



INFORMATICS  
INSTITUTE OF  
TECHNOLOGY

UNIVERSITY OF  
WESTMINSTER

Informatics Institute of Technology  
Department of Computing  
(B.Sc.) in Computer Science

Module: Object Oriented Programming  
Module Leader: Mr. Guhanathan Poravi  
OOP CW Report Submission

Name: Gibran Rumaiz

UoW ID: W1761211

Student ID: 2019176

"I confirm that I understand what plagiarism / collusion / contract cheating is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."

~ Gibran Kasif, UoW:w1761211, IIT ID: 2019176.

## Table of Contents

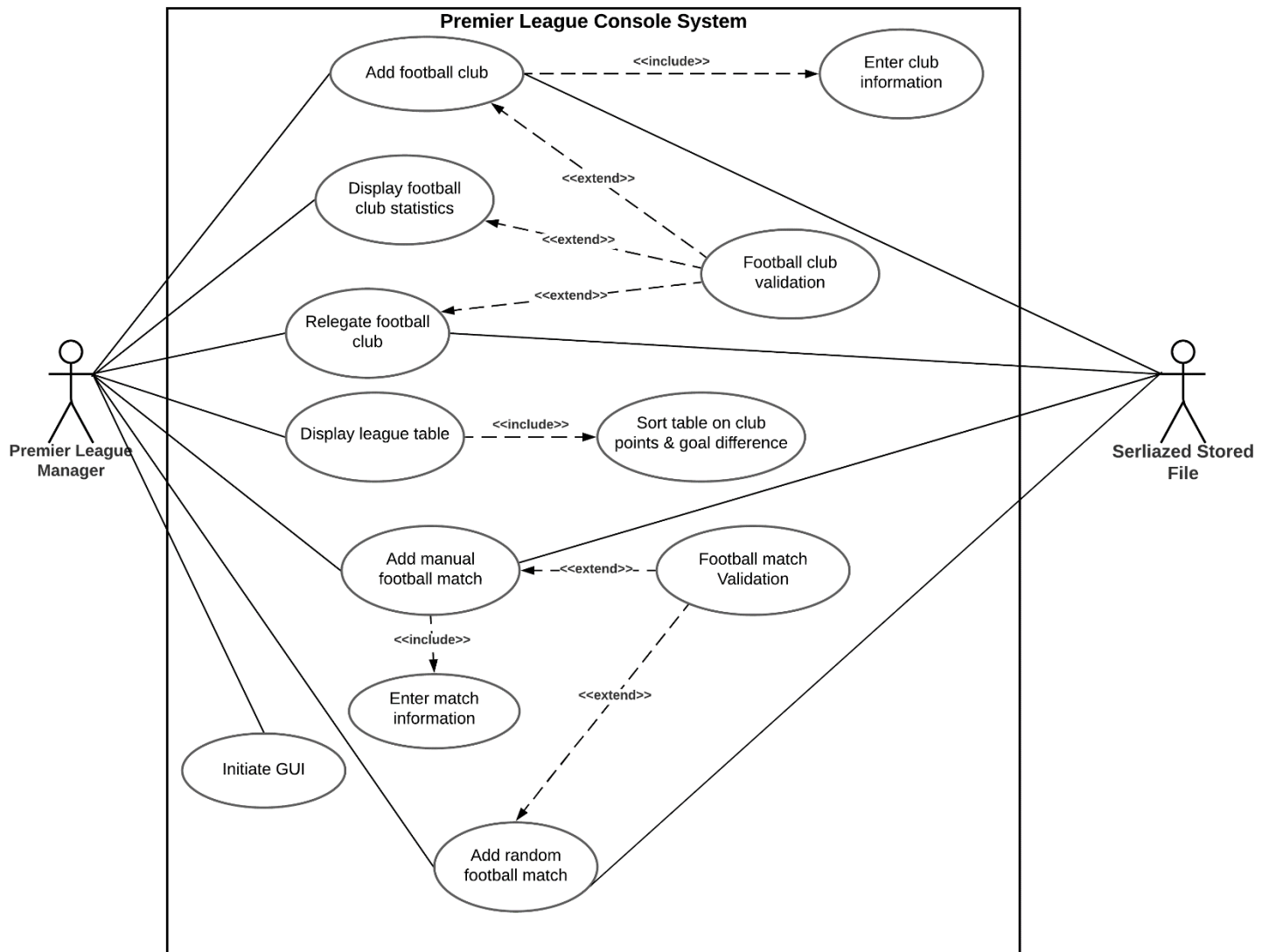
<b>1.UML Design</b>	4
1.1 Use Case Diagram	4
1.1.1Use Case Console Application Diagram	4
1.1.2 Use Case GUI Application Diagram	5
1.2 UML Class Diagram	6
<b>2. Source Code</b>	7
2.1 Console Application	7
2.2 Controllers	13
2.2.1 LeagueController	13
2.3 Entities	15
2.3.1 SportsClub.java	15
2.3.2 FootballClub.java	16
2.3.3 UniversityFootballClub.java	20
2.3.4 SchoolFootballClub.java	22
2.3.5 FootballMatch.java	23
2.4.Services	26
2.4.1 LeagueManager.java	26
2.4.2 PremierLeagueManager.java	27
<b>3. Angular GUI src</b>	35
3.1 app component	35
3.1.1 app.component.html	35
3.1.2 app.component.scss	35
3.1.3 app.service.ts	35
3.1.4 app-routing.module.ts	36

3.2 league-table component .....	37
3.2.1 league-table.component.html.....	37
3.2.2 league-table.component.ts .....	38
3.3 match component .....	39
3.3.1 match.component.html.....	39
3.3.2 match.component.ts .....	40
3.4 dialog-message component .....	42
3.4.1 dialog-message.component.html.....	42
3.4.2 dialog-message.component.ts .....	42
3.5 Angular GUI Screenshots .....	43
3.5.1 Premier League Table .....	43
3.5.2 Match Table.....	43
3.5.3 Premier League Table .....	44
4. Playframe backend Routes .....	44
5. Junit Testing class .....	44
5.1 PremierLeagueTest.java .....	44
5.2 PremierLeagueTest.java output .....	46
6. Testing .....	46
References .....	47

