INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Department of computing

BSc Computer Science

Module: 5COSC007C OOP

Module Leader: Guhanathan Poravi

Coursework

Date of submission	1st January 2021
Student ID	2019163
Student UoW ID	W1761196
Student First Name	M. Ammar
Student Surname	Raneez
Student Group	C

Contents

01 – Design 2
1.1 Use Case Diagram 2
1.2 Class Diagram 4
02 – All Code 5
2.1 Controllers 5
2.2 Models 36
2.3 Services 99
2.4 Utils 119
2.5 Conf 124
2.6 UI 125
2.6.1 components 131
2.6.2 Models 207
2.6.3 Services 215
03 – Design Screenshots 223
04 – JUnit Testing 233

01 – Design

1.1 Use Case Diagram

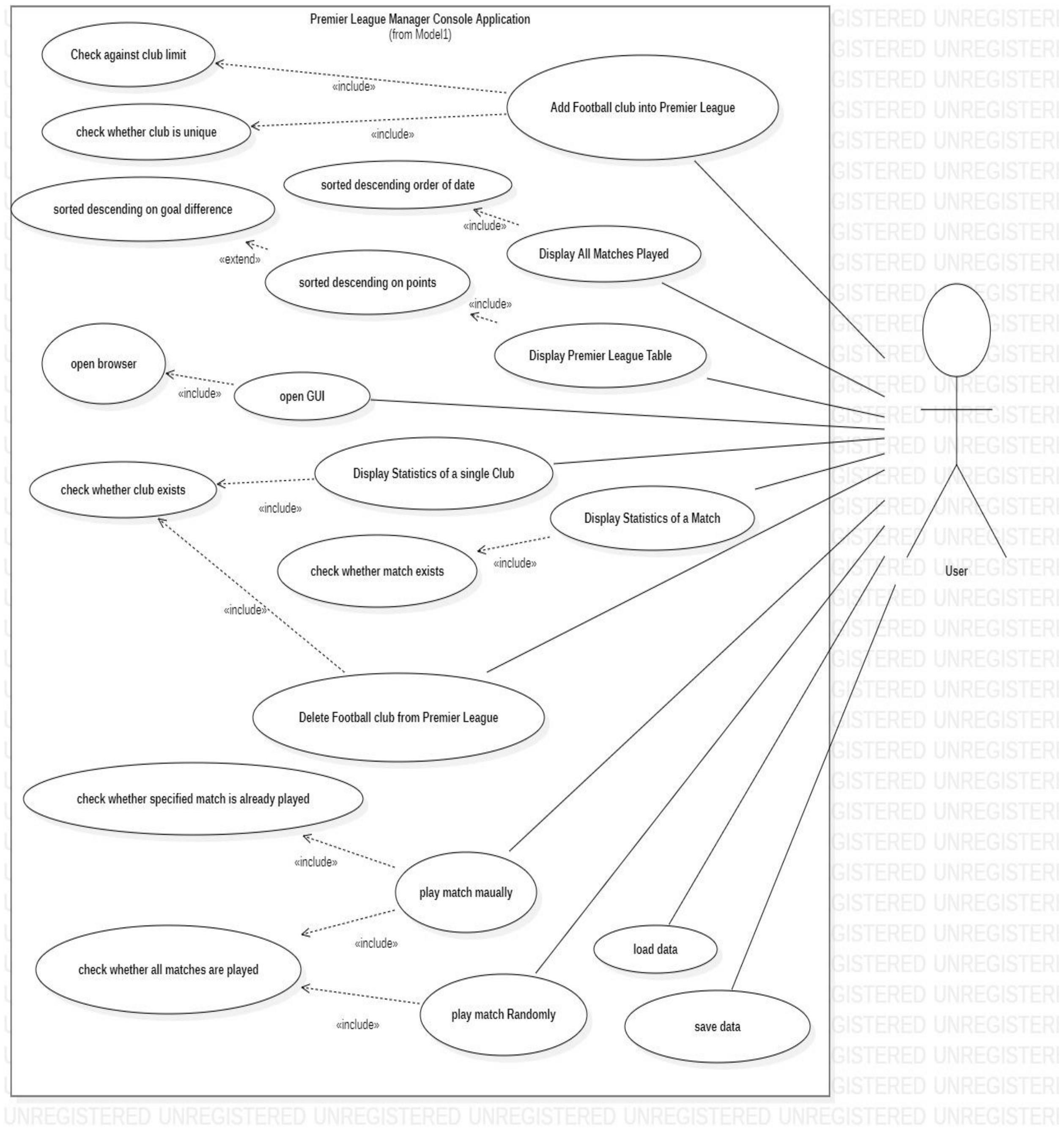

Premier League Manager Console Application
(from Model1)
Check against club limit
check whether club is unique
Add Football club into Premier League
«include»
«include»
sorted descending order of date
sorted descending on goal difference
«include»
Display All Matches Played
«extend»
sorted descending on points
«include»
Display Premier League Table
open browser
«include»
open GUI
Display Statistics of a single Club
check whether club exists
«include»
Display Statistics of a Match
check whether match exists
«include»
«include»
User
Delete Football club from Premier League
check whether specified match is already played
«include»
play match maually
«include»
check whether all matches are played
load data
«include»
play match Randomly
save data
UNREGISTERED

Figure 1 Use Case Diagram - CLI

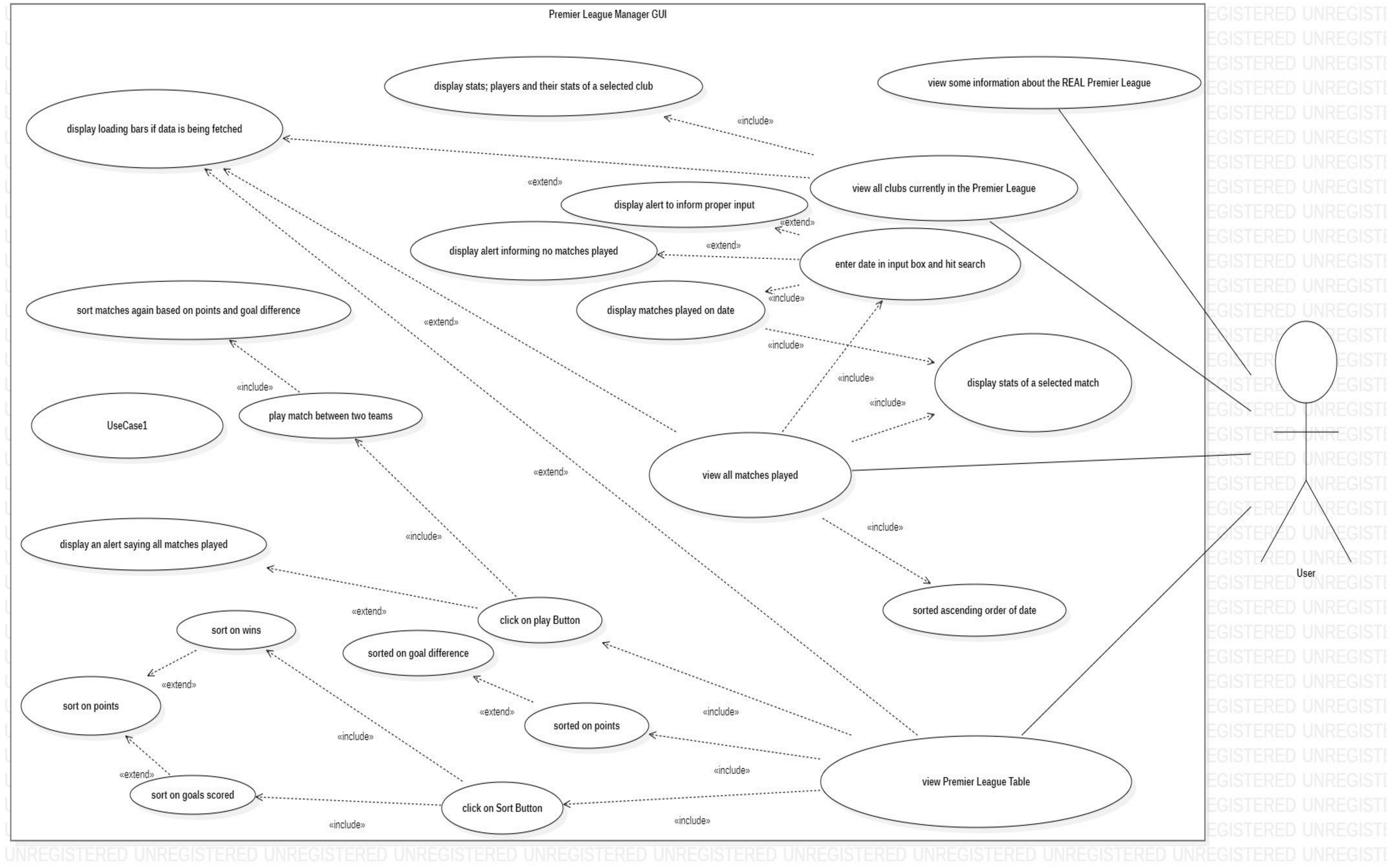

Premier League Manager GUI
display stats; players and their stats of a selected club
view some information about the REAL Premier League
display loading bars if data is being fetched
view all clubs currently in the Premier League
display alert to inform proper input
display alert informing no matches played
enter date in input box and hit search
sort matches again based on points and goal difference
display matches played on date
display stats of a selected match
UseCase1
play match between two teams
view all matches played
display an alert saying all matches played
sorted ascending order of date
click on play Button
sort on wins
sorted on goal difference
sort on points
sorted on points
view Premier League Table
sort on goals scored
click on Sort Button
«include»
«extend»
User
UNREGISTERED

Figure 2 Use Case Diagram – GUI

1.2 Class Diagram

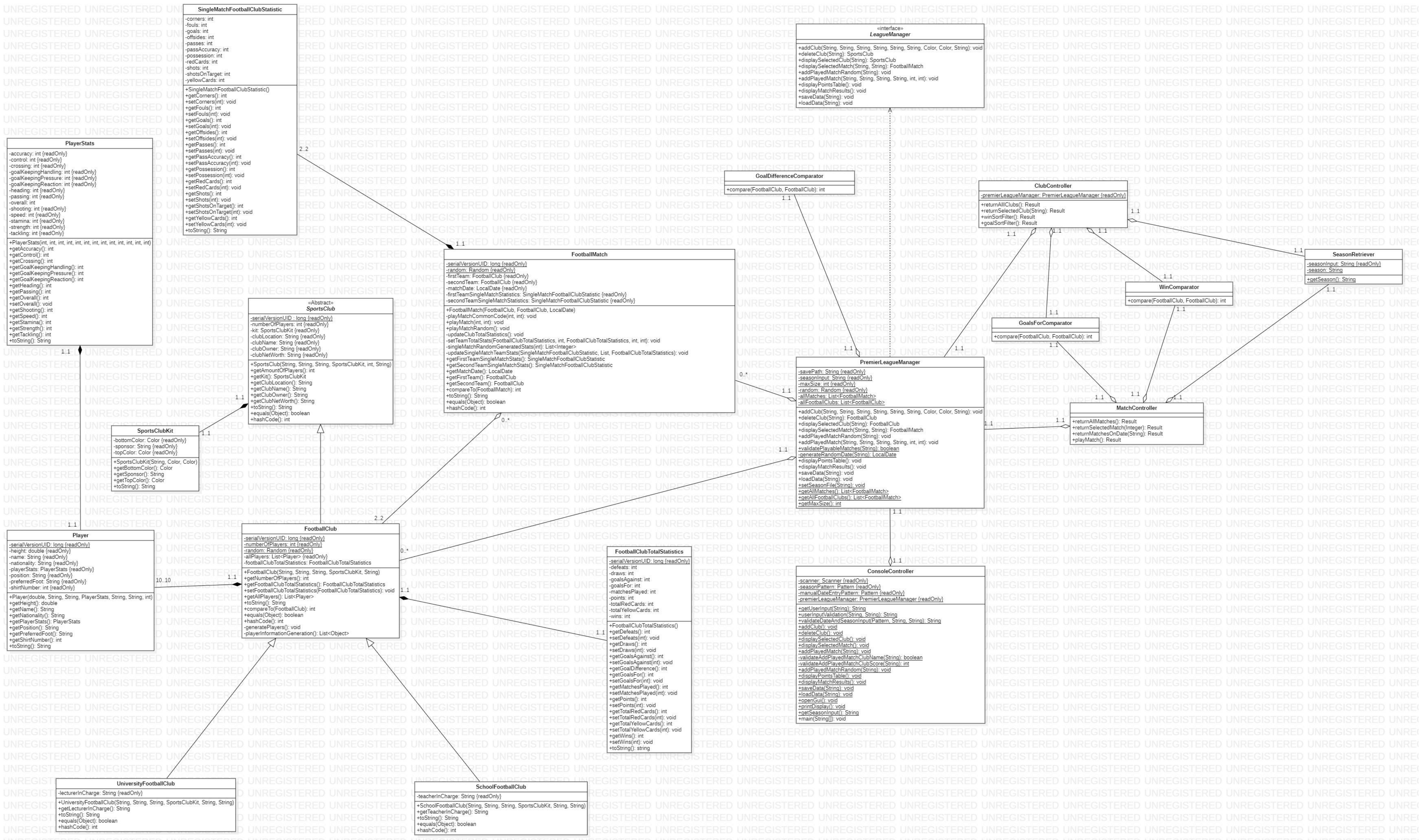


Figure 3 Class Diagram

02 – All Code

2.1 Controllers

ClubController

```
package coursework.controllers;


/*
 * ClubController
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import coursework.utils.GoalsForComparator;
import coursework.services.PremierLeagueManager;
import coursework.utils.SeasonRetriever;
import coursework.utils.WinComparator;
import coursework.models.FootballClub;
import play.mvc.*;
import play.libs.Json;
import java.util.*;


/**
 * ClubController class, will handle serving data related to Football clubs
 * @version 1.x December 10th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class ClubController extends Controller {
    private final PremierLeagueManager PREMIER_LEAGUE_MANAGER = new
PremierLeagueManager();
```

```
    /**
     * Serves all the football clubs in the Premier League
     * @return - Json format of all the football clubs
     */
    public Result returnAllClubs() {
        //*The common functionality in Play is*//
        //*Get season -> load data of that season -> fetch match/club of that season -> call cli
methods to handle*//
        //*functionality*//
        //*Fetches the season input from the file, parameters weren't being received by Play! From
the Console*//
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        List<FootballClub> allClubs = PremierLeagueManager.getAllFootballClubs();
        allClubs.sort(Collections.reverseOrder());
        ConsoleController.saveData(season);
        return ok(Json.toJson(allClubs));
    }


    /**
     * Serves a selected football club in the Premier League
     * @param clubName - parameter passed in the path URL, will be an index to obtain the
football club
     * at that appropriate index, through array-based indexing (Each club in at an index)
     * @return - Json format of the selected club
     */
    public Result returnSelectedClub(String clubName) {
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        //*whitespaces are passed through the url containing "%20"*//
        //*so if it was the case that a club name has two words, separated by a whitespace*//
        //*in order to obtain the actual name, the string has to be separated at the "%20"*//
        if (clubName.contains("%")) {
            String[] whitespaceParamSplit = clubName.split("%20");
            clubName = whitespaceParamSplit[0] + " " + whitespaceParamSplit[1];
        }
        FootballClub selectedClub =
PREMIER_LEAGUE_MANAGER.displaySelectedClub(clubName);
        ConsoleController.saveData(season);
        return ok(Json.toJson(selectedClub));
    }


    /**
     * Serves all the clubs sorted by wins in descending order
     * @return - Json format of the sorted clubs
     */
    public Result winSortClubFilter() {
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        List<FootballClub> allClubs = PremierLeagueManager.getAllFootballClubs();
        //*The reason to sort based on points too, is because, if there are teams that have the same
number of wins*//
        //*The points obtained by the teams will be considered to give the higher position*//
        allClubs.sort(Collections.reverseOrder());
        allClubs.sort(new WinComparator().reversed());
        ConsoleController.saveData(season);
        return ok(Json.toJson(allClubs));
    }
    /**
     * Serves all the clubs sorted by goalsFor in descending order
     * @return - Json format of the sorted clubs
     */
    public Result goalSortClubFilter() {
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        List<FootballClub> allClubs = PremierLeagueManager.getAllFootballClubs();
        //*The reason to sort based on points too, is because, if there are teams that have the same
number of goals*//
        //*The points obtained by the teams will be considered to give the higher position*//
        allClubs.sort(Collections.reverseOrder());
        allClubs.sort(new GoalsForComparator().reversed());
        ConsoleController.saveData(season);
        return ok(Json.toJson(allClubs));
    }
}
```

ConsoleController

```
package coursework.controllers;


/*
 * ConsoleController
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */



import coursework.models.FootballClub;
import coursework.models.FootballMatch;
import coursework.services.PremierLeagueManager;

import java.awt.*;
import java.io.IOException;
import java.lang.reflect.Field;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Scanner;
import java.util.regex.Pattern;

/**
 * ConsoleController class, the main cli runner class
 * @version 1.x November 15th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class ConsoleController {
    private static final Scanner SCANNER = new Scanner(System.in);
    //*regex for season input validation*//
    //*any 4 digit number*//
    private static final Pattern SEASON_PATTERN = Pattern.compile("^\\d{4}");
    //*validation to ensure month is 1-12, and date 1-31*//
    private static final Pattern MANUAL_DATE_ENTRY_PATTERN = Pattern.compile("(0[1-
9]|1[0-9]|2[0-9]|3[0-1])-(0[1-9]|1[0-2])");
    private static final PremierLeagueManager PREMIER_LEAGUE_MANAGER = new
PremierLeagueManager();


    //**********************************************HELPER
METHODS****************************************************//
    /**
     * Public helper method that displays a sentence and returns user's input transformed to
lowercase and whitespaces
     * trimmed
     * Marked static since it's common for any object created
     * @param printSentence - sentence to use as a prompt
     * @return - the input of the user
     */
    public static String getUserInput(String printSentence) {
        System.out.println(printSentence);
        return SCANNER.nextLine().toLowerCase().trim();
    }


    /**
     * public helper method that is used for the common input string validation
     * @param printMessage - sentence to use as an initial prompt
     * @param retryMessage - sentence to use as the retry prompt (even after the initial one failed)
     * @return - valid user input
     */
    public static String userInputValidation(String printMessage, String retryMessage) {
        String userInput = getUserInput(printMessage);
        while (true) {
            if (userInput.equals("")) {
                userInput = getUserInput(retryMessage);
            } else {
                break;
            }
        }
        return userInput;
    }

    /**
     * public helper method that is used for the date or season input validation
     * @param regex - the Pattern regex to use
     * @param inputMsg - sentence to use as an initial prompt
     * @param validationMsg  sentence to use as the retry prompt
     * @return a valid inputted date or season
     */
    public static String validateDateAndSeasonInput(Pattern regex, String inputMsg, String
validationMsg) {
        String dateOrSeason = ConsoleController.userInputValidation(inputMsg, validationMsg);
        while (true) {
            if (regex.matcher(dateOrSeason).matches()) {
                break;
            } else {
                dateOrSeason = getUserInput(validationMsg);
            }
        }
        return dateOrSeason;
    }
    //*********************************************END HELPER
METHODS************************************************//




    /**
     * static method, that handles the adding of a club
     * Calls the addClub() method of PremierLeagueManager, passing the inputs obtained as
parameters
     * @throws ClassNotFoundException - thrown in Color input
     * @throws IllegalAccessException - thrown in the get() method of Field
     */
    public static void addClub() throws ClassNotFoundException, IllegalAccessException {
        //*only 20 teams can be added in the Premier League*//
        if (PremierLeagueManager.getAllMatches().size() ==
PremierLeagueManager.getMaxSize()) {
            System.out.println("[ERROR] ==> There cannot be more than 20 teams in the premier
league!");
                //*prevent further execution of program*//
            return;
        }

        //*validate club type input ==> allow only 3 values, loop till a valid input has been given*//
        //*loop endlessly, until a proper input has been given(non empty empty)*//
        String clubTypeInput = getUserInput("Please enter the type of club " +
                "(University /School /" + "League level)");
        while (true) {
            if (clubTypeInput.equals("university") || clubTypeInput.equals("school") ||
clubTypeInput.equals("league")) {
                break;
            } else {
                clubTypeInput = getUserInput("Please specify appropriately! " +
                        "(university/school/league)");
            }
        }
        //*based on type of club input, club type based inputs are taken for school and university*//
        String lecOrTeachInput = null;
        if (clubTypeInput.equals("university") || clubTypeInput.equals("school")) {
            String formattedPrint = clubTypeInput.equals("university") ? "Please enter the lecturer in
charge" :
                    "Please enter the teacher in charge";
            lecOrTeachInput = getUserInput(formattedPrint);
            while (true) {
                if (lecOrTeachInput.equals("")) {
                    lecOrTeachInput = getUserInput(formattedPrint + "!");
                } else {
                    break;
                }
            }
        }


        //*club name validation - validate all inputs the same way*//
        String clubNameInput = userInputValidation("Enter Club name", "Please Enter a Club
name!");
        //*allow only unique club names (unique clubs)*//
        boolean clubExists = false;
        while (true) {
            for (FootballClub footballClub : PremierLeagueManager.getAllFootballClubs()) {
                if (footballClub.getClubName().equals(clubNameInput)) {
                    clubExists = true;
                    //*club already exists*//
                    clubNameInput = userInputValidation("[ERROR] ==> " + clubNameInput + "
already exists! Please try again",
                            "Please Enter a Club name!");
                    //*stop continuous looping, after a club has been found, to prevent unnecessary
continuous looping*//
                    break;
                }
            }
            if (!clubExists) {
                break;
            } else {
                clubExists = false;
            }
        }


        //*club location validation*//
        String clubLocationInput = userInputValidation("Enter club location", "Please Enter a Club
location!");
        //*club owner validation*//
        String clubOwnerInput = userInputValidation("Enter club owner", "Please Enter a Club
owner!");
        //*club sponsor validation*//
        String clubSponsorInput = userInputValidation("Enter club sponsor", "Please Enter a Club
sponsor!");
```

```
//*Color and Field variables necessary for sports kit*//
Color colorTop;
Color colorShort;
Field fieldTop;
Field fieldShort;
//*color top validation ==> only specific colors are allowed*//
while (true) {
   try {
      String topColorInput = getUserInput("Enter club kit top color");
      fieldTop = Class.forName("java.awt.Color").getField(topColorInput);
      colorTop = (Color) fieldTop.get(null);
      break;
   } catch (NoSuchFieldException e) {
      System.out.println("[ERROR] ==> Please choose one of the valid colors! [black, blue,
cyan, gray, green, "
            + "magenta, orange, pink, red, white, yellow]");
   }
}
//*color short validation ==> allow only specific colors*//
while (true) {
   try {
      String shortColorInput = getUserInput("Enter club kit short color");
      fieldShort = Class.forName("java.awt.Color").getField(shortColorInput);
      colorShort = (Color) fieldShort.get(null);
      break;
   } catch (NoSuchFieldException e) {
      System.out.println("[ERROR] ==> Please choose one of the valid colors! [black, blue,
cyan, gray, green, "
            + "magenta, orange, pink, red, white, yellow]");
            }
        }


        //*club net worth validation*//
        String clubNetWorth = userInputValidation("Enter club net worth", "Please Enter the Clubs
net worth!");


        //*call add method passing all the inputs as arguments*//
        PREMIER_LEAGUE_MANAGER.addClub(clubTypeInput, lecOrTeachInput,
clubNameInput, clubLocationInput, clubOwnerInput,
                                  clubSponsorInput, colorTop, colorShort, clubNetWorth);
        System.out.println("Now adding Football Club " + clubNameInput);
        System.out.println(clubNameInput + " was successfully Promoted to the Premier League!");
    }


    /**
     * static method, that handles the deletion of a club
     * Calls the deleteClub() method of PremierLeagueManager, passing the club name obtained
as the parameter
-    */
    public static void deleteClub() {
        String clubNameInput = userInputValidation("Enter Club Name you wish to delete",
"Please Enter a Club name!");
        FootballClub deletedClub =
PREMIER_LEAGUE_MANAGER.deleteClub(clubNameInput);


        if (deletedClub != null) {
            System.out.println("Now deleting " + clubNameInput);
            System.out.println("Size decreased: " +
PremierLeagueManager.getAllFootballClubs().size());
```

```
            System.out.println(clubNameInput + " was successfully Relegated from the Premier League!");
        } else {
            System.out.println("[ERROR] ==> No Such Club Exists");
        }
    }

    /**
     * static method, that handles the displaying of a selected club
     * Calls the displaySelectedClub() method of PremierLeagueManager, passing the club name obtained as a parameter
     * Obtains the club and prints the details
     */
    public static void displaySelectedClub() {
        String clubNameInput = userInputValidation("Enter club name to display", "Please Enter a Club name!");
        FootballClub foundClub = PREMIER_LEAGUE_MANAGER.displaySelectedClub(clubNameInput);

        if (foundClub != null) {
            System.out.println(
                    "Club: " + foundClub.getClubName() + " | Location: " + foundClub.getClubLocation() + " | Owner: " +
                            foundClub.getClubOwner() + " | Net Worth: " + foundClub.getClubNetWorth() + " | Points: " +
                            foundClub.getFootballClubTotalStatistics().getPoints() + " | Matches Played: " +
                            foundClub.getFootballClubTotalStatistics().getMatchesPlayed() + " | Wins: " +
                            foundClub.getFootballClubTotalStatistics().getWins() + " | Draws: " +
                            foundClub.getFootballClubTotalStatistics().getDraws() + " | Defeats: " +
                            foundClub.getFootballClubTotalStatistics().getDefeats()
            );
        } else {
            System.out.println("[ERROR] ==> No Such Club Exists");
        }
    }


    /**
     * static method, that handles the displaying of statistics of a selected match
     * Calls the displaySelectedMatchStatistics() method of PremierLeagueManager passing the
two clubs involved in a match as its parameters
     * obtains the match and prints the statistics
     */
    public static void displaySelectedMatch() {
        String firstTeamInput = userInputValidation("Enter First Club's Name:", "Please Enter the
first Clubs name!");
        String secondTeamInput = userInputValidation("Enter Second Club's Name:", "Please Enter
the second Clubs name!");
        FootballMatch foundFootballMatch =
PREMIER_LEAGUE_MANAGER.displaySelectedMatch(firstTeamInput, secondTeamInput);


        if (foundFootballMatch != null) {
            //*display all stats of the match, in a simple formatted way*//

System.out.println("=====================================================
============");
            System.out.format("%-27s %-10s %28s",
foundFootballMatch.getFirstTeam().getClubName().toUpperCase(),
                    "TEAM STATS",
foundFootballMatch.getSecondTeam().getClubName().toUpperCase());
            System.out.println();
            System.out.format("%-30s %-5s %30s",
foundFootballMatch.getFirstTeamSingleMatchStats().getGoals(),
                    "Goals", foundFootballMatch.getSecondTeamSingleMatchStats().getGoals());
            System.out.println();
            System.out.format("%-30s %-5s %30s",
foundFootballMatch.getFirstTeamSingleMatchStats().getShots(),
                    "Shots", foundFootballMatch.getSecondTeamSingleMatchStats().getShots());
            System.out.println();
            System.out.format("%-25s %-15s %25s",
foundFootballMatch.getFirstTeamSingleMatchStats().getShotsOnTarget(),
                    "Shots on target",
foundFootballMatch.getSecondTeamSingleMatchStats().getShotsOnTarget());
            System.out.println();
            System.out.format("%-27s %-10s %28s",
foundFootballMatch.getFirstTeamSingleMatchStats().getPossession(),
                    "Possession",
foundFootballMatch.getSecondTeamSingleMatchStats().getPossession());
            System.out.println();
            System.out.format("%-29s %-6s %30s",
foundFootballMatch.getFirstTeamSingleMatchStats().getPasses(),
                    "Passes", foundFootballMatch.getSecondTeamSingleMatchStats().getPasses());
            System.out.println();
            System.out.format("%-26s %-13s %26s",
foundFootballMatch.getFirstTeamSingleMatchStats().getPassAccuracy(),
                    "Pass accuracy",
foundFootballMatch.getSecondTeamSingleMatchStats().getPassAccuracy());
            System.out.println();
            System.out.format("%-30s %-5s %30s",
foundFootballMatch.getFirstTeamSingleMatchStats().getFouls(),
                    "Fouls", foundFootballMatch.getSecondTeamSingleMatchStats().getFouls());
            System.out.println();
```

```
            System.out.format("%-26s %-12s %27s",
foundFootballMatch.getFirstTeamSingleMatchStats().getYellowCards(),
                    "Yellow cards",
foundFootballMatch.getSecondTeamSingleMatchStats().getYellowCards());
            System.out.println();
            System.out.format("%-28s %-9s %28s",
foundFootballMatch.getFirstTeamSingleMatchStats().getRedCards(),
                    "Red cards",
foundFootballMatch.getSecondTeamSingleMatchStats().getRedCards());
            System.out.println();
            System.out.format("%-28s %-8s %29s",
foundFootballMatch.getFirstTeamSingleMatchStats().getOffsides(),
                    "Offsides", foundFootballMatch.getSecondTeamSingleMatchStats().getOffsides());
            System.out.println();
            System.out.format("%-29s %-7s %29s",
foundFootballMatch.getFirstTeamSingleMatchStats().getCorners(),
                    "Corners", foundFootballMatch.getSecondTeamSingleMatchStats().getCorners());

System.out.println("\n===========================================================
=============");
        } else {
            System.out.println("[ERROR] ==> No such Football Match exists!");
        }
    }


    /**
     * static method, that handles the playing of a manually entered match
     * @param season - user season input
     */
    public static void addPlayedMatch(String season) {
        //*this if condition checks whether there are enough teams to play a match in the first
place*//
        if (PremierLeagueManager.getAllFootballClubs().size() < 2) {
            System.out.println("[ERROR] ==> There isn't enough teams to play a match!");
            return;
        }

        boolean allMatchesPlayed = PremierLeagueManager.validatePlayableMatches(season);
        //*if all matches are already played, then the method can stop immediately*//
        if (allMatchesPlayed) {
            System.out.println("[ERROR] ==> All Possible Matches have already been played!");
            return;
        }

        //*validate date against regex*//
        String date =
ConsoleController.validateDateAndSeasonInput(MANUAL_DATE_ENTRY_PATTERN,
"Please enter a date (dd-mm)",
                "Invalid input! Please specify a date! (dd-mm)");

        //*get validated inputs associated with the clubs and their score for the match, loop
continuously till a valid club name is entered*//
        String firstTeam; /*= ConsoleController.userInputValidation("Please enter the first clubs
name", "Please specify a club name!");*/
        boolean validFirstTeam; /* = validateAddPlayedMatchClubName(firstTeam);*/
//        if (!validFirstTeam) {
//            return;
//        }
        do {
            firstTeam = ConsoleController.userInputValidation("Please enter the first clubs name", "Please specify a club name!");
            validFirstTeam = validateAddPlayedMatchClubName(firstTeam);
        } while (!validFirstTeam);


        String firstTeamScore = ConsoleController.userInputValidation("Please enter the first clubs score", "Please specify a score!");
        int validatedFirstTeamScore = validateAddPlayedMatchClubScore(firstTeamScore);


        String secondTeam; /*= ConsoleController.userInputValidation("Please enter the second clubs name", "Please specify a club name!");*/
        boolean validSecondTeam; /*= validateAddPlayedMatchClubName(secondTeam);*/
//        if (!validSecondTeam) {
//            return;
//        }
        do {
            secondTeam = ConsoleController.userInputValidation("Please enter the second clubs name", "Please specify a club name!");
            validSecondTeam = validateAddPlayedMatchClubName(secondTeam);
        } while (!validSecondTeam);


        //*the same club can't play against themselves*//
        if (firstTeam.equals(secondTeam)) {
            System.out.println("[ERROR] ==> the same club cannot play against itself!");
            return;
        }
        String secondTeamScore = ConsoleController.userInputValidation("Please enter the second clubs score", "Please specify a score!");
        int validatedSecondTeamScore = validateAddPlayedMatchClubScore(secondTeamScore);
        //*if the program were to reach this statement, all inputs are valid, and hence a match is
playable*//
        PREMIER_LEAGUE_MANAGER.addPlayedMatch(season, date, firstTeam, secondTeam,
validatedFirstTeamScore, validatedSecondTeamScore);
    }
    /**
     * private helper method of addPlayedMatch()
     * To validate the club name input
     * @param clubName - input club name
     * @return t/f whether club entered is actually a club in the Premier League
     */
    private static boolean validateAddPlayedMatchClubName(String clubName) {
        boolean validFirstTeam = false;
        //*check whether the list of clubs contain the specified club or not, if not, the method is
terminated*//
        for (FootballClub club : PremierLeagueManager.getAllFootballClubs()) {
            if (club.getClubName().equals(clubName)) {
                validFirstTeam = true;
                break;
            }
        }

        if (!validFirstTeam) {
            System.out.println("[ERROR] ==> club specified does not exist!");
        }
        return validFirstTeam;
    }
    /**
     *  private helper method of addPlayedMatch()
     *  To validate club score input
     * @param score - input club score
     * @return the score parsed into an integer
     */
    private static int validateAddPlayedMatchClubScore(String score) {
        int parsedScore;
        while (true) {
            //*Try to convert the input to integer, if errored, ask score input again*//
            try {
                parsedScore = Integer.parseInt(score);
                //*only integers greater than 0 is accepted*//
                while(parsedScore < 0) {
                    parsedScore = Integer.parseInt(getUserInput("Goal cannot be less than 0!"));
                }
                break;
            } catch (Exception e) {
                score = getUserInput("Please specify an Integer as the score!");
            }
        }
        return parsedScore;
    }


    /**
     * static method, that handles the playing of a randomly generated match
     * Calls the addPlayedMatchRandom() method of PremierLeagueManager
     * @param season - user season input
     */
    public static void addPlayedMatchRandom(String season) {
PREMIER_LEAGUE_MANAGER.addPlayedMatchRandom(season); }
```

```
    /**
     * static method, that handles the displaying of the points table
     * Calls the displayPointsTable() method of PremierLeagueManager
     */
    public static void displayPointsTable() {
PREMIER_LEAGUE_MANAGER.displayPointsTable(); }


    /**
     * static method, that handles the displaying of all match results
     * Calls the displayMatchResults() method of PremierLeagueManager
     */
    public static void displayMatchResults() {
PREMIER_LEAGUE_MANAGER.displayMatchResults(); }


    /**
     * static method, that handles the saving of data
     * Calls the saveData() method of PremierLeagueManager
     * @param season - user season input
     */
    public static void saveData(String season) {
PREMIER_LEAGUE_MANAGER.saveData(season); }


    /**
     * static method, that handles the loading of data
     * Calls the loadData() method of PremierLeagueManager
     * @param season - user season input
     */
    public static void loadData(String season) {
PREMIER_LEAGUE_MANAGER.loadData(season); }
```

```
/**
 * static method, that handles opening of the localhost of angular for the GUI
 */
public static void openGui() {
    Desktop desktop = Desktop.getDesktop();

    try {
        desktop.browse(new URI("http://localhost:4200/"));
    } catch (URISyntaxException e) {
        System.out.println("URI was incorrect!");
    } catch (IOException e) {
        System.out.println("Something went wrong!");
    }
}

/**
 * Displays the Menu on the console
 */
public static void printDisplay() {
    System.out.println("**************************************************");
    System.out.println("WELCOME TO THE PREMIER LEAGUE");
    System.out.println("**************************************************");
    System.out.println("Enter a to promote a club to the Premier League");
    System.out.println("Enter d to relegate a club from the Premier League");
    System.out.println("Enter p to play a match (Random Generation)");
    System.out.println("Enter l to play a match (Manual Entry)");
    System.out.println("Enter z to display the current Premier League Standings");
```

```
        System.out.println("Enter c to display all Match Scores of the Premier League");
        System.out.println("Enter x to display a selected club");
        System.out.println("Enter s to display a selected match statistic");
        System.out.println("Enter g to display GUI");
        System.out.println("Enter q to exit");
    }

    /**
     * Gets and returns the user input of the season year
     * Also calls the method that will write the season input to a file (Needed for Play to get
access)
     * @return - the season year
     */
    public static String getSeasonInput() {
        String userSeasonChoice = validateDateAndSeasonInput(SEASON_PATTERN, "Please
enter the season (Ex 2020 or 2021)",
                "Please enter the season (Ex: 2020 OR 2021) appropriately!");
        PremierLeagueManager.setSeasonFile(userSeasonChoice);
        return userSeasonChoice;
    }

    /**
     * Main method, brain of the program, keep running the program, displaying the menu after
each method, for a continuous endless loop
     * Only end if explicitly commanded to, upon prompted for a choice on the menu
     * @param args - command line arguments
     * @throws IllegalAccessException - thrown in get() method of Field
     * @throws ClassNotFoundException - thrown in Color input
     */
    public static void main(String[] args) throws IllegalAccessException,
ClassNotFoundException {
        String season = getSeasonInput();
        printDisplay();
        String userChoice = userInputValidation("Please choose an option", "Please choose an
option!");
        infiniteLoop:
        while (true) {
            switch (userChoice) {
                case "a":
                    //*save and load before and after each method respectively. So that the content is
always updated*//
                    loadData(season);
                    addClub();
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "d":
                    loadData(season);
                    deleteClub();
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "p":
                    loadData(season);
                    addPlayedMatchRandom(season);
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "l":
                    loadData(season);
                    addPlayedMatch(season);
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "z":
                    loadData(season);
                    displayPointsTable();
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "c":
                    loadData(season);
                    displayMatchResults();
                    saveData(season);
                    printDisplay();
                    userChoice = getUserInput("Please choose an option");
                    break;
                case "x":
                    loadData(season);
                    displaySelectedClub();
                    saveData(season);
```

```
                printDisplay();
                userChoice = getUserInput("Please choose an option");
                break;
            case "s":
                loadData(season);
                displaySelectedMatch();
                saveData(season);
                printDisplay();
                userChoice = getUserInput("Please choose an option");
                break;
            case "g":
                openGui();
                printDisplay();
                userChoice = getUserInput("Please choose an option");
                break;
            case "q":
                saveData(season);
                break infiniteLoop;
            default:
                printDisplay();
                userChoice = getUserInput("Please choose a valid option");
        }
      }
   }
}
```

