**Cricket Club Management System**

****

Session: 2022 – 2026

**Submitted by:**

Mohammad Abdullah 2022-CS-155

**Supervised by:**

Prof. Dr. Muhammad Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents**

**Introduction………………………………………………………………………………...……3**

**Class Responsibility Collaboration Card……………………………...………………4**

**Object Oriented Programming………………………………………………………...…6**

**Design Pattern Implementation..………………………………………………………...8**

**Class Details……........……………………………………………………………..…….......….9**

**Code………….…........……………………………………………………………..……........…..10**

**Wireframes.….........……………………………………………………………..……........…..64**

**Conclusion……........……………………………………………………………..……........…..67**

**Project Link**

**https://youtu.be/uwTlOU4sttY**

* **Introduction**
* **Overview**

Cricket Clubs face difficulties when managing their players performance, finances. The objective of cricket club management system is to provide an organized software to the problems for cricket club, so that they could manage their operations in a much better way.

This management system have four different users. Its users include Admin, Player, Coach, Fan. All these users have different kinds of functionalities. This project will solve the problems of many cricket clubs.

* **Objectives**

The objectives of this system includes

1. Provision of a functionality to achieve efficient management of Clubs affairs.
2. To facilitate the clubs in match scheduling and its provision to the players, coaches and fans.
3. To manage the team easily.

* **Functionality**

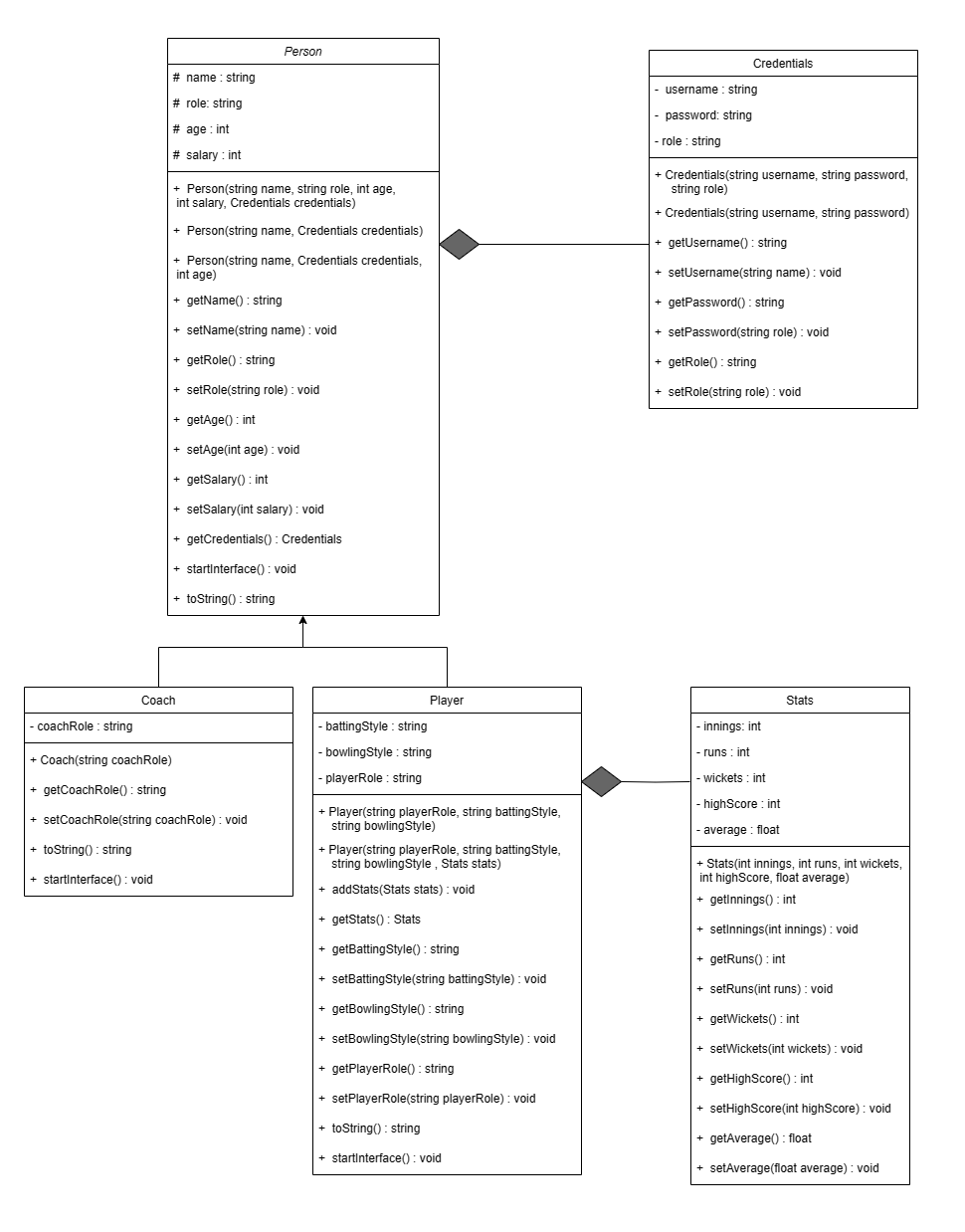
The intended functionality for the project includes the Sign in Sign Up system for the users. There is a built-in user for Admin. Only fans can sign up for the system while the Players and Coaches are added by the admin. Its functionality also includes the Create, Retrieve, Update, Delete (CRUD) operations. It adds, removes, updates and prints the players, coaches data. It also includes the functionality of adding and showing the statistics of the players. Some of its functions are given below.

1. The system effectively handles data storage and data loading.
2. It creates an appealing user interface for the users.
3. It provides the functionality of scheduling for the players.
4. It also contains the functionality of adding, removing, updating and searching of players and coaches.
5. Finances can be managed for players and coaches.
6. There is only one Admin.
7. No one can sign up as an Admin, Player or Coach.
8. This authority is only for the Admin to register users.
9. Only fans can sign up.

* **Class Responsibility Collaboration Diagrams**



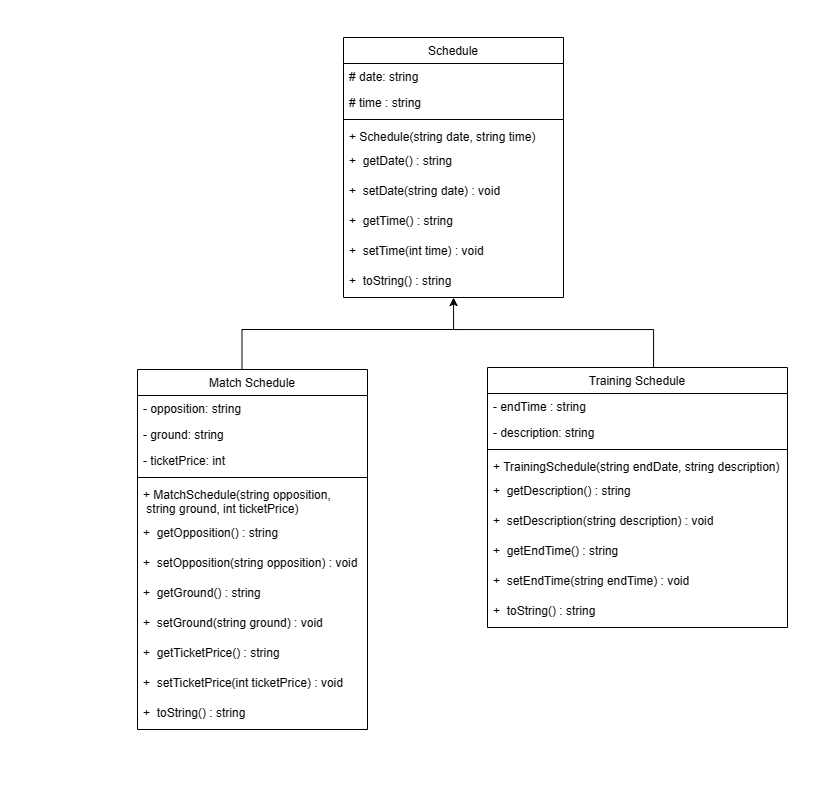


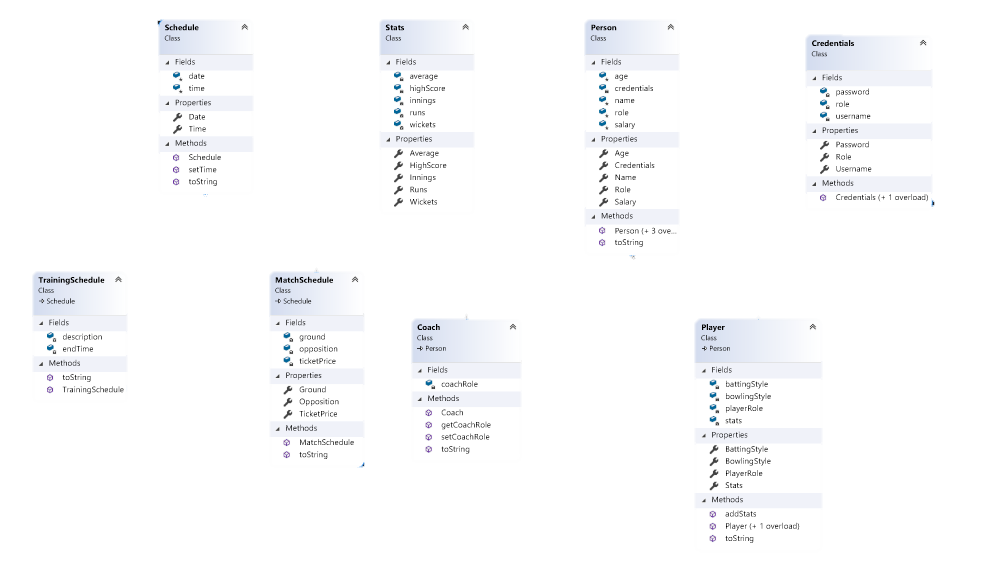










****

contains

contains

* **Object Oriented Programming**

Object-oriented programming (OOP) is a computer-programming model which is based on the concepts of the “objects”. The objects contains data and functions related to a class which is most of the time a real-world entity. 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 defines relation between two classes of 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. Aggregation is a relation in which the contained object can exist without the lifetime of container object while Composition is the opposite of Aggregation as the objects in this type of Association cannot exist without container object’s lifetime. I have used Composition in my project. It is used in two places.