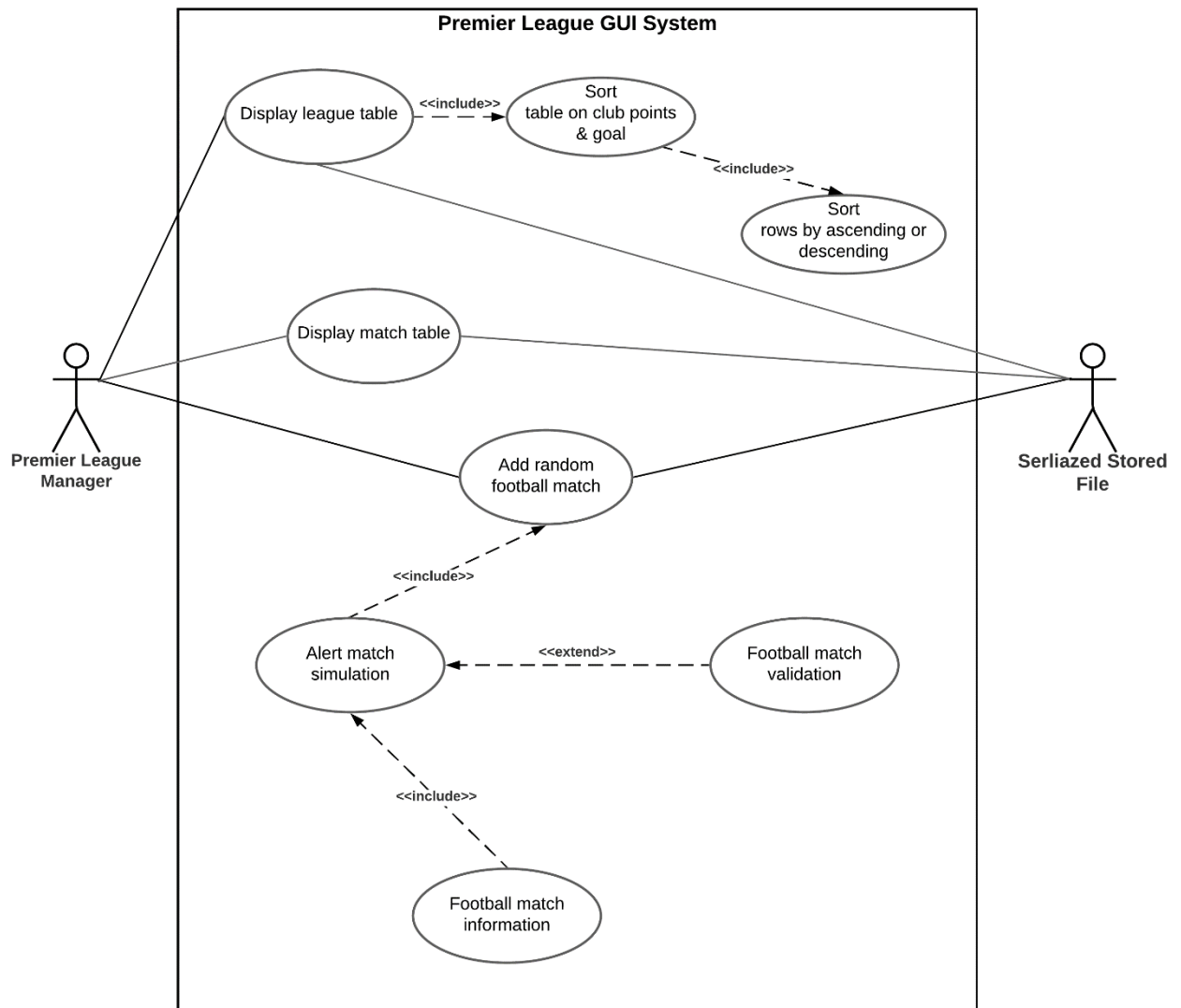
## 1.UML Design

### 1.1 Use Case Diagram

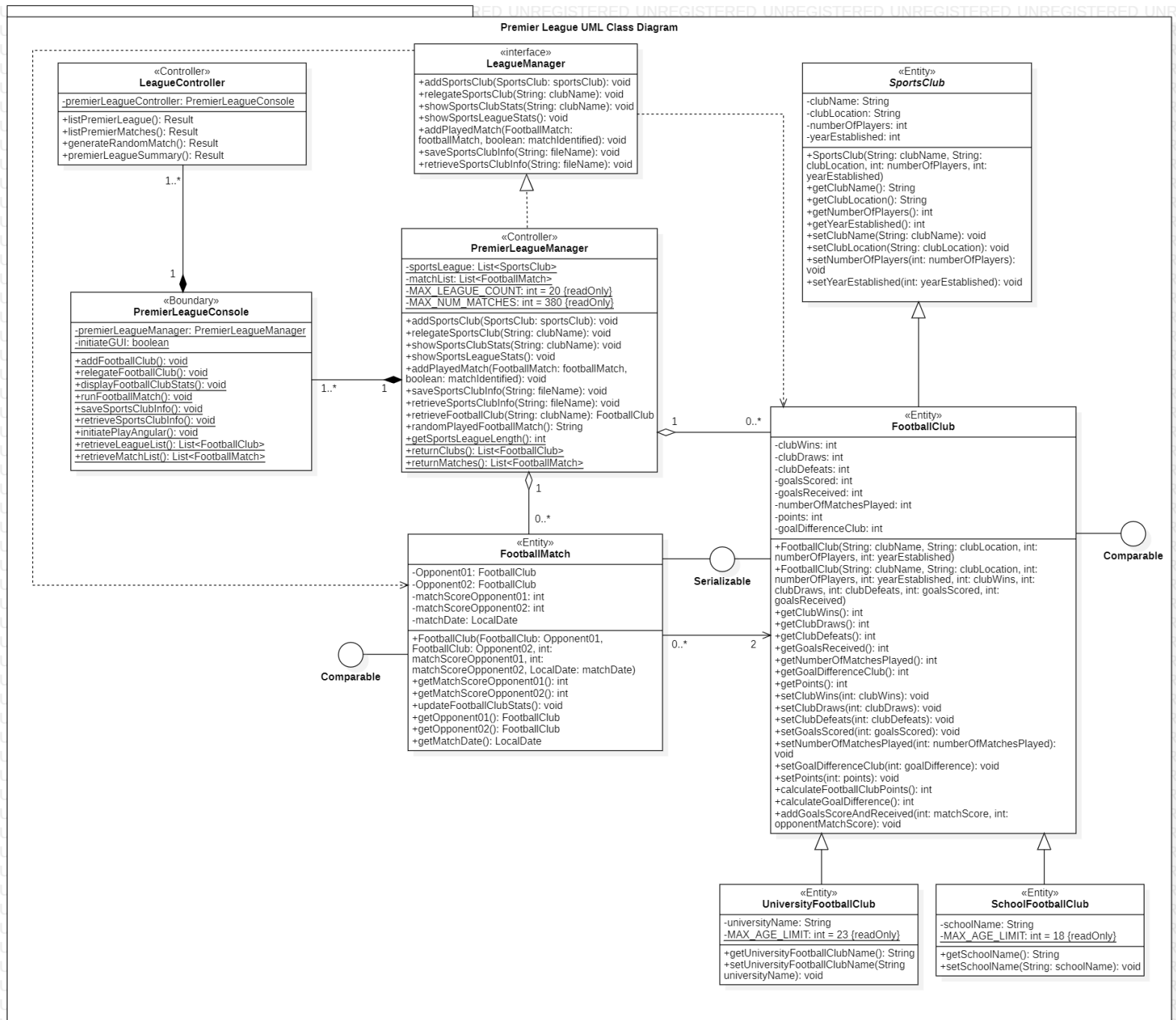
#### 1.1.1 Use Case Console Application Diagram



### 1.1.2 Use Case GUI Application Diagram



## 1.2 UML Class Diagram



## 2. Source Code

### 2.1 Console Application

```
package consoleApplication;

import entities.*;
import services.PremierLeagueManager;
import java.io.IOException;
import java.time.DateTimeException;
import java.time.LocalDate;
import java.util.*;

public class PremierLeagueConsole {
    private static PremierLeagueManager premierLeagueManager = new PremierLeagueManager();
    private static boolean initiateGUI = false;
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        //Calling retrieve method to load back previously saved info to the Premier League
        retrieveSportsClubInfo();
        System.out.println("Premier League information loaded successfully");
        /*A label statement was placed to the switch case, which is used to break out of the switch
case*/
        consoleMenuLoop:
        while(true) {
            System.out.println("-----");
            System.out.println("Press \"A\" to add a Football club");
            System.out.println("Press \"D\" to relegate a Football club");
            System.out.println("Press \"T\" to print the Premier League table chart");
            System.out.println("Press \"F\" to find club statistics and information of a specific
club");

            System.out.println("Press \"M\" to add a played match");
            System.out.println("Press \"R\" to generate a random played match");
            System.out.println("Press \"S\" to save the current progress");
            System.out.println("Press \"I\" to initiate Play-frame/Angular project");
            System.out.println("Press \"Q\" to quit and save all progress");
            System.out.println("-----");

            Scanner consoleSC = new Scanner(System.in);
            String userChoice = consoleSC.nextLine().toUpperCase();
            retrieveSportsClubInfo();

            switch (userChoice) {
                case "A":
                    addFootballClub();
                    saveSportsClubInfo();
                    break;
                case "D":
                    relegateFootballClub();
                    saveSportsClubInfo();
                    break;
```

```

        case "T":
            //Displays an entire table chart of each and every club statistics
            premierLeagueManager.showSportsLeagueStats();
            break;
        case "F":
            displayFootballClubStats();
            break;
        case "M":
            runFootballMatch();
            saveSportsClubInfo();
            break;
        case "R":
            premierLeagueManager.randomPlayedFootballMatch();
            saveSportsClubInfo();
            break;
        case "S":
            saveSportsClubInfo();
            System.out.println("Premier League Football Clubs & Matches
have been saved successfully");
            break;
        case "I":
            initiatePlayAngular();
            break;
        case "Q":
            /*Once the user enters Q to end the program, all progress
would be saved
update the data structures
on the premierLeagueFile.txt, which would be loaded and
with the file's last saved progress*/
            saveSportsClubInfo();
            System.out.println("Premier League Football Clubs & Matches
have been saved successfully");
            break consoleMenuLoop;
        default:
            System.out.println("Invalid option, please insert another
option!");
    }
}

/**The following addFootballClub method contains the premierLeague add method, inorder to call
this method, inputs were placed, for the club's name, location, player base and year of
establishment
which would be placed in an initialised FootballClub object which is the only parameter required,
for the
premier league manager's addSportsClub() method
Alongside this, input validations have also been placed*/
public static void addFootballClub() {
    Scanner footballClubInput = new Scanner(System.in);
    SportsClub sportsClub;

```



```

String clubName = "";
String clubLocation = "";
int numOfPlayers;
boolean verifyInput = true;

while (verifyInput) {
    System.out.println("Enter the name of the Football Club: ");
    clubName = footballClubInput.nextLine().toUpperCase().trim();
    if (clubName.equals("")) {
        System.out.println("Please enter a valid name");
    } else {
        verifyInput=false;
    }
}
if(premierLeagueManager.retrieveFootballClub(clubName) != null){
    System.out.println("The following club is already in the premier league!");
}
else {
    verifyInput = true;
    while(verifyInput) {
        System.out.println("Enter the location of the Football Club: ");
        clubLocation = footballClubInput.nextLine().toUpperCase().trim();
        if (clubLocation.equals("")) {
            System.out.println("Please enter a valid location");
        } else {
            verifyInput=false;
        }
    }
    System.out.println("Enter the number of players in the Football Club: ");
    try {
        verifyInput = true;
        do {
            numOfPlayers = footballClubInput.nextInt();
            if (numOfPlayers <= 0) {
                System.out.println("Enter a valid number of players: ");
            } else{
                verifyInput = false;
            }
        } while (verifyInput);

        verifyInput = true;
        int clubYear = 0;

        while (verifyInput) {
            System.out.println("Enter year of the Football Club's
establishment: ");

            clubYear = footballClubInput.nextInt();

            if (!(clubYear >= 1800 && clubYear < 2021)) {

```

```

        System.out.println("Please enter a valid year!");
    } else {
        verifyInput = false;
    }
}
sportsClub = new FootballClub(clubName, clubLocation, numOfPlayers,
clubYear);

premierLeagueManager.addSportsClub(sportsClub);
}
catch (InputMismatchException inputMismatchException){
    System.out.println("Please enter a valid number!");
}
}

}

/**The following method calls the premier league manager's relegateSportsClub() method,
along with just a string input to identify the club's name to be deleted.*/
public static void relegateFootballClub() {
    Scanner footballClubInput = new Scanner(System.in);
    System.out.println("Enter the name of the Football Club you wish to relegate: ");
    String clubName = footballClubInput.nextLine().toUpperCase().trim();
    premierLeagueManager.relegateSportsClub(clubName);
}

/**The following method calls the premier league manager's showSportsClubStats() method,
along with just a string input to identify a specific club's information & statistics*/
public static void displayFootballClubStats() {
    Scanner footballClubInput = new Scanner(System.in);
    System.out.println("Enter the name of the Football Club you wish to view its statistics: ");
    String clubName = footballClubInput.nextLine().toUpperCase().trim();
    premierLeagueManager.showSportsClubStats(clubName);
}

/**The final method runFootballMatch() gets down the premier league manager's
addPlayedMatch()
**method requires two FootballClubs objects as its arguments, so before doing that, additional
input validations
and necessary validations to confirm the existence of either one or both of the competing clubs*/
public static void runFootballMatch() {
    if(premierLeagueManager.getSportsLeagueLength() > 1) {
        Scanner footballClubInput = new Scanner(System.in);

        System.out.println("Enter the name of the first opponent club: ");
        String opp01 = footballClubInput.nextLine().toUpperCase().trim();
        System.out.println("Enter the name of the second opponent club: ");
        String opp02 = footballClubInput.nextLine().toUpperCase().trim();

        //Retrieving two footballClub from the premier league based on the entered
name of each opponent
        FootballClub opponent01 = premierLeagueManager.retrieveFootballClub(opp01);
        FootballClub opponent02 = premierLeagueManager.retrieveFootballClub(opp02);