FrontendController -> Scala file

```
package coursework.controllers


import controllers.Assets
import javax.inject._
import play.api.Configuration
import play.api.http.HttpErrorHandler
import play.api.mvc._


/**
 * Frontend controller managing all static resource associate routes.
 * @param assets Assets controller reference.
 * @param cc Controller components reference.
 */
@Singleton
class FrontendController @Inject()(assets: Assets, errorHandler: HttpErrorHandler, config:
Configuration,
                                   cc: ControllerComponents) extends AbstractController(cc) {


  def index: Action[AnyContent] = assets.at("index.html")


  def assetOrDefault(resource: String): Action[AnyContent] = if
(resource.startsWith(config.get[String]("apiPrefix"))){
    Action.async(r => errorHandler.onClientError(r, NOT_FOUND, "Not found"))
  } else {
    if (resource.contains(".")) assets.at(resource) else index
  }
}
```

MatchController

```
package coursework.controllers;


/*
 * MatchController
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import coursework.services.PremierLeagueManager;
import coursework.models.FootballClub;
import coursework.models.FootballMatch;
import coursework.utils.SeasonRetriever;
import play.mvc.*;
import play.libs.Json;


import java.util.*;


/**
 * MatchController, will handle serving related to football matches
 * @version 1.x December 10th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class MatchController extends Controller {
    PremierLeagueManager premierLeagueManager = new PremierLeagueManager();
    /**
     * Serves all the football matches played in the Premier League
     * The same logic is used for all methods -> Matches obtained, any filters are applied, then
sorted
     * Then
     * @return - Json format of all the matches
     */
    public Result returnAllMatches() {
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        List<FootballMatch> allMatches = PremierLeagueManager.getAllMatches();
        Collections.sort(allMatches);
        ConsoleController.saveData(season);
        return ok(Json.toJson(allMatches));
    }

    /**
     * Serves the selected football match
     * @param arrIndex - parameter passed in the path URL, will be an index to obtain the
football match
     * at that appropriate index, through array-based indexing (Each match in at an index)
     * @return - Json format of the selected football match
     */
    public Result returnSelectedMatch(Integer arrIndex) {
        String season = SeasonRetriever.getSeason();
        ConsoleController.loadData(season);
        FootballMatch selectedMatch = PremierLeagueManager.getAllMatches().get(arrIndex);
        ConsoleController.saveData(season);
        return ok(Json.toJson(selectedMatch));
    }

    /**
     * Serves all matches of at a specific date
     * @param obtainedDate - date parameter passed in URL
```

```
 * @return - Json format of the filtered matches
 */
public Result returnMatchesOnDate(String obtainedDate) {
    String season = SeasonRetriever.getSeason();
    ConsoleController.loadData(season);
    List<FootballMatch> allMatches = PremierLeagueManager.getAllMatches();
    List<FootballMatch> filteredMatches = new ArrayList<>();

    //*get all the matches and then filter based on date*//
    for (FootballMatch footballMatch : allMatches) {
        if (String.valueOf(footballMatch.getMatchDate()).equals(obtainedDate)) {
            filteredMatches.add(footballMatch);
        }
    }

    ConsoleController.saveData(season);
    return ok(Json.toJson(filteredMatches));
}

/**
 * Method that handles playing of a match (clicking on play match button triggers this)
 * @return - Json format of the updatedClubs (The clubs are sent here and not the matches cuz
 * the play match button is in the leaderboard tab, therefore the clubs are what must be
 * updated on the spot)
 */
public Result playMatch() {
    String season = SeasonRetriever.getSeason();
    ConsoleController.loadData(season);
```

```
        boolean allMatchesPlayed = PremierLeagueManager.validatePlayableMatches(season);


        //*Return false if all matches have been played*//
        if (allMatchesPlayed) {
            return ok(Json.toJson(false));
        }


        FootballMatch playedMatch = premierLeagueManager.addPlayedMatchRandom(season);
        List<FootballClub> updatedClubs = PremierLeagueManager.getAllFootballClubs();
        updatedClubs.sort(Collections.reverseOrder());
        ConsoleController.saveData(season);
        List<Object> matchAndClubs = new ArrayList<>(Arrays.asList(updatedClubs,
playedMatch));
        return ok(Json.toJson(matchAndClubs));
    }
}
```

2.2 Models

FootballClub

```
package coursework.models;


/*
 * FootballClub
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.util.*;


/**
 * FootballClub class, which will be used to represent any football club (sub class of SportsClub)
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class FootballClub extends SportsClub implements Comparable<FootballClub> {
   private static final long serialVersionUID = 2272608864290797607L;
   private static final int NUMBER_OF_PLAYERS = 11;
   private static final Random RANDOM = new Random();
   private final List<Player> ALL_PLAYERS =  new ArrayList<>();
   private FootballClubTotalStatistics footballClubTotalStatistics;


   /**
    * Constructor - takes in values and initializes a Football club object
    * (common attributes initialized through super class)
    * footballClubTotalStatistics - another helper class that contains all club statistics
    * singleMatchFootballClubStatistic - another helper class that will holds statistics of a club of a single match
```

```
     * @param clubName - name of club
     * @param clubLocation - location of club
     * @param clubOwner - owner of club
     * @param kit - club kit (a helper class)
     * @param clubWorth - club net worth
\     */
    public FootballClub(String clubName, String clubLocation, String clubOwner, SportsClubKit
kit,
                        String clubWorth) {
        super(clubName, clubLocation, clubOwner, kit, NUMBER_OF_PLAYERS, clubWorth);
        this.footballClubTotalStatistics = new FootballClubTotalStatistics();
        this.generatePlayers();
    }

    /**
     * @return number of players in a football team
     */
    public static int getNumberOfPlayers() {
        return NUMBER_OF_PLAYERS;
    }

    /**
     * @return this clubs total statistic
     */
    public FootballClubTotalStatistics getFootballClubTotalStatistics() {
        return footballClubTotalStatistics;
    }

    /**
```

```
     * sets the total statistics for this club
     * @param footballClubTotalStatistics - this clubs total statistics
     */
    public void setFootballClubTotalStatistics(FootballClubTotalStatistics
footballClubTotalStatistics) {
        this.footballClubTotalStatistics = footballClubTotalStatistics;
    }

    /**
     * @return list of all Players
     */
    public List<Player> getAllPlayers() {
        return ALL_PLAYERS;
    }

    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "FootballClub{" +
                super.toString() +
                "footballClubTotalStatistics=" + footballClubTotalStatistics +
                ", allPlayers=" + ALL_PLAYERS +
                '}';
    }

    /**
     * Overridden compareTo() method - to sort the clubs based on points
     * Needed for the displayPointsTable() method - to display club with most points on top
(descending order of points)
     * @param o - compare this FootballClub with o
     * @return - +ve value if this clubs points > o's points, -ve value if vice-versa, 0 if equal
     */
    @Override
    public int compareTo(FootballClub o) {
        return this.getFootballClubTotalStatistics().getPoints() -
o.getFootballClubTotalStatistics().getPoints();
    }


    /**
     * Equals() method called from the super class - SportsClub
     * @param o - compare this club with o
     * @return - t/f on whether the equality is satisfied
     */
    @Override
    public boolean equals(Object o) {
        return super.equals(o);
    }


    /**
     * If the above equals method returns true, the objects must have the same hashcode as well
     * @return - a hash value for the objects
     */
    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode());
    }
    /**
     * Private method that sets the randomly generated values provided by
playerInformationGeneration()
     * to 11 players (since a football club consists of 11 players) for the club being added
     */
    private void generatePlayers() {
        for (int i=0; i<FootballClub.getNumberOfPlayers(); i++) {
            List<Object> allPlayerInformation = playerInformationGeneration();
            //*each value is casted to their respective data types*//
            ALL_PLAYERS.add(new Player((double) allPlayerInformation.get(0), (String)
allPlayerInformation.get(1),
                    (String) allPlayerInformation.get(2), (PlayerStats) allPlayerInformation.get(3),
                    (String) allPlayerInformation.get(4), (String) allPlayerInformation.get(5),
                    (int) allPlayerInformation.get(6)
                    ));
        }
    }

    /**
     * Private helper method that handles the random generation of values for each attribute of a
player
     * @return a list of objects consisting of all the necessary values
     */
    private List<Object> playerInformationGeneration() {
        List<String> nationalities = new ArrayList<>(
                Arrays.asList("Germany", "Portugal", "Spain", "Italy", "England", "Argentina",
"Brazil",
                        "Netherlands", "Croatia", "Belgium", "Hungary", "Uruguay", "Czech
Republic",
                        "Bosnia and Herzegovina", "Denmark", "Ivory Coast", "Sweden", "Paraguay",
"Japan",
                        "France", "Cameroon", "Egypt", "Chile", "Colombia", "Wales"
            )
      );


      List<String> commonNames = new ArrayList<>(
            Arrays.asList("James", "John", "Robert", "Michael", "William", "David", "Richard",
"Joseph",
                        "Thomas", "Charles", "Christopher", "Daniel", "Matthew", "Anthony",
"Donald",
                        "Mark", "Paul", "Steven", "Andrew", "Kenneth", "Joshua", "Kevin", "Brian",
                        "George", "Edward", "Ronald", "Timothy", "Jason", "Jeffrey", "Ryan"
            )
      );


      List<String> positions = new ArrayList<>(
            Arrays.asList("GK", "CB", "LB", "RB", "LWB", "RWB", "SW", "DM", "CM", "AM",
"LM", "RM",
                        "CF", "S", "SS", "WF"
            )
      );
      List<String> preferredFoot = new ArrayList<>(Arrays.asList("R", "L", "B"));


      double playerHeight = FootballClub.RANDOM.nextInt(250 - 150 + 1) + 150;
      String playerName =
commonNames.get(FootballClub.RANDOM.nextInt(commonNames.size()));
      String playerNationality =
nationalities.get(FootballClub.RANDOM.nextInt(nationalities.size()));
      PlayerStats playerStats = new PlayerStats(
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1),
            FootballClub.RANDOM.nextInt(100 + 1)
      );
      //*call method that calculates and sets overall stat*//
      playerStats.setOverall();


      String playerPosition = positions.get(FootballClub.RANDOM.nextInt(positions.size()));
      String playerPreferredFoot =
preferredFoot.get(FootballClub.RANDOM.nextInt(preferredFoot.size()));
      int shirtNumber = FootballClub.RANDOM.nextInt(15 - 1 + 1) + 1;


      //*list consists of all randomly generated values for a specific player*//
      //*values are placed in an Object list, cuz it consists of values of different data types*//
      return Arrays.asList(playerHeight, playerName, playerNationality, playerStats,
playerPosition,
                              playerPreferredFoot, shirtNumber);
    }
}
```

FootballClubTotalStatistics

```
package coursework.models;


/*
 * FootballClubTotalStatistics
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.io.Serializable;


/**
 * FootballClubTotalStatistics class, which will be used to represent any football clubs total
statistics
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class FootballClubTotalStatistics implements Serializable {
    private static final long serialVersionUID = -30110101487621832L;
    private int defeats;
    private int draws;
    private int goalsAgainst;
    private int goalsFor;
    private int matchesPlayed;
    private int points;
    private int totalRedCards;
    private int totalYellowCards;
    private int wins;


    /**
```

```
     * initializes a club statistic object
     * all attributes initialized to 0, since a new club will not have any record
     */
    public FootballClubTotalStatistics() { }

    /**
     * @return total defeats of this club
     */
    public int getDefeats() {
        return defeats;
    }

    /**
     * sets defeats of this club
     * @param defeats - number of defeats of this club
     */
    public void setDefeats(int defeats) {
        this.defeats = defeats;
    }

    /**
     * @return total draws of this club
     */
    public int getDraws() {
        return draws;
    }

    /**
```

```
     * sets draws of this club
     * @param draws - number of draws of this club
     */
    public void setDraws(int draws) {
        this.draws = draws;
    }

    /**
     * @return total goals scored against this club
     */
    public int getGoalsAgainst() {
        return goalsAgainst;
    }

    /**
     * sets goals scored against this club
     * @param goalsAgainst - total number of goals scored against this club
     */
    public void setGoalsAgainst(int goalsAgainst) {
        this.goalsAgainst = goalsAgainst;
    }

    /**
     * @return total goal difference scored against this club
     */
    public int getGoalDifference() { return goalsFor - goalsAgainst; }

    /**
```

```
 * @return total goals scored by this club
 */
public int getGoalsFor() {
    return goalsFor;
}

/**
 * sets goals scored by this club
 * @param goalsFor - number of goals scored by this club
 */
public void setGoalsFor(int goalsFor) {
    this.goalsFor = goalsFor;
}

/**
 * @return total matches played by this club
 */
public int getMatchesPlayed() {
    return matchesPlayed;
}

/**
 * sets matches played by this club
 * @param matchesPlayed - total number of matches played by this club
 */
public void setMatchesPlayed(int matchesPlayed) {
    this.matchesPlayed = matchesPlayed;
}
```

```
/**
 * @return total points of this club
 */
public int getPoints() {
    return points;
}

/**
 * sets points obtained by this club
 * @param points - total number of points obtained by this club
 */
public void setPoints(int points) {
    this.points = points;
}

/**
 * @return total red cards obtained by this club
 */
public int getTotalRedCards() {
    return totalRedCards;
}

/**
 * sets red cards obtained by this club
 * @param totalRedCards - total number of red cards obtained by this club
 */
public void setTotalRedCards(int totalRedCards) {
    this.totalRedCards = totalRedCards;
}

/**
 * @return total yellow cards obtained by this club
 */
public int getTotalYellowCards() {
    return totalYellowCards;
}

/**
 * sets yellow cards obtained by this club
 * @param totalYellowCards - total number of yellow cards obtained by this club
 */
public void setTotalYellowCards(int totalYellowCards) {
    this.totalYellowCards = totalYellowCards;
}

/**
 * @return total wins of this club
 */
public int getWins() {
    return wins;
}

/**
 * sets wins of this club
 * @param wins - number of wins of this club
 */
public void setWins(int wins) {
    this.wins = wins;
}

/**
 * @return overrun toString() method
 */
@Override
public String toString() {
    return "FootballClubStatistics{" +
            "wins=" + wins +
            ", draws=" + draws +
            ", defeats=" + defeats +
            ", goalsFor=" + goalsFor +
            ", goalsAgainst=" + goalsAgainst +
            ", points=" + points +
            ", matchesPlayed=" + matchesPlayed +
            ", totalRedCards=" + totalRedCards +
            '}';
}
}
```

FootballMatch

```
package coursework.models;


/*
 * FootballMatch
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import coursework.services.PremierLeagueManager;


import java.io.Serializable;
import java.time.LocalDate;
import java.util.*;


/**
 * FootballMatch class, which will be used to represent any match between two Football clubs
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class FootballMatch implements Serializable, Comparable<FootballMatch> {
    //*serialization happened to refer to a different serial id for objects of FootballMatch,
therefore*//
    //*the expected value shown up on the terminal was hardcoded*//
    private static final long serialVersionUID = 1900807394549689165L;
    private static final Random RANDOM = new Random();
    private final FootballClub FIRST_TEAM;
    private final FootballClub SECOND_TEAM;
    private final LocalDate MATCH_DATE;
    private final SingleMatchFootballClubStatistic
FIRST_TEAM_SINGLE_MATCH_STATISTICS = new SingleMatchFootballClubStatistic();
    private final SingleMatchFootballClubStatistic
SECOND_TEAM_SINGLE_MATCH_STATISTICS = new
SingleMatchFootballClubStatistic();


    /**
     * initializes a match object
     * @param firstTeam - first team
     * @param secondTeam - second team
     * @param matchDate - date of a match
     */
    public FootballMatch(FootballClub firstTeam, FootballClub secondTeam, LocalDate
matchDate) {
        this.FIRST_TEAM = firstTeam;
        this.SECOND_TEAM = secondTeam;
        this.MATCH_DATE = matchDate;
    }


    /**
     * private helper method for both methods of match play
     * Will set the statistics of the match
     * @param  firstTeamScore - first teams score
     * @param  secondTeamScore - second teams score
     */
    private void playMatchCommonCode(int firstTeamScore, int secondTeamScore) {
        //*randomly generated values of statistics for a single match*//
        List<Integer> firstTeamRandomStats =
singleMatchRandomGeneratedStats(firstTeamScore);
```

```
        List<Integer> secondTeamRandomStats =
singleMatchRandomGeneratedStats(secondTeamScore);


        //*total club statistics of the two teams*//

        FootballClubTotalStatistics firstTeamTotalStats =
FIRST_TEAM.getFootballClubTotalStatistics();

        FootballClubTotalStatistics secondTeamTotalStats =
SECOND_TEAM.getFootballClubTotalStatistics();

        int firstTeamPossession = FootballMatch.RANDOM.nextInt(100 + 1);


        //*not randomly generated for both teams, since teams two's possession = 100 - team one's
possession*//

        FIRST_TEAM_SINGLE_MATCH_STATISTICS.setPossession(firstTeamPossession);


        //*use the randomly generated values to update the attributes of a club*//

        updateSingleMatchTeamStats(FIRST_TEAM_SINGLE_MATCH_STATISTICS,
firstTeamRandomStats, firstTeamTotalStats);


        //*second team's possession = 100 - first team's*//

        SECOND_TEAM_SINGLE_MATCH_STATISTICS.setPossession((100 -
firstTeamPossession));

        updateSingleMatchTeamStats(SECOND_TEAM_SINGLE_MATCH_STATISTICS,
secondTeamRandomStats, secondTeamTotalStats);

    }


    //**********************************MANUAL PLAY MATCH BETWEEN TWO
TEAMS****************************************//

    /**

     * Main function that handles playing of a manual match.

     * This function is called in the PremierLeagueManager class in addPlayedMatch() method

     * Handles the necessary updates
     * @param firstTeamScore - first teams score
     * @param secondTeamScore - second teams score
     */
    public void playMatch(int firstTeamScore, int secondTeamScore) {
        this.playMatchCommonCode(firstTeamScore, secondTeamScore);
        updateClubTotalStatistics();
    }
    //**********************************END MANUAL PLAY MATCH BETWEEN
TWO TEAMS***************************************//


    //*********************************RANDOM PLAY MATCH METHOD BETWEEN
TWO TEAMS*************************************//
    /**
     * Main function that handles playing of a single match randomly
     * This function is called in the PremierLeagueManager class in the
addPlayedMatchRandom() method
     * Which handles all the necessary updates
     */
    public void playMatchRandom() {
//        List<Integer> firstTeamRandomStats = singleMatchRandomGeneratedStats(-1);
//        List<Integer> secondTeamRandomStats = singleMatchRandomGeneratedStats(-1);
//        FootballClubTotalStatistics firstTeamTotalStats =
FIRST_TEAM.getFootballClubTotalStatistics();
//        FootballClubTotalStatistics secondTeamTotalStats =
SECOND_TEAM.getFootballClubTotalStatistics();
//        int firstTeamPossession = FootballMatch.RANDOM.nextInt(100 + 1);
//        FIRST_TEAM_SINGLE_MATCH_STATISTICS.setPossession(firstTeamPossession);
//        updateSingleMatchTeamStats(FIRST_TEAM_SINGLE_MATCH_STATISTICS,
firstTeamRandomStats, firstTeamTotalStats);
//        SECOND_TEAM_SINGLE_MATCH_STATISTICS.setPossession((100 -
firstTeamPossession));
```

```
//      updateSingleMatchTeamStats(SECOND_TEAM_SINGLE_MATCH_STATISTICS,
secondTeamRandomStats, secondTeamTotalStats);
        this.playMatchCommonCode(-1, -1);


        //*to prevent matches from having the same score results*//
        List<FootballMatch> footballMatches = PremierLeagueManager.getAllMatches();
        boolean uniqueGoal;
        //*the logic here is to continuously loop and generate goals until this match's score is
unique*//
        //*checking this football match's goals against all the FootballMatches in the list*//
        do {
            uniqueGoal = true;
            for (FootballMatch footballMatch : footballMatches) {
                if ((footballMatch.getFirstTeamSingleMatchStats().getGoals() ==
FIRST_TEAM_SINGLE_MATCH_STATISTICS.getGoals() &&
                        footballMatch.getSecondTeamSingleMatchStats().getGoals() ==
SECOND_TEAM_SINGLE_MATCH_STATISTICS.getGoals()) ||
                        (footballMatch.getSecondTeamSingleMatchStats().getGoals() ==
FIRST_TEAM_SINGLE_MATCH_STATISTICS.getGoals() &&
                                footballMatch.getFirstTeamSingleMatchStats().getGoals() ==
SECOND_TEAM_SINGLE_MATCH_STATISTICS.getGoals())) {
                    //*changing one teams goal will be sufficient for generating a new result*//

FIRST_TEAM_SINGLE_MATCH_STATISTICS.setGoals(FootballMatch.RANDOM.nextInt(1
5 + 1));
                    uniqueGoal = false;
                }
            }
        } while (!uniqueGoal);
        //method that updates the total overall statistics (the above methods generate stats of a team
in a match)
        updateClubTotalStatistics();
    }


    /**
     * updateClubTotalStatistics()
     * Private helper method that uses the private helper method setTeamTotalStats()
     * To update the stats of a club
     */
    private void updateClubTotalStatistics() {
        FootballClubTotalStatistics firstTeamTotalStats =
FIRST_TEAM.getFootballClubTotalStatistics();
        FootballClubTotalStatistics secondTeamTotalStats =
SECOND_TEAM.getFootballClubTotalStatistics();
        int firstTeamMatchGoal = FIRST_TEAM_SINGLE_MATCH_STATISTICS.getGoals();
        int secondTeamMatchGoal =
SECOND_TEAM_SINGLE_MATCH_STATISTICS.getGoals();


        //*if first team wins, they get 3 points, if they lose the second teams gets 3, if its a draw*//
        //*both get a point each*//
        //*According to the team that is passes as the first argument the logic above occurs*//
        if (firstTeamMatchGoal > secondTeamMatchGoal) {
            setTeamTotalStats(firstTeamTotalStats, firstTeamMatchGoal, secondTeamTotalStats,
secondTeamMatchGoal,
                    3);
        } else if (secondTeamMatchGoal > firstTeamMatchGoal) {
            setTeamTotalStats(secondTeamTotalStats, secondTeamMatchGoal, firstTeamTotalStats,
firstTeamMatchGoal,
                    3);
        } else {
            setTeamTotalStats(firstTeamTotalStats, firstTeamMatchGoal, secondTeamTotalStats,
secondTeamMatchGoal,
                    1);
        }
        FIRST_TEAM.setFootballClubTotalStatistics(firstTeamTotalStats);
        SECOND_TEAM.setFootballClubTotalStatistics(secondTeamTotalStats);
    }

    /**
     * A private helper method of updateClubTotalStatistics() that will set the stats of each team
     * To avoid duplication of code, since both teams values have to be set
     * @param firstTeam first teams total statistics
     * @param firstTeamMatchGoal first teams goals scored for this match
     * @param secondTeam second teams goals scored for this match
     * @param secondTeamMatchGoal second teams total statistics
     * @param points points of first team (to classify between wins and losses)
     */
    private void setTeamTotalStats(FootballClubTotalStatistics firstTeam, int
firstTeamMatchGoal,
                                   FootballClubTotalStatistics secondTeam, int secondTeamMatchGoal, int
points) {
        firstTeam.setGoalsAgainst(firstTeam.getGoalsAgainst() + secondTeamMatchGoal);
        firstTeam.setGoalsFor(firstTeam.getGoalsFor() + firstTeamMatchGoal);
        firstTeam.setMatchesPlayed(firstTeam.getMatchesPlayed() + 1);

        //*if points = 3, then a team has won*//
        if (points == 3) {
            firstTeam.setPoints(firstTeam.getPoints() + points);
            firstTeam.setWins(firstTeam.getWins() + 1);
            secondTeam.setDefeats(secondTeam.getDefeats() + 1);
        //*if not, it was a draw*//
        } else {
            firstTeam.setPoints(firstTeam.getPoints() + 1);
            secondTeam.setPoints(secondTeam.getPoints() + 1);
            firstTeam.setDraws(firstTeam.getDraws() + 1);
            secondTeam.setDraws(secondTeam.getDraws() + 1);
        }

        secondTeam.setGoalsAgainst(secondTeam.getGoalsAgainst() + firstTeamMatchGoal);
        secondTeam.setGoalsFor(secondTeam.getGoalsFor() + secondTeamMatchGoal);
        secondTeam.setMatchesPlayed(secondTeam.getMatchesPlayed() + 1);
    }

    /**
     * private helper method of playMatch() that generates random values for the stats of a team
for a match
     * @param goals - will hold the goals scored entered, if manual play, else will be -1 for
random generation
     * @return list of randomly generated values
     */
    private List<Integer> singleMatchRandomGeneratedStats(int goals) {
        //generate random values within reasonable ranges for each stat recorded in a match**//
        int corners = FootballMatch.RANDOM.nextInt(30 - 5 + 1) + 5;
        if (goals == -1) {
            goals = FootballMatch.RANDOM.nextInt(15 + 1);
        }
        int fouls = FootballMatch.RANDOM.nextInt(15 - 5 + 1) + 5;
        int offsides = FootballMatch.RANDOM.nextInt(10 - 5 + 1) + 5;
        int passes = FootballMatch.RANDOM.nextInt(700 - 300 + 1) + 300;
        int passAccuracy = FootballMatch.RANDOM.nextInt(100 + 1);
        int redCards = FootballMatch.RANDOM.nextInt(3 + 1);
        int shots = FootballMatch.RANDOM.nextInt(50 - 20 + 1) + 20;
        int shotsOnTarget = FootballMatch.RANDOM.nextInt(50 + 1);
        int yellowCards = FootballMatch.RANDOM.nextInt(5 + 1);
        return Arrays.asList(corners, goals, fouls, offsides, passes, passAccuracy, redCards, shots,
shotsOnTarget,
                yellowCards);
    }


    /**
     * private helper method of playMatch() that uses the list of values to set the stats for each
team of the match
     * @param singleMatchFootballClubStatistic - a clubs statistics for a single match (ex: shots,
shots on target,
     *                                          fouls etc...)
     * @param teamRandomStats - the list that holds all the randomly generated values (generated
from the above helper
     *                          function
     * @param footballClubTotalStatistics - a clubs total overall statistics (required here to modify
total number of
     *                                      yellow and red cards obtained by a team
     */
    private void updateSingleMatchTeamStats(SingleMatchFootballClubStatistic
singleMatchFootballClubStatistic,
                                            List<Integer> teamRandomStats,
                                            FootballClubTotalStatistics footballClubTotalStatistics) {
        //*method that uses the randomly generated values to set each field*//
        singleMatchFootballClubStatistic.setCorners(teamRandomStats.get(0));
        singleMatchFootballClubStatistic.setGoals(teamRandomStats.get(1));
        singleMatchFootballClubStatistic.setFouls(teamRandomStats.get(2));
        singleMatchFootballClubStatistic.setOffsides(teamRandomStats.get(3));
        singleMatchFootballClubStatistic.setPasses(teamRandomStats.get(4));
        singleMatchFootballClubStatistic.setPassAccuracy(teamRandomStats.get(5));
        singleMatchFootballClubStatistic.setRedCards(teamRandomStats.get(6));
        singleMatchFootballClubStatistic.setShots(teamRandomStats.get(7));
        singleMatchFootballClubStatistic.setShotsOnTarget(teamRandomStats.get(8));
        singleMatchFootballClubStatistic.setYellowCards(teamRandomStats.get(9));

        //*also update the overall stats (total red and yellow cards obtained by a club)*//

footballClubTotalStatistics.setTotalRedCards(footballClubTotalStatistics.getTotalRedCards() +
                                                  teamRandomStats.get(6));

footballClubTotalStatistics.setTotalYellowCards(footballClubTotalStatistics.getTotalYellowCar
ds() +
                teamRandomStats.get(9));
    }
    //************************************END PLAY MATCH METHOD BETWEEN
TWO TEAMS************************************//


    /**
     * Returns the first team's statistics for this match
     * @return - first team match stats
     */
    public SingleMatchFootballClubStatistic getFirstTeamSingleMatchStats() {
        return FIRST_TEAM_SINGLE_MATCH_STATISTICS;
    }
    /**
     * Returns the second team's statistics for this match
     * @return - second team match stats
     */
    public SingleMatchFootballClubStatistic getSecondTeamSingleMatchStats() {
        return SECOND_TEAM_SINGLE_MATCH_STATISTICS;
    }

    /**
     * @return date of a match
     */
    public LocalDate getMatchDate() {
        return MATCH_DATE;
    }

//    /**
//     * sets date of a match
//     * @param matchDate - date of a specific match
//     */
//    public void setMatchDate(LocalDate matchDate) {
//        this.matchDate = matchDate;
//    }

    /**
     * @return football club 1
     */
    public FootballClub getFirstTeam() {
        return FIRST_TEAM;
    }

    /**
     * @return football club 2
     */
    public FootballClub getSecondTeam() {
        return SECOND_TEAM;
    }

    /**
     * overridden compareTo() method - to sort the list of matches based on date
     * Needed for the displayMatchResults() method - to display most recent on top (descending
order of date)
     * @param o - compares this date with o's date
     * @return - if this date > o's date returns a +ve value, a -ve vice versa, and 0 if they equal
     */
    @Override
    public int compareTo(FootballMatch o) {
        return this.getMatchDate().compareTo(o.getMatchDate());
    }

    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "FootballMatch{" +
                "footballClub1=" + FIRST_TEAM +
                ", footballClub1 match stats=" + FIRST_TEAM_SINGLE_MATCH_STATISTICS +
                ", footballClub2=" + SECOND_TEAM +
                ", footballClub2 match stats=" + SECOND_TEAM_SINGLE_MATCH_STATISTICS
+
                ", matchDate=" + MATCH_DATE +
                '}';
    }


    /**
     * Overrun equals method to check for any FootballMatch equality
     * Special equals method, that checks first team against first and second, and does the same for
second team
     * This is done since the first team and second team could be in different ordering, but the
match it refers to is
     * still the same
     * Not all attributes are checked against, since they can be duplicated
     * @param o - compare this FootballMatch with o
     * @return - t/f on whether the equality is satisfied
     */
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        FootballMatch that = (FootballMatch) o;
        return (Objects.equals(FIRST_TEAM, that.FIRST_TEAM) &&
                Objects.equals(SECOND_TEAM, that.SECOND_TEAM)) ||
                (Objects.equals(SECOND_TEAM, that.FIRST_TEAM) &&
                Objects.equals(FIRST_TEAM, that.SECOND_TEAM));
    }
    /**
     * If the above equals method returns true, the objects must have the same hashcode as well
     * @return - a hash value for the objects
     */
    @Override
    public int hashCode() {
        return Objects.hash(FIRST_TEAM, SECOND_TEAM);
    }
}
```

