Welcome to itucsdb1930's documentation!

Team

itucsdb1930

Members

- Muhammed Enes Tırnakçı
- Beraat Buz
- Ahmet Yılmaz

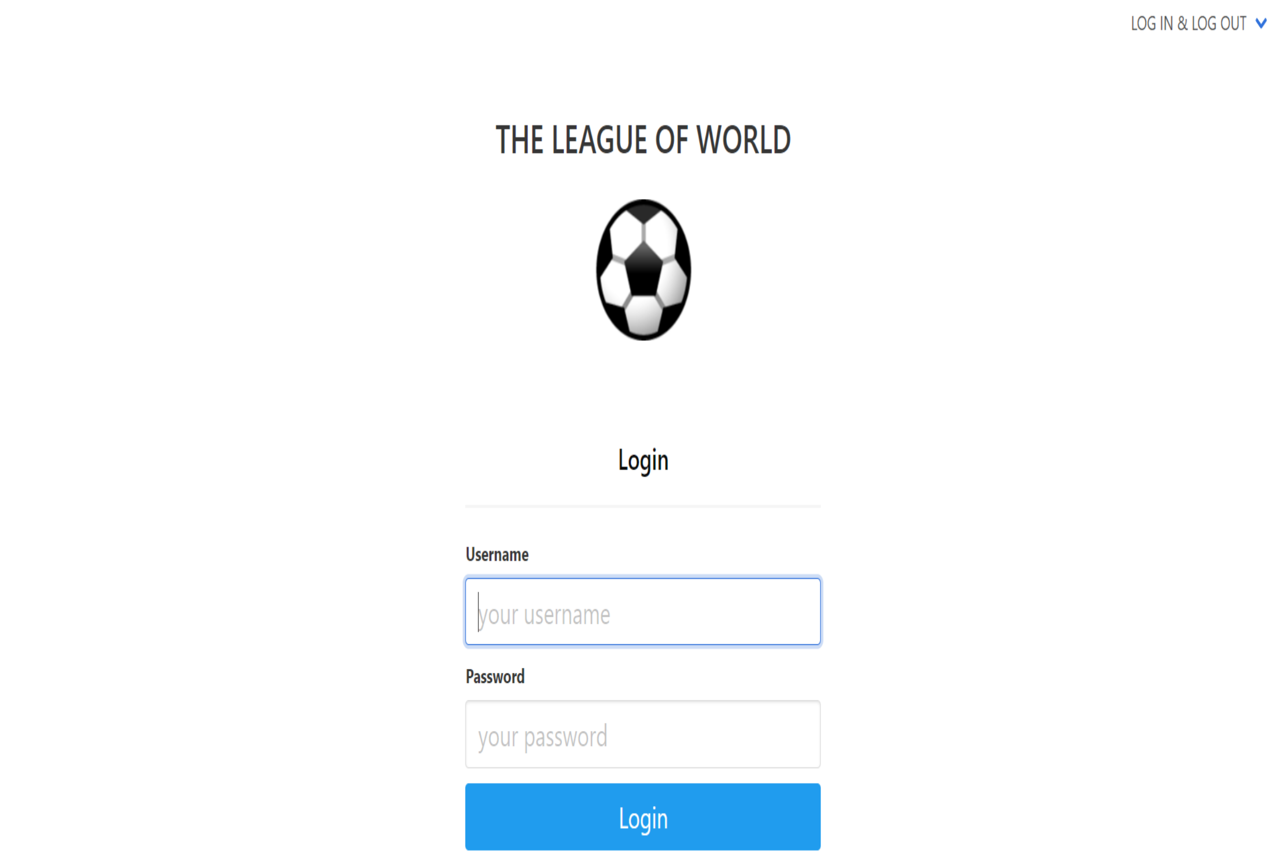
This system follows live football matches and keep some statistic about footballers, teams, matches over the matches played before in England Premier League.

App: http://itucsdb1930.herokuapp.com/

Github Page: https://github.com/itucsdb1930/itucsdb1930

Contents:

Parts Implemented by Beraat Buz



LOG IN & LOG OUT

THE LEAGUE OF WORLD

Login

Username

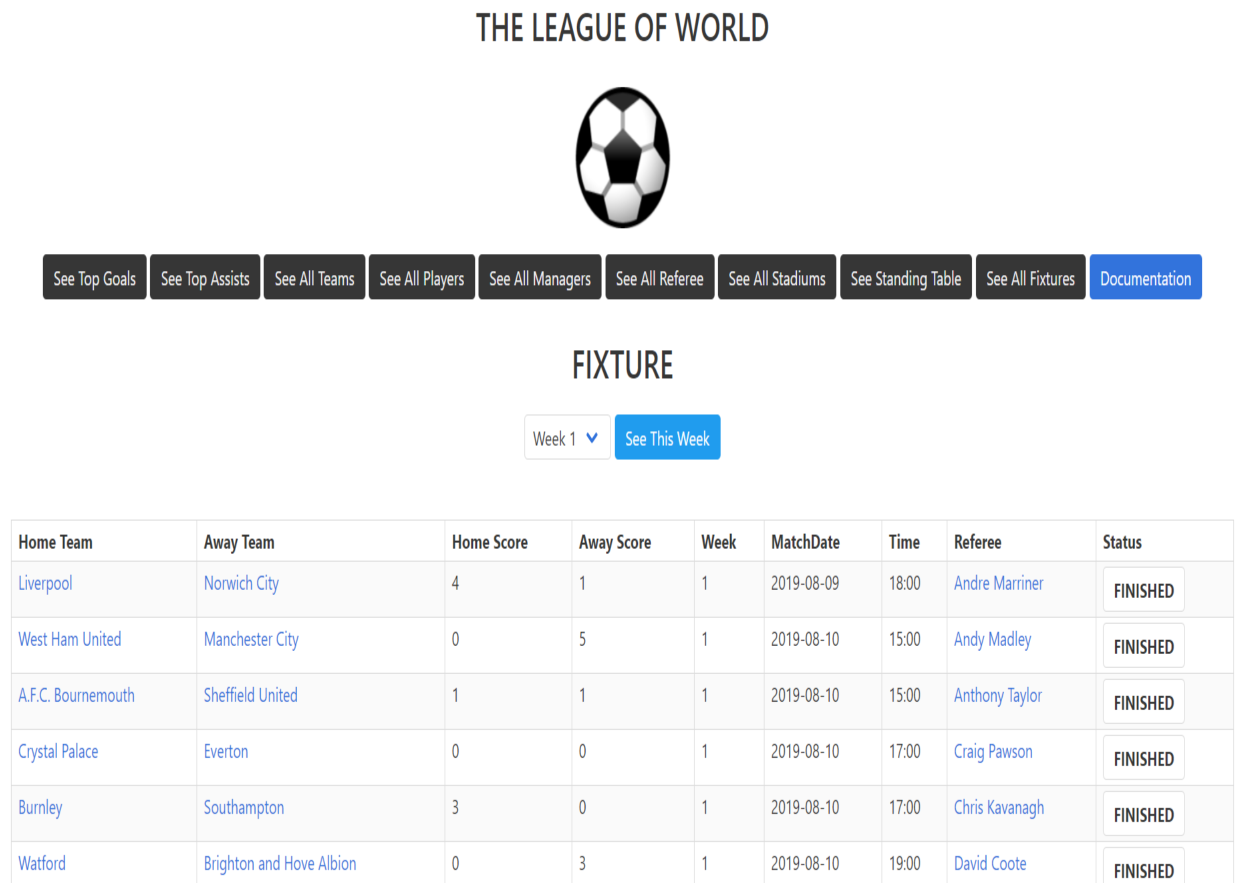
your username

Password

your password

Login

The administrator can login to operate CRUD operations



THE LEAGUE OF WORLD

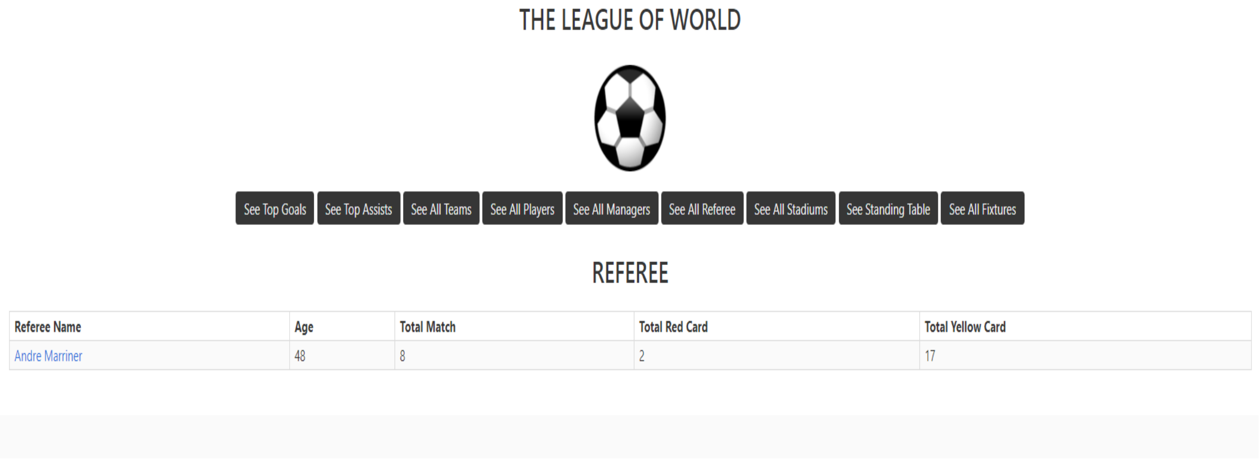
See Top Goals | See Top Assists | See All Teams | See All Players | See All Managers | See All Referee | See All Stadiums | See Standing Table | See All Fixtures | Documentation

FIXTURE

Week 1 | See This Week

Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status
Liverpool	Norwich City	4	1	1	2019-08-09	18:00	Andre Marriner	FINISHED
West Ham United	Manchester City	0	5	1	2019-08-10	15:00	Andy Madley	FINISHED
A.F.C. Bournemouth	Sheffield United	1	1	1	2019-08-10	15:00	Anthony Taylor	FINISHED
Crystal Palace	Everton	0	0	1	2019-08-10	17:00	Craig Pawson	FINISHED
Burnley	Southampton	3	0	1	2019-08-10	17:00	Chris Kavanagh	FINISHED
Watford	Brighton and Hove Albion	0	3	1	2019-08-10	19:00	David Coote	FINISHED

The user can see all fixtures up to week 4, and control the status of matches. Moreover, the user can reach more information about the match using status buttons.

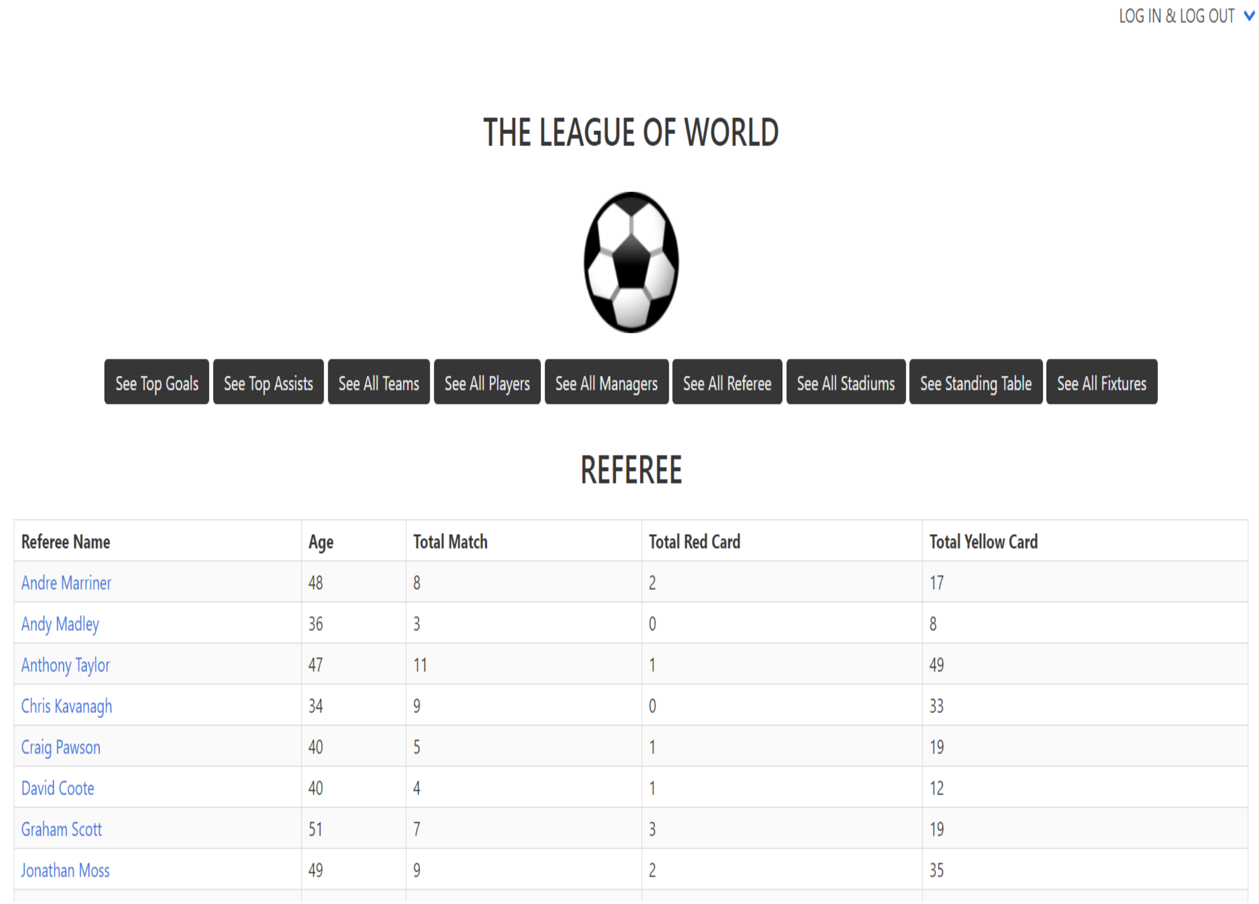


THE LEAGUE OF WORLD

REFEREE

Referee Name	Age	Total Match	Total Red Card	Total Yellow Card
Andre Marriner	48	8	2	17

The user can see some information about the referee by clicking referee name at fixture page.

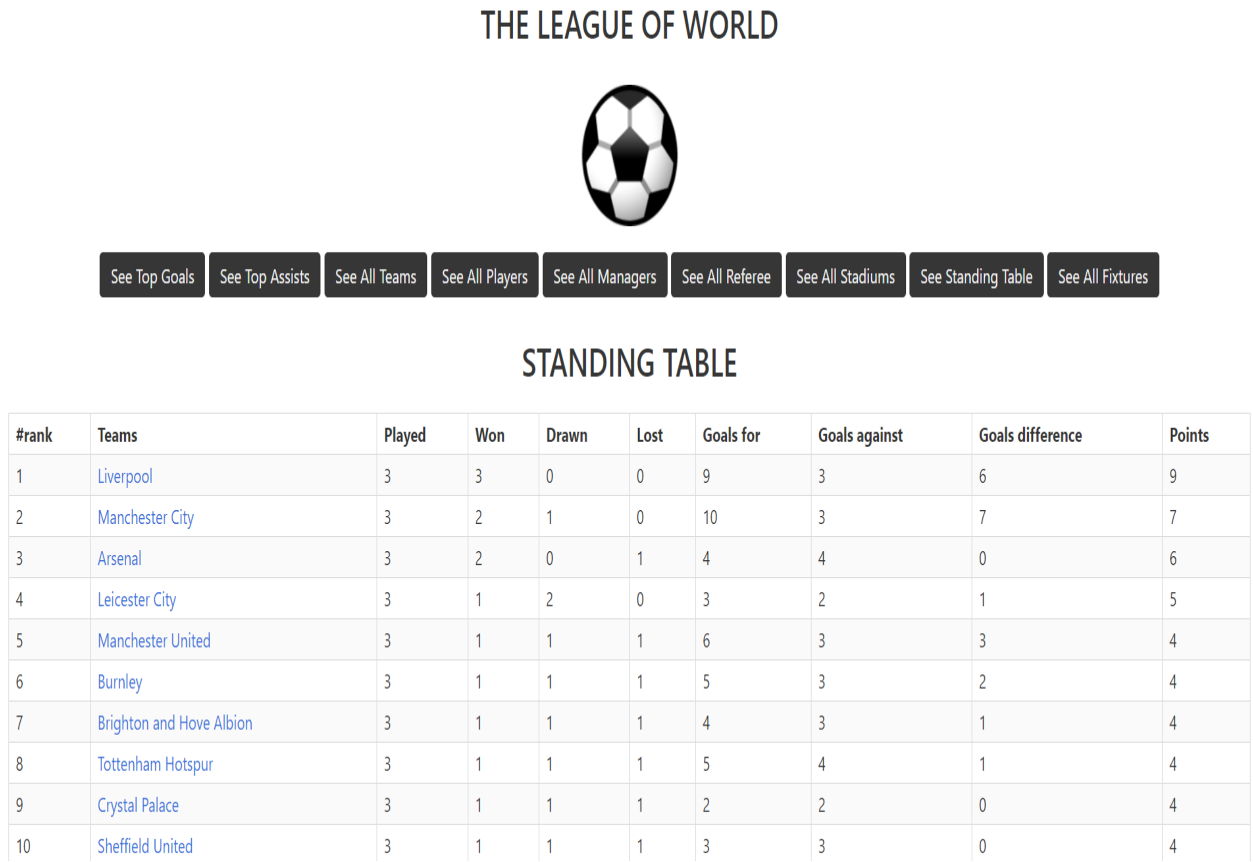


THE LEAGUE OF WORLD

REFEREE

Referee Name	Age	Total Match	Total Red Card	Total Yellow Card
Andre Marriner	48	8	2	17
Andy Madley	36	3	0	8
Anthony Taylor	47	11	1	49
Chris Kavanagh	34	9	0	33
Craig Pawson	40	5	1	19
David Coote	40	4	1	12
Graham Scott	51	7	3	19
Jonathan Moss	49	9	2	35

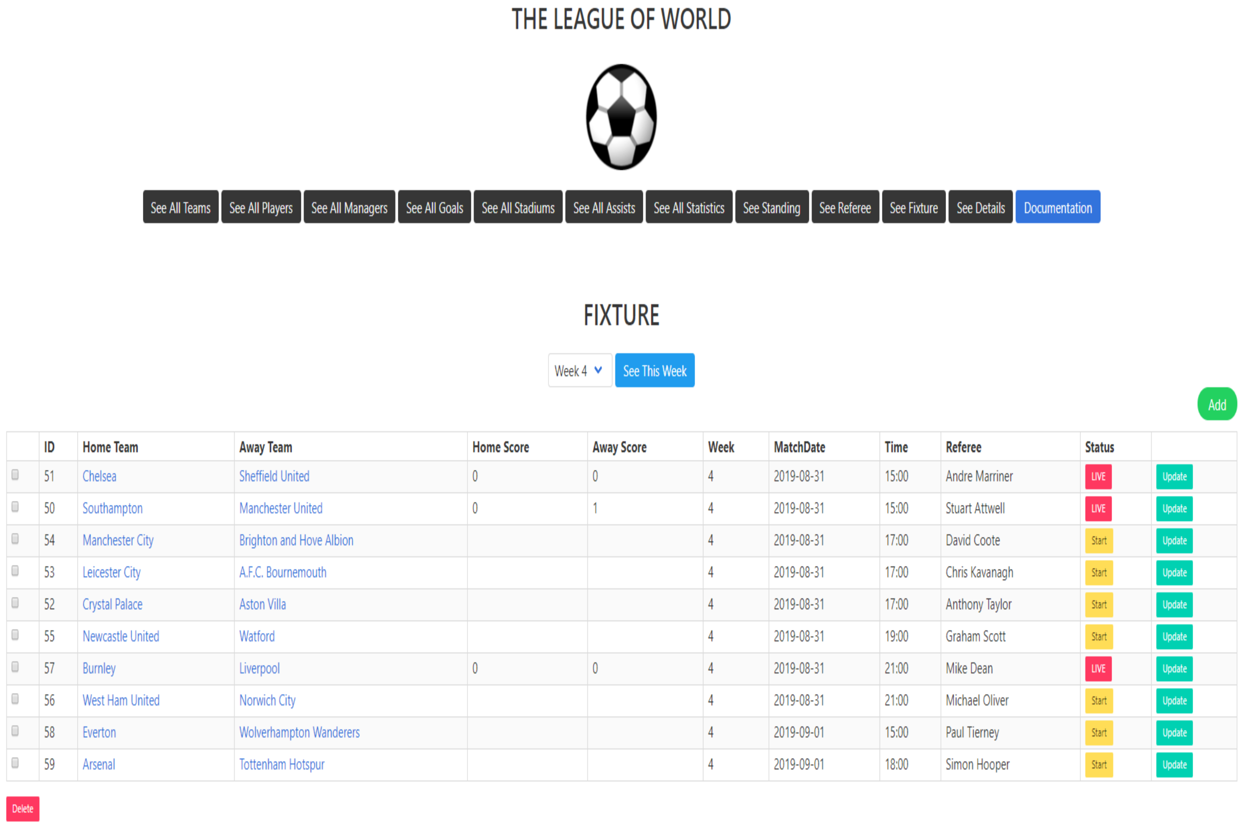
The user can see some information about all referees.



STANDING TABLE

#rank	Teams	Played	Won	Drawn	Lost	Goals for	Goals against	Goals difference	Points
1	Liverpool	3	3	0	0	9	3	6	9
2	Manchester City	3	2	1	0	10	3	7	7
3	Arsenal	3	2	0	1	4	4	0	6
4	Leicester City	3	1	2	0	3	2	1	5
5	Manchester United	3	1	1	1	6	3	3	4
6	Burnley	3	1	1	1	5	3	2	4
7	Brighton and Hove Albion	3	1	1	1	4	3	1	4
8	Tottenham Hotspur	3	1	1	1	5	4	1	4
9	Crystal Palace	3	1	1	1	2	2	0	4
10	Sheffield United	3	1	1	1	3	3	0	4

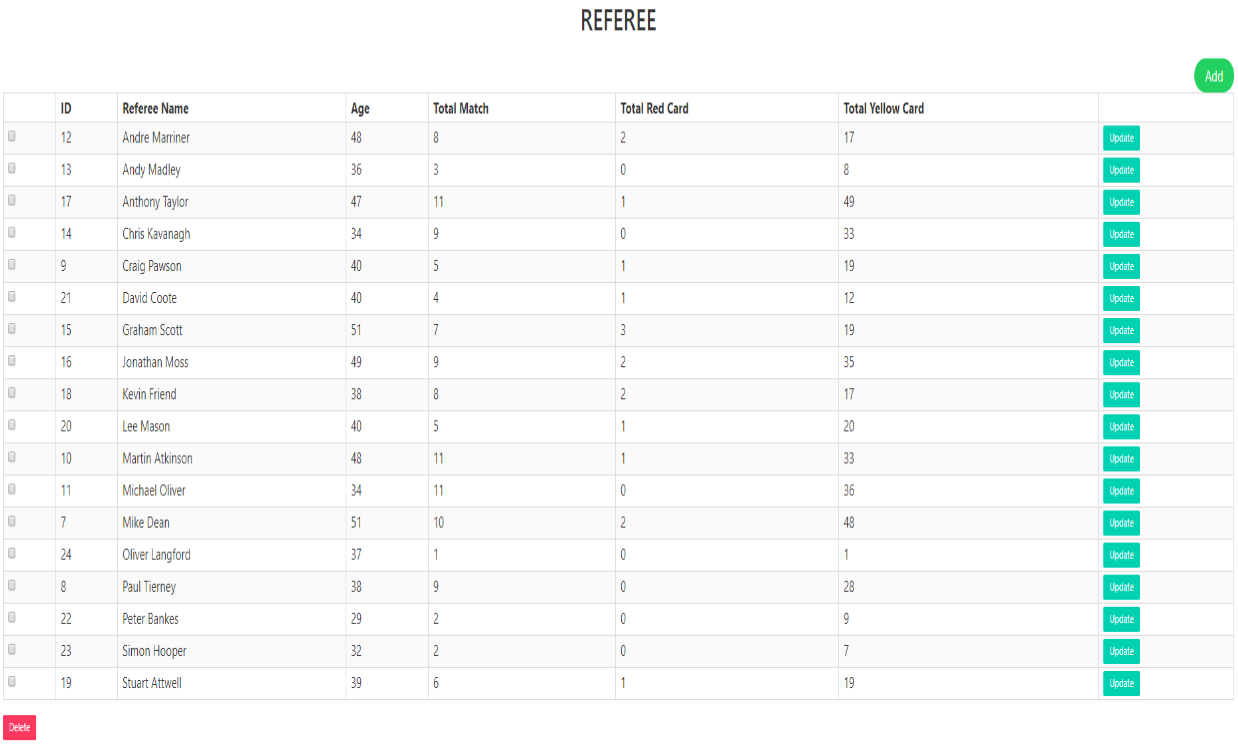
The user can see standing table in the EPL.



FIXTURE

	ID	Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status	
□	51	Chelsea	Sheffield United	0	0	4	2019-08-31	15:00	Andre Marriner	LIVE	Update
□	50	Southampton	Manchester United	0	1	4	2019-08-31	15:00	Stuart Attwell	LIVE	Update
□	54	Manchester City	Brighton and Hove Albion			4	2019-08-31	17:00	David Coote	Start	Update
□	53	Leicester City	A.F.C. Bournemouth			4	2019-08-31	17:00	Chris Kavanagh	Start	Update
□	52	Crystal Palace	Aston Villa			4	2019-08-31	17:00	Anthony Taylor	Start	Update
□	55	Newcastle United	Watford			4	2019-08-31	19:00	Graham Scott	Start	Update
□	57	Burnley	Liverpool	0	0	4	2019-08-31	21:00	Mike Dean	LIVE	Update
□	56	West Ham United	Norwich City			4	2019-08-31	21:00	Michael Oliver	Start	Update
□	58	Everton	Wolverhampton Wanderers			4	2019-09-01	15:00	Paul Tierney	Start	Update
□	59	Arsenal	Tottenham Hotspur			4	2019-09-01	18:00	Simon Hooper	Start	Update

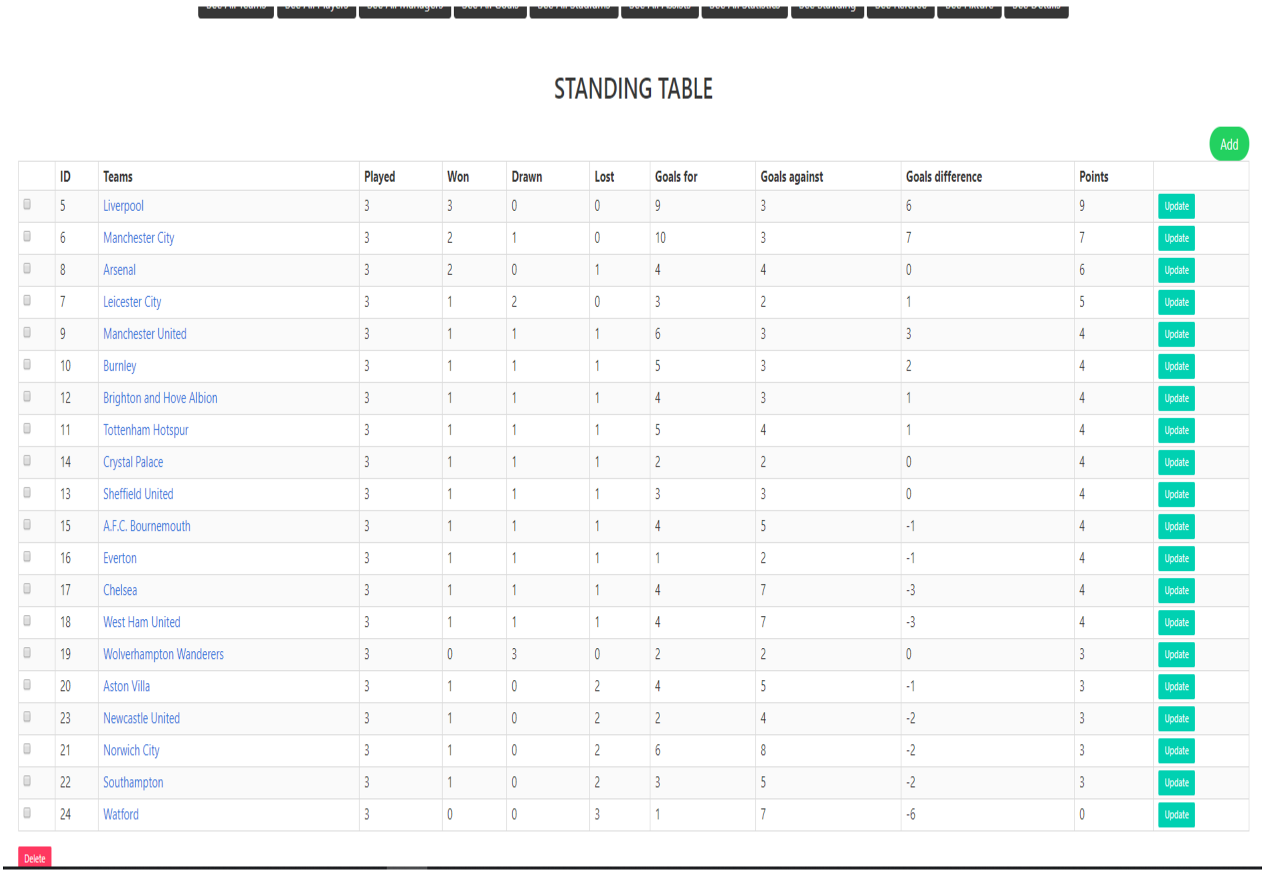
After login, in the beginnig the page in which add, update, delete and starting the match operations about fixture can be done seen. The administrator can do these operations. Delete operation is operated by using checkboxes.