```

```

//Input validation in case, the club name entered does not actually exist in the
league
if(opponent01 == null || opponent02 == null) {
    if(opponent01 == null && opponent02 == null) {
        System.out.println("Both clubs do not exist in the Premier
League!");
    }
    else if(opponent01 == null) {
        System.out.println(opp01 + " does not exist in the Premier
League!");
    }
    else{
        System.out.println(opp02 + " does not exist in the Premier
League!");
    }
}

//Input validation in case, both of the same clubName has been entered twice.
else if(opp01.equals(opp02)) {
    System.out.println("The same opponent cannot compete each other!");
}

else {
    try {
        System.out.println("Enter the score of the first opponent: ");
        int score01 = footballClubInput.nextInt();

        System.out.println("Enter the score of the second opponent: ");
        int score02 = footballClubInput.nextInt();

        //The following condition prevents negative scores from been
entered
        if (score01 < 0 || score02 < 0) {
            System.out.println("Negative scores cannot occur
during a match!");
        } else {
            // try catch has been used to catch incorrect date
            inputs
            System.out.println("Enter the day of the match: ");
            int day = footballClubInput.nextInt();

            System.out.println("Enter the month of the match: ");
            int month = footballClubInput.nextInt();

            int year = Calendar.getInstance().get(Calendar.YEAR);

```

```

        FootballMatch footballMatch = new
FootballMatch(opponent01, opponent02, score01, score02, LocalDate.of(year, month, day));

        premierLeagueManager.addPlayedMatch(footballMatch, false);
    }
    }
    catch(InputMismatchException inputMismatchException){
        System.out.println("Please enter a valid int value!");
    }
    catch (DateTimeException dateTimeException) {
        System.out.println("An invalid date has been entered!");
    }
    }
}
else {
    //Only this msg will be printed if the Premier League contains 0 or 1 clubs
    System.out.println("Please add at least two Football clubs in the league, to
initiate a match!");
}
}
//The following methods below were implemented for the use of the JavaFX GUI
public static void retrieveSportsClubInfo() throws IOException, ClassNotFoundException {
    premierLeagueManager.retrieveSportsClubInfo("premierLeagueFile.txt");
}
public static void saveSportsClubInfo() throws IOException {
    premierLeagueManager.saveSportsClubInfo("premierLeagueFile.txt");
}
//returns the football league arraylist for use in the leagueTableView
public static List<FootballClub> retrieveLeagueList(){
    return premierLeagueManager.returnClubs();
}
//returns the football match arraylist for use in the matchTableView
public static List<FootballMatch> retrieveMatchList(){
    return premierLeagueManager.returnMatches();
}

//Returns a reference to the premier league manager for use in the backend controller methods
public static PremierLeagueManager getPremierLeagueManager() {
    return premierLeagueManager;
}

/**
 * The following method is used to initiate the PlayFramework & Angular host
 * this method will be only called once whenever the console application is run*/
public static void initiatePlayAngular() throws IOException {
    if (!initiateGUI) {
        /**The following process is used to launch cmd.exe and execute a specified cmd
prompt

```

```

framework backend
option in the console

    *from the existing project path. This will automatically launch the play-
    * which then automatically runs the Angular local host, this was given as an
    * as required from the rubric marking sheet*/
    Runtime.getRuntime().exec("cmd /c start cmd.exe /K \"sbt run\"");
    initiateGUI = true;
}
else{
    System.out.println("Both Play-frame and Angular host are already running");
}
}
}

```

## 2.2 Controllers

### 2.2.1 LeagueController

```

package controllers;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
import consoleApplication.PremierLeagueConsole;
import play.libs.Json;
import play.mvc.Controller;
import play.mvc.Result;

import java.io.IOException;

public class LeagueController extends Controller {
    private static PremierLeagueConsole premierLeagueController = new PremierLeagueConsole();

    /**The following GET method will return the FootballClub arraylist into JSON format
    *to backend localhost:9000 which will be accessed and displayed into the Angular host,
    the following list will be loading the following clubs saved in the serialized file*/
    public Result listPremierLeague() throws IOException, ClassNotFoundException {
        premierLeagueController.retrieveSportsClubInfo();
        ObjectMapper leagueMapper = new ObjectMapper();
        JsonNode leagueData = leagueMapper.convertValue(premierLeagueController.retrieveLeagueList(),
        JsonNode.class);

        System.out.println((premierLeagueController.retrieveLeagueList()));
        return ok(leagueData).as("application/json");
    }

    /**The following GET method will return the FootballMatch arraylist into JSON format
    *to backend localhost:9000 which will be accessed and displayed into the Angular host,

```

```

the following list will be loading the following matches saved in the serialized file*/
public Result listPremierMatches() throws IOException, ClassNotFoundException {

    premierLeagueController.retrieveSportsClubInfo();
    ObjectMapper matchMapper = new ObjectMapper();
    matchMapper.registerModule(new JavaTimeModule());
    //The following mapper will reformat Date attribute into a readable format in JSON
    //Tutorial referenced from https://stackoverflow.com/questions/28802544/java-8-localdate-jackson-format
    matchMapper.configure(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS, false);

    JsonNode matchData = matchMapper.convertValue(premierLeagueController.retrieveMatchList(),
    JsonNode.class);
    System.out.println(premierLeagueController.retrieveMatchList());

    return ok(matchData).as("application/json");
}

/**The following GET method will call out the premierleagueconsole random match method
 * which would add a match each time the method is called */
public Result generateRandomMatch() throws IOException, ClassNotFoundException {
    premierLeagueController.retrieveSportsClubInfo();
    String matchMsg =
premierLeagueController.getPremierLeagueManager().randomPlayedFootballMatch();
    ObjectMapper randomMatchMapper = new ObjectMapper();
    //Reformatting
    String matchData =
randomMatchMapper.writerWithDefaultPrettyPrinter().writeValueAsString(matchMsg);

    premierLeagueController.saveSportsClubInfo();
    return ok(matchData).as("application/json");
}

/**The following method is a summary of the premier league*/
public Result premierLeagueSummary() throws IOException, ClassNotFoundException {
    premierLeagueController.retrieveSportsClubInfo();
    String summaryMsg = ("There are " + premierLeagueController.retrieveLeagueList().size() + " football
clubs, placed into the league & "+
    + premierLeagueController.retrieveMatchList().size()+ " matches have already taken place.");
    return ok(Json.toJson(summaryMsg)).as("application/json");
}
}

```

## 2.3 Entities

### 2.3.1 SportsClub.java

```
package entities;
```

```
import java.io.Serializable;
```

```
import java.util.Objects;
```

```
/**
```

```
*The SportsClub class is the main parent object, the following fields  
are based on the information of the sports club which is then passed on to its child object the  
FootBallClub*/
```

```
public abstract class SportsClub implements Serializable{
```

```
    private String clubName;
```

```
    private String clubLocation;
```

```
    private int numberOfPlayers;
```

```
    private int yearEstablished;
```

```
//Default Constructor for FootballClub
```

```
    public SportsClub(String clubName, String clubLocation, int numberOfPlayers, int yearEstablished) {
```

```
        this.clubName = clubName;
```

```
        this.clubLocation = clubLocation;
```

```
        this.numberOfPlayers = numberOfPlayers;
```

```
        this.yearEstablished = yearEstablished;
```

```
    }
```

```
    //returns the club name
```

```
    public String getClubName() {
```

```
        return clubName;
```

```
    }
```

```
    //returns the club location
```

```
    public String getClubLocation() {
```

```
        return clubLocation;
```

```
    }
```

```
    //returns the player base
```

```
    public int getNumberOfPlayers() {
```

```
        return numberOfPlayers;
```

```
    }
```

```
    //returns the established year of the club
```

```
    public int getYearEstablished() {
```

```
        return yearEstablished;
```

```
    }
```

```
    //updates the club name
```

```
    public void setClubName(String clubName) {
```

```
        this.clubName = clubName;
```

```
    }
```

```
    //updates the club location
```

```
    public void setClubLocation(String clubLocation) {
```

```
        this.clubLocation = clubLocation;
```

```
    }
```

```

//updates the number of players in the club
public void setNumberOfPlayers(int numberOfPlayers) {
    this.numberOfPlayers = numberOfPlayers;
}

public void setYearEstablished(int yearEstablished) {
    this.yearEstablished = yearEstablished;
}

@Override
public boolean equals(Object object) {
    if (this == object)
        return true;

    if(object instanceof SportsClub) {
        SportsClub sportsClub = (SportsClub)object;
        return sportsClub.clubName.equals(this.clubName);
    }

    else
        return false;
}

@Override
public int hashCode() {
    return Objects.hash(this.clubName, this.clubLocation, this.numberOfPlayers, this.yearEstablished);
}

//This is the default format returned when printing a SportsClub object.
@Override
public String toString() {
    return "CLUB INFORMATION" +
        String.format("%-17s", "\nCLUB NAME") + " --> " + this.clubName +
        String.format("%-17s", "\nLOCATION") + " --> " + this.clubLocation +
        String.format("%-17s", "\nPLAYER BASE") + " --> " + this.numberOfPlayers +
        String.format("%-17s", "\nYEAR ESTABLISHED") + " --> " + this.yearEstablished;
}
}

