source;
}

// Method to set the source of the flight
public void SetSource(string source)
{
    this.Source = source;
}

// Method to get the destination of the flight
public string GetDestination()
{
    return Destination;
```

```
}

// Method to set the destination of the flight
public void SetDestination(string destination)
{
    this.Destination = destination;
}

// Method to get the travel date of the flight
public string GetTravelDate()
{
    return TravelDate;
}

// Method to set the travel date of the flight
public void SetTravelDate(string TravelDate)
{
    this.TravelDate = TravelDate;
}

// Method to get the takeoff time of the flight
public string GetTakeoffTime()
{
    return TakeoffTime;
}

// Method to set the takeoff time of the flight
public void SetTakeoffTime(string time)
```

```
{
    this.TakeoffTime = time;
}

// Method to get the discount applied to the flight
public double GetDiscount()
{
    return Discount;
}

// Method to set the discount applied to the flight
public void SetDiscount(double Discount)
{
    this.Discount = Discount;
}

// Method to get the price of the flight
public double GetPrice()
{
    return Price;
}

// Method to set the price of the flight
public void SetPrice(double price)
{
    this.Price = price;
}
```

```
        // Method to get the number of seats in the flight
        public double GetSeats()
        {
            return Seats;
        }

        // Method to set the number of seats in the flight
        public void SetSeats(double seats)
        {
            this.Seats = seats;
        }

        // Method to issue discount on the flight
        public void IssueDiscount(string FlightID, double Discount)
        {
            this.Discount = Discount;
            Price -= Price * (Discount / 100);
        }

        // Method to calculate revenue for the flight based on booked seats
        public double GetRevenue(int bookedseats)
        {
            double revenue = bookedseats * Price;
            return revenue;
        }
    }
}
```

//FlightInterface

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

namespace SkyLinesLibrary
{
    public interface IFlightDL
    {
        void AddFlight(Flight f);
        void EditFlight(string name, string flightID, string source, string destination, string date,
string takeoff, double price, double seats);
        bool CheckValidFlightID(string ID);
        bool IsFlightExist(string ID);
        void LoadFlights();

        void StoreFlights(Flight fl);
        void UpdateFlight(string originalID, string source, string destination, string date, string
takeoff, double price, double seats);

        void UpdateDiscount(string FlightID, double Discount, double Price);
        List<Flight> GetAllFlights();
        Flight GetFlightByID(string FlightID);
    }
}
```

//FlightDL:FH

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.IO;
namespace SkyLinesLibrary
{
    public class FlightDL_FH : IFlightDL
    {
        public static List<Flight> Flights = new List<Flight>();

        private static string filepath;
        private static FlightDL_FH FlightDL_FHInstance;

        // Constructor to initialize FlightDL_FH instance
        private FlightDL_FH(string FilePath)
        {
            filepath = FilePath;
            LoadFlights();
        }

        // Method to get an instance of FlightDL_FH
        public static FlightDL_FH GetFlightDL_FHInstance(string FilePath)
        {
```

```
        if (FlightDL_FHInstance == null)
        {
            FlightDL_FHInstance = new FlightDL_FH(FilePath); // Create new instance if not
exists
        }
        return FlightDL_FHInstance; // Return instance
    }

    // Method to add a flight
    public void AddFlight(Flight f)
    {
        Flights.Add(f); // Add flight to the list
        StoreFlights(f); // Store flight in the file
    }

    // Method to edit flight details
    public void EditFlight(string name, string flightID, string source, string destination, string
date, string takeoff, double price, double seats)
    {
        // Find the flight with the given ID and update its details
        for (int i = 0; i < Flights.Count; i++)
        {
            if (Flights[i].GetFlightID() == flightID)
            {
                Flights[i].SetSource(source);
                Flights[i].SetDestination(destination);
                Flights[i].SetTravelDate(date);
                Flights[i].SetTakeoffTime(takeoff);
                Flights[i].SetPrice(price);
```

```
            Flights[i].SetSeats(seats);
            break;
        }
    }
    UpdateFlight(flightID, source, destination, date, takeoff, price, seats); // Update flight in the file
}