1. Person class have an object of Credentials. It is a relation of one-to-one because a person contains only one username, password and role. Credentials cannot exist without the object of Person, so I used composition for this particular purpose.
2. Player class contains an object of Stats. This association also is of one-to-one relation as a player can contain only one instance of runs, innings, wickets, etc. As the object of Stats should not exist without the existence of Player object, so we used composition here as well.

**Advantage**

If I compare this with my procedural programming project, I can observe that there is a clear advantage of OOP. There was disjoint data of Stats and Credentials in procedural programming which is rectified in OOP. Now the objects of Credentials and Stats are within the objects of Person and Player respectively.

* **Inheritance**

Inheritance is one of the core concepts of Object-oriented programming. It is a feature that allows a new class to be derived from an existing class. The new class inherits all the public or protected attributes and the member functions of the base class. Inheritance is used most of the time to extend the functionality of a base class. I have used this OOP concept in two places in this management system.

1. There is a general class Schedule. Two classes of Match Schedule and Training Schedule. As these two are type of schedule, so the application of inheritance makes sense here.
2. Person class is a parent class. Player and Coach class are derived from this class. Player and Coach is a person. So, that is the reason I have applied the concept of inheritance here.

**Advantage**

Inheritance is very useful concept and is very widely used in practical world. It allows the programmers to extend the functionality of a particular class. This feature can be very useful in writing code as a software engineer. This thing is not possible while writing codes in procedural programming approach.

* **Polymorphism**

Polymorphism is one of the pillars of Object-oriented programming approach. Polymorphism basically is a combination of two words, Poly-many and Morphism-states. It means many states. There are two types of Polymorphism, Dynamic and Static. Dynamic Polymorphism decides at the run-time that which method will be called while static polymorphism decides at the compile time. I have used Dynamic Polymorphism in few places.

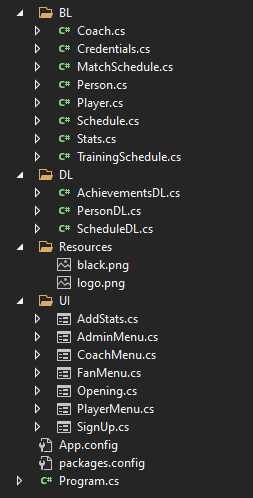
1. toString() functions in both inheritances (Schedule – MatchSchedule & TrainingSchedule, Person – Player & Coach). The function in parent class of Schedule will return the “Schedule” string while it will return “Match Schedule” in class Match Schedule and “Training Schedule” in class Training Schedule. This function in parent class of Person will return the user role which could be “Admin” or “Fan”. But when it is overridden in child classes, it will return the “Player” and “Coach” in their respective classes. This is used for sign in functionality.
2. I have also used this type of Polymorphism in startInterface() function. This function would load the interface of the user according to their role when logged in correctly.

**Advantage**

Polymorphism allowed us to extend the functionalities from the base class to use them for the child classes. It saves time by reusing the code and results in the reduction of code redundancy. It allows different implementations of the same method. Its basic purpose is to make it easier to extend the aspects of program. For example, if we have written a code for three types but after production, we are told by the client that there should be another type, then we can easily maintain and extend the code but it will create a lot of mess 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 Person, Player, Coach, Credentials, Schedule, MatchSchedule, TrainingSchedule, Stats.

* **Data Layer (DL)**

The Data Layer folder contains the static Lists and functions of the project. It contains the list of Achievements for the club, list of Persons which contains the objects of Players and Coaches. 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. This folder also contains a class of Interface. This class includes the static functions of interface which directs the functions depending on the user’s choice.

* **User Interface (UI)**

This folder includes the Graphical User Interface (GUI) of the system. The user will be able to interact with the system using controls efficiently. This also creates a visual representation to the user which creates an appealing scene for him.

* **Resources**

This folder contains the images which are used in the GUI.

* **Class Details**
* **Person**

A person class is used for generalizing the two classes of Player and Coach. It is a parent class of these two classes. It contains the common attributes of the both classes. Its attributes includes the name, role, age and salary of the players and coaches. All its attributes are protected so they can be accessed by the child classes. It also contains an object of Credentials since every person object created will have their own credentials.

* **Player**

This class inherits the attributes and behavior of the parent class Player. Its attributes includes the player role, batting and bowling styles of the player. It contains an object of stats. All the attributes of the player class are private and are accessed by the getter and setter functions.

* **Coach**

Coach class is a child class of Person class. It inherits the attributes and behavior of the parent class. All the attributes of the Coach class are also private. Its attributes include the coach role only.

* **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.

* **Stats**

Stats class contains the attributes of innings, runs, wickets, high score and average. The object of this class is contained inside the object of the Player.

* **Schedule**

This class is the general class for two classes of Match Schedule and Training Schedule. Its attributes are date, time. Its access modifier is protected.

* **Match Schedule**

Match Schedule class is a child class of Schedule. Its attributes are opposition, ground and ticket price.

* **Training Schedule**

Training Schedule class is also a child class of Schedule. Its attributes are end time and description.

* **Code**
  + **Person**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.BL

{

public class Person

{

protected string name;

protected string role;

protected int age;

protected int salary;

private Credentials credentials;

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

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

public int Age { get => age; set => age = value; }

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

public Credentials Credentials { get => credentials; set => credentials = value; }

public Person(string name, string role, int age, int salary, Credentials credentials)

{

this.Name = name;

this.Age = age;

this.Role = role;

this.Salary = salary;

this.Credentials = credentials;

}

public Person(string name, Credentials credentials)

{

this.Name = name;

this.Age = 0;

this.Credentials = credentials;

Role = null;

Salary = 0;

}

public Person(string name, int age, Credentials credentials)

{

this.Name = name;

this.Age = age;

this.Credentials = credentials;

Role = null;

Salary = 0;

}

public Person(string name, int age)

{

this.Name = name;

this.Age = age;

}

public string getName()

{

return Name;

}

public int getAge()

{

return Age;

}

public string getRole()

{

return Role;

}

public int getSalary()

{

return Salary;

}

public void setName(string name)

{

this.Name = name;

}

public void setAge(int age)

{

this.Age = age; }

public void setRole(string role)

{

this.Role = role;

}

public void setSalary(int salary)

{

this.Salary = salary;

}

public Credentials getCredentials()

{

return Credentials;

}

public virtual string toString()

{

return Role;

}

}

}

* + **Player**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.BL

{

public class Player : Person

{

private string battingStyle;

private string bowlingStyle;

private string playerRole;

private Stats stats;

public string BattingStyle { get => battingStyle; set => battingStyle = value; }

public string BowlingStyle { get => bowlingStyle; set => bowlingStyle = value; }

public string PlayerRole { get => playerRole; set => playerRole = value; }

public Stats Stats { get => stats; set => stats = value; }

public Player(string name, int age, string playerRole, string battingStyle, string bowlingStyle, int salary, Credentials credentials) : base(name, credentials.getRole(), age, salary, credentials)

{

this.BattingStyle = battingStyle;

this.BowlingStyle = bowlingStyle;

this.PlayerRole = playerRole; }

public Player(string name, int age, string playerRole, string battingStyle, string bowlingStyle, int salary, Stats stats, Credentials credentials) : base(name, "Player", age, salary, credentials)

{

this.BattingStyle = battingStyle;

this.BowlingStyle = bowlingStyle;

this.Stats = stats;

this.PlayerRole = playerRole;

}

public void addStats(Stats stats)

{

this.Stats = stats;

}

public string getBattingStyle()

{

return BattingStyle;

}

public string getBowlingStyle()

{

return BowlingStyle;

}

public string getPlayerRole()

{

return PlayerRole;

}

public void setBattingStyle(string battingStyle)

{

this.BattingStyle = battingStyle;

}

public void setPlayerRole(string playerRole)

{

this.PlayerRole = playerRole;

}

public void setBowlingStyle(string bowlingStyle)

{

this.BowlingStyle = bowlingStyle;

}

public Stats getStats()

{

return Stats;

}

public override string toString()

{

return "Player";

}

}

}

* + **Coach**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.BL

{

class Coach : Person

{

private string coachRole;

public Coach(string name, string coachRole, int age, int salary, Credentials credentials) : base(name, credentials.getRole(), age, salary, credentials)

{

this.coachRole = coachRole;

}

public void setCoachRole(string coachRole)

{

this.coachRole = coachRole;

}

public string getCoachRole()

{

return coachRole;

}

public override string toString()

{

return "Coach";

}

}

}

* + **Credentials**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buisness\_App\_Final.BL

{

public class Credentials

{

private string username;

private string password;

private string role;

public string Username { get => username; set => username = value; }

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

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

public Credentials(string username, string password)

{

this.Username = username;

this.Password = password;

}

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

{

this.Username = username;

this.Password = password;

this.Role = role;

}

public void setUsername(string username)

{

this.Username = username;

}

public void setPassword(string password)

{

this.Password = password;

}

public void setRole(string role)