```

### 2.3.2 FootballClub.java

```
package entities;
```

```
import java.util.Objects;
```

```
/**
```

```
*The FootballClub class extends from the SportClub object
```

```
which inherits its methods.*
```

```
public class FootballClub extends SportsClub implements Comparable<FootballClub> {
```



```

private int clubWins; /*Number of matches won*/
private int clubDraws; /*Number of matches tied*/
private int clubDefeats; /*Number of matches lost*/
private int goalsScored; /*Totals goals scored from matches*/
private int goalsReceived; /*Goals won against opponent*/
private int numberOfMatchesPlayed; /*The total numbers of matches based on wins, draws and
losses*/

private int points; /*The points based on the wins, draws and losses*/
private int goalDifferenceClub; /*Difference between the goals scored and the goals received*/


//Default Constructor for FootballClub
public FootballClub(String clubName, String clubLocation, int numberOfPlayers, int
yearEstablished){
    super(clubName, clubLocation, numberOfPlayers, yearEstablished);
}
//Overloaded Football Method Constructor
public FootballClub(String clubName, String clubLocation, int numberOfPlayers, int yearEstablished,
int clubWins, int clubDraws, int clubDefeats, int goalsScored, int goalsReceived) {
    //calling out the sportsclub constructor
    super(clubName, clubLocation, numberOfPlayers, yearEstablished);
    this.clubWins = clubWins;
    this.clubDraws = clubDraws;
    this.clubDefeats = clubDefeats;
    this.goalsScored = goalsScored;
    this.goalsReceived = goalsReceived;
    this.numberOfMatchesPlayed = clubWins + clubDraws + clubDefeats;
}
//returns number of matches won
public int getClubWins() {
    return clubWins;
}
//returns number of matches drawn
public int getClubDraws() {
    return clubDraws;
}
//returns number of matches lost
public int getClubDefeats() {
    return clubDefeats;
}
//returns goals recieved
public int getGoalsReceived() {
    return goalsReceived;
}

//returns goals scored
public int getGoalsScored() {
    return goalsScored;
}

```

```

//returns number of matches played
public int getNumberOfMatchesPlayed() {
    return numberOfMatchesPlayed;
}

//returns the goal-difference
public int getGoalDifferenceClub() {return goalDifferenceClub; }

//returns the points of the club
public int getPoints() {
    return points;
}

//used to update club wins
public void setClubWins(int clubWins) {
    this.clubWins = clubWins;
}

//used to update club draws
public void setClubDraws(int clubDraws) {
    this.clubDraws = clubDraws;
}

//used to update club defeats
public void setClubDefeats(int clubDefeats) {
    this.clubDefeats = clubDefeats;
}

//used to update goals received
public void setGoalsReceived(int goalsReceived) {
    this.goalsReceived = goalsReceived;
}

//used to update goals scored
public void setGoalsScored(int goalsScored) {
    this.goalsScored = goalsScored;
}

//used to update number of played matches
public void setNumberOfMatchesPlayed(int numberOfMatchesPlayed) {
    this.numberOfMatchesPlayed = numberOfMatchesPlayed;
}

//used to update goal difference based on match scores
public void setGoalDifferenceClub(int goalDifference) {
    this.goalDifferenceClub = goalDifference;
}

```

```

//used to update points earned based on the outcome of the match
public void setPoints(int points) {
    this.points = points;
}

//A separate method used to calculate the football club points which is used for stats & sorting
public int calculateFootballClubPoints() {
    return this.clubWins * 3 + this.clubDraws * 1 + this.clubDefeats * 0;
}

//A separate method used to calculate the football goal difference which is necessary for sorting
public int calculateGoalDifference() {
    return this.goalsScored - this.goalsReceived;
}

//Method used to update Football stats once a match has been called out
public void addGoalsScoreAndReceived(int matchScore, int opponentMatchScore){
    this.goalsScored += matchScore;
    this.goalsReceived += opponentMatchScore;
    this.numberOfMatchesPlayed += 1;
}

//The following equals methods check to see if two football objects are the same by checking both
club names
@Override
public boolean equals(Object object) {
    if (this == object)
        return true;

    if(object instanceof FootballClub){
        return super.equals(object);
    }

    if(object instanceof SportsClub){
        return super.equals(object);
    }

    else{
        return false;
    }
}

@Override
public int hashCode() {
    return Objects.hash(super.hashCode(), clubWins, clubDraws, clubDefeats, goalsScored,
goalsReceived, numberOfMatchesPlayed);
}

```

```

        /*compareTo method is the default order in which the football clubs are sorted based on club
points and goal difference.
inorder to override the method, the FootballClub has implemented the Comparable interface*/
@Override
public int compareTo(FootballClub footballClub) {
    int comparePoints = footballClub.calculateFootballClubPoints() -
this.calculateFootballClubPoints();
    if (comparePoints == 0) {
        if (this.calculateGoalDifference() > footballClub.calculateGoalDifference())
            return -1;
        else {
            if (this.calculateGoalDifference() <
footballClub.calculateGoalDifference())
                return 1;
            else
                return 0;
        }
    }
    return comparePoints;
}
//This is the default format returned when printing a FootballClub object.
@Override
public String toString() {
    return "\n" + super.toString() + "\n" + "\nCLUB STATISTICS" +
String.format("%-17s", "\nMATCHES PLAYED") + " --> " + this.numberOfMatchesPlayed +
String.format("%-17s", "\nWINS") + " --> " + this.clubWins +
String.format("%-17s", "\nDRAWS") + " --> " + this.clubDraws +
String.format("%-17s", "\nDEFEATS") + " --> " + this.clubDefeats +
String.format("%-17s", "\nGOALS SCORED") + " --> " + this.goalsScored +
String.format("%-17s", "\nGOALS CONCEDED") + " --> " + this.goalsReceived +
String.format("%-17s", "\nGOAL DIFFERENCE") + " --> " + this.calculateGoalDifference() +
String.format("%-17s", "\nPOINTS") + " --> " + this.calculateFootballClubPoints();
}
}

```

### 2.3.3 UniversityFootballClub.java

```
package entities;
```

```
import java.util.Objects;
```

```

/*The UniversityFootballClub class extends from the FootballClub object
which inherits its methods.*/
public class UniversityFootballClub extends FootballClub {
    private String universityName;
    private static final int MAX_AGE_LIMIT = 23; //Constant set based on the mentioned age limit

    //Default Constructor for UniversityFootballClub
    public UniversityFootballClub(String clubName, String clubLocation, int numberOfPlayers, int
yearEstablished, String universityName) {
        super(clubName, clubLocation, numberOfPlayers, yearEstablished);
    }
}

```

```

        this.universityName = universityName;
    }

    //Overloaded method Constructor for UniversityFootballClub
    public UniversityFootballClub(String clubName, String clubLocation, int numberOfPlayers, int
yearEstablished,
        int clubWins, int clubDraws, int clubDefeats, int goalsScored, int goalsReceived, String
universityName) {
        super(clubName, clubLocation, numberOfPlayers, yearEstablished, clubWins, clubDraws,
clubDefeats, goalsScored, goalsReceived);
        this.universityName = universityName;
    }

    //returns university name
    public String getUniversityFootballClubName() {
        return universityName;
    }

    //update university name
    public void setUniversityFootballClubName(String universityName) {
        this.universityName = universityName;
    }

    @Override
    public boolean equals(Object object) {
        if(this == object)
            return true;

        if (object instanceof UniversityFootballClub){
            UniversityFootballClub universityFootballClub = (UniversityFootballClub) object;
            return universityFootballClub.universityName == universityName;
        }

        if(object instanceof FootballClub) {

            if(object instanceof SchoolFootballClub)
                return false;
            else
                return super.equals(object);
        }

        if(object instanceof SportsClub)
            return super.equals(object);

        else
            return false;
    }

    @Override

```

```

        public int hashCode() {
            return Objects.hash(super.hashCode(), universityName);
        }

        //This is the default format returned when printing a UniversityFootballClub object.
        @Override
        public String toString() {
            return super.toString() + String.format("%-17s", "\nUNIVERSITY NAME") + " --> " +
this.universityName;
        }

    }

```

#### 2.3.4 SchoolFootballClub.java

package entities;

import java.util.Objects;

```

/**
 *The SchoolFootballClub class extends from the FootballClub object
 which inherits its methods.*/
public class SchoolFootballClub extends FootballClub {
    private String schoolName;
    private static final int MAX_AGE_LIMIT = 18; //Constant set based on the mentioned age limit

    //Default Constructor for UniversityFootballClub
    public SchoolFootballClub(String clubName, String clubLocation, int numberOfPlayers, int
yearEstablished, String schoolName) {
        super(clubName, clubLocation, numberOfPlayers, yearEstablished);
        this.schoolName = schoolName;
    }

    //Overloaded method Constructor for SchoolFootballClub
    public SchoolFootballClub(String clubName, String clubLocation, int numberOfPlayers, int
yearEstablished,
        int clubWins, int clubDraws, int clubDefeats, int goalsScored, int goalsReceived, String
schoolName) {
        super(clubName, clubLocation, numberOfPlayers, yearEstablished, clubWins, clubDraws,
clubDefeats, goalsScored, goalsReceived);
        this.schoolName = schoolName;
    }

    //returns school name
    public String getSchoolName() {
        return schoolName;
    }

    //changes school name
    public void setSchoolName(String schoolName) {

```

```

        this.schoolName = schoolName;
    }

    @Override
    public boolean equals(Object object) {
        if(this == object)
            return true;

        if(object instanceof SchoolFootballClub) {
            SchoolFootballClub schoolFootballClub = (SchoolFootballClub) object;
            return schoolFootballClub.schoolName == schoolName && super.equals(object);
        }

        if(object instanceof FootballClub) {

            if(object instanceof UniversityFootballClub)
                return false;
            else
                return super.equals(object);
        }

        if(object instanceof SportsClub)
            return super.equals(object);

        else
            return false;
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), schoolName);
    }

    //This is the default format returned when printing a SchoolFootballClub object.
    @Override
    public String toString() {
        return super.toString() + String.format("%-17s", "\nSCHOOL NAME") + " --> " +
this.schoolName;
    }
}

