```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.OleDb;
using System.Security.Cryptography;

namespace DAL
{
    public static class userFunction
    {
        public static void AddUser(string username, string password)
        {
           string com = "insert into [users] ([username],[password],[isAdmin])
             VALUES ('" + username+"' ,'"+ AesCryp.encrypt(password)+"','n')";
           oledbhelper.Execute(com);
        }
        public static bool isAdmin(string username)
        {
            string com = "SELECT [isAdmin] FROM [users] where [username] = '" +
              username + "'";
            DataTable dt = oledbhelper.GetTable(com);
            if (dt.Rows[0].ItemArray[0].ToString() == "y")
                return true;
            return false;
        }
        public static void toggleAdmin(string username)
        {
            string com = "SELECT [isAdmin] FROM [users] where [username] = '" +
              username + "'";
            DataTable dt = oledbhelper.GetTable(com);
            if (dt.Rows[0].ItemArray[0].ToString() == "y")
            {
                string com2 = $"update [users] set [isAdmin]='n' where [username]
                  ='{username}'";
                oledbhelper.Execute(com2);
            }
            else
            {
                string com2 = $"update [users] set [isAdmin]='y' where [username]
                  ='{username}'";
                oledbhelper.Execute(com2);
            }
        }
        public static bool checkPassword(string username, string password)
        {
            string com = "SELECT [password] FROM [users] where [username] =
              '"+username+"'";
            DataTable dt = oledbhelper.GetTable(com);
            string pass =dt.Rows[0].ItemArray[0].ToString();
```

```
            if (AesCryp.Decrypt(pass) == password)
                return true;
            return false;
        }
        public static bool isUsername(string username)
        {
            string com = "SELECT * FROM [users] where [username] = '"+username
              +"'";
            DataTable dt = oledbhelper.GetTable(com);
            if (dt.Rows.Count == 0)
                return false;
            else
                return true;
        }
        public static void UpdatePassword(string username, string password)
        {
            string com = $"update [users] set [password]='{password}' where
              [username]='{username}'";
            oledbhelper.Execute(com);
        }

        public static void Deleteuser(string username)
        {
            string com = $"DELETE FROM [users] WHERE [username] ='{username}'";
            oledbhelper.Execute(com);
        }
        public static string GetUsers()
        {
            string sql = "Select * from [users]";
            return oledbhelper.printDataTable(sql);
        }
    }
}
```

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

namespace DAL
{
    public static class predictor
    {
        public static void predict (List<prediction> ls)
        {
            foreach (prediction pred in ls)
            {
                int hplay = FPLFunctions.getNumHPlayByTeamId(1);
                int hGF = FPLFunctions.getGFHByTeamId(1);
                int hGA = FPLFunctions.getGAHByTeamId(1);
                float hAttStrength = hGF / hplay;
                Console.WriteLine(hAttStrength);
            }
        }
    }
}

```

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

namespace DAL
{
    public class prediction
    {
        public int gameID { get; set; }
        public int hteam { get; set; }
        public int ateam { get; set; }

        public int hwin { get; set; }
        public int draw { get; set; }
        public int awin { get; set; }

        public prediction(int gameID ,int hteam, int ateam)
        {
            this.gameID = gameID;
            this.hteam = hteam;
            this.ateam = ateam;
        }
    }
}

```

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

namespace DAL
{
    public static class PackFunctions
    {
        public static string[] getByPackId(int Id)
        {
            string com = "SELECT * FROM [pack] where [packID] = " + Id;
            DataTable dt = oledbhelper.GetTable(com);
            int itmLength = dt.Rows[0].ItemArray.Length;
            string[] mrow = new string[itmLength - 1];
            for (int i = 1; i < itmLength; i++)
            {
                mrow[i - 1] = dt.Rows[0].ItemArray[i].ToString();
            }
            return mrow;
        }
        public static void addPack(string username, int packID)
        {
            if (packID == 0)
                packID = 1;
            string com = "insert into [packinventory] ([username],[PackID]) VALUES ('" + username + "' ," + packID + ")";
            oledbhelper.Execute(com);
        }
        public static List<int> packsByUsername(string username)
        {
            List<int> ret = new List<int>();
            string com = "SELECT * FROM [packinventory] where [username] = '" + username+"'";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                ret.Add(Convert.ToInt32(dr.ItemArray[2]));
            }
            return ret;
        }
        public static void deletePack(string username, int packID)
        {
            string com = "SELECT * FROM [packinventory] where [username]='" + username + "' AND [PackID] = " + packID + "";
            string com2 = "DELETE * FROM [packinventory] where [ID] =" +Convert.ToInt32(oledbhelper.GetTable(com).Rows[0].ItemArray[0]);
            oledbhelper.Execute(com2);
        }
    }
```