LeagueManager

```
package coursework.models;

/*
 * LeagueManager
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import java.awt.*;


/**
 * LeagueManager interface, which will be used to hold generic methods that applies to any sport
league
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public interface LeagueManager {
   void addClub(String clubTypeInput, String lecOrTeachInput, String clubNameInput, String
clubLocationInput,
                  String clubOwnerInput, String clubSponsorInput, Color colorTop, Color colorShort,
String netWorth);
   SportsClub deleteClub(String clubNameInput);
   SportsClub displaySelectedClub(String clubNameInput);
   FootballMatch displaySelectedMatch(String firstTeamInput, String secondTeamInput);
   FootballMatch addPlayedMatchRandom(String season);
   void addPlayedMatch(String season, String date, String firstTeamInput, String
secondTeamInput, int firstTeamScore, int secondTeamScore);
   void displayPointsTable();
   void displayMatchResults();
   void saveData(String season);
   void loadData(String season);
}
```

Player

```
package coursework.models;

/*
 * Player
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import java.io.Serializable;

/**
 * Player class, which will be used to represent any Player
 * @version 1.x November 17th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class Player implements Serializable {
    private static final long serialVersionUID = 5625338058257534947L;
    private final double HEIGHT;
    private final String NAME;
    private final String NATIONALITY;
    private final PlayerStats PLAYER_STATS;
    private final String POSITION;
    private final String PREFERRED_FOOT;
    private final int SHIRT_NUMBER;

    /**
     * initializes all info of a Player
     * @param height - player height
     * @param name - player name
     * @param nationality - player nationality
     * @param playerStats - player Stats
     * @param position - player position
     * @param preferredFoot - player preferred foot
     * @param shirtNumber - player shirt number
     */
    public Player(double height, String name, String nationality, PlayerStats playerStats, String
position,
                  String preferredFoot, int shirtNumber) {
        this.HEIGHT = height;
        this.NAME = name;
        this.NATIONALITY = nationality;
        this.PLAYER_STATS = playerStats;
        this.POSITION = position;
        this.PREFERRED_FOOT = preferredFoot;
        this.SHIRT_NUMBER = shirtNumber;
    }

    /**
     * @return - this players height
     */
    public double getHeight() {
        return HEIGHT;
    }

    /**
     * @return - this players name
     */
    public String getName() {
        return NAME;
    }

    /**
     * @return - this players nationality
     */
    public String getNationality() {
        return NATIONALITY;
    }

    /**
     * @return - this players stats
     */
    public PlayerStats getPlayerStats() {
        return PLAYER_STATS;
    }

    /**
     * @return - this players position
     */
    public String getPosition() {
        return POSITION;
    }

    /**
     * @return - this players preferred foot
     */
    public String getPreferredFoot() {
        return PREFERRED_FOOT;
    }

    /**
     * @return - this players shirt number
     */
    public int getShirtNumber() {
        return SHIRT_NUMBER;
    }

    /**
     * @return - overrun toString() method
     */
    @Override
    public String toString() {
        return "Player{" +
                "height=" + HEIGHT +
                ", name='" + NAME + '\'' +
                ", nationality='" + NATIONALITY + '\'' +
                ", playerStats=" + PLAYER_STATS +
                ", position='" + POSITION + '\'' +
                ", preferredFoot='" + PREFERRED_FOOT + '\'' +
                ", shirtNumber=" + SHIRT_NUMBER +
                '}';
    }
}
```

PlayerStats

```
package coursework.models;


/*
 * PlayerStats
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.io.Serializable;


/**
 * PlayerStats class, which will be used to represent any Players stats
 * @version 1.x November 17th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class PlayerStats implements Serializable {
    private final int ACCURACY;
    private final int CONTROL;
    private final int CROSSING;
    private final int GOAL_KEEPING_HANDLING;
    private final int GOAL_KEEPING_PRESSURE;
    private final int GOAL_KEEPING_REACTION;
    private final int HEADING;
    private final int PASSING;
    private int overall;
    private final int SHOOTING;
    private final int SPEED;
    private final int STAMINA;
    private final int STRENGTH;

    private final int TACKLING;


    /**

     * Initializes all the player stats

     * @param accuracy - player accuracy

     * @param control - player control

     * @param crossing - player crossing

     * @param goalKeeperHandling - player GKH

     * @param goalKeepingPressure - player GKP

     * @param goalKeepingReaction - player GKR

     * @param heading - player heading

     * @param passing - player passing

     * @param shooting - player shooting

     * @param speed - player speed

     * @param stamina - player stamina

     * @param strength - player strength

     * @param tackle - player tackling

     */

    public PlayerStats(int accuracy, int control, int crossing, int goalKeeperHandling, int 
goalKeepingPressure,

                       int goalKeepingReaction, int heading, int passing, int shooting, int speed, int 
stamina,

                       int strength, int tackle) {

        this.ACCURACY = accuracy;

        this.CONTROL = control;

        this.CROSSING = crossing;

        this.GOAL_KEEPING_HANDLING = goalKeeperHandling;

        this.GOAL_KEEPING_PRESSURE = goalKeepingPressure;
        this.GOAL_KEEPING_REACTION = goalKeepingReaction;
        this.HEADING = heading;
        this.PASSING = passing;
        this.SHOOTING = shooting;
        this.SPEED = speed;
        this.STAMINA = stamina;
        this.STRENGTH = strength;
        this.TACKLING = tackle;
    }

    /**
     * @return - this players accuracy
     */
    public int getAccuracy() {
        return ACCURACY;
    }

    /**
     * @return - this players control
     */
    public int getControl() {
        return CONTROL;
    }

    /**
     * @return - this players crossing
     */
    public int getCrossing() {
        return CROSSING;
    }

    /**
     * @return - this players GKH
     */
    public int getGoalKeeperHandling() {
        return GOAL_KEEPING_HANDLING;
    }

    /**
     * @return - this players GKP
     */
    public int getGoalKeepingPressure() {
        return GOAL_KEEPING_PRESSURE;
    }

    /**
     * @return - this players GKR
     */
    public int getGoalKeepingReaction() {
        return GOAL_KEEPING_REACTION;
    }

    /**
     * @return - this players heading
     */
    public int getHeading() {
        return HEADING;
    }

    /**
     * @return - this players passing
     */
    public int getPassing() {
        return PASSING;
    }

    /**
     * @return - this players overall
     */
    public int getOverall() {
        return overall;
    }

    /**
     * sets overall stat of the player
     */
    public void setOverall() {
        this.overall = (this.ACCURACY + this.CONTROL + this.CROSSING +
this.GOAL_KEEPING_HANDLING +
                this.GOAL_KEEPING_PRESSURE + this.GOAL_KEEPING_REACTION +
this.HEADING + this.PASSING +
                this.SHOOTING + this.SPEED + this.STAMINA + this.STRENGTH +
this.TACKLING) / 13;
    }
```

```
/**
 * @return - this players speed
 */
public int getSpeed() {
    return SPEED;
}

/**
 * @return - this players stamina
 */
public int getStamina() {
    return STAMINA;
}

/**
 * @return - this players strength
 */
public int getStrength() {
    return STRENGTH;
}

/**
 * @return - this players tackling
 */
public int getTackle() {
    return TACKLING;
}
/**
```

```
     * @return - this players shooting
     */
    public int getShooting() {
        return SHOOTING;
    }
    /**
     * @return - overrun toString() method
     */
    @Override
    public String toString() {
        return "PlayerStats{" +
                "accuracy=" + ACCURACY +
                ", control=" + CONTROL +
                ", crossing=" + CROSSING +
                ", goalKeeperHandling=" + GOAL_KEEPING_HANDLING +
                ", goalKeepingPressure=" + GOAL_KEEPING_PRESSURE +
                ", goalKeepingReaction=" + GOAL_KEEPING_REACTION +
                ", heading=" + HEADING +
                ", passing=" + PASSING +
                ", overall=" + overall +
                ", shooting=" + SHOOTING +
                ", speed=" + SPEED +
                ", stamina=" + STAMINA +
                ", strength=" + STRENGTH +
                ", tackle=" + TACKLING +
                '}';
    }
}
```

SchoolFootballClub

```
package coursework.models;


/*
 * SchoolFootballClub
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.util.Objects;


/**
 * SchoolFootballClub class, which will be used to represent any school football club (subclass
of FootballClub)
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class SchoolFootballClub extends FootballClub {
    private final String TEACHER_IN_CHARGE;


    /**
     * initializes a new SchoolFootballClub object
     * @param clubName - name of club
     * @param clubLocation - location of club
     * @param clubOwner - owner of club
     * @param kit - club kit (a helper class)
     * @param clubWorth - club net worth
     * @param teacherInCharge - teacher in-charge of this school football club
     */
    public SchoolFootballClub(String clubName, String clubLocation, String clubOwner,
SportsClubKit kit,
                              String teacherInCharge, String clubWorth) {
        super(clubName, clubLocation, clubOwner, kit, clubWorth);
        this.TEACHER_IN_CHARGE = teacherInCharge;
    }

    /**
     * @return teacher in-charge of the school club
     */
    public String getTeacherInCharge() {
        return TEACHER_IN_CHARGE;
    }

//    /**
//     * sets teacher in-charge of the school club
//     * @param teacherInCharge - teacher in-charge of the school club
//     */
//    public void setTeacherInCharge(String teacherInCharge) {
//        this.teacherInCharge = teacherInCharge;
//    }

    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "SchoolFootballClub{" +
                super.toString() +
                ", teacherInCharge='" + TEACHER_IN_CHARGE + '\'' +
                '}';
    }

    /**
     * Equals() method called from the super class - FootballClub
     * @param o - compare this club with o
     * @return - t/f on whether the equality is satisfied
     */
    @Override
    public boolean equals(Object o) {
        return super.equals(o);
    }

    /**
     * If the above equals method returns true, the objects must have the same hashcode as well
     * @return - a hash value for the objects
     */
    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode());
    }
}
```

SingleMatchFootballClubStatistic

```
package coursework.models;


/*
 * SingleMatchFootballClubStatistic
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.io.Serializable;


/**
 * SingleMatchFootballClubStatistic class, which will be used to represent any football clubs
single match statistic
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class SingleMatchFootballClubStatistic implements Serializable {
    private int corners;
    private int fouls;
    private int goals;
    private int offsides;
    private int passes;
    private int passAccuracy;
    private int possession;
    private int redCards;
    private int shots;
    private int shotsOnTarget;
    private int yellowCards;
```

```
/**
 * initializes a single match club statistic object
 * all attributes initialized to 0, since a new club will not have any record
 */
public SingleMatchFootballClubStatistic() { }

/**
 * @return total corners of this club for a match
 */
public int getCorners() {
   return corners;
}

/**
 * sets total corners of this club for a match
 * @param corners - number of corners of this club for a match
 */
public void setCorners(int corners) {
   this.corners = corners;
}

/**
 * @return total fouls of this club for a match
 */
public int getFouls() {
   return fouls;
}
```

```
/**
 * sets total fouls of this club for a match
 * @param fouls - number of fouls of this club for a match
 */
public void setFouls(int fouls) {
    this.fouls = fouls;
}

/**
 * @return total goals of this club for a match
 */
public int getGoals() {
    return goals;
}

/**
 * sets total goals of this club for a match
 * @param goals - number of goals of this club for a match
 */
public void setGoals(int goals) {
    this.goals = goals;
}

/**
 * @return total offsides of this club for a match
 */
public int getOffsides() {
    return offsides;
}

/**
 * sets total offsides of this club for a match
 * @param offsides - number of offsides of this club for a match
 */
public void setOffsides(int offsides) {
    this.offsides = offsides;
}

/**
 * @return total passes of this club for a match
 */
public int getPasses() {
    return passes;
}

/**
 * sets total passes of this club for a match
 * @param passes - number of passes of this club for a match
 */
public void setPasses(int passes) {
    this.passes = passes;
}

/**
 * @return pass accuracy of this club for a match
 */
```

```
    public int getPassAccuracy() {
        return passAccuracy;
    }

    /**
     * sets pass accuracy of this club for a match
     * @param passAccuracy - pass accuracy of this club for a match
     */
    public void setPassAccuracy(int passAccuracy) {
        this.passAccuracy = passAccuracy;
    }

    /**
     * @return possession of this club for a match
     */
    public int getPossession() {
        return possession;
    }

    /**
     * sets possession of this club for a match
     * @param possession - possession of this club for a match
     */
    public void setPossession(int possession) {
        this.possession = possession;
    }

    /**
```

```
     * @return total red of this club for a match
     */
    public int getRedCards() {
        return redCards;
    }

    /**
     * sets total red cards of this club for a match
     * @param redCards - number of red cards this club for a match
     */
    public void setRedCards(int redCards) {
        this.redCards = redCards;
    }

    /**
     * @return total shots of this club for a match
     */
    public int getShots() {
        return shots;
    }

    /**
     * sets total shots of this club for a match
     * @param shots - number of shots this club for a match
     */
    public void setShots(int shots) {
        this.shots = shots;
    }
    /**
     * @return total shots on target of this club for a match
     */
    public int getShotsOnTarget() {
        return shotsOnTarget;
    }

    /**
     * sets total of shorts on target of this club for a match
     * @param shotsOnTarget - number of shots on target this club for a match
     */
    public void setShotsOnTarget(int shotsOnTarget) {
        this.shotsOnTarget = shotsOnTarget;
    }

    /**
     * @return total yellow cards of this club for a match
     */
    public int getYellowCards() {
        return yellowCards;
    }

    /**
     * sets total yellow cards of this club for a match
     * @param yellowCards - number of yellow cards this club for a match
     */
    public void setYellowCards(int yellowCards) {
        this.yellowCards = yellowCards;
    }

    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "SingleMatchFootballClubStatistic{" +
                "corners=" + corners +
                ", fouls=" + fouls +
                ", goals=" + goals +
                ", offsides=" + offsides +
                ", passes=" + passes +
                ", passAccuracy=" + passAccuracy +
                ", possession=" + possession +
                ", redCards=" + redCards +
                ", shots=" + shots +
                ", shotsOnTarget=" + shotsOnTarget +
                ", yellowCards=" + yellowCards +
                '}';
    }
}
```

SportsClub

```
package coursework.models;


/*
 * SportsClub
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.io.Serializable;
import java.util.Objects;


/**
 * SportsClub class, which will be used to represent any generic sport club (super class of
FootballClub)
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public abstract class SportsClub implements Serializable {
    private static final long serialVersionUID = -1367205871791753063L;
    private final int AMOUNT_OF_PLAYERS;
    private final SportsClubKit KIT;
    private final String CLUB_LOCATION;
    private final String CLUB_NAME;
    private final String CLUB_OWNER;
    private final String CLUB_NET_WORTH;


    /**
     * initializes a new sports club
     * @param clubName - name of club
     * @param clubLocation - location of club
     * @param clubOwner - owner of club
     * @param kit - club kit (a helper class)
     * @param amountOfPlayers - number of players in a team of a specific sports club
     * @param clubNetWorth - net worth of this club
     */
    public SportsClub(String clubName, String clubLocation, String clubOwner, SportsClubKit
kit, int amountOfPlayers,
                      String clubNetWorth) {
        this.CLUB_NAME = clubName;
        this.CLUB_LOCATION = clubLocation;
        this.CLUB_OWNER = clubOwner;
        this.KIT = kit;
        this.AMOUNT_OF_PLAYERS = amountOfPlayers;
        this.CLUB_NET_WORTH = clubNetWorth;
    }

    /**
     * @return amount of players in a team of a sports club
     */
    public int getAmountOfPlayers() {
        return AMOUNT_OF_PLAYERS;
    }

//    /**
//     * sets number of players of a sports club
//     * @param amountOfPlayers - number of players of a sports club
//     */
//    public void setAmountOfPlayers(int amountOfPlayers) {
```

```
//        this.amountOfPlayers = amountOfPlayers;
//    }

    /**
     * @return kit of club
     */
    public SportsClubKit getKit() {
        return KIT;
    }

//    /**
//     * sets a new club kit  of a club
//     * @param kit - club kit of a club
//     */
//    public void setKit(SportsClubKit kit) {
//        this.kit = kit;
//    }

    /**
     * @return location of club
     */
    public String getClubLocation() {
        return CLUB_LOCATION;
    }

//    /**
//     * sets location of a club
//     * @param clubLocation - location of club
//     */
//    public void setClubLocation(String clubLocation) {
//        this.clubLocation = clubLocation;
//    }

    /**
     * @return name of club
     */
    public String getClubName() {
        return CLUB_NAME;
    }
//    /**
//     * sets name of a club
//     * @param clubName - name of club
//     */
//    public void setClubName(String clubName) {
//        this.clubName = clubName;
//    }

    /**
     * @return owner of club
     */
    public String getClubOwner() {
        return CLUB_OWNER;
    }
//    /**
//     * sets owner of a club
//     * @param clubOwner - owner of club
//     */
//    public void setClubOwner(String clubOwner) {
//        this.clubOwner = clubOwner;
//    }
    /**
     * @return club net worth
     */
    public String getClubNetWorth() {
        return CLUB_NET_WORTH;
    }
//    /**
//     * sets net worth of a club
//     * @param clubNetWorth - net worth of club
//     */
//    public void setClubNetWorth(String clubNetWorth) {
//        this.clubNetWorth = clubNetWorth;
//    }
    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "SportsClub{" +
                "amountOfPlayers=" + AMOUNT_OF_PLAYERS +
                ", kit=" + KIT +
                ", clubLocation='" + CLUB_LOCATION + '\'' +
                ", clubName='" + CLUB_NAME + '\'' +
                ", clubOwner='" + CLUB_OWNER + '\'' +
                ", clubNetWorth=" + CLUB_NET_WORTH +
                '}';
    }
    /**
     * Overrun equals method to check for any SportsClub equality
     * Not all attributes are checked against, since they can be duplicated
     * This is placed in the super class, so every class that inherits this, get the equals() method
     * @param o - compare this SportsClub with o
     * @return - t/f on whether the equality is satisfied
     */
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SportsClub that = (SportsClub) o;
        return Objects.equals(CLUB_LOCATION, that.CLUB_LOCATION) &&
                Objects.equals(CLUB_NAME, that.CLUB_NAME) &&
                Objects.equals(CLUB_OWNER, that.CLUB_OWNER);
    }
    /**
     * If the above equals method returns true, the objects must have the same hashcode as well
     * @return - a hash value for the objects
     */
    @Override
    public int hashCode() {
        return Objects.hash(CLUB_LOCATION, CLUB_NAME, CLUB_OWNER);
    }
}
```

SportsClubKit

```
package coursework.models;


/*
 * SportsClubKit
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.awt.*;
import java.io.Serializable;


/**
 * SportsClubKit class, which will be used to represent any generic sports club kit
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class SportsClubKit implements Serializable {
    private final Color BOTTOM_COLOR;
    private final String SPONSOR;
    private final Color TOP_COLOR;


    /**
     * initializes a sports club kit
     * @param sponsor - sponsor of the club
     * @param topColor - t-shirt color
     * @param bottomColor - short color
     */
    public SportsClubKit(String sponsor, Color topColor, Color bottomColor) {
        this.SPONSOR = sponsor;
        this.TOP_COLOR = topColor;
        this.BOTTOM_COLOR = bottomColor;
    }

    /**
     * @return bottom color
     */
    public Color getBottomColor() {
        return BOTTOM_COLOR;
    }

    /**
     * @return sponsor
     */
    public String getSponsor() {
        return SPONSOR;
    }

    /**
     * @return top color
     */
    public Color getTopColor() {
        return TOP_COLOR;
    }
    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "ClubKit{" +
                "sponsor='" + SPONSOR + '\'' +
                ", topColor=" + TOP_COLOR +
                ", bottomColor=" + BOTTOM_COLOR +
                '}';
    }
}
```

UniversityFootballClub

```
package coursework.models;


/*
 * UniversityFootballClub
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.util.Objects;


/**
 * UniversityFootballClub class, which will be used to represent any university football club
(subclass of FootballClub)
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class UniversityFootballClub extends FootballClub {
    private final String LECTURER_IN_CHARGE;


    /**
     * initializes a new UniversityFootballClub object
     * @param clubName - name of club
     * @param clubLocation - location of club
     * @param clubOwner - owner of club
     * @param kit - club kit (a helper class)
     * @param clubWorth - club net worth
     * @param lecturerInCharge - lecturer in-charge of this university football club
     */
    public UniversityFootballClub(String clubName, String clubLocation, String clubOwner,
SportsClubKit kit,
                                  String lecturerInCharge, String clubWorth) {
        super(clubName, clubLocation, clubOwner, kit, clubWorth);
        this.LECTURER_IN_CHARGE = lecturerInCharge;
    }

    /**
     * @return lecturer in-charge of the university club
     */
    public String getLecturerInCharge() {
        return LECTURER_IN_CHARGE;
    }

//    /**
//     * sets lecturer in-charge of the university club
//     * @param lecturerInCharge - teacher in-charge of the university club
//     */
//    public void setLecturerInCharge(String lecturerInCharge) {
//        this.lecturerInCharge = lecturerInCharge;
//    }

    /**
     * @return overrun toString() method
     */
    @Override
    public String toString() {
        return "UniversityFootballClub{" +
                super.toString() +
                ",lecturerInCharge='" + LECTURER_IN_CHARGE + '\'' +
                '}';
    }

    /**
     * Equals() method called from the super class - FootballClub
     * @param o - compare this club with o
     * @return - t/f on whether the equality is satisfied
     */
    @Override
    public boolean equals(Object o) {
        return super.equals(o);
    }

    /**
     * If the above equals method returns true, the objects must have the same hashcode as well
     * @return - a hash value for the objects
     */
    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode());
    }
}
```