```

### 2.3.5 FootballMatch.java

```
package entities;
```

```
import java.io.Serializable;
import java.time.LocalDate;
import java.util.Objects;
```

```

/**
 *The following FootballMatch object was created, inorder to place a match between two football clubs
 where the premier league manager choses the desired scores along with the date of the match
 */
//Default constructor for a FootballMatch
public class FootballMatch implements Serializable, Comparable<FootballMatch> {
    private FootballClub Opponent01;
    private FootballClub Opponent02;
    private int matchScoreOpponent01;
    private int matchScoreOpponent02;
    private LocalDate matchDate;

    public FootballMatch(FootballClub Opponent01, FootballClub Opponent02, int
matchScoreOpponent01, int matchScoreOpponent02,
    LocalDate matchDate) {
        this.Opponent01 = Opponent01;
        this.Opponent02 = Opponent02;
        this.matchScoreOpponent01 = matchScoreOpponent01;
        this.matchScoreOpponent02 = matchScoreOpponent02;
        this.matchDate = matchDate;
    }

    public int getMatchScoreOpponent01() {
        return matchScoreOpponent01;
    }

    public int getMatchScoreOpponent02() {
        return matchScoreOpponent02;
    }

    /**The following method is used to update each opponent's stats,
    **which is only called once a FootballMatch has created
    if it has gone through the necessary validation*/
    public void updateFootballClubStats(){
        Opponent01.addGoalsScoreAndReceived(matchScoreOpponent01,
matchScoreOpponent02);

        Opponent02.addGoalsScoreAndReceived(matchScoreOpponent02,matchScoreOpponent01);

        Opponent01.setGoalDifferenceClub(Opponent01.getGoalsScored()-
Opponent01.getGoalsReceived());
        Opponent02.setGoalDifferenceClub(Opponent02.getGoalsScored()-
Opponent02.getGoalsReceived());

        if(matchScoreOpponent01 > matchScoreOpponent02) {
            Opponent01.setClubWins(Opponent01.getClubWins() + 1);
            Opponent02.setClubDefeats(Opponent02.getClubDefeats() + 1);
        }
    }
}

```



```

        else if(matchScoreOpponent01 < matchScoreOpponent02) {
            Opponent02.setClubWins(Opponent02.getClubWins() + 1);
            Opponent01.setClubDefeats(Opponent01.getClubDefeats() + 1);
        }
        else {
            Opponent01.setClubDraws(Opponent01.getClubDraws() + 1);
            Opponent02.setClubDraws(Opponent02.getClubDraws() + 1);
        }
        Opponent01.setPoints(Opponent01.calculateFootballClubPoints());
        Opponent02.setPoints(Opponent02.calculateFootballClubPoints());
    }
    //Used to return the first football club opponent
    public FootballClub getOpponent01() {
        return Opponent01;
    }
    //Used to return the second football club opponent
    public FootballClub getOpponent02() {
        return Opponent02;
    }
    //Used to return the date of which the match was played at.
    public LocalDate getMatchDate() {
        return matchDate;
    }
}

/**The following method footballMatchDateCheck is used
to identify the number of occurrences of two teams competing
amongst themselves. So if there is at least one instance of
a match happening with one opponent, it would then return true.*/

public boolean footballMatchDateCheck(FootballMatch footballMatch) {
    return footballMatch.Opponent01.equals(Opponent01) ||
        footballMatch.Opponent02.equals(Opponent02) ||
        footballMatch.Opponent01.equals(Opponent02) ||
        footballMatch.Opponent02.equals(Opponent01);
}

@Override
public boolean equals(Object object) {
    if (this == object)
        return true;
    if (object instanceof FootballMatch){
        FootballMatch footballMatch = (FootballMatch)object;
        return Opponent01.equals(footballMatch.Opponent01) &&
            Opponent02.equals(footballMatch.Opponent02);
    }
    else
        return false;
}

```

```

@Override
public int hashCode() {
    return Objects.hash(Opponent01, Opponent02, matchScoreOpponent01,
matchScoreOpponent02, matchDate);
}

@Override
public int compareTo(FootballMatch footballMatch) {
    return this.getMatchDate().compareTo(footballMatch.getMatchDate());
}

@Override
public String toString() {
    return "\n----SCORE RESULTS----" + String.format("%-12s", "\nMatch Date") + " --> " +
this.matchDate +
        String.format("%-12s", "\n"+Opponent01.getClubName()) + " --> " +
matchScoreOpponent01 +
        String.format("%-12s", "\n"+Opponent02.getClubName()) + " --> " +
matchScoreOpponent02 +
        String.format("\n%s", matchScoreOpponent01 > matchScoreOpponent02 ?
            Opponent01.getClubName() + " has won the match against " +
Opponent02.getClubName()
            : matchScoreOpponent01 < matchScoreOpponent02 ?
            Opponent02.getClubName() + " has won the match against " +
Opponent01.getClubName():
            "The following match is a tie!");
}
}

```

## 2.4.Services

### 2.4.1 LeagueManager.java

```
package services;
```

```
import entities.FootballMatch;
```

```
import entities.SportsClub;
```

```
import java.io.IOException;
```

```
public interface LeagueManager {
    /* addSportsClub method is used to add a club, */
    void addSportsClub(SportsClub sportsClub);
    /* relegateSportsClub method is used to remove a club, based on the club name */
    void relegateSportsClub(String clubName);
    /* showSportsClubStats method is used to view a club stats and info, based on the club name */
    void showSportsClubStats(String clubName);
    /* showSportsLeagueStats method is used to view the stats of each sports club in a league*/
}

```

```

        void showSportsLeagueStats();
        /* addPlayedMatch method is used to generate a simulation of match between two clubs*/
        void addPlayedMatch(FootballMatch footballMatch, boolean matchIdentified);
        /* saveSportsClubInfo method is used to generate a text file containing the objects*/
        void saveSportsClubInfo(String fileName) throws IOException;
        /* retrieveSportsClubInfo method is used to load previously saved data*/
        void retrieveSportsClubInfo(String fileName) throws IOException, ClassNotFoundException;
    }

```

## 2.4.2 PremierLeagueManager.java

```
package services;
```

```

import java.io.*;
import java.time.LocalDate;
import java.time.Month;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

import entities.*;

/**
 *The PremierLeagueManager class implements the LeagueManager interface
 which inherits the following methods needed to add, remove, and view
 statistics of a specified football club along with the league table of
 all existing Football clubs.*/
public class PremierLeagueManager implements LeagueManager {
    private static List<SportsClub> sportsLeague = new ArrayList<>(); //The List used to store football
clubs
    private static List<FootballMatch> matchList = new ArrayList<>();

    private static final int MAX_LEAGUE_COUNT = 20; //The maximum
limit of clubs the Premier League can hold
    private static final int MAX_NUM_MATCHES = 380;

    /**
     * addSportsClub method is used to add a club,
     which is stored to the sportsLeague ArrayList*/
    @Override
    public void addSportsClub(SportsClub sportsClub) {
        if(sportsLeague.size() < MAX_LEAGUE_COUNT) {
            for(SportsClub individualSportsClub: sportsLeague) {

                if((sportsClub.getClubName()).equals(individualSportsClub.getClubName())) {
                    //Prints the following message, if the same club is already
entered more than once.

```

```

        System.out.println("The following football club is already in the
Premier League");
        return;
    }
    if(sportsClub instanceof FootballClub) {
        sportsLeague.add(sportsClub);
        //Prints the following message, after club is added in to sportsLeague.
        System.out.println("Football club successfully added into the Premier
League");
    }
    else {
        /*Prints the following message, if the sportsLeague arraylist size has reached
the maximum limit of the premier league.*/
        System.out.println("The Premier League is at full capacity!");
    }
}
/**
 * relegateSportsClub method is used to remove a club,
which is removed from the sportsLeague ArrayList*/
@Override
public void relegateSportsClub(String clubName) {
    boolean clubIdentified = false;
    for(SportsClub individualSportsClub: sportsLeague) {
        if(clubName.equals(individualSportsClub.getClubName())){
            sportsLeague.remove(individualSportsClub);
            clubIdentified = true;
            //Prints the following message, after club is removed from the
sportsLeague.
            System.out.println(clubName + " has been removed from the Premier
League");
            break;
        }
    }
    if(!clubIdentified){
        //Prints the following message, is the club to be removed does not exist in the
sportsLeague.
        System.out.println("The following club is not apart of the Premier League");
    }
}

@Override
public void showSportsClubStats(String clubName) {
    boolean clubIdentified = false;
    for(SportsClub individualSportsClub: sportsLeague) {
        if(clubName.equals(individualSportsClub.getClubName())) {

```

```

        clubIdentified = true;
        //Prints out the following fields and messages mentioned in the
overridden toString method
        System.out.println(individualSportsClub);
        break;
    }
}
if(!clubIdentified){
    //Prints the following message, is the club to be displayed does not exist in the
sportsLeague.
    System.out.println("The following club is not apart of the Premier League");
}
}

@Override
public void showSportsLeagueStats(){
    List<FootballClub> footballLeague = new ArrayList<>();
    for(SportsClub individualSportsClub: sportsLeague){
        //downcasting each sportsClub into its child entity (footballClub)
        footballLeague.add((FootballClub) individualSportsClub);
    }
    //Used to sort out the entire league of football clubs
    Collections.sort(footballLeague);
    //League Table String format

    System.out.println("_____");
    System.out.printf("%-10s%-30s%6s%7s%7s%9s%10s%12s%14s%8s\n",
    "|POSITION", "CLUB NAME", "PLAYED", " WINS", " DRAWS", " DEFEATS",
    " G-SCORED", " G-RECEIVED", " G-DIFFERENCE", " POINTS |");
    System.out.println("|-----|");
    -----|");

    int positionCount = 1;
    for(FootballClub individualFootballClub: footballLeague){
        System.out.printf("%-4s%-6s%-28s%6s%8s%6s%9s%9s%11s%13s%10s%4s\n",
"|",
        positionCount, individualFootballClub.getClubName().toUpperCase(),
individualFootballClub.getNumberOfMatchesPlayed(),
        individualFootballClub.getClubWins(), individualFootballClub.getClubDraws(),
individualFootballClub.getClubDefeats(),
        individualFootballClub.getGoalsScored(),
individualFootballClub.getGoalsReceived(), individualFootballClub.calculateGoalDifference(),
        individualFootballClub.calculateFootballClubPoints(), "|");

        System.out.println("|_____");
        positionCount++;
    }
}