```
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Data;
6  using System.Data.OleDb;
7  using System.Web;
8
9  public static class oledbhelper
10 {
11     static OleDbConnection cn = new OleDbConnection(ConnectionString);
12
13     public static string ConnectionString
14     {
15         get
16         {
17             string path = @"C:\DB\football.accdb";
18             return @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + path;
19         }
20     }
21
22     //שאילתות עדכון מבנה ה -Database מחיקה והוספת רשומות, ושאילתות עדכון.
23
24     public static void Execute(string com)
25     {
26         //Connection  יצירת אובייקט מסוג
27         //OleDbConnection cn = new OleDbConnection(ConnectionString);
28         //cn.Open();
29
30         if (cn.State != ConnectionState.Open)
31         {
32             cn.Open();
33         }
34         // command יצירת אובייקט מסוג
35         OleDbCommand command = new OleDbCommand();
36         command.Connection = cn;
37         command.CommandText = com;
38
39         try
40         {
41             command.ExecuteNonQuery();
42         }
43         catch (Exception)
44         {
45             throw;
46         }
47
48     }
49     public static DataTable GetTable(string com)
50     {
51         //Connection  יצירת אובייקט מסוג
52         OleDbConnection cn = new OleDbConnection(ConnectionString);
```

```
        // command יצירת אובייקט מסוג
        OleDbCommand command = new OleDbCommand();
        command.Connection = cn;
        command.CommandText = com;

        //יצירת אובייקט מסוג דטהסט - אוסף טבלאות בזיכרון
        DataTable dt = new DataTable();
        dt.TableName = "tbl";

        //יצירת אובייקט אדפטר מטרתו לתאם בין הדטהסט לדטהבייס
        OleDbDataAdapter adapter = new OleDbDataAdapter(command);

        cn.Open();

        try
        {
            // הפעולה פותחת את הדטהבייס ומחזירה את כל הנתונים לתוך טבלה חדשה בדטהסט

            adapter.Fill(dt);
        }
        catch (Exception e)
        {
            throw;
        }
        finally
        {
            cn.Close();
        }

        return dt;
    }
    public static string printDataTable(string sql)
    {    // הפעולה המקבלת מסד נתונים ושאילתה
        // HTML הפעולה מחזירה מחרוזת השומרת את הטבלה בתור
        DataTable dt = GetTable(sql);

        string printStr = "<table border='1'>";
        printStr += "<tr><td></td><td name='test'>name</td><td>Password</td><td>is admin</td> </tr>";
        foreach (DataRow row in dt.Rows)
        {
            printStr += "<tr>";
            printStr += "<td><input type='radio' name='key' value='" + row.ItemArray[0] + "'>";
            foreach (object myItemArray in row.ItemArray)
            {

                printStr += "<td>" + myItemArray.ToString() + "</td>";
            }
            printStr += "</tr>";
        }
```

```
        printStr += "</table>";

        return printStr;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DAL.apiClases;
using DAL;
using System.Web.UI.WebControls;

namespace DAL
{
    public static class GlobalFunctions
    {
        public static string createCard(Card player, clubColour clr, Element els)
        {
            //create card
            string card = "<div class='card'>";
            card += "<div class='card__inner'>";



            //front
            card += "<div class='card__face card__face--front' style='background-color:" + clr.mcolour + "'>";

            //create Rating text
            card += "<div class='rating'><p>" + player.rating + "</p></div>";

            //add gradient
            card += "<div class='gradient' style='background: linear-gradient(0deg, " + clr.mcolour + " 0%, " + clr.mcolour + "CC 60%," + clr.mcolour + "00 100%);'></div>";

            //add player img
            card += "<img class='player' src='" + player.img + "'>";

            //add badge
            card += "<div class='badgecont'>";
            card += "<img class='Cbadge' src='images/badges/" + player.club + ".png'>";
            card += "</div>";

            //add name
            card += "<div class='namecont'>";
            string[] namesplit = player.name.Split(' ');
            try { card += "<p class='fname'>" + namesplit[namesplit.Length - 2] + "</p>"; } catch { card += "<p class='fname' style='visibility:hidden;'>aa</p>"; }
            card += "<p class='lname' style='color:" + clr.scolour + "'>" + namesplit[namesplit.Length - 1] + "</p>";
            card += "</div>";

```

```
            card += "</div>";


            //back of card
            card += "<div class='card__face card__face--back' style='background-color:" + clr.mcolour + "'>";
            card += "<div class='mins'><p style='color:" + clr.scolour + "'>Minutes: " + els.minutes + "</p></div>";
            if (player.pos != "Goalkeeper")
            {
                card += "<div class='goals'><p style='color:" + clr.scolour + "'>Goals scored: " + els.goals_scored + "</p></div>";
                card += "<div class='assits'><p style='color:" + clr.scolour + "'>Assits: " + els.assists + "</p></div>";