{

this.Role = role;

}

public string getUsername()

{

return Username;

}

public string getPassword()

{

return Password;

}

public string getRole()

{

return Role;

}

}

}

* + **Stats**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buisness\_App\_Final.BL

{

public class Stats

{

private int innings;

private int runs;

private int wickets;

private int highScore;

private float average;

public int Innings { get => innings; set => innings = value; }

public int Runs { get => runs; set => runs = value; }

public int Wickets { get => wickets; set => wickets = value; }

public int HighScore { get => highScore; set => highScore = value; }

public float Average { get => average; set => average = value; }

public Stats(int innings, int runs, int wickets, int highScore, float average)

{

this.Innings = innings;

this.Runs = runs;

this.Wickets = wickets;

this.HighScore = highScore;

this.Average = average;

}

public int getInnings()

{

return Innings;

}

public int getRuns()

{

return Runs;

}

public int getWickets()

{

return Wickets;

}

public int getHighScore()

{

return HighScore;

}

public float getAverage()

{

return Average;

}

public void setInnings(int innings)

{

this.Innings = innings;

}

public void setRuns(int runs)

{

this.Runs = runs;

}

public void setWickets(int wickets)

{

this.Wickets = wickets;

}

public void setHighScore(int highScore)

{

this.HighScore = highScore;

}

public void setAverage(float average)

{

this.Average = average;

}

}

}

* + **Schedule**

­­ using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buisness\_App\_Final.BL

{

class Schedule

{

protected string date;

protected string time;

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

public string Time { get => time; set => time = value; }

public Schedule(string date, string time)

{

this.Date = date;

this.Time = time;

}

public void setDate(string date)

{

this.Date = date;

}

public void setTime(string time)

{

this.Time = time;

}

public string getDate()

{

return Date;

}

public string getTime()

{

return Time;

}

public virtual string toString()

{

return "Schedule";

}

}

}

* + **Match Schedule**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buisness\_App\_Final.BL

{

class MatchSchedule : Schedule

{

private string opposition;

private string ground;

private int ticketPrice;

public string Opposition { get => opposition; set => opposition = value; }

public string Ground { get => ground; set => ground = value; }

public int TicketPrice { get => ticketPrice; set => ticketPrice = value; }

public MatchSchedule(string date, string time, string opposition, string ground, int ticketPrice) : base(date, time)

{

this.Opposition = opposition;

this.Ground = ground;

this.TicketPrice = ticketPrice;

}

public void setTicketPrice(int price)

{

this.TicketPrice = price;

}

public int getTicketPrice()

{

return ticketPrice;

}

public void setOpposition(string opposition)

{

this.Opposition = opposition;

}

public void setGround(string ground)

{

this.Ground = ground;

}

public string getOpposition()

{

return opposition;

}

public string getGround()

{

return ground;

}

public override string toString()

{

return "Match " + base.toString();

}

}

}

* + **Training Schedule**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buisness\_App\_Final.BL

{

class TrainingSchedule : Schedule

{

private string endTime;

private string description;

public TrainingSchedule(string date, string time, string endTime, string description) : base(date, time)

{

this.endTime = endTime;

this.description = description;

}

public void setEndTime(string endTime)

{

this.endTime = endTime;

}

public void setDescription(string description)

{

this.description = description;

}

public string getDescription()

{

return description;

}

public string getEndTime()

{

return endTime;

}

public override string toString()

{

return "Training " + base.toString();

}

}

}

* + **Person (DL)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Buisness\_App\_Final.BL;

using System.IO;

namespace Buisness\_App\_Final.DL

{

class PersonDL

{

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

public static void addPlayerToList(Player player)

{

people.Add(player);

}

public static void removePlayer(Player player)

{

people.Remove(player);

}

public static void removePlayer(int index)

{

people.RemoveAt(index);

}

public static void addCoachToList(Coach coach)

{

people.Add(coach);

}

public static void removeCoach(int index)

{

people.RemoveAt(index);

}

public static void removeCoach(Coach coach)

{

people.Remove(coach);

}

public static void addPeopleToList(Person fan)

{

people.Add(fan);

}

public static void addUser(string name, Credentials user)

{

user.setRole("Fan");

Person person = new Person(name, user);

PersonDL.addPeopleToList(person);

}

public static Person login(Credentials user)

{

foreach (Person person in people)

{

if ((user.getUsername() == person.getCredentials().getUsername()) && (user.getPassword() == person.getCredentials().getPassword()))

{

return person;

}

}

return null;

}

public static Person isValidCoach(string name)

{

foreach (Person person in people)

{

if (person.toString() == "Coach")

{

if (person.getName() == name)

{

return person;

}

}

}

return null;

}

public static int isValidCoachAndReturnIndex(string name, string role)

{

for (int i = 0; i < people.Count; i++)

{

if (people[i].toString() == "Coach")

{

Coach coach = (Coach)people[i];

if (people[i].getName() == name && coach.getCoachRole() == role)

{

return i;

}

}

}

return -1;

}

public static Player isValidPlayer(string name)

{

foreach (Person person in people)

{

if (person.toString() == "Player")

{

if (person.getName() == name)

{

return (Player)person;

}

}

}

return null;

}

public static Player isValidPlayer(string name, string role)

{

foreach (Person person in people)

{

if (person.toString() == "Player")

{

Player player = (Player)person;

if (person.getName() == name && player.getPlayerRole() == role)

{

return (Player)person;

}

}

}

return null;

}

public static List<Player> returnPlayingXI()

{

List<Player> players = returnPlayersList();

List<Player> returningList = new List<Player>();

if (players.Count > 10)

{

for (int i = 0; i < 11; i++)

{

returningList.Add(players[i]);

}

}

else

{

for (int i = 0; i < players.Count; i++)

{

returningList.Add(players[i]);

}

}

return returningList;

}

public static int isValidPlayerAndReturnIndex(string name, string role)

{

for (int i = 0; i < people.Count; i++)

{

if (people[i].toString() == "Player")

{

Player player = (Player)people[i];

if (people[i].getName() == name && player.getPlayerRole() == role)

{

return i;

}

}

}

return -1;

}

public static bool changePlayerName(string currentName, string newName)

{

foreach (Person person in people)

{

if (person.toString() == "Player")

{

if (person.getName() == currentName)

{

person.setName(newName);

return true;

}

}

}

return false;

}

public static int sumPlayerSalary()

{

int sum = 0;

foreach (Person person in people)

{

if (person.toString() == "Player")

{

sum += person.getSalary();

}

}

return sum;

}

public static List<Player> returnPlayersList()

{

List<Player> players = new List<Player>();

foreach (Person person in people)

{

if (person.toString() == "Player")

{

players.Add((Player)person);

}

}

return players;

}

public static List<Coach> returnCoachList()

{

List<Coach> coaches = new List<Coach>();

foreach (Person person in people)

{

if (person.toString() == "Coach")

{

coaches.Add((Coach)person);

}

}

return coaches;

}

public static int sumCoachSalary()

{

int sum = 0;

foreach (Person person in people)

{

if (person.toString() == "Coach")

{

sum += person.getSalary();

}

}

return sum;

}

public static Player searchPlayer(string search)

{

for (int idx = 0; idx < people.Count; idx++)

{

if (search == people[idx].getName() && people[idx].toString() == "Player")

{

return (Player)people[idx];

}

}

return null;

}

public static void LoadCricketersData(string path)

{

string line = "";

StreamReader file = new StreamReader(path);

if (File.Exists(path))

{

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

{

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

string name = splittedRecord[0];

string playerRole = splittedRecord[1];

string battingStyle = splittedRecord[2];

string bowlingStyle = splittedRecord[3];

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

int salary = int.Parse(splittedRecord[5]);

string[] splittedRecordForCredentials = splittedRecord[7].Split(';');

string username = splittedRecordForCredentials[0];

string password = splittedRecordForCredentials[1];

string role = splittedRecordForCredentials[2];

Credentials credentials = new Credentials(username, password, role);

if (splittedRecord[6] != "null")

{

string[] splittedRecordForStats = splittedRecord[6].Split(';');

int innings = int.Parse(splittedRecordForStats[0]);

int runs = int.Parse(splittedRecordForStats[1]);

int wickets = int.Parse(splittedRecordForStats[2]);

int highScore = int.Parse(splittedRecordForStats[3]);

float average = float.Parse(splittedRecordForStats[4]);

Stats stats = new Stats(innings, runs, wickets, highScore, average);

Player player = new Player(name, age, playerRole, battingStyle, bowlingStyle, salary, stats, credentials);

people.Add(player);

}

else

{

Player player = new Player(name, age, playerRole, battingStyle, bowlingStyle, salary, credentials);

people.Add(player);

}

}

file.Close();

}

else

{

return;

}

}

public static void changeplayingXI(int currentPlayer, int substitutedPlayer)

{

currentPlayer = returnGeneralListIndex(currentPlayer);

substitutedPlayer = returnGeneralListIndex(substitutedPlayer);

Person temp = people[currentPlayer];

people[currentPlayer] = people[substitutedPlayer];

people[substitutedPlayer] = temp;

}

public static List<Stats> returnStatsList()

{

List<Stats> stats = new List<Stats>();

foreach (Person person in people)

{

if (person.toString() == "Player")

{

Player player = (Player)person;

stats.Add(player.getStats());

}

}

return stats;

}

public static int returnGeneralListIndex(int number)

{

int newIndex = 0;

int check = 0;

for (int i = 0; i < people.Count; i++)

{

if (people[i].toString() == "Player")

{

if (check == number)

{

return newIndex;

}

check++;

}

newIndex++;

}

return newIndex;

}

public static void LoadCoachesData(string path)

{

string line = "";