```

```

    }

    @Override
    public void addPlayedMatch(FootballMatch footballMatch, boolean matchIdentified) {
        matchIdentified = false;
        for(FootballMatch individualFootballMatch: matchList) {
            if(footballMatch.equals(individualFootballMatch)) {
                matchIdentified = true;
                //Checks to see if the same match has happend before.
                System.out.println("The following match has already taken place!");
                break;
            }
            /*The following condition has been placed to prevent from adding a match with
            an opponent who has been already placed on another which is on the same
            date*/
            else
            if(footballMatch.getMatchDate().equals(individualFootballMatch.getMatchDate())) {
                if ((footballMatch.footballMatchDateCheck(individualFootballMatch) ||
                individualFootballMatch.footballMatchDateCheck(footballMatch))) {
                    matchIdentified = true;
                    System.out.println("Either one or both opponents have been
                    allocated, to a different match during this date");
                    break;
                }
                break;
            }
        }

        if(!matchIdentified) {
            //Validation to prevent the same club going against itself
            if(footballMatch.getOpponent01().equals(footballMatch.getOpponent02())) {
                System.out.println("A football club cannot compete amongst its self");
            }

            else {
                /*The match would be added and opponents stats would become
                updated

                **only until it passes through the above validations*/
                footballMatch.updateFootballClubStats();
                matchList.add(footballMatch);
                //System.out.println(footballMatch);
            }
        }
    }

    @Override
    public void saveSportsClubInfo(String fileName) throws IOException {
        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
    }

```

```

//first line would contain the length of the sports league arraylist
objectOutputStream.writeObject(sportsLeague.size());
//next line would contain the length of the match arraylist
objectOutputStream.writeObject(matchList.size());

//From here on, all Sportsclub objects would be written in the file
for(SportsClub sportsClub : sportsLeague) {
    objectOutputStream.writeObject(sportsClub);
}
//And then finally all Football matches would be written last in the file
for(FootballMatch footballMatch : matchList) {
    objectOutputStream.writeObject(footballMatch);
}
}

@Override
public void retrieveSportsClubInfo(String fileName) throws IOException, ClassNotFoundException {
    int numOfFootballClubs;
    int numOfFootballMatches;

    sportsLeague.clear();
    matchList.clear();

    try{
        FileInputStream fileInputStream = new FileInputStream(fileName);
        ObjectInputStream objectInputStream = new
ObjectInputStream(fileInputStream);

        for(;;) {
            try {
                /**From here the both arraylist sizes would be retrieved
                **And based on the length, each line would read each object
                which would be placed in the appropriate arraylist based on
                the size of each arraylist*/
                numOfFootballClubs = (int)objectInputStream.readObject();
                numOfFootballMatches = (int)objectInputStream.readObject();

                for (int lineNo = 0; lineNo < numOfFootballClubs; lineNo++) {
                    sportsLeague.add((SportsClub)objectInputStream.readObject());
                }

                for (int nxtLine = 0; nxtLine < numOfFootballMatches;
nxtLine++) {
                    matchList.add((FootballMatch)
objectInputStream.readObject());
                }
            }
            catch EOFException e {
                break;
            }
        }
    }
}

```

```

    }
}
} //Used to create a file if it doesn't exist
catch(FileNotFoundException f){
    //System.out.println(fileName +" file not available!");
    saveSportsClubInfo(fileName);
}
}

public String randomPlayedFootballMatch() {
    String responseMsg = null;

    int countMatch = 0;
    //Retrieving footballLeague size
    int leagueSize = getSportsLeagueLength();
    //380 matches can only occurs by the maximum (20 * (20 - 19))
    int maxNumberOfMatches = MAX_NUM_MATCHES;

    //If less than two clubs are added a alert will pop up.
    if(leagueSize > 1) {
        boolean randomMatchLoop = true;
        while(randomMatchLoop) {
            //Picking random football clubs by picking out a random index value.
            FootballClub opponent01 = returnClubs().get(new
Random().nextInt(leagueSize));
            FootballClub opponent02 = returnClubs().get(new
Random().nextInt(leagueSize));

            if (opponent01.equals(opponent02)) {
                continue;
            }
            int maxMatchScore = 10;
            int minMatchScore = 0;
            int score01 = new Random().nextInt((maxMatchScore - minMatchScore)
+ 1) + minMatchScore;
            int score02 = new Random().nextInt((maxMatchScore - minMatchScore)
+ 1) + minMatchScore;

            LocalDate startDate = LocalDate.of(2021, Month.JANUARY, 1);
            LocalDate endDate = LocalDate.of(2021, Month.DECEMBER, 31);
            //Used to find the difference in the starting & ending time period
            long days = ChronoUnit.DAYS.between(startDate, endDate);
            LocalDate randomMatchDate = startDate.plusDays(new
Random().nextInt((int) days + 1));
            FootballMatch randomFootballMatch = new FootballMatch(opponent01,
opponent02, score01, score02, randomMatchDate);

            /** Used to avoid matches from been repeated, if so the loop will
continue running through

```



```

        all 380 match fixtures */
        boolean matchPlaced = false;
        for (FootballMatch individualFootballMatch : returnMatches()) {
            if (randomFootballMatch.equals(individualFootballMatch)) {
                matchPlaced = true;
                break;
            }
        }
        countMatch += 1;
        //Condition below runs the following loop at a maximum of 380 loops
        if (maxNumberOfMatches == (leagueSize * (leagueSize - 1)) ||
maxNumberOfMatches == countMatch) {
            //System.out.println("All possible matches have been played!");
            responseMsg = "All possible matches have been played!";
            System.out.println(responseMsg);
            return responseMsg;

        } else if (matchPlaced) {
            continue;
        } else {
            boolean matchOccupied = false;
            addPlayedMatch(randomFootballMatch, matchOccupied);
            if (matchOccupied){
                continue;
            }
            else {
                String opponent01Name =
randomFootballMatch.getOpponent01().getClubName();
                String opponent02Name =
randomFootballMatch.getOpponent02().getClubName();

                responseMsg = String.valueOf("-----" +
-----" +
took place on ") + randomMatchDate +
opponent01Name) + " scored " + score01 + " goals" +
opponent02Name) + " scored " + score02 + " goals "+
-----" +
> score02 ?
" has won the match against " + opponent02Name
?
" has won the match against " + opponent01Name

                String.format("%-16s", "<br> Match
String.format("%-16s", "<br>" +
String.format("%-16s", "<br>" +
"<br>-----"
String.format("<br><br>%s", score01
opponent01Name +
: score01 < score02
opponent02Name +

```

```

match is a tie!"));

                                System.out.println(randomFootballMatch);
                                return responseMsg;
                                }
                                }
                                }
else {
    responseMsg = "Please add at least two clubs to run a match!";
    System.out.println(responseMsg);
    return responseMsg;
}
return responseMsg;
}

//Additional method created for placing football clubs into a football match
public static FootballClub retrieveFootballClub(String clubName) {
    for (SportsClub individualSportsClub : sportsLeague) {
        if (clubName.equals(individualSportsClub.getClubName())) {
            return (FootballClub)individualSportsClub;
        }
    }
    return null;
}

/*Additional method created to return arraylist as a validation constraint
for adding a football match in the console.*/
public static int getSportsLeagueLength(){
    return sportsLeague.size();
}

public static List<FootballClub> returnClubs() {
    List<FootballClub> footballLeague = new ArrayList<>();
    for(SportsClub individualSportsClub: sportsLeague){
        //down-casting each sportsClub into its child entity (footballClub)
        footballLeague.add((FootballClub) individualSportsClub);
    }
    Collections.sort(footballLeague);
    return footballLeague;
}

public static List<FootballMatch> returnMatches(){
    Collections.sort(matchList);
    return matchList;
}
}

```

### 3. Angular GUI src

#### 3.1 app component

##### 3.1.1 app.component.html

```
<mat-toolbar color="primary">
  <mat-toolbar-row>
    <span></span>
    <span style="font-size: 30px">PREMIER LEAGUE</span>
  </mat-toolbar-row>
</mat-toolbar>

<div class = "page-container">
  <div class="colored box" style="cursor: pointer; flex-direction: row; box-sizing: border-box; display: flex;">
    <div style="flex: 1 1 100%; box-sizing: border-box; max-width: 10%;"></div>
    <div fxflex="60" style="flex: 1 1 100%; box-sizing: border-box; max-width: 80%;">
      <router-outlet></router-outlet>
    </div>
    <div fxflex style="flex: 1 1 0%; box-sizing: border-box;"></div>
  </div>
</div>
```

##### 3.1.2 app.component.scss

```
#logo{
  width: 52px;
  height: 60px;
  padding-right: 10px;
}

table, th, td
{
  margin: 10px 0;
  border: solid 1px #333;
  padding: 2px 4px;
  font: 15px Verdana;
}

th {
  font-weight:bold;
}
```

##### 3.1.3 app.service.ts

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {map} from 'rxjs/operators';
```

/\*The following service page is used to extract all JSON data sent from the backend each url is based on the backend controller method from the PremierLeagueConsole\*\*/

```
@Injectable()
export class appService {
  //FootballClub list
  private leagueUrl = '/api/league';
  //FootballMatch list
  private matchUrl = '/api/matches';
  //Random match method from java src
  private randomMatch = '/api/randomMatch';

  constructor(private http: HttpClient) {
  }
  public getPremierClubs() {
    return this.http.get(this.leagueUrl).pipe(
      map(response => response)
    );
  }
  public getPremierMatches() {
    return this.http.get(this.matchUrl).pipe(
      map(response => response)
    );
  }
  public getRandomMatchMsg(){
    return this.http.get(this.randomMatch).pipe(
      map(response => response)
    );
  }
}
```

### 3.1.4 app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { MatchComponent } from './match/match.component';
import { LeagueTableComponent } from './league-table/league-table.component';
```

```
const routes: Routes = [
  {path: '',component: LeagueTableComponent},
  {path: 'Match', component: MatchComponent}
];
```