REFEREE

	ID	Referee Name	Age	Total Match	Total Red Card	Total Yellow Card	
☐	12	Andre Marriner	48	8	2	17	Update
☐	13	Andy Madley	36	3	0	8	Update
☐	17	Anthony Taylor	47	11	1	49	Update
☐	14	Chris Kavanagh	34	9	0	33	Update
☐	9	Craig Pawson	40	5	1	19	Update
☐	21	David Coote	40	4	1	12	Update
☐	15	Graham Scott	51	7	3	19	Update
☐	16	Jonathan Moss	49	9	2	35	Update
☐	18	Kevin Friend	38	8	2	17	Update
☐	20	Lee Mason	40	5	1	20	Update
☐	10	Martin Atkinson	48	11	1	33	Update
☐	11	Michael Oliver	34	11	0	36	Update
☐	7	Mike Dean	51	10	2	48	Update
☐	24	Oliver Langford	37	1	0	1	Update
☐	8	Paul Tierney	38	9	0	28	Update
☐	22	Peter Bankes	29	2	0	9	Update
☐	23	Simon Hooper	32	2	0	7	Update
☐	19	Stuart Attwell	39	6	1	19	Update

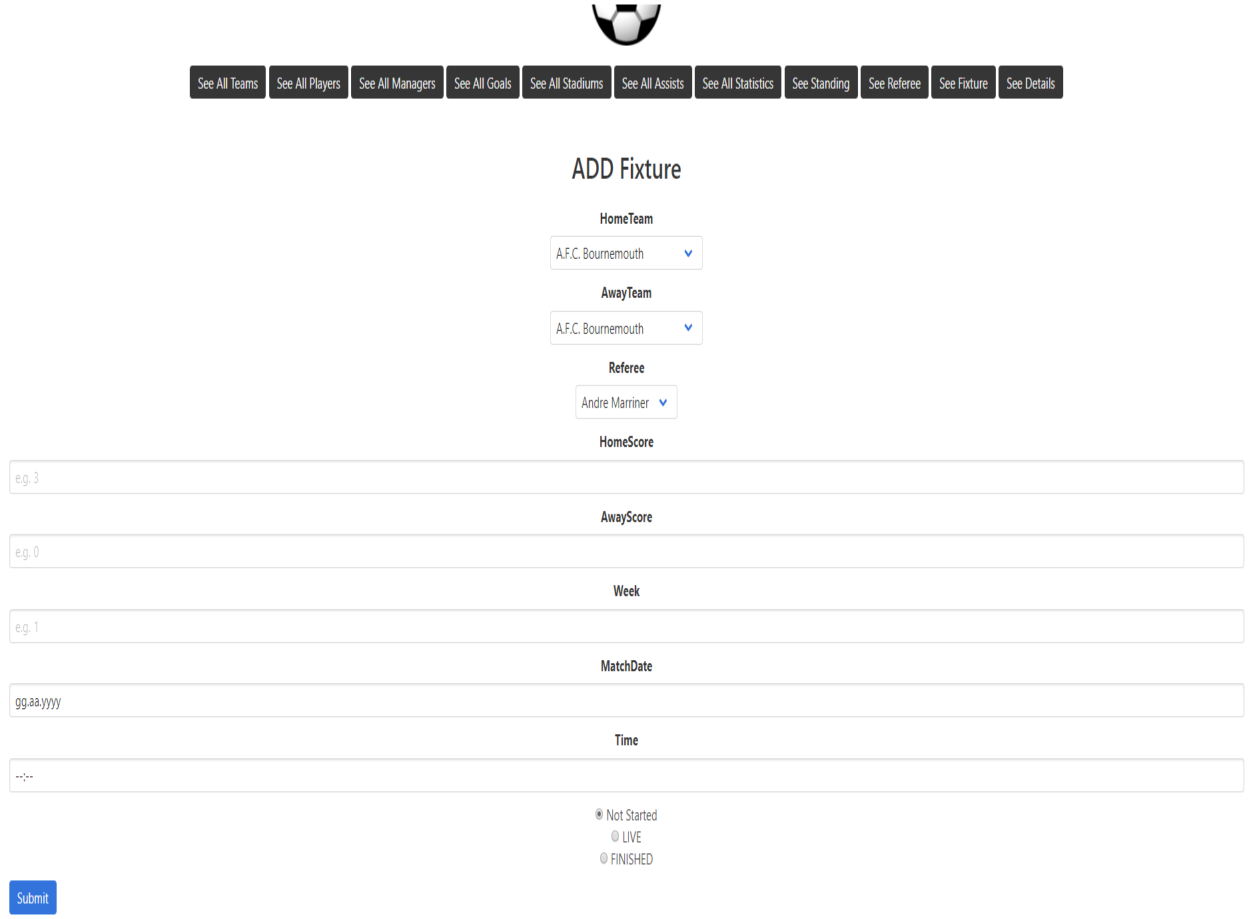
After login, by clicking see all referee the page in which add, update and delete operations about referees can be done seen. The administrator can do these operations. Delete operation is operated by using checkboxes.



STANDING TABLE

	ID	Teams	Played	Won	Drawn	Lost	Goals for	Goals against	Goals difference	Points	
☐	5	Liverpool	3	3	0	0	9	3	6	9	Update
☐	6	Manchester City	3	2	1	0	10	3	7	7	Update
☐	8	Arsenal	3	2	0	1	4	4	0	6	Update
☐	7	Leicester City	3	1	2	0	3	2	1	5	Update
☐	9	Manchester United	3	1	1	1	6	3	3	4	Update
☐	10	Burnley	3	1	1	1	5	3	2	4	Update
☐	12	Brighton and Hove Albion	3	1	1	1	4	3	1	4	Update
☐	11	Tottenham Hotspur	3	1	1	1	5	4	1	4	Update
☐	14	Crystal Palace	3	1	1	1	2	2	0	4	Update
☐	13	Sheffield United	3	1	1	1	3	3	0	4	Update
☐	15	A.F.C. Bournemouth	3	1	1	1	4	5	-1	4	Update
☐	16	Everton	3	1	1	1	1	2	-1	4	Update
☐	17	Chelsea	3	1	1	1	4	7	-3	4	Update
☐	18	West Ham United	3	1	1	1	4	7	-3	4	Update
☐	19	Wolverhampton Wanderers	3	0	3	0	2	2	0	3	Update
☐	20	Aston Villa	3	1	0	2	4	5	-1	3	Update
☐	23	Newcastle United	3	1	0	2	2	4	-2	3	Update
☐	21	Norwich City	3	1	0	2	6	8	-2	3	Update
☐	22	Southampton	3	1	0	2	3	5	-2	3	Update
☐	24	Watford	3	0	0	3	1	7	-6	0	Update

After login, by clicking see standing table the page in which add, update and delete operations about standing table can be done seen. The administrator can do these operations. Delete operation is operated by using checkboxes.



ADD Fixture

HomeTeam: A.F.C. Bournemouth

AwayTeam: A.F.C. Bournemouth

Referee: Andre Marriner

HomeScore: e.g. 3

AwayScore: e.g. 0

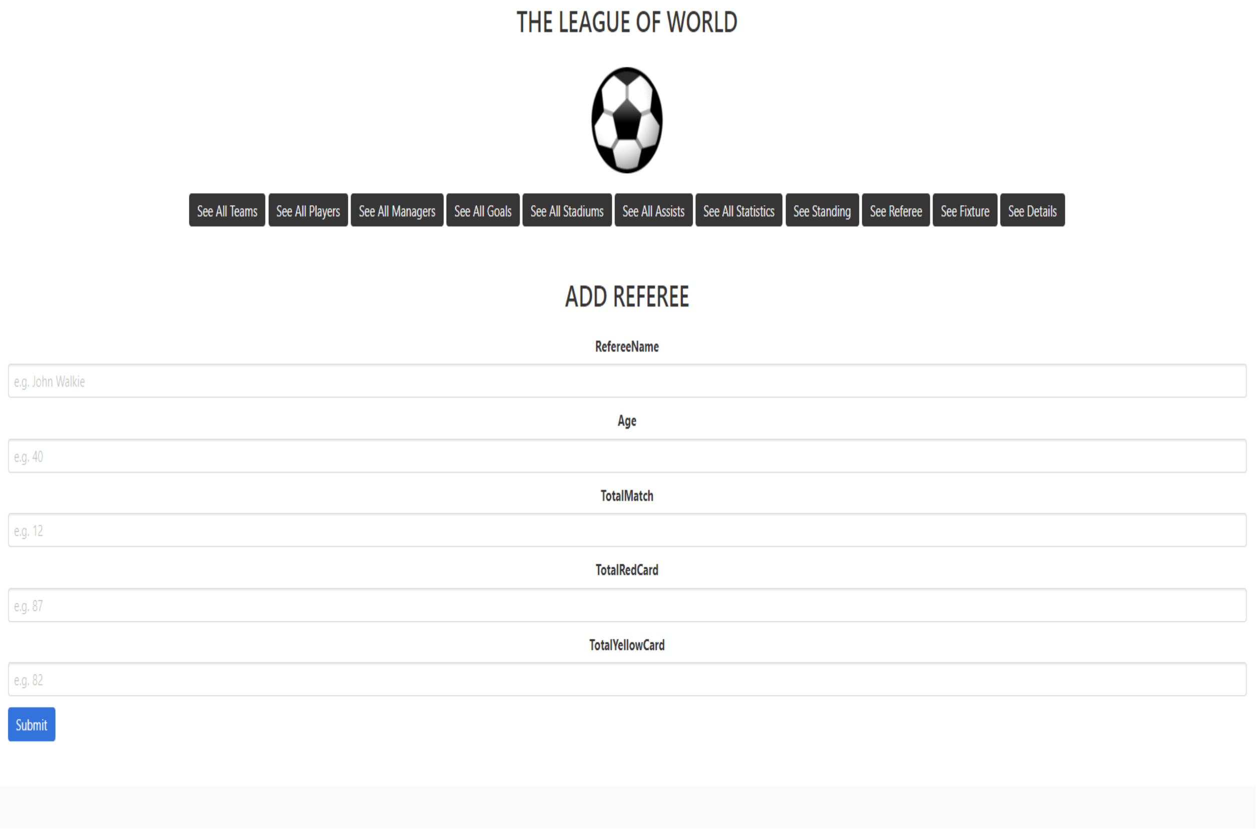
Week: e.g. 1

MatchDate: gg.aa.yyyy

Time: --:--

● Not Started
○ LIVE
○ FINISHED

The administrator can add more fixtures in this page.



ADD REFEREE

RefereeName: e.g. John Walkie

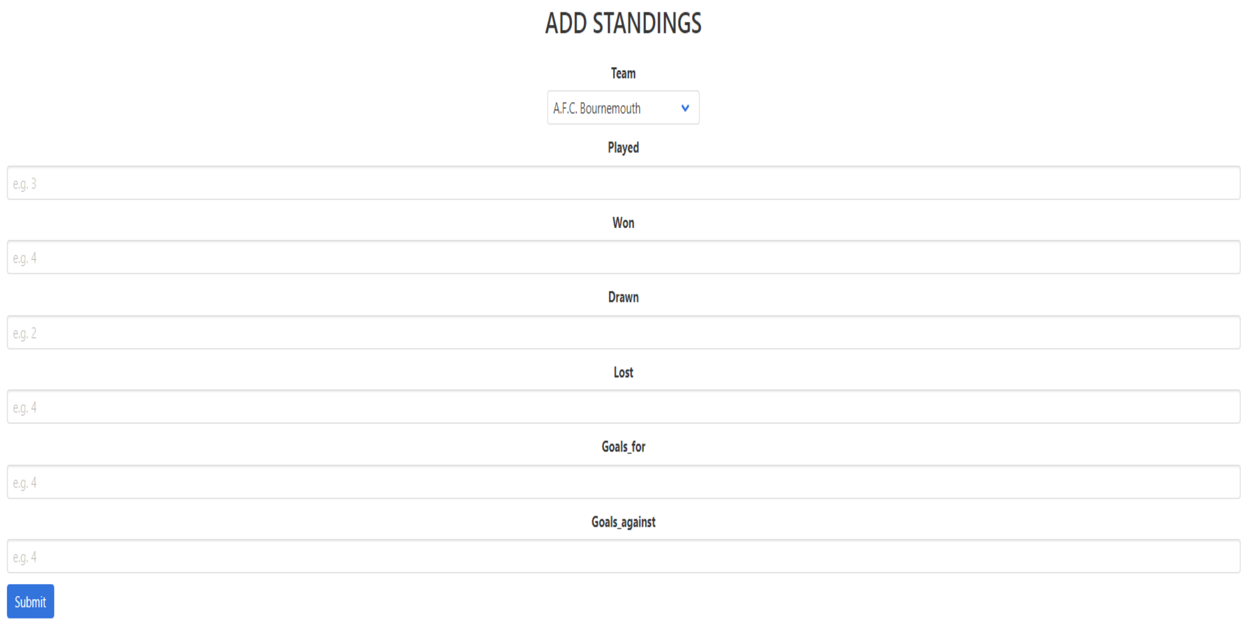
Age: e.g. 40

TotalMatch: e.g. 12

TotalRedCard: e.g. 87

TotalYellowCard: e.g. 82

The administrator can add more referees in this page.



ADD STANDINGS

Team: A.F.C. Bournemouth

Played: e.g. 3

Won: e.g. 4

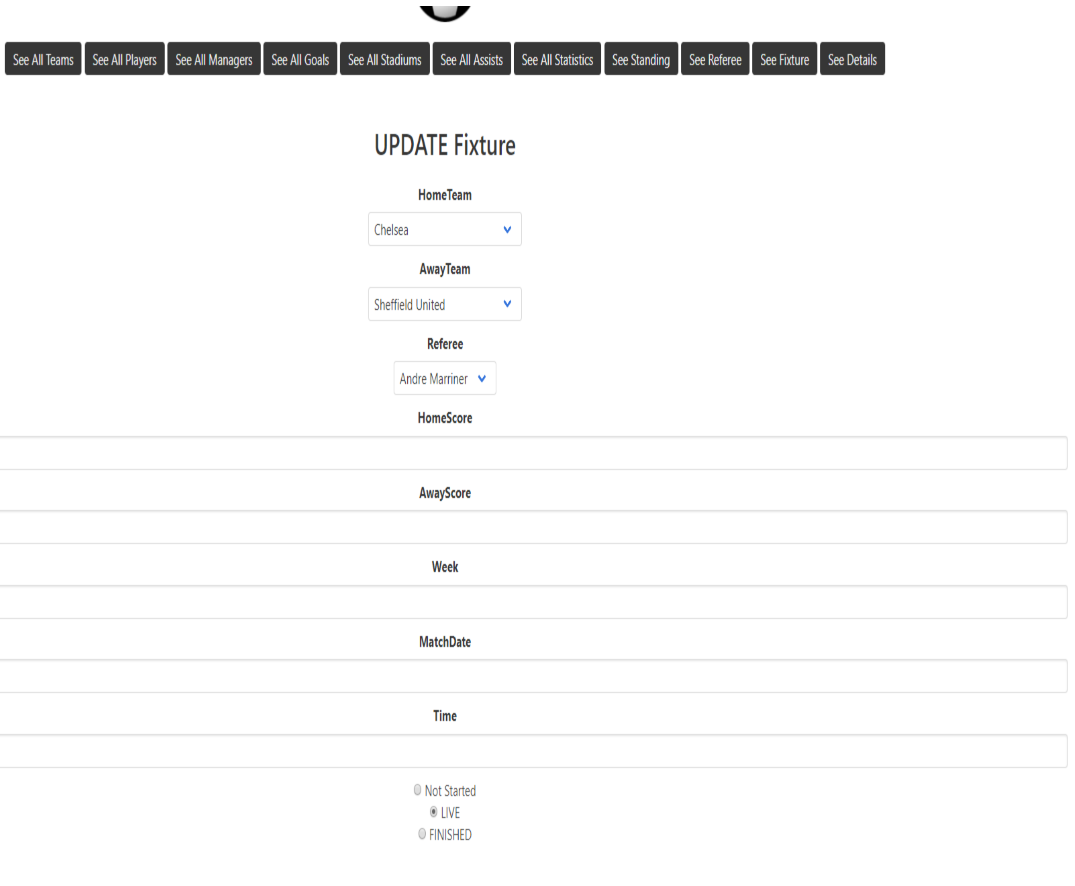
Drawn: e.g. 2

Lost: e.g. 4

Goals_for: e.g. 4

Goals_against: e.g. 4

The administrator can add more teams to standing table in this page.



UPDATE Fixture

HomeTeam: Chelsea

AwayTeam: Sheffield United

Referee: Andre Marriner

HomeScore: 0

AwayScore: 0

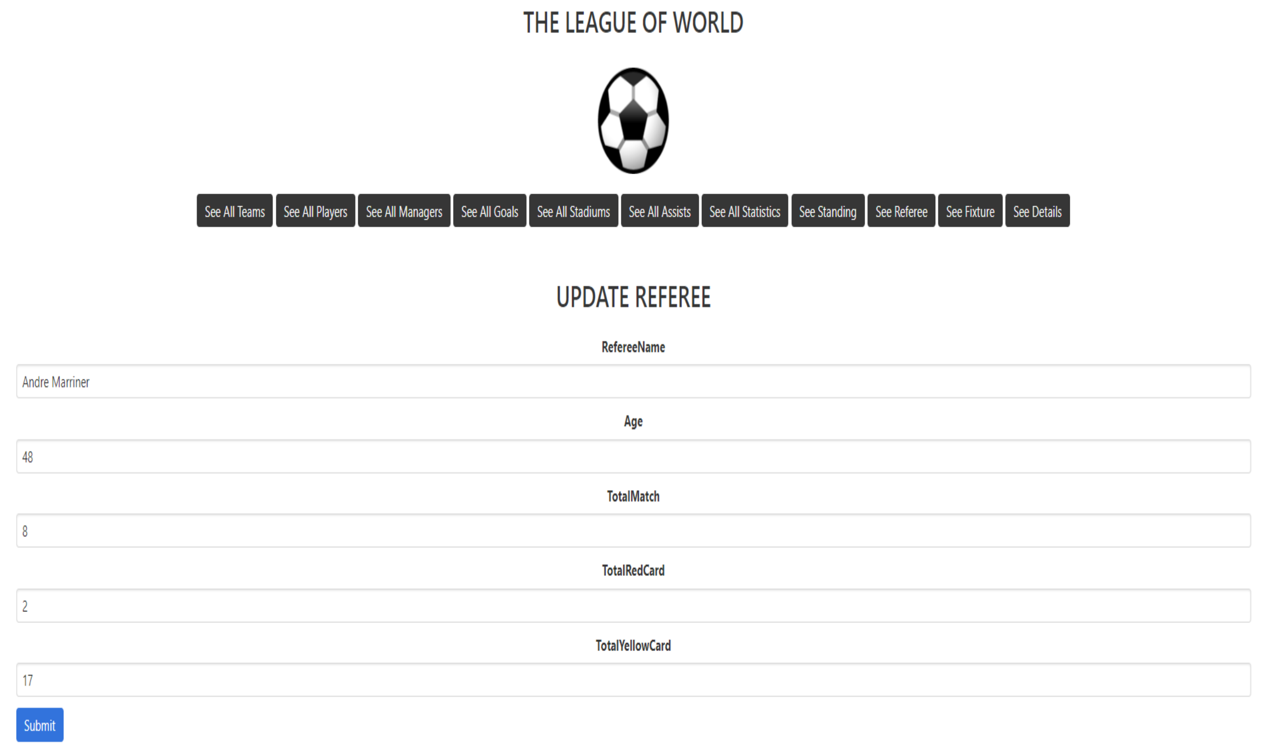
Week: 4

MatchDate: 31.08.2019

Time: 15:00

○ Not Started
● LIVE
○ FINISHED

The administrator can update the fixture which is selected in this page.



UPDATE REFEREE

RefereeName

Andre Marriner

Age

48

TotalMatch

8

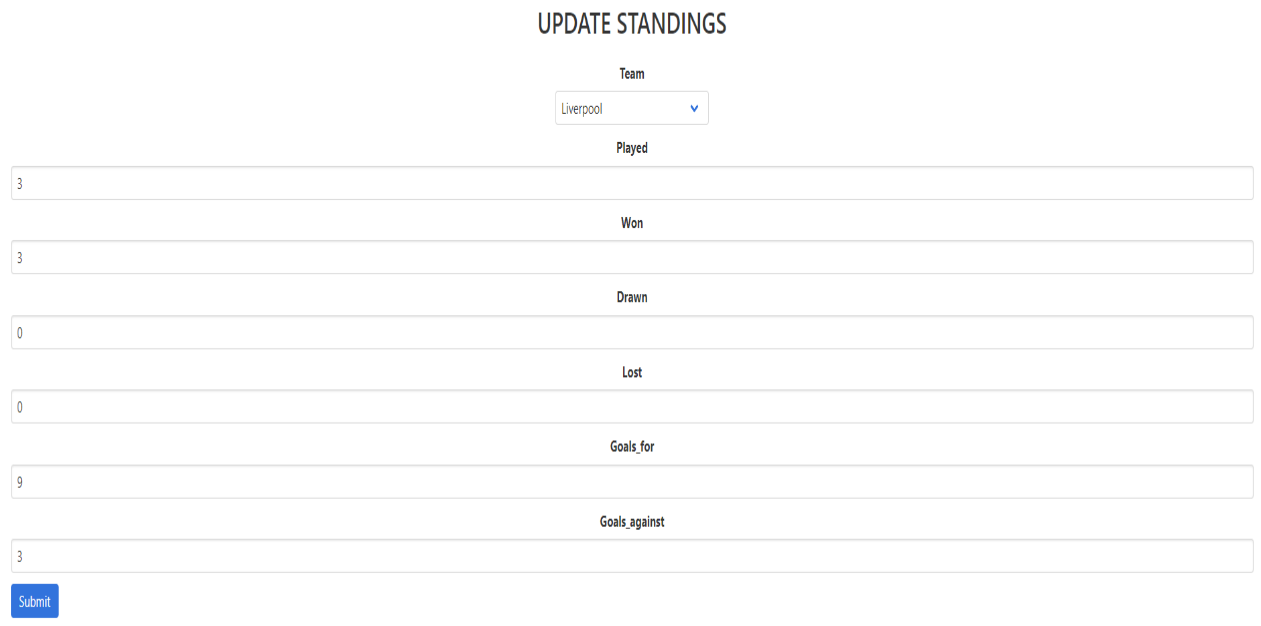
TotalRedCard

2

TotalYellowCard

17

The administrator can update the referee which is selected in this page.



UPDATE STANDINGS

Team

Liverpool

Played

3

Won

3

Drawn

0

Lost

0

Goals_for

9

Goals_against

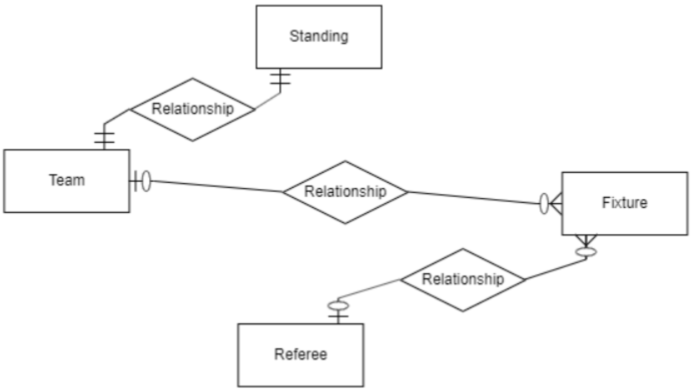
3

The administrator can update the team’s status in the standing table which is selected in this page.

Parts Implemented by Beraat Buz

Database Design

explain the database design of your project



Code

explain the technical structure of your code

to include a code listing, use the following example:

```
lm = LoginManager()

        @lm.user_loader
        def load_user(user_id):
                return get_user(user_id)

        app = Flask(__name__)
        app.config['SECRET_KEY'] = 'ThisisSecret'
        app.config.from_object("settings")
        @app.route("/")
        def home_page():
                 logout_user()
                 return redirect(url_for("fixture_user_page"))
        class LoginForm(FlaskForm):
                username = StringField("Username", validators=[DataRequired()])
                password = PasswordField("Password", validators=[DataRequired()])

        @app.route("/login",methods=['GET', 'POST'])
        def login_page():
                form = LoginForm()
                if form.is_submitted():
                        username = form.data["username"]
                        user = get_user(username)
                        if user is not None:
                                password = form.data["password"]
                                if hasher.verify(password, user.password):
                                        login_user(user)
                                        flash("You have logged in.")
                                        next_page = request.args.get("next",
url_for("fixture_page"))
                                        return redirect(next_page)
                        flash("Invalid credentials.")
                        #abort(401)
                return render_template("login.html", form=form)
```

This code is required for login and login page. ınput validation is done. Hasher is used to hide the password.

```
@app.route("/fixture_user", methods=['GET','POST'])
def fixture_user_page():
        obje = forms.FootballStats()
        if request.method == "GET":
                cursor=obje.Fixtures(1)
                return render_template("user-fixture.html",cursor=cursor)
        else:
                process = request.form.get('buttonName')
                week = request.form.get('select')
                cursor=obje.Fixtures(week)
                return render_template("user-fixture.html",cursor=cursor)
```

This code is required for fixture page at user side.

```
def Fixtures(self,week):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """Select Fixtures.ID,T1.TeamName
,T2.TeamName,Week,MatchDate,Time,HomeScore,AwayScore,Status,RefereeName, HomeTeam,
AwayTeam,Refereeid FROM Fixtures,Teams AS T1,Teams AS T2,Referee WHERE Week = %s AND
T1.ID=HomeTeam AND T2.ID=AwayTeam AND Refereeid=Referee.id ORDER BY MatchDate,Time;"""
                          cursor.execute(statement,([week]))
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

This code is required for fixture page at user side for week selection and table creation. The cursor is used in HTML file.

```
@app.route("/referee_user", methods=['GET'])
    def referee_user_page():
           obje = forms.FootballStats()
                  if request.method == "GET":
                          cursor=obje.Referee()
                          return render_template("user-referee.html",cursor=cursor)
```

This code is required for referee page at user side.

```
def Referee(self):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """Select * FROM Referee ORDER BY RefereeName"""
                          cursor.execute(statement)
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

This code is required for referee page at user side for table creation. The cursor is used in HTML file.

```
@app.route("/standing_user", methods=['GET'])
          def standing_user_page():
                  obje = forms.FootballStats()
                  if request.method == "GET":
                          cursor=obje.Standings()
                          return render_template("user-standing.html",cursor=cursor)
```

This code is required for standing table page at user side.

```
def Standings(self):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """Select 
Standings.ID,TeamName,Played,Won,Drawn,Lost,Goals_for,Goals_against,Goals_difference,Points,
 Teams.id FROM Standings,Teams WHERE Teams.ID=TeamID ORDER BY Points DESC,Goals_difference 
DESC,TeamName;"""
                          cursor.execute(statement)
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

This code is required for standing page at user side for table creation. The cursor is used in HTML file.

```
@app.route("/fixture", methods=['GET','POST'])
        @login_required
        def fixture_page():
                if not current_user.is_admin:
                        abort(401)
                obje = forms.FootballStats()
                if request.method == "GET":
                        cursor=obje.Fixtures(1)
                        return render_template("fixture.html",cursor=cursor)
                else:
                        process = request.form.get('buttonName')
                        processStart = request.form.get('Start')
                        processLive = request.form.get('Live')
                        update = request.form.get('Update')
                        if (process == "add"):
                                return redirect(url_for("fixture_adding_page"))
                        elif (processStart):
                                return fixture_update_page(processStart)
                        elif (processLive):
                                return live_match_page(processLive)
                        elif (process == "week"):
                                week = request.form.get('select')
                                cursor=obje.Fixtures(week)
                                return render_template("fixture.html",cursor=cursor)

                        elif (process == "Delete"):
                                form_fixture_keys = request.form.getlist('fixture')
                                for form_fixture_key in form_fixture_keys:
                                        obje.Fixture_delete(int(form_fixture_key))
                                return redirect(url_for("fixture_page"))
                        else:
                                return fixture_update_page(process)
```

This code is required for fixture page at administrator side for table creation. Firstly, using GET method creates table for week 1. Then, by clicking desired button operates add, delete, update or starting match operations.

```
@app.route("/add_fixture", methods=['GET','POST'])
        @login_required
        def fixture_adding_page():
                obje = forms.FootballStats()
                cursor1=obje.Team()
                cursor2=obje.Referee()
                if not current_user.is_admin:
                        abort(401)
                if request.method == 'GET':
                        return render_template('add_fixture.html',cursor=[cursor1,cursor2])

                elif request.method == 'POST':
                        HomeTeam = request.form["HomeTeam"]
                        AwayTeam = request.form["AwayTeam"]
                        HomeScore = request.form["HomeScore"]
                        AwayScore =  request.form["AwayScore"]
                        Week =  request.form["Week"]
                        MatchDate =  request.form["MatchDate"]
                        Time =  request.form["Time"]
                        Status = request.form["Status"]
                        Refereeid=request.form["Refereeid"]
                        obje = forms.FootballStats()

obje.Fixture_add(HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid)

                        flash("You have added.")
                        return render_template("add_fixture.html",cursor=[cursor1,cursor2])
```

This code is required for fixture adding page at administrator side. Two cursors are used to post input by using selection.

```
def Fixture_add(self,
HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """ INSERT INTO
Fixtures(HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);"""
                          cursor.execute(statement,
([HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid]))
```

This code is required for fixture adding page at administrator side. This query provides insertion to Fixtures Table.

```
@app.route("/update_fixture", methods=['GET','POST'])
@login_required
def fixture_update_page(process):
        obje = forms.FootballStats()
        if not current_user.is_admin:
                abort(401)
        update = request.form.get('Update')
        if request.method == 'GET':
                return render_template("fixture.html")
        elif request.method == 'POST':
                if update is not None:
                        HomeTeam = request.form["HomeTeam"]
                        AwayTeam = request.form["AwayTeam"]
                        HomeScore = request.form["HomeScore"]
                        AwayScore =  request.form["AwayScore"]
                        Week =  request.form["Week"]
                        MatchDate =  request.form["MatchDate"]
                        Time =  request.form["Time"]
                        Status = request.form["Status"]
                        Refereeid=request.form["Refereeid"]

obje.Fixture_update(update,HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,R

                        return redirect(url_for("fixture_page"))
                cursor=obje.Fixture_update_info(process)
                cursor1=obje.Team()
                cursor2=obje.Referee()
                return render_template("update_fixture.html",cursor=
[cursor,cursor1,cursor2])
```

This code is required for fixture update page at administrator side. Two cursors are used to post input by using selection, one cursor is used to select the row from database we want to update.

```
def Fixture_update_info(self, ID):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """ Select * From Fixtures where ID = %s;"""
                          cursor.execute(statement,([ID]))
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

To select the row from database we want to update.

```
def Fixture_update(self,
FixtureID,HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Update Fixtures Set HomeTeam=%s, AwayTeam=%s,
HomeScore=%s, AwayScore=%s, Week=%s,MatchDate=%s,Time=%s,Status=%s,Refereeid=%s Where
ID=%s;"""
                          cursor.execute(statement,
([HomeTeam,AwayTeam,HomeScore,AwayScore,Week,MatchDate,Time,Status,Refereeid,FixtureID]))
```

To update database.

```
def Fixture_delete(self,FixtureId):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Delete From Fixtures Where ID = %s;"""
                          cursor.execute(statement,([FixtureId]))
```

To delete checked rows.

```
    @app.route("/standing", methods=['GET','POST'])
    @login_required
    def standing_page():
            if not current_user.is_admin:
                    abort(401)
            obje = forms.FootballStats()
            if request.method == "GET":
                    cursor=obje.Standings()
                    return render_template("standing.html",cursor=cursor)
            else:
                    process = request.form.get('buttonName')
                    update = request.form.get('Update')
                    if (process == "add"):
                            return redirect(url_for("standing_adding_page"))
                    elif (process == "Delete"):
                            form_standing_keys = request.form.getlist('standing')
                            for form_standing_key in form_standing_keys:
                                    obje.Standing_delete(int(form_standing_key))
                            return redirect(url_for("standing_page"))
                    else:
return standing_update_page(process)
```

This code is required for standing page at administrator side for table creation. By clicking desired button operates add, delete or update operations.