2.3 Services

PremierLeagueManager

```
package coursework.services;


/*
 * PremierLeagueManager
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import coursework.models.*;
import coursework.utils.GoalDifferenceComparator;


import java.awt.*;
import java.io.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.List;


/**
 * PremierLeagueManager class, the class that will implement all the manipulation of a
FootballClub
 * @version 1.x November 9th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class PremierLeagueManager implements LeagueManager {
    private static final String SAVE_PATH =
"C:\\Users\\Ammuuu\\Downloads\\learning\\UNI\\OOP-Module\\Coursework\\" +
                                            "OOP-COURSEWORK\\saveFile";
    private static final String SEASON_INPUT =
"C:\\Users\\Ammuuu\\Downloads\\learning\\UNI\\OOP-Module\\Coursework\\" +

            "OOP-COURSEWORK\\SeasonInputPlay";

    private static final int MAX_SIZE = 20;

    private static final Random RANDOM = new Random();

    private static List<FootballMatch> allMatches = new ArrayList<>();

    private static List<FootballClub> allFootballClubs = new ArrayList<>();




    //*************************************************ADD
METHOD*******************************************************//

    /**

     * Method implementation of addClub(), overrun from the LeagueManager interface

     * This method handles the functionality of adding/promoting a club to the PremierLeague

     * @param clubTypeInput - type of club input (university, school or league)

     * @param lecOrTeachInput - if chosen club type was school or university, holds the name of
the teacher or lecturer

     * @param clubNameInput - name of the club

     * @param clubLocationInput - location of the club

     * @param clubOwnerInput - owner of the club

     * @param clubSponsorInput  clubs sponsor

     * @param colorTop - t-shirt color

     * @param colorShort - short color

     * @param netWorth - clubs net worth

     */

    @Override

    public void addClub(String clubTypeInput, String lecOrTeachInput, String clubNameInput,
String clubLocationInput,
                        String clubOwnerInput, String clubSponsorInput, Color colorTop, Color
colorShort, String netWorth) {
        FootballClub footballClub = null;


        //*creates a FootballClub based on the input club type*//
        switch (clubTypeInput) {
            case "university":
                footballClub = new UniversityFootballClub(clubNameInput, clubLocationInput,
clubOwnerInput,
                                                            new SportsClubKit(clubSponsorInput, colorTop,
colorShort),
                                                            lecOrTeachInput, netWorth);
                break;
            case "school":
                footballClub = new SchoolFootballClub(clubNameInput, clubLocationInput,
clubOwnerInput,
                                                        new SportsClubKit(clubSponsorInput, colorTop, colorShort),
                                                        lecOrTeachInput, netWorth);
                break;
            case "league":
                footballClub = new FootballClub(clubNameInput, clubLocationInput,
clubOwnerInput,
                                                  new SportsClubKit(clubSponsorInput, colorTop, colorShort),
netWorth);
                break;
        }


        allFootballClubs.add(footballClub);
    }
    //***********************************************END OF ADD
METHOD*************************************************//
```

```
    //*********************************************DELETE
METHOD*************************************************//
    /**
     * Method implementation of deleteClub(), overrun from the LeagueManager interface
     * This method handles the functionality of deleting/relegating a club from the PremierLeague
     * Deletes based on club name input
     * @param clubNameInput - name of club wished to relegate
     * @return - the deleted club if found, else returns null
     */
    @Override
    public FootballClub deleteClub(String clubNameInput) {
        FootballClub removedClub = null;

        for (FootballClub footballClub : allFootballClubs) {
            if (footballClub.getClubName().equals(clubNameInput)) {
                removedClub = allFootballClubs.remove(allFootballClubs.indexOf(footballClub));
                break; //*improve efficiency ==> Once found, there's no need to keep looking*//
            }
        }

        return removedClub;
    }
    //*******************************************END OF DELETE
METHOD*******************************************//
```

```
    //*********************************************DISPLAY SELECTED
CLUB*********************************************//
    /**
     * Method implementation of displaySelectedClub(), overrun from the LeagueManager
interface
     * This method handles the functionality of displaying a club in the PremierLeague
     * Identifies based on club name, follows same logic as the deleteClub() method
     * @param clubNameInput - name of club wished to display
     * @return - the found club, if found, else returns null
     */
    @Override
    public FootballClub displaySelectedClub(String clubNameInput) {
        FootballClub foundClub = null;

        for (FootballClub footballClub : allFootballClubs) {
            if (footballClub.getClubName().equals(clubNameInput)) {
                foundClub = footballClub;
                break;
            }
        }

        return foundClub;
    }
    //*****************************************END DISPLAY SELECTED
CLUB********************************************//
```

```
    //*******************************************DISPLAY SELECTED MATCH*********************************************//
    /**
     * Method implementation of displaySelectedMatch(), overrun from the LeagueManager interface
     * This method handles the functionality of displaying statistics of a selected match in the Premier League
     * Identifies based on the club name inputs that are involved in the match
     * @param firstTeamInput - club 1/2
     * @param secondTeamInput - club 1/2
     * @return  - the found match, if found, else returns null
     */
    @Override
    public FootballMatch displaySelectedMatch(String firstTeamInput, String secondTeamInput) {
        FootballMatch foundFootballMatch = null;
        for (FootballMatch footballMatch: allMatches) {
            //*if condition that checks both possibilities, so that if user enters football club 2 as the first input*//
            //*and football club 1 as the second output, the same match is considered*//
            if ((footballMatch.getFirstTeam().getClubName().equals(firstTeamInput) &&
                    footballMatch.getSecondTeam().getClubName().equals(secondTeamInput)) ||
                    (footballMatch.getSecondTeam().getClubName().equals(firstTeamInput) &&
                            footballMatch.getFirstTeam().getClubName().equals(secondTeamInput))) {
                foundFootballMatch = footballMatch;
                break;
            }
        }

        return foundFootballMatch;
    }
    //******************************************END DISPLAY SELECTED
MATCH*******************************************//




    //*************************************ADD PLAYED MATCH BETWEEN TWO
CLUBS*****************************************//
    /**
     * Method implementation of addPlayedMatch(), overrun from the LeagueManager interface
     * This method handles the functionality of playing a manually entered match in the
PremierLeague
     * @param season - user season input
     * @param date - user date input
     * @param firstTeamInput - club name of first/second team
     * @param secondTeamInput - club name of second/first team
     * @param firstTeamScore - score of first/second team
     * @param secondTeamScore - score of second/first team
     */
    @Override
    public void addPlayedMatch(String season, String date, String firstTeamInput, String
secondTeamInput, int firstTeamScore, int secondTeamScore) {
        //*Manipulate and create a valid date object*//
        date = date.replaceAll("-", "/");
        date = date + "/" + season;
        LocalDate validDate = LocalDate.parse(date,
DateTimeFormatter.ofPattern("dd/MM/yyyy"));

        FootballClub firstTeam = null;
        FootballClub secondTeam = null;
        //*based on club name input, we assign the FootballClub to a football club object, so that a
match can be played*//
        for (FootballClub footballClub : allFootballClubs) {
            if (footballClub.getClubName().equals(firstTeamInput)) {
                firstTeam = footballClub;
                break;
            }
        }
        for (FootballClub footballClub : allFootballClubs) {
            if (footballClub.getClubName().equals(secondTeamInput)) {
                secondTeam = footballClub;
                break;
            }
        }

        FootballMatch match = new FootballMatch(firstTeam, secondTeam, validDate);
        boolean matchPlayed = false;
        //*Check to see whether the match specified has already been played*//
        for (FootballMatch footballMatch : allMatches) {
            if (match.equals(footballMatch)) {
                matchPlayed = true;
                break;
            }
        }

        //*if the match is unique, play it*//
        if (!matchPlayed) {
            System.out.println("Now playing match between " +
match.getFirstTeam().getClubName() + " and " +
                match.getSecondTeam().getClubName()
            );
            match.playMatch(firstTeamScore, secondTeamScore);
            allMatches.add(match);
        } else {
            System.out.println("[ERROR] ==> Match between " + firstTeamInput + " and " +
secondTeamInput + " has already been played!");
        }
    }
    //**************************************END ADD PLAYED MATCH BETWEEN
TWO CLUBS**************************************//



    //**********************************ADD RANDOM PLAYED MATCH BETWEEN
TWO CLUBS**************************************//
    /**
     * Method implementation of addPlayedMatchRandom(), overrun from the LeagueManager
interface
     * This method handles the functionality of playing a random match in the PremierLeague
     * @param season - user input season
     * @return footballMatch - the football match that was played (Used for GUI frontend)
     */
    @Override
    public FootballMatch addPlayedMatchRandom(String season) {
        FootballClub firstTeam;
        FootballClub secondTeam;

        //*this if condition checks whether there are enough teams to play a match in the first
place*//
```

```
        if (allFootballClubs.size() < 2) {
            System.out.println("[ERROR] ==> There isn't enough teams to play a match!");
            return null;
        }

        //*We check to make sure that all the matches playable have already been played or not*//
        boolean allMatchesPlayed = validatePlayableMatches(season);
        if (allMatchesPlayed) {
            System.out.println("[ERROR] ==> All Possible Matches have already been played!");
            return null;
        }

        //*loop till two unique teams are selected (two same teams cannot play against each
other)*//
        do {
            firstTeam =
allFootballClubs.get(PremierLeagueManager.RANDOM.nextInt(allFootballClubs.size()));
            secondTeam =
allFootballClubs.get(PremierLeagueManager.RANDOM.nextInt(allFootballClubs.size()));
        } while (firstTeam.getClubName().equals(secondTeam.getClubName()));



        FootballMatch footballMatch;
        boolean hasMatch;
        //*the logic here is to loop infinitely, till a unique match has been generated*//
        while (true){
            LocalDate localDate = generateRandomDate(season);
            hasMatch = false;
            footballMatch = new FootballMatch(firstTeam, secondTeam, localDate);
            //*check the above generated match against all the matches already played*//
            for (FootballMatch match : allMatches) {
                //*overridden equals() method*//
                if (match.equals(footballMatch)) {
                    hasMatch = true;
                    break;
                }
            }

            //*if and only if the hasMatch flag had not turned true (the generated match is unique)*//
            //*the match is played and added into the list of matches, and the entire while loop is
broken*//
            //*since its reason has been fulfilled*//
            if(!hasMatch) {
                System.out.println("Now playing match between " +
footballMatch.getFirstTeam().getClubName() + " and " +
                    footballMatch.getSecondTeam().getClubName()
                );
                footballMatch.playMatchRandom();
                allMatches.add(footballMatch);
                return footballMatch;
            }

            //*if the program has reached this statement, it means that the generated random match
has already been*//
            //*played, but there are more legal matches that can be generated that can be played*//
            //*if it was the case that the match generated was already played (hasMatch turned true
during for loop)*//
```

```
            //*another two teams are picked at random, and the while loop continues from the
beginning*//
            do {
                firstTeam =
allFootballClubs.get(PremierLeagueManager.RANDOM.nextInt(allFootballClubs.size()));
                secondTeam =
allFootballClubs.get(PremierLeagueManager.RANDOM.nextInt(allFootballClubs.size()));
            } while (firstTeam.getClubName().equals(secondTeam.getClubName()));
        }
    }
    /**
     * Helper method that is used to check whether all matches have been played or not. Public
cuz MatchController
     * Needs access to this as well
     * @param season - user season input
     * @return - t/f, whether all matches have been played
     */
    public static boolean validatePlayableMatches(String season) {
        //*ensure whether all matches have already been played this is done by generating every
possible match of*//
        //*a team, and checking whether the list of matches consists of that match, if and only if the
list does*//
        //*contain all possible matches, no more matches can be played*//
        boolean allMatchesPlayed = true;

        outerMostLoop:
        for(FootballClub eachClub : allFootballClubs) {
            for(FootballClub otherClub : allFootballClubs) {
                if (eachClub.getClubName().equals(otherClub.getClubName())) {
                    continue;
                }
                FootballMatch checkMatch = new FootballMatch(eachClub, otherClub,
generateRandomDate(season));
//                System.out.println(eachClub.getClubName() + " " + otherClub.getClubName());


//                for(FootballMatch match : allMatches) {
//                    if (checkMatch.equals(match)) {
//                        allMatchesPlayed = true;
//                        break ;
//                    }
//                }
                allMatchesPlayed = allMatches.contains(checkMatch);
//                System.out.println(allMatchesPlayed);


                if (!allMatchesPlayed) {
                    break outerMostLoop;
                }
            }
        }
        return allMatchesPlayed;
    }
    /**
     * Private helper method that generates a random date
     * If generated value is less than 10, appends a 0 at the beginning so that it can be parsed into a
LocalDate
     * @param season - takes season parameter
     * @return - a LocalDate object containing the random date created
     */
    private static LocalDate generateRandomDate(String season) {
        String randomDay = String.valueOf(RANDOM.nextInt((31)) + 1);
        if (Integer.parseInt(randomDay) < 10) randomDay = "0" + randomDay;
        String randomMonth = String.valueOf(RANDOM.nextInt((12)) + 1);
        if (Integer.parseInt(randomMonth) < 10) randomMonth = "0" + randomMonth;
        String dateString = randomDay + "/" + randomMonth + "/" + season;
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        return LocalDate.parse(dateString, formatter);
    }
    //*********************************END ADD RANDOM PLAYED MATCH
BETWEEN TWO CLUBS***********************************//




    //*******************************************DISPLAY POINTS
TABLE**********************************************//
    /**
     * Method implementation of displayPointsTable(), overrun from the LeagueManager
interface
     * This method handles the functionality of displaying the standings of the Premier League
     */
    @Override
    public void displayPointsTable() {
        //*sort based on the GoalDifferenceComparator created, in descending order*//
        allFootballClubs.sort(new GoalDifferenceComparator().reversed());
        //*then use the compareTo() method of FootballClub to sort based on points
        //*(GD sorted first so the GD order is maintained)*//
        allFootballClubs.sort(Collections.reverseOrder());
```

```
System.out.println("==========================================================
=================");
        System.out.println("PREMIER LEAGUE STANDINGS");

System.out.println("==========================================================
=================");
        System.out.format("%-20s %-5s %-5s %-5s %-5s %-5s %-5s %-5s %-5s", "Club", "MP",
"W", "D", "L", "GF", "GA",
                   "GD", "Pts");
        System.out.println();
        //*A simple formatted string is used to print*//
        for (FootballClub footballClub : allFootballClubs) {
            System.out.format("%-20s %-5s %-5s %-5s %-5s %-5s %-5s %-5s %-5s",
footballClub.getClubName(),
                        footballClub.getFootballClubTotalStatistics().getMatchesPlayed(),
                        footballClub.getFootballClubTotalStatistics().getWins(),
                        footballClub.getFootballClubTotalStatistics().getDraws(),
                        footballClub.getFootballClubTotalStatistics().getDefeats(),
                        footballClub.getFootballClubTotalStatistics().getGoalsFor(),
                        footballClub.getFootballClubTotalStatistics().getGoalsAgainst(),
                        footballClub.getFootballClubTotalStatistics().getGoalDifference(),
                        footballClub.getFootballClubTotalStatistics().getPoints());
            System.out.println();
        }

System.out.println("==========================================================
=================");
    }
    //*****************************************END DISPLAY POINTS
TABLE**********************************************//
```

```
    //**********************************************DISPLAY MATCH
SCORES************************************************//

    /**

     * Method implementation of displayMatchResults(), overrun from the LeagueManager
interface

     * This method handles the functionality of displaying all the match results in the Premier
League

     */

    @Override

    public void displayMatchResults() {

        //*FootballMatch compareTo() method is used to sort (Date sorting), in descending order*//

        //*So it's ordered from the most recent to the least*//

        allMatches.sort(Collections.reverseOrder());

        if (allMatches.size()>0) {

            System.out.println("============================================");

            System.out.format("%7s %5s", "", "PREMIER LEAGUE - ALL MATCHES");

            System.out.println();

            System.out.println("============================================");


            for (FootballMatch footballMatch : allMatches) {

                System.out.format("%17s %5s", "", footballMatch.getMatchDate());

                System.out.println();

                System.out.format("%-19s %2s %2s %19s",
footballMatch.getFirstTeam().getClubName(),

                        footballMatch.getFirstTeamSingleMatchStats().getGoals(),

                        footballMatch.getSecondTeamSingleMatchStats().getGoals(),

                        footballMatch.getSecondTeam().getClubName()
                );
                System.out.println();
                System.out.println("------------------------------------------");
            }
        } else {
            System.out.println("[ERROR] ==> No matches have been played yet!");
        }
    }
    //****************************************END DISPLAY MATCH
SCORES*********************************************//




//************************************************SAVE
DATA****************************************************//
    /**
     * Method implementation of saveData(), overrun from the LeagueManager interface
     * This method handles the functionality of saving all required data
     */
    @Override
    public void saveData(String season) {
        //*list of type Object, to store both the Clubs and Matches*//
        List<Object> allData = new ArrayList<>();
        allData.add(allFootballClubs);
        allData.add(allMatches);

        try (FileOutputStream fileOutputStream = new FileOutputStream(new File(SAVE_PATH +
"\\" + season + ".txt"));
         ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream)) {
```

```
            objectOutputStream.writeObject(allData);
            System.out.println("Data saved successfully!");
        } catch (Exception e) {
            System.out.println("[ERROR] ==> Something went wrong while saving the file! " +
e.getMessage());
        }
    }
    //*********************************************END SAVE
DATA*****************************************************//




    //***********************************************LOAD
DATA*****************************************************//
    /**
     * Method implementation of loadData(), overrun from the LeagueManager interface
     * This method handles the functionality of loading all the data that had been saved
     */
    @Override
    public void loadData(String season) {
        List<Object> allData;

        try (FileInputStream fileInputStream = new FileInputStream(new File(SAVE_PATH + "\\"
+ season + ".txt"));
             ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream)) {
            System.out.println(season + ".txt found!");
            allData = (List<Object>) objectInputStream.readObject();
            //*load based on the index it was saved (allData is a list of two lists)*//
            allFootballClubs = (List<FootballClub>) allData.get(0);
            allMatches = (List<FootballMatch>) allData.get(1);
            System.out.println("Data loaded successfully!");


        } catch (FileNotFoundException ex) {
            System.out.println("[ERROR] ==> There weren't any files to load!");
        } catch (Exception e) {
            System.out.println("[ERROR] ==> Something went wrong in loading the file! " +
e.getMessage());
        }
    }
    //*********************************************END LOAD
DATA**************************************************//


    //***************************GETTER AND SETTER METHODS TO ACCESS
THROUGHOUT THE PROJECT*************************//
    /**
     * sets the season choice
     * This is needed for the Play Controllers to get access to the season of choice
     */
    public static void setSeasonFile(String season) {
        try (FileWriter file = new FileWriter(SEASON_INPUT + "\\season.txt")) {
            file.write(season);
        } catch (Exception e) {
            System.out.println("[ERROR] ==> Something went wrong!");
        }
    }


    /**
     * @return list containing all the matches
     */
    public static List<FootballMatch> getAllMatches() {
        return allMatches;
    }

    /**
     * @return list containing all the football clubs
     */
    public static List<FootballClub> getAllFootballClubs() { return allFootballClubs; }

    /**
     * @return max number of clubs in premier league
     */
    public static int getMaxSize() {
        return MAX_SIZE;
    }
}
```

2.4 Utils

GoalDifferenceComparator

```
package coursework.utils;

/*

 * GoalDifferenceComparator

 * Copyright © 2020 Ammar Raneez. All Rights Reserved.

 */

import coursework.models.FootballClub;

import java.util.Comparator;

/**

 * GoalDifferenceComparator class, which will be used to sort the list of football clubs based on
GD

 * Needed for an additional sorting of FootballClubs that have the same amount of points

 * @version 1.x November 11th 2020

 * @author Ammar Raneez | 2019163 | W1761196

 */

public class GoalDifferenceComparator implements Comparator<FootballClub> {

    /**

     * overridden compare method, that can compare any two football clubs based on any attribute

     * @param footballClub1 - first football club

     * @param footballClub2 - second football club

     * @return - positive value if club1 GD > club2 GD. Negative if vice-versa. And 0 if equal

     */

    @Override

    public int compare(FootballClub footballClub1, FootballClub footballClub2) {

        return footballClub1.getFootballClubTotalStatistics().getGoalDifference() -

                footballClub2.getFootballClubTotalStatistics().getGoalDifference();

    }

}
```

GoalsForComparator

```
package coursework.utils;
/*
 * GoalsForComparator
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import coursework.models.FootballClub;
import java.util.Comparator;


/**
 * GoalsForComparator class, which will be used to sort the list of football clubs based on goals
 * scored
 * @version 1.x November 20th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class GoalsForComparator implements Comparator<FootballClub> {
    /**
     * overridden compare method, that can compare any two football clubs based on any attribute
     * @param footballClub1 - first football club
     * @param footballClub2 - second football club
     * @return - positive value if club1 goals for > club2 goals for. Negative if vice-versa. And 0
if equal
     */
    @Override
    public int compare(FootballClub footballClub1, FootballClub footballClub2) {
        return footballClub1.getFootballClubTotalStatistics().getGoalsFor() -
                footballClub2.getFootballClubTotalStatistics().getGoalsFor();
    }
}
```

SeasonRetriever

```
package coursework.utils;


/*
 * SeasonRetriever
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;


/**
 * SeasonRetriever class, responsible for loading in the season input saved in
PremierLeagueManager
 * For Play to get access to the season
 * @version 1.x December 20th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class SeasonRetriever {
    private static final String SEASON_INPUT =
"C:\\Users\\Ammuuu\\Downloads\\learning\\UNI\\OOP-Module\\Coursework\\" +
            "OOP-COURSEWORK\\SeasonInputPlay";
    private static String season = "";


    /**
     * Retrieves and returns the season input from the file
     * @return - the season input
     */
    public static String getSeason() {
        try {
            File myObj = new File(SEASON_INPUT + "\\season.txt");
            Scanner sc = new Scanner(myObj);
            while (sc.hasNextLine()) {
                season = sc.nextLine().trim();
            }
            sc.close();
        } catch (FileNotFoundException e) {
            System.out.println("[ERROR] ==> File could not be found!");
        }
        return season;
    }
}
```

WinComparator

```
package coursework.utils;


/*
 * WinComparator
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import coursework.models.FootballClub;
import java.util.Comparator;


/**
 * WinComparator class, which will be used to sort the list of football clubs based on wins
 * @version 1.x November 20th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
public class WinComparator implements Comparator<FootballClub> {
   /**
    * overridden compare method, that can compare any two football clubs based on any attribute
    * @param footballClub1 - first football club
    * @param footballClub2 - second football club
    * @return - positive value if club1 wins > club2 wins. Negative if vice-versa. And 0 if equal
    */
   @Override
   public int compare(FootballClub footballClub1, FootballClub footballClub2) {
      return footballClub1.getFootballClubTotalStatistics().getWins() -
            footballClub2.getFootballClubTotalStatistics().getWins();
   }
}
```

2.5 Conf

Routes file

```
# Routes

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

# ~~~~


#All Getter Methods

#Get method for the inital display will render the index.html file of Angular

GET     /                           coursework.controllers.FrontendController.index()

GET     /pointstable                coursework.controllers.ClubController.returnAllClubs

GET     /pointstable/winfilter      coursework.controllers.ClubController.winSortClubFilter

GET     /pointstable/goalfilter     coursework.controllers.ClubController.goalSortClubFilter

GET     /pointstable/playmatch      coursework.controllers.MatchController.playMatch

#Accepts in url parameters

GET     /clubs/*clubName
coursework.controllers.ClubController.returnSelectedClub(clubName : String)

GET     /allmatches/*arrIndex
coursework.controllers.MatchController.returnSelectedMatch(arrIndex : Integer)

GET     /allmatches                 coursework.controllers.MatchController.returnAllMatches

GET     /specificMatches/*date
coursework.controllers.MatchController.returnMatchesOnDate(date : String)

# Map static resources from the /public folder to the /assets URL path

GET     /assets/*file               controllers.Assets.at(path="/public", file)
```

2.6 UI

Index.html

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <title>Premier League</title>

    <base href="/">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="icon" type="image/x-icon" href="favicon.ico">

    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>

  </head>

  <body>

    <!--  Render AppComponent  -->

    <app-root></app-root>

  </body>

</html>
```

app.component.html

```
<!--Navbar will be present for any page, rest of the content will change according to the Routing-
->

<!--Effectively creating a SPA-->

<app-navbar></app-navbar>

<div class="container">

    <router-outlet></router-outlet>

</div>
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  //*Tag that will render the template, that is styled with the styleUrls*//
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title : string = 'Premier League';
}
```

app.constants.ts

```
//*Play API URL*//
export const BASE_URL = "http://localhost:9000";
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NavbarComponent } from './components/navbar/navbar.component';
import { PointsTableComponent } from './components/points-table/points-table.component';
import { HttpClientModule } from '@angular/common/http';
import { AllMatchesComponent } from './components/all-matches/all-matches.component';
import { SelectedMatchComponent } from './components/selected-match/selected-
match.component';
import { FormsModule } from '@angular/forms';
import { HomePageComponent } from './components/home-page/home-page.component';
import { NgxSpinnerModule } from 'ngx-spinner';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { AllClubsComponent } from './components/all-clubs/all-clubs.component';
import { SelectedClubComponent } from './components/selected-club/selected-club.component';


@NgModule({
  //*All used components*//
  declarations: [
    AppComponent,
    NavbarComponent,
    PointsTableComponent,
    AllMatchesComponent,
    SelectedMatchComponent,
    HomePageComponent,
    AllClubsComponent,
    SelectedClubComponent
  ],
  //*All used Modules*//
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    NgxSpinnerModule,
    BrowserAnimationsModule
  ],
  providers: [],
  //*Bootstrap AppComponent to AppModule, AppComponent holds router-outlet, which will
display everything*//
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AllClubsComponent } from './components/all-clubs/all-clubs.component';
import { AllMatchesComponent } from './components/all-matches/all-matches.component';
import { HomePageComponent } from './components/home-page/home-page.component';
import { PointsTableComponent } from './components/points-table/points-table.component';
import { SelectedClubComponent } from './components/selected-club/selected-club.component';
import { SelectedMatchComponent } from './components/selected-match/selected-
match.component';

//*Define all routes navigable*//
const routes: Routes = [
  //* path '/' *//
  {
    path: "",
    component: HomePageComponent
  },
  //* path '/clubs' *//
  {
    path: "clubs",
    component: AllClubsComponent
  },
  //* path '/clubs/clubParam' *//
  {
    path: "clubs/:clubName",
    component: SelectedClubComponent
  },
  //* path '/standings' *//
  {
    path: "standings",
    component: PointsTableComponent
  },
  //* path '/matches' *//
  {
    path: "matches",
    component: AllMatchesComponent
  },
  //* path '/matches/matchParam' *//
  {
    path: "matches/:id",
    component: SelectedMatchComponent
  },
  //* wildcard, will match any url (used at the end, to avoid error 404 pages) *//
  {
    path: "**",
    component: HomePageComponent
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

2.6.1 components

2.6.1.1 all-clubs

all-clubs.component.html

```
<div class="container">
  <div class="top-header-text">
    <h1>Witness the Greatness</h1>
  </div>
  <div class="main-center-container">
    <!--   Display a spinner as a loading indication, if the array hasn't been populated yet  -->
    <ngx-spinner size="medium" *ngIf="!getAllFootballClubs()" bdColor="rgba(51,51,51,0)"
type="line-scale-pulse-out" [zIndex]="0" [fullScreen]="false">
      <p style="color : white">Loading Clubs...</p>
    </ngx-spinner>
    <!--  Display a styled div tag for each club in the array of clubs. Style the div tags based on
the specific club kit  -->
    <div [ngStyle] = "{ 'border' : '1px solid rgb(' + club.kit.topColor.red + ',' +
club.kit.topColor.green + ',' + club.kit.topColor.blue + ')' }"
      class="main-center-content" *ngFor="let club of getAllFootballClubs()" routerLink="{{
club.clubName }}"
    >
      <!--     Two way data binding,      -->
      <h1>{{ club.clubName }}</h1>
      <h3>Sponsored By</h3>
      <h3>{{ club.kit.sponsor }}</h3>
      <div class="main-center-hoverContainer"
        [ngStyle] ="{ 'background-image' : 'linear-gradient(rgba(' + club.kit.topColor.red + ',' +
club.kit.topColor.green + ',' + club.kit.topColor.blue + ', 0.2' + '), rgba(' +
club.kit.bottomColor.red + ',' + club.kit.bottomColor.green + ',' + club.kit.bottomColor.blue + ',
0.2))' }"
      >
      </div></div></div></div>
```

all-clubs.component.css

```
@import
url('https://fonts.googleapis.com/css2?family=East+Sea+Dokdo&family=Sansita+Swashed:wght
@600&display=swap');

@import url('https://fonts.googleapis.com/css2?family=Satisfy&display=swap');

.container {

    background-color: #222;

    padding: 0px 20px;

    height: 100%;

    min-height: 94vh;

    margin-top: 3%;

    animation: fadein 1s;

    padding: 70px 60px 0px 60px;

    -moz-animation: fadein 1s;

    -webkit-animation: fadein 1s;

    -o-animation: fadein 1s;

    border-left: 5px solid purple;

    border-right: 5px solid purple;

    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 1)),
url(/assets/images/PL-salah.jpg);

    background-size: cover;

}

@keyframes fadein {

    from {

        opacity:0;

        transform: translate(0, -10%)

    }

    to {

        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-o-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10px)
    }
    to {
        opacity:1;
        transform: translate(0, 0px)
    }
}
.top-header-text {
    display: flex;
    align-items: center;
    margin-bottom: 40px;
}
.top-header-text > h1 {
    font-weight: bolder;
    font-family: 'East Sea Dokdo', cursive;
    font-family: 'Sansita Swashed', cursive;
    font-family: 'Satisfy', cursive;
    font-size: 50px;
    color: white;
}
.main-center-container {
    display: flex;
    min-height: 400px;
    flex-wrap: wrap;
    align-items: center;
    justify-content: center;
    text-align: center;
}
.main-center-container .main-center-content {
    color: #62ffaf;
    position: relative;

    transition: 0.3s ease;

    width: 380px;

    height: 200px;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

    margin-bottom: 40px;

    margin-right: 20px;

    background-size: cover;

    background-repeat: no-repeat;

    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 1)),
url(/assets/images/PL-Lion-transparent.png);

}

.main-center-container .main-center-content:hover {

    transform: scale(1.1);

    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 1)),
url(/assets/images/PL-lionTextPink.jpg);

    background-size: cover;

}

.main-center-hoverContainer {

    position: absolute;

    top: 0;

    left: 0;

    right: 0;

    bottom: 0;

    width: 100%;

    height: auto;
    opacity: 0;
    text-align: center;
    transition: 0.3s ease;
}
.main-center-hoverContainer:hover {
    opacity: 1;
    cursor: pointer;
}
```

all-clubs.component.ts

```
/*
 * AllClubsComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { Component, OnInit } from '@angular/core';
import { FootballClub } from 'src/app/models/FootballClub.model';
import { AllClubsService } from 'src/app/services/all-clubs/all-clubs.service';

@Component({
  selector: 'app-all-clubs',
  templateUrl: './all-clubs.component.html',
  styleUrls: ['./all-clubs.component.css']
})

/**
 * AllClubsComponent class, which will be used to render all the clubs
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export class AllClubsComponent implements OnInit {
  //*Create an array of FootballClub objects to hold all Football clubs*//
  private allClubs : FootballClub[];

  //*create an instance of AllClubsService via Dependency Injection*//
  constructor(private allClubsService : AllClubsService) { }

  //*Once component is mounted get all the football clubs*//
  ngOnInit(): void {
    this.getFootballClubs();
  }

  /**
   * Gets all the footballClubs, by subscribing to the Observable returned in
getAllFootballClubs()
   */
  public getFootballClubs() : void {
    this.allClubsService.getAllFootballClubs().subscribe(
      response => this.handleSuccessfulResponse(response),
      error => this.handleErrorResponse(error)
    );
  }

  /**
   * Populates the array of clubs
   * @param response - successful response (holds all club data in json format)
   */
  private handleSuccessfulResponse(response : any) : void {
    console.log(response)
    this.allClubs = response;
  }
  /**
   * Populates array of clubs
   * @param error - error response
   */
  private handleErrorResponse(error : any) : void {
    this.allClubs = error.message;
    console.log(error);
  }

  /**
   * Getter method, for the template file to get access to the array of football clubs
   * @return FootballClub[] - array of football clubs
   */
  public getAllFootballClubs() : FootballClub[] {
    return this.allClubs;
  }
}
```

2.6.1.2 all-matches

all-matches.component.html

```
<div class="container">
  <div class="top-header">
    <h1 class="top-header-text">Reminisce the Highlights</h1>
    <div class="top-header-right">
      <!--   tie date attribute to input tag    -->
      <input class="top-header-right-input" type="text" placeholder="yyyy-mm-dd"
[(ngModel)] = "date" />
      <button class="top-header-right-search" (click) =
"getFootballMatchesOnDate()">Search</button>
      <button class="top-header-right-reset" (click) = "getFootballMatches()">Reset</button>
    </div>
  </div>
  <div class="main-center">
    <div class="main-center-center">
      <ngx-spinner size="medium" *ngIf="!getAllFootballMatches()"
bdColor="rgba(51,51,51,0)" type="line-scale-pulse-out" [zIndex]="0" [fullScreen]="false">
        <p style="color : white">Loading Matches...</p>
      </ngx-spinner>
      <!--   routerLink -> on click of a div, navigate to allmatches/:index, index being the
specific match    -->
      <div *ngFor="let match of getAllFootballMatches(); let i = index"  routerLink="{{ i }}"
class="main-center-center-row">
        <div class="main-center-center-row-left" >
          <h3 class="main-center-center-row-clubName">{{
match.firstTeam.clubName.toLocaleUpperCase() }}</h3>
          <h2 class="main-center-center-row-clubScore">{{
match.firstTeamSingleMatchStats.goals }}</h2>
        </div>
        <div class="main-center-center-row-center">
```

```
                <h3 class="main-center-center-row-center-date">{{ match.matchDate }}</h3>
                <h1 class="main-center-center-row-center-vs">VS</h1>
            </div>
            <div class="main-center-center-row-right">
                <h3 class="main-center-center-row-clubName">{{
match.secondTeam.clubName.toLocaleUpperCase() }}</h3>
                <h2 class="main-center-center-row-clubScore">{{
match.secondTeamSingleMatchStats.goals }}</h2>
            </div>
        </div>
    </div>
  </div>
</div>
```

all-matches.component.css

```
@import
url('https://fonts.googleapis.com/css2?family=East+Sea+Dokdo&family=Sansita+Swashed:wght
@600&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Satisfy&display=swap');
.container {
  padding: 0px 20px;
  height: 94vh;
  margin-top: 3%;
  animation: fadein 1s;
  -moz-animation: fadein 1s;
  -webkit-animation: fadein 1s;
  -o-animation: fadein 1s;
  border-left: 5px solid purple;
  border-right: 5px solid purple;
```

```
    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 1)),
url(/assets/images/PL-aguero.jpg);
    background-size: cover;
}
@keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
```

```
  to {
    opacity:1;
    transform: translate(0, 0%)
  }
}
@-o-keyframes fadein {
  from {
    opacity:0;
    transform: translate(0, -10px)
  }
  to {
    opacity:1;
    transform: translate(0, 0px)
  }
}
.top-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding-right: 60px;
  margin-top: 50px;
  margin-bottom: 40px;
  padding-top: 70px;
}
.top-header-text {
  color: white;
  font-size: 50px;
  margin-left: 50px;
```

```
    font-weight: bolder;
    font-family: 'East Sea Dokdo', cursive;
    font-family: 'Sansita Swashed', cursive;
    font-family: 'Satisfy', cursive;
}
.top-header-right > input {
    width: 300px;
    margin: 4px;
    padding: 8px;
    border: 1px solid #04f5ff;
    background-color: transparent;
    color: white;
    font-size: 16px;
}
.top-header-right > input:focus, .top-header-right > input:active {
    outline: none;
}
.top-header-right button {
    margin: 4px;
    padding: 8px;
    background-color: transparent;
    font-size: 16px;
    transition: 0.2s ease;
}
.top-header-right-search {
    color: #04f5ff;
    border: 1px solid #04f5ff;
}
```

```
.top-header-right-reset {
    color: #f00;
    border: 1px solid #f00;
}
.top-header-right-search:hover, .top-header-right-reset:hover {
    color: white;
    cursor: pointer;
}
.main-center {
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
}
.main-center-center {
    border: 1px solid #f00;
    height: 400px;
    overflow: scroll;
    display: flex;
    flex-direction: column;
    width: 1200px;
    position: relative;
}
.main-center-center-row {
    display: flex;
    justify-content: space-around;
    margin: 12px;
    transition: 0.3s ease;
    height: 80px;
}
.main-center-center-row:hover {
    transform: scale(1.1);
    border: 1px solid #62ffa7;
    cursor: pointer;
}
.main-center-center-row-left, .main-center-center-row-right, .main-center-center-row-center {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
.main-center-center-row-left, .main-center-center-row-right {
    width: 300px;
}
.main-center-center-row-center {
    width: 100px;
}
.main-center-center-row-clubName {
    font-weight: 600;
    color: white;
    text-shadow: 1.5px 0px #62ffa7;
    font-size: 18px;
}
.main-center-center-row-clubScore {
    font-weight: 600;
    color: #04f5ff;
    font-size: 30px;
}
.main-center-center-row-center-date {
    color: white;
}
.main-center-center-row-center-vs {
    color: #555;
    font-size: 40px;
    font-weight: 900;
}
```

all-matches.component.ts

```
/*
 * AllMatchesComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { Component, OnInit } from '@angular/core';
import { FootballMatch } from 'src/app/models/FootballMatch.model';
import { AllMatchesService } from 'src/app/services/all-matches/all-matches.service';
import { MatchesOnDateService } from 'src/app/services/matches-on-date/matches-on-
date.service';
import Swal from 'sweetalert2';
/**
 * AllMatchesComponent class, which will be used to render all the matches
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({ selector: 'app-all-matches',
  templateUrl: './all-matches.component.html',  styleUrls: ['./all-matches.component.css'], })
export class AllMatchesComponent implements OnInit {
  //*Regex to validate date input*//
  private dateRegex : RegExp = /^\d{4}-\d{2}-\d{2}$/;
  //*Array of FootballMatch objects to hold all matches*//
  private allMatches : FootballMatch[];
  public date : string;
  constructor(private allMatchesService : AllMatchesService, private matchesByDateService :
MatchesOnDateService) { }
  ngOnInit(): void {    this.getFootballMatches();  }
  /**
   * Gets the matches that have been played the same date as that has been input
   */
  public getFootballMatchesOnDate() : void {
    if(this.date.match(this.dateRegex)) {
      this.matchesByDateService.getMatchesOnDate(this.date).subscribe(
        response => this.handleSuccessfulResponse(response),
        error => this.handleErrorResponse(error)
      );
    } else {
      Swal.fire({
        title: '▯▯▯▯',
        imageUrl: "/assets/images/PL-lionAlert.png",
        imageHeight: 200,
        imageWidth: 200,
        text: 'Please Specify a Date with Format yyyy-mm-dd',
        showClass: {
          popup: 'animate__animated animate__fadeInDown'
        },
        hideClass: {
          popup: 'animate__animated animate__fadeOutUp'
        }
      })
    }
  }
  /**
   * Gets all football matches, subscribes to the observable returned by getAllFootballMatches()
   */
  public getFootballMatches() : void {
    //*Resets the date input to blank*//
    this.date = "";
    this.allMatchesService.getAllFootballMatches().subscribe(
      response => this.handleSuccessfulResponse(response),
      error => this.handleErrorResponse(error)
    );
  }
  /**
   * Populates the allMatches array. Successful response
   * @param response - will hold the matches in json format
   */
  private handleSuccessfulResponse(response : any) : void {
    this.allMatches = response;
    if (this.allMatches.length == 0) {
      Swal.fire({
        title: '🧐🧐🧐',
        imageUrl: "/assets/images/PL-lionAlert.png",
        imageHeight: 200,
        imageWidth: 200,
        text: 'Hmm... There appears to be no matches played at this date',
        showClass: {
          popup: 'animate__animated animate__fadeInDown'
        },
        hideClass: {
          popup: 'animate__animated animate__fadeOutUp'
        }
      })
    }
    console.log(response);
  }
  /**
   * Populates the allMatches array. Error response
   * @param error - will hold an error message
   */
  private handleErrorResponse(error : any) : void {
    this.allMatches = error.message;
    console.log(error);
  }
  /**
   * Getter method to return the array of FootballMatches
   * @return FootballMatch[] - array of football matches
   */
  public getAllFootballMatches() : FootballMatch[] {
    return this.allMatches;
  }}
```