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingComponents = [LeagueTableComponent, MatchComponent];
```

## 3.2 league-table component

### 3.2.1 league-table.component.html

```
<div fxLayout fxLayoutAlign="left center">
  <h2 style="color: white">PREMIER LEAGUE TABLE</h2>
</div>

<!--<div class="example-container mat-elevation-z8" >-->
<mat-table [dataSource]="leagueDataSource" matSort class="mat-elevation-z8">
  <!-- Position Column -->
  <ng-container matColumnDef="rank">
    <mat-header-cell *matHeaderCellDef mat-sort-header>POSITION</mat-header-cell>
    <mat-cell *matCellDef="let element index as i;" [ngClass]="w-75"> {{ i + 1 }} </mat-cell>
  </ng-container>
  <!-- Football Club Name Column -->
  <ng-container matColumnDef="clubName">
    <mat-header-cell *matHeaderCellDef mat-sort-header>CLUB NAME</mat-header-cell>
    <mat-cell *matCellDef="let row" class="header-align-right"> {{row.clubName}} </mat-cell>
  </ng-container>

  <!-- Played Matches Column -->
  <ng-container matColumnDef="numberOfMatchesPlayed">
    <mat-header-cell *matHeaderCellDef mat-sort-header>PLAYED</mat-header-cell>
    <mat-cell *matCellDef="let row" id = "center-values"> {{row.numberOfWorkMatchesPlayed}} </mat-cell>
  </ng-container>

  <!-- Club Wins Column -->
  <ng-container matColumnDef="clubWins">
    <mat-header-cell *matHeaderCellDef mat-sort-header>WINS</mat-header-cell>
    <mat-cell *matCellDef="let row"> {{row.clubWins}} </mat-cell>
  </ng-container>

  <!-- Club Draws Column -->
  <ng-container matColumnDef="clubDraws">
    <mat-header-cell *matHeaderCellDef mat-sort-header>DRAWS</mat-header-cell>
    <mat-cell *matCellDef="let row"> {{row.clubDraws}} </mat-cell>
  </ng-container>

  <!-- Club Losses Column -->
  <ng-container matColumnDef="clubDefeats">
    <mat-header-cell *matHeaderCellDef mat-sort-header>DEFEATS</mat-header-cell>
    <mat-cell *matCellDef="let row"> {{row.clubDefeats}} </mat-cell>
  </ng-container>

  <!-- GS Column -->
  <ng-container matColumnDef="goalsScored">
    <mat-header-cell *matHeaderCellDef mat-sort-header>GS</mat-header-cell>
    <mat-cell *matCellDef="let row"> {{row.goalsScored}} </mat-cell>
```

```

</ng-container>

<!-- GR Column -->
<ng-container matColumnDef="goalsReceived">
  <mat-header-cell *matHeaderCellDef mat-sort-header>GR</mat-header-cell>
  <mat-cell *matCellDef="let row"> {{row.goalsReceived}} </mat-cell>
</ng-container>

<!-- GD Column -->
<ng-container matColumnDef="goalDifferenceClub">
  <mat-header-cell *matHeaderCellDef mat-sort-header>GD</mat-header-cell>
  <mat-cell *matCellDef="let row"> {{row.goalDifferenceClub}} </mat-cell>
</ng-container>

<!-- POINTS Column -->
<ng-container matColumnDef="points">
  <mat-header-cell *matHeaderCellDef mat-sort-header>POINTS</mat-header-cell>
  <mat-cell *matCellDef="let row"> {{row.points}} </mat-cell>
</ng-container>

<mat-header-row *matHeaderRowDef="leagueTableColumns"></mat-header-row>
<mat-row *matRowDef="let row; columns: leagueTableColumns;">
</mat-row>
</mat-table>
<mat-paginator [pageSizeOptions]="[5, 10, 20]" showFirstLastButtons></mat-paginator>

<!-- <mat-paginator [pageSizeOptions]="[5, 10, 25, 100]"></mat-paginator>-->
<!--</div>-->

```

### 3.2.2 league-table.component.ts

```

import {Component, OnInit, ViewChild} from '@angular/core';
import {MatTableDataSource} from '@angular/material/table';
import {HttpErrorResponse} from '@angular/common/http';
import {MatSort, Sort} from '@angular/material/sort';
import {appService} from '../app.service';
import {MatPaginator} from '@angular/material/paginator';

@Component({
  selector: 'app-league-table',
  templateUrl: './league-table.component.html',
  styleUrls: ['./league-table.component.scss']
})
export class LeagueTableComponent implements OnInit {
  @ViewChild(MatSort) leagueSort: MatSort;
  @ViewChild(MatPaginator) leaguePaginator: MatPaginator;

```

```
leagueTableColumns: string[] = ['rank', 'clubName', 'numberOfMatchesPlayed', 'clubWins', 'clubDraws',
'clubDefeats', 'goalsScored', 'goalsReceived', 'goalDifferenceClub', 'points'];
leagueDataSource = new MatTableDataSource([]);
```

```
constructor(private leagueService: appService) {}
//The following datasource will receive the league list from the backend
ngOnInit(): void {
  this.leagueService.getPremierClubs().subscribe(
    (data: any) => {
      this.leagueDataSource = new MatTableDataSource(data);
      this.leagueDataSource.paginator = this.leaguePaginator;
      setTimeout(() => this.leagueDataSource.sort = this.leagueSort);
      const sortState: Sort = {active: 'name', direction: 'desc'};
      this.leagueSort.active = sortState.active;
      this.leagueSort.direction = sortState.direction;
      this.leagueSort.sortChange.emit(sortState);
    },
    (err: HttpErrorResponse) => {
      console.log(err.message);
    });
  }
}
```

### 3.3 match component

#### 3.3.1 match.component.html

```
<div fxLayout>
  <div fxFlex="80%">
    <h2 style="color: white">PREMIER LEAGUE MATCHES</h2>
  </div>
  <div fxLayoutAlign="right center">
    <mat-form-field fxFlex="100%" floatLabel="never">
      <input matInput type="text" (keyup)="doFilter($event.target.value)" placeholder="Search Date">
    </mat-form-field>
  </div>
  <div id="btn-container" fxFlex="0%" fxLayoutAlign="end center">
    <a mat-raised-button color="primary" (click)="openDialog()">RANDOM MATCH</a>
  </div>
</div>

<mat-table [dataSource]="matchDataSource" matSort class="mat-elevation-z8">
  <!-- Football Club Name Column -->
  <ng-container matColumnDef="matchDate">
    <mat-header-cell *matHeaderCellDef mat-sort-header>DATE</mat-header-cell>
    <mat-cell *matCellDef="let row" class="header-align-right"> {{row.matchDate}} </mat-cell>
  </ng-container>
```

```

<!-- Played Matches Column -->
<ng-container matColumnDef="opponent01.clubName">
  <mat-header-cell *matHeaderCellDef mat-sort-header>OPPONENT01</mat-header-cell>
  <mat-cell *matCellDef="let FootballMatch" id = "center-values">
    {{FootballMatch.opponent01.clubName}} </mat-cell>
  </ng-container>

<!-- Club Wins Column -->
<ng-container matColumnDef="opponent02.clubName">
  <mat-header-cell *matHeaderCellDef mat-sort-header>OPPONENT02</mat-header-cell>
  <mat-cell *matCellDef="let FootballMatch"> {{FootballMatch.opponent02.clubName}} </mat-cell>
</ng-container>

<!-- Club Draws Column -->
<ng-container matColumnDef="matchScoreOpponent01">
  <mat-header-cell *matHeaderCellDef mat-sort-header>SCORE01</mat-header-cell>
  <mat-cell *matCellDef="let row"> {{row.matchScoreOpponent01}} </mat-cell>
</ng-container>

<!-- Club Losses Column -->
<ng-container matColumnDef="matchScoreOpponent02">
  <mat-header-cell *matHeaderCellDef mat-sort-header>SCORE02</mat-header-cell>
  <mat-cell *matCellDef="let row"> {{row.matchScoreOpponent02}} </mat-cell>
</ng-container>

<mat-header-row *matHeaderRowDef="matchTableColumns"></mat-header-row>
<mat-row *matRowDef="let row; columns: matchTableColumns;">
</mat-row>
</mat-table>
<mat-paginator [pageSizeOptions]="[10, 20, 380]" showFirstLastButtons></mat-paginator>

```

### 3.3.2 match.component.ts

```

import {Component, OnInit, ViewChild} from '@angular/core';
import {MatTableDataSource} from '@angular/material/table';
import {HttpErrorResponse} from '@angular/common/http';
import {MatSort, Sort} from '@angular/material/sort';
import {appService} from '../app.service';
import {MatDialog} from '@angular/material/dialog';
import {DialogMessageComponent} from '../dialog-message/dialog-message.component';
import {MatPaginator} from '@angular/material/paginator';
class NestedMatTableDataSource<T> extends MatTableDataSource<T> {

  constructor(initialData: T[] = []) {
    super(initialData);
  }
  // @ts-ignore
  sortingDataAccessor =
    (data: object, sortHeaderId: string): string | number => {
      const propPath = sortHeaderId.split('.');

```



```

        const value: any = propPath
        .reduce((curObj, property) => curObj[property], data);
        return isNaN(value) ? Number(value) : value;
    };
}

@Component({
    selector: 'app-match',
    templateUrl: './match.component.html',
    styleUrls: ['./match.component.scss']
})
export class MatchComponent implements OnInit {
    @ViewChild(MatSort) matchSort: MatSort;
    @ViewChild(MatPaginator) matchPaginator: MatPaginator;

    matchTableColumns: string[] = ['matchDate', 'opponent01.clubName', 'opponent02.clubName',
    'matchScoreOpponent01', 'matchScoreOpponent02'];
    matchDataSource = new NestedMatTableDataSource([]);

    constructor(private leagueService: appService, public dialog: MatDialog ) {}

    ngOnInit(): void {
        this.leagueService.getPremierMatches().subscribe(
            (data: any) => {
                this.matchDataSource = new NestedMatTableDataSource(data);
                this.matchDataSource.paginator = this.matchPaginator;

                setTimeout(() => this.matchDataSource.sort = this.matchSort);
                const sortState: Sort = {active: 'name', direction: 'desc'};
                this.matchSort.active = sortState.active;
                this.matchSort.direction = sortState.direction;
                this.matchSort.sortChange.emit(sortState);