StreamReader file = new StreamReader(path);

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

{

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

string name = splittedRecord[0];

string coachRole = splittedRecord[1];

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

int salary = int.Parse(splittedRecord[3]);

string[] splittedRecordForCredentials = splittedRecord[4].Split(';');

string username = splittedRecordForCredentials[0];

string password = splittedRecordForCredentials[1];

string role = splittedRecordForCredentials[2];

Credentials credentials = new Credentials(username, password, role);

Coach coach = new Coach(name, coachRole, age, salary, credentials);

people.Add(coach);

}

file.Close();

}

public static void storeCricketersData(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < people.Count(); idx++)

{

string record = "null";

string recordUser;

if (people[idx].toString() == "Player")

{

Player player = (Player)people[idx];

if (player.getStats() != null)

{

Stats stats = player.getStats();

record = stats.getInnings() + ";" + stats.getRuns() + ";" + stats.getWickets() + ";" + stats.getHighScore() + ";" + stats.getAverage();

}

Credentials user = people[idx].getCredentials();

recordUser = user.getUsername() + ";" + user.getPassword() + ";" + user.getRole();

file.WriteLine(people[idx].getName() + "," + player.getPlayerRole() + "," + player.getBattingStyle() + "," + player.getBowlingStyle() + "," + people[idx].getAge() + "," + people[idx].getSalary() + "," + record + "," + recordUser);

}

}

file.Flush();

file.Close();

}

public static void storeCoachesData(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < people.Count(); idx++)

{

string record = "";

if (people[idx].toString() == "Coach")

{

Coach coach = (Coach)people[idx];

Credentials user = people[idx].getCredentials();

record = user.getUsername() + ";" + user.getPassword() + ";" + user.getRole();

file.WriteLine(people[idx].getName() + "," + coach.getCoachRole() + "," + people[idx].getAge() + "," + people[idx].getSalary() + "," + record);

}

}

file.Flush();

file.Close();

}

public static void storePeoplesData(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < people.Count(); idx++)

{

string record = "";

if (people[idx].getCredentials().getRole() == "Fan" || people[idx].getCredentials().getRole() == "Admin")

{

Credentials user = people[idx].getCredentials();

record = user.getUsername() + ";" + user.getPassword() + ";" + user.getRole();

file.WriteLine(people[idx].getName() + "," + people[idx].getAge() + "," + record);

}

}

file.Flush();

file.Close();

}

public static void loadPeoplesData(string path)

{

string line = "";

StreamReader file = new StreamReader(path);

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

{

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

string name = splittedRecord[0];

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

string[] splittedRecordForCredentials = splittedRecord[2].Split(';');

string username = splittedRecordForCredentials[0];

string password = splittedRecordForCredentials[1];

string role = splittedRecordForCredentials[2];

Credentials credentials = new Credentials(username, password, role);

Person person = new Person(name, age, credentials);

people.Add(person);

}

file.Close();

}

}

}

* + **Schedule (DL)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using Buisness\_App\_Final.BL;

namespace Buisness\_App\_Final.DL

{

class ScheduleDL

{

public static List<Schedule> schedules = new List<Schedule>();

public static void addTrainingSchedule(TrainingSchedule trainingSchedule)

{

schedules.Add(trainingSchedule);

}

public static void addMatchSchedule(MatchSchedule matchSchedule)

{

schedules.Add(matchSchedule);

}

public static void removeTrainingSchedule(TrainingSchedule trainingSchedule)

{

schedules.Remove(trainingSchedule);

}

public static void removeMatchSchedule(MatchSchedule matchSchedule)

{

schedules.Remove(matchSchedule);

}

public static List<TrainingSchedule> returnTrainingSchedules()

{

List<TrainingSchedule> trainingSchedules = new List<TrainingSchedule>();

foreach (Schedule schedule in schedules)

{

if (schedule.toString() == "Training Schedule")

{

trainingSchedules.Add((TrainingSchedule)schedule);

}

}

return trainingSchedules;

}

public static List<MatchSchedule> returnMatchSchedules()

{

List<MatchSchedule> matchSchedules = new List<MatchSchedule>();

foreach (Schedule schedule in schedules)

{

if (schedule.toString() == "Match Schedule")

{

matchSchedules.Add((MatchSchedule)schedule);

}

}

return matchSchedules;

}

public static void storeMatchSchedule(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < schedules.Count; idx++)

{

if (schedules[idx].toString() == "Match Schedule")

{

MatchSchedule schedule = (MatchSchedule)schedules[idx];

if (idx == (schedules.Count - 1) && idx != 0)

{

file.WriteLine(schedule.getOpposition() + "," + schedules[idx].getDate() + "," + schedules[idx].getTime() + "," + schedule.getGround() + "," + schedule.getTicketPrice());

}

else

{

file.WriteLine(schedule.getOpposition() + "," + schedules[idx].getDate() + "," + schedules[idx].getTime() + "," + schedule.getGround() + "," + schedule.getTicketPrice());

}

}

}

file.Flush();

file.Close();

}

public static void storeTraining(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < schedules.Count; idx++)

{

if (schedules[idx].toString() == "Training Schedule")

{

TrainingSchedule schedule = (TrainingSchedule)schedules[idx];

if (idx == (schedules.Count - 1) && idx != 0)

{

file.WriteLine(schedules[idx].getDate() + "," + schedules[idx].getTime() + "," + schedule.getEndTime() + "," + schedule.getDescription());

}

else

{

file.WriteLine(schedules[idx].getDate() + "," + schedules[idx].getTime() + "," + schedule.getEndTime() + "," + schedule.getDescription());

}

}

}

file.Flush();

file.Close();

}

public static void loadSchedule(string path)

{

string line = "";

StreamReader file = new StreamReader(path);

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

{

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

string opposition = splittedRecord[0];

string date = splittedRecord[1];

string time = splittedRecord[2];

string ground = splittedRecord[3];

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

MatchSchedule matchSchedule = new MatchSchedule(opposition, date, time, ground, ticketPrice);

schedules.Add(matchSchedule);

}

file.Close();

}

public static void loadTraining(string path)

{

string line = "";

StreamReader file = new StreamReader(path);

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

{

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

string date = splittedRecord[0];

string time = splittedRecord[1];

string endTime = splittedRecord[2];

string description = splittedRecord[3];

TrainingSchedule trainingSchedule = new TrainingSchedule(date, time, endTime, description);

schedules.Add(trainingSchedule);

}

file.Close();

}

}

}

* + **Achievement (DL)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace Buisness\_App\_Final.DL

{

class AchievementsDL

{

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

public static void addAchievements(string achievement)

{

achievements.Add(achievement);

}

public static void removeAchievement(int index)

{

achievements.RemoveAt(index);

}

public static void loadAchievements(string path)

{

StreamReader file = new StreamReader(path);

string line = "";

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

{

achievements.Add(line);

}

file.Close();

}

public static void storeAchievements(string path)

{

StreamWriter file = new StreamWriter(path, false);

for (int idx = 0; idx < achievements.Count; idx++)

{

file.WriteLine(achievements[idx]);

}

file.Flush();

file.Close();

}

public static List<string> returnListOfAchievements()

{

return achievements;

}

}

}

* + **Opening Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.UI;

using Buisness\_App\_Final.BL;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final

{

public partial class Opening : Form

{

public Opening()

{

InitializeComponent();

ControlBox = false;

}

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

{

SignUp i = new SignUp();

i.Show();

this.Hide();

}

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

{

PersonDL.storePeoplesData("data.txt");

PersonDL.storeCricketersData("CricketersData.txt");

PersonDL.storeCoachesData("CoachesData.txt");

ScheduleDL.storeMatchSchedule("Schedule.txt");

ScheduleDL.storeTraining("Training.txt");

AchievementsDL.storeAchievements("Achievements.txt");

this.Close();

}

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

{

Credentials credentials = new Credentials(txtUsername.Text, txtPassword.Text);

Person user = PersonDL.login(credentials);

if (user != null)

{

if (user.getCredentials().getRole() == "Admin")

{

AdminMenu menu = new AdminMenu();

menu.Show();

this.Hide();

}

else if (user.getCredentials().getRole() == "Coach")

{

CoachMenu menu = new CoachMenu();

menu.Show();

this.Hide();

}

else if (user.getCredentials().getRole() == "Player")

{

PlayerMenu menu = new PlayerMenu();

menu.Show();

this.Hide();

}

else if (user.getCredentials().getRole() == "Fan")

{

FanMenu menu = new FanMenu();

menu.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Wrong Credentials");

}

}

}

}

* + **Sign Up Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.BL;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.UI

{

public partial class SignUp : Form

{

public SignUp()

{

InitializeComponent();

ControlBox = false;

}

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

{

Opening i = new Opening();

i.Show();

this.Hide();

}

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

{

if (txtUsername.Text != "" && txtPassword.Text != "")

{

Credentials user = new Credentials(txtUsername.Text, txtPassword.Text, "Fan");

Person people = new Person(txtUsername.Text, user);

PersonDL.addPeopleToList(people);

MessageBox.Show("The user has been added.");

}

if (txtUsername.Text == "" || txtPassword.Text == "")

{

MessageBox.Show("Fill the fields correctly");

}

}

}

}

* + **Admin Menu Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.DL;

using Buisness\_App\_Final.BL;

namespace Buisness\_App\_Final.UI

{

public partial class AdminMenu : Form

{

public AdminMenu()

{

InitializeComponent();