// Method to check if a flight ID is valid
public bool CheckValidFlightID(string ID)
{
    // Check if flight with the given ID already exists
    for (int i = 0; i < Flights.Count; i++)
    {
        if (Flights[i].GetFlightID() == ID)
        {
            return false; // Flight ID is not valid
        }
    }
    return true; // Flight ID is valid
}

// Method to check if a flight exists
public bool IsFlightExist(string ID)
{
    // Check if flight with the given ID exists
    for (int i = 0; i < Flights.Count; i++)
    {
        if (Flights[i].GetFlightID() == ID)
```

```
        {
            return true; // Flight exists
        }
    }
    return false; // Flight does not exist
}

// Method to load flights from the file
public void LoadFlights()
{
    string name, ID, source, destination, date, takeoff, record;
    double price, seats, discount;
    if (File.Exists(filepath))
    {
        StreamReader flightfile = new StreamReader(filepath);
        while ((record = flightfile.ReadLine()) != null)
        {
            string[] data = record.Split(';');
            ID = data[0];
            name = data[1];
            source = data[2];
            destination = data[3];
            date = data[4];
            takeoff = data[5];
            price = double.Parse(data[6]);
            seats = double.Parse(data[7]);
            discount = double.Parse(data[8]);
            Flight f = new Flight(ID, name, source, destination, date, takeoff, price, seats);
```

```
            f.SetDiscount(discount);
            Flights.Add(f);
        }
        flightfile.Close();
    }
    else { return; }
}

// Method to store flights in the file
public void StoreFlights(Flight fl)
{
    StreamWriter Flightfile = new StreamWriter(filepath, true);
    Flightfile.WriteLine($"{fl.GetFlightID()};
{fl.GetFlightName()};{fl.GetSource()};{fl.GetDestination()}; {fl.GetTravelDate()};
{fl.GetTakeoffTime()}; {fl.GetPrice()}; {fl.GetSeats()};{fl.GetDiscount()}");
    Flightfile.Flush();
    Flightfile.Close();
}

// Method to update flight details in the file
public void UpdateFlight(string originalID, string source, string destination, string date,
string takeoff, double price, double seats)
{
    File.WriteAllText(filepath, string.Empty); // Clear file contents
    foreach (Flight fl in Flights)
    {
        StoreFlights(fl); // Store flights again in the file
    }
}
```

```
// Method to update flight discount and price in the file
public void UpdateDiscount(string FlightID, double Discount, double Price)
{
    File.WriteAllText(filepath, string.Empty); // Clear file contents
    foreach (Flight fl in Flights)
    {
        StoreFlights(fl); // Store flights again in the file
    }
}

// Method to get all flights
public List<Flight> GetAllFlights()
{
    return Flights; // Return list of all flights
}

