Informatics Institute of Technology
in collaboration with
University of Westminster, UK

5COSC007C.1 Object Oriented Programming

Student Name	Mohamed Rumy Mohamed Infaz
IIT ID	2019818
UOW ID	w1761153
Course	BSc. Computer Science

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

Table of Contents

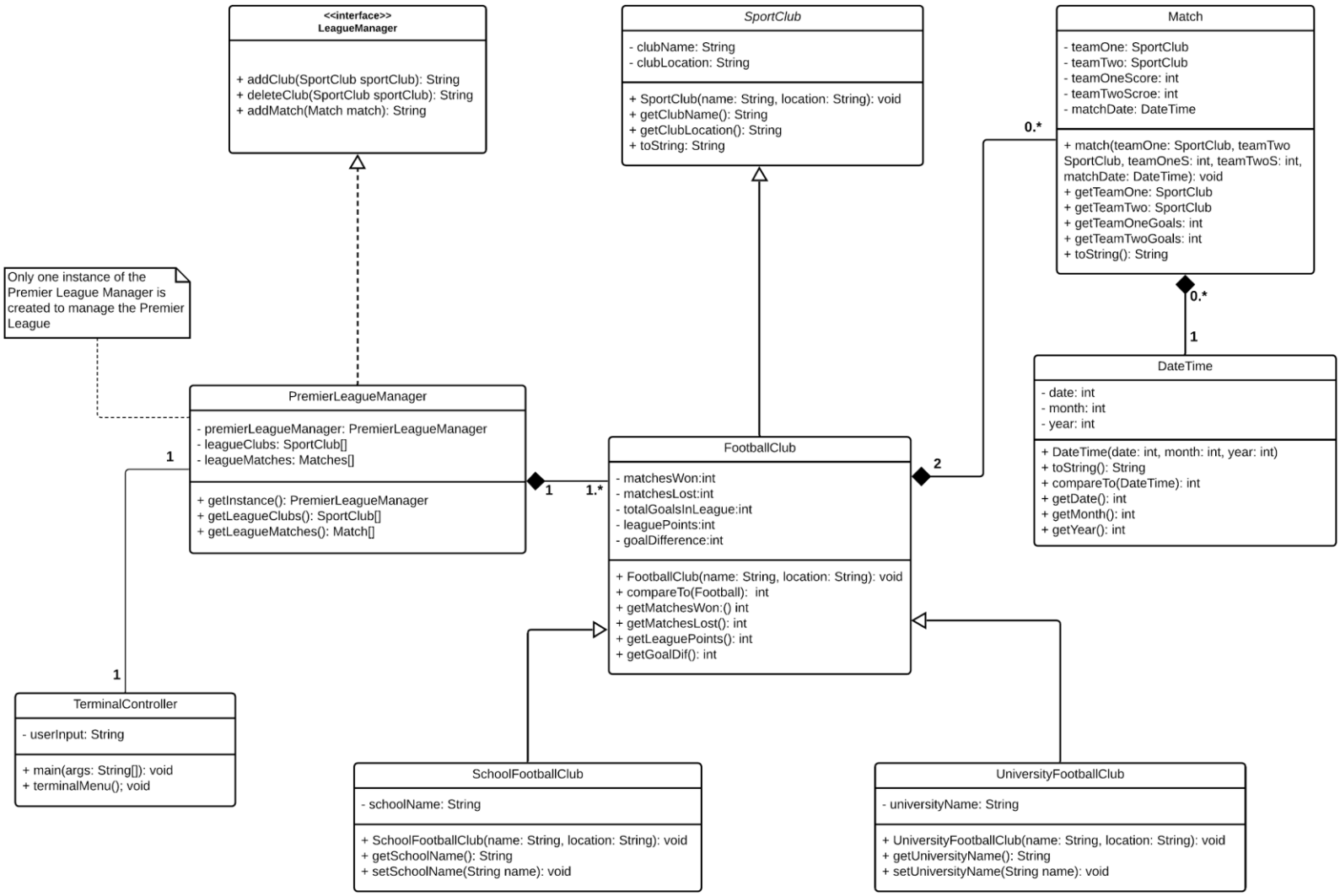
1. Assumptions made for the implementation

Assumption: One Premier League Manager is responsible for managing the Premier League (add a Football Club, delete a Football Club, add a Match)

Assumption: In the Points table if two Football clubs are having same points the team should come first is decided by the goal difference of each team.

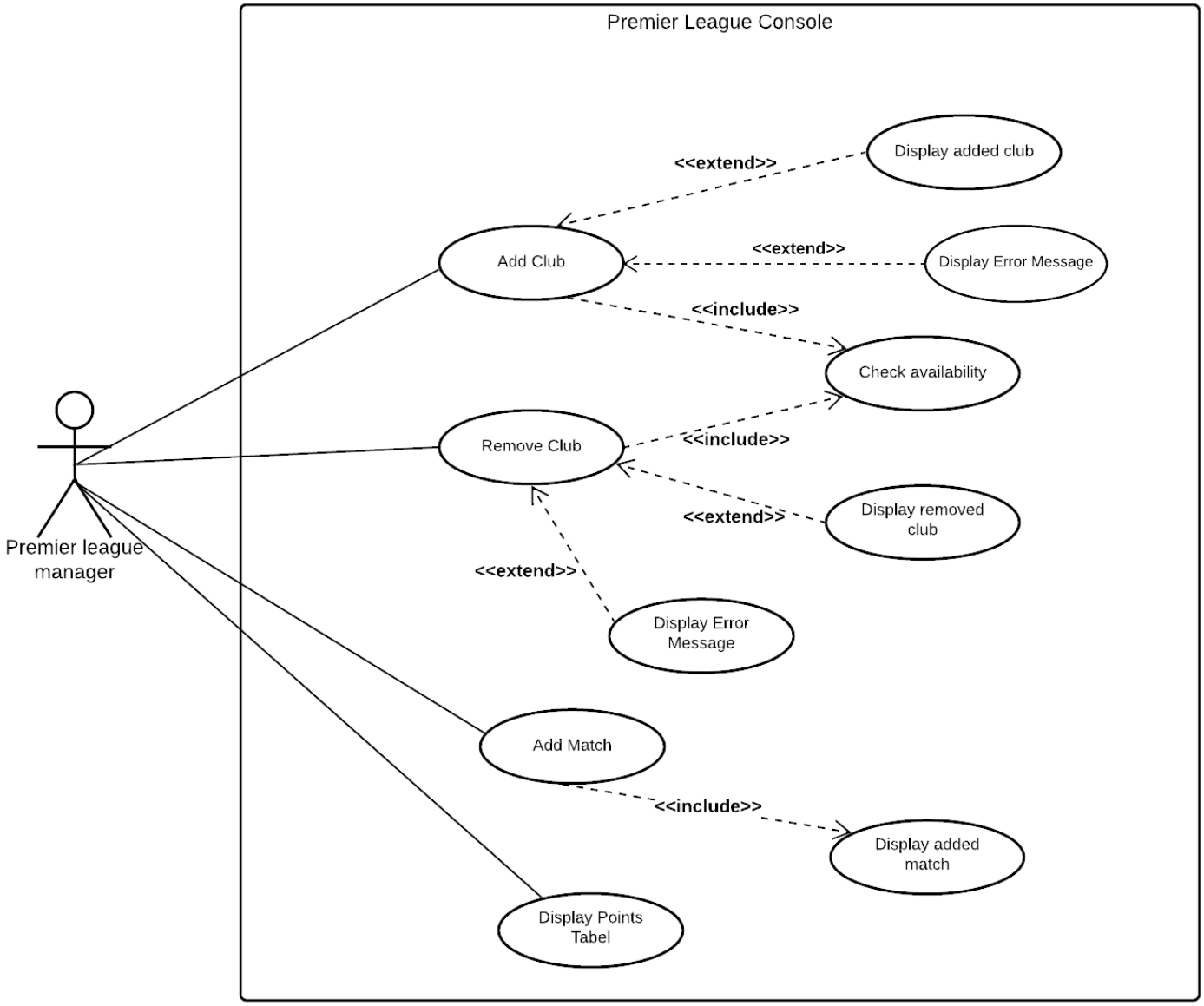
Goal Difference of a team = total numbers of goals relevant club scored in the league – total number of goals scored in the league against the team.