            }
            else
            {
                card += "<div class='saves'><p style='color:" + clr.scolour + "'>saves: " + els.saves + "</p></div>";
                card += "<div class='cleansheet'><p style='color:" + clr.scolour + "'>clean sheet: " + els.clean_sheets + "</p></div>";
                card += "<div class='yellow'><p style='color:" + clr.scolour + "'>yellow cards: " + els.yellow_cards + "</p></div>";
                card += "<div class='red'><p style='color:" + clr.scolour + "'>red cards: " + els.red_cards + "</p></div>";
            }
            card += "</div>";


            //end
            card += "</div>";
            card += "</div>";



            return card;
        }
        public static string createClubPrec(string[] a, int curnum, clubColour clr)
        {

            string stylecalc = "stroke-dashoffset:calc(440 - (440 * "+ (100 * curnum)/ Convert.ToInt32(a[1]) + ") / 100); stroke:"+clr.mcolour;


            string club = "<div class='precent'>";
            club += "<svg>";
            club += "<circle style='"+stylecalc+"' cx='70' cy='70' r='70'></circle>";
            club += "<circle style='" + stylecalc + "' cx='70' cy='70' r='70'></
```

```
                circle>";
            club += "</svg>";
            club += "<div class='clubbdg'>";
            club+= "<img src='images/badges/" + a[0] + ".png'>";
            club += "</div>";
            club += "</div>";
            club += "<h2>"+ (100 * curnum) / Convert.ToInt32(a[1]) + "%</h2>";


            return club;
        }
        public static string CreateGame(Root game, Dictionary<int,string> clubs)
        {
            //creating vairables
            string homeT = clubs[game.team_h];
            string awayT = clubs[game.team_a];

            DateTime time = game.kickoff_time ?? DateTime.Now.AddYears (-19999999);

            int? homeS = 0;
            if (game.team_h_score != null)
                homeS = game.team_h_score;
            int? awayS = 0;
            if (game.team_a_score != null)
                awayS = game.team_a_score;



            //create fixture
            string fixture = "<div class='row'>";

            fixture += "<div class='col - sm'>";
            fixture += "<img src='images/badges/" + homeT + ".png'>";
            fixture += "</div>";

            fixture += "<div class='col - sm'>";
            fixture += "<div class='row'>";
            fixture += "<div class='col - sm'>";
            fixture += "<p>"+ homeT + "</p>";
            fixture += "</div>";
            fixture += "<div class='col - sm'>";
            if (game.started == true)
            {
                fixture += "<p> (" + homeS + " - " + awayS + ") </p>";
            }
            else
            {
                fixture += "<p> "+ time.ToString("dddd, dd MMMM h:mm tt") +" </p>";
                fixture+= "<asp:Button runat='server' class='btn btn-primary' id='" + game.id + "' Text='bet' OnClick='bet_click' />";
```

```
            }
            fixture += "</div>";
            fixture += "<div class='col - sm'>";
            fixture += "<p>" + awayT + "</p>";
            fixture += "</div>";
            fixture += "</div>";
            fixture += "</div>";

            fixture += "<div class='col - sm'>";
            fixture += "<img src='images/badges/" + awayT + ".png'>";
            fixture += "</div>";

            fixture += "</div>";
            return fixture;
        }
    }
}

```

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

namespace DAL
{
    public static class GameFunctions
    {
        public static List<prediction> byGameweek(int gameweek)
        {
            string com = "SELECT * FROM [game] where [gw] = '" + (gameweek) + "'";
            DataTable dt = oledbhelper.GetTable(com);
            List<prediction> ls = new List<prediction>();
            Console.WriteLine(dt.Rows.Count);
            foreach (DataRow row in dt.Rows)
            {
                ls.Add(new prediction(Convert.ToInt32(row.ItemArray[0].ToString
                  ()), Convert.ToInt32(row.ItemArray[1].ToString()),
                  Convert.ToInt32(row.ItemArray[2].ToString())));
            }
            return ls;
        }
        public static void addPrecent(double hteam, double draw, double ateam, int
           gameID)
        {
            string com = "UPDATE game SET [hteamW] = " + Convert.ToInt32(hteam) +
              ", [draw] = " + Convert.ToInt32(draw) + ", [ateamW] =" +
              Convert.ToInt32(ateam) + " Where gameID = " + gameID;
            oledbhelper.Execute(com);
        }
        public static int getPrecentbyMatchID(int gameID, int isH)
        {
            string com = "SELECT * FROM [game] where [gameID] = " + gameID;
            DataTable dt = oledbhelper.GetTable(com);
            int[] ret = new int[3];
            ret[0] = Convert.ToInt32(dt.Rows[0].ItemArray[7]);
            ret[1] = Convert.ToInt32(dt.Rows[0].ItemArray[8]);
            ret[2] = Convert.ToInt32(dt.Rows[0].ItemArray[9]);
            return ret[isH];
        }

    }
}