2.6.1.3 home-page

home-page.component.html

```
<!DOCTYPE html>

<div class="container">

    <div class="top-header">

        <img id="backToTop" src="assets/images/PL-LionText-transparent.png"
alt="transparentLionText"/>

    </div>

    <div class="main-center">

        <a href="https://www.premierleague.com/nike-ball-hub" target="_blank">

            <div class="main-center-row">

                <div class="main-center-row-lower">

                    <div class="main-center-row-lower-left">

                        <img src="assets/images/PL-ball.png" alt="ball" />

                    </div>

                    <div class="main-center-row-lower-right">

                        <div class="main-center-row-upper">

                            <h1>NIKE FLIGHT</h1>

                        </div>

                        <h1>

                            Angular Chevrons nod to the speed and accuracy of Premier League gameplay.
The use of attention-grabbing graphics like the ‘hazard tape’, in combination with the contrasting
hi-viz colours, give the ball maximum visibility for the winter months.

                            Iridescent graphics call out technical details of the ball and the development of
its Aerosculpt technology. The execution is inspired by precise in-flight instrumentation,
perfectly in-keeping with the precision flight the ball allows players to achieve.

                        </h1>

                    </div>

                </div>

            </div>
    </a>
    <a href="https://www.premierleague.com/news/1608983" target="_blank">
      <div class="main-center-row">
        <div class="main-center-row-lower">
          <div class="main-center-row-lower-left">
            <img src="assets/images/PL-trophy.png" alt="trophy" />
          </div>
          <div class="main-center-row-lower-right">
            <div class="main-center-row-upper">
              <h1>TROPHY</h1>
            </div>
            <h1>
              The Trophies were cast by Asprey London, the Crown Jewellers, with the main
bodies made from solid sterling silver. The crowns are cast from 24-carat silver gilt. The bases
are made from malachite, a semi-precious stone found in Africa, with the green colour of the
opaque stone representing the field of play.
              The design of the Trophy is based on the theme of "The Three Lions of English
Football". Two of the lions are above the handles on either side. When the captain of the title-
winning team raises the Trophy, and its gold crown, above his head at the end of the season, he
becomes the third lion.
            </h1>
          </div>
        </div>
      </div>
    </a>
    <a href="https://www.premierleague.com/news/1608983" target="_blank">
      <div class="main-center-row">
        <div class="main-center-row-lower">
          <div class="main-center-row-lower-left">
            <img src="assets/images/PL-boot.png" alt="boot" />
```

```
                    </div>
                    <div class="main-center-row-lower-right">
                        <div class="main-center-row-upper">
                            <h1>GOLDEN BOOT</h1>
                        </div>
                        <h1>
                            The award has been presented since the start of the Premier League, in
1992/93. Between 1994 and 2001 it was sponsored by Carling before Barclaycard took over until
2005.
                            Until 2016 the award was sponsored by the competition's title sponsor,
Barclays, before in 2017 it was presented by Cadbury, the official snack partner of the Premier
League.
                            In the event of joint top scorers, the accolade is shared, with awards for each
player.
                        </h1>
                    </div>
                </div>
            </div>
        </a>
    </div>


    <h1 class="main-center-lowerSectionTopic">A Trip Down Memory Lane...</h1>
    <div class="main-center-lowerSection">
        <div class="main-center-lowerSectionContainer">
            <a target="_blank"
href="https://en.wikipedia.org/wiki/2019%E2%80%9320_Premier_League">
                <div class="main-center-lowerSectionI">
                    <img src="assets/images/PL-2019-20.jpg" alt="winners" />
                    <div style="background-image: url('assets/images/PL-liverpool.jpg');" class="main-
center-lowerSectionIO">
                        <div class="main-center-lowerSectionIOT">
                <h1>Liverpool</h1>
                <h3>Winners <br>2019/20</h3>
              </div>
            </div>
          </div>
        </a>
        <a target="_blank"
href="https://en.wikipedia.org/wiki/2018%E2%80%9319_Premier_League">
          <div class="main-center-lowerSectionI">
            <img src="assets/images/PL-2018-19.jpg" alt="winners" />
            <div style="background-image: url('assets/images/PL-mancity.jpg');" class="main-
center-lowerSectionIO">
              <div class="main-center-lowerSectionIOT">
                <h1>Manchester City</h1>
                <h3>Winners <br>2018/19</h3>
              </div>
            </div>
          </div>
        </a>
        <a target="_blank"
href="https://en.wikipedia.org/wiki/2017%E2%80%9318_Premier_League">
          <div class="main-center-lowerSectionI">
            <img src="assets/images/PL-2017-18.jpg" alt="winners" />
            <div style="background-image: url('assets/images/PL-mancity.jpg');" class="main-
center-lowerSectionIO">
              <div class="main-center-lowerSectionIOT">
                <h1>Manchester City</h1>
                <h3>Winners <br>2017/18</h3>
              </div>
            </div>
          </div>

        </a>

        <a target="_blank"
href="https://en.wikipedia.org/wiki/2016%E2%80%9317_Premier_League">

          <div class="main-center-lowerSectionI">

            <img src="assets/images/PL-2016-17.jpg" alt="winners" />

            <div style="background-image: url('assets/images/PL-chelsea.jpg');" class="main-
center-lowerSectionIO">

              <div class="main-center-lowerSectionIOT">

                <h1>Chelsea</h1>

                <h3>Winners <br>2016/17</h3>

              </div>

            </div>

          </div>

        </a>

        <a target="_blank"
href="https://en.wikipedia.org/wiki/2015%E2%80%9316_Premier_League">

          <div class="main-center-lowerSectionI">

            <img src="assets/images/PL-2015-16.jpg" alt="winners" />

            <div style="background-image: url('assets/images/PL-leicester.png');" class="main-
center-lowerSectionIO">

              <div class="main-center-lowerSectionIOT">

                <h1>Leicester City</h1>

                <h3>Winners <br>2015/16</h3>

              </div>

            </div>

          </div>

        </a>

        <a target="_blank"
href="https://en.wikipedia.org/wiki/2014%E2%80%9315_Premier_League">
```

```
            <div class="main-center-lowerSectionI">
               <img src="assets/images/PL-2014-15.jpg" alt="winners" />
               <div style="background-image: url('assets/images/PL-chelsea.jpg');" class="main-
center-lowerSectionIO">
                  <div class="main-center-lowerSectionIOT">
                     <h1>Chelsea</h1>
                     <h3>Winners <br>2014/15</h3>
                  </div></div></div></a></div></div>
   <a href="#backToTop"><button class="backToTopButton">Back to Top ></button></a>
</div>
```

home-page.component.css

```
.container {
   background-color: #222;
   border: 5px solid purple;
   height: 100%;
   padding: 8px;
   margin-top: 3%;
   width: 100%;
   background-image: url(/assets/images/background.png);
   background-size: contain;
   background-repeat: no-repeat;
}
.container {
   font-size: 21px;
   text-align: center;
   animation: fadein 1s;
   -moz-animation: fadein 1s;
```

```
    -webkit-animation: fadein 1s;
    -o-animation: fadein 1s;
}
@keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {opacity:1; transform: translate(0, 0%)}
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-o-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10px)
    }
    to {
        opacity:1;
        transform: translate(0, 0px)
    }
}
a {
    text-decoration: none;
}
.top-header {
    display: flex;
    justify-content: center;
    align-items: center;
}
.top-header > img {
    height: 250px;
}
.main-center {
    display: flex;
    flex-direction: column;
    padding: 0px 50px 50px 50px;
}
```

```
.main-center > a, .main-center-row, .main-center-row-upper, .main-center-row-upper > h1,
.main-center-row-lower,

.main-center-row-lower-left, .main-center-row-lower-left > img, .main-center-row-lower-right,

.main-center-row-lower-right > h1

{

   background: transparent;

}

.main-center-row {

   display: flex;

   position: relative;

   align-items: center;

   flex-direction: column;

   justify-content: space-between;

   padding: 10px;

   border-radius: 30px;

   transition: 0.3s ease;

         background-size:auto;

         animation-timing-function: ease;

   animation: image infinite 20s;

   margin-bottom: 16px;

}

@keyframes image {

   100% {

                  background-position: 0%;

         }

   50% {

                  background-position: 100%;

         }

         0% {
                background-position: 0%;
        }
}
.main-center-row:hover {
   transform: scale(1.02);
   border: 2px solid #ff2882;
   cursor: pointer;
}
.main-center-row-upper {
   width: 100%;
}
.main-center-row-upper > h1 {
   color: white;
   display: flex;
   justify-content: flex-start;
}
.main-center-lowerSectionTopic {
   text-align: left;
   color: white;
   margin-left: 35px;
   margin-bottom: 10px;
   font-family: 'Oxanium';
}
.main-center-row-lower {
   display: flex;
   justify-content: center;
   align-items: center;
}
```

```
.main-center-row-lower-right {
   padding: 10px 16px;
}
.main-center-row-lower-left > img {
   height: 100%;
   width: 300px;
   padding: 12px;
   object-fit: contain;
   transition: 0.2s ease;
}
.main-center-row-lower-left > img:hover {
   transform: scale(1.1);
}
.main-center-row-lower-right > h1 {
   font-size: 20px;
   color: #62ffaf;
   font-family: 'Roboto';
   text-align: justify;
}
.main-center-lowerSection {
   margin-left: 16px;
   margin-right: 16px;
}
.main-center-lowerSectionContainer {
   display: flex;
   overflow-y: hidden;
   overflow-x: scroll;
   padding: 10px 10px 0 0;
}
.main-center-lowerSectionContainer::-webkit-scrollbar {
   display: none;
}
.main-center-lowerSectionI {
   position: relative;
   margin-bottom: 40px;
   margin-right: 20px;
   margin-left: 10px;
   transition: 0.3s ease;
}
.main-center-lowerSectionI img {
   width: 400px;
   height: 225px;
   object-fit: cover;
}
.main-center-lowerSectionIO {
   position: absolute;
   top: 0;
   left: 0;
   right: 0;
   bottom: 0;
   width: 100%;
   height: auto;
   background-size: contain;
   opacity: 0;
   transition: 0.3s ease;
}
```

```
.main-center-lowerSectionIOT {
    position: absolute;
    color: white;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
.main-center-lowerSectionI:hover {
    transform: scale(1.1);
}
.main-center-lowerSectionIO:hover {
    border: 2px solid purple;
    opacity: 1;
    cursor: pointer;
}
.backToTopButton {
    display: flex;
    justify-content: left;
    padding: 5px 15px 5px 15px;
    font-size: 20px;
    outline: none;
    color: #ffffff;
    font-weight: bold;
    border: 1px solid #ff2882;
    background-color: #222;
    transition: 0.2s ease;
}
.backToTopButton:hover {
    background-color: #ff2882;
    border-color: #ff2882;
    cursor: pointer;
    transform: scaleX(1.05);
}
```

home-page.component.ts

```
/*
 * HomePageComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { Component, OnInit } from '@angular/core';
/**
 * HomePageComponent class, which will be used to render the home page
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({
  selector: 'app-home-page',
  templateUrl: './home-page.component.html',
  styleUrls: ['./home-page.component.css']
})
export class HomePageComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

2.6.1.4 navbar

navbar.component.html

```
<div class="topnav" id="myTopnav">

   <div class="logo">

      <img src="/assets/images/PL-Lion-transparent.png" alt="plLionTransparent" />

   </div>

   <!-- Routing provided through navbar. exact:true passed for routes that should be matched
exactly -->

   <!-- clubs and matches don't take this cuz they will accept url parameters -->

   <a routerLink="/" routerLinkActive="active"
[routerLinkActiveOptions]="{exact:true}">Home</a>

   <a routerLink="/clubs" routerLinkActive="active">Clubs</a>

   <a routerLink="/standings" routerLinkActive="active"
[routerLinkActiveOptions]="{exact:true}">Standings</a>

   <a routerLink="/matches" routerLinkActive="active">Matches</a>

</div>
```

navbar.component.css

```
.topnav {

   background-color: #222;

   overflow: hidden;

   position: fixed;

   top: 0;

   width: 100%;

   z-index: 9999;

}


.topnav a {

   float: left;
```

```
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
    transition: 0.2s ease;
    font-weight: bold;
}

.topnav a:hover {
    color: black;
    color: #ff2882;
}
.topnav a.active {
    color: #ff2882;
    font-weight: bolder;
}
.topnav .logo {
    display: flex;
    justify-content: center;
    align-items: center;
    float: left;
}
.topnav img {
    height: 50px;
    object-fit: contain;
}
```

navbar.component.ts

```
/*
 * NavbarComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { Component, OnInit } from '@angular/core';

/**
 * NavbarComponent class, which will be used to render the navbar
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {
  constructor() { }

  ngOnInit(): void {
  }
}
```

2.6.1.5 points-table

points-table.component.html

```
<div class="container">

    <div class="pane">

        <ngx-spinner size="medium" *ngIf="!getAllFootballClubs()" bdColor="rgba(51,51,51,0)"
type="line-scale-pulse-out" [zIndex]="0" [fullScreen]="false">

            <p style="color : white">Loading Standings...</p>

        </ngx-spinner>

        <!--   Render table only if the clubs have been populated     -->

        <table *ngIf="getAllFootballClubs()" class="table">

            <tr>

                <th class="longSpan">Club</th>

                <th>MP</th>

                <th>W</th>

                <th>D</th>

                <th>L</th>

                <th>GF</th>

                <th>GA</th>

                <th>GD</th>

                <th>Pts</th>

            </tr>

            <!--    loop over the array of clubs and render each as a table row        -->

            <tr *ngFor="let club of getAllFootballClubs()">

                <td class="longSpan">{{ club.clubName }}</td>

                <td>{{ club.footballClubTotalStatistics.matchesPlayed }}</td>

                <td>{{ club.footballClubTotalStatistics.wins }}</td>

                <td>{{ club.footballClubTotalStatistics.draws }}</td>

                <td>{{ club.footballClubTotalStatistics.defeats }}</td>

                <td>{{ club.footballClubTotalStatistics.goalsFor }}</td>
                <td>{{ club.footballClubTotalStatistics.goalsAgainst }}</td>
                <td>{{ club.footballClubTotalStatistics.goalDifference }}</td>
                <td>{{ club.footballClubTotalStatistics.points }}</td>
            </tr>
        </table>
    </div>
    <div class="lower-footer">
        <button class="lower-footer-sortBtn reset-btn" (click) = "getFootballClubs()" >R E S E
T</button>
        <button class="lower-footer-sortBtn" (click) = "getFootballClubsSortedOnWins()">W I N
&nbsp; S O R T</button>
        <img class="lower-footer-img" src="assets/images/PL-Lion-transparent.png"
alt="plLionTransparent" />
        <button class="lower-footer-sortBtn" (click) = "getFootballClubsSortedOnGoals()" >G O A
L &nbsp; S O R T</button>
        <button class="lower-footer-sortBtn play-btn" (click) = "updateFootballClubsAfterPlay()"
>P L A Y !</button>
    </div>
</div>
```

points-table.component.css

```
.container {
    height: 94vh;
    margin-top: 3%;
    animation: fadein 1s;
    -moz-animation: fadein 1s;
    -webkit-animation: fadein 1s;
    -o-animation: fadein 1s;
    border-left: 5px solid purple;
    border-right: 5px solid purple;
    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 1)),
url(/assets/images/PL-kane2.jpg);
    background-size: cover;
}
@keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-o-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10px)
    }
    to {
        opacity:1;
        transform: translate(0, 0px)
    }
}

.pane {
    display: inline-block;
    overflow-y: scroll;
    max-height:400px;
    width: 100%;
    scroll-behavior: smooth;
    position: relative;
    height: 400px;
}

.table {
    background-color: transparent;
    width: 100%;
    border-collapse: collapse;
    color: #04f5ff;
    height: 400px;
    margin: 0;
}

th {
    color: #62ffa4;
    padding: 10px;
}

td {
    padding: 10px;
}

th, td {
    width: 8%;
    text-align: center;
}

.table tr .longSpan {
    width: 28%;
    text-align: left;
    font-weight: bolder;
}
.table tr td.longSpan {
    color: #f00;
}

.table tr:nth-child(even) {
    background-color: transparent;
}



.lower-footer {
    height: 213px;
    background-color: transparent;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
}

.lower-footer-img {
    object-fit: contain;
    height: 210px;
}

.lower-footer button {
    background-color: transparent;
}

.lower-footer-sortBtn {
    color: #04f5ff;
    height: 50px;
    padding: 8px;
    width: 200px;
    border-color: #04f5ff;
    transition: 0.3s ease;
    font-size: 18px;
    outline: none;
    font-weight: bolder;

}
.lower-footer-sortBtn:hover {
    cursor: pointer;
    color: white;
    transform: scaleX(1.1);
}

.reset-btn {
    border: 1px solid #f00;
    color: #f00;
}

.play-btn {
    border: 1px solid #62ffa4;
    color: #62ffa4;
}
.play-btn:hover {
    animation: shake 0.82s cubic-bezier(.36,.07,.19,.97) both infinite;
    transform: translate3d(0, 0, 0);
    perspective: 1000px;
}

@keyframes shake {
    10%, 90% {
        transform: translate3d(-1px, 0, 0);
    }
    20%, 80% {
        transform: translate3d(2px, 0, 0);
    }
    30%, 50%, 70% {
        transform: translate3d(-4px, 0, 0);
    }
    40%, 60% {
        transform: translate3d(4px, 0, 0);
    }
}
```

Points-table.component.ts

```
/*
 * PointsTableComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { Component, OnInit } from '@angular/core';
import { FootballClub } from 'src/app/models/FootballClub.model';
import { AllClubsService } from 'src/app/services/all-clubs/all-clubs.service';
import { AllClubsFilterService } from 'src/app/services/all-clubs-filter/all-clubs-filter.service';
import { PlayMatchService } from 'src/app/services/play-match/play-match.service';
import { NgxSpinnerService } from "ngx-spinner";
import Swal from 'sweetalert2';
import {FootballMatch} from "../../models/FootballMatch.model";
/**
 * PointsTableComponent class, which will be used to render the points table view
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({ selector: 'app-points-table',
  templateUrl: './points-table.component.html',  styleUrls: ['./points-table.component.css'] })
export class PointsTableComponent implements OnInit {
  private allClubs : FootballClub[];
  private newMatch: FootballMatch;
  //*instances of required Services injected via dependency injection*//
  constructor(private allClubsService : AllClubsService, private allClubsFilterService :
AllClubsFilterService,
    private playMatchService : PlayMatchService, private ngxSpinnerService :
NgxSpinnerService) { }
  ngOnInit(): void {
    this.getFootballClubs();
    //*show a spinner until the data has been retrieved.*//
    if(this.allClubs) {
      this.ngxSpinnerService.hide();
    } else {
      this.ngxSpinnerService.show();
    }
  }
  /**
```

```
   * Updates the list of footballClubs upon clicking the play match button
   */
  public updateFootballClubsAfterPlay() : void {
    this.playMatchService.playMatch().subscribe(
      response => this.handleSuccessfulResponseAfterPlay(response),
      error => this.handleErrorResponse(error)
    )
  }
  /**
   * Handles the subscribe of Play match. Successful response (Error response is same as for other services)
   * Throws an alert, if all matches have been played, else simply updates the standings, showing an alert
   * @param response - false/the clubs with new match on whether a match has actually been played or not in the backend
   */
  private handleSuccessfulResponseAfterPlay(response : any) : void {
    console.log(response);
    if (!response) {
      Swal.fire({
        imageUrl: "/assets/images/PL-lionAlert.png",
        imageHeight: 200,
        imageWidth: 200,
        title: '❕❕❕❕',
        text: 'All Playable Matches have Already been Played!',
        showClass: {
          popup: 'animate__animated animate__fadeInDown'
        },
        hideClass: {
```

```
        popup: 'animate__animated animate__fadeOutUp'
      }
    })
  }
  else {
    this.allClubs = response[0];
    this.newMatch = response[1];
    let firingString: string = this.constructAlertBodyString();
    let headerString: string = this.constructAlertHeaderString();
    Swal.fire({
      imageUrl: "/assets/images/PL-lionAlert.png",
      imageHeight: 200,
      imageWidth: 200,
      title: headerString,
      html: firingString,
      showClass: {
        popup: 'animate__animated animate__fadeInDown'
      },
      hideClass: {
        popup: 'animate__animated animate__fadeOutUp'
      }
    });
  }
}
/**
 * Helper method of handleSuccessfulResponseAfterPlay, to construct the alert string body
 */
private constructAlertBodyString() : string {
  let firingString : string =  ``;
  firingString += `
    <div style="display: flex; justify-content: space-between">
      <div style="width: 50%">
          <h3>
              ${this.newMatch.firstTeam.clubName.toLocaleUpperCase()}
          </h3>
          <h1 style="color: #ff2882">
              ${this.newMatch.firstTeamSingleMatchStats.goals}
          </h1>
      </div>
      <div style="width: 50%">
          <h3>
              ${this.newMatch.secondTeam.clubName.toLocaleUpperCase()}
          </h3>
          <h1 style="color: #ff2882;">
              ${this.newMatch.secondTeamSingleMatchStats.goals}
          </h1>
      </div>
    </div>
  `
  return firingString;
}
/**
 * Helper method of handleSuccessfulResponseAfterPlay, to construct the alert string header
 */
private constructAlertHeaderString(): string {
  let headerString: string;
```

```
    headerString = this.newMatch.firstTeamSingleMatchStats.goals >
this.newMatch.secondTeamSingleMatchStats.goals ?

      this.newMatch.firstTeam.clubName + " won!" :
this.newMatch.secondTeamSingleMatchStats.goals >
this.newMatch.firstTeamSingleMatchStats.goals ?

        this.newMatch.secondTeam.clubName + " won!" : "This match was a draw!";

    return headerString;

  }

  /**

   * Gets the json format of the football clubs sorted based on wins in descending order

   */

  public getFootballClubsSortedOnWins() : void {

    this.allClubsFilterService.getClubsWinFilter().subscribe(

      response => this.handleSuccessfulResponse(response),

      error => this.handleErrorResponse(error)

    );

  }

  /**

   * Gets the json format of the football clubs sorted based on goals for in descending order

   */

  public getFootballClubsSortedOnGoals() : void {

    this.allClubsFilterService.getClubsGoalFilter().subscribe(

      response => this.handleSuccessfulResponse(response),

      error => this.handleErrorResponse(error)

    );

  }

  /**

   * Gets the json format of all the football clubs (Is called by ngOnInit cuz the regular array is to
be displayed first)

   */
```

```
 public getFootballClubs() : void {
   this.allClubsService.getAllFootballClubs().subscribe(
     response => this.handleSuccessfulResponse(response),
     error => this.handleErrorResponse(error)
   );
 }
 /**
  * Handles the subscribe. Successful response
  * @param response - will hold the json format of the clubs
  */
 private handleSuccessfulResponse(response : any) : void {
   this.allClubs = response;
 }
 /**
  * Handles the subscribe. Error response
  * @param error - will hold an error
  */
 private handleErrorResponse(error : any) : void {
   this.allClubs = error.message;
   console.log(error);
 }
 /**
  * Getter method for the template file to get access to the array of clubs
  */
 public getAllFootballClubs() : FootballClub[] {
   return this.allClubs;
 }
}
```

2.6.1.6 selected-club

selected-club.component.html

```
<div class="container">

   <div class="top-header-text">

      <h1>{{ getTheSelectedClub().clubName }}</h1>

      <h2 class="clubDetails">

        <!--     Render a specific message based on attributes       -->

         <span class="clubDetailsStat">{{ getTheSelectedClub().clubName }}</span>

         is worth <span class="clubDetailsStat">${{ getTheSelectedClub().clubNetWorth
}}</span>.

         It is currently being owned by <span class="clubDetailsStat">{{
getTheSelectedClub().clubOwner }}</span>

         and located at <span class="clubDetailsStat">{{ getTheSelectedClub().clubLocation
}}</span>.

         {{ getTheSelectedClub().lecturerInCharge?

            " This university club is further handled by " + getTheSelectedClub().lecturerInCharge
+ '.'

               :

               getTheSelectedClub().teacherInCharge?

                  " This school club is further handled by " +
getTheSelectedClub().teacherInCharge + '.'