This code is required for standing row adding page at administrator side. The cursor is used to post input by using selection.

```
def Standing_add(self, TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """ INSERT INTO
Standings(TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against,Goals_difference,Points)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);"""
                          cursor.execute(statement,
([TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against,(int(Goals_for)-int(Goals_against)),
(3*int(Won)+int(Drawn))]))
```

This query provides insertion to Standings Table.

```
@app.route("/update_standing", methods=['GET','POST'])
        @login_required
        def standing_update_page(process):
                if not current_user.is_admin:
                        abort(401)
                obje = forms.FootballStats()
                update = request.form.get('Update')
                if request.method == 'GET':
                        return render_template("standing.html")
                elif request.method == 'POST':
                        if update is not None:
                                TeamID = request.form["TeamID"]
                                Played = request.form["Played"]
                                Won = request.form["Won"]
                                Drawn =  request.form["Drawn"]
                                Lost =  request.form["Lost"]
                                Goals_for =  request.form["Goals_for"]
                                Goals_against =  request.form["Goals_against"]
                                obje = forms.FootballStats()

obje.Standing_update(update,TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against)
                                return redirect(url_for("standing_page"))
                        cursor=obje.Standing_update_info(process)
                        cursor1=obje.Team()
                        return render_template("update_standing.html",cursor=
[cursor,cursor1])
```

This code is required for standing update page at administrator side. One cursor is used to post input by using selection, one cursor is used to select the row from database we want to update.

```
def Standing_update_info(self, ID):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """ Select * From Standings where ID = %s;"""
                          cursor.execute(statement,([ID]))
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

To select the row of standing table we want to update.

```
def Standing_update(self, StandingId,TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Update Standings Set TeamID=%s, Played=%s, Won=%s,
Drawn=%s, Lost=%s,Goals_for=%s,Goals_against=%s,Goals_difference=%s,Points=%s Where
ID=%s;"""
                          cursor.execute(statement,
([TeamID,Played,Won,Drawn,Lost,Goals_for,Goals_against,(int(Goals_for)-int(Goals_against)),
(3*int(Won)+int(Drawn)),StandingId]))
```

This querry updates the row of standings table we have selected.

```
def Standing_delete(self,StandingId):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Delete From Standings Where ID = %s;"""
                          cursor.execute(statement,([StandingId]))
```

This querry deletes the row of standings table we have checked.

```
@app.route("/referee", methods=['GET','POST'])
        @login_required
        def referee_page():
                if not current_user.is_admin:
                        abort(401)
                obje = forms.FootballStats()
                if request.method == "GET":
                        cursor=obje.Referee()
                        return render_template("referee.html",cursor=cursor)
                else:
                        process = request.form.get('buttonName')
                        update = request.form.get('Update')
                        if (process == "add"):
                                return redirect(url_for("referee_adding_page"))
                        if(process == "Delete"):
                                form_referee_keys = request.form.getlist('referee')
                                for form_referee_key in form_referee_keys:
                                        print(form_referee_key)
                                        obje.Referee_delete(int(form_referee_key))
                                return redirect(url_for("referee_page"))
                        else:
                                return referee_update_page(process)
```

This code is required for referee page at administrator side for table creation. By clicking desired button operates add, delete or update operations.

```
@app.route("/add_referee", methods=['GET','POST'])
@login_required
def referee_adding_page():
        if not current_user.is_admin:
                abort(401)
        if request.method == 'GET':
                return render_template('add_referee.html')

        elif request.method == 'POST':
                RefereeName = request.form["RefereeName"]
                Age = request.form["Age"]
                TotalMatch = request.form["TotalMatch"]
                TotalRedCard =  request.form["TotalRedCard"]
                TotalYellowCard =  request.form["TotalYellowCard"]
                obje = forms.FootballStats()
                obje.Referee_add(RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard)
                flash("You have added.")
                return render_template("add_referee.html")
```

This code is required for referee adding page at administrator side.

```
def Referee_add(self, RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """ INSERT INTO
Referee(RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard) VALUES(%s,%s,%s,%s,%s);"""
                          cursor.execute(statement,
([RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard]))
```

This querry is needed for adding new referee to referee table.

```
@app.route("/update_referee", methods=['GET','POST'])
@login_required
def referee_update_page(process):
        if not current_user.is_admin:
                abort(401)
        obje = forms.FootballStats()
        update = request.form.get('Update')
        if request.method == 'GET':
                return render_template("referee.html")
        elif request.method == 'POST':
                if update is not None:
                        RefereeName = request.form["RefereeName"]
                        Age = request.form["Age"]
                        TotalMatch = request.form["TotalMatch"]
                        TotalRedCard =  request.form["TotalRedCard"]
                        TotalYellowCard =  request.form["TotalYellowCard"]
                        obje = forms.FootballStats()

obje.Referee_update(update,RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard)
                        return redirect(url_for("referee_page"))
                cursor=obje.Referee_update_info(process)
                return render_template("update_referee.html",cursor=cursor)
```

This code is required for referee update page at administrator side. The cursor is used to select the row from database we want to update.

To select the row of referee table we want to update.

```
def Referee_update(self,
RefereeID,RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Update Referee Set RefereeName=%s, Age=%s,
TotalMatch=%s, TotalRedCard=%s, TotalYellowCard=%s Where ID=%s;"""
                          cursor.execute(statement,
([RefereeName,Age,TotalMatch,TotalRedCard,TotalYellowCard,RefereeID]))
```

This querry updates the row of referee table we have selected.

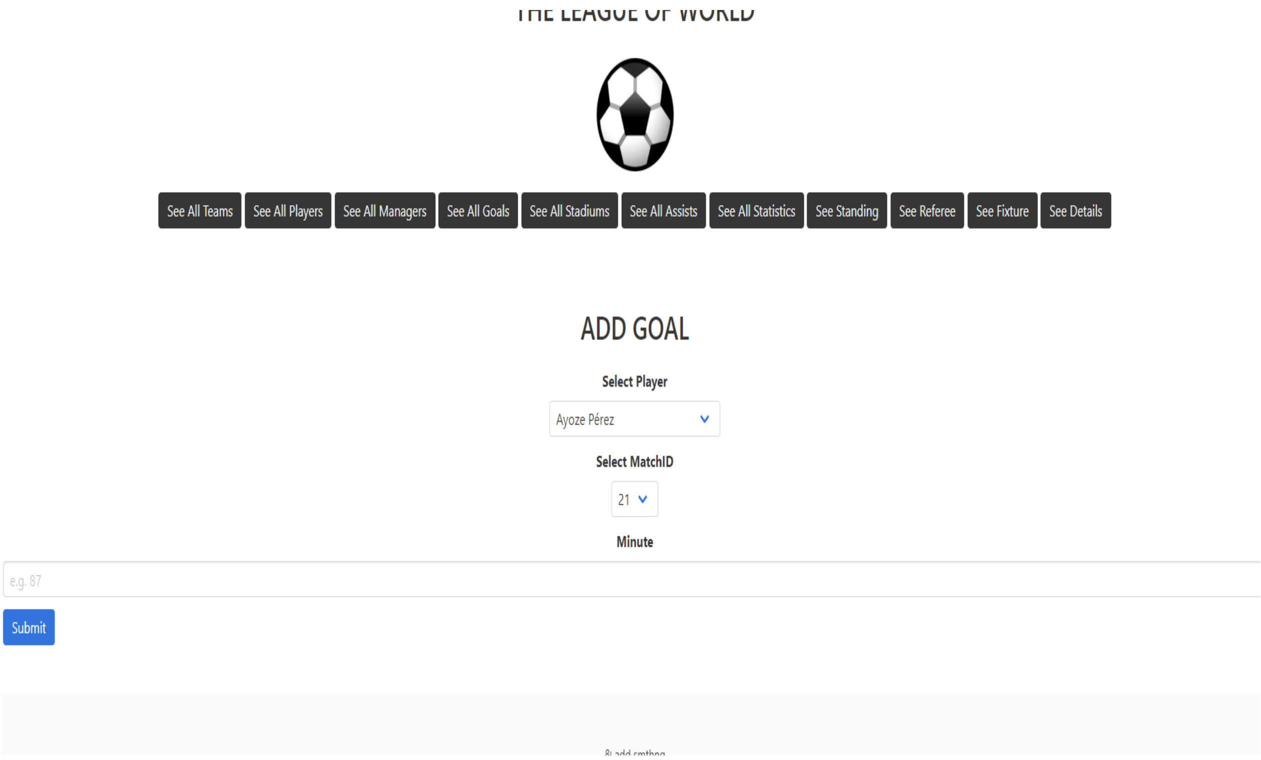
```
def Referee_delete(self,RefereeId):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement="""Delete From Referee Where ID = %s;"""
                          cursor.execute(statement,([RefereeId]))
```

This querry deletes the row of referee table we have checked.