```

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

namespace DAL
{
    public class FPLFunctions
    {
        public static string getByClubID(int ID)
        {
            string com = "SELECT * FROM [Clubs] where [ID] = "+ID;
            DataTable dt = oledbhelper.GetTable(com);
            string ret = dt.Rows[0].ItemArray[1].ToString();
            return ret;
        }
        public static Dictionary<int, string> getdicOfClubs()
        {
            Dictionary<int, string> clubs = new Dictionary<int, string>();
            string com = "SELECT * FROM [Clubs]";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                clubs.Add(Convert.ToInt32(dr.ItemArray[0].ToString()),
                  dr.ItemArray[1].ToString());
            }
            return clubs;
        }
        public static string getByCardID(int ID)
        {
            string com = "SELECT * FROM [card] where [CardID] = " + ID;
            DataTable dt = oledbhelper.GetTable(com);
            return dt.Rows[0].ItemArray[1].ToString();
        }
        public static int getGFHByTeamId(int ID)
        {
            string com = "SELECT homes FROM [game] where [hteam] = '" + ID + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int getGFAByTeamId(int ID)
        {
```

```
            string com = "SELECT ascore FROM [game] where [ateam] = '" + ID + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int getGAHByTeamId(int ID)
        {
            string com = "SELECT ascore FROM [game] where [hteam] = '" + ID + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int getGAAAByTeamId(int ID)
        {
            string com = "SELECT homes FROM [game] where [ateam] = '" + ID + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int getNumHPlayByTeamId(int ID)
        {
            string com = "SELECT homes FROM [game] where [hteam] = '" + ID + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total++;
```

```
                }
            }
            return total;
        }
        public static int getNumAPlayByTeamId(int ID)
        {
            string com = "SELECT ascore FROM [game] where [ateam] = '" + ID +
              "'";
            DataTable dt = oledbhelper.GetTable(com);
            return dt.Rows.Count;
        }
        public static int getallH()
        {
            string com = "SELECT homes FROM [game]";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int getallA()
        {
            string com = "SELECT ascore FROM [game]";
            DataTable dt = oledbhelper.GetTable(com);
            int total = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (dr.ItemArray[0].ToString() != "")
                {
                    total += Convert.ToInt32(dr.ItemArray[0].ToString());
                }
            }
            return total;
        }
        public static int numofgames()
        {
            string com = "SELECT ascore FROM [game]";
            DataTable dt = oledbhelper.GetTable(com);
            return dt.Rows.Count;
        }
        public static void numofplayers()
        {
            string com = "SELECT name FROM [Clubs]";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                string com2 = "SELECT CardID FROM [card] where [club]= '"+
```

```
                    dr.ItemArray[0].ToString()+"'";
                DataTable dt2 = oledbhelper.GetTable(com2);


                string com3 = "UPDATE [Clubs] Set [Oplayers] = '" +
                    dt2.Rows.Count + "' Where [name] = '"+ dr.ItemArray[0].ToString
                    () + "'";
                oledbhelper.Execute(com3);
            }
        }
    }
}

```

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

namespace DAL
{
    public class clubColour
    {
        public string mcolour { get; set; }
        public string scolour { get; set; }
        public clubColour(string mcolour, string scolour)
        {
            this.mcolour = mcolour;
            this.scolour = scolour;
        }
    }
}