ControlBox = false;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

dataGridViewDeletePlayer.DataSource = PersonDL.returnPlayersList();

dataGridViewDeletePlayer.Refresh();

dataGridViewDeletePlayer.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

dataGridSalariesPlayer.DataSource = null;

dataGridSalariesPlayer.DataSource = PersonDL.returnPlayersList();

dataGridSalariesPlayer.Columns["Stats"].Visible = false;

dataGridSalariesPlayer.Columns["BattingStyle"].Visible = false;

dataGridSalariesPlayer.Columns["BowlingStyle"].Visible = false;

dataGridSalariesPlayer.Columns["Credentials"].Visible = false;

dataGridSalariesCoach.DataSource = PersonDL.returnCoachList();

dataGridSalariesCoach.Columns["Credentials"].Visible = false;

dataGridSalariesPlayer.Refresh();

dataGridSalariesCoach.Refresh();

dataGridSalariesCoach.Visible = true;

dataGridSalariesPlayer.Visible = true;

PlayerSalary.Text = string.Concat(PersonDL.sumPlayerSalary());

CoachesSalary.Text = string.Concat(PersonDL.sumCoachSalary());

}

private void makePanelsVisibilityFalse()

{

panelAddPlayer.Visible = false;

panelRemovePlayer.Visible = false;

panelChkFinance.Visible = false;

panelSearchPlayer.Visible = false;

panelAddCoach.Visible = false;

panelRemoveCoach.Visible = false;

panelAddAchievements.Visible = false;

panelChkTeam.Visible = false;

panelChkMgmt.Visible = false;

panelCallChngName.Visible = false;

panelAddSchedule.Visible = false;

lblOutputAddPlayer.Visible = false;

lblOutputRemovePlayer.Visible = false;

lblOutputSearch.Visible = false;

lblOutputAddCoach.Visible = false;

lblOutputRemoveCoach.Visible = false;

lblOutputAchievement.Visible = false;

lblOutputChangeName.Visible = false;

lblOutputMatchSchedule.Visible = false;

dataGridSalariesCoach.Visible = false;

dataGridSalariesPlayer.Visible = false;

dataGridViewChkMgmt.Visible = false;

dataGridViewChkTeam.Visible = false;

dataGridViewDelCoach.Visible = false;

dataGridViewDeletePlayer.Visible = false;

dataGridViewSearchPlayer.Visible = false;

}

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

{

if (txtAge.Text != "" && txtBatStyle.Text != "" && txtBowlStyle.Text != "" && txtName.Text != "" && txtRole.Text != "" && txtSalary.Text != "" && txtUsername.Text != "" && txtPassword.Text != "")

{

Player player = new Player(txtName.Text, int.Parse(txtAge.Text), txtRole.Text, txtBatStyle.Text, txtBowlStyle.Text, int.Parse(txtSalary.Text), new Credentials(txtUsername.Text, txtPassword.Text, "Player"));

PersonDL.people.Add(player);

lblOutputAddPlayer.ForeColor = Color.Green;

lblOutputAddPlayer.Text = "The player has been added.";

lblOutputAddPlayer.Visible = true;

}

else if (txtAge.Text == "" || txtBatStyle.Text == "" || txtBowlStyle.Text == "" || txtName.Text == "" || txtRole.Text == "" || txtSalary.Text == "" || txtUsername.Text == "" || txtPassword.Text == "")

{

lblOutputAddPlayer.Text = "Enter all the fields.";

lblOutputAddPlayer.ForeColor = Color.Red;

lblOutputAddPlayer.Visible = true;

}

}

private void dataBind(DataGridView dataGridViewDelete)

{

dataGridViewDelete.DataSource = null;

dataGridViewDelete.DataSource = PersonDL.returnPlayersList();

dataGridViewDelete.Refresh();

}

private void dataGridViewDeletePlayer\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

Player user = (Player)dataGridViewDeletePlayer.CurrentRow.DataBoundItem;

PersonDL.removePlayer(user);

dataBind(dataGridViewDeletePlayer);

lblOutputRemovePlayer.Visible = true;

dataGridViewDeletePlayer.Refresh();

}

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

{

List<Player> players = new List<Player>();

Player player;

if (txtSearchPlayerName.Text != "")

{

player = PersonDL.isValidPlayer(txtSearchPlayerName.Text);

players.Add(player);

if (player != null)

{

dataGridViewSearchPlayer.DataSource = players;

dataGridViewSearchPlayer.Columns["Credentials"].Visible = false;

dataGridViewSearchPlayer.Columns["Stats"].Visible = false;

dataGridViewSearchPlayer.Refresh();

dataGridViewSearchPlayer.Visible = true;

lblOutputSearch.Visible = false;

}

else

{

lblOutputSearch.Text = "There is no such player in the team.";

lblOutputSearch.Visible = true;

dataGridViewSearchPlayer.Visible = false;

}

}

if (txtSearchPlayerName.Text == null)

{

lblOutputSearch.Text = "Fill the fields first.";

lblOutputSearch.Visible = true;

dataGridViewSearchPlayer.Visible = false;

}

}

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

{

if (txtAgeAddCoach.Text != "" && txtNameAddCoach.Text != "" && txtRoleAddCoach.Text != "" && txtSalaryAddCoach.Text != "" && txtPasswordAddCoach.Text != "" && txtUsernameAddCoach.Text != "")

{

Coach player = new Coach(txtNameAddCoach.Text, txtRoleAddCoach.Text, int.Parse(txtAgeAddCoach.Text), int.Parse(txtSalary.Text), new Credentials(txtUsername.Text, txtPassword.Text, "Coach"));

PersonDL.people.Add(player);

lblOutputAddCoach.ForeColor = Color.Green;

lblOutputAddCoach.Text = "The coach has been added.";

lblOutputAddCoach.Visible = true;

}

else if (txtAgeAddCoach.Text == "" || txtNameAddCoach.Text == "" || txtRoleAddCoach.Text == "" || txtSalaryAddCoach.Text == "" || txtPasswordAddCoach.Text == "" || txtUsernameAddCoach.Text == "")

{

lblOutputAddCoach.Text = "Enter all the fields.";

lblOutputAddCoach.ForeColor = Color.Red;

lblOutputAddCoach.Visible = true;

}

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

dataGridViewDelCoach.DataSource = PersonDL.returnCoachList();

dataGridViewDelCoach.Refresh();

dataGridViewDelCoach.Visible = true;

}

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

{

Coach user = (Coach)dataGridViewDelCoach.CurrentRow.DataBoundItem;

PersonDL.removeCoach(user);

dataBind(dataGridViewDelCoach);

lblOutputRemoveCoach.Visible = true;

dataGridViewDelCoach.Refresh();

dataGridViewDelCoach.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

}

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

{

if (txtAchievement.Text != "")

{

AchievementsDL.addAchievements(txtAchievement.Text);

lblOutputAchievement.Visible = true;

}

else

{

lblOutputAchievement.Text = "Fill the field correctly.";

lblOutputAchievement.Visible = true;

}

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

panelAddAchievements.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

panelAddAchievements.Visible = true;

panelChkTeam.Visible = true;

dataGridViewChkTeam.DataSource = PersonDL.returnPlayersList();

dataGridViewChkTeam.Columns["Stats"].Visible = false;

dataGridViewChkTeam.Columns["Credentials"].Visible = false;

dataGridViewChkTeam.Refresh();

dataGridViewChkTeam.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

panelAddAchievements.Visible = true;

panelChkTeam.Visible = true;

panelChkMgmt.Visible = true;

dataGridViewChkMgmt.DataSource = PersonDL.returnCoachList();

dataGridViewChkMgmt.Columns["Credentials"].Visible = false;

dataGridViewChkMgmt.Refresh();

dataGridViewChkMgmt.Visible = true;

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

panelAddAchievements.Visible = true;

panelChkTeam.Visible = true;

panelChkMgmt.Visible = true;

panelCallChngName.Visible = true;

}

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

{

if (txtCurrentNameChangeName.Text != "" && txtNewNameChangeName.Text != "")

{

bool flag = PersonDL.changePlayerName(txtCurrentNameChangeName.Text, txtNewNameChangeName.Text);

if (flag == true)

{

lblOutputChangeName.Text = "The name of the player has been changed.";

lblOutputChangeName.ForeColor = Color.Green;

lblOutputChangeName.Visible = true;

}

else

{

lblOutputChangeName.Text = "No such player is found in the team.";

lblOutputChangeName.ForeColor = Color.Red;

lblOutputChangeName.Visible = true;

}

}

else if (txtCurrentNameChangeName.Text == "" || txtNewNameChangeName.Text == "")

{

lblOutputChangeName.Text = "Fill the fields correctly.";

lblOutputChangeName.ForeColor = Color.Red;

lblOutputChangeName.Visible = true;

}

}

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

{

makePanelsVisibilityFalse();

panelAddPlayer.Visible = true;

panelRemovePlayer.Visible = true;

panelChkFinance.Visible = true;

panelSearchPlayer.Visible = true;

panelAddCoach.Visible = true;

panelRemoveCoach.Visible = true;

panelAddAchievements.Visible = true;

panelChkTeam.Visible = true;

panelChkMgmt.Visible = true;

panelCallChngName.Visible = true;

panelAddSchedule.Visible = true;

timePickerMatchSchedule.Format = DateTimePickerFormat.Time;

timePickerMatchSchedule.Format = DateTimePickerFormat.Custom;

timePickerMatchSchedule.CustomFormat = "HH:mm tt";

}

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

{

DateTime currentDate = datePickerMatchSchedule.Value.Date;

string Date = currentDate.ToString("dd-MM-yyyy");

DateTime currentTime = timePickerMatchSchedule.Value;

string Time = currentTime.ToString("HH:mm");

if (txtGroundMatchSchedule.Text != "" && txtOppositionMatchSchedule.Text != "" && txtPriceMatchSchedule.Text != "")