```
def Fixture_key(self,Key):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """Select Fixtures.ID,T1.TeamName
,T2.TeamName,Week,MatchDate,Time,HomeScore,AwayScore,Status,RefereeName,HomeTeam,AwayTeam,Refe
 FROM Fixtures,Teams AS T1,Teams AS T2,Referee
                          WHERE T1.ID=HomeTeam AND T2.ID=AwayTeam AND Refereeid=Referee.id
and Fixtures.id=%s ORDER BY MatchDate,Time;"""
                          cursor.execute(statement, [Key])
                          cursor_list=cursor.fetchall()
                          return cursor_list
    def Fixtures2(self):
          with dbapi.connect(url) as connection:
                  with connection.cursor() as cursor:
                          statement = """Select Fixtures.ID,T1.TeamName
,T2.TeamName,Week,MatchDate,Time,HomeScore,AwayScore,Status,RefereeName FROM Fixtures,Teams
AS T1,Teams AS T2,Referee WHERE T1.ID=HomeTeam AND T2.ID=AwayTeam AND Refereeid=Referee.id
ORDER BY MatchDate,Time;"""
                          cursor.execute(statement)
                          cursor_list=cursor.fetchall()
                          return cursor_list
```

These codes is needed other tables which is references to fixture table.

Parts Implemented by Muhammed Enes Tırnakçı



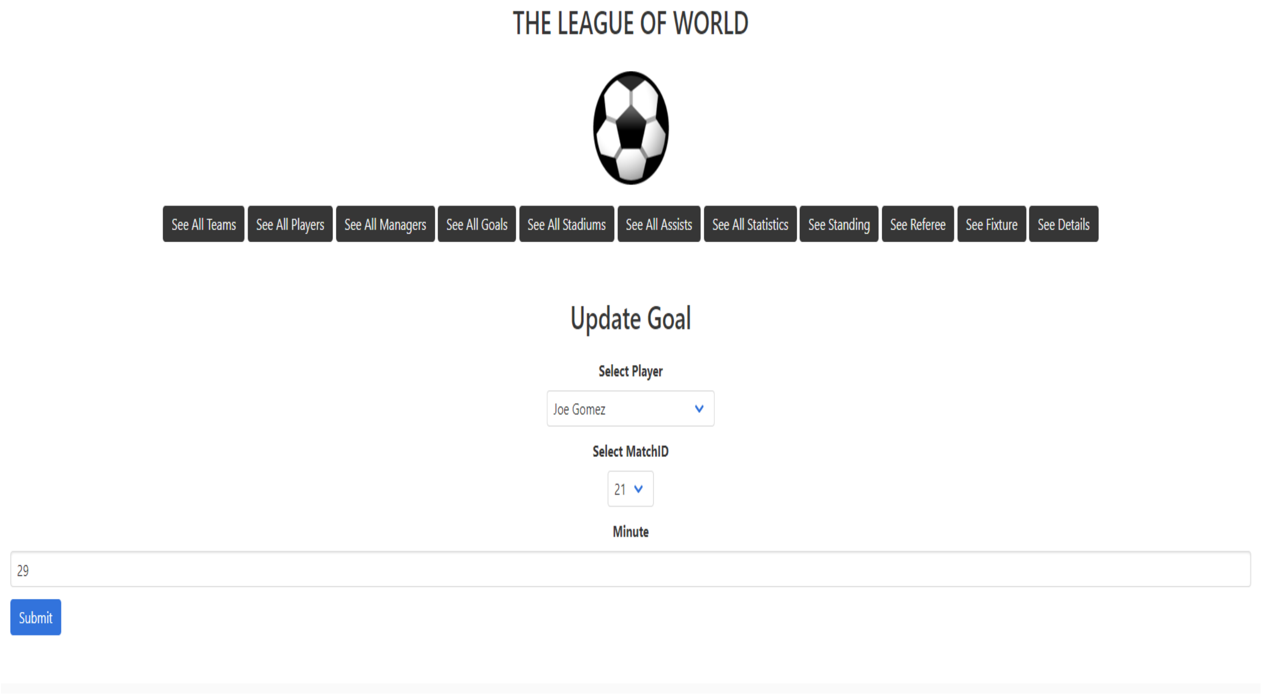
ADD GOAL

Select Player: Ayoze Pérez

Select MatchID: 21

Minute: e.g. 87

Goal adddding page for developer.



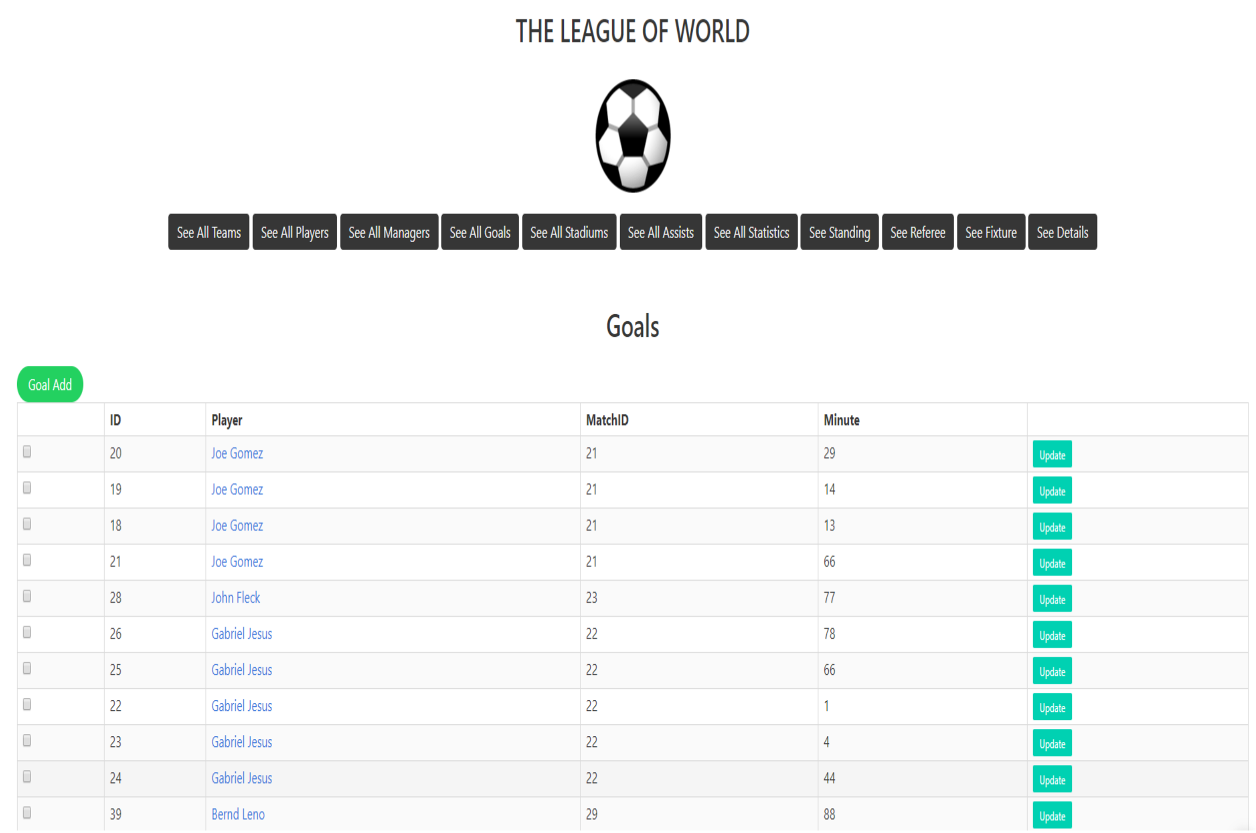
Update Goal

Select Player: Joe Gomez

Select MatchID: 21

Minute: 29

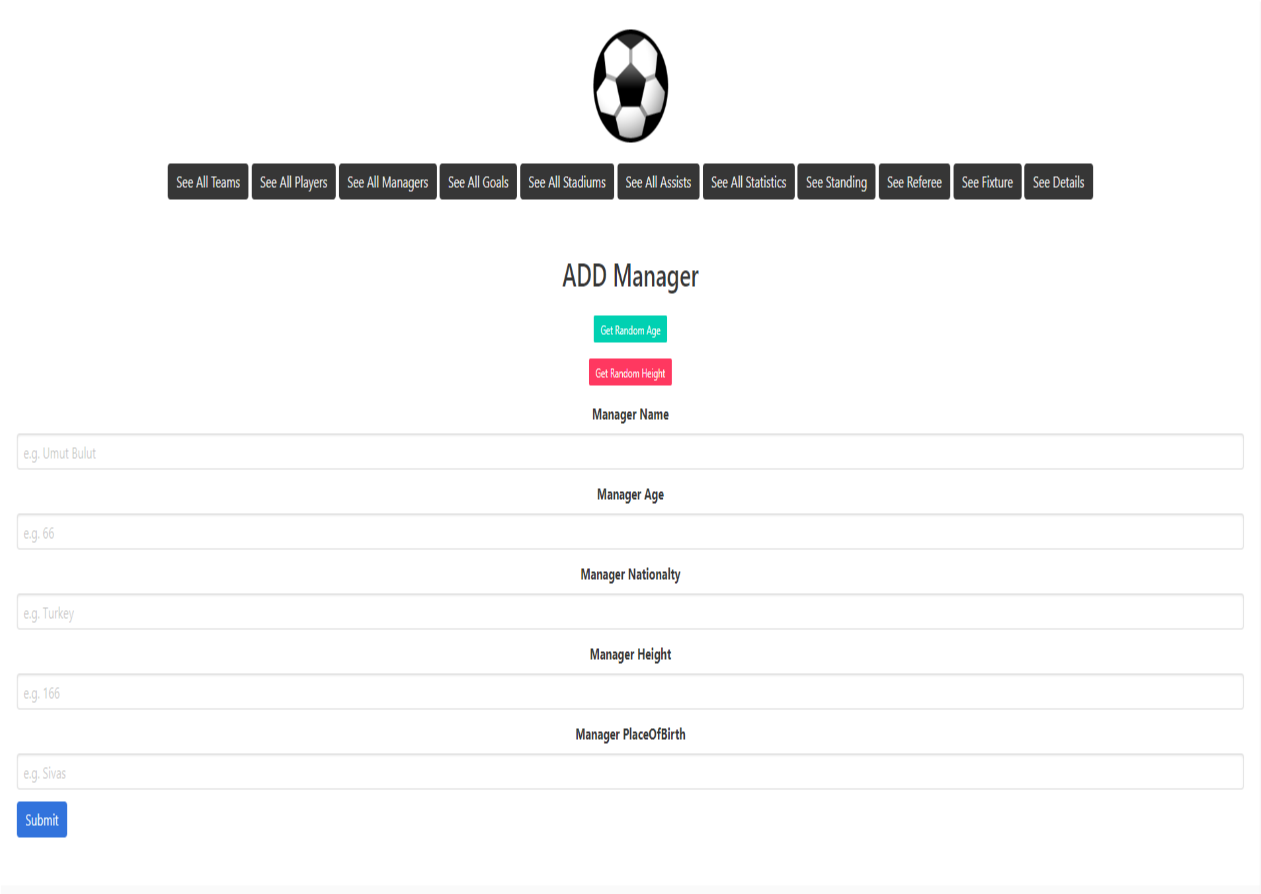
Goal update page for developer.



Goals

	ID	Player	MatchID	Minute	
□	20	Joe Gomez	21	29	Update
□	19	Joe Gomez	21	14	Update
□	18	Joe Gomez	21	13	Update
□	21	Joe Gomez	21	66	Update
□	28	John Fleck	23	77	Update
□	26	Gabriel Jesus	22	78	Update
□	25	Gabriel Jesus	22	66	Update
□	22	Gabriel Jesus	22	1	Update
□	23	Gabriel Jesus	22	4	Update
□	24	Gabriel Jesus	22	44	Update
□	39	Bernd Leno	29	88	Update

The developer can see all goals in this page by clicking the 'See All Goals' button. Developer can delete and update the goals.



ADD Manager

Manager Name

e.g. Umut Bulut

Manager Age

e.g. 66

Manager Nationalty

e.g. Turkey

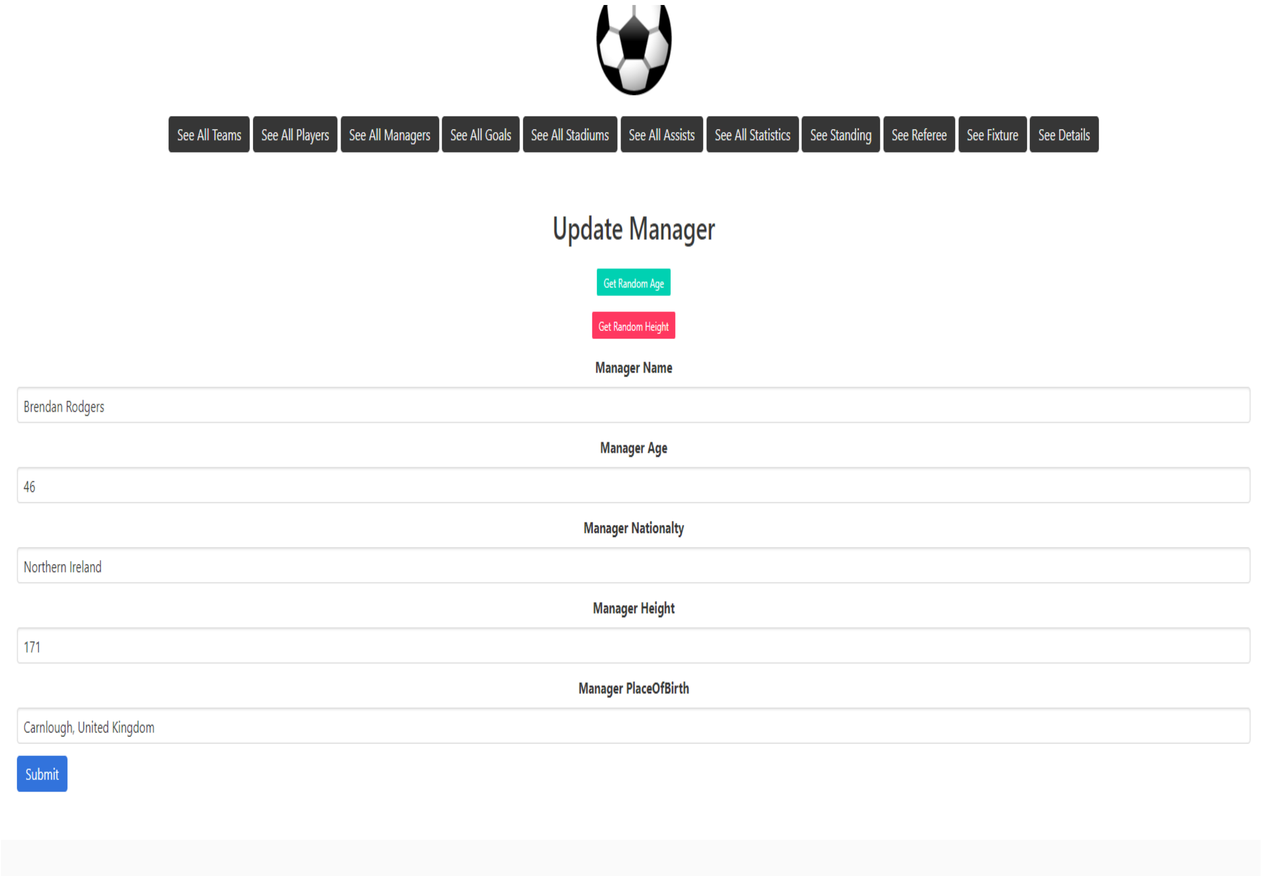
Manager Height

e.g. 166

Manager PlaceOfBirth

e.g. Sivas

Manager adddding page for developer.



Update Manager

Manager Name

Brendan Rodgers

Manager Age

46

Manager Nationalty

Northern Ireland

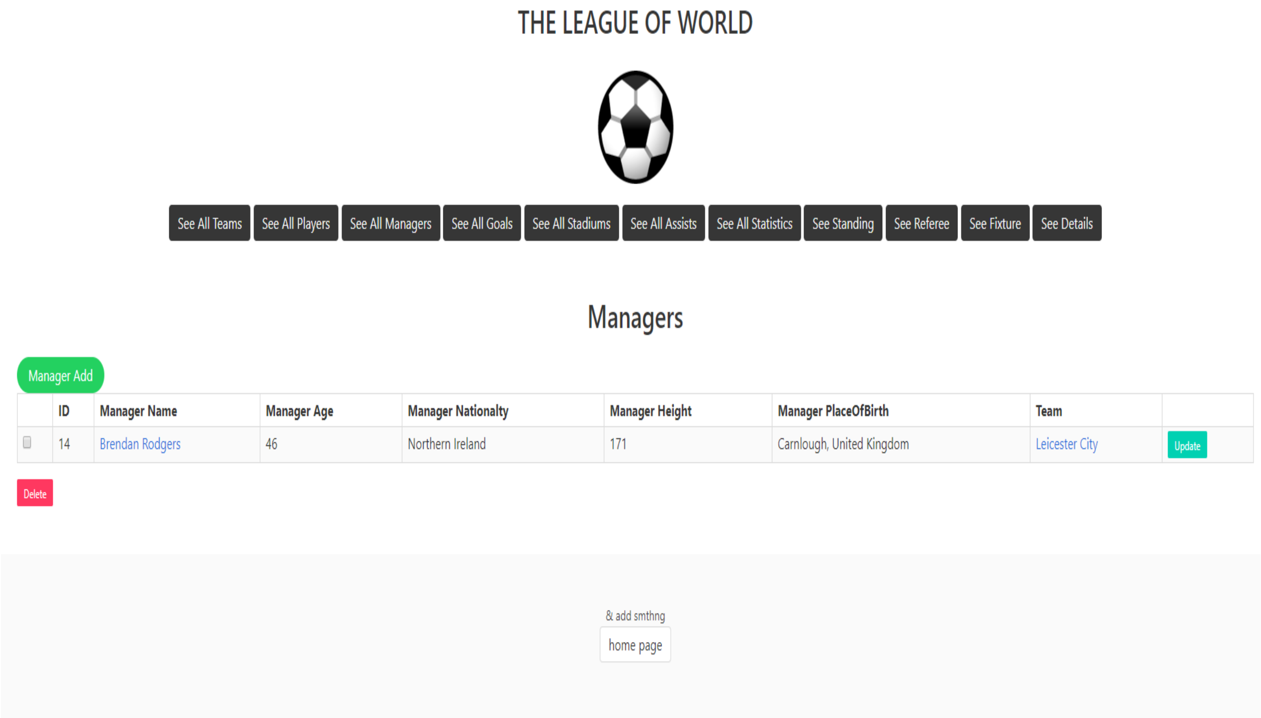
Manager Height

171

Manager PlaceOfBirth

Carnlough, United Kingdom

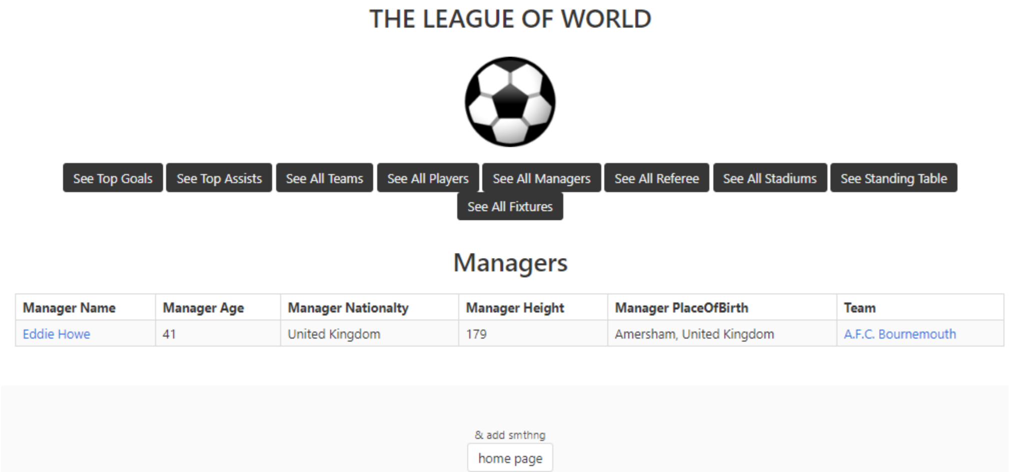
Manager update page for developer.


THE LEAGUE OF WORLD

Managers

	ID	Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team	
□	14	Brendan Rodgers	46	Northern Ireland	171	Carnlough, United Kingdom	Leicester City	Update

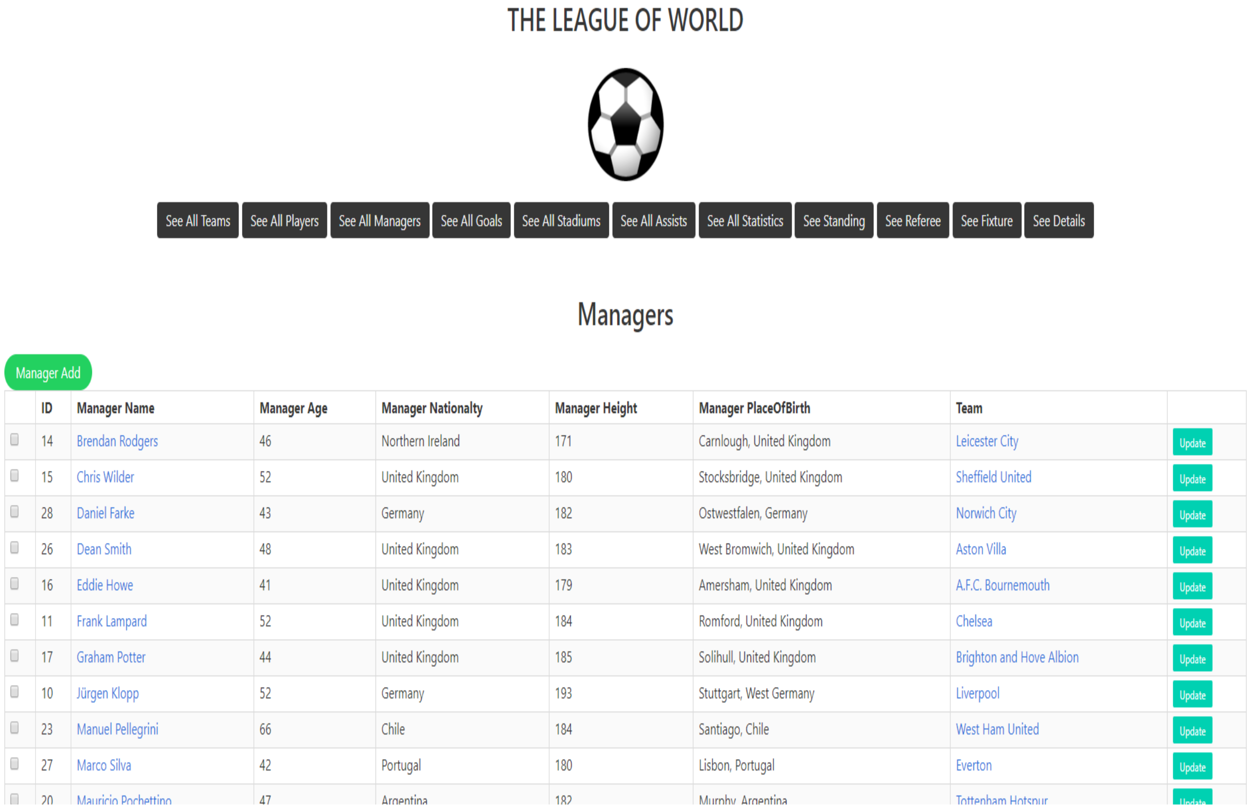
Key access for managers table in developer mode.



Managers

Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team
Eddie Howe	41	United Kingdom	179	Amersham, United Kingdom	A.F.C. Bournemouth

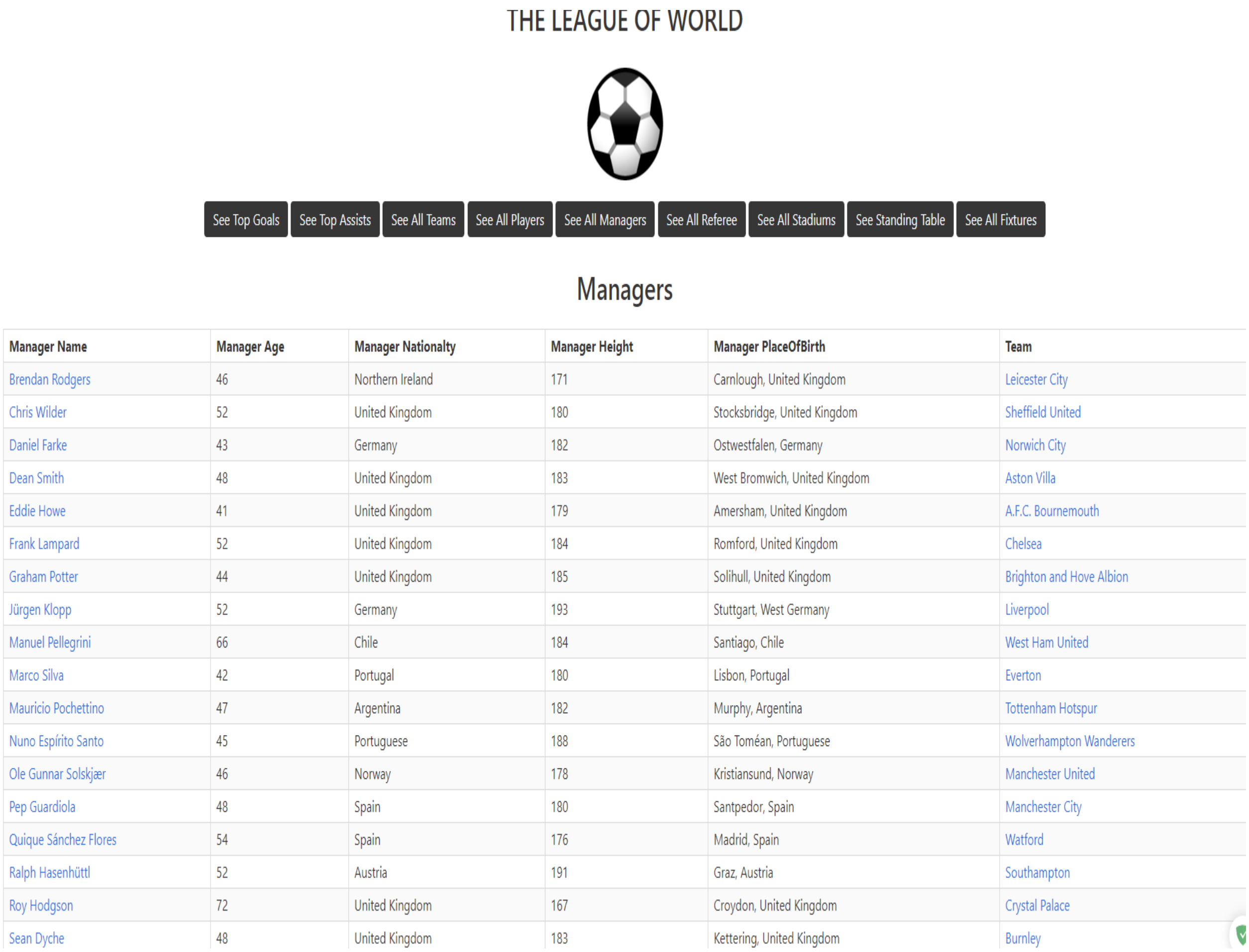
Key access for managers table in user mode.



Managers

	ID	Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team	
□	14	Brendan Rodgers	46	Northern Ireland	171	Carnlough, United Kingdom	Leicester City	Update
□	15	Chris Wilder	52	United Kingdom	180	Stocksbridge, United Kingdom	Sheffield United	Update
□	28	Daniel Farke	43	Germany	182	Ostwestfalen, Germany	Norwich City	Update
□	26	Dean Smith	48	United Kingdom	183	West Bromwich, United Kingdom	Aston Villa	Update
□	16	Eddie Howe	41	United Kingdom	179	Amersham, United Kingdom	A.F.C. Bournemouth	Update
□	11	Frank Lampard	52	United Kingdom	184	Romford, United Kingdom	Chelsea	Update
□	17	Graham Potter	44	United Kingdom	185	Solihull, United Kingdom	Brighton and Hove Albion	Update
□	10	Jürgen Klopp	52	Germany	193	Stuttgart, West Germany	Liverpool	Update
□	23	Manuel Pellegrini	66	Chile	184	Santiago, Chile	West Ham United	Update
□	27	Marco Silva	42	Portugal	180	Lisbon, Portugal	Everton	Update
□	20	Mauricio Pochettino	47	Argentina	182	Murphy, Argentina	Tottenham Hotspur	Update

The developer can see all managers in this page by clicking the 'See All Managers' button. Developer can delete and update the managers.

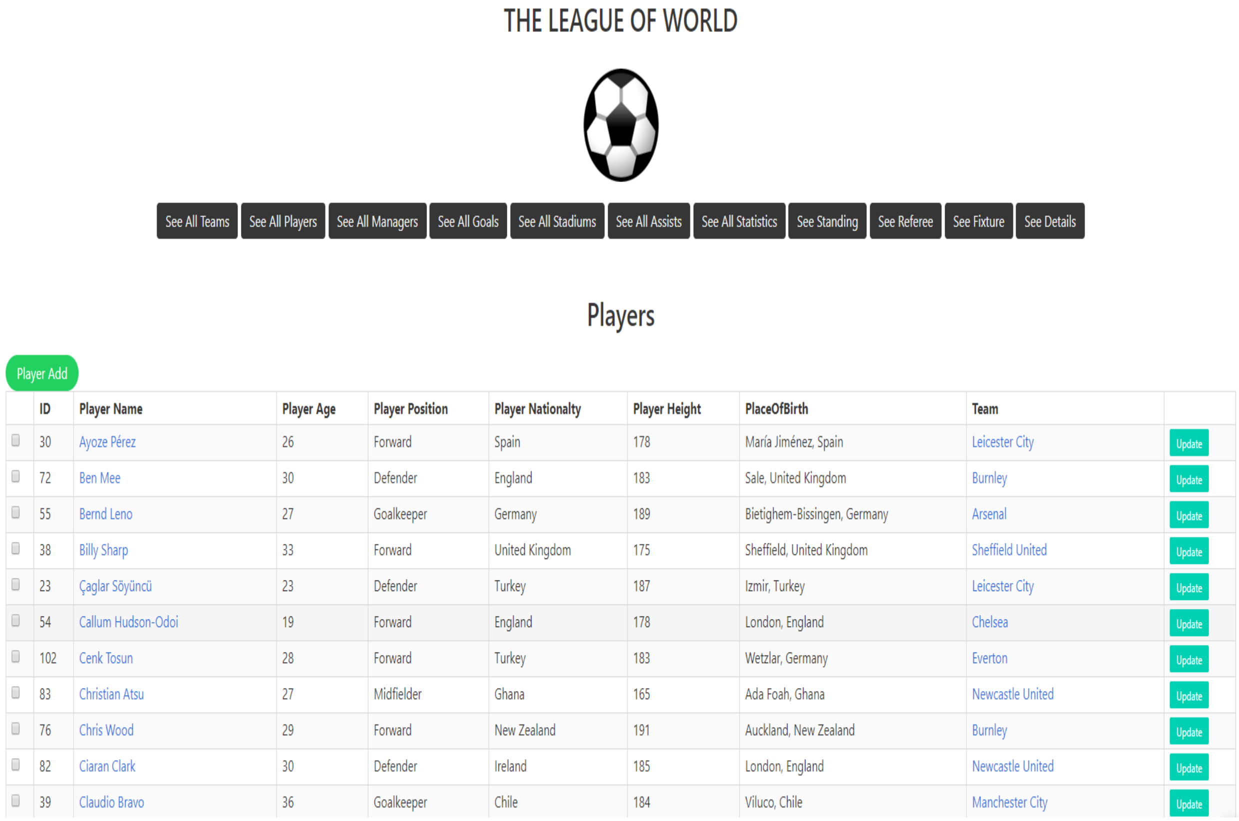


THE LEAGUE OF WORLD

Managers

Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team
Brendan Rodgers	46	Northern Ireland	171	Carnlough, United Kingdom	Leicester City
Chris Wilder	52	United Kingdom	180	Stocksbridge, United Kingdom	Sheffield United
Daniel Farke	43	Germany	182	Ostwestfalen, Germany	Norwich City
Dean Smith	48	United Kingdom	183	West Bromwich, United Kingdom	Aston Villa
Eddie Howe	41	United Kingdom	179	Amersham, United Kingdom	A.F.C. Bournemouth
Frank Lampard	52	United Kingdom	184	Romford, United Kingdom	Chelsea
Graham Potter	44	United Kingdom	185	Solihull, United Kingdom	Brighton and Hove Albion
Jürgen Klopp	52	Germany	193	Stuttgart, West Germany	Liverpool
Manuel Pellegrini	66	Chile	184	Santiago, Chile	West Ham United
Marco Silva	42	Portugal	180	Lisbon, Portugal	Everton
Mauricio Pochettino	47	Argentina	182	Murphy, Argentina	Tottenham Hotspur
Nuno Espírito Santo	45	Portuguese	188	São Toméan, Portuguese	Wolverhampton Wanderers
Ole Gunnar Solskjær	46	Norway	178	Kristiansund, Norway	Manchester United
Pep Guardiola	48	Spain	180	Santpedor, Spain	Manchester City
Quique Sánchez Flores	54	Spain	176	Madrid, Spain	Watford
Ralph Hasenhüttl	52	Austria	191	Graz, Austria	Southampton
Roy Hodgson	72	United Kingdom	167	Croydon, United Kingdom	Crystal Palace
Sean Dyche	48	United Kingdom	183	Kettering, United Kingdom	Burnley

The user can see all managers in this page by clicking the 'See All Managers' button.

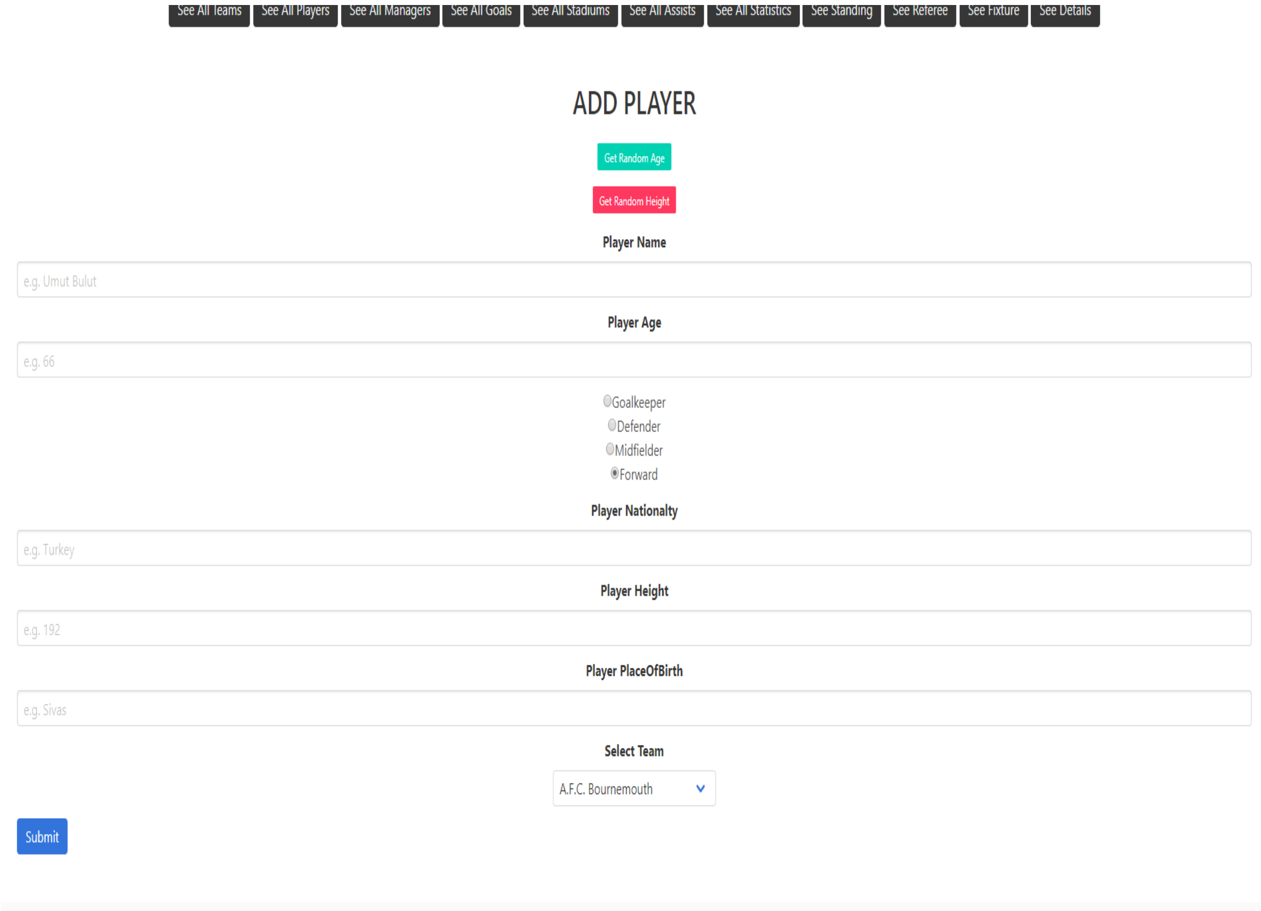


THE LEAGUE OF WORLD

Players

	ID	Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team	
☐	30	Ayoze Pérez	26	Forward	Spain	178	María Jiménez, Spain	Leicester City	Update
☐	72	Ben Mee	30	Defender	England	183	Sale, United Kingdom	Burnley	Update
☐	55	Bernd Leno	27	Goalkeeper	Germany	189	Bietighem-Bissingen, Germany	Arsenal	Update
☐	38	Billy Sharp	33	Forward	United Kingdom	175	Sheffield, United Kingdom	Sheffield United	Update
☐	23	Çaglar Söyüncü	23	Defender	Turkey	187	Izmir, Turkey	Leicester City	Update
☐	54	Callum Hudson-Odoi	19	Forward	England	178	London, England	Chelsea	Update
☐	102	Cenk Tosun	28	Forward	Turkey	183	Wetzlar, Germany	Everton	Update
☐	83	Christian Atsu	27	Midfielder	Ghana	165	Ada Foah, Ghana	Newcastle United	Update
☐	76	Chris Wood	29	Forward	New Zealand	191	Auckland, New Zealand	Burnley	Update
☐	82	Ciaran Clark	30	Defender	Ireland	185	London, England	Newcastle United	Update
☐	39	Claudio Bravo	36	Goalkeeper	Chile	184	Viluco, Chile	Manchester City	Update

The developer can see all players in this page by clicking the 'See All Players button. Developer can delete and update the players.



ADD PLAYER

Player Name

e.g. Umut Bulut

Player Age

e.g. 66

○Goalkeeper
○Defender
○Midfielder
●Forward

Player Nationalty

e.g. Turkey

Player Height

e.g. 192

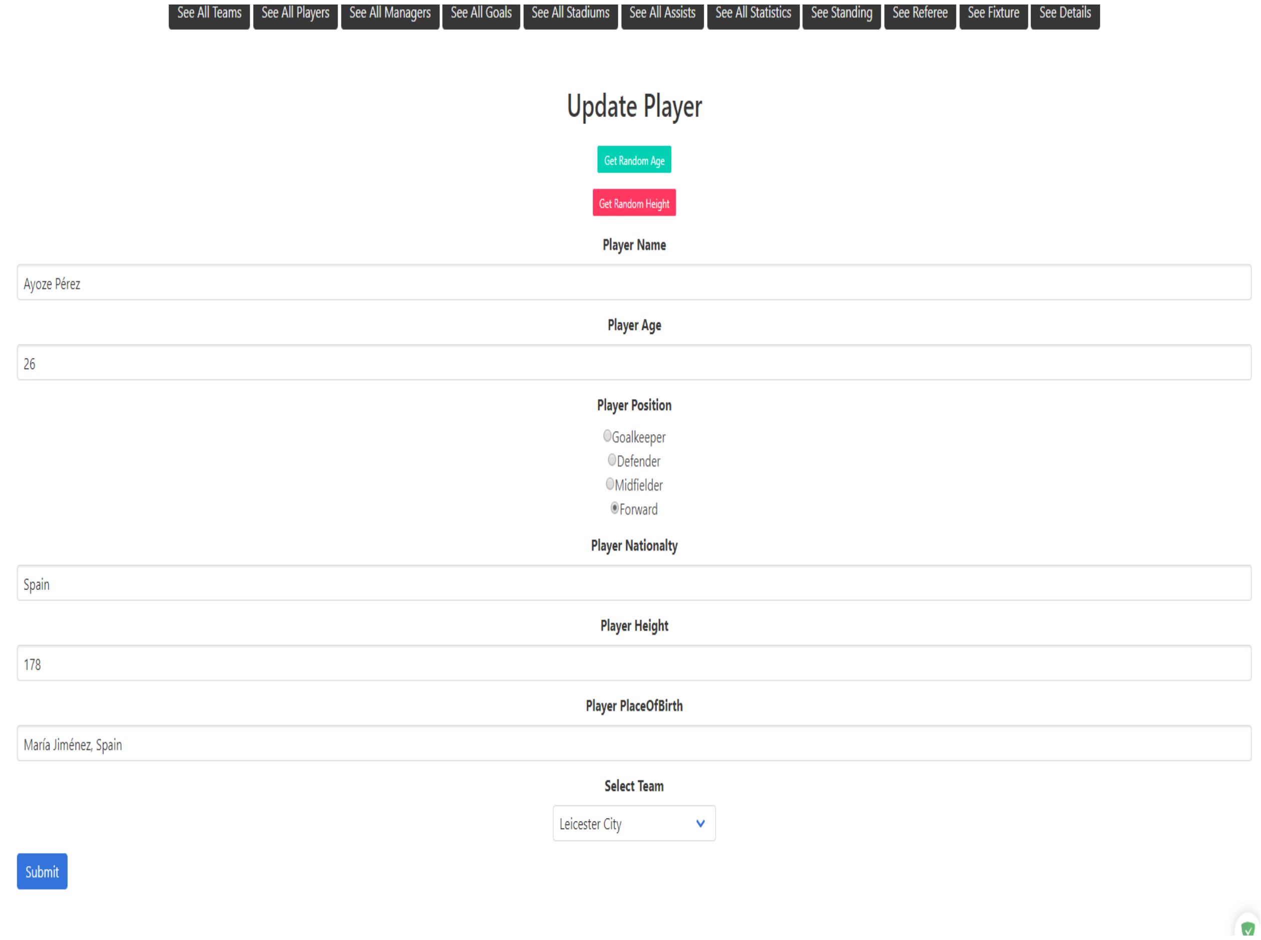
Player PlaceOfBirth

e.g. Sivas

Select Team

A.F.C. Bournemouth

Player adding page for developer.



Update Player

Player Name

Ayoze Pérez

Player Age

26

Player Position

○Goalkeeper
○Defender
○Midfielder
●Forward

Player Nationalty

Spain

Player Height

178

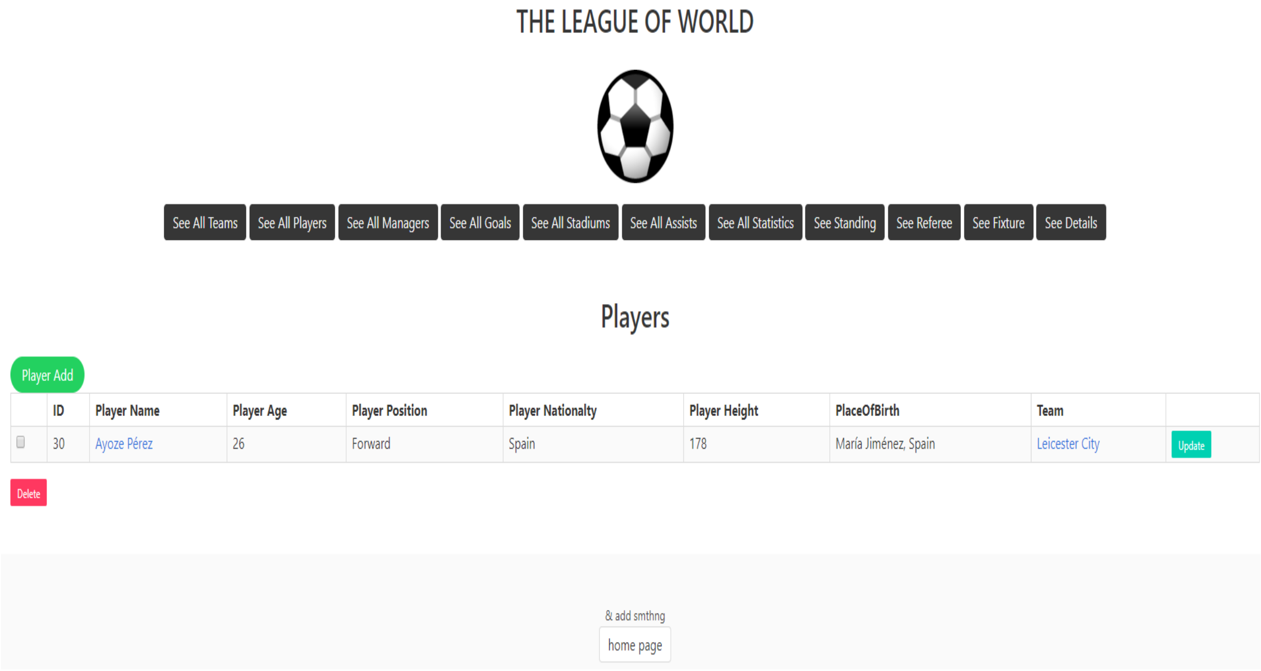
Player PlaceOfBirth

María Jiménez, Spain

Select Team

Leicester City

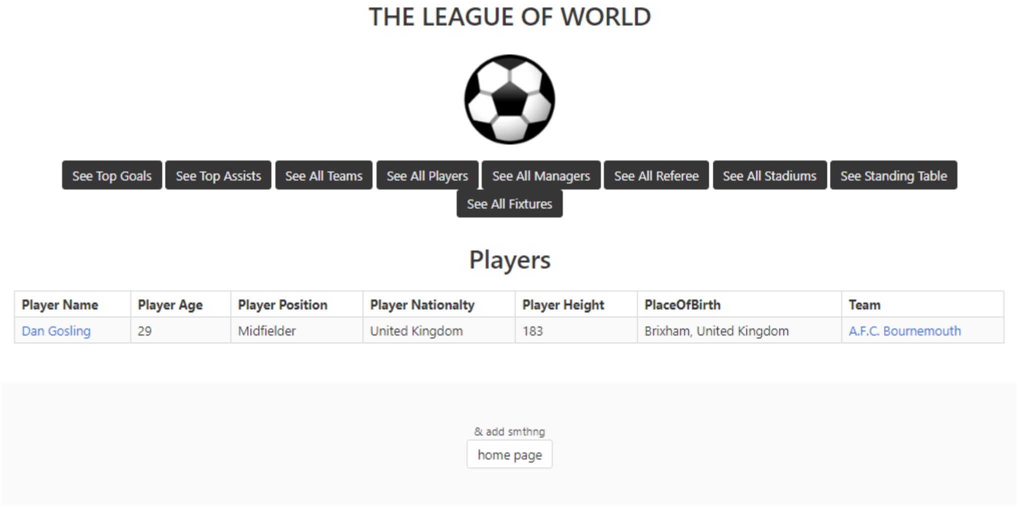
Player update page for developer.


THE LEAGUE OF WORLD

Players

	ID	Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team	
□	30	Ayoze Pérez	26	Forward	Spain	178	María Jiménez, Spain	Leicester City	Update

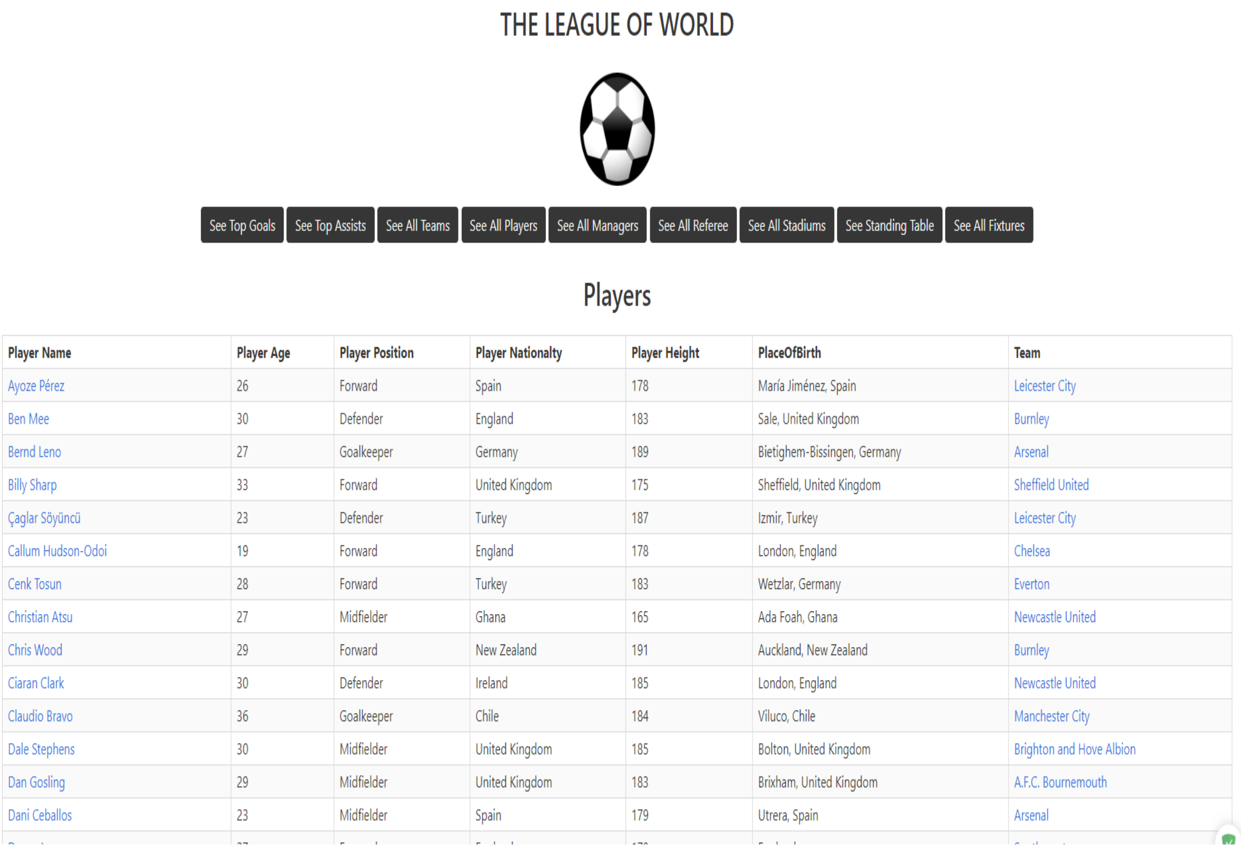
Key access for players table in developer mode.


THE LEAGUE OF WORLD

Players

Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team
Dan Gosling	29	Midfielder	United Kingdom	183	Brixham, United Kingdom	A.F.C. Bournemouth

Key access for players table in user mode.

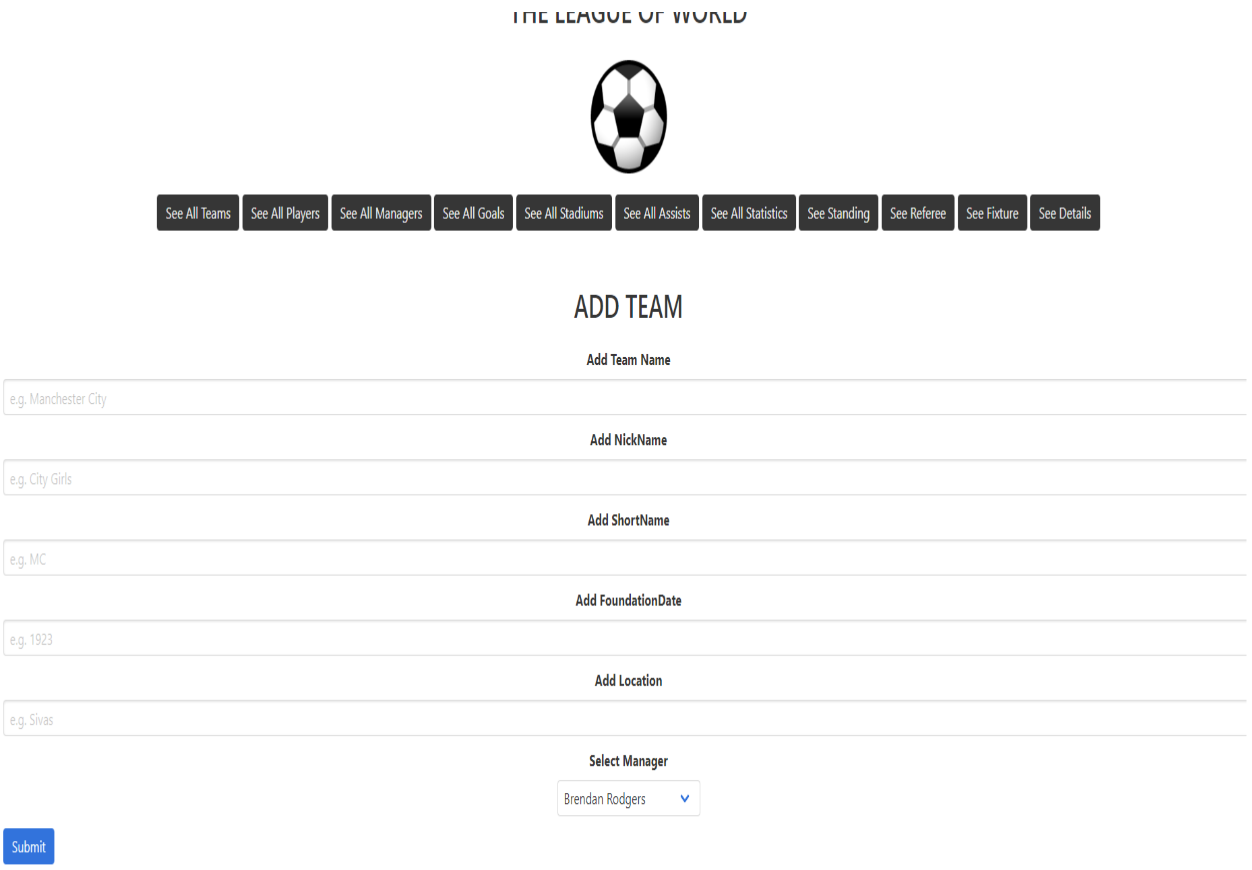


THE LEAGUE OF WORLD

Players

Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team
Ayoze Pérez	26	Forward	Spain	178	María Jiménez, Spain	Leicester City
Ben Mee	30	Defender	England	183	Sale, United Kingdom	Burnley
Bernd Leno	27	Goalkeeper	Germany	189	Bietighem-Bissingen, Germany	Arsenal
Billy Sharp	33	Forward	United Kingdom	175	Sheffield, United Kingdom	Sheffield United
Çaglar Söyüncü	23	Defender	Turkey	187	Izmir, Turkey	Leicester City
Callum Hudson-Odoi	19	Forward	England	178	London, England	Chelsea
Cenk Tosun	28	Forward	Turkey	183	Wetzlar, Germany	Everton
Christian Atsu	27	Midfielder	Ghana	165	Ada Foah, Ghana	Newcastle United
Chris Wood	29	Forward	New Zealand	191	Auckland, New Zealand	Burnley
Ciaran Clark	30	Defender	Ireland	185	London, England	Newcastle United
Claudio Bravo	36	Goalkeeper	Chile	184	Viluco, Chile	Manchester City
Dale Stephens	30	Midfielder	United Kingdom	185	Bolton, United Kingdom	Brighton and Hove Albion
Dan Gosling	29	Midfielder	United Kingdom	183	Brixham, United Kingdom	A.F.C. Bournemouth
Dani Ceballos	23	Midfielder	Spain	179	Utrera, Spain	Arsenal

The user can see all players in this page by clicking the ‘See All Players’ button.



THE LEAGUE OF WORLD

ADD TEAM

Add Team Name

e.g. Manchester City

Add NickName

e.g. City Girls

Add ShortName

e.g. MC

Add FoundationDate

e.g. 1923

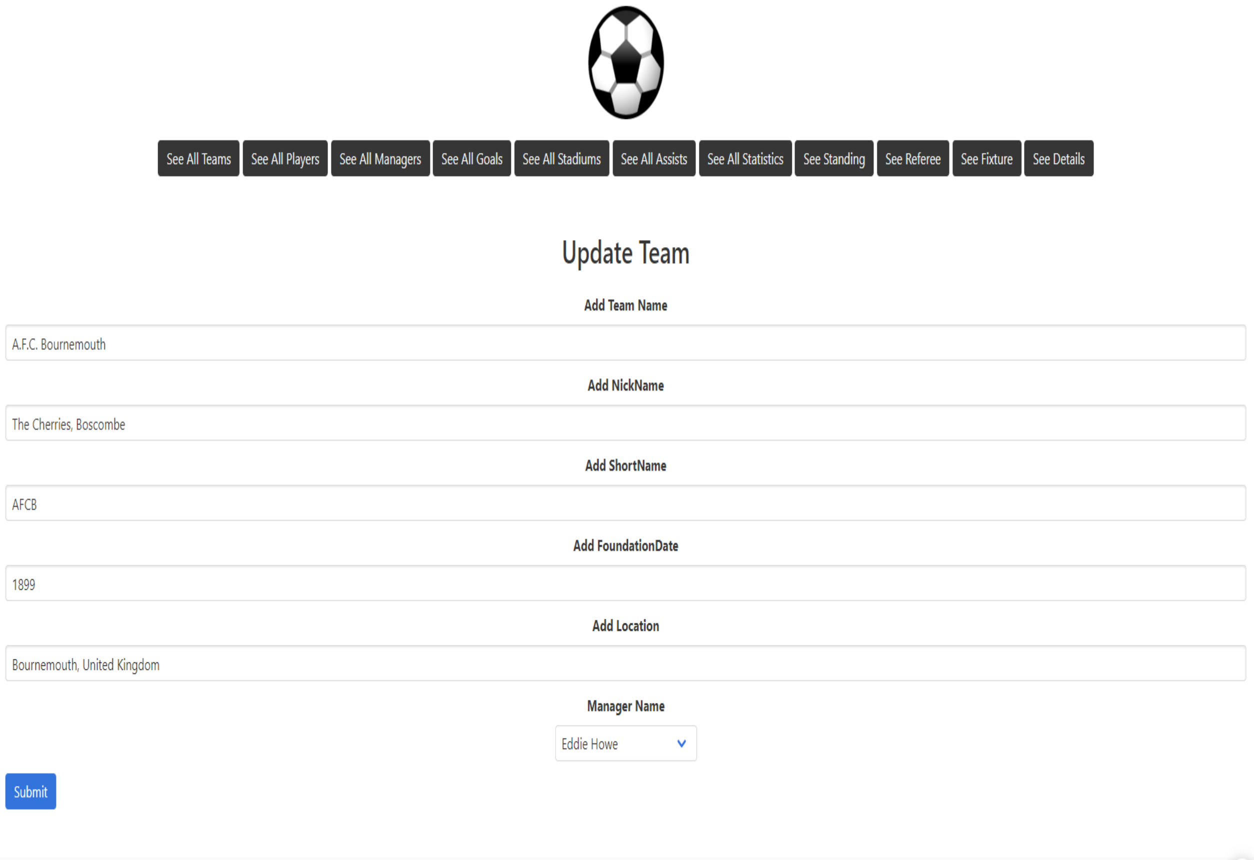
Add Location

e.g. Sivas

Select Manager

Brendan Rodgers

Team addding page for developer.



Update Team

Add Team Name

A.F.C. Bournemouth

Add NickName

The Cherries, Boscombe

Add ShortName

AFCB

Add FoundationDate