```

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

namespace DAL
{
    public static class cardInv
    {
        public static void Addplayer(string username, int cardId)
        {
            string com = "insert into [cardinventory] ([username],[cardID]) VALUES ('" + username + "' ," + cardId + ")";
            oledbhelper.Execute(com);
        }
        public static bool checkDuplicate(string username, int cardId)
        {
            string com = "SELECT [cardID] FROM [cardinventory] where [username] = '" + username+ "' AND [cardID] = "+cardId+"";
            DataTable dt = oledbhelper.GetTable(com);
            if (dt.Rows.Count <= 0)
                return false;
            return true;
        }
        public static string getAllcardId(string username)
        {
            string mrow = "";
            string com = "SELECT [cardID] FROM [cardinventory] where [username] = '" + username + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int itmLength = dt.Rows.Count;
            for (int i = 0; i < itmLength; i++)
            {
                mrow += dt.Rows[i].ItemArray[0].ToString();
                if (i != itmLength - 1)
                    mrow += ",";
            }
            return mrow;
        }
    }
}

```

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

namespace DAL
{
    public class CardFunctions
    {
        public static List<Card> getByClub(string club)
        {
            List<Card> ret = new List<Card>();
            string com = "SELECT * FROM [card] where [club] = '" + club + "'";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                Card c = new Card(Convert.ToInt32(dr.ItemArray[0].ToString()),
                  dr.ItemArray[1].ToString(), dr.ItemArray[2].ToString(),
                  dr.ItemArray[3].ToString(), dr.ItemArray[4].ToString(),
                  Convert.ToInt32(dr.ItemArray[5].ToString()), dr.ItemArray
                  [6].ToString(), dr.ItemArray[7].ToString());

                ret.Add(c);
            }
            return ret;
        }
        public static List<string> getByClubNames(string club)
        {
            List<string> ret = new List<string>();
            string com = "SELECT * FROM [card] where [club] = '" + club + "'";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                ret.Add(dr.ItemArray[1].ToString());
            }
            return ret;
        }
        public static List<Card> getByNation(string nation)
        {
            List<Card> ret = new List<Card>();
            string com = "SELECT * FROM [card] where [country] = '" + nation +
              "'";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                Card c = new Card(Convert.ToInt32(dr.ItemArray[0].ToString()),
                  dr.ItemArray[1].ToString(), dr.ItemArray[2].ToString(),
                  dr.ItemArray[3].ToString(), dr.ItemArray[4].ToString(),
                  Convert.ToInt32(dr.ItemArray[5].ToString()), dr.ItemArray
                  [6].ToString(), dr.ItemArray[7].ToString());
```

```
                ret.Add(c);
            }
            return ret;
        }
        public static Card getByRating(int ratinglow, int ratinghigh)
        {
            string com = "SELECT * FROM [card] where [rating] >= " + ratinglow + " and [rating] <=" + ratinghigh;
            DataTable dt = oledbhelper.GetTable(com);
            Random rnd = new Random();
            int num = rnd.Next(0, dt.Rows.Count);
            DataRow dr = dt.Rows[num];
            int itemNum = dr.ItemArray.Length;
            Card c = new Card(Convert.ToInt32(dr.ItemArray[0].ToString()), dr.ItemArray[1].ToString(), dr.ItemArray[2].ToString(), dr.ItemArray[3].ToString(), dr.ItemArray[4].ToString(), Convert.ToInt32(dr.ItemArray[5].ToString()), dr.ItemArray[6].ToString(), dr.ItemArray[7].ToString());
            return c;
        }
        public static Card getByCardId(int id)
        {
            string com = "SELECT * FROM [card] where [cardID] = " + id;
            DataTable dt = oledbhelper.GetTable(com);
            DataRow dr = dt.Rows[0];
            int itemNum = dr.ItemArray.Length;
            Card c = new Card(Convert.ToInt32(dr.ItemArray[0].ToString()), dr.ItemArray[1].ToString(), dr.ItemArray[2].ToString(), dr.ItemArray[3].ToString(), dr.ItemArray[4].ToString(), Convert.ToInt32(dr.ItemArray[5].ToString()), dr.ItemArray[6].ToString(), dr.ItemArray[7].ToString());
            return c;
        }
        public static List<Card> getALLByCardId(string Id)
        {
            try
            {
                string com = "SELECT * FROM [card] where [cardID] IN " + Id;
                DataTable dt = oledbhelper.GetTable(com);
                List<Card> lc = new List<Card>();
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    DataRow dr = dt.Rows[i];
                    int itemNum = dr.ItemArray.Length;
                    lc.Add(new Card(Convert.ToInt32(dr.ItemArray[0].ToString()), dr.ItemArray[1].ToString(), dr.ItemArray[2].ToString(), dr.ItemArray[3].ToString(), dr.ItemArray[4].ToString(), Convert.ToInt32(dr.ItemArray[5].ToString()), dr.ItemArray[6].ToString(), dr.ItemArray[7].ToString()));
                }
                return lc;
```

```
            }
            catch { return new List<Card>(); }
        }
        public static Dictionary<string, clubColour> getcolours()
        {
            var dic =new Dictionary<string, clubColour>();
            string com = "SELECT * FROM [Clubs]";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                dic.Add(dr.ItemArray[1].ToString(),new clubColour(dr.ItemArray[2].ToString(), dr.ItemArray[3].ToString()));
            }
            return dic;
        }
        public static List<string[]> getClubsTotal()
        {
            var li = new List<string[]>();
            string com = "SELECT * FROM [Clubs]";
            DataTable dt = oledbhelper.GetTable(com);
            foreach (DataRow dr in dt.Rows)
            {
                li.Add(new string[] {dr.ItemArray[1].ToString(),dr.ItemArray[4].ToString() });
            }
            return li;
        }
    }
}
```

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

namespace DAL
{
    public class Card
    {
        public int id { get; set; }
        public string name { get; set; }
        public string img { get; set; }
        public string country { get; set; }
        public string club { get; set; }
        public int rating { get; set; }
        public string pos { get; set; }
        public string type { get; set; }
        public Card()
        {

        }
        public Card(int id, string name, string img, string country, string club, int rating, string pos, string type)
        {
            this.id = id;
            this.name = name;
            this.img = img;
            this.country = country;
            this.club = club;
            this.rating = rating;
            this.pos = pos;
            this.type = type;
        }
    }
}

```