2. UML Class Diagram

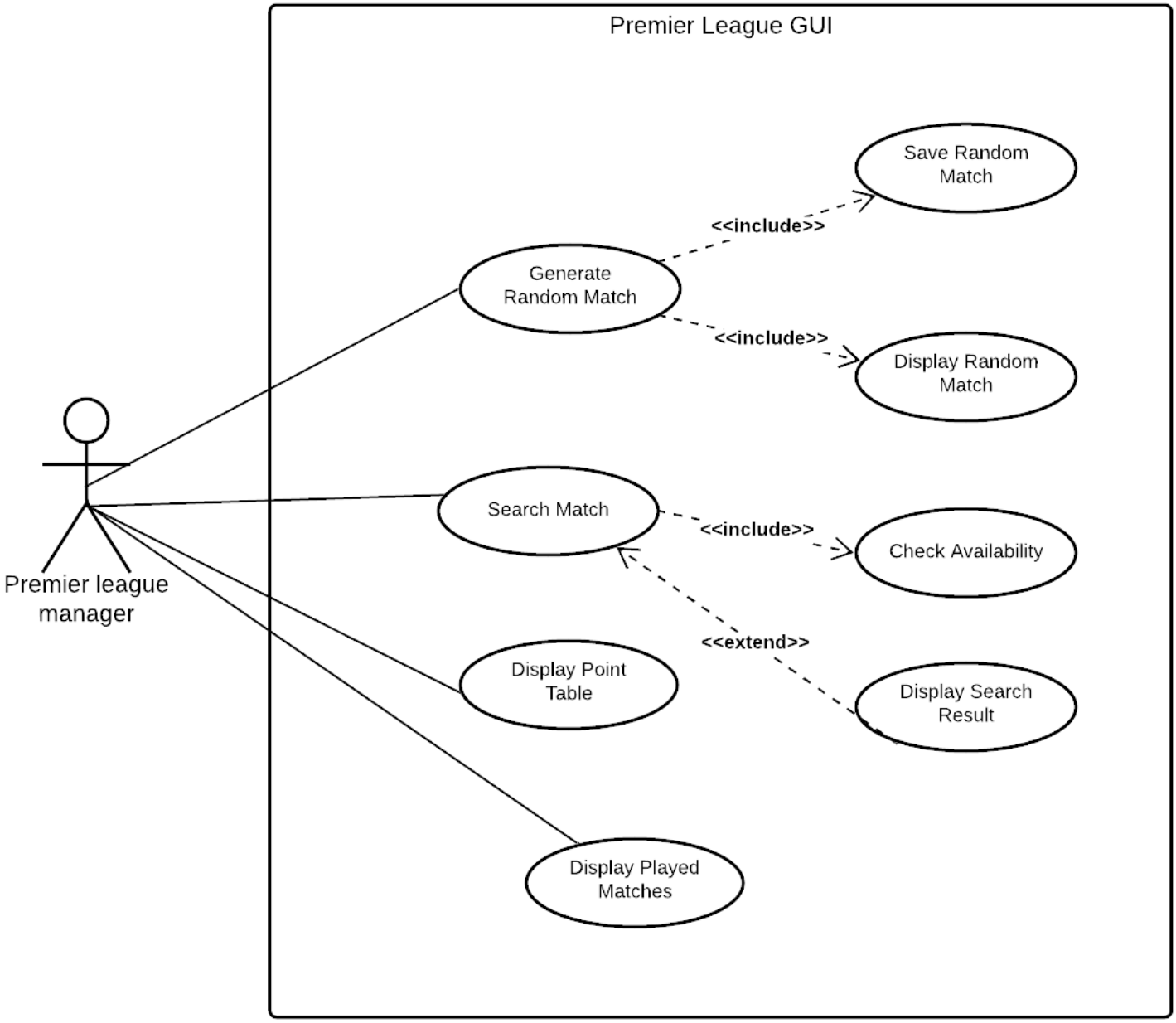


3. Use Case Diagram

1. Diagram One



2. Diagram Two



4. Test Case Coverage

Test Number	Test Case	Expected Result	Actual Result
T1	Add a Football Club	Chelsea added to the league successfully	Chelsea added to the league successfully
T2	Try to add an existing Football Club	Could not add the Club to the League	Could not add the Club to the League
T3	Remove a Football Club from the League	Chelsea has been removed successfully	Chelsea has been removed successfully
T4	Try to Remove a Club not in the League	Couldn't delete the Club	Couldn't delete the Club
T5	Add a Match to the League	{ Team One: Barcelona Score: 2 Team Two: Manchester City Score: 3 Match Date: 2/12/2021 } added successfully	{ Team One: Barcelona Score: 2 Team Two: Manchester City Score: 3 Match Date: 2/12/2021 } added successfully
T6	Saving League Information	League Information saved to a .ser format file	League Information saved to a ser format file
T7	Load League Information	League information loaded from .ser format file	League information loaded from .ser format file