1899

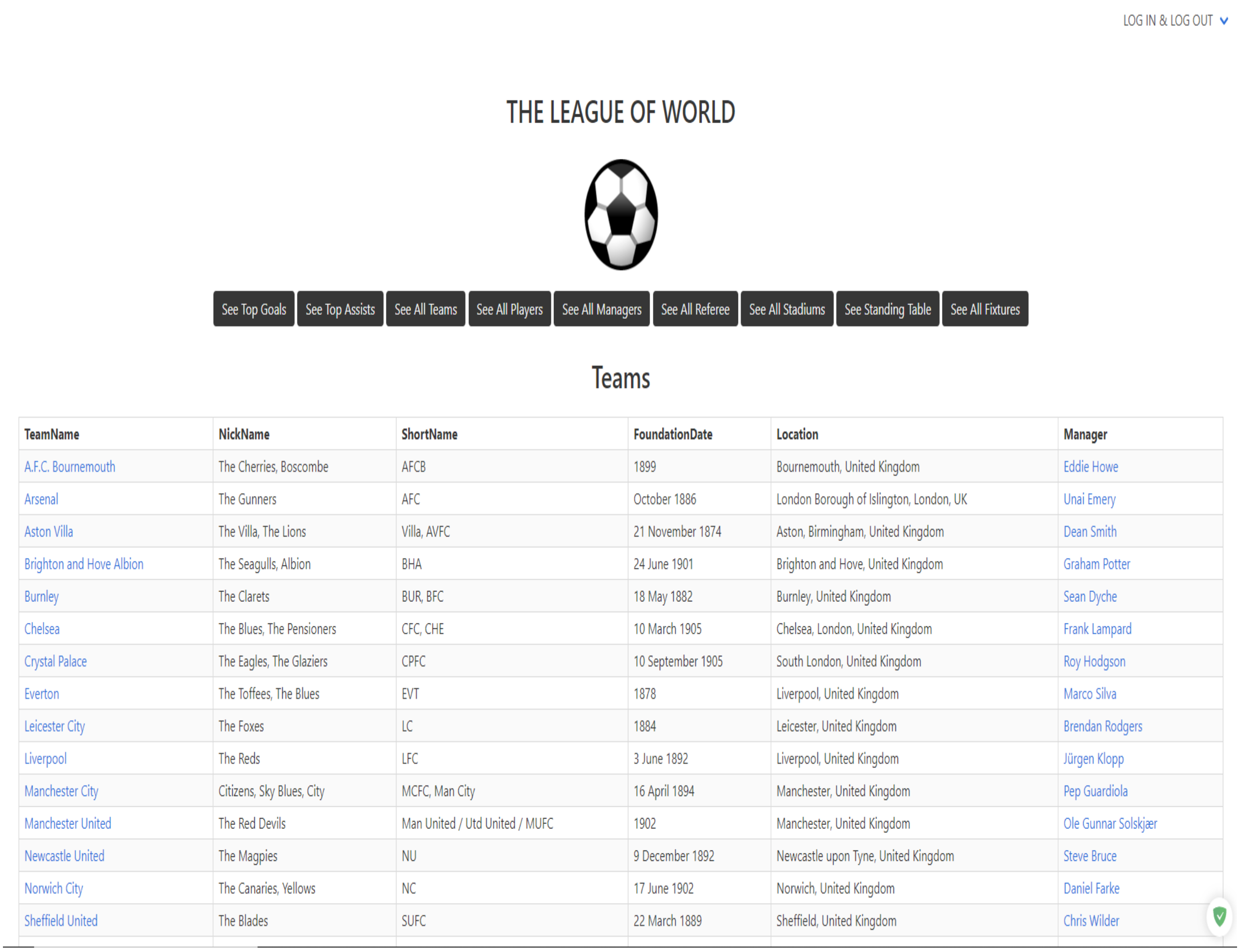
Add Location

Bournemouth, United Kingdom

Manager Name

Eddie Howe

Team update page for developer.

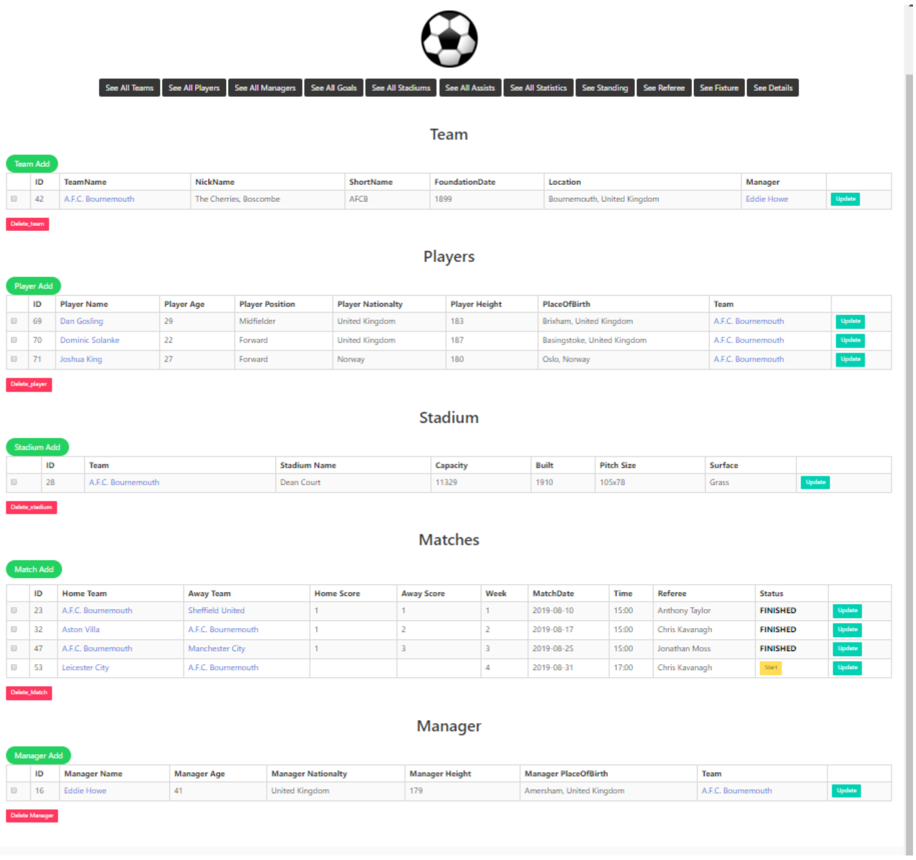


THE LEAGUE OF WORLD

Teams

TeamName	NickName	ShortName	FoundationDate	Location	Manager
A.F.C. Bournemouth	The Cherries, Boscombe	AFCB	1899	Bournemouth, United Kingdom	Eddie Howe
Arsenal	The Gunners	AFC	October 1886	London Borough of Islington, London, UK	Unai Emery
Aston Villa	The Villa, The Lions	Villa, AVFC	21 November 1874	Aston, Birmingham, United Kingdom	Dean Smith
Brighton and Hove Albion	The Seagulls, Albion	BHA	24 June 1901	Brighton and Hove, United Kingdom	Graham Potter
Burnley	The Clarets	BUR, BFC	18 May 1882	Burnley, United Kingdom	Sean Dyche
Chelsea	The Blues, The Pensioners	CFC, CHE	10 March 1905	Chelsea, London, United Kingdom	Frank Lampard
Crystal Palace	The Eagles, The Glaziers	CPFC	10 September 1905	South London, United Kingdom	Roy Hodgson
Everton	The Toffees, The Blues	EVT	1878	Liverpool, United Kingdom	Marco Silva
Leicester City	The Foxes	LC	1884	Leicester, United Kingdom	Brendan Rodgers
Liverpool	The Reds	LFC	3 June 1892	Liverpool, United Kingdom	Jürgen Klopp
Manchester City	Citizens, Sky Blues, City	MCFC, Man City	16 April 1894	Manchester, United Kingdom	Pep Guardiola
Manchester United	The Red Devils	Man United / Utd United / MUFC	1902	Manchester, United Kingdom	Ole Gunnar Solskjær
Newcastle United	The Magpies	NU	9 December 1892	Newcastle upon Tyne, United Kingdom	Steve Bruce
Norwich City	The Canaries, Yellows	NC	17 June 1902	Norwich, United Kingdom	Daniel Farke
Sheffield United	The Blades	SUFC	22 March 1889	Sheffield, United Kingdom	Chris Wilder

The user can see all teams in this page by clicking the 'See All Teams' button.



Team

	ID	TeamName	NickName	ShortName	FoundationDate	Location	Manager	
□	42	A.F.C. Bournemouth	The Cherries, Boscombe	AFCB	1899	Bournemouth, United Kingdom	Eddie Howe	Update

Players

	ID	Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team	
□	69	Dan Gosling	29	Midfielder	United Kingdom	183	Brixham, United Kingdom	A.F.C. Bournemouth	Update
□	70	Dominic Solanke	22	Forward	United Kingdom	187	Basingstoke, United Kingdom	A.F.C. Bournemouth	Update
□	71	Joshua King	27	Forward	Norway	180	Oslo, Norway	A.F.C. Bournemouth	Update

Stadium

	ID	Team	Stadium Name	Capacity	Built	Pitch Size	Surface	
□	28	A.F.C. Bournemouth	Dean Court	11329	1910	105x78	Grass	Update

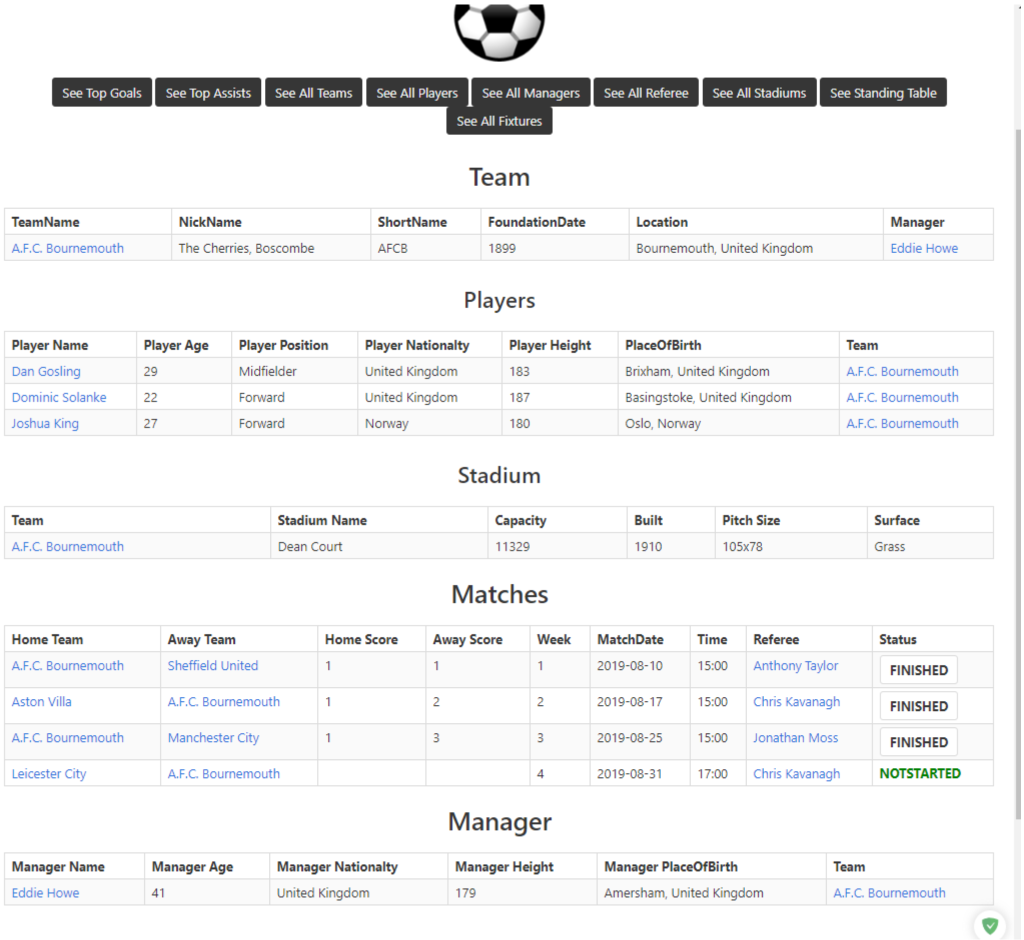
Matches

	ID	Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status	
□	23	A.F.C. Bournemouth	Sheffield United	1	1	1	2019-08-10	15:00	Anthony Taylor	FINISHED	Update
□	32	Aston Villa	A.F.C. Bournemouth	1	2	2	2019-08-17	15:00	Chris Kavanagh	FINISHED	Update
□	47	A.F.C. Bournemouth	Manchester City	1	3	3	2019-08-25	15:00	Jonathan Moss	FINISHED	Update
□	53	Leicester City	A.F.C. Bournemouth			4	2019-08-31	17:00	Chris Kavanagh	Start	Update

Manager

	ID	Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team	
□	16	Eddie Howe	41	United Kingdom	179	Amersham, United Kingdom	A.F.C. Bournemouth	Update

Key access for teams table in developer mode.



Team

TeamName	NickName	ShortName	FoundationDate	Location	Manager
A.F.C. Bournemouth	The Cherries, Boscombe	AFCB	1899	Bournemouth, United Kingdom	Eddie Howe

Players

Player Name	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth	Team
Dan Gosling	29	Midfielder	United Kingdom	183	Brixham, United Kingdom	A.F.C. Bournemouth
Dominic Solanke	22	Forward	United Kingdom	187	Basingstoke, United Kingdom	A.F.C. Bournemouth
Joshua King	27	Forward	Norway	180	Oslo, Norway	A.F.C. Bournemouth

Stadium

Team	Stadium Name	Capacity	Built	Pitch Size	Surface
A.F.C. Bournemouth	Dean Court	11329	1910	105x78	Grass

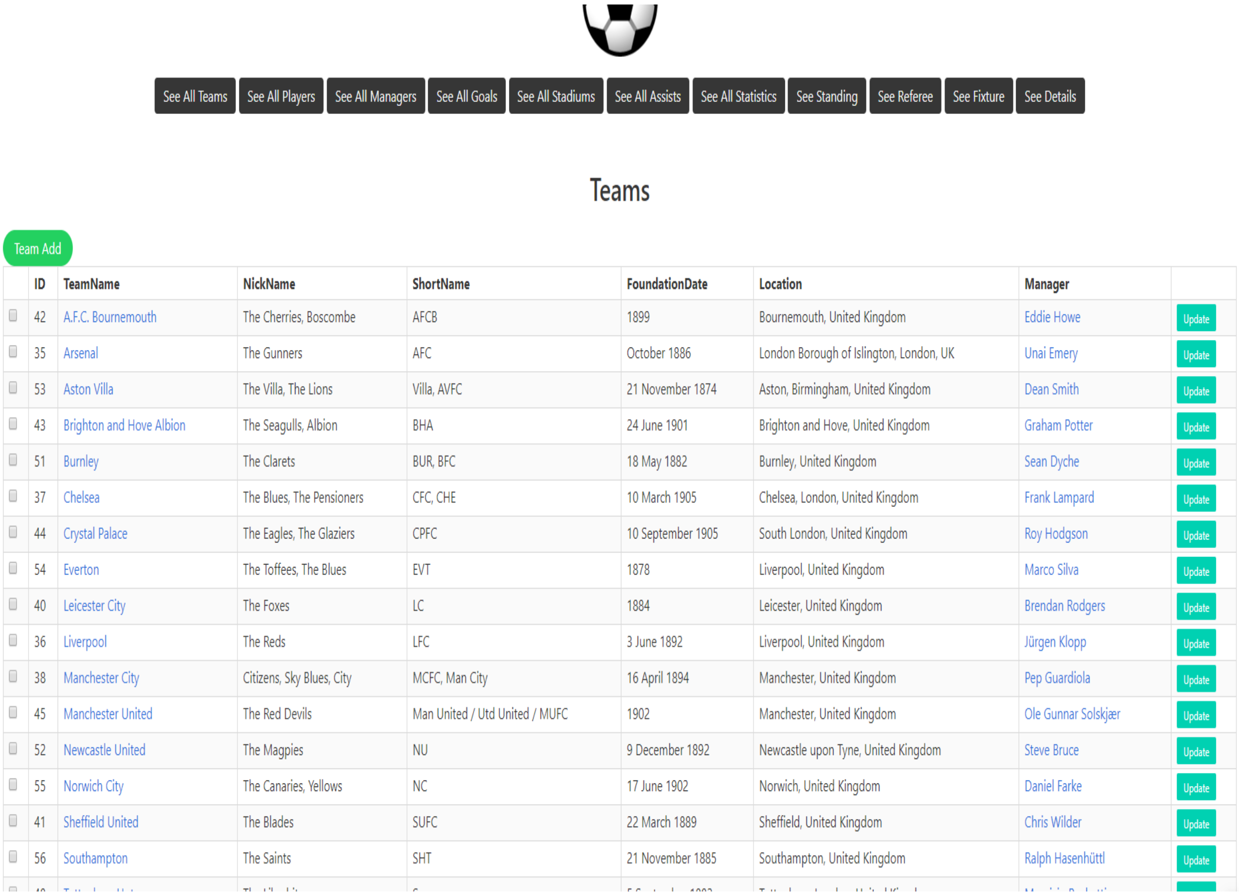
Matches

Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status
A.F.C. Bournemouth	Sheffield United	1	1	1	2019-08-10	15:00	Anthony Taylor	FINISHED
Aston Villa	A.F.C. Bournemouth	1	2	2	2019-08-17	15:00	Chris Kavanagh	FINISHED
A.F.C. Bournemouth	Manchester City	1	3	3	2019-08-25	15:00	Jonathan Moss	FINISHED
Leicester City	A.F.C. Bournemouth			4	2019-08-31	17:00	Chris Kavanagh	NOTSTARTED

Manager

Manager Name	Manager Age	Manager Nationalty	Manager Height	Manager PlaceOfBirth	Team
Eddie Howe	41	United Kingdom	179	Amersham, United Kingdom	A.F.C. Bournemouth

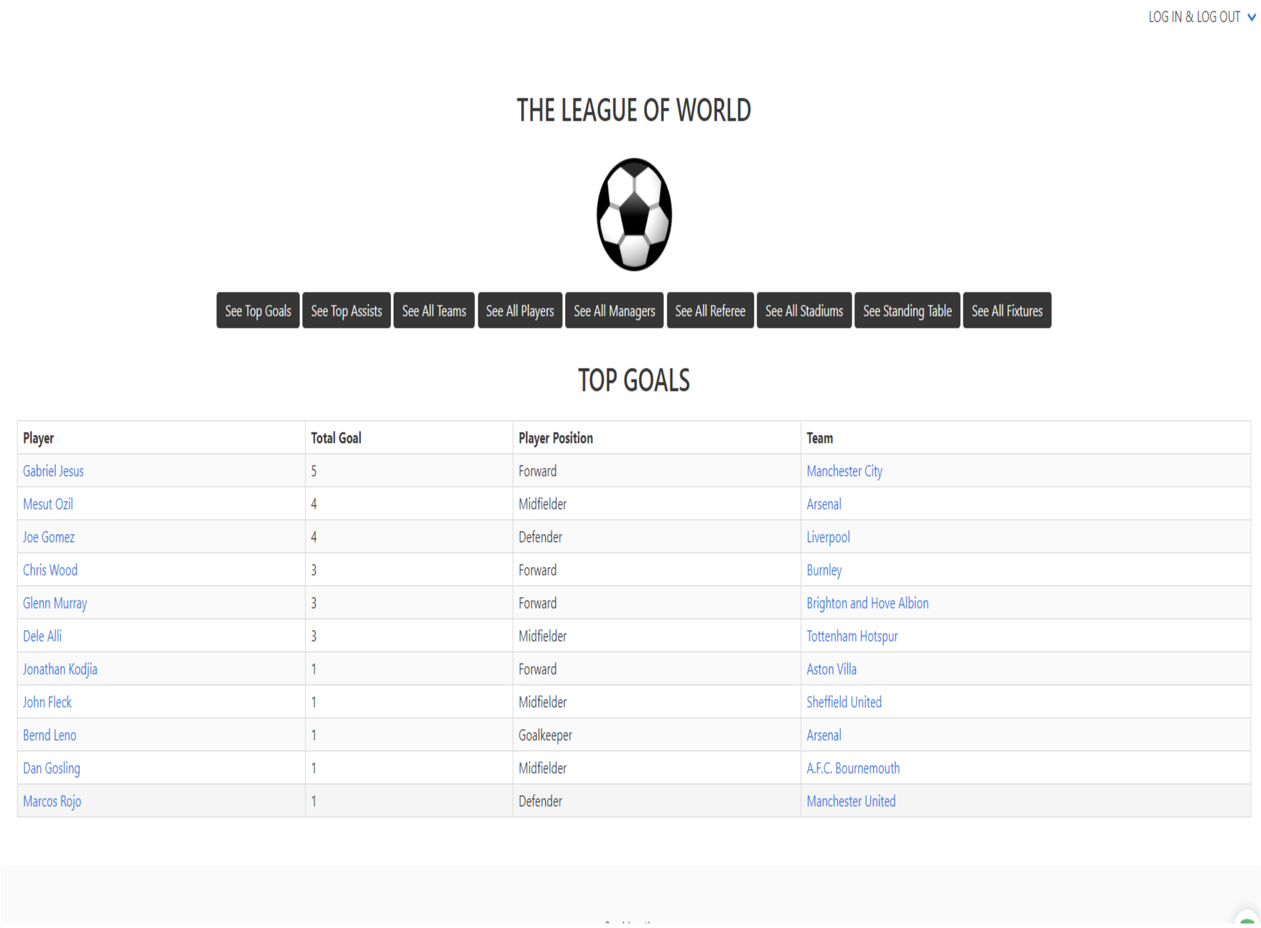
Key access for teams table in user mode.



Teams

	ID	TeamName	NickName	ShortName	FoundationDate	Location	Manager	
□	42	A.F.C. Bournemouth	The Cherries, Boscombe	AFCB	1899	Bournemouth, United Kingdom	Eddie Howe	Update
□	35	Arsenal	The Gunners	AFC	October 1886	London Borough of Islington, London, UK	Unai Emery	Update
□	53	Aston Villa	The Villa, The Lions	Villa, AVFC	21 November 1874	Aston, Birmingham, United Kingdom	Dean Smith	Update
□	43	Brighton and Hove Albion	The Seagulls, Albion	BHA	24 June 1901	Brighton and Hove, United Kingdom	Graham Potter	Update
□	51	Burnley	The Clarets	BUR, BFC	18 May 1882	Burnley, United Kingdom	Sean Dyche	Update
□	37	Chelsea	The Blues, The Pensioners	CFC, CHE	10 March 1905	Chelsea, London, United Kingdom	Frank Lampard	Update
□	44	Crystal Palace	The Eagles, The Glaziers	CPFC	10 September 1905	South London, United Kingdom	Roy Hodgson	Update
□	54	Everton	The Toffees, The Blues	EVT	1878	Liverpool, United Kingdom	Marco Silva	Update
□	40	Leicester City	The Foxes	LC	1884	Leicester, United Kingdom	Brendan Rodgers	Update
□	36	Liverpool	The Reds	LFC	3 June 1892	Liverpool, United Kingdom	Jürgen Klopp	Update
□	38	Manchester City	Citizens, Sky Blues, City	MCFC, Man City	16 April 1894	Manchester, United Kingdom	Pep Guardiola	Update
□	45	Manchester United	The Red Devils	Man United / Utd United / MUFC	1902	Manchester, United Kingdom	Ole Gunnar Solskjær	Update
□	52	Newcastle United	The Magpies	NU	9 December 1892	Newcastle upon Tyne, United Kingdom	Steve Bruce	Update
□	55	Norwich City	The Canaries, Yellows	NC	17 June 1902	Norwich, United Kingdom	Daniel Farke	Update
□	41	Sheffield United	The Blades	SUFC	22 March 1889	Sheffield, United Kingdom	Chris Wilder	Update
□	56	Southampton	The Saints	SHT	21 November 1885	Southampton, United Kingdom	Ralph Hasenhüttl	Update

The developer can see all teams in this page by clicking the 'See All Teams' button. Developer can delete and update the teams.

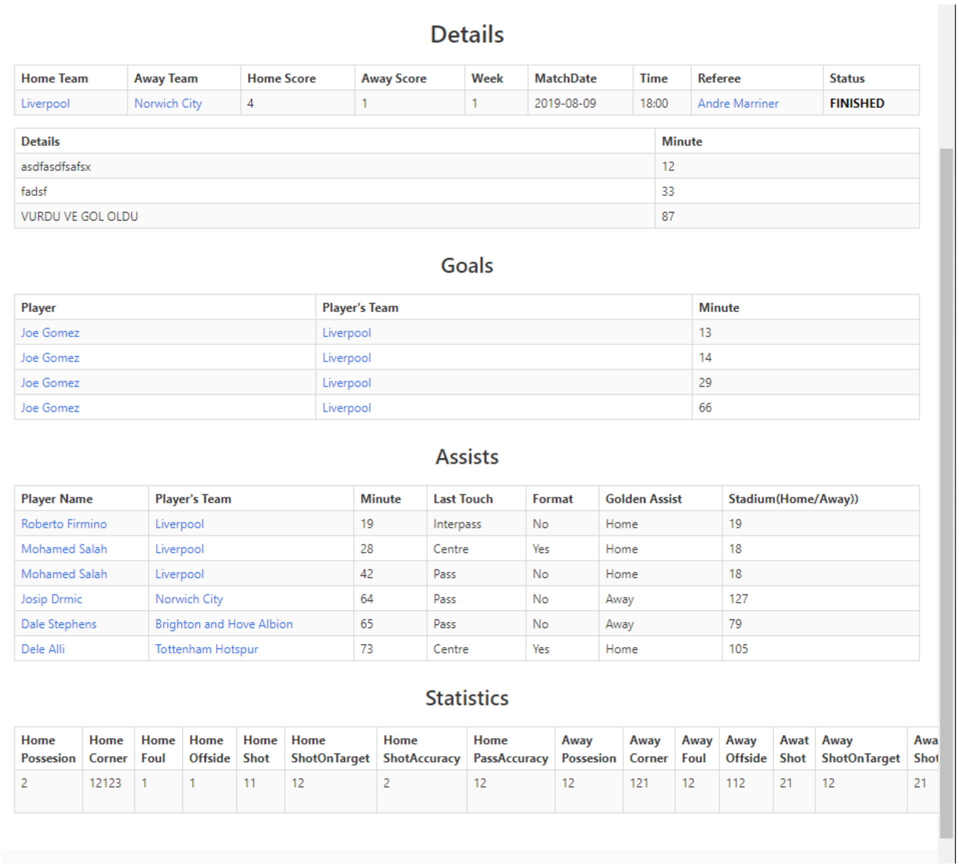


THE LEAGUE OF WORLD

TOP GOALS

Player	Total Goal	Player Position	Team
Gabriel Jesus	5	Forward	Manchester City
Mesut Ozil	4	Midfielder	Arsenal
Joe Gomez	4	Defender	Liverpool
Chris Wood	3	Forward	Burnley
Glenn Murray	3	Forward	Brighton and Hove Albion
Dele Alli	3	Midfielder	Tottenham Hotspur
Jonathan Kodjia	1	Forward	Aston Villa
John Fleck	1	Midfielder	Sheffield United
Bernd Leno	1	Goalkeeper	Arsenal
Dan Gosling	1	Midfielder	A.F.C. Bournemouth
Marcos Rojo	1	Defender	Manchester United

The user can see top goal player in this page by clicking the 'See Top Goal' button.



Details

Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status
Liverpool	Norwich City	4	1	1	2019-08-09	18:00	Andre Marriner	**FINISHED**

Details	Minute
asdfasdfsafsx	12
fadsf	33
VURDU VE GOL OLDU	87

Goals

Player	Player's Team	Minute
Joe Gomez	Liverpool	13
Joe Gomez	Liverpool	14
Joe Gomez	Liverpool	29
Joe Gomez	Liverpool	66

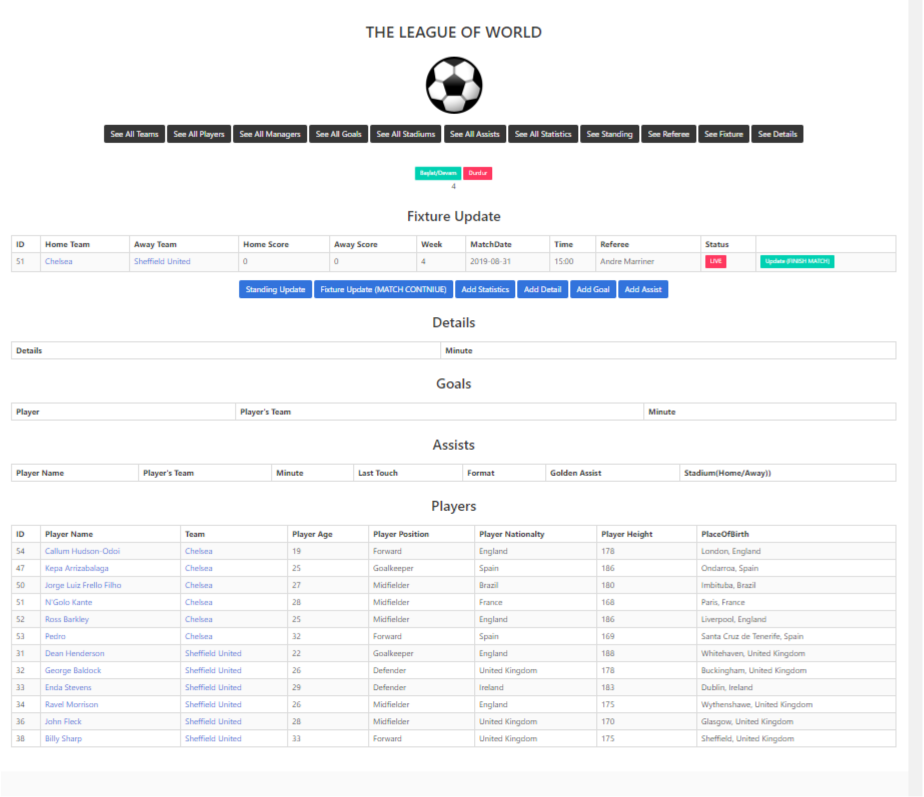
Assists

Player Name	Player's Team	Minute	Last Touch	Format	Golden Assist	Stadium(Home/Away))
Roberto Firmino	Liverpool	19	Interpass	No	Home	19
Mohamed Salah	Liverpool	28	Centre	Yes	Home	18
Mohamed Salah	Liverpool	42	Pass	No	Home	18
Josip Drmic	Norwich City	64	Pass	No	Away	127
Dale Stephens	Brighton and Hove Albion	65	Pass	No	Away	79
Dele Alli	Tottenham Hotspur	73	Centre	Yes	Home	105

Statistics

Home Possesion	Home Corner	Home Foul	Home Offside	Home Shot	Home ShotOnTarget	Home ShotAccuracy	Home PassAccuracy	Away Possesion	Away Corner	Away Foul	Away Offside	Awat Shot	Away ShotOnTarget	Awa Shot
2	12123	1	1	11	12	2	12	12	121	12	112	21	12	21

The user can see detailed match information in this page by clicking the 'Finished or Live buttons'.



THE LEAGUE OF WORLD

Fixture Update

ID	Home Team	Away Team	Home Score	Away Score	Week	MatchDate	Time	Referee	Status	
51	Chelsea	Sheffield United	0	0	4	2019-08-31	15:00	Andre Marriner	LIVE	Update (FINISH MATCH)

Details

Details	Minute

Goals

Player	Player's Team	Minute

Assists

Player Name	Player's Team	Minute	Last Touch	Format	Golden Assist	Stadium(Home/Away))

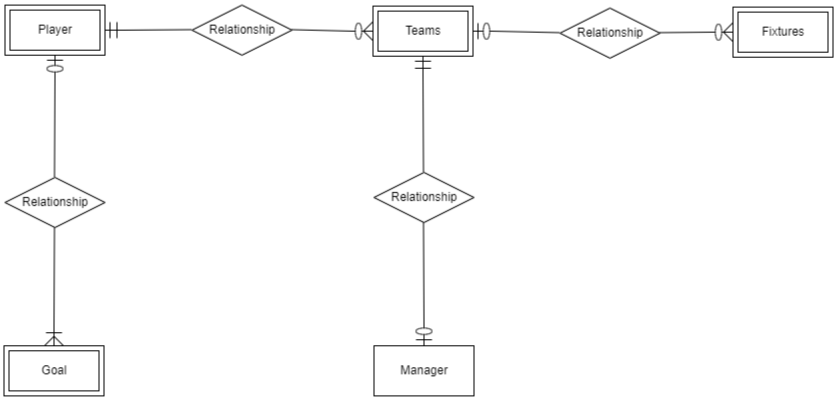
Players

ID	Player Name	Team	Player Age	Player Position	Player Nationalty	Player Height	PlaceOfBirth
54	Callum Hudson-Odoi	Chelsea	19	Forward	England	178	London, England
47	Kepa Arrizabalaga	Chelsea	25	Goalkeeper	Spain	186	Ondarroa, Spain
50	Jorge Luiz Frello Filho	Chelsea	27	Midfielder	Brazil	180	Imbituba, Brazil
51	N'Golo Kante	Chelsea	28	Midfielder	France	168	Paris, France
52	Ross Barkley	Chelsea	25	Midfielder	England	186	Liverpool, England
53	Pedro	Chelsea	32	Forward	Spain	169	Santa Cruz de Tenerife, Spain
31	Dean Henderson	Sheffield United	22	Goalkeeper	England	188	Whitehaven, United Kingdom
32	George Baldock	Sheffield United	26	Defender	United Kingdom	178	Buckingham, United Kingdom
33	Enda Stevens	Sheffield United	29	Defender	Ireland	183	Dublin, Ireland
34	Ravel Morrison	Sheffield United	26	Midfielder	England	175	Wythenshawe, United Kingdom
36	John Fleck	Sheffield United	28	Midfielder	United Kingdom	170	Glasgow, United Kingdom
38	Billy Sharp	Sheffield United	33	Forward	United Kingdom	175	Sheffield, United Kingdom

The developer can start and control the match by clicking the 'Live button' in fixture table.

Parts Implemented by Muhammed Enes Tırnakçı

Database Design



Code