// Method to get flight by ID
public Flight GetFlightByID(string FlightID)
{
    foreach (Flight f in Flights)
    {
        if (f.GetFlightID() == FlightID)
        {
            return f; // Return flight with the given ID
        }
    }
    return null; // Return null if flight not found}}}
```

//FlightDL:DB

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

namespace SkyLinesLibrary
{
    public class FlightDL_DB : IFlightDL
    {
        // Static list to store flight data
        public static List<Flight> Flights = new List<Flight>();

        // Instance of database configuration
        private static DbConfig db = DbConfig.GetInstance();

        // Instance of FlightDL_DB
        private static FlightDL_DB FlightDL_DBInstance;

        // Constructor to initialize FlightDL_DB instance
        private FlightDL_DB(string connectionstring)
        {
            LoadFlights(); // Load flights from the database
        }
```

```
// Method to get an instance of FlightDL_DB
public static FlightDL_DB GetFlightDL_DBInstance(string connectionstring)
{
    if (FlightDL_DBInstance == null)
    {
        FlightDL_DBInstance = new FlightDL_DB(connectionstring); // Create new instance
if not exists
    }
    return FlightDL_DBInstance; // Return instance
}

// Method to add a flight
public void AddFlight(Flight f)
{
    Flights.Add(f); // Add flight to the list
    StoreFlights(f); // Store flight in the database
}

// Method to edit flight details
public void EditFlight(string name, string flightID, string source, string destination, string
date, string takeoff, double price, double seats)
{
    // Find the flight with the given ID and update its details
    for (int i = 0; i < Flights.Count; i++)
    {
        if (Flights[i].GetFlightID() == flightID)
        {
            Flights[i].SetSource(source);
            Flights[i].SetDestination(destination);
```

```
                Flights[i].SetTravelDate(date);
                Flights[i].SetTakeoffTime(takeoff);
                Flights[i].SetPrice(price);
                Flights[i].SetSeats(seats);
                break;
            }
        }
        UpdateFlight(flightID, source, destination, date, takeoff, price, seats); // Update flight in the database
    }

    // Method to check if a flight ID is valid
    public bool CheckValidFlightID(string ID)
    {
        // Check if flight with the given ID already exists
        for (int i = 0; i < Flights.Count; i++)
        {
            if (Flights[i].GetFlightID() == ID)
            {
                return false; // Flight ID is not valid
            }
        }
        return true; // Flight ID is valid
    }

    // Method to check if a flight exists
    public bool IsFlightExist(string ID)
    {
        // Check if flight with the given ID exists
```

```
    for (int i = 0; i < Flights.Count; i++)
    {
        if (Flights[i].GetFlightID() == ID)
        {
            return true; // Flight exists
        }
    }
    return false; // Flight does not exist
}

// Method to load flights from the database
public void LoadFlights()
{
    string name, ID, source, destination, date, takeoff;
    double price, seats, discount;
    string searchquery = "Select * From Flights"; // SQL query to select all flights
    SqlCommand command = new SqlCommand(searchquery, db.GetConnection());
    SqlDataReader reader = command.ExecuteReader();
    while (reader.Read())
    {
        // Read flight details from the database
        ID = reader.GetString(0);
        name = reader.GetString(1);
        source = reader.GetString(2);
        destination = reader.GetString(3);
        date = reader.GetString(4);
        takeoff = reader.GetString(5);
        price = reader.GetDouble(6);
```

```
            seats = reader.GetDouble(7);
            discount = reader.GetDouble(8);

            // Create Flight object with retrieved data
            Flight f = new Flight(ID, name, source, destination, date, takeoff, price, seats);
            f.SetDiscount(discount); // Set flight discount
            Flights.Add(f); // Add flight to the list
        }
        reader.Close();
    }

    // Method to store flights in the database
    public void StoreFlights(Flight fl)
    {
        // SQL query to insert flight details into the database
        string query = string.Format("INSERT INTO
Flights(FlightID,FlightName,Source,Destination,TravelDate,TakeoffTime,Price,Seats,Discount)
" + "Values ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}')",
                                 fl.GetFlightID(), fl.GetFlightName(), fl.GetSource(),
fl.GetDestination(), fl.GetTravelDate(), fl.GetTakeoffTime(), fl.GetPrice(), fl.GetSeats(),
fl.GetDiscount());
        SqlCommand cmd = new SqlCommand(query, db.GetConnection());
        cmd.ExecuteNonQuery();
    }

    // Method to update flight details in the database
    public void UpdateFlight(string originalID, string source, string destination, string date,
string takeoff, double price, double seats)
    {
        // SQL query to update flight details in the database
```

```
        string query = string.Format("UPDATE Flights SET
Source='{0}',Destination='{1}',TravelDate='{2}',TakeoffTime='{3}',Price='{4}',Seats='{5}'WH
ERE FlightID='{6}'",
                                source, destination, date, takeoff, price, seats, originalID);
        SqlCommand cmd = new SqlCommand(query, db.GetConnection());
        cmd.ExecuteNonQuery();
    }

    // Method to update flight discount and price in the database
    public void UpdateDiscount(string FlightID, double Discount, double Price)
    {
        // SQL query to update flight discount and price in the database
        string query = string.Format("UPDATE Flights SET Discount='{0}',Price='{1}' WHERE
FlightID='{2}'", Discount, Price, FlightID);
        SqlCommand cmd = new SqlCommand(query, db.GetConnection());
        cmd.ExecuteNonQuery();
    }

    // Method to get all flights
    public List<Flight> GetAllFlights()
    {
        return Flights; // Return list of all flights
    }

    // Method to get flight by ID
    public Flight GetFlightByID(string FlightID)
    {
        foreach (Flight f in Flights)
        {
```

```
            if (f.GetFlightID() == FlightID)
            {
                return f; // Return flight with the given ID
            }
        }
        return null; // Return null if flight not found
    }
  }
}
```