Output of the Test Cases

Source Code for Test Case Coverage

```
public class TerminalControllerTest extends WithApplication {

    @Override
    protected Application provideApplication() {
        return new GuiceApplicationBuilder().build();
    }

    PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();

    String clubOneName = "Liverpool";
    String clubOneLocation = "London";
    String clubTwoName = "Chelsea";
    String clubTwoLocation = "London";

    String falseClubName = "FalseClubName";

    SportClub sportClubOne = new Football(clubOneName, clubOneLocation);
    SportClub sportClubTwo = new Football(clubTwoName, clubTwoLocation);

    DateTime dateTime = new DateTime(12, 2,2021);

    Match match = new Match(sportClubOne, sportClubTwo,dateTime,1,3);

    @Test
    public void addClub(){
        String message =
premierLeagueManager.addSportClub(this.sportClubOne);
        assertEquals(sportClubOne.getClubName() + " added to the league
successfully\n", message);
    }

    @Test
    public void tryToAddExistingClub(){
        premierLeagueManager.addSportClub(this.sportClubOne);
        String message =
premierLeagueManager.addSportClub(this.sportClubOne);
        assertEquals("Could not add the Club to the League\n", message);
    }

    @Test
    public void removeClub(){
        String message = premierLeagueManager.removeSportClub(clubOneName);
        assertEquals(clubOneName + " has been removed
successfully\n",message);
    }

    @Test
    public void removeNotExistingClub(){
        String message =
premierLeagueManager.removeSportClub(this.falseClubName);
        assertEquals("Could not remove " + this.falseClubName +
"\n",message);
    }
```

```
    @Test
    public void addMatch(){
        String message = premierLeagueManager.addMatch(match);
        assertEquals("{ Team One: " + this.match.getTeamOne() + " Score: " +
this.match.getTeamOneGoals() + " Team Two: " +
                this.match.getTeamTwo() + " Score: "+
this.match.getTeamTwoGoals() +" Match Date: "+ this.dateTime + " } added
successfully\n",message);
    }

    @Test
    public void saveLeagueInformation(){
        premierLeagueManager.addSportClub(sportClubOne);
        premierLeagueManager.addMatch(match);

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
    }

    @Test
    public void loadLeagueInformation(){
        premierLeagueManager.addSportClub(sportClubOne);
        premierLeagueManager.addMatch(match);

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
        FileHandlingService.loadLeagueInformation();
    }
}
```

5. Source Code

Play Source Code

ILeagueManager.java

```
package entities;

import entities.Match;
import entities.SportClub;

public interface ILeagueManager {

/**
 * Add a sport club to the league
 * @param sportClub Sport club needed to add to the list
 * @return String message of added club
 */
String addSportClub(SportClub sportClub);

/**
 * Remove a sport club from the league
 * @param clubName Sport club needed to remove from the list
 * @return String message of removed club
 */
String removeSportClub(String clubName);

/**
 * Add a match to the league matches
 * @param match Match needed to add to the match list
 * @return String message of added match
 */
String addMatch(Match match);

}
```

PremierLeagueManager.java

```
package entities;

import utils.ApplicationUtils;

import java.util.ArrayList;

/**
 * <b>assumption</b> - there can be only one premier league manager to manage
 * the premier league. used Singleton design pattern to implement the
assumption.
 */

public class PremierLeagueManager implements ILeagueManager {

    private static PremierLeagueManager premierLeagueManager = new
```

```
PremierLeagueManager();

    private PremierLeagueManager() {
    };

    public static PremierLeagueManager getInstance() {
        return premierLeagueManager;
    }

    private ArrayList<SportClub> footballClub = new ArrayList<>();
    private ArrayList<Match> leagueMatches = new ArrayList<>();

    public ArrayList<SportClub> getFootballClub() {
        return footballClub;
    }

    public void setFootballClub(ArrayList<SportClub> footballClub) {
        this.footballClub = footballClub;
    }

    public ArrayList<Match> getLeagueMatches() {
        return leagueMatches;
    }

    public void setLeagueMatches(ArrayList<Match> leagueMatches) {
        this.leagueMatches = leagueMatches;
    }

    @Override
    public String addSportClub(SportClub sportClub) {
        if(!ApplicationUtils.isClubAvailable(sportClub.getClubName())){
            footballClub.add(sportClub);
            return sportClub.getClubName() + " added to the league
successfully\n";
        }else{
            return "Could not add the Club to the League\n";
        }
    }

    @Override
    public String removeSportClub(String clubName) {

        for(SportClub sportClub: footballClub){
            if(sportClub.getClubName().equals(clubName)){
                this.footballClub.remove(sportClub);
                return sportClub.getClubName() + " has been removed
successfully\n";
            }
        }
        return "Could not remove " + clubName + "\n";
    }

    @Override
    public String addMatch(Match match) {
        this.leagueMatches.add(match);
        return "{ " + match + " } added successfully\n";
```

```
    }
}
```

SportClub.java

```
package entities;

import java.io.Serializable;

public abstract class SportClub implements Serializable{

   private String clubName;
   private String clubLocation;

   /**
    * set the club name and the location
    * @param clubName name of the sport club
    * @param clubLocation location of the sport club
    */
    public SportClub(String clubName, String clubLocation){
        this.clubName = clubName;
        this.clubLocation = clubLocation;
    }


   /**
    * @return name of the sport club
    */
   public String getClubName() {
    return this.clubName;
   }

    /**
     * @return location of the sport club
     */
   public String getClubLocation() {
       return this.clubLocation;
   }
}
```

FootballClub.java

```
package entities;

import java.io.Serializable;

public class Football extends SportClub implements Comparable<Football>,
Serializable {

    private int matchesWon;
    private int matchesLost;
    private int leaguePoints;
    private int totalLeagueGoals;
    private int goalDifference;
```

```
public Football(String clubName, String clubLocation){
    super(clubName, clubLocation);
    this.leaguePoints = 0;
    this.matchesLost = 0;
    this.matchesWon = 0;
    this.totalLeagueGoals = 0;
    this.goalDifference = 0;
}

public int getGoalDifference() {
    return goalDifference;
}

public int getTotalLeagueGoals() {
    return totalLeagueGoals;
}

public int getMatchesWon() {
    return matchesWon;
}

public int getMatchesLost() {
    return matchesLost;
}

public int getLeaguePoints() {
    return leaguePoints;
}

public void setMatchesWon(int matchesWon) {
    this.matchesWon = matchesWon;
}

public void setMatchesLost(int matchesLost) {
    this.matchesLost = matchesLost;
}

public void setLeaguePoints(int leaguePoints) {
    this.leaguePoints = leaguePoints;
}

public void setTotalLeagueGoals(int totalLeagueGoals) {
    this.totalLeagueGoals = totalLeagueGoals;
}

public void setGoalDifference(int goalDifference) {
    this.goalDifference = goalDifference;
}

@Override
public String toString() {
    return super.getClubName();
}

@Override
public int compareTo(Football arg0) {
    return this.leaguePoints - arg0.leaguePoints;
```

```
    }
}
```

SchoolFootballClub.java

```
package entities;

public class SchoolFootballClub {

    private String schoolName;

    public String getSchoolName() {
        return schoolName;
    }

    public void setSchoolName(String schoolName) {
        this.schoolName = schoolName;
    }

    @Override
    public String toString() {
        return "SchoolFootballClub{" +
                "schoolName='" + schoolName + '\'' +
                '}';
    }
}
```

UniversityFootballClub.java

```
package entities;

public class UniversityFootballClub {

    private String universityName;

    public String getUniversityName() {
        return universityName;
    }

    public void setUniversityName(String universityName) {
        this.universityName = universityName;
    }

    @Override
    public String toString() {
        return "UniversityFootballClub{" +
                "universityName='" + universityName + '\'' +
                '}';
    }
}
```