```
"""

CREATE TABLE IF NOT EXISTS  Teams
(
         ID SERIAL PRIMARY KEY,
         Teamname VARCHAR(30) NOT NULL,
         NickName VARCHAR(30),
         ShortName VARCHAR(30) NOT NULL,
         FoundationDate VARCHAR(30),
         ManagerID INTEGER REFERENCES Manager (ID) ON DELETE cascade,
         Location VARCHAR(50)
)
""",
"""

CREATE TABLE IF NOT EXISTS  Player
(
         ID SERIAL PRIMARY KEY,
         PlayerName VARCHAR(30) NOT NULL,
         PlayerAge INTEGER NOT NULL,
         Position VARCHAR(30),
         PlayerNationalty VARCHAR(30) NOT NULL,
         PlayerHeight INTEGER NOT NULL,
         PlaceOfBirth VARCHAR(30) NOT NULL,
         TeamID INTEGER NOT NULL REFERENCES Teams (ID) ON DELETE cascade
)
""",
"""

CREATE TABLE IF NOT EXISTS  Manager
(
         ID SERIAL PRIMARY KEY,
         Name VARCHAR(30) NOT NULL,
         Age INTEGER NOT NULL,
         Nationalty VARCHAR(30) NOT NULL,
         Height INTEGER NOT NULL,
         PlaceOfBirth VARCHAR(30) NOT NULL,
         teamid integer REFERENCES teams (ID) ON DELETE cascade
)
""",
"""

CREATE TABLE IF NOT EXISTS  Goal
(
   ID serial,
   PlayerID integer NOT NULL REFERENCES Player (ID) ON DELETE cascade,
   MatchID integer NOT NULL REFERENCES Fixtures (ID) ON DELETE cascade,
                        Minute INTEGER NOT NULL,
   PRIMARY KEY (ID)
)
""",
```

These methods create tables.

```
def Team_add(self, TeamName, NickName, ShortName, FoundationDate, ManagerID,Location):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ INSERT INTO Teams(TeamName, NickName, ShortName, FoundationDate,
ManagerID,Location) VALUES(%s,%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([TeamName, NickName, ShortName, FoundationDate,
ManagerID,Location]))
```

This method adds new team.

```
def Team_update(self, TeamID, TeamName, NickName, ShortName, FoundationDate,
ManagerID,Location):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Teams Set Teamname=%s, NickName=%s, ShortName=%s,
FoundationDate=%s, ManagerID=%s,Location=%s  Where ID=%s;"""
         cursor.execute(statement,([TeamName, NickName, ShortName, FoundationDate,
ManagerID, Location, TeamID]))
```

This method updates teams.

```
def Team_delete(self, TeamID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ DELETE FROM Teams WHERE ID = %s;"""
         cursor.execute(statement,[TeamID])
```

This method deletes teams.

```
def Goal_add(self, PlayerID, MatchID, Minute):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ INSERT INTO Goal(PlayerID, MatchID, Minute) VALUES(%s,%s,%s);"""
         cursor.execute(statement,([PlayerID, MatchID,Minute]))
```

This method adds new goal.

```
def Goal_update(self, GoalID, PlayerID, MatchID, Minute):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Goal Set PlayerID=%s, MatchID=%s, Minute=%s Where ID=%s;"""
         cursor.execute(statement,([PlayerID, MatchID, Minute, GoalID]))
```

This method updates goals.

```
def Goal_delete(self, GoalID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ DELETE FROM Goal WHERE ID = %s;"""
         cursor.execute(statement,[GoalID])
```

This method deletes goals.

```
def Player_add(self, PlayerName, PlayerAge, Position, PlayerNationalty, PlayerHeight,
PlaceOfBirth, TeamID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """
                           INSERT INTO Player(PlayerName, PlayerAge, Position,
PlayerNationalty, PlayerHeight, PlaceOfBirth, TeamID) VALUES(%s,%s,%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([PlayerName, PlayerAge, Position, PlayerNationalty,
PlayerHeight, PlaceOfBirth, TeamID]))
```

This method adds new player.

```
def Player_delete(self, PlayerID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ DELETE FROM Player WHERE ID = %s;"""
         cursor.execute(statement,[PlayerID])
```

This method deletes players.

```
def Player_update(self, PlayerID, PlayerName, PlayerAge, Position, PlayerNationalty,
PlayerHeight, PlaceOfBirth, TeamID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Player Set PlayerName=%s, PlayerAge=%s, Position=%s,
PlayerNationalty=%s, PlayerHeight=%s, PlaceOfBirth=%s, TeamID=%s Where ID=%s;"""
         cursor.execute(statement,([PlayerName, PlayerAge, Position, PlayerNationalty,
PlayerHeight, PlaceOfBirth, TeamID, PlayerID]))
```

This method updates players.

```
def Manager_add(self, Name, Age, Nationalty, Height, PlaceOfBirth):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """INSERT INTO Manager(Name, Age, Nationalty, Height, PlaceOfBirth)
VALUES(%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([Name, Age, Nationalty, Height, PlaceOfBirth]))
```

This method adds new manager.

```
def Manager_update(self, ManagerID, Name, Age, Nationalty, Height, PlaceOfBirth):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Manager Set Name=%s, Age=%s, Nationalty=%s, Height=%s,
PlaceOfBirth=%s Where ID=%s;"""
         cursor.execute(statement,([Name, Age, Nationalty, Height, PlaceOfBirth,
ManagerID]))
```

This method updates managers.

```
def Manager_delete(self, ManagerID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ DELETE FROM Manager WHERE ID = %s;"""
         cursor.execute(statement,[ManagerID])
```

This method deletes managers.

```
def Team(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select 
Teams.ID,Teamname,NickName,ShortName,FoundationDate,Name,Location, ManagerID FROM 
Teams,Manager WHERE Teams.ID=Teams.ID and Manager.ID=ManagerID ORDER BY Teamname ASC;"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list

def Player(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select 
Player.ID,PlayerName,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOfBirth,Teamname,Te
 FROM Player,Teams WHERE Player.ID=Player.ID and Teams.ID=TeamID ORDER BY 
PlayerName,Teamname ASC;"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list

def Goal(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Goal.ID, PlayerName, MatchID,Minute FROM 
Goal,Fixtures,Player WHERE Goal.ID=Goal.ID and Player.ID=PlayerID and MatchID=Fixtures.ID 
ORDER BY PlayerID,MatchID ASC;"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list

def Manager(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Manager.id, Name, Age,Nationalty,height,placeofbirth, 
Teamname,Teams.id FROM Manager,teams Where ManagerID=manager.id ORDER BY Name"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

These methods are used for getting all teams, players, goals, and managers by join operation.

```
def Goal_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Goal where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list

def Manager_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Manager where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list

def Team_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Teams where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list

def Player_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Player where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list
```

These methods are used for getting information of goals, managers, players, teams that will be updated and show in the .html page.

```
def Player_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select
Player.ID,PlayerName,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOfBirth,Teamname,Te
 FROM Player,Teams WHERE Player.ID=%s and Teams.ID=TeamID ORDER BY Teamname ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list

def Team_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select
Teams.ID,Teamname,NickName,ShortName,FoundationDate,Name,Location, ManagerID FROM
Teams,Manager WHERE Teams.ID=%s and Manager.ID=ManagerID ORDER BY Teamname ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list

def Goal_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Goal.ID, PlayerName, MatchID,Minute FROM
Goal,Fixtures,Player WHERE Goal.ID=%s and Player.ID=PlayerID and MatchID=Fixtures.ID ORDER
BY PlayerID,MatchID ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list

def Manager_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """SELECT manager.id, manager.name, manager.age, manager.nationalty,
manager.height, manager.placeofbirth, teams.teamname from manager left join teams on
manager.id = teams.managerid where manager.id=%s Order By Name"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method are used for accessing indivual tuple by its primal key.

```
def Top_goal(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Player.ID, PlayerName,
count(PlayerID),Position,Teamname,Player.TeamID FROM Goal,Player,Teams WHERE
Goal.ID=Goal.ID and Player.ID=PlayerID and Player.TeamID=Teams.ID Group BY
PlayerName,player.id,Teams.Teamname ORDER BY count(PlayerID) DESC;"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used to show top goal players in user mode by countig each player how many goal that they have.

```
def Manager_user(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """SELECT manager.id, manager.name, manager.age, manager.nationalty,
manager.height, manager.placeofbirth, teams.teamname, teams.id from manager left join teams
on manager.id = teams.managerid where manager.id=manager.id Order By Name"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used for showing all managers with their teams by joining Teams and Manager table in user mode.

```
def Team_user_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select
Teams.ID,Teamname,NickName,ShortName,FoundationDate,Name,Location, ManagerID FROM
Teams,Manager WHERE Teams.ID=%s and Manager.ID=ManagerID ORDER BY Teamname ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used for accessing one single team by their primal key, and to show its manager, I used joined Manager table and Teams table.

```
def Player_team_user(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select
Player.ID,PlayerName,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOfBirth,Teamname,Te
 FROM Player,Teams WHERE Player.ID=Player.ID and Teams.ID=TeamID and Teams.ID=%s  ORDER BY
Teamname ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method gets all informations about all player in one team, and that one team will be accessed by user using key accessing.

```
def Player_team(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select 
Player.ID,PlayerName,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOfBirth,Teamname,Te
 FROM Player,Teams WHERE Player.ID=Player.ID and Teams.ID=TeamID and Teams.ID=%s  ORDER BY 
Teamname ASC;"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method gets all informations about all player in one team, and that one team will be accessed by developer using key accessing.

```
def Goal_user(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select distinct Goal.ID, PlayerName, Teamname,  Goal.Minute, 
Fixtures.id, Goal.MatchID,PlayerID,teams.id FROM Goal,Fixtures,Player,MatchDetails,Teams
                     WHERE Goal.ID=Goal.id and Player.ID=PlayerID
                     and Goal.MatchID=Fixtures.ID and Player.TeamID=Teams.id and 
fixtures.id=%s
                     ORDER BY Goal.Minute ASC;"""
         cursor.execute(statement,[Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method shows goal in user live match page.

```
def Player_fixture_team(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select 
Player.ID,PlayerName,Teamname,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOfBirth,
 Fixtures.HomeTeam, Fixtures.ID,Player.TeamID  From Player, Fixtures, Teams Where 
((Teams.ID=Fixtures.HomeTeam and Player.TeamID=Fixtures.HomeTeam)
         or (Teams.ID=Fixtures.AwayTeam and Player.TeamID=Fixtures.AwayTeam))
         and Fixtures.ID=%s Order By Teamname"""
         cursor.execute(statement,[Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used for showing each teams’ players in live match page.

```
def Fixture_team_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select distinct Fixtures.ID,T1.Teamname
,T2.Teamname,Week,MatchDate,Time,HomeScore,AwayScore,Status,RefereeName,HomeTeam,AwayTeam,Refe
 FROM Fixtures,Teams AS T1,Teams AS T2,Referee
         WHERE (( T1.ID=HomeTeam AND T2.ID=AwayTeam and Fixtures.Hometeam=T1.ID and
HomeTeam=T1.ID) )
         AND Refereeid=Referee.id  and (T1.ID=%s or T2.ID=%s)  ORDER BY MatchDate,Time"""
         cursor.execute(statement,[Key,Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used for when accessing a team by key access in order to show teams' played or unplayed matches.

```
def Manager_team_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """SELECT manager.id, manager.name, manager.age, manager.nationalty,
manager.height, manager.placeofbirth, teams.teamname, Teams.ID from manager left join teams
on manager.id = teams.managerid
         where manager.id=manager.id and Teams.ManagerID=Manager.ID and Teams.ID=%s Order
By Name"""
         cursor.execute(statement,[Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

This method used for when accessing a team by key access in order to show teams' managers.

```
@app.route("/add_team", methods=['GET','POST'])
@login_required
def team_adding_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      obje = forms.FootballStats()
      managerCursor=obje.Manager()
      return render_template('add_team.html',cursor=managerCursor)
   elif request.method == 'POST':
      Teamname = str(request.form["Teamname"])
      NickName = str(request.form["NickName"])
      ShortName = str(request.form["ShortName"])
      FoundationDate = str(request.form["FoundationDate"])
      ManagerID =  str(request.form["ManagerID"])
      Location =  str(request.form["Location"])
      obje = forms.FootballStats()
      obje.Team_add(Teamname,NickName,ShortName,FoundationDate,ManagerID,Location)
      flash("You have added.")
      return redirect(url_for("team_adding_page"))
```

In this method, if our method is post, we get the form information and we add new team.

```
@app.route("/team", methods=['GET','POST'])
@login_required
def team_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Team()
      print(cursor)
      return render_template("teams.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("team_adding_page"))
      elif(process == "Delete"):
            form_team_keys = request.form.getlist("team_keys")
            for team_key in form_team_keys:
               obje.Team_delete(team_key)
            flash("You have deleted.")
            return redirect(url_for("team_page"))
      else:
            return team_update_page(process)
```

In this method, if we enter the team page with get method, we list information of teams. If it is post, we investigate the button value. If the value is add we go to team adding page, if it is delete, we call team delete method. In the other possibility we go to team update page by calling team information method that gets the information of team that will be updated.

```
@app.route("/add_player", methods=['GET','POST'])
@login_required
def player_adding_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      obje = forms.FootballStats()
      teamCursor=obje.Team()
      return render_template('add_player.html',cursor=teamCursor)

   elif request.method == 'POST':
      PlayerName = str(request.form["PlayerName"])
      PlayerAge = str(request.form["PlayerAge"])
      Position = str(request.form["Position"])
      PlayerNationalty = str(request.form["PlayerNationalty"])
      PlayerHeight = str(request.form["PlayerHeight"])
      PlaceOfBirth = str(request.form["PlaceOfBirth"])
      TeamID = str(request.form["TeamID"])
      obje = forms.FootballStats()
      obje.Player_add(PlayerName, PlayerAge, Position, PlayerNationalty, PlayerHeight,
PlaceOfBirth, TeamID)
      flash("You have added.")
      return redirect(url_for("player_adding_page"))
```

If our method is post, we get the form information and we add new player.

```
@app.route("/player", methods=['GET','POST'])
@login_required
def player_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Player()
      return render_template("players.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("player_adding_page"))
      elif(process == "Delete"):
            form_player_keys = request.form.getlist("player_keys")
            for form_player_key in form_player_keys:
               obje.Player_delete(int(form_player_key))
            return redirect(url_for("player_page"))
      else:
            return player_update_page(process)
```

In this method, if we enter the player page with get method, we list information of players. If it is post, we investigate the button value. If the value is add we go to player adding page, if it is delete, we call player delete method. In the other possibility we go to player update page by calling player information method that gets the information of player that will be updated.

```
@app.route("/add_manager", methods=['GET','POST'])
@login_required
def manager_adding_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      return render_template('add_manager.html')

   elif request.method == 'POST':
      Name = str(request.form["Name"])
      Age = str(request.form["Age"])
      Nationalty = str(request.form["Nationalty"])
      Height = str(request.form["Height"])
      PlaceOfBirth = str(request.form["PlaceOfBirth"])
      obje = forms.FootballStats()
      obje.Manager_add(Name, Age, Nationalty, Height, PlaceOfBirth)
      flash("You have added.")
      return redirect(url_for("manager_adding_page"))
```

If our method is post, we get the form information and we add new manager.

```
@app.route("/manager", methods=['GET','POST'])
@login_required
def manager_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Manager_user()
      return render_template("managers.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("manager_adding_page"))
      elif(process == "Delete"):
            form_manager_keys = request.form.getlist("manager_keys")
            for form_manager_key in form_manager_keys:
               obje.Manager_delete(int(form_manager_key))
            return redirect(url_for("manager_page"))
      else:
            return manager_update_page(process)
```

In this method, if we enter the manager page with get method, we list information of managers. If it is post, we investigate the button value. If the value is add we go to manager adding page, if it is delete, we call manager delete method. In the other possibility we go to manager update page by calling manager information method that gets the information of manager that will be updated.

```
@app.route("/add_goal", methods=['GET','POST'])
@login_required
def goal_adding_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      obje = forms.FootballStats()
      playerCursor=obje.Player()
      matchCursor=obje.Fixtures2()
      return render_template('add_goal.html',cursor=[playerCursor,matchCursor])

   elif request.method == 'POST':
      PlayerID = str(request.form["PlayerID"])
      MatchID = str(request.form["MatchID"])
      Minute = str(request.form["Minute"])
      obje = forms.FootballStats()
      obje.Goal_add(PlayerID, MatchID, Minute)
      flash("You have added.")
      return redirect(url_for("goal_adding_page"))
```

If our method is post, we get the form information and we add new goal.

```
@app.route("/goal", methods=['GET','POST'])
@login_required
def goal_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Goal()
      print(cursor)
      return render_template("goals.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("goal_adding_page"))
      elif(process == "Delete"):
            form_goal_keys = request.form.getlist("goal_keys")
            for form_goal_key in form_goal_keys:
               obje.Goal_delete(int(form_goal_key))
            return redirect(url_for("goal_page"))
      else:
            return goal_update_page(process)
```

In this method, if we enter the goal page with get method, we list information of goals. If it is post, we investigate the button value. If the value is add we go to goal adding page, if it is delete, we call goal delete method. In the other possibility we go to goal update page by calling goal information method that gets the information of goal that will be updated.

```
@app.route("/update_goal", methods=['GET','POST'])
@login_required
def goal_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("goals.html")
   elif request.method == 'POST':
      if update is not None:
            PlayerID = str(request.form["PlayerID"])
            MatchID = str(request.form["MatchID"])
            Minute = str(request.form["Minute"])
            obje = forms.FootballStats()
            obje.Goal_update(update,PlayerID,MatchID,Minute)
            return redirect(url_for("goal_page"))
      cursor=obje.Goal_update_info(process)
      playerCursor = obje.Player()
      matchCursor = obje.Fixtures2()
      print(cursor)
      return render_template("update_goal.html",cursor=[cursor,playerCursor,matchCursor])

@app.route("/update_manager", methods=['GET','POST'])
@login_required
def manager_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("managers.html")
   elif request.method == 'POST':
      if update is not None:
            Name = str(request.form["Name"])
            Age = str(request.form["Age"])
            Nationalty = str(request.form["Nationalty"])
            Height = str(request.form["Height"])
            PlaceOfBirth = str(request.form["PlaceOfBirth"])
            obje = forms.FootballStats()
            obje.Manager_update(update,Name,Age,Nationalty,Height,PlaceOfBirth)
            return redirect(url_for("manager_page"))
      cursor=obje.Manager_update_info(process)
      print(cursor)
      return render_template("update_manager.html",cursor=cursor)

@app.route("/update_player", methods=['GET','POST'])
@login_required
def player_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("players.html")
   elif request.method == 'POST':
      if update is not None:
            PlayerName = str(request.form["PlayerName"])
            PlayerAge = str(request.form["PlayerAge"])
            Position = str(request.form["Position"])
            PlayerNationalty = str(request.form["PlayerNationalty"])
            PlayerHeight = str(request.form["PlayerHeight"])
            PlaceOfBirth = str(request.form["PlaceOfBirth"])
            TeamID = str(request.form["TeamID"])
            obje = forms.FootballStats()

obje.Player_update(update,PlayerName,PlayerAge,Position,PlayerNationalty,PlayerHeight,PlaceOf
            return redirect(url_for("player_page"))
      cursor=obje.Player_update_info(process)
      teamCursor = obje.Team()
      print(cursor)
      return render_template("update_player.html",cursor=[cursor,teamCursor])

@app.route("/update_team", methods=['GET','POST'])
@login_required
def team_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("teams.html")
   elif request.method == 'POST':
      if update is not None:
            Teamname = str(request.form["Teamname"])
            NickName = str(request.form["NickName"])
            ShortName = str(request.form["ShortName"])
            FoundationDate = str(request.form["FoundationDate"])
            ManagerID = str(request.form["ManagerID"])
            Location = str(request.form["Location"])
            obje = forms.FootballStats()

obje.Team_update(update,Teamname,NickName,ShortName,FoundationDate,ManagerID,Location)
            return redirect(url_for("team_page"))
      cursor=obje.Team_update_info(process)
      managerCursor = obje.Manager()
      print(cursor)
      return render_template("update_team.html",cursor=[cursor, managerCursor])
```

If our method is post, we update the team, player, goal or manager according the id value received as a parameter.

```
i = 0

@app.route("/teams",methods=['GET','POST'])
@login_required
def teams_page(team_keys):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Team_key(team_keys)
      playerCursor=obje.Player_team_user(team_keys)
      stadiumCursor=obje.Stadium_key(team_keys)
      fixtureCursor=obje.Fixture_team_key(team_keys)
      managerCursor=obje.Manager_team_key(team_keys)
      print(cursor)
      return render_template("teams_player.html",cursor=
[cursor,playerCursor,stadiumCursor,fixtureCursor,managerCursor])
   else:
      global i
      process = request.form.get('buttonName')
      processStadium = request.form.get('buttonStadium')
      processMatches = request.form.get('buttonMatch')
      processManager = request.form.get('buttonManager')
      processTeam = request.form.get('buttonTeam')
      processStart = request.form.get('Start')
      processPlayer = request.form.get('buttonPlayer')
      if(processStadium):
            i=1
            return stadium_update_page(processStadium)
      elif (processMatches):
            i=2
            return fixture_update_page(processMatches)
      elif (processPlayer):
            i=3
            return player_update_page(processPlayer)
      elif (processManager):
            i=4
            return manager_update_page(processManager)
      elif (processTeam):
            i=5
            return team_update_page(processTeam)
      elif (processStart):
            i=6
            return fixture_update_page(processStart)
      elif (process == "add"):
            return redirect(url_for("team_adding_page"))
      elif (process == "add_player"):
            return redirect(url_for("player_adding_page"))
      elif (process == "add_stadium"):
            return redirect(url_for("stadium_add_page"))
      elif (process == "add_Match"):
            return redirect(url_for("fixture_adding_page"))
      elif (process == "add_Manager"):
            return redirect(url_for("manager_adding_page"))
      elif(process == "Delete"):
            form_team_keys = request.form.getlist("team_keys")
            for team_key in form_team_keys:
               obje.Team_delete(team_key)
            flash("You have deleted.")
            return redirect(url_for("team_page"))
      elif(process == "Delete_player"):
            form_player_keys = request.form.getlist("player_keys")
            for form_player_key in form_player_keys:
               obje.Player_delete(int(form_player_key))
            return redirect(url_for("team_page"))
      elif(process == "Delete_stadium"):
            form_stadium_keys = request.form.getlist('stadium_keys')
            for form_stadium_key in form_stadium_keys:
               obje.Stadium_delete(int(form_stadium_key))
            return redirect(url_for("team_page"))
      elif(process == "Delete_match"):
            form_fixture_keys = request.form.getlist('fixture')
            for form_fixture_key in form_fixture_keys:
               obje.Fixture_delete(int(form_fixture_key))
            return redirect(url_for("team_page"))
      elif(process == "Delete_manager"):
            form_manager_keys = request.form.getlist("manager_keys")
            for form_manager_key in form_manager_keys:
               obje.Manager_delete(int(form_manager_key))
            return redirect(url_for("team_page"))
      else:
            if(i==1):
               stadium_update_page(processStadium)
            elif(i==2):
               fixture_update_page(processMatches)
            elif(i==3):
               player_update_page(processPlayer)
            elif(i==4):
               manager_update_page(processManager)
            elif(i==5):
               team_update_page(processTeam)
            elif(i==6):
               fixture_update_page(processStart)
            cursor=obje.Team_key(team_keys)
            playerCursor=obje.Player_team_user(team_keys)
            stadiumCursor=obje.Stadium_key(team_keys)
            fixtureCursor=obje.Fixture_team_key(team_keys)
            managerCursor=obje.Manager_team_key(team_keys)
            return render_template("teams_player.html",cursor=
[cursor,playerCursor,stadiumCursor,fixtureCursor,managerCursor])
app.add_url_rule("/team/<team_keys>", view_func=teams_page,methods=['GET','POST'])
```