{

MatchSchedule schedule = new MatchSchedule(Date, Time, txtOppositionMatchSchedule.Text, txtGroundMatchSchedule.Text, int.Parse(txtPriceMatchSchedule.Text));

ScheduleDL.schedules.Add(schedule);

lblOutputMatchSchedule.Text = "Schedule Added !";

lblOutputMatchSchedule.ForeColor = Color.Green;

lblOutputMatchSchedule.Visible = true;

}

else if (txtGroundMatchSchedule.Text == "" || txtOppositionMatchSchedule.Text == "" || txtPriceMatchSchedule.Text == "")

{

lblOutputMatchSchedule.Text = "Fill all the fields correctly";

lblOutputMatchSchedule.ForeColor = Color.Red;

lblOutputMatchSchedule.Visible = true;

}

}

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

{

Opening open = new Opening();

open.Show();

this.Hide();

}

}

}

* + **Coach Menu Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.BL;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.UI

{

public partial class CoachMenu : Form

{

public CoachMenu()

{

InitializeComponent();

ControlBox = false;

}

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

{

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

dataGridViewSearchPlayer.DataSource = null;

}

private void makePanelVisibilityFalse()

{

panelSearchPlayer.Visible = false;

panelChkSchedule.Visible = false;

panelChkTeam.Visible = false;

panelChangePlayingXI.Visible = false;

panelChkStats.Visible = false;

panelChkAchvmnts.Visible = false;

panelAddStats.Visible = false;

panelAddTraining.Visible = false;

lblOutputSearch.Visible = false;

lblOutputChangeXI.Visible = false;

lblOutputTrainingSchedule.Visible = false;

dataGridViewSearchPlayer.Visible = false;

dataGridViewMatches.Visible = false;

dataGridViewChkTeam.Visible = false;

dataGridViewChngPlaying11.Visible = false;

dataGridViewStats.Visible = false;

dataGridViewAchievements.Visible = false;

dataGridViewAddStats.Visible = false;

}

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

{

List<Player> players = new List<Player>();

Player player;

if (txtSearchPlayerName.Text != "")

{

player = PersonDL.isValidPlayer(txtSearchPlayerName.Text);

players.Add(player);

if (player != null)

{

dataGridViewSearchPlayer.DataSource = players;

dataGridViewSearchPlayer.Refresh();

dataGridViewSearchPlayer.Visible = true;

lblOutputSearch.Visible = false;

dataGridViewSearchPlayer.Columns["Credentials"].Visible = false;

dataGridViewSearchPlayer.Columns["Stats"].Visible = false;

}

else

{

lblOutputSearch.Text = "There is no such player in the team.";

lblOutputSearch.Visible = true;

dataGridViewSearchPlayer.Visible = false;

}

}

if (txtSearchPlayerName.Text == null)

{

lblOutputSearch.Text = "Fill the fields first.";

lblOutputSearch.Visible = true;

dataGridViewSearchPlayer.Visible = false;

}

}

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

{

List<MatchSchedule> list = ScheduleDL.returnMatchSchedules();

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

dataGridViewMatches.DataSource = null;

dataGridViewMatches.Visible = true;

dataGridViewMatches.DataSource = list;

dataGridViewMatches.Refresh();

}

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

{

dataGridViewChkTeam.DataSource = null;

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

dataGridViewChkTeam.Visible = true;

dataGridViewChkTeam.DataSource = PersonDL.returnPlayersList();

dataGridViewChkTeam.Columns["Credentials"].Visible = false;

dataGridViewChkTeam.Columns["Stats"].Visible = false;

dataGridViewChkTeam.Columns["Name"].DisplayIndex = 0;

dataGridViewChkTeam.Refresh();

}

private void cmdCallChangePlaying11\_Click(object sender, EventArgs e)

{

dataGridViewChngPlaying11.DataSource = null;

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

panelChangePlayingXI.Visible = true;

dataGridViewChngPlaying11.Visible = true;

}

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

{

if (cBCurrentPlayer.SelectedItem != null && cBSubstitution.SelectedItem != null)

{

int currentIndex = int.Parse(cBCurrentPlayer.SelectedItem.ToString()) - 1;

int substitutedIndex = int.Parse(cBSubstitution.SelectedItem.ToString()) - 1;

if (substitutedIndex < PersonDL.returnPlayersList().Count)

{

PersonDL.changeplayingXI(currentIndex, substitutedIndex);

lblOutputChangeXI.Visible = true;

}

else

{

lblOutputChangeXI.Text = "Fill the forms correctly.";

lblOutputChangeXI.ForeColor = Color.Red;

lblOutputChangeXI.Visible = true;

}

}

if (cBCurrentPlayer.SelectedItem == null || cBSubstitution.SelectedItem == null)

{

lblOutputChangeXI.Text = "Fill the forms correctly.";

lblOutputChangeXI.ForeColor = Color.Red;

lblOutputChangeXI.Visible = true;

}

}

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

{

dataGridViewStats.DataSource = null;

dataGridViewStats.Columns.Clear();

dataGridViewStats.Rows.Clear();

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

panelChangePlayingXI.Visible = true;

panelChkStats.Visible = true;

dataGridViewStats.Columns.Add("Name", "Name");

dataGridViewStats.Columns.Add("Innings", "Innings");

dataGridViewStats.Columns.Add("Runs", "Runs");

dataGridViewStats.Columns.Add("Wickets", "Wickets");

dataGridViewStats.Columns.Add("HighScore", "HighScore");

dataGridViewStats.Columns.Add("Average", "Average");

foreach (Player player in PersonDL.returnPlayersList())

{

DataGridViewRow row = new DataGridViewRow();

row.CreateCells(dataGridViewStats);

row.Cells[0].Value = player.Name;

if (player.Stats != null)

{

row.Cells[1].Value = player.Stats.Innings;

row.Cells[2].Value = player.Stats.Runs;

row.Cells[3].Value = player.Stats.Wickets;

row.Cells[4].Value = player.Stats.HighScore;

row.Cells[5].Value = player.Stats.Average;

}

dataGridViewStats.Rows.Add(row);

}

dataGridViewStats.AllowUserToAddRows = false;

dataGridViewStats.Visible = true;

dataGridViewStats.Refresh();

}

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

{

dataGridViewAchievements.DataSource = null;

dataGridViewAchievements.Columns.Clear();

dataGridViewAchievements.Rows.Clear();

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

panelChangePlayingXI.Visible = true;

panelChkStats.Visible = true;

panelChkAchvmnts.Visible = true;

dataGridViewAchievements.Visible = true;

dataGridViewAchievements.Columns.Add("Index", "Index");

dataGridViewAchievements.Columns.Add("Achievement", "Achievement");

for (int i = 0; i < AchievementsDL.returnListOfAchievements().Count; i++)

{

dataGridViewAchievements.Rows.Add(i, AchievementsDL.returnListOfAchievements()[i]);

}

dataGridViewAchievements.AllowUserToAddRows = false;

dataGridViewAchievements.Refresh();

}

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

{

dataGridViewAddStats.DataSource = null;

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

panelChangePlayingXI.Visible = true;

panelChkStats.Visible = true;

panelChkAchvmnts.Visible = true;

panelAddStats.Visible = true;

dataGridViewAddStats.DataSource = PersonDL.returnPlayersList();

dataGridViewAddStats.Columns["Credentials"].Visible = false;

dataGridViewAddStats.Columns["Role"].Visible = false;

dataGridViewAddStats.Columns["Stats"].Visible = false;

dataGridViewAddStats.Columns["Salary"].Visible = false;

dataGridViewAddStats.Columns["Name"].DisplayIndex = 1;

dataGridViewAddStats.Columns["Age"].DisplayIndex = 2;

dataGridViewAddStats.Columns["PlayerRole"].DisplayIndex = 3;

}

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

{

Player user = (Player)dataGridViewAddStats.CurrentRow.DataBoundItem;

AddStats form = new AddStats(user);

form.Show();

}

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

{

DateTime currentDate = datePickerTrainingSchedule.Value.Date;

string Date = currentDate.ToString("dd-MM-yyyy");

DateTime startTime = timePickerStartTrainingSchedule.Value;

string Time = startTime.ToString("HH:mm");

DateTime endTime = timePickerEndSchedule.Value;

string eTime = endTime.ToString("HH:mm");

if (txtDescTrainingSchedule.Text != "")

{

TrainingSchedule schedule = new TrainingSchedule(Date, Time, eTime, txtDescTrainingSchedule.Text);

ScheduleDL.schedules.Add(schedule);

lblOutputTrainingSchedule.Text = "Schedule Added !";

lblOutputTrainingSchedule.ForeColor = Color.Green;

lblOutputTrainingSchedule.Visible = true;

}

else if (txtDescTrainingSchedule.Text == "")

{

lblOutputTrainingSchedule.Text = "Fill all the fields correctly";

lblOutputTrainingSchedule.ForeColor = Color.Red;

lblOutputTrainingSchedule.Visible = true;

}

}

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

{

makePanelVisibilityFalse();

panelSearchPlayer.Visible = true;

panelChkSchedule.Visible = true;

panelChkTeam.Visible = true;

panelChangePlayingXI.Visible = true;

panelChkStats.Visible = true;

panelChkAchvmnts.Visible = true;

panelAddStats.Visible = true;

panelAddTraining.Visible = true;

timePickerStartTrainingSchedule.Format = DateTimePickerFormat.Custom;

timePickerStartTrainingSchedule.CustomFormat = "HH:mm tt";

timePickerEndSchedule.Format = DateTimePickerFormat.Custom;

timePickerEndSchedule.CustomFormat = "HH:mm tt";

}

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

{

Opening open = new Opening();

open.Show();

this.Hide();