Match.java

```
package entities;
```

```
import java.io.Serializable;

public class Match implements Serializable{

    private int teamOneGoals;
    private int teamTwoGoals;
    private SportClub teamOne;
    private SportClub teamTwo;
    private DateTime matchDate;

    public Match(SportClub teamOne, SportClub teamTwo, DateTime matchDate,
int teamOneGoals, int teamTwoGoals){

        this.teamOne = teamOne;
        this.teamTwo = teamTwo;
        this.matchDate = matchDate;
        this.teamOneGoals = teamOneGoals;
        this.teamTwoGoals = teamTwoGoals;

        Football footballClubOne = (Football) teamOne;

footballClubOne.setTotalLeagueGoals(footballClubOne.getTotalLeagueGoals() +
teamOneGoals);

footballClubOne.setGoalDifference(footballClubOne.getTotalLeagueGoals() -
teamTwoGoals);

        Football footballClubTwo = (Football) teamTwo;

footballClubTwo.setTotalLeagueGoals(footballClubTwo.getTotalLeagueGoals() +
teamTwoGoals);

footballClubTwo.setGoalDifference(footballClubTwo.getTotalLeagueGoals() -
teamOneGoals);

        if(teamOneGoals > teamTwoGoals){

            footballClubOne.setLeaguePoints(footballClubOne.getLeaguePoints()
+ 1);
            footballClubOne.setMatchesWon(footballClubOne.getMatchesWon() +
1);

        }else if (teamOneGoals < teamTwoGoals){

            footballClubTwo.setLeaguePoints(footballClubTwo.getLeaguePoints()
+ 1);
            footballClubTwo.setMatchesLost(footballClubTwo.getMatchesLost() +
1);

        }
    }

    public DateTime getMatchDate() {
        return matchDate;
    }
```

```
    public int getTeamOneGoals() {
        return teamOneGoals;
    }

    public int getTeamTwoGoals() {
        return teamTwoGoals;
    }

    public void setMatchDate(DateTime matchDate) {
        this.matchDate = matchDate;
    }

    public void setTeamOne(Football teamOne) {
        this.teamOne = teamOne;
    }

    public void setTeamTwo(Football teamTwo) {
        this.teamTwo = teamTwo;
    }

    public void setTeamOneGoals(int teamOneGoals) {
        this.teamOneGoals = teamOneGoals;
    }

    public void setTeamTwoGoals(int teamTwoGoals) {
        this.teamTwoGoals = teamTwoGoals;
    }

    public SportClub getTeamOne() {
        return teamOne;
    }

    public void setTeamOne(SportClub teamOne) {
        this.teamOne = teamOne;
    }

    public SportClub getTeamTwo() {
        return teamTwo;
    }

    public void setTeamTwo(SportClub teamTwo) {
        this.teamTwo = teamTwo;
    }

    @Override
    public String toString() {
        return "Team One: " + teamOne + " Score: " + teamOneGoals +
        " Team Two: " + teamTwo + " Score: " + teamTwoGoals + " Match Date: "
+ matchDate;
    }
}
```

DateTime.java

```
package entities;
import java.io.Serializable;

public class DateTime implements Comparable<DateTime>, Serializable{

    private int date;
    private int month;
    private int year;

    public DateTime(int date, int month, int year){
        this.date = date;
        this.month = month;
        this.year = year;
    }

    public int getDate() {
        return date;
    }

    public int getMonth() {
        return month;
    }

    public int getYear() {
        return year;
    }

    @Override
    public String toString() {
        return String.valueOf(date+"/"+month+"/"+year);
    }

    @Override
    public int compareTo(DateTime dateTime) {
        return this.date - dateTime.date;
    }
}
```

ApplicationsUtils.java

```
package utils;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import entities.PremierLeagueManager;
import entities.DateTime;
import entities.Football;
import entities.Match;
import entities.SportClub;
import play.libs.Json;
import java.util.ArrayList;
import java.util.Scanner;
```

```java
/**
 * Utils class contains the statics methods that is used for methods
 * that is required to prompt user input etc.
 */
public class ApplicationUtils {

    public static Scanner scanner = new Scanner(System.in);


    /**
     * prompt for user input and return a football object based on the input
     * @return Football club object
     */
    public static SportClub collectingClubInformation(){

        SportClub sportClub;
        String clubName, clubLocation;

        System.out.print("Please enter club name: ");
        clubName = scanner.nextLine();

        while(clubName.equals("")){
            System.out.println("Invalid Name !");

            System.out.print("Please enter club name: ");
            clubName = scanner.nextLine();
        }


        System.out.print("Please enter club location: ");
        clubLocation = scanner.nextLine();

        while(clubLocation.equals("")){
            System.out.println("Invalid Club Location !");

            System.out.print("Please enter club location: ");
            clubLocation = scanner.nextLine();
        }
        sportClub = new Football(clubName, clubLocation);
        return sportClub;
    }



    /**
     * prompt for user input and return a match object based on the input
     * @return Match object
     */
    public static Match collectingMatchInformation(){

        Match match;
        SportClub sportClubOne, sportClubTwo;
        int date, month, year;
        DateTime matchDate;
        int teamOneGoals, teamTwoGoals;

        PremierLeagueManager premierLeagueManager  =
```

```
PremierLeagueManager.getInstance();

        for(SportClub sportClub: premierLeagueManager.getFootballClub()){

System.out.println(premierLeagueManager.getFootballClub().indexOf(sportClub)+
" - " + sportClub);
        }

        System.out.print("Please Select a Club (Club ID): ");
        sportClubOne =
premierLeagueManager.getFootballClub().get(scanner.nextInt());

        for(SportClub sportClub: premierLeagueManager.getFootballClub()){
            if(!sportClub.equals(sportClubOne)){

System.out.println(premierLeagueManager.getFootballClub().indexOf(sportClub)+
" - " + sportClub);
            }
        }
        System.out.print("Please Select a Club (Club ID): ");
        sportClubTwo =
premierLeagueManager.getFootballClub().get(scanner.nextInt());

        System.out.print("Please enter the goals scored by " +
sportClubOne.getClubName()+ ":" );
        teamOneGoals = scanner.nextInt();

        while(sportClubOne.equals(sportClubTwo)){
            System.out.println("Select two different teams !");
            for(SportClub sportClub: premierLeagueManager.getFootballClub()){
                if(!sportClub.equals(sportClubOne)){

System.out.println(premierLeagueManager.getFootballClub().indexOf(sportClub)+
" - " + sportClub);
                }
            }
            System.out.print("Please Select a Club (Club ID): ");
            sportClubTwo =
premierLeagueManager.getFootballClub().get(scanner.nextInt());
        }

        System.out.print("Please enter the goals scored by " +
sportClubTwo.getClubName()+ ":" );
        teamTwoGoals = scanner.nextInt();

        System.out.print("Please enter match date: ");
        date = scanner.nextInt();

        System.out.print("Please enter match month: ");
        month = scanner.nextInt();

        System.out.print("Please enter match year: ");
        year = scanner.nextInt();

        matchDate = new DateTime(date, month, year);

        match = new Match(sportClubOne, sportClubTwo, matchDate,
```

```
teamOneGoals, teamTwoGoals);
        return match;
    }

    /**
     * Return the sport club based on given name
     * @param clubName string value of the club name
     * @return sport club object
     */
    public static SportClub findClubByName(String clubName){
        PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();
        SportClub sportClub = null;
        for(SportClub football: premierLeagueManager.getFootballClub()){
            if(football.getClubName().equals(clubName)){
                sportClub = football;
            }
        }
        return sportClub;
    }

    public static ObjectNode createResponse(Object response, boolean ok) {
        ObjectNode result = Json.newObject();
        result.put("status", ok);
        if (response instanceof String)
            result.put("response", (String) response);
        else result.set("response", (JsonNode) response);

        return result;
    }

    public static Match randomMatch() {

        PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();
        ArrayList<SportClub> footballClubs =
premierLeagueManager.getFootballClub();

        int min = 0;
        int max = footballClubs.size() - 1;

        int teamOne = (int)(Math.random() * (max - min + 1) + min);
        int teamTwo = (int)(Math.random() * (max - min + 1) + min);

        while (teamTwo == teamOne){
            if(teamOne == teamTwo){
                teamTwo = (int)(Math.random() * (max - min + 1) + min);
            }
        }

        int teamOneScore = (int)(Math.random() * (3) + min);
        int teamTwoScore = (int)(Math.random() * (3) + min);

        int date = (int) (Math.random() * (31 - 1 + 1) + 1);
        int month = (int) (Math.random() * (12 - 1 + 1) + 1);
        int year = 2021;
```

```
        DateTime dateTime = new DateTime(date, month, year);

        Match match = new
Match(footballClubs.get(teamOne),footballClubs.get(teamTwo), dateTime,
teamOneScore, teamTwoScore);
        return match;
    }

    public static boolean isClubAvailable(String clubName){
        PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();
        boolean isClubAvailable = false;
        for(SportClub football: premierLeagueManager.getFootballClub()){
            if(football.getClubName().equalsIgnoreCase(clubName)){
                isClubAvailable = true;
            }
        }
        return isClubAvailable;
    }

}
```