This method creates a page for single team with its players, stadium, manager, fixture informations by using key access when form is GET. When form is post, since all of the team, players, stadium, manager, fixture buttons which are delete, add, update are directed to the its right function.

```
@app.route("/goals",methods=['GET','POST'])
@login_required
def goals_page(goal_keys):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Goal_key(goal_keys)
      print(cursor)
      return render_template("goals.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("goal_adding_page"))
      elif(process == "Delete"):
            form_goal_keys = request.form.getlist("goal_keys")
            for form_goal_key in form_goal_keys:
               obje.Goal_delete(int(form_goal_key))
            return redirect(url_for("goal_page"))
      else:
            return goal_update_page(process)
app.add_url_rule("/goal/<goal_keys>", view_func=goals_page,methods=['GET','POST'])

@app.route("/managers", methods=['GET','POST'])
@login_required
def managers_page(manager_keys):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Manager_key(manager_keys)
      print(cursor)
      return render_template("managers.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("manager_adding_page"))
      elif(process == "Delete"):
            form_manager_keys = request.form.getlist("manager_keys")
            for form_manager_key in form_manager_keys:
               obje.Manager_delete(int(form_manager_key))
            return redirect(url_for("manager_page"))
      else:
            return manager_update_page(process)
app.add_url_rule("/manager/<manager_keys>", view_func=managers_page,methods=['GET','POST'])

@app.route("/player",methods=['GET','POST'])
@login_required
def players_page(player_key):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Player_key(player_key)
      print(cursor)
      return render_template("players.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if (process == "add"):
            return redirect(url_for("player_adding_page"))
      elif(process == "Delete"):
            form_player_keys = request.form.getlist("player_keys")
            for form_player_key in form_player_keys:
               obje.Player_delete(int(form_player_key))
            return redirect(url_for("player_page"))
      else:
            return player_update_page(process)
app.add_url_rule("/player/<player_key>", view_func=players_page,methods=['GET','POST'])
```

In this method is used for creating pages for single tuples by using key access. In POST form Delete, add, update are same as methods that do not have key access.

```
@app.route("/top_goal", methods=['GET'])
def top_goal_page():
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Top_goal()
      print(cursor)
      return render_template("user_top_goal.html",cursor=cursor)

@app.route("/teams_user", methods=['GET'])
def team_user_page():
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Team()
      print(cursor)
      return render_template("user_teams.html",cursor=cursor)

@app.route("/managers_user", methods=['GET'])
def manager_user_page():
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Manager_user()
      return render_template("user_managers.html",cursor=cursor)

@app.route("/players_user", methods=['GET'])
def player_user_page():
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Player()
      return render_template("user_players.html",cursor=cursor)
```

These methods are used for creating user players, managers, teams, and top goal pages.

```
@app.route("/managers_user")
def managers_user_page(manager_keys):
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Manager_key(manager_keys)
      print(cursor)
      return render_template("user_managers.html",cursor=cursor)
app.add_url_rule("/managers_user/<manager_keys>", view_func=managers_user_page)


@app.route("/players_user")
def players_user_page(player_key):
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Player_key(player_key)
      print(cursor)
      return render_template("user_players.html",cursor=cursor)
app.add_url_rule("/players_user/<player_key>", view_func=players_user_page)

@app.route("/teams_user")
def teams_user_page(team_keys):
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Team_user_key(team_keys)
      playerCursor=obje.Player_team_user(team_keys)
      stadiumCursor=obje.Stadium_key(team_keys)
      fixtureCursor=obje.Fixture_team_key(team_keys)
      managerCursor=obje.Manager_team_key(team_keys)
      print(cursor)
      return render_template("user_teams_player.html",cursor=
[cursor,playerCursor,stadiumCursor,fixtureCursor,managerCursor])
app.add_url_rule("/teams_user/<team_keys>", view_func=teams_user_page)
```

These methods are used for creating user players, managers, teams, and top goal pages. These function perform key access, and teams_user_page not only show teams' informations but also shows that team's manager, stadium, players, and played or unplayed matches.

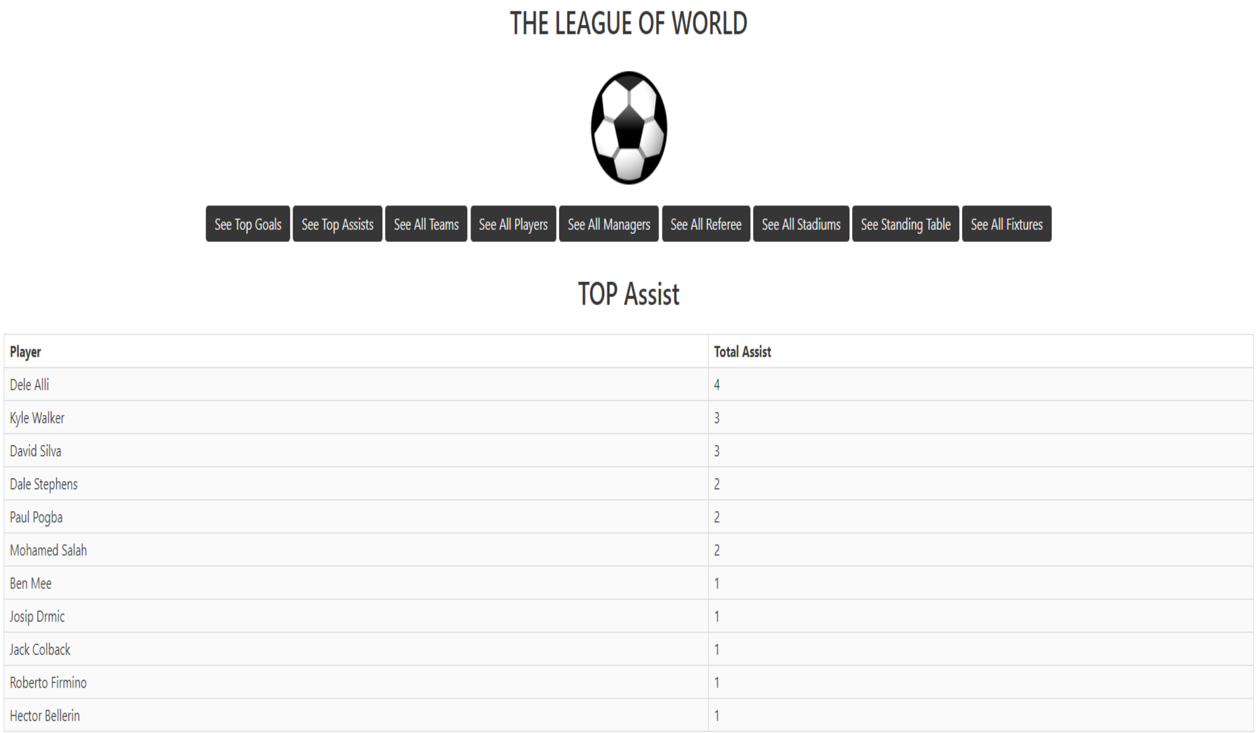
```
@app.route("/live_match", methods=['GET','POST'])
@login_required
def live_match_page(processLive):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   cursorFixture = obje.Fixture_key(processLive)
   cursorStanding = obje.Standing_key(processLive)
   cursorPlayer = obje.Player_fixture_team(processLive)
   cursorDetail = obje.Detail_user(processLive)
   cursorGoal = obje.Goal_user(processLive)
   cursorAssist = obje.Assist_user(processLive)
   return render_template("live_match.html", cursor=
[cursorFixture,cursorStanding,cursorPlayer,cursorDetail,cursorGoal,cursorAssist])
```

This method creates a live match page in order to control matches by developer. Cursor used for showing details, players, current fixture state, etc. to show current state of match to developer.

Parts Implemented by Ahmet YILMAZ

explain how yyour application works from the user perspective, use screenshots whereever appropriate

to add a picture, use the following example:

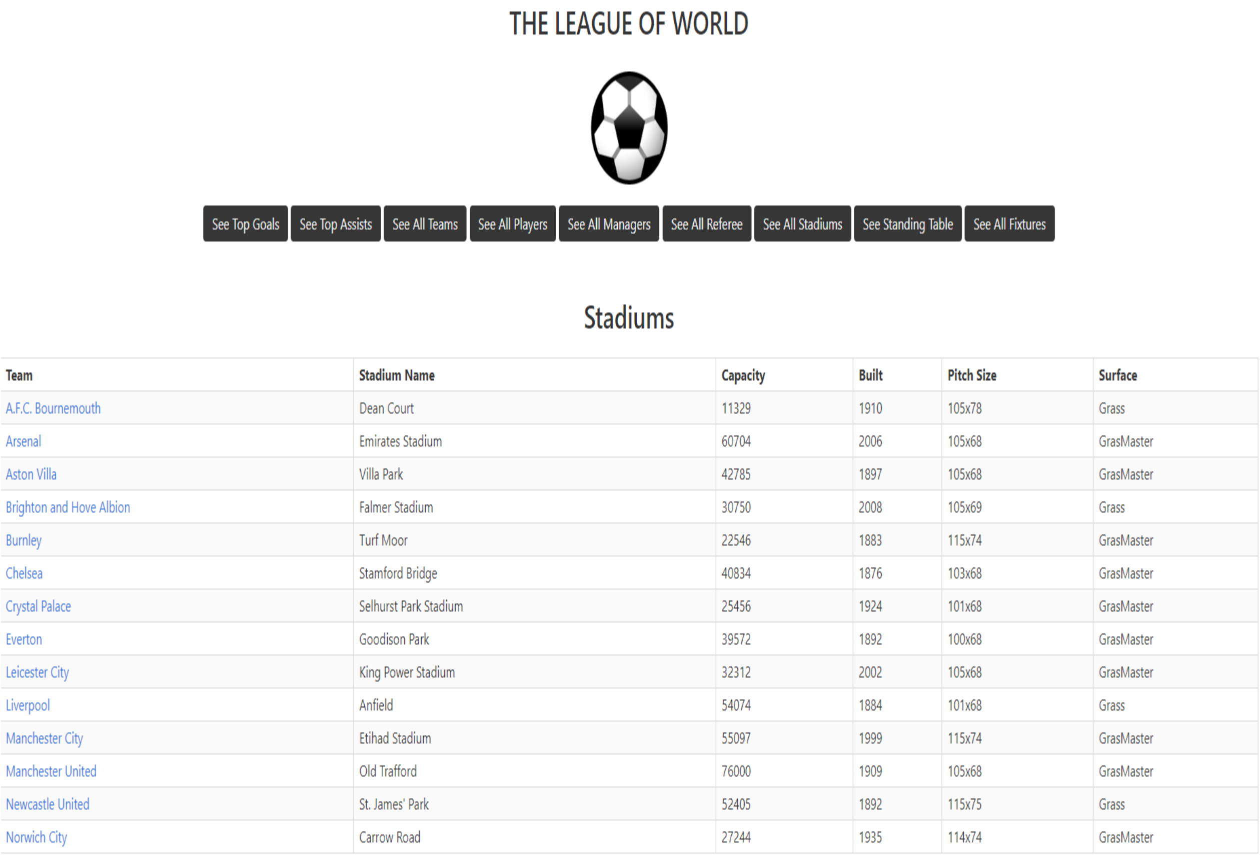


THE LEAGUE OF WORLD

TOP Assist

Player	Total Assist
Dele Alli	4
Kyle Walker	3
David Silva	3
Dale Stephens	2
Paul Pogba	2
Mohamed Salah	2
Ben Mee	1
Josip Drmic	1
Jack Colback	1
Roberto Firmino	1
Hector Bellerin	1

The user can see top assist player in this page by clicking the 'See Top Assists' button.

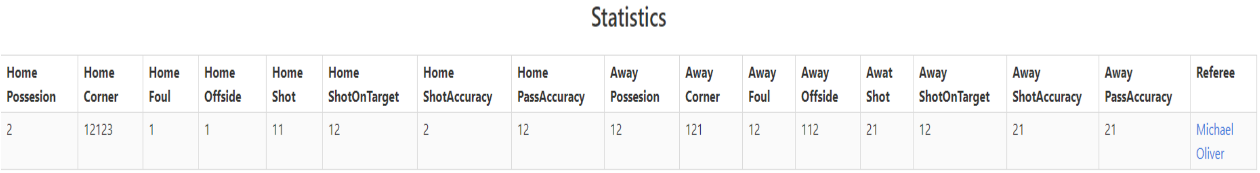


THE LEAGUE OF WORLD

Stadiums

Team	Stadium Name	Capacity	Built	Pitch Size	Surface
A.F.C. Bournemouth	Dean Court	11329	1910	105x78	Grass
Arsenal	Emirates Stadium	60704	2006	105x68	GrasMaster
Aston Villa	Villa Park	42785	1897	105x68	GrasMaster
Brighton and Hove Albion	Falmer Stadium	30750	2008	105x69	Grass
Burnley	Turf Moor	22546	1883	115x74	GrasMaster
Chelsea	Stamford Bridge	40834	1876	103x68	GrasMaster
Crystal Palace	Selhurst Park Stadium	25456	1924	101x68	GrasMaster
Everton	Goodison Park	39572	1892	100x68	GrasMaster
Leicester City	King Power Stadium	32312	2002	105x68	GrasMaster
Liverpool	Anfield	54074	1884	101x68	Grass
Manchester City	Etihad Stadium	55097	1999	115x74	GrasMaster
Manchester United	Old Trafford	76000	1909	105x68	GrasMaster
Newcastle United	St. James' Park	52405	1892	115x75	Grass
Norwich City	Carrow Road	27244	1935	114x74	GrasMaster

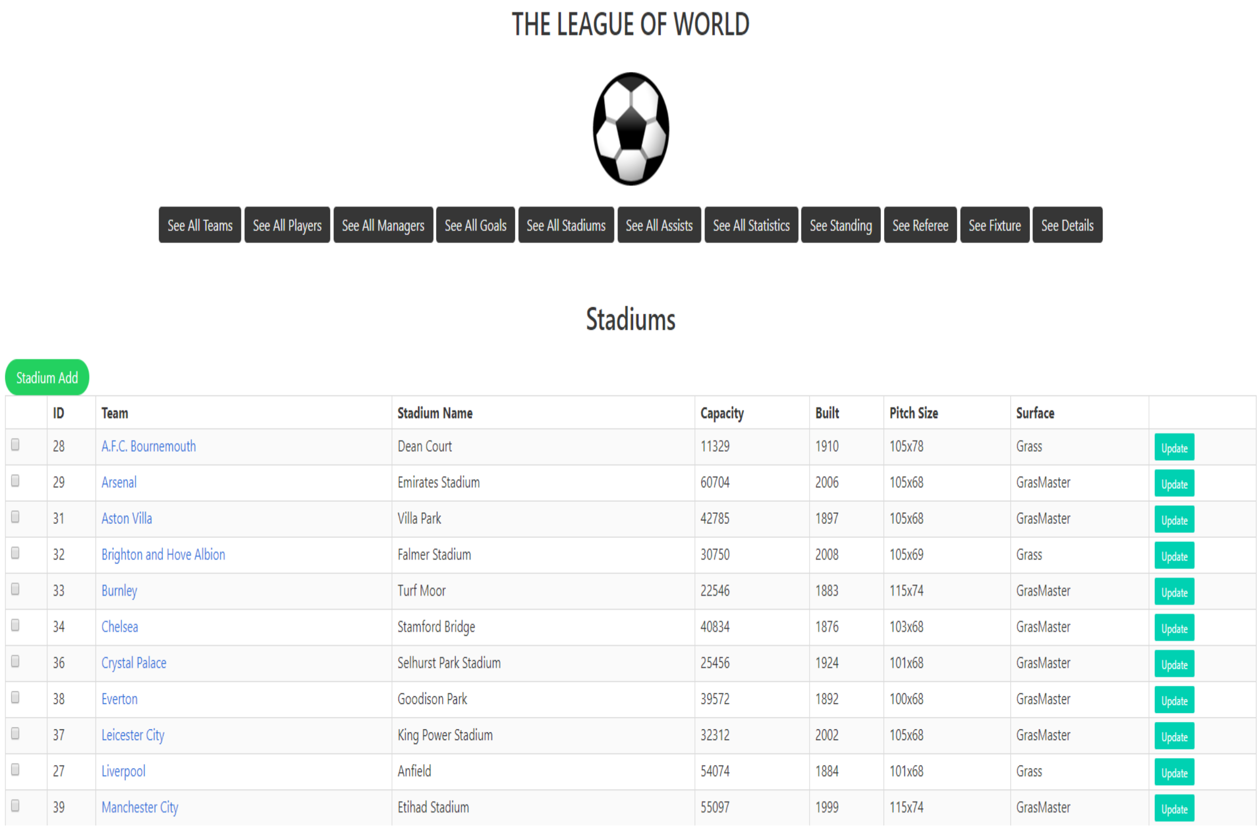
The user can see all stadium of teams by clicking the 'See All Stadiums' button.



Statistics

Home Possesion	Home Corner	Home Foul	Home Offside	Home Shot	Home ShotOnTarget	Home ShotAccuracy	Home PassAccuracy	Away Possesion	Away Corner	Away Foul	Away Offside	Awat Shot	Away ShotOnTarget	Away ShotAccuracy	Away PassAccuracy	Referee
2	12123	1	1	11	12	2	12	12	121	12	112	21	12	21	21	Michael Oliver

The user can see statistic of a match by clicking match details on the fixture page.

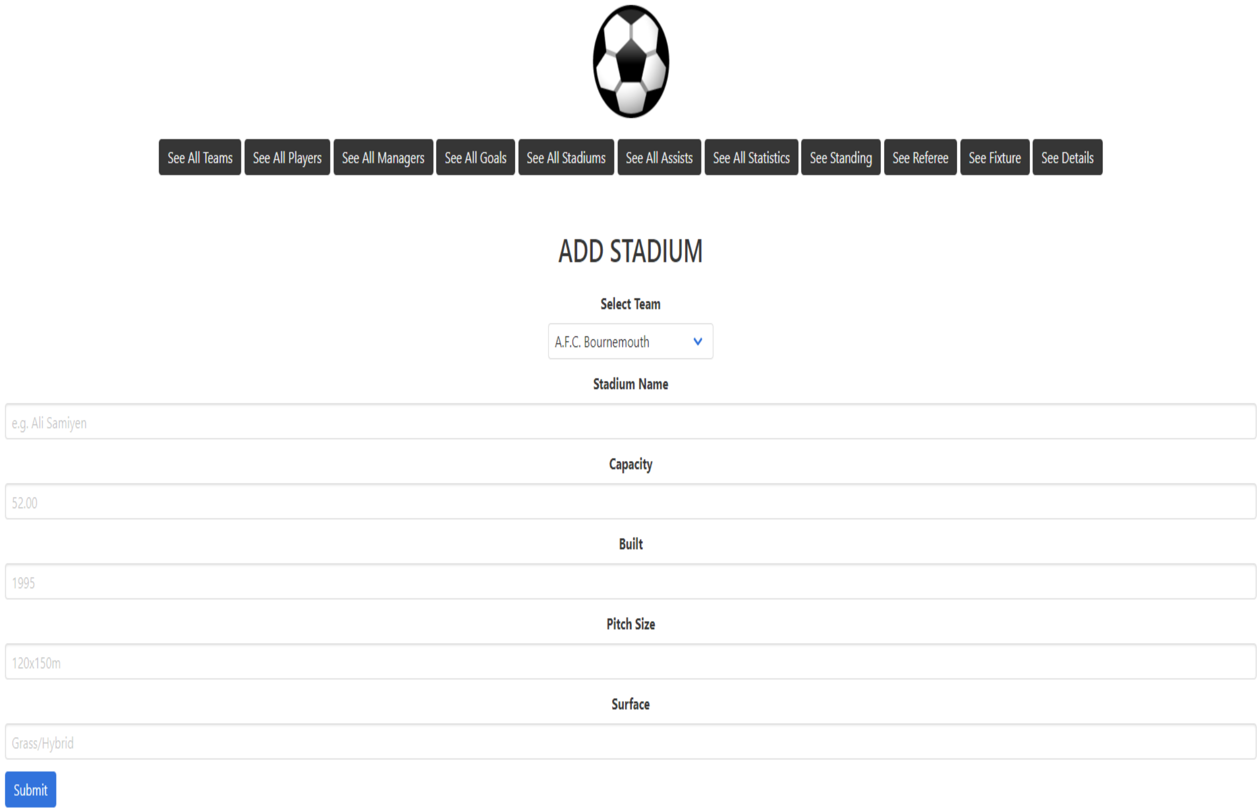


THE LEAGUE OF WORLD

Stadiums

	ID	Team	Stadium Name	Capacity	Built	Pitch Size	Surface	
☐	28	A.F.C. Bournemouth	Dean Court	11329	1910	105x78	Grass	Update
☐	29	Arsenal	Emirates Stadium	60704	2006	105x68	GrasMaster	Update
☐	31	Aston Villa	Villa Park	42785	1897	105x68	GrasMaster	Update
☐	32	Brighton and Hove Albion	Falmer Stadium	30750	2008	105x69	Grass	Update
☐	33	Burnley	Turf Moor	22546	1883	115x74	GrasMaster	Update
☐	34	Chelsea	Stamford Bridge	40834	1876	103x68	GrasMaster	Update
☐	36	Crystal Palace	Selhurst Park Stadium	25456	1924	101x68	GrasMaster	Update
☐	38	Everton	Goodison Park	39572	1892	100x68	GrasMaster	Update
☐	37	Leicester City	King Power Stadium	32312	2002	105x68	GrasMaster	Update
☐	27	Liverpool	Anfield	54074	1884	101x68	Grass	Update
☐	39	Manchester City	Etihad Stadium	55097	1999	115x74	GrasMaster	Update

After login the website, the developer guy can see the stadiums for all teams by clicking 'See All Stadiums' button. The developer also can add, update and delete the by clicking the buttons.



ADD STADIUM

Select Team

A.F.C. Bournemouth

Stadium Name

e.g. Ali Samiyen

Capacity

52.00

Built

1995

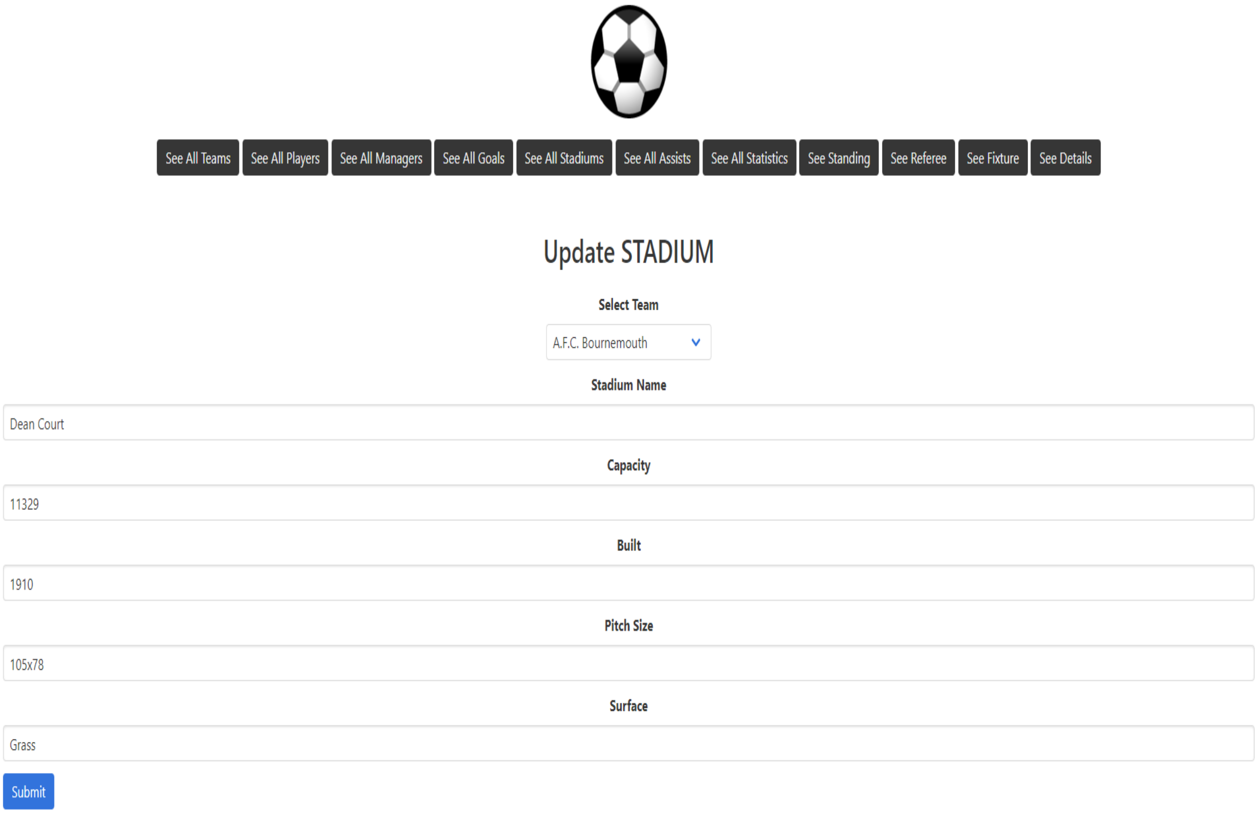
Pitch Size

120x150m

Surface

Grass/Hybrid

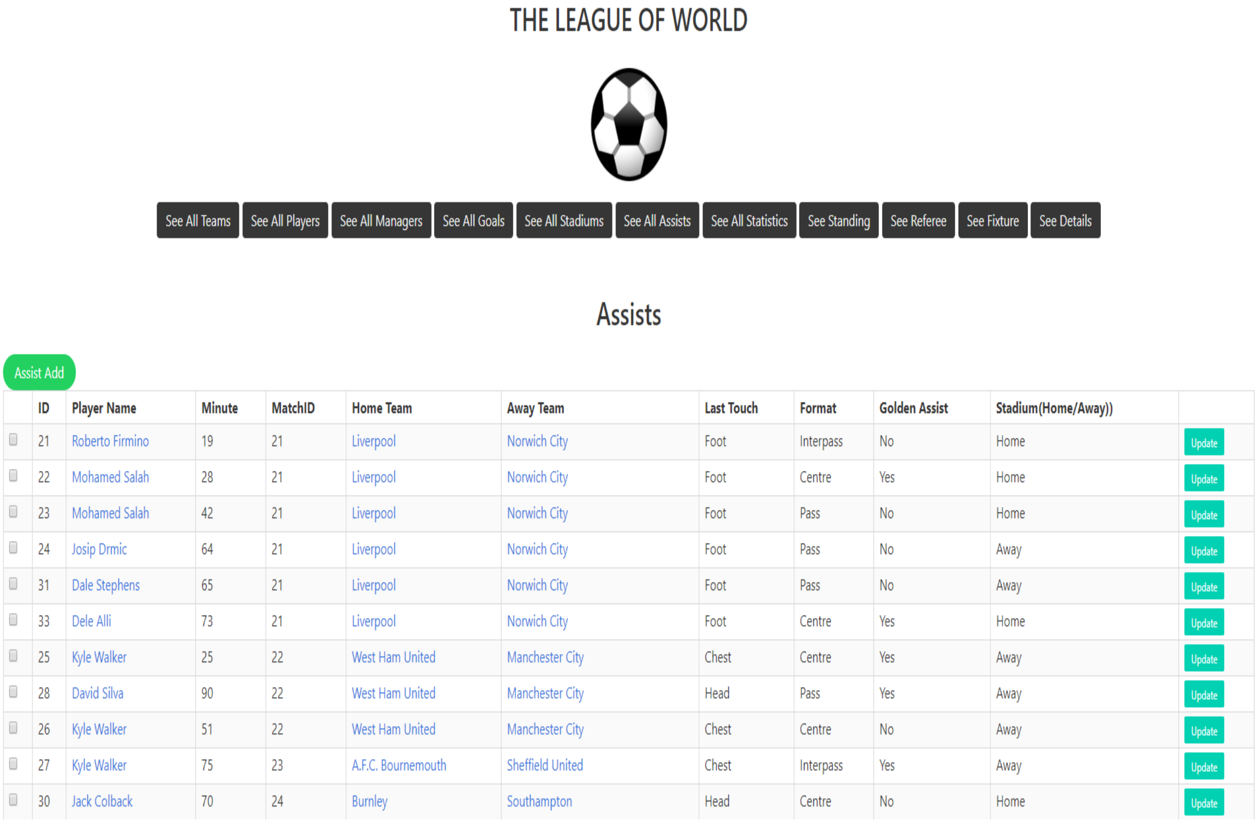
Stadium addding page for developer.



Update STADIUM

Select Team	A.F.C. Bournemouth
Stadium Name	Dean Court
Capacity	11329
Built	1910
Pitch Size	105x78
Surface	Grass

To update a stadium data, developer can use the this page after click update button.