}

}

}

* + **Player Menu Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.BL;

using Buisness\_App\_Final.DL;

namespace Buisness\_App\_Final.UI

{

public partial class PlayerMenu : Form

{

public PlayerMenu()

{

InitializeComponent();

ControlBox = false;

}

private void makePanelVisibilityFalse()

{

panelChkSchedule.Visible = false;

panelChk11.Visible = false;

panelChkStats.Visible = false;

panelChkAchievements.Visible = false;

panelChkTeam.Visible = false;

dataGridViewMatches.Visible = false;

dataGridViewChk11.Visible = false;

dataGridViewChkStats.Visible = false;

dataGridViewChkAchievements.Visible = false;

dataGridViewChkTeam.Visible = false;

}

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

{

List<MatchSchedule> list = ScheduleDL.returnMatchSchedules();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

dataGridViewMatches.DataSource = null;

dataGridViewMatches.Visible = true;

dataGridViewMatches.DataSource = list;

dataGridViewMatches.Refresh();

}

private void cmdChkPlaying11\_Click(object sender, EventArgs e)

{

dataGridViewChk11.DataSource = null;

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChk11.Visible = true;

dataGridViewChk11.DataSource = PersonDL.returnPlayingXI();

dataGridViewChk11.Columns["Credentials"].Visible = false;

dataGridViewChk11.Columns["Stats"].Visible = false;

dataGridViewChk11.Columns["Role"].Visible = false;

dataGridViewChk11.Columns["Salary"].Visible = false;

dataGridViewChk11.Columns["Name"].DisplayIndex = 0;

dataGridViewChk11.Columns["PlayerRole"].DisplayIndex = 1;

dataGridViewChk11.Refresh();

dataGridViewChk11.Visible = true;

}

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

{

dataGridViewChkStats.DataSource = null;

dataGridViewChkStats.Columns.Clear();

dataGridViewChkStats.Rows.Clear();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChk11.Visible = true;

panelChkStats.Visible = true;

dataGridViewChkStats.Columns.Add("Name", "Name");

dataGridViewChkStats.Columns.Add("Innings", "Innings");

dataGridViewChkStats.Columns.Add("Runs", "Runs");

dataGridViewChkStats.Columns.Add("Wickets", "Wickets");

dataGridViewChkStats.Columns.Add("HighScore", "HighScore");

dataGridViewChkStats.Columns.Add("Average", "Average");

foreach (Player player in PersonDL.returnPlayersList())

{

DataGridViewRow row = new DataGridViewRow();

row.CreateCells(dataGridViewChkStats);

row.Cells[0].Value = player.Name;

if (player.Stats != null)

{

row.Cells[1].Value = player.Stats.Innings;

row.Cells[2].Value = player.Stats.Runs;

row.Cells[3].Value = player.Stats.Wickets;

row.Cells[4].Value = player.Stats.HighScore;

row.Cells[5].Value = player.Stats.Average;

}

dataGridViewChkStats.Rows.Add(row);

}

dataGridViewChkStats.AllowUserToAddRows = false;

dataGridViewChkStats.Visible = true;

dataGridViewChkStats.Refresh();

}

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

{

dataGridViewChkAchievements.DataSource = null;

dataGridViewChkAchievements.Columns.Clear();

dataGridViewChkAchievements.Rows.Clear();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChk11.Visible = true;

panelChkStats.Visible = true;

panelChkAchievements.Visible = true;

dataGridViewChkAchievements.Visible = true;

dataGridViewChkAchievements.Columns.Add("Index", "Index");

dataGridViewChkAchievements.Columns.Add("Achievement", "Achievement");

for (int i = 0; i < AchievementsDL.returnListOfAchievements().Count; i++)

{

dataGridViewChkAchievements.Rows.Add(i, AchievementsDL.returnListOfAchievements()[i]);

}

dataGridViewChkAchievements.AllowUserToAddRows = false;

dataGridViewChkAchievements.Refresh();

}

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

{

dataGridViewChkTeam.DataSource = null;

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChk11.Visible = true;

panelChkStats.Visible = true;

panelChkAchievements.Visible = true;

panelChkTeam.Visible = true;

dataGridViewChkTeam.Visible = true;

dataGridViewChkTeam.DataSource = PersonDL.returnPlayersList();

dataGridViewChkTeam.Columns["Credentials"].Visible = false;

dataGridViewChkTeam.Columns["Stats"].Visible = false;

dataGridViewChkTeam.Columns["Name"].DisplayIndex = 0;

dataGridViewChkTeam.Refresh();

}

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

{

Opening open = new Opening();

open.Show();

this.Hide();

}

}

}

* + **Fan Menu Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Buisness\_App\_Final.DL;

using Buisness\_App\_Final.BL;

namespace Buisness\_App\_Final.UI

{

public partial class FanMenu : Form

{

public FanMenu()

{

InitializeComponent();

ControlBox = false;

}

private void makePanelVisibilityFalse()

{

panelChkSchedule.Visible = false;

panelChkStats.Visible = false;

panelChkAchievements.Visible = false;

dataGridViewSchedule.Visible = false;

dataGridViewChkStats.Visible = false;

dataGridViewAchievements.Visible = false;

}

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

{

dataGridViewSchedule.DataSource = null;

List<MatchSchedule> list = ScheduleDL.returnMatchSchedules();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

dataGridViewSchedule.Visible = true;

dataGridViewSchedule.DataSource = list;

dataGridViewSchedule.Refresh();

}

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

{

dataGridViewChkStats.DataSource = null;

dataGridViewChkStats.Columns.Clear();

dataGridViewChkStats.Rows.Clear();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChkStats.Visible = true;

dataGridViewChkStats.Columns.Add("Name", "Name");

dataGridViewChkStats.Columns.Add("Innings", "Innings");

dataGridViewChkStats.Columns.Add("Runs", "Runs");

dataGridViewChkStats.Columns.Add("Wickets", "Wickets");

dataGridViewChkStats.Columns.Add("HighScore", "HighScore");

dataGridViewChkStats.Columns.Add("Average", "Average");

foreach (Player player in PersonDL.returnPlayersList())

{

DataGridViewRow row = new DataGridViewRow();

row.CreateCells(dataGridViewChkStats);

row.Cells[0].Value = player.Name;

if (player.Stats != null)

{

row.Cells[1].Value = player.Stats.Innings;

row.Cells[2].Value = player.Stats.Runs;

row.Cells[3].Value = player.Stats.Wickets;

row.Cells[4].Value = player.Stats.HighScore;

row.Cells[5].Value = player.Stats.Average;

}

dataGridViewChkStats.Rows.Add(row);

}

dataGridViewChkStats.AllowUserToAddRows = false;

dataGridViewChkStats.Visible = true;

dataGridViewChkStats.Refresh();

}

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

{

dataGridViewAchievements.DataSource = null;

dataGridViewAchievements.Columns.Clear();

dataGridViewAchievements.Rows.Clear();

dataGridViewAchievements.Refresh();

makePanelVisibilityFalse();

panelChkSchedule.Visible = true;

panelChkStats.Visible = true;

panelChkAchievements.Visible = true;

dataGridViewAchievements.Visible = true;

dataGridViewAchievements.Columns.Add("Index", "Index");

dataGridViewAchievements.Columns.Add("Achievement", "Achievement");

for (int i = 0; i < AchievementsDL.returnListOfAchievements().Count; i++)

{

dataGridViewAchievements.Rows.Add(i, AchievementsDL.returnListOfAchievements()[i]);

}

dataGridViewAchievements.AllowUserToAddRows = false;

dataGridViewAchievements.Refresh();

}

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

{

Opening open = new Opening();