FileHandlingService.java

```
package services;

import entities.PremierLeagueManager;
import entities.Match;
import entities.SportClub;
import java.io.*;
import java.util.ArrayList;

public class FileHandlingService {

    /**
     * saving premier league matches and clubs into a file in '.ser' format
     * @param sportClubs getting list of clubs playing in the league as
parameter
     * @param matches getting list of played matches in the league as
parameter
     */
    public static void saveLeagueInformation(ArrayList<SportClub> sportClubs,
ArrayList<Match> matches){
        try{
            FileOutputStream writeData;
            ObjectOutputStream writeStream;

            writeData = new FileOutputStream("clubs.ser");
            writeStream = new ObjectOutputStream(writeData);
            writeStream.writeObject(sportClubs);

            writeData = new FileOutputStream("matches.ser");
            writeStream = new ObjectOutputStream(writeData);
            writeStream.writeObject(matches);

            writeStream.flush();
```

```
            writeStream.close();
            writeData.flush();
            writeData.close();

        }catch(IOException e){
            e.printStackTrace();
        }
    }

    /**
     * load premier league clubs and matches information into premier league
managers' instances
     */
    public static void load() throws IOException {
        PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();

        FileInputStream fileInputStream;
        ObjectInputStream objectInputStream;

        while (true){
            try{
                fileInputStream = new FileInputStream("clubs.ser");
                objectInputStream = new ObjectInputStream(fileInputStream);
                ArrayList<SportClub> footballClub =
premierLeagueManager.getFootballClub();

                footballClub = (ArrayList<SportClub>)
objectInputStream.readObject();
                premierLeagueManager.setFootballClub(footballClub);

                fileInputStream = new FileInputStream("matches.ser");
                objectInputStream = new ObjectInputStream(fileInputStream);

                ArrayList<Match> matches =
premierLeagueManager.getLeagueMatches();

                matches = (ArrayList<Match>) objectInputStream.readObject();
                premierLeagueManager.setLeagueMatches(matches);

                objectInputStream.close();
                fileInputStream.close();
                break;
            }catch (EOFException | ClassNotFoundException |
FileNotFoundException e){
                break;
            }
        }
    }
}
```

APIController.java

```
package controllers;

import com.fasterxml.jackson.databind.JsonNode;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import entities.*;
import play.libs.Json;
import play.mvc.Controller;
import play.mvc.Http;
import play.mvc.Result;
import services.FileHandlingService;
import utils.ApplicationUtils;

import java.io.IOException;
import java.util.ArrayList;

/**
 * This class handles all the API requests of premier league manager
 */

public class APIController extends Controller {

    PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();

    public Result listClubs() throws IOException {

        FileHandlingService.load();
        ArrayList<SportClub> result = premierLeagueManager.getFootballClub();

        ClubRankingComparator clubRankingComparator = new
ClubRankingComparator();
        premierLeagueManager.getFootballClub().sort(clubRankingComparator);

        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode jsonNode = objectMapper.convertValue(result,JsonNode.class);
        return ok(jsonNode);
    }

    public Result listMatches() throws IOException {

        FileHandlingService.load();
        ArrayList<Match> result = premierLeagueManager.getLeagueMatches();

        DateOrderComparator dateOrderComparator = new DateOrderComparator();
        premierLeagueManager.getLeagueMatches().sort(dateOrderComparator);

        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode jsonNode = objectMapper.convertValue(result,
JsonNode.class);
        return ok(jsonNode);
    }

    public Result sendRandomMatch(){

        Match match = ApplicationUtils.randomMatch();
        premierLeagueManager.addMatch(match);

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
        ObjectMapper objectMapper = new ObjectMapper();
```

```
        JsonNode jsonNode = objectMapper.convertValue(match,JsonNode.class);
        return ok(jsonNode);

    }

    //This method can be used if need to add a match through client side
    public Result addMatch(Http.Request request){
        JsonNode json = request.body().asJson();

        if(json == null){
            return badRequest(ApplicationUtils.createResponse("Could not
complete",false));
        }else {
            SportClub sportClubOne =
ApplicationUtils.findClubByName(json.get("teamOne").asText());
            SportClub sportClubTwo =
ApplicationUtils.findClubByName(json.get("teamTwo").asText());
            int teamOneGoals = json.get("teamOneGoals").asInt();
            int teamTwoGoals = json.get("teamTwoGoals").asInt();

            int matchDate = json.get("matchDate").asInt();
            int matchMonth = json.get("matchMonth").asInt();
            int matchYear = json.get("matchYear").asInt();

            DateTime matchDay = new DateTime(matchDate, matchMonth,
matchYear);
            Match match = new Match(sportClubOne, sportClubTwo, matchDay,
teamOneGoals, teamTwoGoals);
            String message = premierLeagueManager.addMatch(match);
            return ok(ApplicationUtils.createResponse(message,true));
        }
    }

    //This method can be used if need to add a sport club through client side
    public Result addClub(Http.Request request){

        JsonNode json = request.body().asJson();

        if(json == null){
            return badRequest(ApplicationUtils.createResponse("Could not
complete",false));
        }else {
            SportClub sportClub = Json.fromJson(json, Football.class);
            String message = premierLeagueManager.addSportClub(sportClub);
            return ok(ApplicationUtils.createResponse(message,true));
        }
    }
}
```