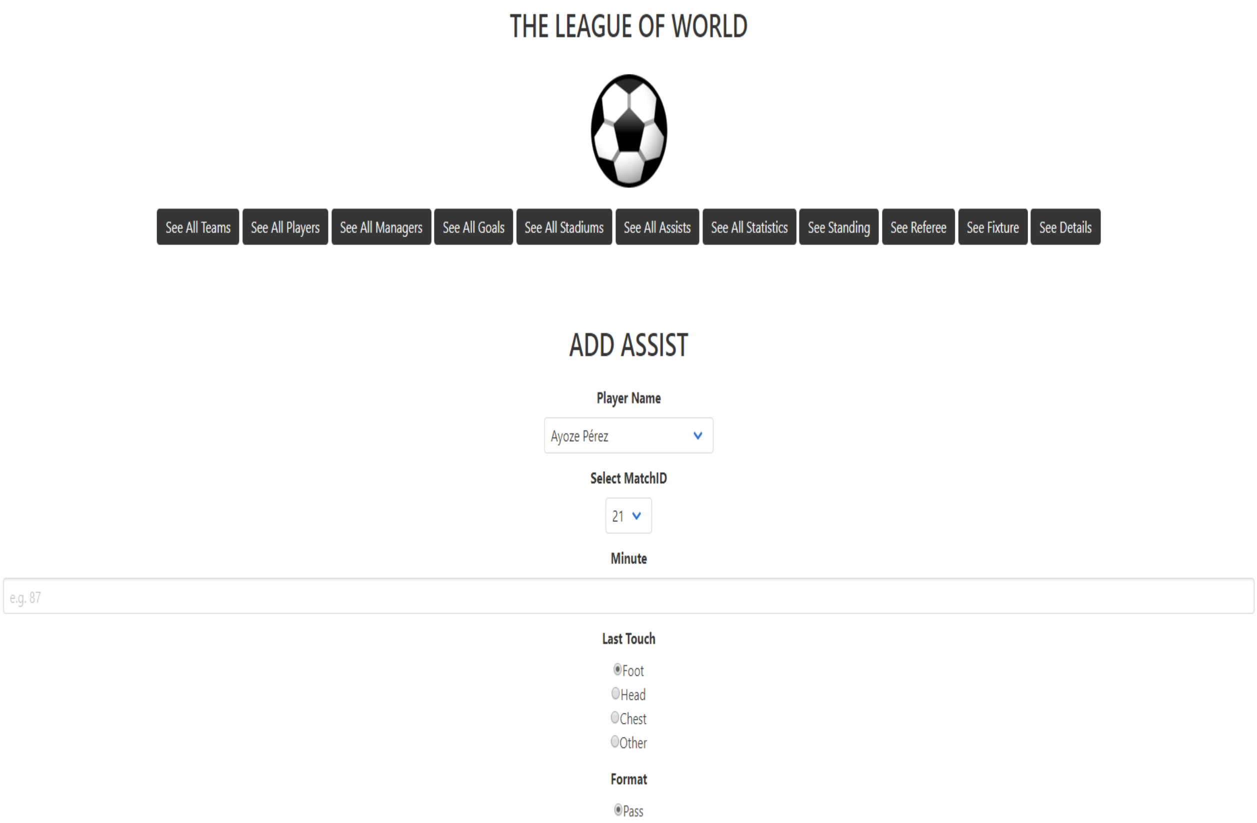


THE LEAGUE OF WORLD

Assists

	ID	Player Name	Minute	MatchID	Home Team	Away Team	Last Touch	Format	Golden Assist	Stadium(Home/Away))	
☐	21	Roberto Firmino	19	21	Liverpool	Norwich City	Foot	Interpass	No	Home	Update
☐	22	Mohamed Salah	28	21	Liverpool	Norwich City	Foot	Centre	Yes	Home	Update
☐	23	Mohamed Salah	42	21	Liverpool	Norwich City	Foot	Pass	No	Home	Update
☐	24	Josip Drmic	64	21	Liverpool	Norwich City	Foot	Pass	No	Away	Update
☐	31	Dale Stephens	65	21	Liverpool	Norwich City	Foot	Pass	No	Away	Update
☐	33	Dele Alli	73	21	Liverpool	Norwich City	Foot	Centre	Yes	Home	Update
☐	25	Kyle Walker	25	22	West Ham United	Manchester City	Chest	Centre	Yes	Away	Update
☐	28	David Silva	90	22	West Ham United	Manchester City	Head	Pass	Yes	Away	Update
☐	26	Kyle Walker	51	22	West Ham United	Manchester City	Chest	Centre	No	Away	Update
☐	27	Kyle Walker	75	23	A.F.C. Bournemouth	Sheffield United	Chest	Interpass	Yes	Away	Update
☐	30	Jack Colback	70	24	Burnley	Southampton	Head	Centre	No	Home	Update

After login the website, the developer guy can see the assists for all match by clicking 'See All Assists' button. The developer also can add, update and delete the assists by clicking the buttons.



THE LEAGUE OF WORLD

ADD ASSIST

Player Name: Ayoze Pérez

Select MatchID: 21

Minute: e.g. 87

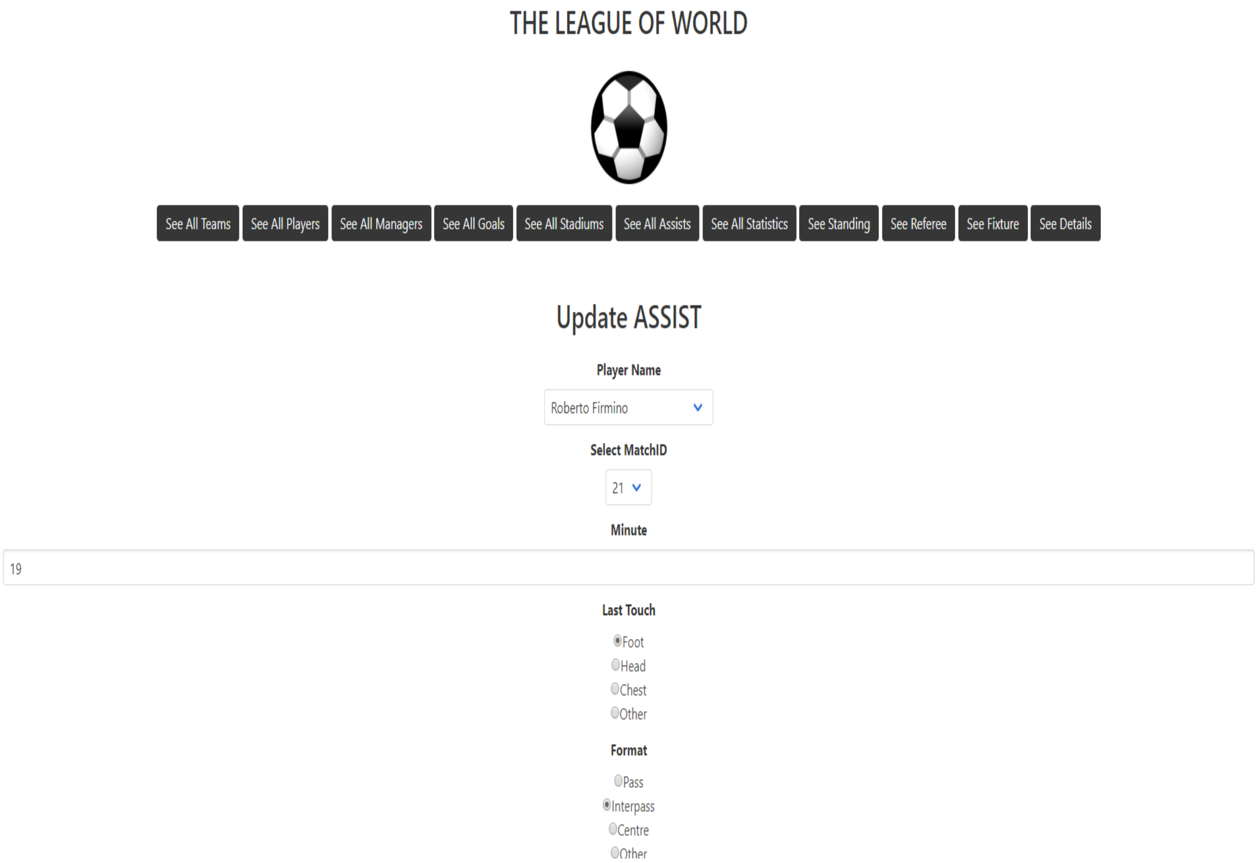
Last Touch

● Foot
○ Head
○ Chest
○ Other

Format

● Pass

Assist adddding page for developer.



THE LEAGUE OF WORLD

Update ASSIST

Player Name: Roberto Firmino

Select MatchID: 21

Minute: 19

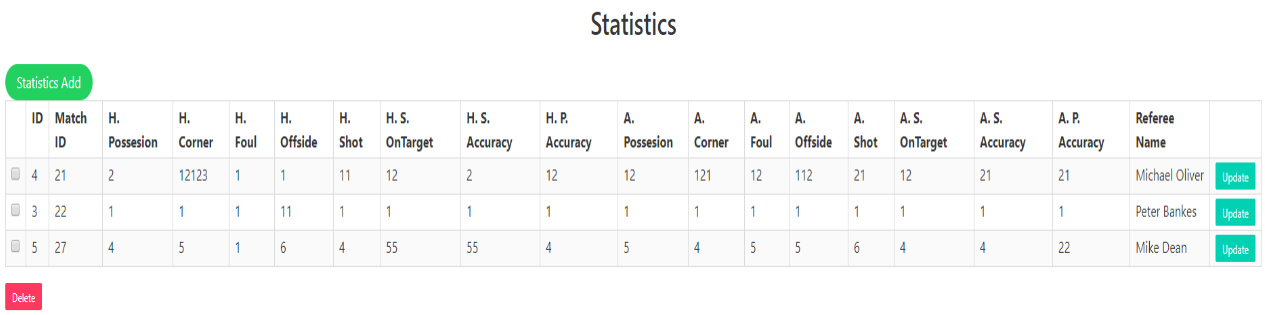
Last Touch

● Foot
○ Head
○ Chest
○ Other

Format

○ Pass
● Interpass
○ Centre
○ Other

To update an assist data, developer can use the this page after click update button.



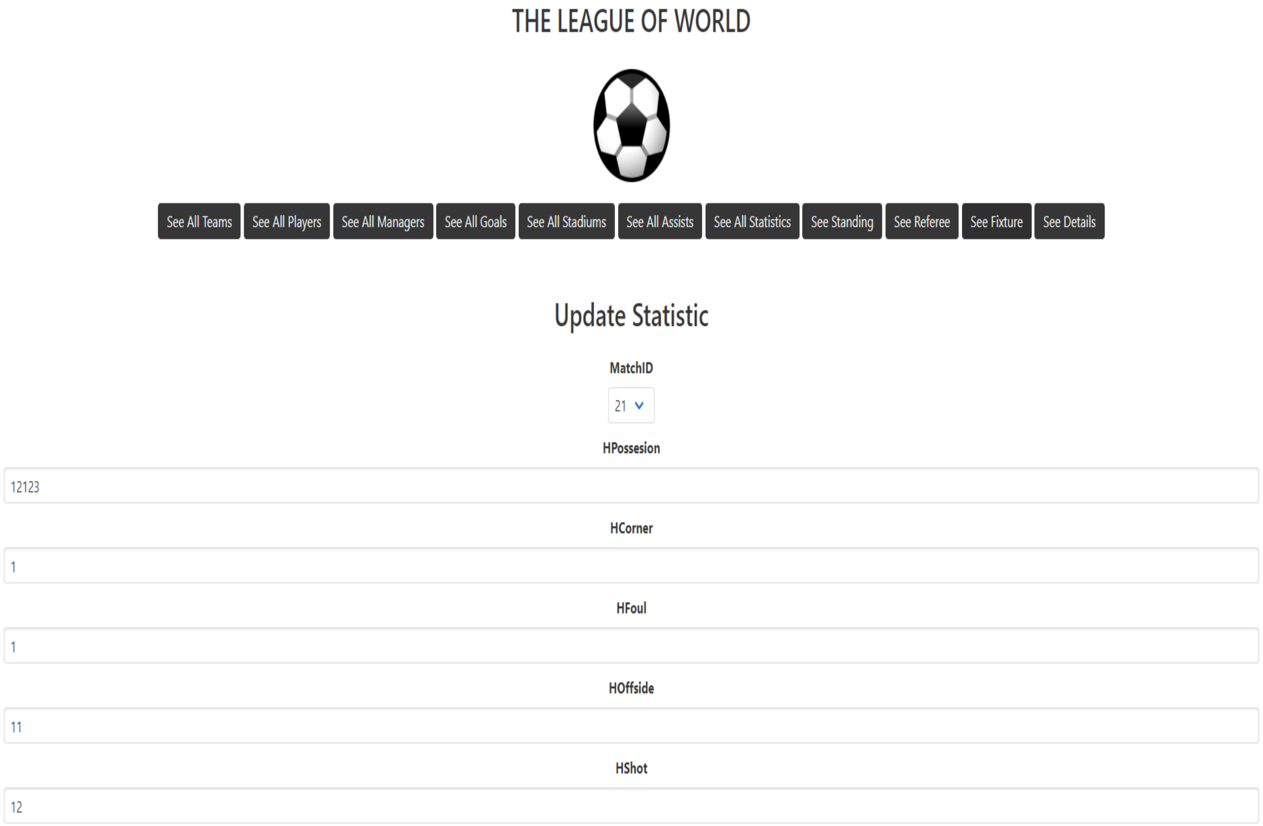
Statistics

	ID	Match ID	H. Possesion	H. Corner	H. Foul	H. Offside	H. Shot	H. S. OnTarget	H. S. Accuracy	H. P. Accuracy	A. Possesion	A. Corner	A. Foul	A. Offside	A. Shot	A. S. OnTarget	A. S. Accuracy	A. P. Accuracy	Referee Name	
□	4	21	2	12123	1	1	11	12	2	12	12	121	12	112	21	12	21	21	Michael Oliver	Update
□	3	22	1	1	1	11	1	1	1	1	1	1	1	1	1	1	1	1	Peter Bankes	Update
□	5	27	4	5	1	6	4	55	55	4	5	4	5	5	6	4	4	22	Mike Dean	Update

After login the website, the developer guy can see the statistic for all match by clicking ‘See All Statistics’ button. The developer also can add, update and delete the statistic by clicking the buttons.

. figure:: DevAddStat.png :scale: 50 % :alt: Statistic Adding Page

Statistic adddding page for developer.



Update Statistic

MatchID: 21

HPossesion: 12123

HCorner: 1

HFoul: 1

HOffside: 11

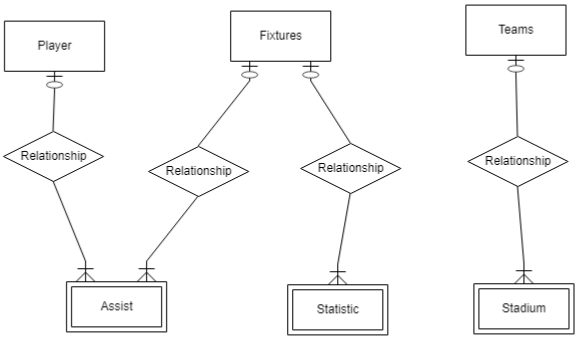
HShot: 12

To update a statistic data, developer can use the this page.

Parts Implemented by Ahmet YILMAZ

Database Design

explain the database design of your project



Code

explain the technical structure of your code

to include a code listing, use the following example:

```
def Assist(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Assist.id,Player.playername,Assist.minute,home.teamname as
HomeTeam, away.teamname as
AwayTeam,Assist.lasttouch,Assist.format,Assist.goldenassist,Assist.stadiumha, Player.id,
HomeTeam,Awayteam,MatchID FROM Assist, Player,Teams as home, Teams as away, Fixtures where
Assist.playerid = Player.id and Assist.matchid = fixtures.id and home.id=fixtures.hometeam
and away.id=fixtures.awayteam ORDER BY fixtures.ID"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

This code get all assist from db by joining Teams and Fixtures tables. To show the assist is done in which match we have to join with fixtures table and teams table for team’s name

```
def Assist_add(self, PlayerId, MatchId, Minute, LastTouch, Format, GoldenAssist,
StadiumHA):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ INSERT INTO
Assist(PlayerId,MatchId,Minute,LastTouch,Format,GoldenAssist,StadiumHA)
VALUES(%s,%s,%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([PlayerId,
MatchId,Minute,LastTouch,Format,GoldenAssist,StadiumHA]))
```

This method adds new assist

```
def Assist_delete(self, AssistId):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Delete From Assist Where ID = %s; """
         cursor.execute(statement,([AssistId]))
```

This method deletes the asistst according to id value.

```
def Assist_update(self, AssistId, PlayerId, MatchId, Minute, LastTouch, Format,
GoldenAssist, StadiumHA):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Assist Set PlayerID=%s, MatchID=%s, Minute=%s, LastTouch=%s,
Format=%s,GoldenAssist=%s,StadiumHA=%s Where ID=%s;"""
         cursor.execute(statement,([PlayerId, MatchId, Minute, LastTouch, Format,
GoldenAssist, StadiumHA, AssistId]))
```

This query updates the assist that exists before

```
def Assist_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Assist where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list
```

To get information of asisst that will be updated and show in the .html page, we use this method.

```
def Assist_user(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Assist.id,Player.playername,Teams.Teamname,
Assist.minute,Assist.lasttouch,Assist.format,Assist.goldenassist,Assist.stadiumha,
Player.id, Player.teamid FROM Assist, Player,Teams,Fixtures where Assist.playerid =
Player.id and Assist.matchid = fixtures.id and fixtures.id=%s and Teams.id=Player.Teamid
ORDER BY minute"""
         cursor.execute(statement,[Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

To show assists information on the user side, we use this query.

```
def Stadium(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Stadium.id, Teamname, StadiumName,
capacity,built,pitchsize,surface,team_id FROM Stadium,teams Where Teams.id=team_id ORDER BY
Teamname"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

This code get all stadium information from db by joining Teams table to show team's name.

```
def Stadium_add(self, TeamId, StadiumName, Capacity, Built, PitchSize, Surface):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ INSERT INTO
Stadium(Team_ID,Stadiumname,Capacity,Built,PitchSize,Surface) VALUES(%s,%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([TeamId, StadiumName, Capacity, Built, PitchSize,
Surface]))
```

This method adds new stadium for teams.

```
def Stadium_delete(self,StadiumId):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Delete From Stadium Where ID = %s;"""
         cursor.execute(statement,([StadiumId]))
```

This method deletes the stadium according to id value.

```
def Stadium_update(self, StadiumId, TeamId, StadiumName, Capacity, Built, PitchSize,
Surface):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Stadium Set Team_ID=%s, Stadiumname=%s, Capacity=%s, Built=%s,
PitchSize=%s, Surface=%s Where ID=%s;"""
         cursor.execute(statement,([TeamId, StadiumName, Capacity, Built, PitchSize,
Surface, StadiumId]))
```

This query updates the stadium that exists before

```
def Stadium_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ Select * From Stadium where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list
```

To get information of stadium that will be updated and show in the .html page, we use this method.

```
def Stadium_key(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select Stadium.id, Teamname, StadiumName,
capacity,built,pitchsize,surface,team_id FROM Stadium,teams Where Teams.id=team_id and
team_id=%s ORDER BY Teamname"""
         cursor.execute(statement, [Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

To show stadium information on the user side, we use this query.

```
def Statistic(self):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select statistic.ID, matchid, HPossesion, HCorner, HFoul, HOffside,
HShot, HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside,
AShot, AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeName FROM Statistic,Referee
where "RefereeID"=Referee.ID ORDER BY MatchID"""
         cursor.execute(statement)
         cursor_list=cursor.fetchall()
         return cursor_list
```

This code get all statistic from db by joining Referee table to show referee’s name.

```
def Statistic_add(self, MatchID, HPossesion, HCorner, HFoul, HOffside, HShot,
HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside, AShot,
AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """ INSERT INTO Statistic(MatchID, HPossesion, HCorner, HFoul,
HOffside, HShot, HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul,
AOffside, AShot, AShotOnTarget, AShotAccuracy, APassAccuracy, "RefereeID")
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s);"""
         cursor.execute(statement,([MatchID, HPossesion, HCorner, HFoul, HOffside, HShot,
HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside, AShot,
AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeID]))
```

This method adds new statistic for the match

```
def Statistic_delete(self, StatisticId):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Delete From Statistic Where ID = %s; """
         cursor.execute(statement,([StatisticId]))
```

This method deletes the statistic according to id value.

```
def Statistic_Update(self, StatisticId, MatchID, HPossesion, HCorner, HFoul, HOffside,
HShot, HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside,
AShot, AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement="""Update Statistic Set MatchID=%s, HPossesion=%s, HCorner=%s, HFoul=%s,
HOffside=%s, HShot=%s, HShotOnTarget=%s, HShotAccuracy=%s, HPassAccuracy=%s, APossesion=%s,
ACorner=%s, AFoul=%s, AOffside=%s, AShot=%s, AShotOnTarget=%s, AShotAccuracy=%s,
APassAccuracy=%s, "RefereeID"=%s Where ID=%s;"""
         cursor.execute(statement,([MatchID, HPossesion, HCorner, HFoul, HOffside, HShot,
HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside, AShot,
AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeID,StatisticId]))
```

This query updates the statistic that exists before

```
def Statistic_update_info(self, ID):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select * FROM Statistic where ID = %s;"""
         cursor.execute(statement,([ID]))
         cursor_list=cursor.fetchall()
         return cursor_list
```

To get information of statistic that will be updated and show in the .html page, we use this method.

```
def Statistic_user(self,Key):
   with dbapi.connect(url) as connection:
      with connection.cursor() as cursor:
         statement = """Select statistic.ID, matchid, HPossesion, HCorner, HFoul, HOffside, HShot, HShotOnTarget, HShotAccuracy, HPassAccuracy, APossesion, ACorner, AFoul, AOffside, AShot, AShotOnTarget, AShotAccuracy, APassAccuracy, RefereeName, Statistic."RefereeID" FROM Statistic, Referee  Where matchid=%s and Statistic."RefereeID"=Referee.id ORDER BY MatchID"""
         cursor.execute(statement,[Key])
         cursor_list=cursor.fetchall()
         return cursor_list
```

To show statistic information on the user side, we use this query.

```
@app.route("/stadium", methods=['GET','POST'])
@login_required
def stadium_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Stadium()
      print(cursor)
      return render_template("stadium.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if(process == "Delete"):
            form_stadium_keys = request.form.getlist('stadium_keys')
            for form_stadium_key in form_stadium_keys:
               obje.Stadium_delete(int(form_stadium_key))
            return redirect(url_for("stadium_page"))
      elif (process == "add"):
            return redirect(url_for("stadium_add_page"))
      else:
            return stadium_update_page(process)
```

In this method, if we enter the stadium page with get method, we list information of stadiums. If it is post, we investigate the button value. If the value is add we go to stadium add page, if it is delete, we call stadium delete method. In the other possibility we go to stadium update page by calling stadium information method that gets the information of stadium that will be updated.

```
@app.route("/add_stadium", methods=['GET','POST'])
@login_required
def stadium_add_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      obje = forms.FootballStats()
      teamCursor=obje.Team()
      return render_template('add_stadium.html',cursor=teamCursor)
   elif request.method == 'POST':
      Team_ID = str(request.form["Team_ID"])
      Stadiumname = str(request.form["Stadiumname"])
      Capacity = str(request.form["Capacity"])
      Built = str(request.form["Built"])
      PitchSize = str(request.form["PitchSize"])
      Surface = str(request.form["Surface"])
      obje = forms.FootballStats()
      obje.Stadium_add(Team_ID,Stadiumname,int(Capacity),Built,PitchSize,Surface)
      flash("Stadium added")
      return redirect(url_for("stadium_add_page"))
```

if our method is post, we get the form information and we add new stadium.

```
@app.route("/update_stadium", methods=['GET','POST'])
@login_required
def stadium_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("stadium.html")
   elif request.method == 'POST':
      if update is not None:
            Team_ID = str(request.form["Team_ID"])
            Stadiumname = str(request.form["Stadiumname"])
            Capacity = str(request.form["Capacity"])
            Built = str(request.form["Built"])
            PitchSize = str(request.form["PitchSize"])
            Surface = str(request.form["Surface"])
            obje = forms.FootballStats()

obje.Stadium_update(update,Team_ID,Stadiumname,int(Capacity),Built,PitchSize,Surface)
            return redirect(url_for("stadium_page"))
      cursor=obje.Stadium_update_info(process)
      teamsCursor = obje.Team()
      print(cursor)
      return render_template("update_stadium.html",cursor=[cursor,teamsCursor])
```

if our method is post, we update the stadium according the id value received as a parameter.

```
@app.route("/stadium_user", methods=['GET'])
def stadium_user_page():
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Stadium()
      print(cursor)
      return render_template("user_stadium.html",cursor=cursor)
```

To send information of stadiums to user, we use this method

```
@app.route("/assist", methods=['GET','POST'])
@login_required
def assist_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   if request.method == "GET":
      cursor=obje.Assist()
      print(cursor)
      return render_template("assist.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      print(update)
      if(process == "Delete"):
            form_assist_keys = request.form.getlist('assist_keys')
            for form_assist_key in form_assist_keys:
               obje.Assist_delete(int(form_assist_key))
            return redirect(url_for("assist_page"))
      elif (process == "add"):
            return redirect(url_for("assist_add_page"))
      else:
            return assist_update_page(process)
```

In this method, if we enter the assist page with get method, we list information of assists. If it is post, we investigate the button value. If the value is add we go to assists add page, if it is delete, we call assist delete method. In the other possibility we go to assist update page by calling assist information method that gets the information of assists that will be updated.

```
@app.route("/add_assist", methods=['GET','POST'])
@login_required
def assist_add_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == 'GET':
      obje = forms.FootballStats()
      playerCursor=obje.Player()
      fixtureCursor=obje.Fixtures2()
      return render_template('add_assist.html',cursor=[playerCursor,fixtureCursor])
   elif request.method == 'POST':
      PlayerID = str(request.form["PlayerID"])
      MatchID = str(request.form["MatchID"])
      Minute = str(request.form["Minute"])
      LastTouch = str(request.form["LastTouch"])
      Format = str(request.form["Format"])
      GoldenAssist = str(request.form["GoldenAssist"])
      StadiumHA = str(request.form["StadiumHA"])
      obje = forms.FootballStats()
      obje.Assist_add(PlayerID,MatchID,Minute,LastTouch,Format,GoldenAssist,StadiumHA)
      return redirect(url_for("assist_add_page"))
```

if our method is post, we get the form information and we add new assist.

```
@app.route("/update_assist", methods=['GET','POST'])
@login_required
def assist_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("assist.html")
   elif request.method == 'POST':
      if update is not None:
            PlayerID = str(request.form["PlayerID"])
            MatchID = str(request.form["MatchID"])
            Minute = str(request.form["Minute"])
            LastTouch = str(request.form["LastTouch"])
            Format = str(request.form["Format"])
            GoldenAssist = str(request.form["GoldenAssist"])
            StadiumHA = str(request.form["StadiumHA"])
            obje = forms.FootballStats()

obje.Assist_update(update,PlayerID,MatchID,Minute,LastTouch,Format,GoldenAssist,StadiumHA)
            return redirect(url_for("assist_page"))
      cursor=obje.Assist_update_info(process)
      playerCursor = obje.Player()
      fixtureCursor = obje.Fixtures2()
      print(cursor)
      return render_template("update_assist.html",cursor=
[cursor,playerCursor,fixtureCursor])
```

if our method is post, we update the assist according the id value received as a parameter.

To send information of top player's assists, we use this method.

```
@app.route("/statistic", methods=['GET','POST'])
@login_required
def statistic_page():
   if not current_user.is_admin:
      abort(401)
   if request.method == "GET":
      obje = forms.FootballStats()
      cursor=obje.Statistic()
      print(cursor)
      return render_template("statistic.html",cursor=cursor)
   else:
      process = request.form.get('buttonName')
      update = request.form.get('Update')
      if(process == "Delete"):
            form_statistic_keys = request.form.getlist('statistic_keys')
            for form_statistic_key in form_statistic_keys:
               obje.Statistic_delete(int(form_statistic_key))
            return redirect(url_for("statistic_page"))
      elif (process == "add"):
            return redirect(url_for("statistic_add_page"))
      else:
            return statistic_update_page(process)
```

In this method, if we enter the statistic page with get method, we list information of statistic. If it is post, we investigate the button value. If the value is add we go to stastistic add page, if it is delete, we call statistc delete method. In the other possibility we go to statistic update page by calling statistic information method that gets the information of statistic that will be updated.

```
@app.route("/add_statistic", methods=['GET','POST'])
@login_required
def statistic_add_page():
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   cursor=obje.Referee()
   cursor2=obje.Fixtures2()
   if request.method == 'GET':
      return render_template('add_statistic.html',cursor=[cursor,cursor2])
   elif request.method == 'POST':
      MatchID = str(request.form["MatchID"])
      HPossesion = str(request.form["HPossesion"])
      HCorner = str(request.form["HCorner"])
      HFoul = str(request.form["HFoul"])
      HOffside = str(request.form["HOffside"])
      HShot = str(request.form["HShot"])
      HShotOnTarget = str(request.form["HShotOnTarget"])
      HShotAccuracy = str(request.form["HShotAccuracy"])
      HPassAccuracy = str(request.form["HPassAccuracy"])
      APossesion = str(request.form["APossesion"])
      ACorner = str(request.form["ACorner"])
      AFoul = str(request.form["AFoul"])
      AOffside = str(request.form["AOffside"])
      AShot = str(request.form["AShot"])
      AShotOnTarget = str(request.form["AShotOnTarget"])
      AShotAccuracy = str(request.form["AShotAccuracy"])
      APassAccuracy = str(request.form["APassAccuracy"])
      RefereeID = str(request.form["RefereeID"])
      obje = forms.FootballStats()
      obje.Statistic_add(MatchID,
HPossesion,HCorner,HFoul,HOffside,HShot,HShotOnTarget,HShotAccuracy,HPassAccuracy,APossesion,

      return render_template("add_statistic.html",cursor=[cursor,cursor2])
```

if our method is post, we get the form information and we add new statistic.

```
@app.route("/update_statistic", methods=['GET','POST'])
@login_required
def statistic_update_page(process):
   if not current_user.is_admin:
      abort(401)
   obje = forms.FootballStats()
   cursorReferee=obje.Referee()
   cursorFixture=obje.Fixtures2()
   update = request.form.get('Update')
   if request.method == 'GET':
      return render_template("statistic.html")
   elif request.method == 'POST':
      if update is not None:
            MatchID = str(request.form["MatchID"])
            HPossesion = str(request.form["HPossesion"])
            HCorner = str(request.form["HCorner"])
            HFoul = str(request.form["HFoul"])
            HOffside = str(request.form["HOffside"])
            HShot = str(request.form["HShot"])
            HShotOnTarget = str(request.form["HShotOnTarget"])
            HShotAccuracy = str(request.form["HShotAccuracy"])
            HPassAccuracy = str(request.form["HPassAccuracy"])
            APossesion = str(request.form["APossesion"])
            ACorner = str(request.form["ACorner"])
            AFoul = str(request.form["AFoul"])
            AOffside = str(request.form["AOffside"])
            AShot = str(request.form["AShot"])
            AShotOnTarget = str(request.form["AShotOnTarget"])
            AShotAccuracy = str(request.form["AShotAccuracy"])
            APassAccuracy = str(request.form["APassAccuracy"])
            RefereeID = str(request.form["RefereeID"])
            obje = forms.FootballStats()
            obje.Statistic_Update(update,MatchID,
HPossesion,HCorner,HFoul,HOffside,HShot,HShotOnTarget,HShotAccuracy,HPassAccuracy,APossesion,

            return redirect(url_for("statistic_page"))
      cursor=obje.Statistic_update_info(process)
      print(cursor)
      return render_template("update_statistic.html",cursor=
[cursor,cursorReferee,cursorFixture])
```

if our method is post, we update the statistic according the id value received as a parameter.