```
using DAL.apiClases;
using System;
using System.Collections.Generic;
using System.Data;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DAL
{
    public static class BettingFunctions
    {
        public static void AddBet(string username, int GameID, string winner, string score, string scorer)
        {
            string com = "insert into [bets] ([username],[GameID],[winner],[score],[scorer],[didClaim]) VALUES ('" + username + "' ," + GameID + " ,'" + winner + "' ,'" + score + "' ,'" + scorer + "', 0)";
            oledbhelper.Execute(com);
        }
        public static bool didBet(string username, int GameID)
        {
            string com = "SELECT [GameID] FROM [bets] where [username] = '" + username + "' AND [GameID] = " + GameID + "";
            DataTable dt = oledbhelper.GetTable(com);
            if (dt.Rows.Count <= 0)
                return false;
            return true;
        }
        public static void claimed(string username, int GameID)
        {
            string com = "UPDATE [bets] Set [didClaim] = 1 WHERE [username] = '" + username+ "' AND [GameID]= "+ GameID +"";
            oledbhelper.Execute(com);
        }
        public static List<Bet> getAllBets(string username)
        {
            string com = "SELECT * FROM [bets] where [username] = '" + username + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int itmLength = dt.Rows.Count;
            List<Bet> lb = new List<Bet>();
            for (int i = 0; i < itmLength; i++)
            {
                Bet b = new Bet(dt.Rows[i].ItemArray[2].ToString(), dt.Rows[i].ItemArray[3].ToString(), dt.Rows[i].ItemArray[4].ToString(), dt.Rows[i].ItemArray[5].ToString(), Convert.ToInt32(dt.Rows[i].ItemArray[6]));
                lb.Add(b);
            }
            return lb;
```