ClubRankingComparator.java

```
package controllers;

import entities.*;
```

```
import java.util.Comparator;

public class ClubRankingComparator implements Comparator<SportClub>{

    @Override
    public int compare(SportClub arg0, SportClub arg1) {
        Football footballOne = (Football) arg0;
        Football footballTwo = (Football) arg1;

        if((footballOne.getLeaguePoints() - footballTwo.getLeaguePoints()) !=
0){
            return footballTwo.getLeaguePoints() -
footballOne.getLeaguePoints();
        }else{
            return footballTwo.getGoalDifference() -
footballOne.getGoalDifference();
        }

    }

}
```

DateOrderComparator.java

```
package controllers;

import entities.*;

import java.util.Comparator;

public class DateOrderComparator implements Comparator<Match> {

    @Override
    public int compare(Match matchOne, Match matchTwo) {

        if((matchOne.getMatchDate().getDate() -
matchTwo.getMatchDate().getDate()) != 0){
            return matchOne.getMatchDate().getMonth() -
matchTwo.getMatchDate().getMonth();
        }else{
            return matchOne.getMatchDate().getDate() -
matchTwo.getMatchDate().getDate();
        }
    }
}
```

TerminalController.java

```
package controllers;

import entities.Football;
import entities.Match;
import entities.PremierLeagueManager;
import entities.SportClub;
import services.FileHandlingService;
import utils.ApplicationUtils;
```

```
import java.io.IOException;
import java.util.Scanner;

public class TerminalController {

    public static void main(String[] args) throws IOException {
        terminalMenu();
    }

    public static void terminalMenu() throws IOException {

        //loading league information at the start of the program
        FileHandlingService.load();

        //calling premierLeagueManager reference to manage console base
operations
        PremierLeagueManager premierLeagueManager =
PremierLeagueManager.getInstance();

        //String object to display the result of the terminal operations
        String message;


        ClubRankingComparator clubRankingComparator = new
ClubRankingComparator();

        String userInput;
        Scanner scanner = new Scanner(System.in);

        do{

            //Application menu
            System.out.println("\n-----------------------------------------
---------- \n\t\t   Premier League 2021" +
                    "\n-----------------------------------------------------
\n");
            System.out.println("1.\t Add Club to the League\n2.\t Delete Club
from the League\n3.\t Premier League Points Table\n"+
                    "4.\t Display League Clubs Information\n5.\t Add a
Match\n6.\t Launch GUI Interface\nQ\t to exit\n"+
                    "\n* League information are updating automatically"+ "\n-
------------------------------------------------- \n"
                    );
            System.out.print("Please select an option: ");


            userInput = scanner.nextLine();
            switch (userInput){
                case "1":
                    FileHandlingService.load();
                    message =
premierLeagueManager.addSportClub(ApplicationUtils.collectingClubInformation(
));

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
                    System.out.println(message);
```

```
                    break;

                case "2":
                    System.out.print("Please enter the Football club name:
");
                    String clubName = scanner.nextLine();
                    if(ApplicationUtils.isClubAvailable(clubName)){
                        System.out.print("Press 'Y' to delete the " +
clubName + " from League of press 'N' to Cancel: ");
                        String confirmation = scanner.nextLine();
                        if(confirmation.equals("Y") ||
confirmation.equals("y")){
                            message =
premierLeagueManager.removeSportClub(clubName);
                            System.out.println(message);
                        }else if (confirmation.equals("n") ||
confirmation.equals("N")){
                            System.out.println("Operation canceled");
                        }else {
                            System.out.println("Invalid Selection");
                        }
                    }else{
                        System.out.println("Couldn't delete the Club\n");
                    }

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
                    break;

                case "3":
                    FileHandlingService.load();

premierLeagueManager.getFootballClub().sort(clubRankingComparator);
                    System.out.println("----------------------------------
----------------------------------------");
                    for(SportClub sportClub:
premierLeagueManager.getFootballClub()){
                        Football football = (Football) sportClub;
                        System.out.format(" %s \t\tMatches
Won:\t"+football.getMatchesWon()
                                +"\tMatches
Lost:\t"+football.getMatchesLost()+"\tLeague
Points:\t"+football.getLeaguePoints(), sportClub.getClubName());
                        System.out.println();
                    }
                    System.out.println("----------------------------------
----------------------------------------");
                    break;

                case "4":
                    System.out.print("Please enter the Football club name:
");
                    String footballClubName = scanner.nextLine();

                    if(ApplicationUtils.isClubAvailable(footballClubName)){
                        try{
                            SportClub sportClub =
```

```
ApplicationUtils.findClubByName(footballClubName);
                            System.out.format("\nSport Club Name: %s "
+"\n"+"Club Location: %s ", sportClub.getClubName(),
sportClub.getClubLocation());
                            Football football  = (Football) sportClub;
                            System.out.format("\nMatches Won: %d \nMatches
lost: %d \nLeague Points: %d \n", football.getMatchesWon(),
football.getMatchesLost(), football.getLeaguePoints());
                        }catch (NullPointerException e){
                            System.out.println("Couldn't find the Club\n");
                        }
                    }else{
                        System.out.println("Couldn't find the Club\n");
                    }
                    break;

                case "5":
                    Match math =
ApplicationUtils.collectingMatchInformation();
                    message = premierLeagueManager.addMatch(math);
                    System.out.println(message);

FileHandlingService.saveLeagueInformation(premierLeagueManager.getFootballClu
b(), premierLeagueManager.getLeagueMatches());
                    break;

                case "6":
                    Runtime runtime = Runtime.getRuntime();
                    String url = "http://localhost:4200/";
                    runtime.exec("rundll32 url.dll,FileProtocolHandler
"+url);
                    break;

                default:
                    if(!userInput.equals("Q")){
                        System.out.println("Invalid selection\n");
                    }
            }
        }while (!userInput.equals("Q"));
    }
}
```

routes

```
GET     /                           controllers.FrontendController.index()
GET     /clubs/list                 controllers.APIController.listClubs()
GET     /matches/list               controllers.APIController.listMatches()
GET     /matches/addRandom
controllers.APIController.sendRandomMatch()

POST    /clubs/add
controllers.APIController.addClub(request: Request)
POST    /matches/add
```

```
controllers.APIController.addMatch(request: Request)

application.conf)
#GET     /api/summary                controllers.HomeController.appSummary

GET     /*file
controllers.FrontendController.assetOrDefault(file)

#POST    /api/postTest               controllers.HomeController.postTest()
```