            },
            (err: HttpErrorResponse) => {
                console.log(err.message);
            }
        );
    }

    public openDialog(): void {
        this.dialog.open(DialogMessageComponent)
            .afterClosed()
            .subscribe(() => this.refreshParent());
    }

    public doFilter = (value: string) => {
        this.matchDataSource.filter = value.trim().toLocaleUpperCase();
    }
}

```

```

    private refreshParent(): void {
        location.reload();
    }
}

```

## 3.4 dialog-message component

### 3.4.1 dialog-message.component.html

```

<div mat-dialog-title id="pop-up-msg">
  <h2 style="text-align: center">Random Match Details</h2>
</div>
<!--Sends random match result to the h2 tag-->
<h2 [innerHTML]="matchMsg"></h2>
<div mat-dialog-actions>
  <a mat-button color="primary" mat-dialog-close >Close</a>
</div>

```

### 3.4.2 dialog-message.component.ts

```


import { Component, OnInit } from '@angular/core';
import { appService } from '../app.service';

@Component({
  selector: 'app-dialog-message',
  templateUrl: './dialog-message.component.html',
  styleUrls: ['./dialog-message.component.css']
})
export class DialogMessageComponent implements OnInit {
  /*The following component is used to display a dialog box whenever a random match is called for
  from the random match button*/
  matchMsg: string;
  constructor(private msgService: appService) {}
  //The appService would send the backend result to matchMsg
  ngOnInit(): void {
    this.msgService.getRandomMatchMsg().subscribe((data: any) => {
      this.matchMsg = data;
      console.log(this.matchMsg);
    });
  }
}

```

## 3.5 Angular GUI Screenshots

### 3.5.1 Premier League Table

**PREMIER LEAGUE** [LEAGUE TABLE](#) [LEAGUE MATCHES](#)

**PREMIER LEAGUE TABLE**

POSITION	CLUB NAME	PLAYED	WINS	DRAWS	DEFEATS	GS	GR	GD	POINTS
1	CHELSEA	11	5	1	5	66	60	6	16
2	ASRSENAL	9	5	0	4	40	43	-3	15
3	BURNLEY	10	4	1	5	43	43	0	13
4	MANCHESTER	11	4	1	6	44	63	-19	13
5	BURTON F.C.	5	2	0	3	31	23	8	6

Items per page: **5** 1 – 5 of 6 |< < > >|

### 3.5.2 Match Table

**PREMIER LEAGUE MATCHES**  [RANDOM MATCH](#)

DATE	OPPONENT01	OPPONENT02	SCORE01	SCORE02
2020-01-19	BURNLEY	ASRSENAL	6	7
2020-01-28	MANCHESTER	CHELSEA	9	5
2020-03-14	CHELSEA	BURNLEY	8	9
2020-03-19	CHELSEA	MANCHESTER	1	3
2020-05-05	BURNLEY	CHELSEA	5	3
2020-05-22	ASRSENAL	MANCHESTER	5	8
2020-05-30	ASRSENAL	CHELSEA	3	9
2020-07-15	ASRSENAL	BURNLEY	3	2
2020-07-17	CHELSEA	EVERTON	9	7
2020-08-10	MANCHESTER	BURNLEY	7	7

Items per page: **10** 1 – 10 of 29 |< < > >|

### 3.5.3 Premier League Table

PREMIER LEAGUE MATCHES					Search Date	RANDOM MATCH
DATE	OPPONENT01	OPPONENT02	SCORE01	SCORE02		
2020-01-19	BURNLEY	ASRSNAL	6	7		
2020-01-28				5		
2020-03-14				9		
2020-03-19				3		
2020-05-05				3		
2020-05-22				8		
2020-05-30				9		
2020-07-15				2		
2020-07-17	CHELSEA	EVERTON	9	7		
2020-08-10	MANCHESTER	BURNLEY	7	7		
					Items per page: 10	1 - 10 of 29

## 4. Playframe backend Routes

# Routes

# This file defines all application routes (Higher priority routes first)

# ~~~~

# Serve index page from public directory

GET / controllers.LeagueController.premierLeagueSummary()

# An example route (Prefix all API routes with apiPrefix defined in application.conf)

GET /api/league controllers.LeagueController.listPremierLeague()

GET /api/matches controllers.LeagueController.listPremierMatches()

GET /api/randomMatch controllers.LeagueController.generateRandomMatch()

## 5. Junit Testing class

### 5.1 PremierLeagueTest.java

```
import org.junit.*;
```

```
import java.time.LocalDate;
```

/\*The following class was used for JUNIT testing which was tested on few of the premier league manager methods

\*\* to highlight that the expected result was produced from certain methods such adding a club, relegating a club

and adding a played match.

This test class was compiled and run through cmd with the following cmd prompts

JUnit compilation :->

```
javac -cp junit-4.12.jar;. PremierLeagueTest.java
```

JUnit run test code :->

```
java -cp junit-4.12.jar;hamcrest-core-1.3.jar;. org.junit.runner.JUnitCore PremierLeagueTest
```

Tutorial reference link : <https://www.codejava.net/testing/how-to-compile-and-run-junit-tests-in-command-line>

\*\*/

```
import static org.junit.Assert.*;
```

```
public class PremierLeagueTest {
```

```
    private PremierLeagueManager plm = new PremierLeagueManager();
```

```
    SportsClub Manchester = new FootballClub("Manchester","UK",20, 2000);
```

```
    SportsClub Chelsea = new FootballClub("Chelsea","UK",22, 2002);
```

```
    @Test
```

```
    public void testAddFootballClub() {
```

```
        plm.addSportsClub (Manchester);
```

```
        // Testing if the following club was added
```

```
        assertEquals("The " + Manchester.getClubName() + " club was successfully  
added!",Manchester.getClubName(), plm.retrieveFootballClub( "Manchester").getClubName());  
    }
```

```
    @Test
```

```
    public void testRelegateFootballClub() {
```

```
        plm.addSportsClub (Chelsea);
```

```
        plm.relegateSportsClub(Chelsea.getClubName());
```

```
        assertEquals(Chelsea.getClubName() + " has been successfully relegated!",null,  
plm.retrieveFootballClub( "Chelsea")); // expected club name  
    }
```

```
    @Test
```

```
    public void testAddPlayedMatch() {
```

```
        plm.addSportsClub(Chelsea);
```

```
        plm.addSportsClub(Manchester);
```

```
        Object foundMatch = null;
```

```
        FootballMatch footballMatch = new FootballMatch((FootballClub)Chelsea,  
(FootballClub)Manchester,10,12, LocalDate.of(2000,10,23));
```

```
        plm.addPlayedMatch(footballMatch, true);
```

```
        for(FootballMatch individualMatch: plm.returnMatches()){
```

```
            if(individualMatch.equals(footballMatch)){
```

```
                foundMatch = footballMatch;
```

```
                break;
```

```
            }
```

```
        }
```

```

        assertEquals("The following Football match has successfully taken place ",
footballMatch,foundMatch);
    }
}

```

## 5.2 PremierLeagueTest.java output

```

C:\Users\Gibran Kasif\Desktop\w1761211_CONSOLE_JAVA\src>javac -cp junit-4.12.jar;. PremierLeagueTest.java
C:\Users\Gibran Kasif\Desktop\w1761211_CONSOLE_JAVA\src>java -cp junit-4.12.jar;hamcrest-core-1.3.jar;. org.junit
JUnit version 4.12
Football club successfully added into the Premier League
Football club successfully added into the Premier League
The following football club is already in the Premier League
The following football club is already in the Premier League
Chelsea has been removed from the Premier League

Time: 0.018

OK (3 tests)

```

## 6. Testing

Test ID	Test Case	Input	Expected Output
1	Adding FootBallClub ("Manchester", "UK", 23, 2000)	Console Input: A or a Name: Manchester Location: UK Club Base: 23 Year: 2000	Console requests the following 4 inputs once completed; football club will be successfully added to the premier league
2	Relegating already added football club, ("Manchester").	Console Input: R or r Input : MANCHESTER, manchester, manchestER.	Console requests the club name, once entered Manchester will be successfully removed from the league.
3	Adding FootBallClub ("Chelsea", "UK", 21, 2010)  Find Chelsea Football Club stats & information.	Console Input: A or a Name: Chelsea Location: UK Club Base: 21 Year: 2010  Console Input F or f Name: Chelsea	Console requests for club name once after that Chelsea stats and information would be displayed.

4	Create a football match between Chelsea & newly added club United F.C.	Console Input: M or m Opponent 1 : Chelsea Opponent 2: United F.C. Score1: 12 Score2: 10 Day: 23 Month: 07	Once the following inputs have been entered, the football match would have successfully taken place.
5	View league table	Console input: T or t	Once console option is entered the League Table would successfully be displayed in cmd.
6	Start GUI	Console input: l or i	Both backend and front end would load up on the browser.
7	Save progress	Console input: S or s	Once S is entered all progress will be saved into the premierLeagueFile.txt
8	Input validations on console	Console input: @, 1, P, C,+ or *.	The console will keep on re-prompting with an incorrect input message until a correct option was entered.

## References

- <https://github.com/dilum1995/IIT-PlayFramework-Session>
- <https://sven-roettering.de/sorting-material-data-tables/>
- <https://stackoverflow.com/questions/15464111/run-cmd-commands-through-java>
- <https://material.angular.io/components/table/overview>
- <https://material.angular.io/components/dialog/overview>
- <https://www.codejava.net/testing/how-to-compile-and-run-junit-tests-in-command-line>