                     :

                        "" }}

         <br>

         As of now, they have <span class="clubDetailsStat">{{
getTheSelectedClub().footballClubTotalStatistics.wins }}</span> wins,

         <span class="clubDetailsStat">{{
getTheSelectedClub().footballClubTotalStatistics.defeats }}</span> defeats,

         and <span class="clubDetailsStat">{{
getTheSelectedClub().footballClubTotalStatistics.draws }}</span> draws to their name,

         thereby currently holding <span class="clubDetailsStat">{{
getTheSelectedClub().footballClubTotalStatistics.points }}</span> points.
```

```
        </h2>
    </div>


    <div class="main-center-player">
        <!--   All used simply for UI. Display all the players of the club     -->
        <div *ngFor="let player of getTheSelectedClub().allPlayers; let i = index" class="main-
center-eachPlayerCard">
            <img src="{{ getImageLink()[i] }}" style="width: 100%; height: 300px; object-fit:
contain" alt="playerImage" />
            <div class="cardContent">
                <h1>
                    {{ player.name.toLocaleUpperCase() }}
                    <i style="color: gold;" class="material-icons" *ngFor="let numOfStar of
[].constructor(getStarRatingCount()[i])">star</i>
                </h1>
            </div>


            <!--    render each player of the club      -->
            <div class="hoverContainer" [ngStyle]="{ 'background-image': 'linear-gradient(to
bottom, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 1))' }">
                <div class="hoverText">
                    <div class="hoverTextName">
                        <h2>{{ player.name.toLocaleUpperCase() }}</h2>
                    </div>
                    <div class="hoverLowerSection">
                        <div class="colOne">
                            <div class="colLabel">
                                <p>SPE</p>
                                <p>ACC</p>
                                <p>STA</p>
                                <p>CON</p>
                                <p>STR</p>
                            </div>
                            <div class="colStat">
                                <p><span class="colStat">{{ player.playerStats.speed < 10? "0" +
player.playerStats.speed : player.playerStats.speed }}</span></p>
                                <p><span class="colStat">{{ player.playerStats.accuracy < 10? "0" +
player.playerStats.accuracy : player.playerStats.accuracy }}</span></p>
                                <p><span class="colStat">{{ player.playerStats.stamina < 10? "0" +
player.playerStats.stamina : player.playerStats.stamina }}</span></p>
                                <p><span class="colStat">{{ player.playerStats.control < 10? "0" +
player.playerStats.control : player.playerStats.control }}</span></p>
                                <p><span class="colStat">{{ player.playerStats.strength < 10? "0" +
player.playerStats.strength : player.playerStats.strength }}</span></p>
                            </div>
                        </div>

                        <div class="colTwo">
                            <div class="colLabel">
                                <p>TAC</p>
                                <p>PAS</p>
                                <p>CRO</p>
                                <p>SHO</p>
                                <p>HEA</p>
                            </div>
                            <div class="colStat">
                                <p><span class="colStat">{{ player.playerStats.tackle < 10? "0" +
player.playerStats.tackle : player.playerStats.tackle }}</span></p>
                                <p><span class="colStat">{{ player.playerStats.passing < 10? "0" +
player.playerStats.passing : player.playerStats.passing }}</span></p>
```

```
                <p><span class="colStat">{{ player.playerStats.crossing < 10? "0" +
player.playerStats.crossing : player.playerStats.crossing }}</span></p>

                <p><span class="colStat">{{ player.playerStats.shooting < 10? "0" +
player.playerStats.shooting : player.playerStats.shooting }}</span></p>

                <p><span class="colStat">{{ player.playerStats.heading < 10? "0" +
player.playerStats.heading : player.playerStats.heading }}</span></p>

              </div>

            </div>



            <div class="colThree">

              <div class="colThreeContainer">

                <p class="playerNationality">{{ player.nationality }}<span class="material-
icons">outlined_flag</span></p>

                <p>🦶{{ player.preferredFoot }}</p>

                <p><i class="material-icons">accessibility</i><i class="material-
icons">height</i>{{ player.height }}</p>

                <p [ngStyle] = "{ 'background-color' : getPositionBgColor()[i] }"
class="playerPosition">{{ player.position }}</p>

              </div>

            </div>

          </div>

          <div class="hoverFooter">

            <p style="font-size: 20px"><span class="colStat" style=" color: #62ffaf">Holds
and maintains an overall stat of {{player.playerStats.overall < 10? "0" +
player.playerStats.overall : player.playerStats.overall }}%</span></p>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>
```

selected-club.component.css

```
@import
url('https://fonts.googleapis.com/css2?family=East+Sea+Dokdo&family=Sansita+Swashed:wght
@600&display=swap');

@import url('https://fonts.googleapis.com/css2?family=Satisfy&display=swap');


.container {
   padding: 70px 60px 0px 60px;
   min-height: 94vh;
   height: 100%;
   margin-top: 3%;
   animation: fadein 1s;
   -moz-animation: fadein 1s;
   -webkit-animation: fadein 1s;
   -o-animation: fadein 1s;
   border-left: 5px solid purple;
   border-right: 5px solid purple;
   background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 1)),
url(/assets/images/PL-kevin.jpg);
   background-size: cover;
   background-repeat: no-repeat;
   background-attachment: fixed;
}
@keyframes fadein {
   from {
      opacity:0;
      transform: translate(0, -10%)
   }
   to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-o-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10px)
    }
    to {
        opacity:1;
        transform: translate(0, 0px)
    }
}

.top-header-text {
    display: flex;
    justify-content: center;
    align-items: flex-start;
    flex-direction: column;
    margin-bottom: 40px;
}

.top-header-text > h1 {
    font-weight: bolder;
    font-family: 'East Sea Dokdo', cursive;
    font-family: 'Sansita Swashed', cursive;
    font-family: 'Satisfy', cursive;
    font-size: 50px;
    color: white;
}

.top-header-text .clubDetails {
    color: #62ffaf;
    font-family: 'East Sea Dokdo', cursive;
    font-family: 'Sansita Swashed', cursive;
```

```
}
.top-header-text .clubDetails .clubDetailsStat {
    font-family: 'East Sea Dokdo', cursive;
    font-family: 'Sansita Swashed', cursive;
    color: white;
    font-weight: bolder;
}

.main-center-player {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.main-center-eachPlayerCard {
    box-shadow: 0 4px 8px 0 rgba(255, 255, 255, 0.8);
    transition: 0.3s;
    width: 30%;
    margin: 20px;
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    background-color: rgba(0, 0, 0, 0.5);
    position: relative;
}

.main-center-eachPlayerCard:hover {
    transform: scale(1.07);
    box-shadow: 0 8px 16px 0 rgba(255,0,0,0.2);
    border: 2px solid red;
}

.cardContent {
    background-color: #fff;
    text-align: center;
    padding: 2px 16px;
}

.cardContent > h1 {
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    letter-spacing: 2px;
}

.hoverContainer {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    width: 100%;
    height: auto;
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    background-size: contain;
    opacity: 0;
    text-align: center;
    transition: 0.3s ease;
    padding: 8px;
}
.hoverContainer:hover {
    opacity: 1;
}
.hoverText {
    position: absolute;
    color: white;
    top: 50%;
    left: 50%;
    width: 100%;
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: space-around;
    transform: translate(-50%, -50%);
    transform: 0.3s ease;
    font-size: 20px;
}

.hoverLowerSection {
    padding: 16px;
    display: flex;
    justify-content: space-between;
}

.colOne, .colTwo, .colThree {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.colStat {
    color: #04f5ff;
    font-weight: bolder;
    margin-left: 4px;
}

.colThree .colThreeContainer .playerPosition {
    color: black;
    font-weight: bolder;
}

.colThree .colThreeContainer .playerNationality {
    font-weight: bolder;
}
```

selected-club.component.ts

```
/*
 * SelectedClubComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { FootballClub } from 'src/app/models/FootballClub.model';
import { SelectedClubService } from 'src/app/services/selected-club/selected-club.service';

/**
 * SelectedClubComponent class, which will be used to render a selected club view
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({
  selector: 'app-selected-club',
  templateUrl: './selected-club.component.html',
  styleUrls: ['./selected-club.component.css']
})
export class SelectedClubComponent implements OnInit {
  private selectedClub : FootballClub;
  private clubName : string;

  //*UI only based arrays*//
  private positionBackgroundColor: string[] = [];
  private starRatingCount: number[] = [];
  private playerImageLink: string[] = [];

  constructor(private selectedClubService : SelectedClubService, private route : ActivatedRoute)
{ }

  ngOnInit(): void {
    //*Obtain the clubName url parameter passed in the URL, this can be then used to obtain the
specific club*//
    this.clubName = this.route.snapshot.params["clubName"];
    this.getSelectedClub();
  }

  /**
   * Get the selected club
   */
  public getSelectedClub() : void {
    this.selectedClubService.getSelectedClub(this.clubName).subscribe(
      response => this.handleSuccessfulResponse(response),
      error => this.handleErrorResponse(error)
    )
  }
  /**
   * Populates the selectedClub. Successful response
   * @param response - will contain the Json format of the selectedClub
   */
  private handleSuccessfulResponse(response : any) : void {
    this.selectedClub = response;
    this.determinePositionBackgroundColor();
    this.determineStarRatingCount();
    this.determinePlayerImageLink();
    console.log(response);
  }
  /**
   * Populates the selectedClub. Error response
   * @param error - an error message
   */
  private handleErrorResponse(error : any) : void {
    this.selectedClub = error.message;
    console.log(error);
  }

  /**
   * Place a background border color for a player based on their pitch position
   */
  private determinePositionBackgroundColor() : void {
    const goldenRod = [ 'SW', 'LM', 'RM', 'CM', 'DM', 'AM' ];
    const red = ['CF', 'SS', 'S', 'WF'];
    const green = ['LB', 'RB', 'CB', 'LWB', 'RWB'];
    // const purple = ['GK']
    if (this.selectedClub) {
      for(let player of this.selectedClub.allPlayers) {
        if(goldenRod.includes(player.position)) {
          this.positionBackgroundColor.push('goldenrod');
        } else if(red.includes(player.position)) {
          this.positionBackgroundColor.push('red');
        } else if(green.includes(player.position)) {
          this.positionBackgroundColor.push('green');
        } else {
          this.positionBackgroundColor.push('purple');
        }
      }
    }
  }
  /**
   * Place a star rating count for a player based on their overall stat
   */
  private determineStarRatingCount() : void {
    if(this.selectedClub) {
      for(let player of this.selectedClub.allPlayers) {
        if(player.playerStats.overall < 50) {
          this.starRatingCount.push(3);
        } else if(player.playerStats.overall < 75) {
          this.starRatingCount.push(4);
        } else {
          this.starRatingCount.push(5);
        }
      }
    }
  }
  /**
   * Place a background image for a player randomly
   */
  private determinePlayerImageLink() : void {
    let imagePaths: string[] = ['assets/images/PL-player.png', 'assets/images/PL-player2.png', 
'assets/images/PL-player3.png', 'assets/images/PL-player4.png', 'assets/images/PL-player5.png', 
'assets/images/PL-player6.png', 'assets/images/PL-player7.png', 'assets/images/PL-player8.png'];
    if(this.selectedClub) {
      for(let _ of this.selectedClub.allPlayers) {
        let randomNum = Math.floor(Math.random() * imagePaths.length);
        this.playerImageLink.push(imagePaths[randomNum]);
      }
    }
  }
  /**
```

```
   * Return the selectedClub
   * @return FootballClub - the selected club
   */
  public getTheSelectedClub() : FootballClub {
    return this.selectedClub;
  }
  /**
   * Return the bgColor array
   * @return string[] - array of bg colors
   */
  public getPositionBgColor() : string[] {
    return this.positionBackgroundColor;
  }
  /**
   * Return the imageLink array
   * @return string[] - array of image links
   */
  public getImageLink() : string[] {
    return this.playerImageLink;
  }
  /**
   * Return the starRating array
   * @return number[] - array of star rating counts
   */
  public getStarRatingCount() : number[] {
    return this.starRatingCount;
  }
}
```

2.6.1.7 selected-match

selected-match.component.html

```
<div class="container">
    <div class="top-header">
        <img src="assets/images/PL-LionText-transparent.png" alt="lionTextTransparent" />
    </div>


    <div class="main-center">
        <div class="main-center-left">
            <!--    render all the stats of both the teams of the selected match      -->
            <h3 class="main-center-left-clubName">{{
getSelectedMatch().firstTeam.clubName.toLocaleUpperCase() }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.goals }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.shots }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.shotsOnTarget }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.possession }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.passes }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.passAccuracy }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.fouls }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.yellowCards }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.redCards }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.offsides }}</h3>
            <h3>{{ getSelectedMatch().firstTeamSingleMatchStats.corners }}</h3>
        </div>
        <div class="main-center-center">
            <h3>TEAM STATS</h3>
            <h3>Goals</h3>
            <h3>Shots</h3>
            <h3>Shots on target</h3>
            <h3>Possession</h3>
            <h3>Passes</h3>
            <h3>Pass accuracy</h3>
            <h3>Fouls</h3>
            <h3>Yellow cards</h3>
            <h3>Red cards</h3>
            <h3>Offsides</h3>
            <h3>Corners</h3>
        </div>
        <div class="main-center-right">
            <h3 class="main-center-right-clubName">{{
getSelectedMatch().secondTeam.clubName.toLocaleUpperCase() }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.goals }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.shots }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.shotsOnTarget }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.possession }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.passes }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.passAccuracy }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.fouls }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.yellowCards }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.redCards }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.offsides }}</h3>
            <h3>{{ getSelectedMatch().secondTeamSingleMatchStats.corners }}</h3>
        </div>
    </div>


    <div class="bottom-footer">
        <h3 class="bottom-footer-text">{{ getFooterMsg() }}</h3>
    </div></div>
```

selected-match.component.css

```
.container {
    background-color: #222;
    height: 94vh;
    margin-top: 3%;
    border-left: 5px solid purple;
    border-right: 5px solid purple;
    animation: fadein 1s;
    -moz-animation: fadein 1s;
    -webkit-animation: fadein 1s;
    -o-animation: fadein 1s;
    background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 1)),
url(/assets/images/PL-mahrez.jpg);
    background-size: cover;
}
@keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-moz-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-webkit-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10%)
    }
    to {
        opacity:1;
        transform: translate(0, 0%)
    }
}
@-o-keyframes fadein {
    from {
        opacity:0;
        transform: translate(0, -10px)
    }
    to {
        opacity:1;
        transform: translate(0, 0px)
    }
}

.top-header {
    display: flex;
    justify-content: center;
    align-items: center;
}
.top-header > img {
    object-fit: contain;
    height: 150px;
}

.main-center {
    display: flex;
    justify-content: space-evenly;
}
.main-center-left, .main-center-center, .main-center-right {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    margin: 10px;
}
.main-center-left h3, .main-center-center h3, .main-center-right h3 {
    margin: 2px;
    font-size: 22px;
}
.main-center-left {
    color: #04f5ff;
    /* margin-left: 50px; */
    width: 500px;
```

```
}
.main-center-left-clubName {
   color: white;
   text-shadow: 1.5px 1.5px #04f5ff;
   font-size: 25px;
}
.main-center-center {
   color: white;
}
.main-center-right {
   color: #f00;
   /* margin-right: 50px; */
   width: 500px;
}
.main-center-right-clubName {
   color: white;
   text-shadow: 1.5px 1.5px #f00;
   font-size: 25px;
}

.bottom-footer {
   display: flex;
   justify-content: center;
   align-items: center;
   margin-top: 8px;
}

.bottom-footer h3 {
   font-size: 30px;
   background: -webkit-linear-gradient(left, #04f5ff , #f00);
   background: -o-linear-gradient(right, #04f5ff, #f00);
   background: -moz-linear-gradient(right, #04f5ff, #f00);
   background: linear-gradient(to right, #04f5ff , #f00);
   -webkit-background-clip: text;
   -webkit-text-fill-color: transparent;
}
```

selected-match.component.ts

```
/*
 * SelectedMatchComponent
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { FootballMatch } from 'src/app/models/FootballMatch.model';
import { SelectedMatchService } from 'src/app/services/selected-match/selected-match.service';

/**
 * SelectedMatchComponent class, which will be used to render a selected match view
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Component({
  selector: 'app-selected-match',
  templateUrl: './selected-match.component.html',
  styleUrls: ['./selected-match.component.css']
})
export class SelectedMatchComponent implements OnInit {
  //*id to index into the array of all matches in the Play backend*//
  private id : string;
  private footballMatch : FootballMatch;
  //*Render a message highlighting the match*//
  private footerMsg : string;

  constructor(private selectedMatchService : SelectedMatchService, private route :
ActivatedRoute) { }

  ngOnInit(): void {
    //*obtain the id passed in the URL as parameter*//
    this.id = this.route.snapshot.params["id"];
    this.getFootballMatch();
  }

  /**
   * Get the selected match
   */
  public getFootballMatch() : void {
    this.selectedMatchService.getSelectedMatch(this.id).subscribe(
      response => this.handleSuccessfulResponse(response),
      error => this.handleErrorResponse(error)
    );
  }
  /**
```

```
   * Populates the footballMatch and footerMsg variables. Successful response
   * @param response - will hold the json format of the match data
   */
  private handleSuccessfulResponse(response : any) : void {
    this.footballMatch = response;
    console.log(response);
    //*create a highlight message based on the match winner*//
    this.footerMsg = this.footballMatch.firstTeamSingleMatchStats.goals >
this.footballMatch.secondTeamSingleMatchStats.goals ?
                    this.footballMatch.firstTeam.clubName + " beat "  +
this.footballMatch.secondTeam.clubName + ", scoring " +
                    this.footballMatch.firstTeamSingleMatchStats.goals + " goals with a possession of "
+
                    this.footballMatch.firstTeamSingleMatchStats.possession + "%" :
                     this.footballMatch.firstTeamSingleMatchStats.goals <
this.footballMatch.secondTeamSingleMatchStats.goals ?
                     this.footballMatch.secondTeam.clubName + " beat "  +
this.footballMatch.firstTeam.clubName + ", scoring " +
                     this.footballMatch.secondTeamSingleMatchStats.goals + " goals with a possession
of " +
                     this.footballMatch.secondTeamSingleMatchStats.possession + "%" :
                      "The match was a draw, with both teams scoring " +
this.footballMatch.firstTeamSingleMatchStats.goals + " goals."
  }
  /**
   * Populates the footballMatch variable. Error response
   * @param error - will hold an error message
   */
  private handleErrorResponse(error : any) : void {
    this.footballMatch = error.message;
    console.log(error);
  }

  /**
   * Returns the selected club
   * @return FootballMatch - the selected club
   */
  public getSelectedMatch() : FootballMatch {
    return this.footballMatch;
  }

  /**
   * Returns the generated footer message
   * @return string - the generated message
   */
  public getFooterMsg() : string {
    return this.footerMsg;
  }
}
```

2.6.2 Models

Color.model.ts

```
/*
 * Color
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


/**
 * Color interface -> To provide a data type for Color.
 * Will be used to hold Color objects that are retrieved, that is a part of a SportsClubKit
 * Will hold the exact same Attributes as the Color object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface Color {
   red : number;
   green : number;
   blue : number;
}
```

FootballClub.model.ts

```
/*
 * FootballClub
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { FootballClubTotalStatistics } from './FootballClubTotalStatistics.model';
import { Player } from './Player.model';
import { SportsClubKit } from './SportsClubKit.model';
/**
 * FootballClub interface -> To provide a data type for FootballClub
 * Will be used to hold FootballClub objects that are retrieved
 * Will hold the exact same Attributes as the FootballClub object of Java
 * Also holds the lecturer and teacher in charge attributes, if in case the club is a university of
school club
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface FootballClub {
   allPlayers : Player[];
   amountOfPlayers : number;
   clubLocation : string;
   clubName : string;
   clubOwner : string;
   clubNetWorth : string;
   footballClubTotalStatistics : FootballClubTotalStatistics;
   kit : SportsClubKit;
   lecturerInCharge : string;
   teacherInCharge : string;
}
```

FootballClubTotalStatistics.model.ts

```
/*
 * FootballClubTotalStatistics
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


/**
 * FootballClubTotalStatistics interface -> To provide a data type for FootballClubTotalStatistics
 * Will be used to hold FootballClubTotalStatistics objects that are retrieved, that are a part of the
FootballCLub interface
 * Will hold the exact same Attributes as the FootballClubTotalStatistics object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface FootballClubTotalStatistics {
   defeats : number;
   draws : number;
   goalsAgainst : number;
   goalsFor : number;
   matchesPlayed : number;
   points : number;
   totalRedCards : number;
   totalYellowCards : number;
   wins : number;
   goalDifference : number;
}
```

FootballMatch.model.ts

```
/*
 * FootballMatch
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { FootballClub } from './FootballClub.model'
import { SingleMatchFootballClubStatistic } from './SingleMatchFootballClubStatistic.model';

/**
 * FootballMatch interface -> To provide a data type for FootballMatch
 * Will be used to hold FootballMatch objects that are retrieved
 * Will hold the exact same Attributes as the FootballClub object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface FootballMatch {
    secondTeamSingleMatchStats : SingleMatchFootballClubStatistic;
    firstTeamSingleMatchStats : SingleMatchFootballClubStatistic;
    matchDate : string;
    secondTeam : FootballClub;
    firstTeam : FootballClub;
}
```

Player.model.ts

```
/*
 * Player
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { PlayerStats } from './PlayerStats.model';

/**
 * Player interface -> To provide a data type for Player
 * Will be used to hold Player objects that are retrieved, that are a part of a FootballClub
 * Will hold the exact same Attributes as the Player object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface Player {
    name : string;
    position : string;
    height : number;
    nationality : string;
    shirtNumber : number;
    preferredFoot : string;
    playerStats : PlayerStats;
}
```

PlayerStats.model.ts

```
/*
 * PlayerStats
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
/**
 * PlayerStats interface -> To provide a data type for PlayerStats
 * Will be used to hold PlayerStats objects that are retrieved, that are a part of a Player
 * Will hold the exact same Attributes as the PlayerStats object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface PlayerStats {
    overall : number;
    tackle : number;
    passing : number;
    stamina : number;
    heading : number;
    shooting : number;
    crossing : number;
    accuracy : number;
    strength : number;
    control : number;
    goalKeeperHandling : number;
    goalKeepingPressure : number;
    goalKeepingReaction : number;
    speed : number;
}
```

SingleMatchFootballClubStatistic.model.ts

```
/*
 * SingleMatchFootballClubStatistic
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */


/**
 * SingleMatchFootballClubStatistic interface -> To provide a data type for
SingleMatchFootballClubStatistic
 * Will be used to hold SingleMatchFootballClubStatistic objects that are retrieved, that are a
part of a FootballMatch
 * Will hold the exact same Attributes as the SingleMatchFootballClubStatistic object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface SingleMatchFootballClubStatistic {
   corners : number;
   fouls : number;
   goals : number;
   offsides : number;
   passes : number;
   passAccuracy : number;
   possession : number;
   redCards : number;
   shots : number;
   shotsOnTarget : number;
   yellowCards : number;
}
```

SportsClubKit.model.ts

```
/*
 * SportsClubKit
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { Color } from './Color.model';

/**
 * SportsClubKit interface -> To provide a data type for SportsClubKit
 * Will be used to hold SportsClubKit objects that are retrieved, that are a part of a FootballClub
 * Will hold the exact same Attributes as the SportsClubKit object of Java
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
export interface SportsClubKit {
   sponsor : string;
   topColor : Color;
   bottomColor : Color;
}
```

2.6.3 Services

2.6.3.1 all-clubs.service.ts

```
/*
 * AllClubsService
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'
import { FootballClub } from 'src/app/models/FootballClub.model';
import { Observable } from 'rxjs';
import { BASE_URL } from 'src/app/app.constants';
/**
 * AllClubsService class, to be used to fetch all the clubs, will be calling the API URL
localhost:9000/pointstable
 * The Controller method associated with /pointstable in the Play backend will be called
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Injectable({  providedIn: 'root' })
export class AllClubsService {
  //*Specifying an object of HttpClient for REST API accessibility. Is accessible throughout*//
  constructor(private httpClient : HttpClient) { }
  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of all clubs
   * @return Observable (handles Asynchronous communication) with all FootballClubs
   */
  public getAllFootballClubs() : Observable<FootballClub> {
    return this.httpClient.get<FootballClub>(`${BASE_URL}/pointstable`); }
}
```

2.6.3.2 all-clubs-filter.service.ts

```
/*
 * AllClubsFilterService
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { BASE_URL } from 'src/app/app.constants';
import { FootballClub } from 'src/app/models/FootballClub.model';

/**
 * AllClubsFilterService class, to be used to fetch all the clubs filtered by a specific filter
 * will be calling the API URL localhost:9000/pointstable/filterType
 * The Controller method associated with /pointstable/filterType in the Play backend will be
called
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Injectable({
  providedIn: 'root'
})
export class AllClubsFilterService {
  constructor(private httpClient : HttpClient) { }

  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of all clubs,
sorted by goals for
```

```
   * @return Observable (handles Asynchronous communication) with all the sorted
FootballClubs
   */
  public getClubsGoalFilter() : Observable<FootballClub> {
    return this.httpClient.get<FootballClub>(`${BASE_URL}/pointstable/goalfilter`);
  }


  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of all clubs,
sorted by wins
   * @return Observable (handles Asynchronous communication) with all the sorted
FootballClubs
   */
  public getClubsWinFilter() : Observable<FootballClub> {
    return this.httpClient.get<FootballClub>(`${BASE_URL}/pointstable/winfilter`);
  }
}
```

2.6.3.3 all-matches.service.ts

```
/*
 * AllMatchesService
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'
import { FootballMatch } from 'src/app/models/FootballMatch.model';
import { Observable } from 'rxjs';
import { BASE_URL } from 'src/app/app.constants';
/**
 * AllMatchesService class, to be used to fetch all the matches, will be calling the API URL
localhost:9000/allmatches
 * The Controller method associated with /pointstable in the Play backend will be called
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Injectable({  providedIn: 'root' })
export class AllMatchesService {
  constructor(private httpClient : HttpClient) { }
  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of all
matches
   * @return Observable (handles Asynchronous communication) with all matches
   */
  public getAllFootballMatches() : Observable<FootballMatch> {
    return this.httpClient.get<FootballMatch>(`${BASE_URL}/allmatches`);
  }
}
```

2.6.3.4 matches-on-date.service.ts

```
/*

 * MatchesOnDateService

 * Copyright © 2020 Ammar Raneez. All Rights Reserved.

 */

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core'; import { Observable } from 'rxjs';

import { BASE_URL } from 'src/app/app.constants';

import { FootballMatch } from 'src/app/models/FootballMatch.model';

/**

 * MatchesOnDateService class, to be used to fetch all the matches at a specified date

 * will be calling the API URL localhost:9000/specificMatches/:date

 * The Controller method associated with /specificMatches/:date in the Play backend will be
called

 * @version 1.x December 5th 2020

 * @author Ammar Raneez | 2019163 | W1761196

 */

@Injectable({  providedIn: 'root' })

export class MatchesOnDateService {

  constructor(private httpClient : HttpClient) { }

  /**

   * Sends a GET request to the url and Returns an Observable holding the Json data of all the
matches played at a specific date

   * @param date - date of matches played. Will be used to obtain the matches played at the
specified date

   * @return Observable (handles Asynchronous communication) with all the specific matches

   */

  public getMatchesOnDate(date : string) : Observable<FootballMatch> {

    return this.httpClient.get<FootballMatch>(`${BASE_URL}/specificMatches/${date}`) }

}
```

2.6.3.5 play-match.service.ts

```
/*
 * PlayMatchService
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { BASE_URL } from 'src/app/app.constants';
import { FootballClub } from 'src/app/models/FootballClub.model';
/**
 * PlayMatchService class, to be used to fetch the updated matches upon playing a match
 * will be calling the API URL localhost:9000/playmatch
 * The Controller method associated with /playmatch in the Play backend will be called
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Injectable({  providedIn: 'root' })
export class PlayMatchService {
  constructor(private httpClient : HttpClient) { }
  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of all the
updated matches
   * @return Observable (handles Asynchronous communication) with all the updated matches
   */
  public playMatch() : Observable<FootballClub> {
    return this.httpClient.get<FootballClub>(`${BASE_URL}/pointstable/playmatch`);
  }
}
```

2.6.3.6 selected-club.service.ts

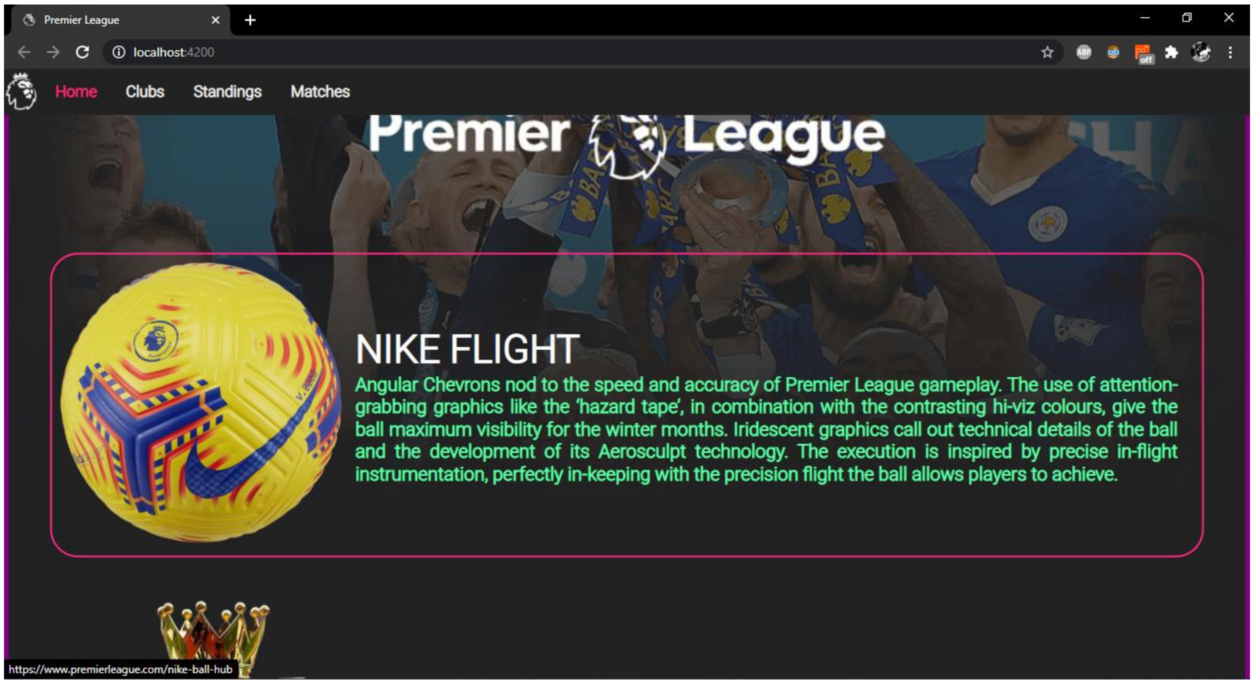
```
/*

 * SelectedClubService

 * Copyright © 2020 Ammar Raneez. All Rights Reserved.