Angular Source Code

interface.ts

```
export interface IFootball {

    clubName: string;
    clubLocation: string;
    matchesWon: number;
    matchesLost: number;
    leaguePoints: number;
    totalLeagueGoals: number;
    goalDifference: number;
}

export interface IMatch {

  teamOne: IFootball;
  teamTwo: IFootball;
  teamOneGoals: number;
  teamTwoGoals: number;
  matchDate: IMatchDate;

}

export interface IMatchDate {

  date : number;
  month: number;
  year: number;

}
```

core.module.ts

```
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';

import { DataService } from './data.service';
import {SortingService} from "./sorting.service";

@NgModule({
```

```
    imports: [ HttpClientModule ],
  providers:  [ DataService, SortingService ]
})
export class CoreModule { }
```

data.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import {IFootball, IMatch} from './../shared/interfaces';
import { map, catchError } from 'rxjs/operators';

@Injectable()
export class DataService {

    constructor(private http: HttpClient){}

    getFootballClubs(): Observable<IFootball[]> {
        return this.http.get<IFootball[]>('/clubs/list')
            .pipe(
                catchError(DataService.handleError)
            );
    }

    getMatches(): Observable<IMatch[]> {
      return this.http.get<IMatch[]>('/matches/list')
        .pipe(
          catchError(DataService.handleError)
        );
    }

    getRandomMatch(): Observable<IMatch> {
      return this.http.get<IMatch>('/matches/addRandom')
        .pipe(
          catchError(DataService.handleError)
        );
    }

    match(): Promise<any> {
      return this.http.get('/matches/list').toPromise();
    }

    private static handleError(error: any){
        console.error("Server error", error);
        if(error.error instanceof Error){
            const errorMessage = error.error.message;
            return Observable.throw(errorMessage);
        }
        return Observable.throw(error || "Server Error");
    }


}
```

sorting.service.ts

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

@Injectable()
export class SortingService {

  property: string = null;
  direction: number = 1;

  sort(collection: any[], prop: any) {
    this.property = prop;
    this.direction = (this.property === prop) ? this.direction * -1 : 1;

    collection.sort((b: any,a: any) => {
      let valueOne: any;
      let valueTwo: any;

      if (prop && prop.indexOf('.') > -1) {
        valueOne = this.resolveProperty(prop, a);
        valueTwo = this.resolveProperty(prop, b);
      }
      else {
        valueOne = a[prop];
        valueTwo = b[prop];
      }

      if (this.isString(valueOne)) valueOne = valueOne.trim().toUpperCase();
      if (this.isString(valueTwo)) valueTwo = valueTwo.trim().toUpperCase();

      if(valueOne === valueTwo){
        return 0;
      }
      else if (valueOne > valueTwo){
        return this.direction * -1;
      }
      else {
        return this.direction;
      }
    });
  }

  isString(value: any) : boolean {
    return (value && (typeof value === 'string' || value instanceof String));
  }

  resolveProperty(path: string, object: any) {
    return path.split('.').reduce(function(previous, current) {
      return (previous ? previous[current] : undefined)
    }, object || self)
  }
}
```

football.component.ts

```
import { Component, OnInit } from '@angular/core';
import { IFootball } from '../shared/interfaces';
import { DataService } from '../core/data.service';
import {SortingService} from "../core/sorting.service";

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

export class FootballComponent implements OnInit {

  title: string;
  iFootballs: any[];

  constructor(private dataService: DataService, private sorterService:
SortingService) { }

  ngOnInit(): void {
    this.title = "Football Clubs"
    this.dataService.getFootballClubs()
      .subscribe((footballClubs: IFootball[]) => this.iFootballs =
footballClubs);
  }

  sort(prop: string){
    this.sorterService.sort(this.iFootballs, prop);
  }
}
```

football.module.ts

```
import { NgModule }      from '@angular/core';
import { FormsModule }      from '@angular/forms';
import { CommonModule } from '@angular/common';
import { FootballComponent } from './football.component';
import { FootballRoutingModule } from "./football-routing.module";

@NgModule({
    imports:        [ CommonModule, FormsModule, FootballRoutingModule ],
    declarations:   [ FootballComponent ],
    exports: [FootballComponent]
})

export class FootballModule {}
```

football.component.html

```
<style>

  table caption {
    font-size: 1.5em;
```

```
    margin: .5em 0 .75em;
  }
  table {
    background: #012B39;
    border-radius: 0.60em;
    border-collapse: collapse;
    margin: 1em;
  }
  th {
    border-bottom: 1px solid #364043;
    color: #E2B842;
    font-size: 0.85em;
    font-weight: 600;
    padding: 0.5em 1em;
    text-align: left;
  }
  td {
    color: #fff;
    font-weight: 400;
    padding: 0.65em 1em;
  }
  .disabled td {
    color: #4F5F64;
  }
  tbody tr {
    transition: background 0.25s ease;
  }
  tbody tr:hover {
    background: #014055;
  }
  br {
    display: block;
    margin: 100px 0;
    content: " ";
  }
  .button {
    background-color: #555555; /* Green */
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    transition-duration: 0.4s;
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
    border-radius: 10px;
  }

</style>

<body>
<h2 style="position: fixed; top: 300px; left: 60px;">⚽  League Summery (Club
Status)</h2>
```

```
<br />
<button type="button" class="button" (click)="sort('totalLeagueGoals')">Sort
by Goals</button>
<button type="button" class="button" (click)="sort('matchesWon')">Sort by
Wins</button>
<div style="overflow-y:auto; height: 400px;">
<table>
  <thead>
  <tr>
    <th style="cursor: pointer" > Club Name </th>
    <th style="cursor: pointer" >Matches Won</th>
    <th style="cursor: pointer" >Matches Lost</th>
    <th style="cursor: pointer" >Goals in League</th>
    <th style="cursor: pointer" >Goals Difference</th>
    <th style="cursor: pointer" >League Points</th>
  </tr>
  </thead>
  <tbody>
  <tr *ngFor="let football of iFootballs">
    <td data-label="Club Name">{{ football.clubName }}</td>
    <td data-label="Matches Won">{{ football.matchesWon }}</td>
    <td data-label="Matches Lost">{{ football.matchesLost }}</td>
    <td data-label="League Points">{{ football.totalLeagueGoals }}</td>
    <td data-label="League Points">{{ football.goalDifference }}</td>
    <td data-label="League Points">{{ football.leaguePoints }}</td>
  </tr>
  </tbody>
</table>
</div>
<br />
</body>
```

matches.component.ts

```
import { Component, OnInit } from '@angular/core';
import { IMatch, IMatchDate} from "../shared/interfaces";
import { DataService } from "../core/data.service";
import {SortingService} from "../core/sorting.service";

@Component({
  selector: 'app-matches',
  templateUrl: './matches.component.html',
  styleUrls: ['./matches.component.css']
})
export class MatchesComponent implements OnInit {

  constructor(private dataService: DataService, private sortingService:
SortingService) { }

  title: string;
  iMatches: IMatch[];
  searchedMatches: IMatch[] = [];
  matchDay: IMatchDate = { date:0, month:0, year:0}

  ngOnInit(): void {
```

```
    this.title = "Match Information"

    this.dataService.getMatches()
      .subscribe((matches: IMatch[]) => this.iMatches = matches);
  }


  search(data: string): void {
    this.matchDay.year = Number(data.slice(0,4));
    this.matchDay.month = Number(data.slice(5,7));
    this.matchDay.date = Number(data.slice(8,10));

    for(let match of this.iMatches){
      if((match.matchDate.date === this.matchDay.date) &&
(match.matchDate.month === this.matchDay.month) && (match.matchDate.year ===
this.matchDay.year)){
            this.searchedMatches.push(match);
      }
    }
    this.iMatches = this.searchedMatches;
  }

  sort(prop: any){
    this.sortingService.sort(this.iMatches, prop);
  }

  previousMatches(): void {

  }

}
```