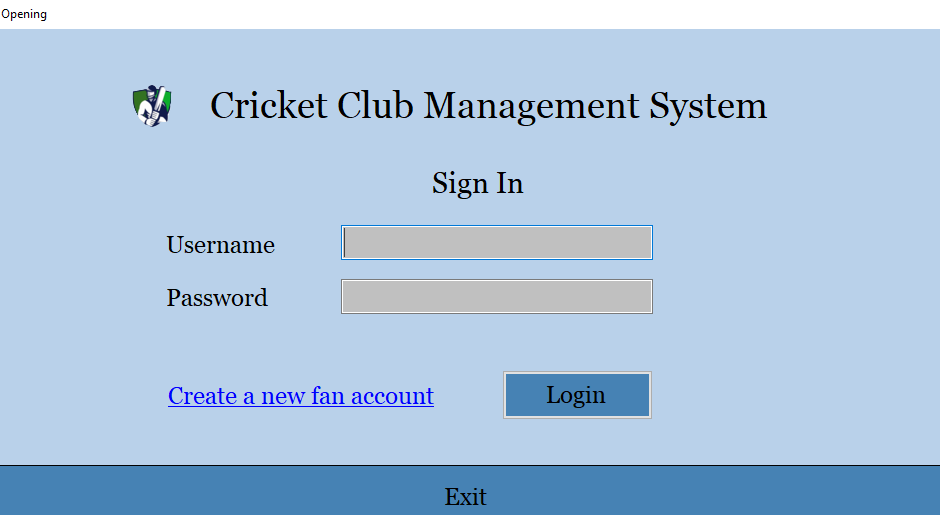
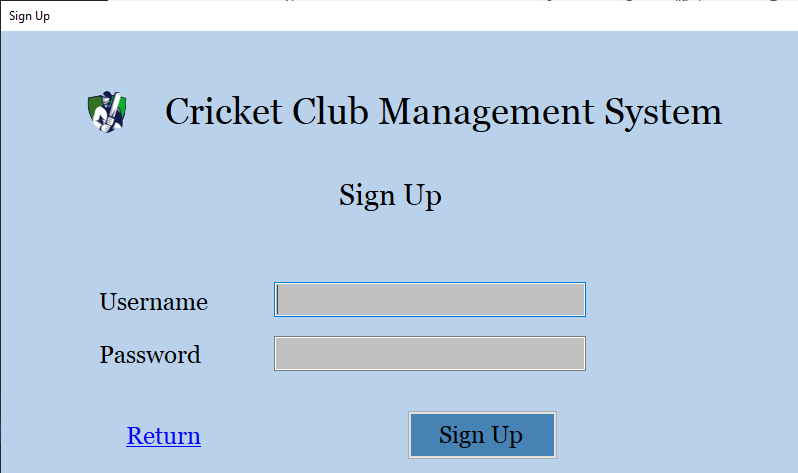
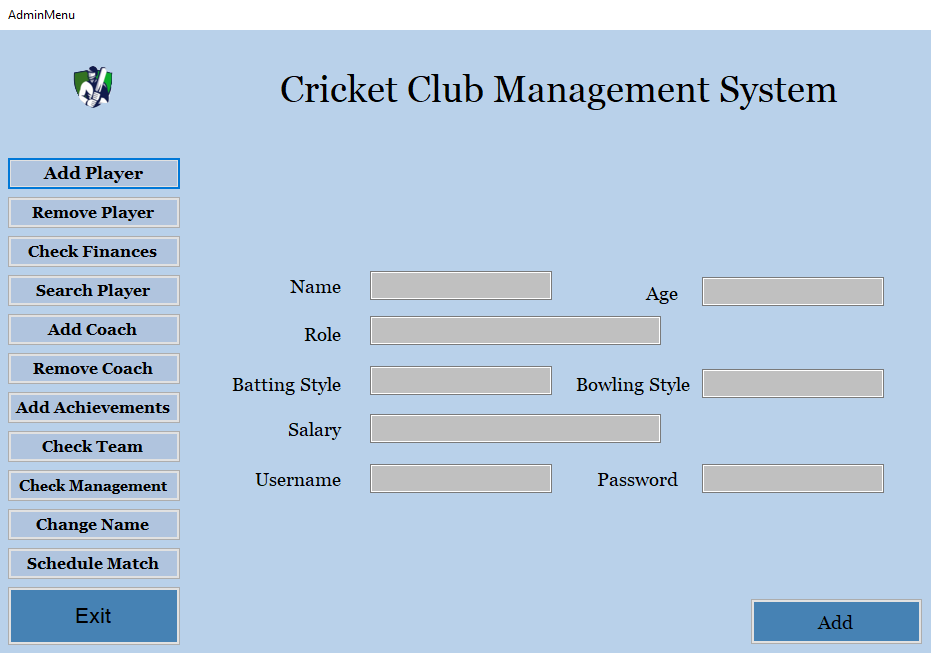
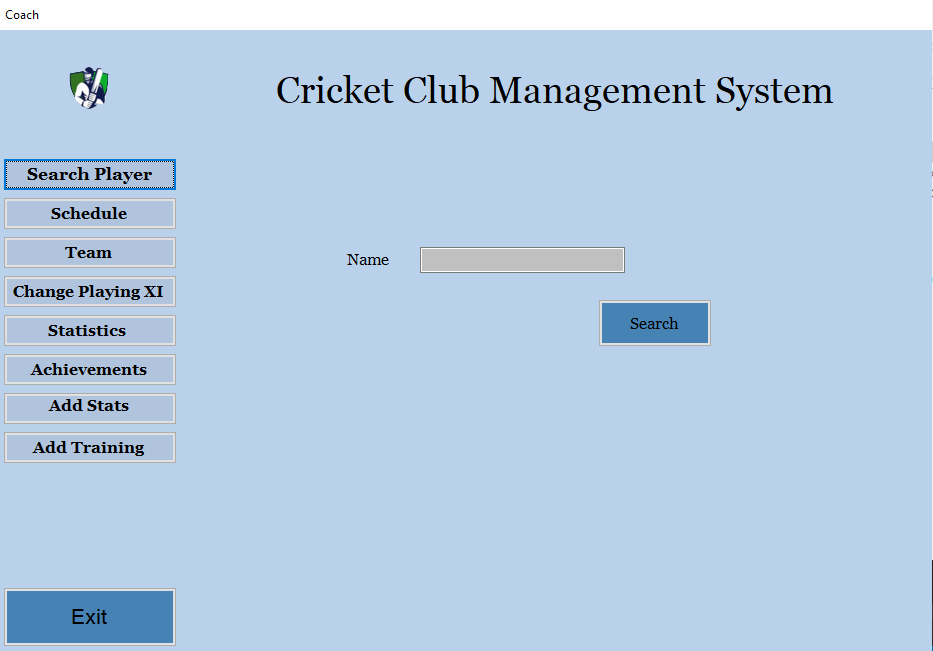
open.Show();

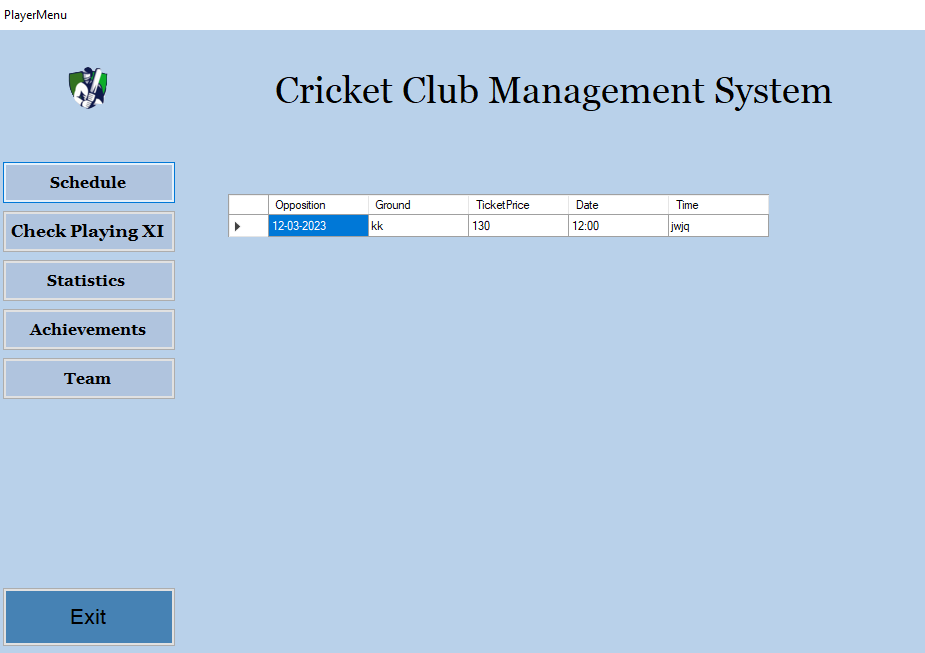
this.Hide();

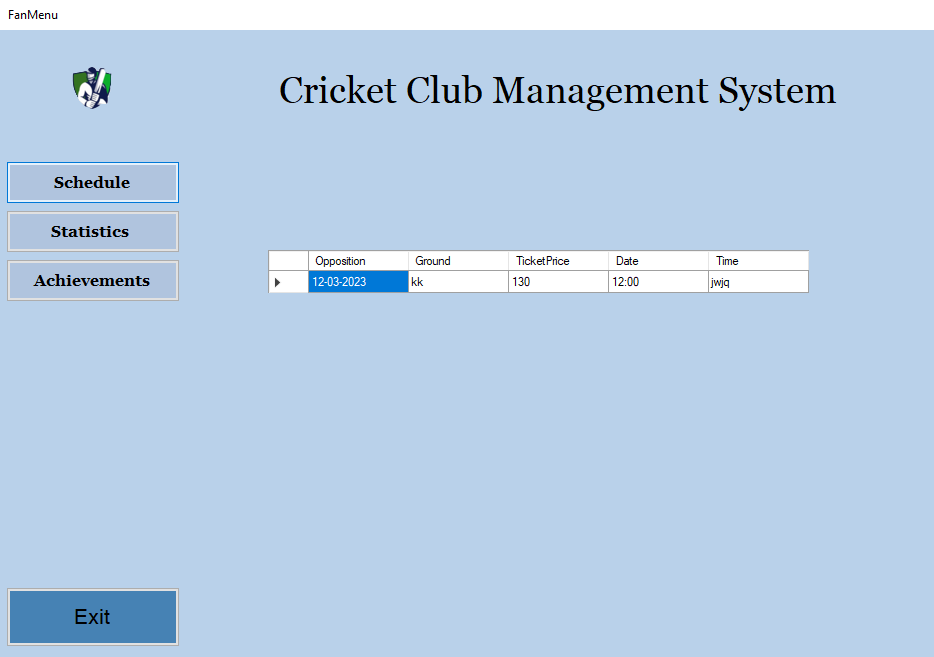
}

}

}

* **Wireframes**
  + **Opening Form**
  + **Sign Up Form**
  + **Admin Form**
  + **Coach Form**
  + **Player Form**

****

* + **Fan Form**
* **Conclusion**

In conclusion, the Cricket Club Management System is built using the object-oriented programming approach. Its key functionalities includes the CRUD operations. Important concepts object-oriented concepts such as association, inheritance and polymorphism are used in this system. I faced several challenges during this phase. I faced difficulty in designing an effective class diagram collaboration model for the system and managing the key concepts of OOP paradigm. Throughout the period of designing and development of this project, I have learned how to create an effective system using object-oriented theory. The object-oriented approach can be really helpful in scaling of the project. It also helps the programmers in future to maintain the software. During the development, I have also learned how to create a well-designed and well-built Graphical User Interface using Window Forms and how to effectively implement them for effortless use of software for the client. In short, this project as a whole has been a great learning curve for us.