//FlightUI

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SkyLinesLibrary;
namespace SkyLines
{
  internal class FlightUI
  {
//This function will print feedback of all clients.
    public static void ViewAllFeedBack()
    {
        List<Client> Clients = ObjectHandler.GetClientDL().GetAllClients();
        Console.WriteLine("\t\t\t View FeedBack\n\n");
        for (int i = 0; i < Clients.Count; i++)
        {
            if (Clients[i].GetFeedBack() != null)
```

```
            {
                Console.WriteLine(Clients[i].ViewFeedBack());
            }
        }
        Console.WriteLine("\n\n Press any key to Continue..");
        Console.ReadKey();
        Console.Clear();
    }
//Thhis function will take input from user for adding new flight.
    public static void AddNewFlight()
    {

        string name; string ID; string source; string destination; string date; string takeoff; double
price; double seats;
        string checkprice, checkseats;
        while (true)
        {
Console.WriteLine("\t\t\t Add Flight\n\n");
            Console.Write(" Enter Flight ID: ");
            ID = Console.ReadLine();
        if(!(Validations.CheckCommaandColon(ID)))
        {
        Console.WriteLine(" Invalid Flight ID.Comma and colon is not Allowed!!!");
                Console.WriteLine(" Press any key to continue!!!");
                Console.ReadKey();
                Console.Clear();
                continue;
        }
         if (!(ObjectHandler.GetFlightDL().CheckValidFlightID(ID)))
```

```
    {
        Console.WriteLine(" This FlightID Already Exist.Kindly Enter any other ID.");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
    }
    Console.Write(" Enter Flight Name: ");
    name = Console.ReadLine();
if(!(Validations.CheckCommaandColon(name)))
{
Console.WriteLine(" Invalid Flight Name.Comma and colon is not Allowed!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
}
    Console.Write(" Enter Departure Airport: ");
    source = Console.ReadLine();
if(!(Validations.CheckCommaandColon(source)))
{
Console.WriteLine(" Invalid Departure Airport.Comma and colon is not Allowed!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
}
    Console.Write(" Enter Arrival Airport: ");
```

```
    destination = Console.ReadLine();
if(!(Validations.CheckCommaandColon(destination)))
{
Console.WriteLine(" Invalid Arrival Airport.Comma and colon is not Allowed!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
}
    Console.Write(" Enter Departure Date (DD-MM-YYYY): ");
    date = Console.ReadLine();
    if (!(Validations.CheckValidDate(date)))
    {
        Console.WriteLine(" Invalid Date!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
    }
    Console.Write(" Enter Departure Time i.e (12:00 AM,05:30 PM): ");
    takeoff = Console.ReadLine();
    if (!(Validations.CheckValidTime(takeoff)))
    {
        Console.WriteLine(" Invalid Departure Time!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
```

```
        }

        Console.Write(" Enter Ticket Price: ");
        checkprice = Console.ReadLine();
        if (!(Validations.CheckNumber(checkprice)))
        {
            Console.WriteLine(" Invalid Price!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
        }
        price = double.Parse(checkprice);
        Console.Write(" Enter Number of Seats: ");
        checkseats = Console.ReadLine();
        if (!(Validations.CheckNumber(checkseats)))
        {
            Console.WriteLine(" Invalid Input!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
        }
        seats = double.Parse(checkseats);
        Flight f = new Flight(ID,name, source, destination, date, takeoff, price, seats);
        ObjectHandler.GetFlightDL().AddFlight(f);
        break;
    }
```

```
        Console.WriteLine(" \nThe Desired Flight is Sucessfully Added.");
        Console.WriteLine(" \n\nPress any key to continue!!!");
        Console.ReadKey();
    }
//This function will take input from user for updating Flight.
    public static void UpdateFlight()
    {

        string name; string ID; string source; string destination; string date; string takeoff;
double price; double seats;
        string checkprice, checkseats;
        Console.Write(" Enter Flight Name: ");
        name = Console.ReadLine();
        Console.Write(" Enter Flight ID: ");
        ID = Console.ReadLine();
        if (ObjectHandler.GetFlightDL().IsFlightExist(ID))
        {
          while (true)
          {
        Console.WriteLine("\t\t\t Edit Flight Schedule\n\n");
            Console.Write(" Enter Departure Airport: ");
            source = Console.ReadLine();
        if(!(Validations.CheckCommaandColon(source)))
        {
        Console.WriteLine(" Invalid Departure Airport.Comma and colon is not Allowed!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
```