matches.module.ts

```
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import {MatchesRoutingModule} from "./matches-routing.module";
import {MatchesComponent} from "./matches.component";

@NgModule({
  imports: [CommonModule, FormsModule, MatchesRoutingModule],
  declarations: [MatchesComponent ],
  exports:[ MatchesComponent ]
})

export class MatchesModule {}
```

matches.component.html

```
<style>
  table caption {
    font-size: 1.5em;
    margin: .5em 0 .75em;
  }
```

```
  table {
    overflow-y: scroll;
    background: #012B39;
    border-radius: 0.60em;
    border-collapse: collapse;
    margin: 1em;
  }
  th {
    border-bottom: 1px solid #364043;
    color: #E2B842;
    font-size: 0.85em;
    font-weight: 600;
    padding: 0.5em 1em;
    text-align: left;
  }
  td {
    color: #fff;
    font-weight: 400;
    padding: 0.65em 1em;
  }
  .disabled td {
    color: #4F5F64;
  }
  tbody tr {
    transition: background 0.25s ease;
  }
  tbody tr:hover {
    background: #014055;
  }
  br {
    display: block;
    margin: 80px 0;
    content: " ";
  }

  .button {
    background-color: #555555;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    transition-duration: 0.4s;
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
    border-radius: 10px;
  }

</style>

<body>
<br>
<br>
```

```
<h2 style="position: fixed; top: 300px; left: 60px;">⚽ Match Summery</h2>
<input  style="width:160px; height: 30px; border-radius: 10px" type="date"
#matchQ placeholder="Search Match Information" />
<button class="button" (click)="search(matchQ.value)">Search</button>
<button class="button" (click)="ngOnInit()">Clear</button>
<br>
<div style="overflow-y:auto; height: 400px;">
<table>
  <thead>
  <tr>
    <th (click)="sort(matchDay.month)">Match Date</th>
    <th >Team One</th>
    <th >Team Two</th>
    <th >Team One Goals</th>
    <th >Team Two Goals</th>
  </tr>
  </thead>
  <tbody>
  <tr *ngFor="let match of iMatches">
    <td data-label="Team One">{{ match.matchDate.date}}/{{
match.matchDate.month}}/{{ match.matchDate.year}}</td>
    <td data-label="Team One">{{ match.teamOne.clubName}}</td>
    <td data-label="Team Two">{{ match.teamTwo.clubName }}</td>
    <td data-label="Team One Goals">{{ match.teamOneGoals }}</td>
    <td data-label="Team Two Goals">{{ match.teamTwoGoals}}</td>
  </tr>
  </tbody>
</table>
</div>
<br />
</body>
```

addMatch.component.ts

```
import { Component, OnInit } from '@angular/core';
import {IFootball, IMatch} from "../shared/interfaces";
import {DataService} from "../core/data.service";

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

export class AddMatchComponent implements OnInit {

  randomMatch: IMatch;
  isMatchGenerated: boolean = false;

  constructor(private dataService: DataService) {}

  ngOnInit(): void {}

  random(){
    this.dataService.getRandomMatch()
```

```
      .subscribe((match: IMatch) => this.randomMatch = match);
      this.isMatchGenerated = true;
      console.log(this.randomMatch);
  }
}
```

addMatch.module.ts

```
import { NgModule }      from '@angular/core';
import { FormsModule }      from '@angular/forms';
import { CommonModule } from '@angular/common';
import {AddMatchComponent} from "./addMatch.component";
import {AddMatchRoutingModule} from "./addMatch-routing.module";

@NgModule({
  imports:        [ CommonModule, FormsModule, AddMatchRoutingModule ],
  declarations:   [ AddMatchComponent ],
  exports:        [ AddMatchComponent ]
})

export class AddMatchModule {}
```

addMatch.component.html

```
<style>
  .button {
    background-color: #555555; /* Green */
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    transition-duration: 0.4s;
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
    border-radius: 10px;
  }

</style>

<body>
<h2 style="position: fixed; top: 300px; left: 60px;">⚽  Add a Match to
league</h2>
<br />
<br />
<br />
<h2><div *ngIf="isMatchGenerated"> {{randomMatch?.teamOne?.clubName}} Vs
{{randomMatch?.teamTwo?.clubName}}</div> </h2>
<h3><div *ngIf="isMatchGenerated"> {{randomMatch?.teamOne?.clubName}} :
```

```
{{randomMatch?.teamOneGoals}}</div> </h3>
<h3><div *ngIf="isMatchGenerated"> {{randomMatch?.teamTwo?.clubName}} :
{{randomMatch?.teamTwoGoals}}</div> </h3>
<h3><div *ngIf="isMatchGenerated"> Match Date :
{{randomMatch?.matchDate?.date}}/{{randomMatch?.matchDate?.month}}/{{randomMa
tch?.matchDate?.year}}</div> </h3>

<button class="button" type="button" (click)="random()">Generate a
Match</button>

</body>
```

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title: 'ui';

  constructor() {}
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HTTP_INTERCEPTORS, HttpClientModule, HttpClientXsrfModule } from
'@angular/common/http';
import { AppRoutingModule} from "./app-routing.module";

import { AppComponent } from './app.component';
import { FootballModule } from './football/football.module';
import { CoreModule } from './core/core.module';

import {MatchesModule} from "./matches/matches.module";
import {AddMatchModule} from "./AddMatch/addMatch.module";

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    HttpClientXsrfModule.withOptions({
      cookieName: 'Csrf-Token',
      headerName: 'Csrf-Token',
    }),
    FootballModule,
    MatchesModule,
```

```
    CoreModule,
    AddMatchModule,
    AppRoutingModule
  ],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

app.component.html

```
<style>
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
    border-radius: 17px;
  }

  li {
    float: left;
  }

  li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
  }

  li a:hover {
    background-color: #111;
  }
</style>
<br />
<br />
<img style="height: 30%; width: 30%; display: block; margin-left: auto;
margin-right: auto;" src="assets/logo.png">
<br />
<ul>
  <li><a href="#">Home</a></li>
  <li><a routerLink="/clubs/list">Club Status</a></li>
  <li><a routerLink="/matches/list">Match Status</a></li>
  <li><a routerLink="/matches/add">Add a Match</a></li>
  <li><a href="#">About League</a></li>
  <li style="float:right"><a href="#">Premier League Manager</a></li>
</ul>

<div style="position: fixed; top: 300px; left: 60px;">
<router-outlet></router-outlet>
</div>
<img style="height: 47%; width: 60%; position: fixed; bottom: -10px; right: -
110px;" src="assets/cover2.png">
```