 */

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core'; import { Observable } from 'rxjs';

import { BASE_URL } from 'src/app/app.constants';

import { FootballClub } from 'src/app/models/FootballClub.model';

/**

 * SelectedClubService class, to be used to fetch a specific club

 * will be calling the API URL localhost:9000/clubs/:clubName

 * The Controller method associated with /clubs/:clubName in the Play backend will be called

 * @version 1.x December 5th 2020

 * @author Ammar Raneez | 2019163 | W1761196

 */

@Injectable({  providedIn: 'root' })

export class SelectedClubService {

  constructor(private httpClient : HttpClient) {  }

  /**

   * Sends a GET request to the url and Returns an Observable holding the Json data of the
specific club

   * @param clubName - clubs name. Will be used to get the specific club, based on club name
equality

   * @return Observable (handles Asynchronous communication) with the specific club

   */

  public getSelectedClub(clubName : string) : Observable<FootballClub> {

    return this.httpClient.get<FootballClub>(`${BASE_URL}/clubs/${clubName}`); }

}
```

2.6.3.7 selected-match.service.ts

```
/*
 * SelectedMatchService
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core'; import { Observable } from 'rxjs';
import { BASE_URL } from 'src/app/app.constants';
import { FootballMatch } from '../../models/FootballMatch.model';
/**
 * SelectedMatchService class, to be used to fetch a specific match
 * will be calling the API URL localhost:9000/allmatches/:id
 * The Controller method associated with /allmatches/:id in the Play backend will be called
 * @version 1.x December 5th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@Injectable({  providedIn: 'root' })
export class SelectedMatchService {
  constructor(private httpClient : HttpClient) { }
  /**
   * Sends a GET request to the url and Returns an Observable holding the Json data of the specific match
   * @param id - match id. Will be used to index into the list of matches, to return the match at the specified index
   * @return Observable (handles Asynchronous communication) with the specific match
   */
  public getSelectedMatch(id : string) : Observable<FootballMatch> {
    return this.httpClient.get<FootballMatch>(`${BASE_URL}/allmatches/${id}`); }
}
```

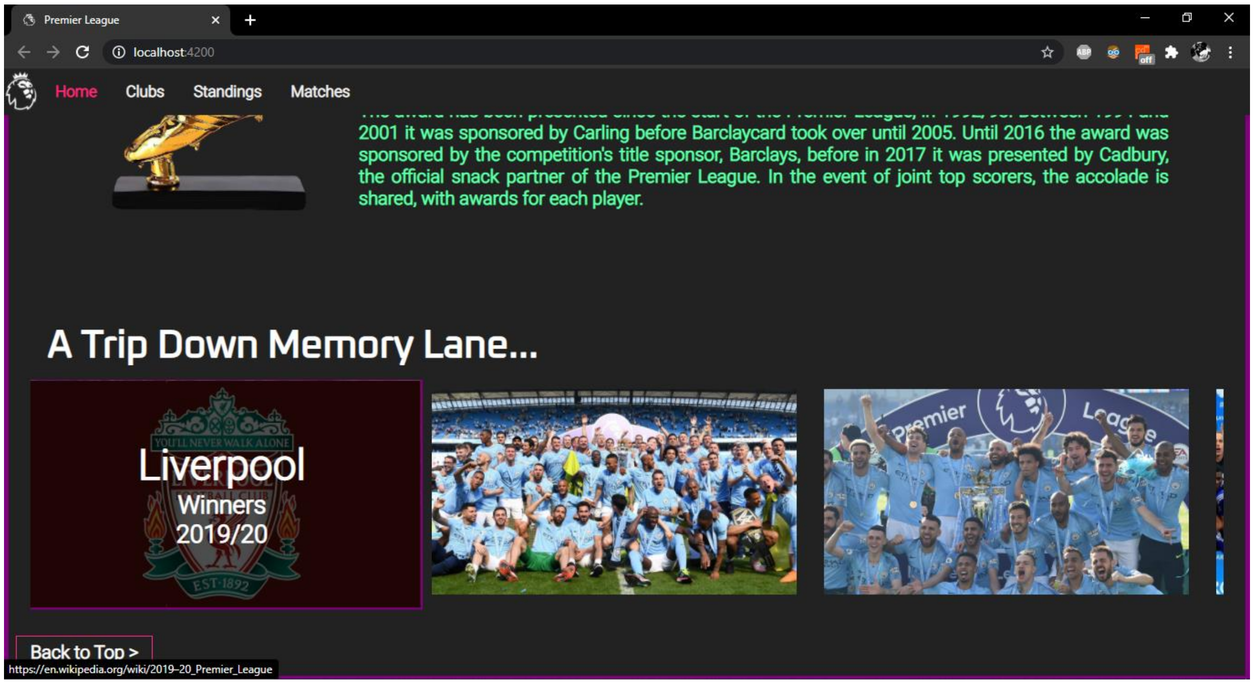
03 – Design Screenshots

A regular home page, that gives information on the REAL Premier League, clicking on each card navigates to websites holding more information.



NIKE FLIGHT

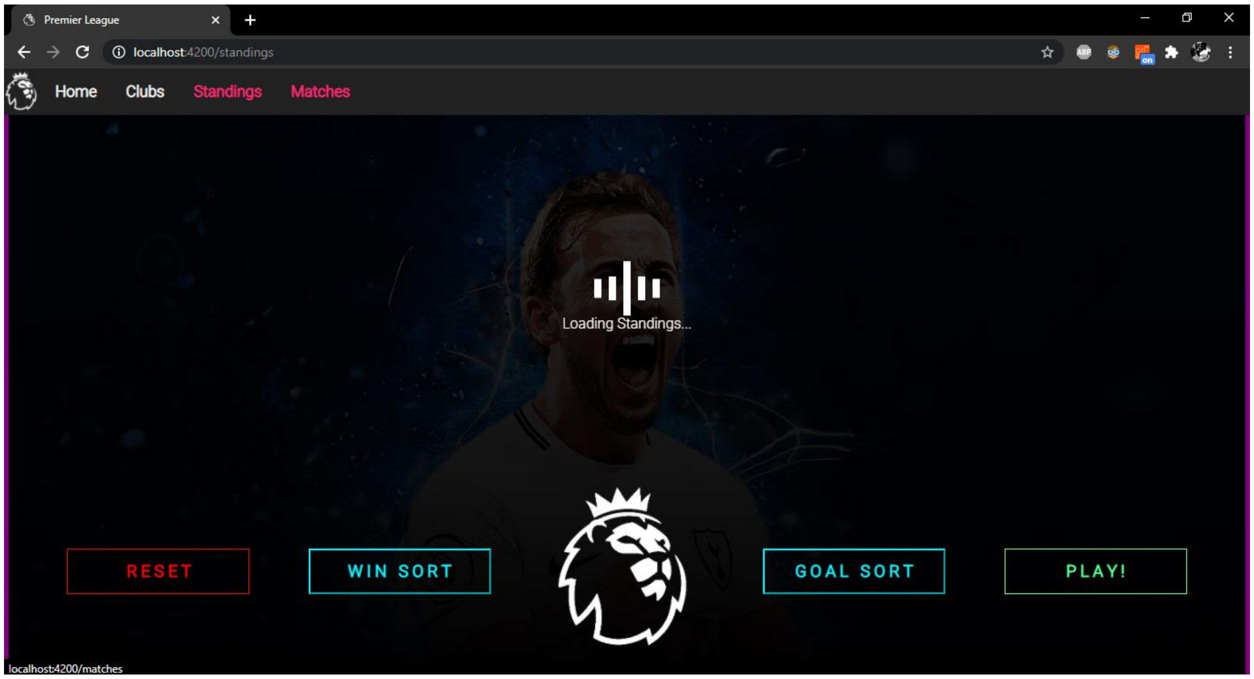
Angular Chevrons nod to the speed and accuracy of Premier League gameplay. The use of attention-grabbing graphics like the 'hazard tape', in combination with the contrasting hi-viz colours, give the ball maximum visibility for the winter months. Iridescent graphics call out technical details of the ball and the development of its Aerosculpt technology. The execution is inspired by precise in-flight instrumentation, perfectly in-keeping with the precision flight the ball allows players to achieve.

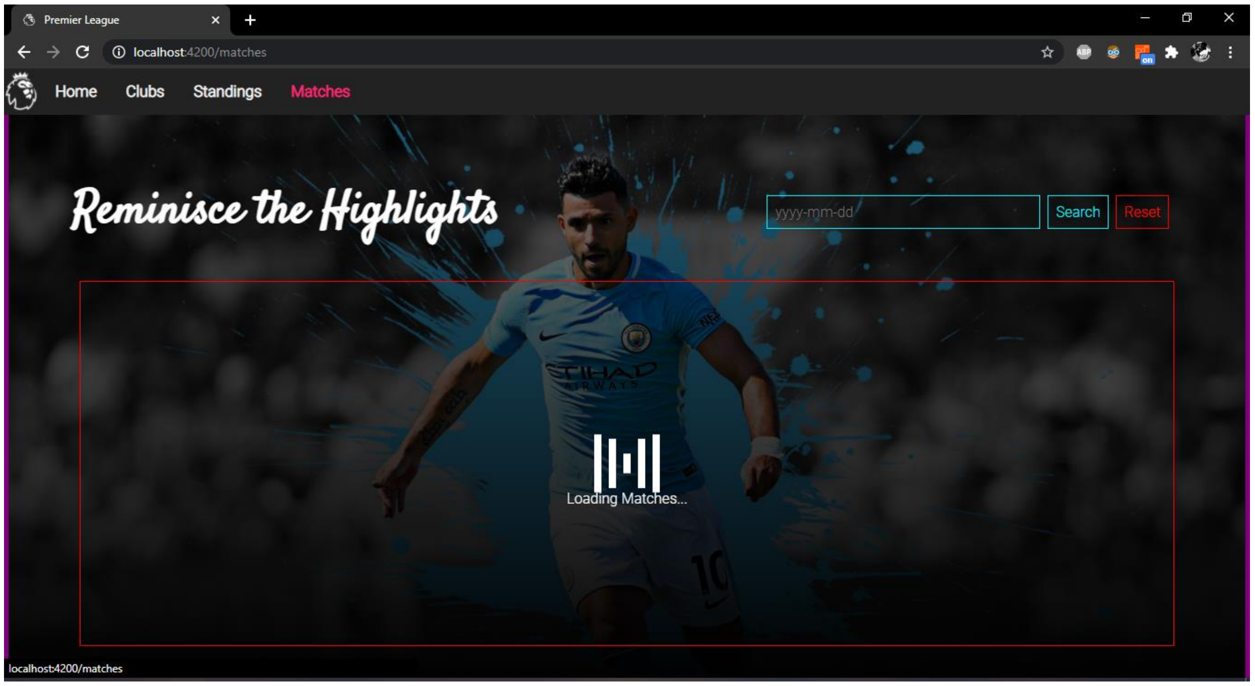


2001 it was sponsored by Carling before Barclaycard took over until 2005. Until 2016 the award was sponsored by the competition's title sponsor, Barclays, before in 2017 it was presented by Cadbury, the official snack partner of the Premier League. In the event of joint top scorers, the accolade is shared, with awards for each player.

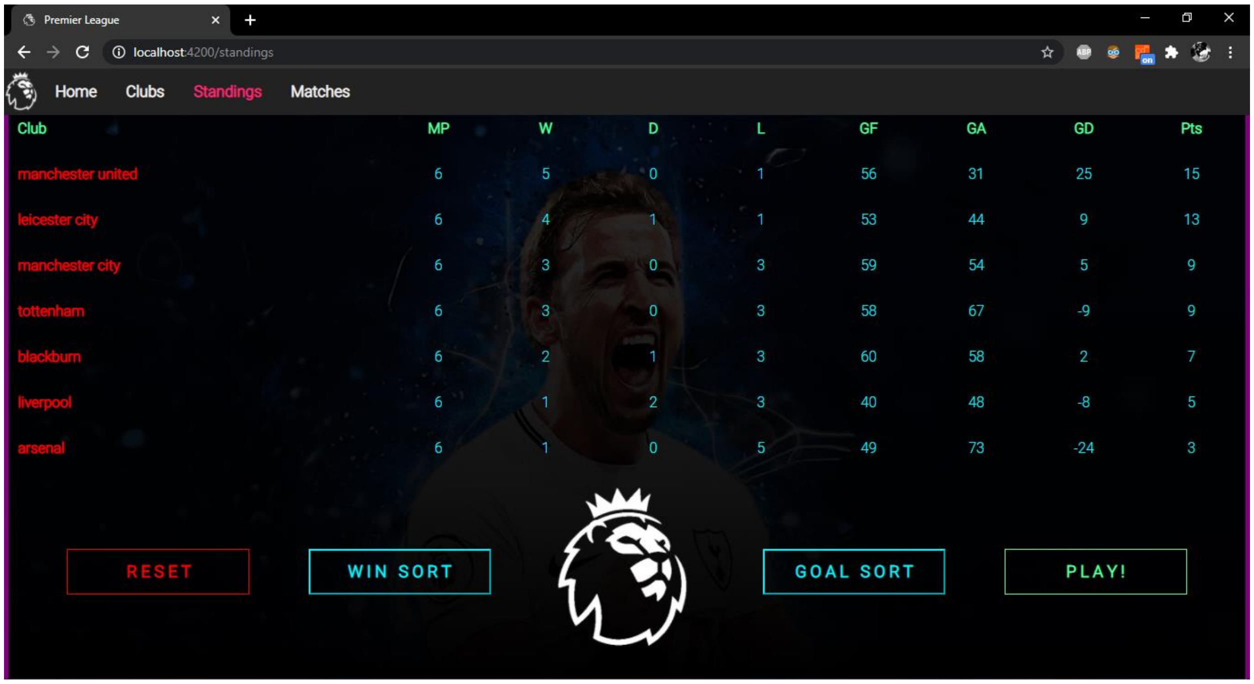
A Trip Down Memory Lane...

A loading icon is displayed on "Clubs", "Standings", and "Matches", to simulate the loading of data



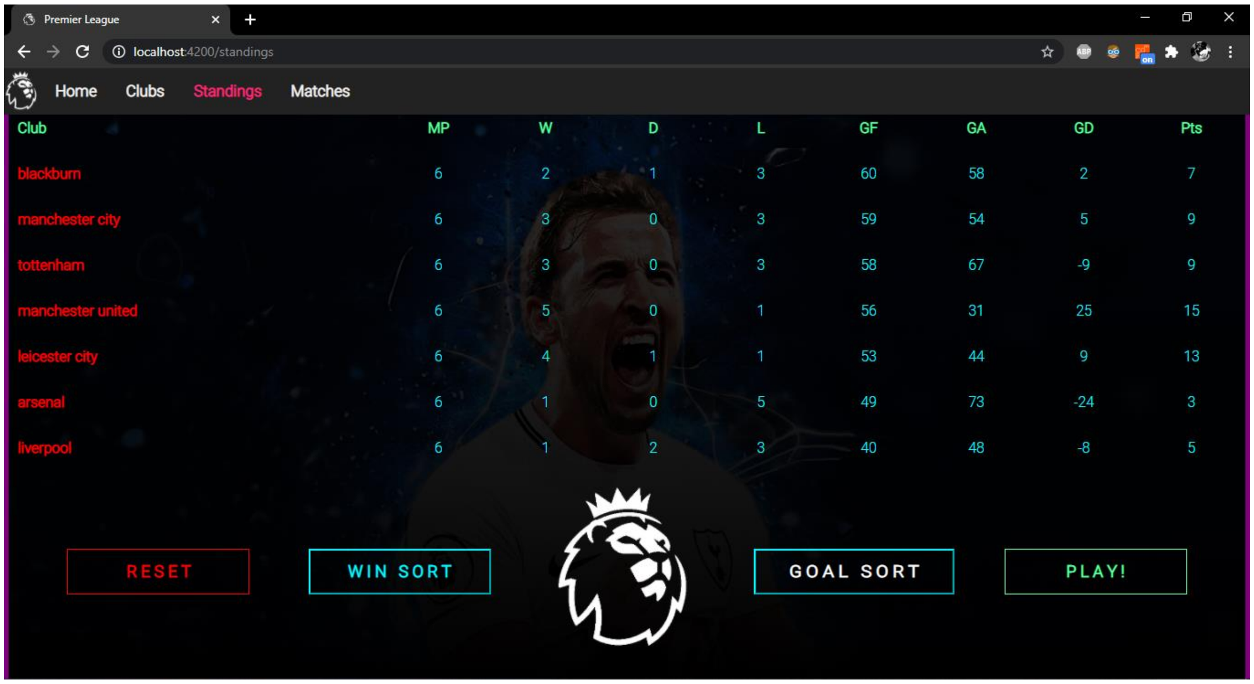


Standings displays the premier league table. Sorted based on "Pts" (Points), and secondarily (if same points the clubs are sorted by this), based on "GD" (Goal Difference).



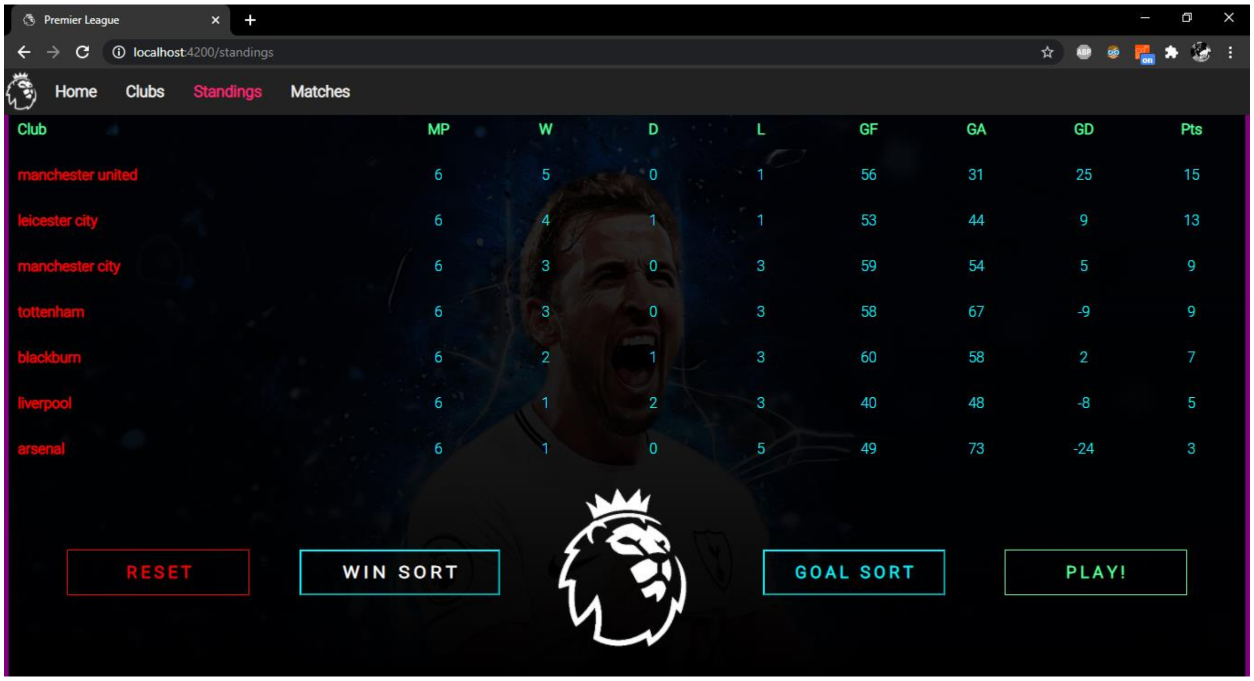
Club	MP	W	D	L	GF	GA	GD	Pts
manchester united	6	5	0	1	56	31	25	15
leicester city	6	4	1	1	53	44	9	13
manchester city	6	3	0	3	59	54	5	9
tottenham	6	3	0	3	58	67	-9	9
blackburn	6	2	1	3	60	58	2	7
liverpool	6	1	2	3	40	48	-8	5
arsenal	6	1	0	5	49	73	-24	3

Sorting based on "GF" (Goals For/ Goals Scored), secondarily sorted by "Pts" (Points) (clubs that have scored the same amount of goals are sorted by points)



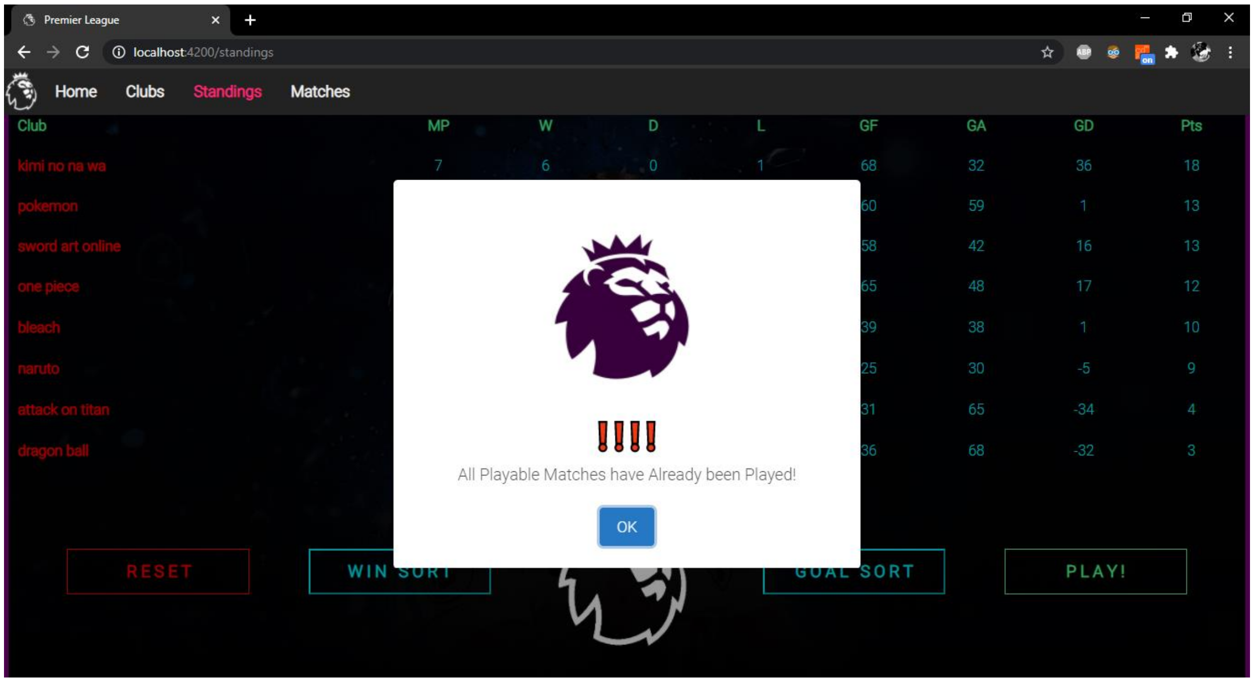
Club	MP	W	D	L	GF	GA	GD	Pts
blackburn	6	2	1	3	60	58	2	7
manchester city	6	3	0	3	59	54	5	9
tottenham	6	3	0	3	58	67	-9	9
manchester united	6	5	0	1	56	31	25	15
leicester city	6	4	1	1	53	44	9	13
arsenal	6	1	0	5	49	73	-24	3
liverpool	6	1	2	3	40	48	-8	5

Sorting based on "W" (Wins), secondarily also sorted by "Pts" (Points) (clubs that have same wins are sorted by points)



Club	MP	W	D	L	GF	GA	GD	Pts
manchester united	6	5	0	1	56	31	25	15
leicester city	6	4	1	1	53	44	9	13
manchester city	6	3	0	3	59	54	5	9
tottenham	6	3	0	3	58	67	-9	9
blackburn	6	2	1	3	60	58	2	7
liverpool	6	1	2	3	40	48	-8	5
arsenal	6	1	0	5	49	73	-24	3

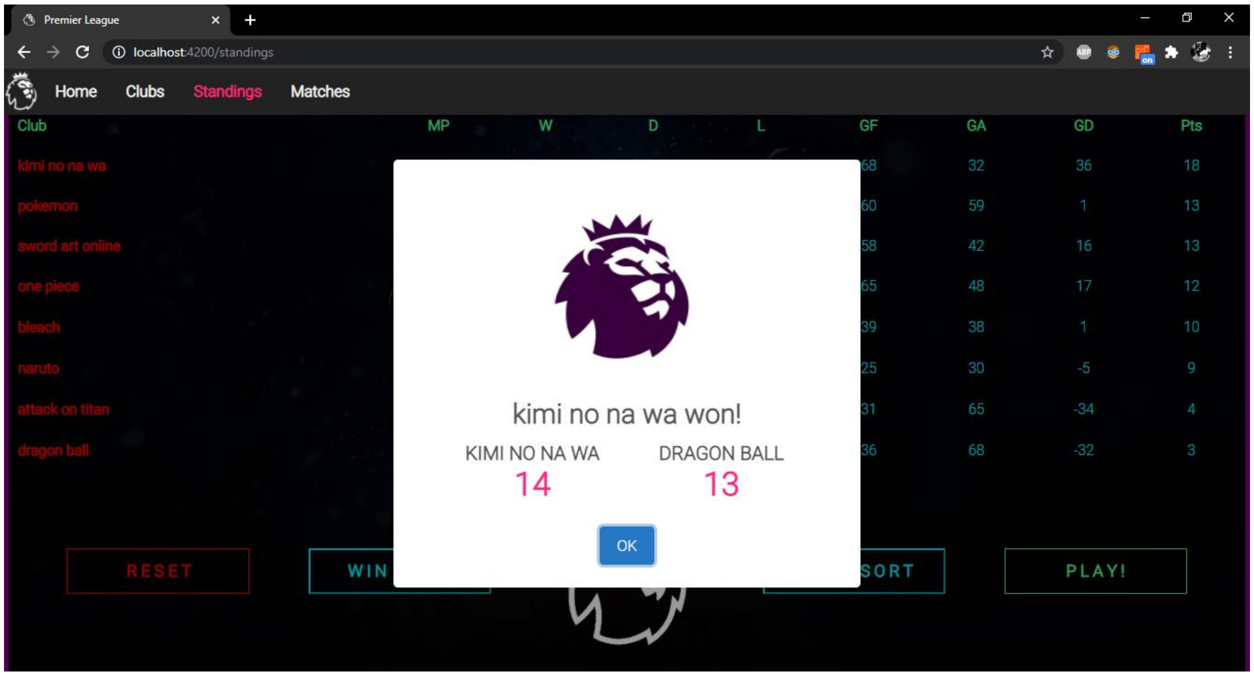
Popup error display: when there is an attempt to play a match but all matches have already been played.



Club	MP	W	D	L	GF	GA	GD	Pts
kimi no na wa	7	6	0	1	68	32	36	18
pokemon					60	59	1	13
sword art online					58	42	16	13
one piece					65	48	17	12
bleach					39	38	1	10
naruto					25	30	-5	9
attack on titan					31	65	-34	4
dragon ball					36	68	-32	3

All Playable Matches have Already been Played!

Popup display, shows the currently played match, if there are any legal matches, upon clicking play.



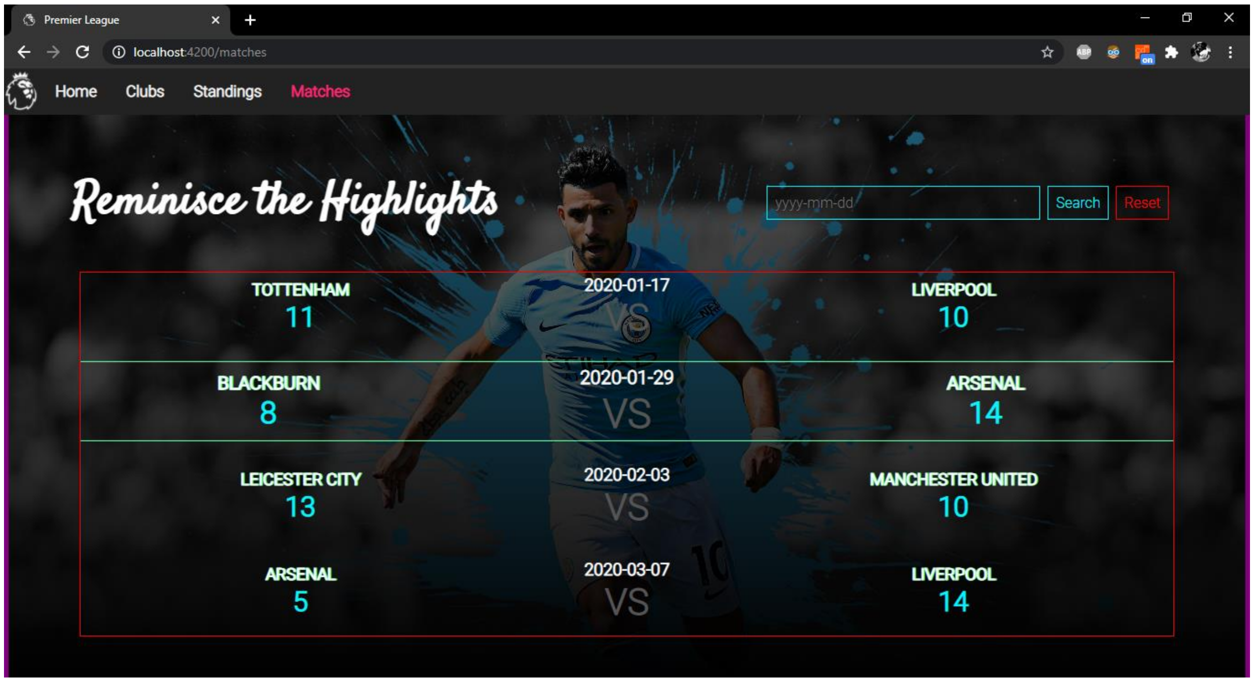
Club	MP	W	D	L	GF	GA	GD	Pts
kimi no na wa					68	32	36	18
pokemon					60	59	1	13
sword art online					58	42	16	13
one piece					65	48	17	12
bleach					39	38	1	10
naruto					25	30	-5	9
attack on titan					31	65	-34	4
dragon ball					36	68	-32	3

kimi no na wa won!

KIMI NO NA WA 14 DRAGON BALL 13

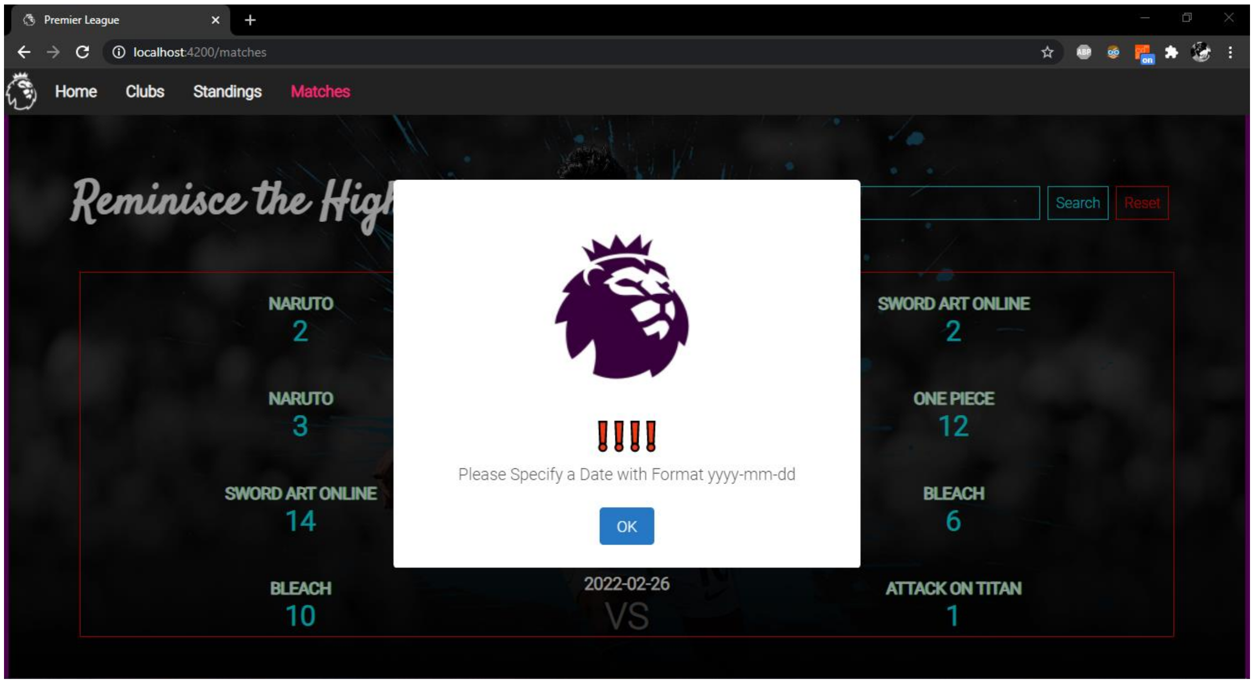
OK

List of all played matches. The matches are sorted in ascending order of the match date



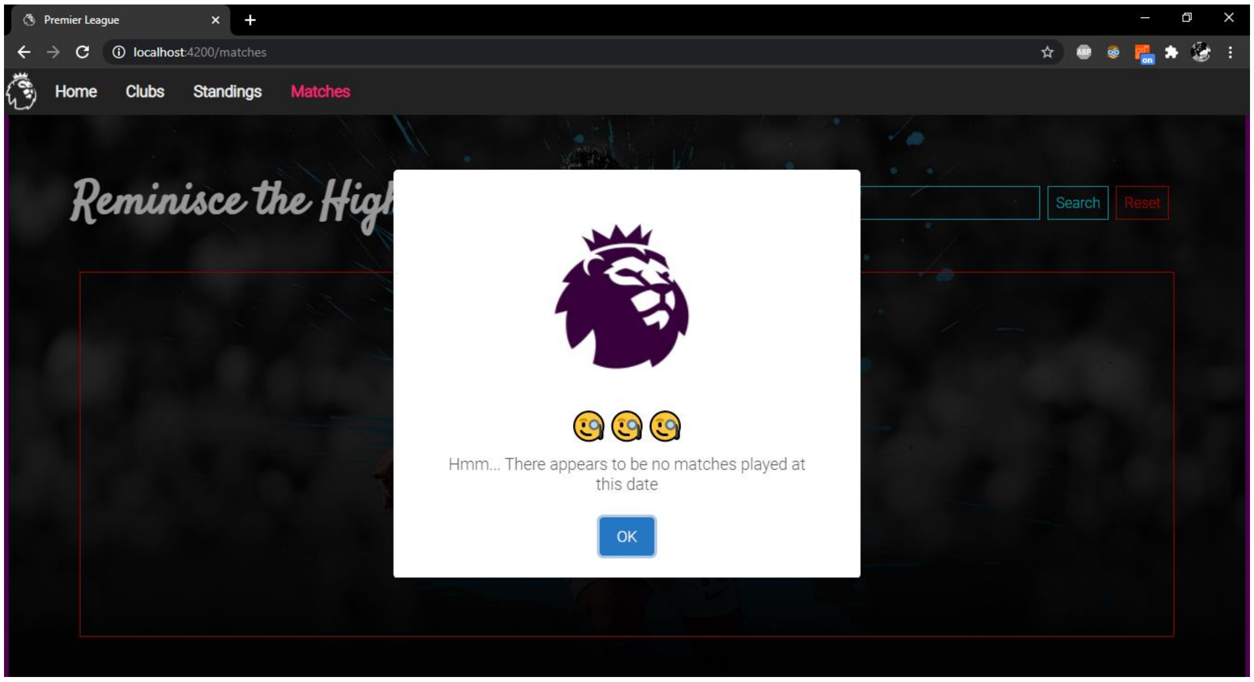
TOTTENHAM 11	2020-01-17 VS	LIVERPOOL 10
BLACKBURN 8	2020-01-29 VS	ARSENAL 14
LEICESTER CITY 13	2020-02-03 VS	MANCHESTER UNITED 10
ARSENAL 5	2020-03-07 VS	LIVERPOOL 14

Popup error display, for invalid date format



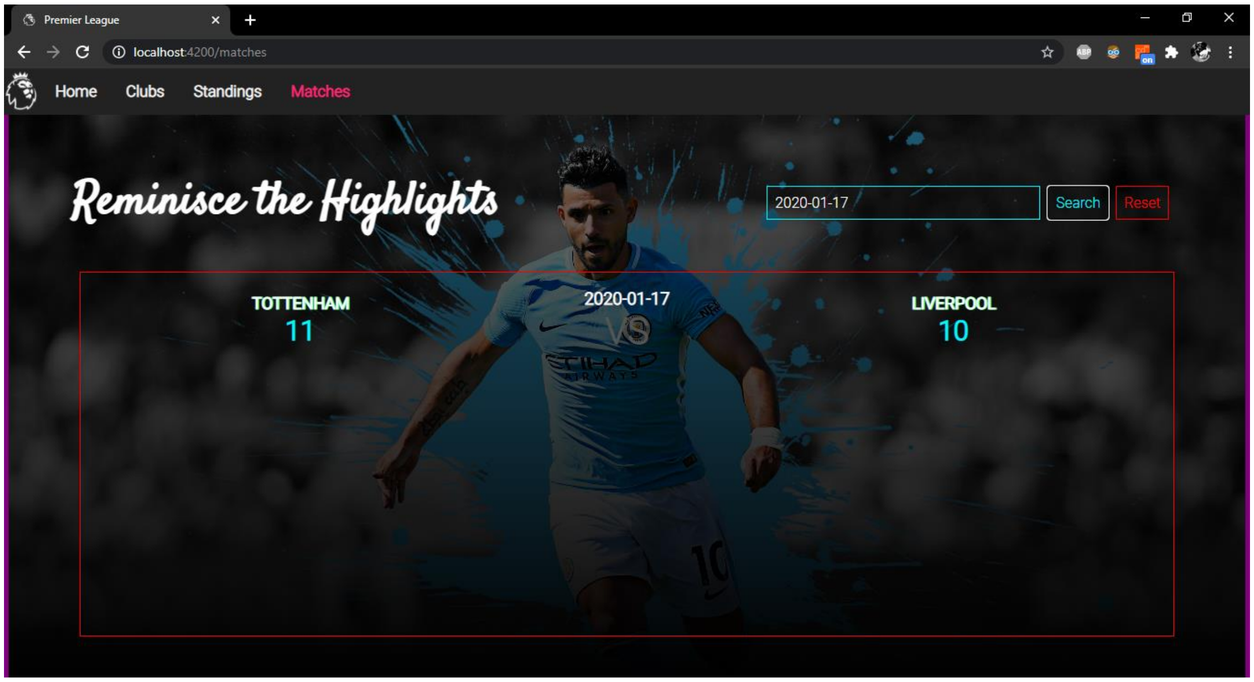
Please Specify a Date with Format yyyy-mm-dd

Popup info display, for no matches played in specified date



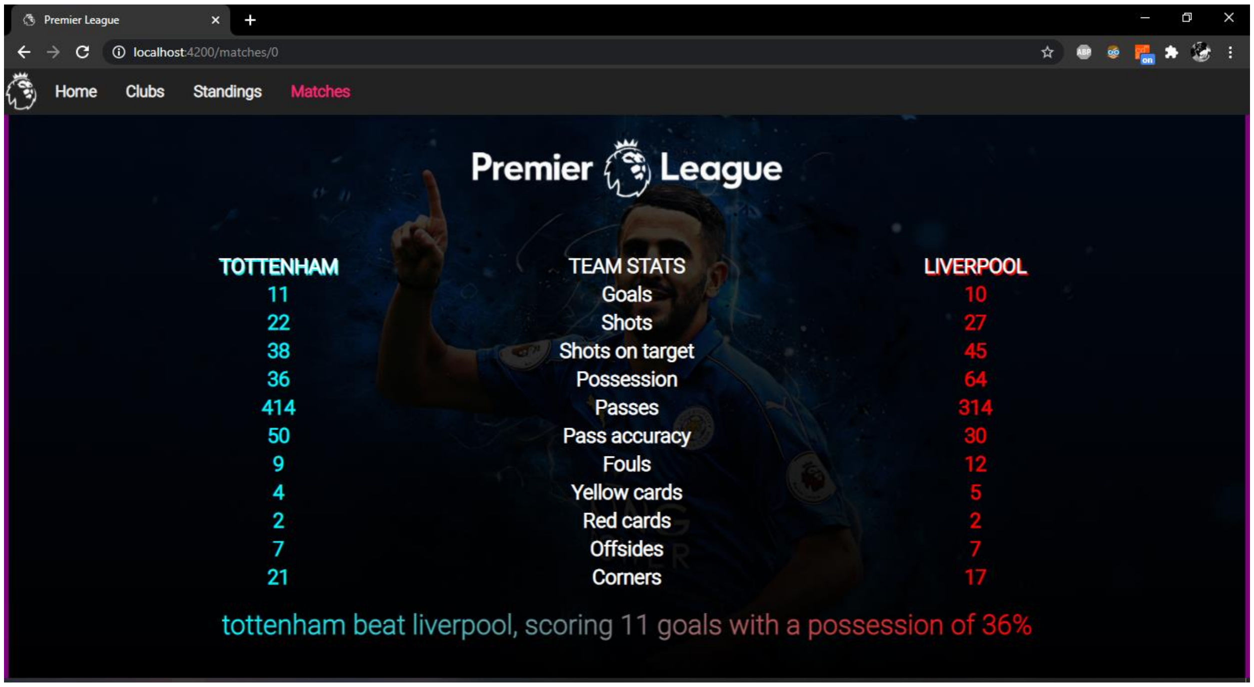
Hmm... There appears to be no matches played at this date

Display matches played on date specifed on the input text box



TOTTENHAM 11 — 2020-01-17 VS — LIVERPOOL 10

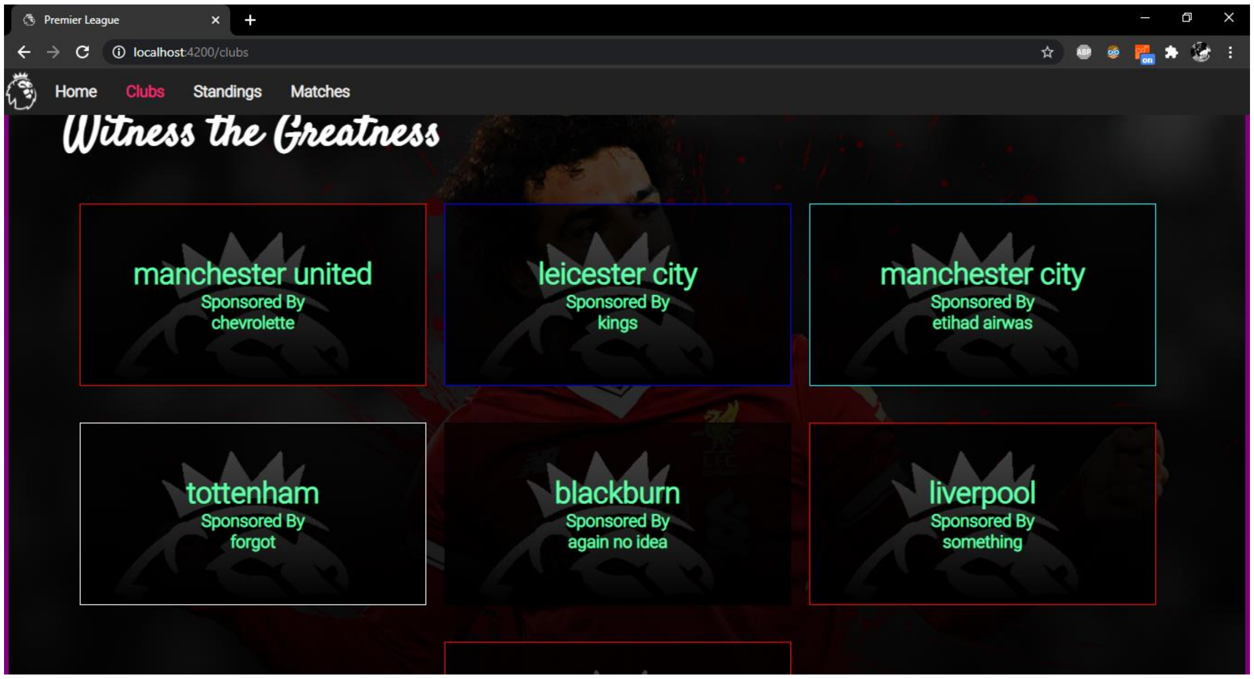
Further information of a match can be accessed by simply clicking on each of the rows, these are the extra statistics that are generated upon playing a match (stats apart from Goals and clubs)



TOTTENHAM	TEAM STATS	LIVERPOOL
11	Goals	10
22	Shots	27
38	Shots on target	45
36	Possession	64
414	Passes	314
50	Pass accuracy	30
9	Fouls	12
4	Yellow cards	5
2	Red cards	2
7	Offsides	7
21	Corners	17

tottenham beat liverpool, scoring 11 goals with a possession of 36%

All clubs added into the Premier League through CLI are displayed here, the border color is based on the top color entered upon adding the club



manchester united
Sponsored By
chevrolette

leicester city
Sponsored By
kings

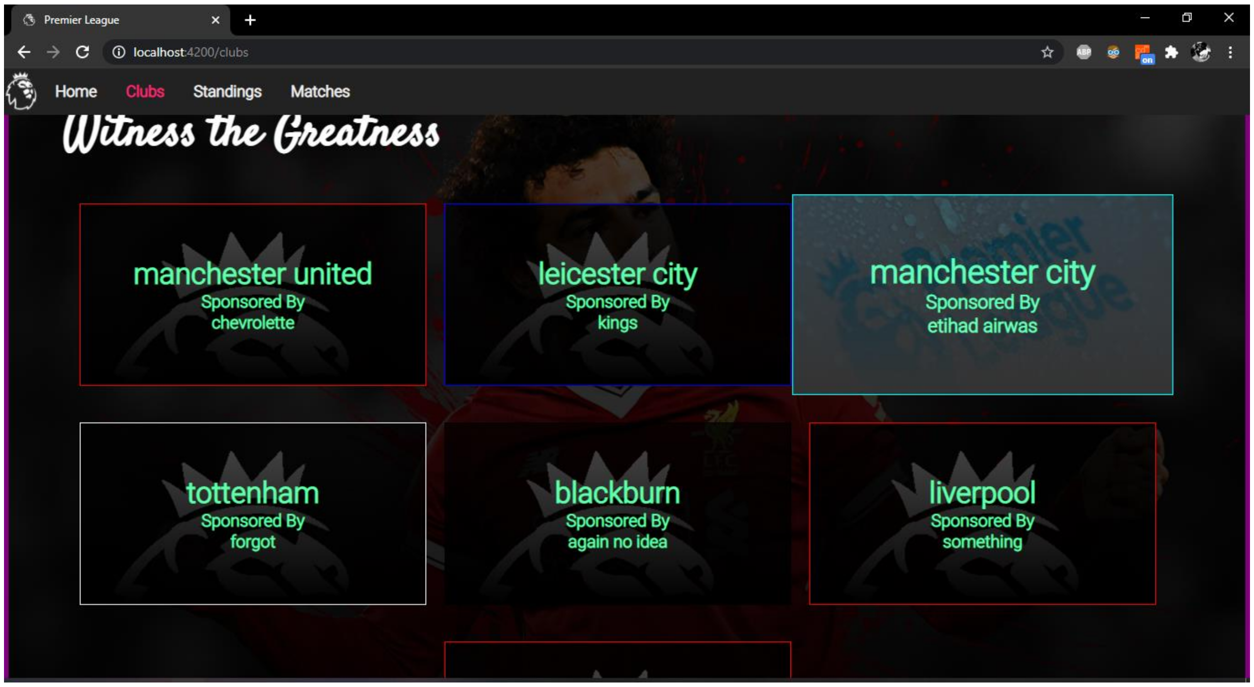
manchester city
Sponsored By
etihad airwas

tottenham
Sponsored By
forgot

blackburn
Sponsored By
again no idea

liverpool
Sponsored By
something

The bottom faded color displayed upon hover is based on the short color entered



manchester united
Sponsored By
chevrolette

leicester city
Sponsored By
kings

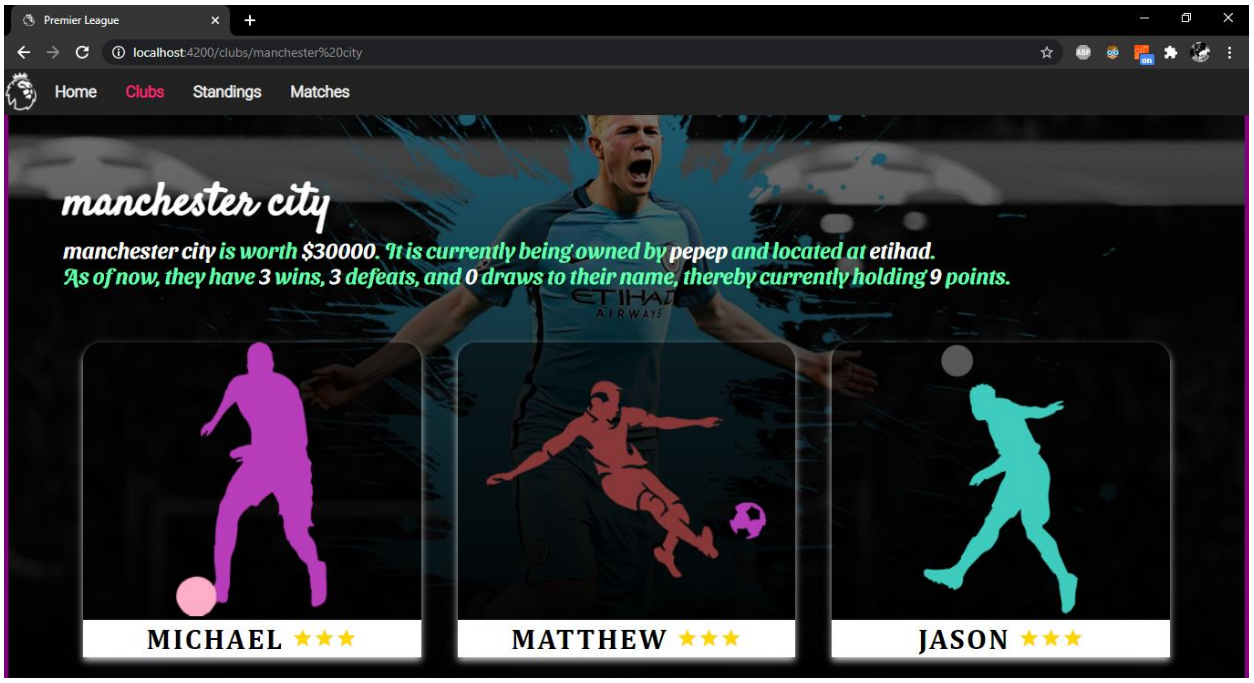
manchester city
Sponsored By
etihad airwas

tottenham
Sponsored By
forgot

blackburn
Sponsored By
again no idea

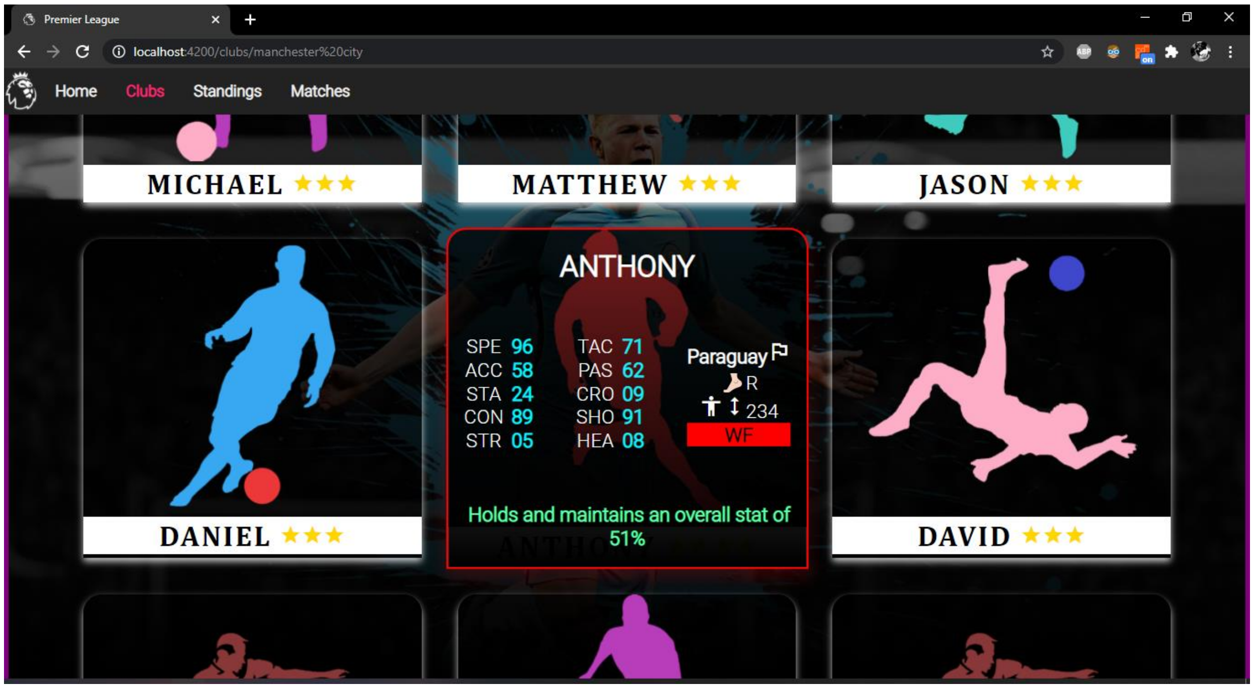
liverpool
Sponsored By
something

Information regarding each club can be accessed simply by clicking on each of the club cards, the 11 players displayed here are all generated upon the addition of a club, where their rating is based on their overall statistic



manchester city is worth $30000. It is currently being owned by pepep and located at etihad.
As of now, they have 3 wins, 3 defeats, and 0 draws to their name, thereby currently holding 9 points.

Stats about each Player can be obtained by hovering over each player. These stats are generated for each player upon the addition of a club. The display of each player stat was inspired by the famous mobile game “Dream League Soccer”

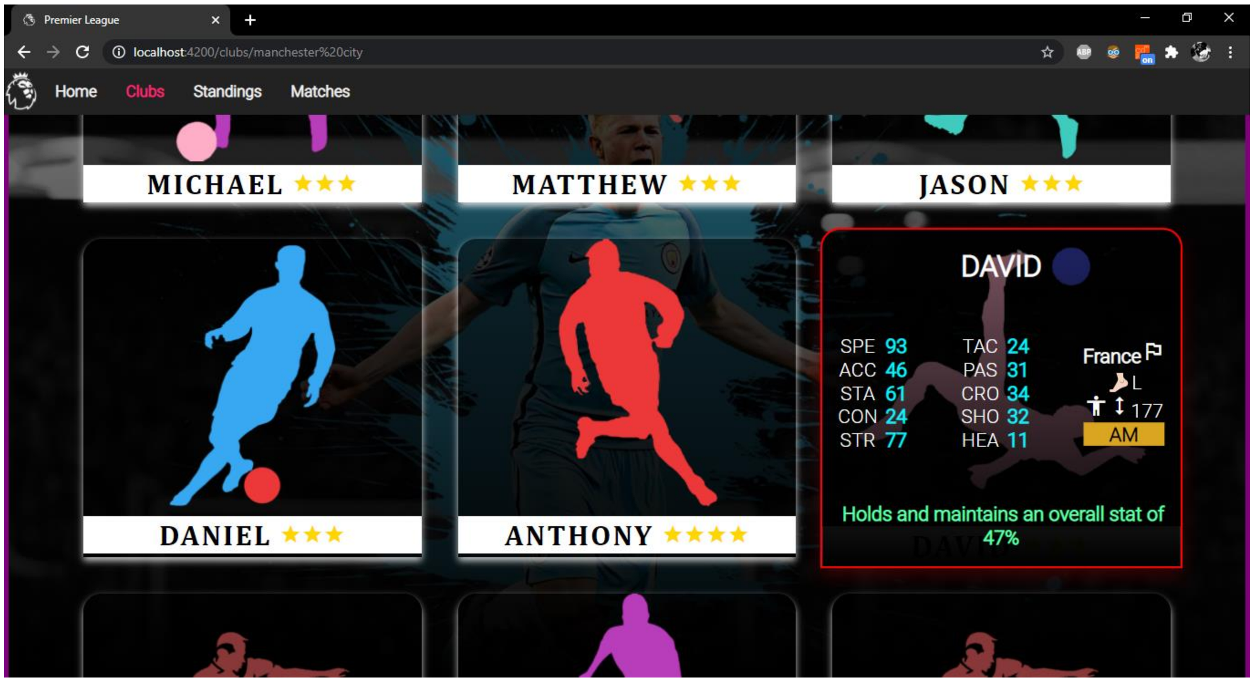


ANTHONY

SPE 96	TAC 71	Paraguay	
ACC 58	PAS 62	R	
STA 24	CRO 09	234	
CON 89	SHO 91	WF	
STR 05	HEA 08		

Holds and maintains an overall stat of 51%

The player position color is different for each type of player. If the position is anything related to defense, the color is green, for mid-field goldenrod, and for striking red.



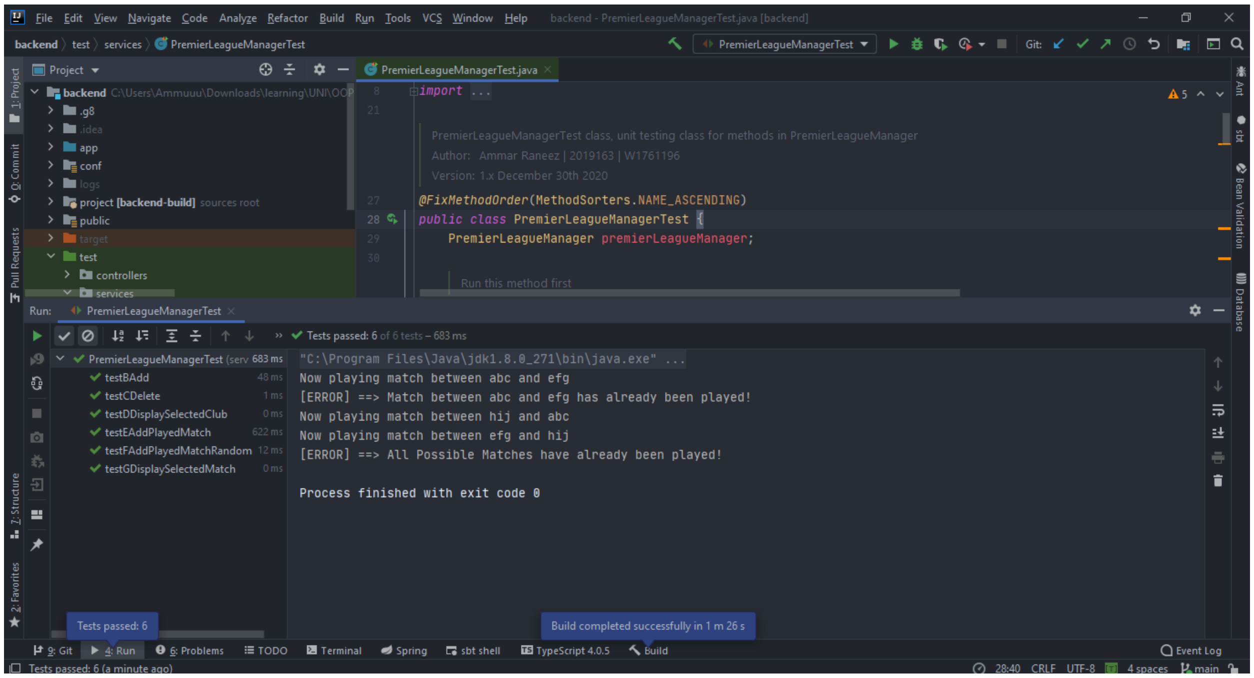
SPE 93	TAC 24	France	
ACC 46	PAS 31	L	
STA 61	CRO 34	177	
CON 24	SHO 32	AM	
STR 77	HEA 11		

Holds and maintains an overall stat of 47%

04 – JUnit Testing

PremierLeagueManagerTest class