```
        }
        public static List<Root> getAllGamestoPage(string gw)
        {
            string com = "SELECT * FROM [game] where [gw] = '" + gw + "'";
            DataTable dt = oledbhelper.GetTable(com);
            int itmLength = dt.Rows.Count;
            List<Root> lb = new List<Root>();
            for (int i = 0; i < itmLength; i++)
            {
                //CultureInfo provider = new CultureInfo("en_US");
                DateTime date = DateTime.ParseExact((dt.Rows[i].ItemArray
                  [3]).ToString(), "EEEE, dd MMMM h:mm a", null); //provider);
                Root r = new Root();
                r.id = Convert.ToInt32((dt.Rows[i].ItemArray[0]).ToString());
                r.team_h = Convert.ToInt32((dt.Rows[i].ItemArray[1]).ToString());
                r.team_a = Convert.ToInt32((dt.Rows[i].ItemArray[2]).ToString());
                lb.Add(r);
            }
            return lb;
        }
    }
}
```

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

namespace DAL
{
    public class Bet
    {
        public string gameId { get; set; }
        public string winner { get; set; }
        public string score { get; set; }
        public string scorer { get; set; }
        public bool didClaim { get; set; }
        public Bet(string gameID, string winner, string score, string scorer, int didClaim)
        {
            this.gameId = gameID;
            this.winner = winner;
            this.score = score;
            this.scorer = scorer;
            if (didClaim == 0)
                this.didClaim = false;
            else
                this.didClaim = true;
        }

    }
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Net;
6  using System.Text;
7  using System.Threading.Tasks;
8  using Newtonsoft.Json;
9  using DAL.apiClases;
10
11 namespace DAL
12 {
13     public class APICall
14     {
15         public static List<Root> GetCall()
16         {
17             ServicePointManager.Expect100Continue = true;
18             ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
19             WebRequest request = HttpWebRequest.Create(@"https://
                 fantasy.premierleague.com/api/fixtures/");
20             WebResponse responce = request.GetResponse();
21             StreamReader reader = new StreamReader(responce.GetResponseStream());
22             string football_Jason = reader.ReadToEnd();
23
24             var call = JsonConvert.DeserializeObject<List<Root>>(football_Jason);
25             return call;
26         }
27         public static Dictionary<string, Element> getListOfStats(List<string>
             ids)
28         {
29
30             ServicePointManager.Expect100Continue = true;
31             ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
32             WebRequest request = HttpWebRequest.Create(@"https://
                 fantasy.premierleague.com/api/bootstrap-static/");
33             WebResponse responce = request.GetResponse();
34             StreamReader reader = new StreamReader(responce.GetResponseStream());
35             string football_Jason = reader.ReadToEnd();
36
37             Dictionary<string, Element> dic = new Dictionary<string, Element>();
38
39             var call = JsonConvert.DeserializeObject<BooStat>(football_Jason);
40             foreach (Element eve in call.elements)
41             {
42                 foreach (string id in ids)
43                 {
44                     if (eve.id == Convert.ToInt32(id))
45                     {
46                         dic.Add(id, eve);
47                     }
48                 }
49             }
```

```
            return dic;
        }
        public static int getCurrentGameweek(int offset)
        {
            ServicePointManager.Expect100Continue = true;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            WebRequest request = HttpWebRequest.Create(@"https://fantasy.premierleague.com/api/bootstrap-static/");
            WebResponse responce = request.GetResponse();
            StreamReader reader = new StreamReader(responce.GetResponseStream());
            string football_Jason = reader.ReadToEnd();

            var call = JsonConvert.DeserializeObject<BooStat>(football_Jason);
            foreach (Event eve in call.events)
            {
                if (eve.finished == false)
                {
                    if (eve.id is int)
                    {
                        return eve.id + offset?? 1;
                    }

                }
            }
            return 1;
        }
        public static List<Root> sortbyNextGameWeek(List<Root> lr, int currGW)
        {
            List<Root> ret = new List<Root>();
            foreach (Root fixture in lr)
            {
                if (fixture.@event == currGW)
                {
                    ret.Add(fixture);
                }
            }
            return ret;
        }
        private static bool DateInsideOneWeek(DateTime date1, DateTime date2)
        {
            DayOfWeek firstDayOfWeek = System.Globalization.CultureInfo.CurrentCulture.DateTimeFormat.FirstDayOfWeek;
            DateTime startDateOfWeek = date1.Date;
            while (startDateOfWeek.DayOfWeek != firstDayOfWeek)
            { startDateOfWeek = startDateOfWeek.AddDays(-1d); }
            DateTime endDateOfWeek = startDateOfWeek.AddDays(6d);
            return date2 >= startDateOfWeek && date2 <= endDateOfWeek;
        }
        public static void addGamesToDB()
        {
            #region connecttoDb
```

```
            ServicePointManager.Expect100Continue = true;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            WebRequest request = HttpWebRequest.Create(@"https://fantasy.premierleague.com/api/fixtures/");
            WebResponse responce = request.GetResponse();
            StreamReader reader = new StreamReader(responce.GetResponseStream());
            string football_Jason = reader.ReadToEnd();

            List<Root> call = JsonConvert.DeserializeObject<List<Root>>(football_Jason);
            #endregion

            foreach (Root game in call)
            {
                string com = "UPDATE game SET [date] = '"+ (game.kickoff_time ?? DateTime.Now.AddYears(-1000)).ToString("dddd, dd MMMM h:mm tt") +"', homes = '"+game.team_h_score+"', ascore='"+game.team_a_score+"' Where gameID = "+game.id;
                oledbhelper.Execute(com);
            }
        }
        public static void addALLGamesToDB()
        {
            //done once to add all games to database
            #region connecttoDb
            ServicePointManager.Expect100Continue = true;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            WebRequest request = HttpWebRequest.Create(@"https://fantasy.premierleague.com/api/fixtures/");
            WebResponse responce = request.GetResponse();
            StreamReader reader = new StreamReader(responce.GetResponseStream());
            string football_Jason = reader.ReadToEnd();

            List<Root> call = JsonConvert.DeserializeObject<List<Root>>(football_Jason);
            #endregion

            foreach (Root game in call)
            {
                string com = "INSERT INTO game (gameID,gw,hteam,ateam,[date],homes,ascore) VALUES("+game.id+",'"+game.@event+"','"+game.team_h+"','"+game.team_a+"','" + (game.kickoff_time ?? DateTime.Now.AddYears(-1000)).ToString("dddd, dd MMMM h:mm tt") + "','"+game.team_h_score+"','"+game.team_a_score+"')";
                oledbhelper.Execute(com);
            }
        }
        public static int getCurrentGw()
        {
            ServicePointManager.Expect100Continue = true;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
```

```
            WebRequest request = HttpWebRequest.Create(@"https://fantasy.premierleague.com/api/bootstrap-static/");
            WebResponse responce = request.GetResponse();
            StreamReader reader = new StreamReader(responce.GetResponseStream());
            string football_Jason = reader.ReadToEnd();

            var call = JsonConvert.DeserializeObject<Root2>(football_Jason);


            foreach (Event2 gw in call.events)
            {
                if ( DateTime.Now.CompareTo(gw.deadline_time) < 0)
                {
                    return gw.id;
                }
            }
            return 1;
        }
    }
}

```

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

namespace DAL
{
    public class AesCryp
    {
        public static string IV = @"Xp2s5v8y5BvDlGcK";// 16 chars = 128 bits
        public static string Key = @"u7xsArDoF6JaNdRgUkXp2s5v8f2BlEaH"; // 32 chars = 256 bits

        public static string encrypt(string decrypted)
        {
            AesCryptoServiceProvider aes = new AesCryptoServiceProvider();
            aes.BlockSize = 128;
            aes.KeySize = 256;
            aes.IV = Encoding.UTF8.GetBytes(IV);
            aes.Key = Encoding.UTF8.GetBytes(Key);
            aes.Mode = CipherMode.CBC;
            aes.Padding = PaddingMode.PKCS7;