```
}
        Console.Write(" Enter Arrival Airport: ");
        destination = Console.ReadLine();
if(!(Validations.CheckCommaandColon(destination)))
{
Console.WriteLine(" Invalid Arrival Airport.Comma and colon is not Allowed!!!");
        Console.WriteLine(" Press any key to continue!!!");
        Console.ReadKey();
        Console.Clear();
        continue;
}
        Console.Write(" Enter Departure Date (DD-MM-YYYY): ");
        date = Console.ReadLine();
        if (!(Validations.CheckValidDate(date)))
        {
            Console.WriteLine(" Invalid Date!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
        }
        Console.Write(" Enter Departure Time i.e (12:00 AM,05:30 PM): ");
        takeoff = Console.ReadLine();
        if (!(Validations.CheckValidTime(takeoff)))
        {
            Console.WriteLine(" Invalid Departure Time!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
```

```
            Console.Clear();
            continue;
        }

        Console.Write(" Enter Ticket Price: ");
        checkprice = Console.ReadLine();
        if (!(Validations.CheckNumber(checkprice)))
        {
            Console.WriteLine(" Invalid Price!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
        }
        price = double.Parse(checkprice);
        Console.Write(" Enter Number of Seats: ");
        checkseats = Console.ReadLine();
        if (!(Validations.CheckNumber(checkseats)))
        {
            Console.WriteLine(" Invalid Input!!!");
            Console.WriteLine(" Press any key to continue!!!");
            Console.ReadKey();
            Console.Clear();
            continue;
        }
        seats = double.Parse(checkseats);
        break;
    }
```

```
            ObjectHandler.GetFlightDL().EditFlight(name, ID, source, destination, date, takeoff,
price, seats);
            Console.WriteLine(" \nThe Desired Flight's schedule is sucessfully updated.");
        }
        else
        {
            Console.WriteLine(" \nNo Such Flight Found.");
        }
        Console.WriteLine(" \n\nPress any key to continue!!!");
        Console.ReadKey();
    }
//Thos function will print all the flights.
    public static void ViewAllFlights()
    {
        List<Flight> Flights = ObjectHandler.GetFlightDL().GetAllFlights();
        Console.WriteLine("\t\t\t View Flights\n\n");
        Console.WriteLine("Flight ID\t\t\t Flight Name\t\t\t Depature Airport\t\t Arrival
Airport\t\t Departure Date\t\t Departure Time \t\t Landing Time\t\t Price\t\t Seats\n");
        for (int i = 0; i < Flights.Count; i++)
        {
            Console.WriteLine(Flights[i].ViewFlight());
        }
        Console.WriteLine("\n\n\n Press any key to continue");
        Console.ReadKey();
        Console.Clear();
    }

//This function will print all flights and revenue generated by them.
public static void ViewFlightsRevenue()
```

```
{

List<Flight> flight = ObjectHandler.GetFlightDL().GetAllFlights();

        if (flight != null && flight.Count > 0)
        {
Console.WriteLine("FlightID\t\tFlightName\t\tPrice\t\tSeatsBooked\t\tRevenueGenerated");
            foreach (Flight fl in flight)
            {
                Console.WriteLine($"{fl.GetFlightID()}\t\t
{fl.GetFlightName()}\t\t{fl.GetPrice()}\t\t{GetSeatsBooked(fl.GetFlightID())}\t\t\t{fl.GetReven
ue(GetSeatsBooked(fl.GetFlightID()))}");
            }
        }
}
//This function will calculate seats booked for every flight.
public  static int GetSeatsBooked(string FlightID)
      {
        int bookedseats = 0;
        List<Client> clients = ObjectHandler.GetClientDL().GetAllClients();
        foreach (Client C in clients)
        {
            foreach (Flight F in C.GetBookedFlights())
            {
                if (FlightID == F.GetFlightID())
                    bookedseats++;
            }
        }
        return bookedseats;}}}
```