```
package services;

/*
 * PremierLeagueManagerTest
 * Copyright © 2020 Ammar Raneez. All Rights Reserved.
 */

import coursework.models.FootballClub;
import coursework.models.FootballMatch;
import coursework.services.PremierLeagueManager;
import org.junit.Before;
import org.junit.FixMethodOrder;
import org.junit.Test;
import org.junit.runners.MethodSorters;
import java.awt.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import static org.junit.Assert.*;

/**
 * PremierLeagueManagerTest class, unit testing class for methods in PremierLeagueManager
 * @version 1.x December 30th 2020
 * @author Ammar Raneez | 2019163 | W1761196
 */
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class PremierLeagueManagerTest {
    PremierLeagueManager premierLeagueManager;



    /**
     * Run this method first
     */
    @Before
    public void testASetUp() {
        this.premierLeagueManager = new PremierLeagueManager();
    }
    /**
     * Test method for add method of PremierLeagueManager
     * A couple of clubs are added, and checked whether they have actually been added
     */
    @Test
    public void testBAdd() {
        this.premierLeagueManager.addClub("league", "abc", "abc", "abc",
                "abc", "abc", new Color(255, 0, 0), new Color(255, 0, 0),
                "abc"
            );


        this.premierLeagueManager.addClub("league", "bcd", "bcd", "bcd",
            "bcd", "bcd", new Color(255, 0, 0), new Color(255, 0, 0),
            "bcd"
        );
        //*attempt to get the first added club, if it throws an IndexOutOfBoundsException, it means
a club*//
        //*had not been added*//
        try {
            PremierLeagueManager.getAllFootballClubs().get(0);
        } catch (IndexOutOfBoundsException ignored) {
            fail("club has not been added");
        }
        //*is expected equal to actual*//
        assertEquals("abc", PremierLeagueManager.getAllFootballClubs().get(0).getClubName());
    }
    /**
     * Test method for delete
     * A club is deleted and it is checked whether the deleted club is what it is supposed to be
     */
    @Test
    public void testCDelete() {
        FootballClub deletedClub = this.premierLeagueManager.deleteClub("bcd");
        //*is the deleted club equal to the expected deleted club*//
        try {
            assertEquals("bcd", deletedClub.getClubName());
        } catch (Exception ignored) {
            fail("[ERROR] ==> Problem in delete method, club bcd exists, but was not deleted and
obtained");
        }
        //*check whether it was actually removed*//
        FootballClub fakeDeletedClub = this.premierLeagueManager.deleteClub("bcd");
        try {
            assertNull(fakeDeletedClub);
        } catch (Exception ignored) {
            fail("[ERROR] ==> Problem in delete method, club bcd does not exist anymore.");
        }
        //**check to see whether the program can handle deleting on non-existent clubs*//
        FootballClub nextDeletedClub = this.premierLeagueManager.deleteClub("non existent
club");
        try {
            assertNull(nextDeletedClub);
        } catch (Exception ignored) {
            fail("[ERROR] ==> Problem in delete method, club must be null");
        }
    }


    /**
     * Test method for display selected club
     * A preferred clubs name is passed, and it is checked whether the returned club is the
specified one
     */
    @Test
    public void testDDisplaySelectedClub() {
        FootballClub selectedClub = this.premierLeagueManager.displaySelectedClub("abc");
        try {
            assertEquals("abc", selectedClub.getClubName());
        } catch (Exception ignored) {
            fail("[ERROR] ==> Problem with display selected club method, club abc exists, but was
not obtained");
        }


        //*null checks*//
        FootballClub nextSelectedClub = this.premierLeagueManager.displaySelectedClub("non
existent club");
        try {
            assertNull(nextSelectedClub);
        } catch (Exception ignored) {
            fail("[ERROR] ==> Problem with display selected club method, club must be null");
        }
    }

    /**
     * Test method for the manual addition of a match
     */
    @Test
    public void testEAddPlayedMatch() {
        this.premierLeagueManager.addClub("league", "efg", "efg", "efg",
                "efg", "efg", new Color(255, 0, 0), new Color(255, 0, 0),
                "efg"
        );

        //*test to check whether the specified match has been played*//
        this.premierLeagueManager.addPlayedMatch("2020", "10/10", "abc", "efg", 5, 5);
        try {
            assertEquals(PremierLeagueManager.getAllMatches().get(0),
                    new FootballMatch(PremierLeagueManager.getAllFootballClubs().get(0),
PremierLeagueManager.getAllFootballClubs().get(1),
                            LocalDate.parse("10/10/2020",
DateTimeFormatter.ofPattern("dd/MM/yyyy")))
            );
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with add played match method, match was not played");
        }

        this.premierLeagueManager.addClub("league", "hij", "hij", "hij",
                "hij", "hij", new Color(255, 0, 0), new Color(255, 0, 0),
                "hij"
        );

        //**check to see whether the program handles the prevention of repeated matches
        this.premierLeagueManager.addPlayedMatch("2020", "10/10", "abc", "efg", 5, 5);
        try {
            assertThrows(IndexOutOfBoundsException.class, () -> {
                PremierLeagueManager.getAllMatches().get(1);
            });
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with add played match method, a match was not supposed
to have been played, since this match has already been played");
        }
    }

    /**
     * Test method for the randomly generated match
     */
    @Test
    public void testFAddPlayedMatchRandom() {
        //*check to see whether all possible matches do play*//
        this.premierLeagueManager.addPlayedMatchRandom("2020");
        try {
            assertNotNull(PremierLeagueManager.getAllMatches().get(1));
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with add played match method there are 3 clubs, therefore
there are 3 total playable matches, 2 more can be played");
        }
        this.premierLeagueManager.addPlayedMatchRandom("2020");
//        System.out.println(PremierLeagueManager.getAllMatches().size());
        try {
            assertNotNull(PremierLeagueManager.getAllMatches().get(2));
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with add played match method there are 3 clubs, therefore
there are 3 total playable matches, 1 more can be played");
        }


        //**check to see whether the program knows when all matches have already been played*//
        this.premierLeagueManager.addPlayedMatchRandom("2020");
        try {
            assertThrows(IndexOutOfBoundsException.class, () -> {
                PremierLeagueManager.getAllMatches().get(3);
            });
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with add played match method, a match was not supposed
to have been played, since all matches hav already been played");
        }
    }


    /**
     * Test for displaying of a preferred match
     */
    @Test
    public void testGDisplaySelectedMatch() {
        //*tests to see whether a played match exists*//
        try {
            assertEquals(this.premierLeagueManager.displaySelectedMatch("abc", "efg"),
PremierLeagueManager.getAllMatches().get(0));
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with display selected match method, this match has been
played");
        }


        //*check to see whether the order of club inputs do not matter*//
        try {
            assertEquals(this.premierLeagueManager.displaySelectedMatch("efg", "abc"),
PremierLeagueManager.getAllMatches().get(0));
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with display selected match method, this match has been
played, " +
                    "there's a problem with the equals method, the order of the clubs involved in the
match should not matter");
        }


        try {
            assertNull(this.premierLeagueManager.displaySelectedMatch("non-existent", "non-
existent-2"));
        } catch (Exception ignored) {
            fail("[ERROR] ==> problem with display selected match method, this match has not
been played, and" +
                    " is supposed to return null ");
        }
    }
}
```



```java
import ...

    PremierLeagueManagerTest class, unit testing class for methods in PremierLeagueManager
    Author:  Ammar Raneez | 2019163 | W1761196
    Version: 1.x December 30th 2020
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class PremierLeagueManagerTest {
    PremierLeagueManager premierLeagueManager;

        Run this method first
```

```
Tests passed: 6 of 6 tests – 683 ms

PremierLeagueManagerTest       683 ms
    testBAdd                    48 ms
    testCDelete                  1 ms
    testDDisplaySelectedClub     0 ms
    testEAddPlayedMatch        622 ms
    testFAddPlayedMatchRandom   12 ms
    testGDisplaySelectedMatch    0 ms
```

```
"C:\Program Files\Java\jdk1.8.0_271\bin\java.exe" ...
Now playing match between abc and efg
[ERROR] ==> Match between abc and efg has already been played!
Now playing match between hij and abc
Now playing match between efg and hij
[ERROR] ==> All Possible Matches have already been played!

Process finished with exit code 0
```

Tests passed: 6

Build completed successfully in 1 m 26 s

Tests passed: 6 (a minute ago)