            // Convert string to byte array
            byte[] src = Encoding.Unicode.GetBytes(decrypted);

            // encryption
            using (ICryptoTransform encrypt = aes.CreateEncryptor())
            {
                byte[] dest = encrypt.TransformFinalBlock(src, 0, src.Length);

                // Convert byte array to Base64 strings
                return Convert.ToBase64String(dest);
            }
        }
        public static string Decrypt(string encrypted)
        {
            AesCryptoServiceProvider aes = new AesCryptoServiceProvider();
            aes.BlockSize = 128;
            aes.KeySize = 256;
            aes.IV = Encoding.UTF8.GetBytes(IV);
            aes.Key = Encoding.UTF8.GetBytes(Key);
            aes.Mode = CipherMode.CBC;
            aes.Padding = PaddingMode.PKCS7;

            // Convert Base64 strings to byte array
            byte[] src = System.Convert.FromBase64String(encrypted);

            // decryption
            using (ICryptoTransform decrypt = aes.CreateDecryptor())
```

```
            {
                byte[] dest = decrypt.TransformFinalBlock(src, 0, src.Length);
                return Encoding.Unicode.GetString(dest);
            }
        }
    }
}

```

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

namespace DAL.apiClases
{
    public class A
    {
        public int value { get; set; }
        public int element { get; set; }
    }

    public class H
    {
        public int value { get; set; }
        public int element { get; set; }
    }

    public class Stat
    {
        public string identifier { get; set; }
        public IList<A> a { get; set; }
        public IList<H> h { get; set; }
    }

    public class Root
    {
        public int code { get; set; }
        public int? @event { get; set; }
        public bool finished { get; set; }
        public bool finished_provisional { get; set; }
        public int id { get; set; }
        public DateTime? kickoff_time { get; set; }
        public int minutes { get; set; }
        public bool provisional_start_time { get; set; }
        public bool? started { get; set; }
        public int team_a { get; set; }
        public int? team_a_score { get; set; }
        public int team_h { get; set; }
        public int? team_h_score { get; set; }
        public IList<Stat> stats { get; set; }
        public int team_h_difficulty { get; set; }
        public int team_a_difficulty { get; set; }
        public int pulse_id { get; set; }
    }
    public class Root2
    {
        public List<Event2> events { get; set; }
    }
    public class Event2
```

```
    {
        public int id { get; set; }
        public string name { get; set; }
        public DateTime deadline_time { get; set; }
        public int average_entry_score { get; set; }
        public bool finished { get; set; }
        public bool data_checked { get; set; }
        public int? highest_scoring_entry { get; set; }
        public int deadline_time_epoch { get; set; }
        public int deadline_time_game_offset { get; set; }
        public int? highest_score { get; set; }
        public bool is_previous { get; set; }
        public bool is_current { get; set; }
        public bool is_next { get; set; }
        public int? most_selected { get; set; }
        public int? most_transferred_in { get; set; }
        public int? top_element { get; set; }
        public int transfers_made { get; set; }
        public int? most_captained { get; set; }
        public int? most_vice_captained { get; set; }
    }
}
```

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

namespace DAL.apiClases
{
    public class Event
    {
        public int? id { get; set; }
        public string name { get; set; }
        public DateTime? deadline_time { get; set; }
        public int? average_entry_score { get; set; }
        public bool? finished { get; set; }
        public bool? data_checked { get; set; }
        public object highest_scoring_entry { get; set; }
        public int? deadline_time_epoch { get; set; }
        public int? deadline_time_game_offset { get; set; }
        public object highest_score { get; set; }
        public bool? is_previous { get; set; }
        public bool? is_current { get; set; }
        public bool? is_next { get; set; }
        public List<object> chip_plays { get; set; }
        public object most_selected { get; set; }
        public object most_transferred_in { get; set; }
        public object top_element { get; set; }
        public object top_element_info { get; set; }
        public int? transfers_made { get; set; }
        public object most_captained { get; set; }
        public object most_vice_captained { get; set; }
    }

    public class GameSettings
    {
    }

    public class BooStat
    {
        public List<Event> events { get; set; }
        public GameSettings game_settings { get; set; }
        public List<object> phases { get; set; }
        public List<object> teams { get; set; }
        public int total_players { get; set; }
        public List<Element> elements { get; set; }
        public List<object> element_stats { get; set; }
        public List<object> element_types { get; set; }
    }
    public class Element
    {
        public int id { get; set; }
        public int clean_sheets { get; set; }
```

```
        public int assists { get; set; }
        public int goals_scored { get; set; }
        public int minutes { get; set; }
        public int saves { get; set; }
        public int red_cards { get; set; }
        public int yellow_cards { get; set; }

    }


}

```