GEBZE TECHNICAL UNIVERSITY COMPUTER ENGINEERING CSE 222 / 505 – SPRING 2020 GROUP 11 – PROJECT REPORT

1. Group members

- Berke BELGİN 171044065
- Medine ASLAN 161044015
- Oğuz Kaan KILIÇ 111044032
- Mustafa SEVİM 171044011
- Sema DİLBER 161044064

2. Problem definition

A software system, that organizes and keeps the informations for different countries about soccer, will be designed. This system has users from different levels such as Federation President, Club President, Coach, Player, Employees etc. The Federation will be the highest foundation in the country in this system. This foundation organizes the league. There must be teams to talk about the league and there is a Referee Committee works for the federation. There can be different amount of teams in different leagues and federation assigns the referees to matches. In every team, there are so many types of employees such as Players, Coach, Club President, Health Personnel etc.

In this system, Federation and Clubs Presidents have permission for adding, removing and adjusting the other things. The Club Presidents can add and remove player and Club Committee can change the president and coach. The Federation President can assign the referees to matches and federation committee can change the president and adjust the date of the matches. Within the federation and clubs have committees that can change their presidents.

3. Users of the system

Users	Main Work
Federation President	Assign new federation president, add and remove federation admin.
Federation Admin	Add and remove referees, clubs and matches.
Club President	Assign new club president and coach, add and remove club admin.
Club Admin	Add and remove club players, change player's health situation.
Club Players	See informations about clubs and other players.
Coach	Determine the player squad before the match.
Fans	See only information about clubs and players.

4. Requirements (detailed)

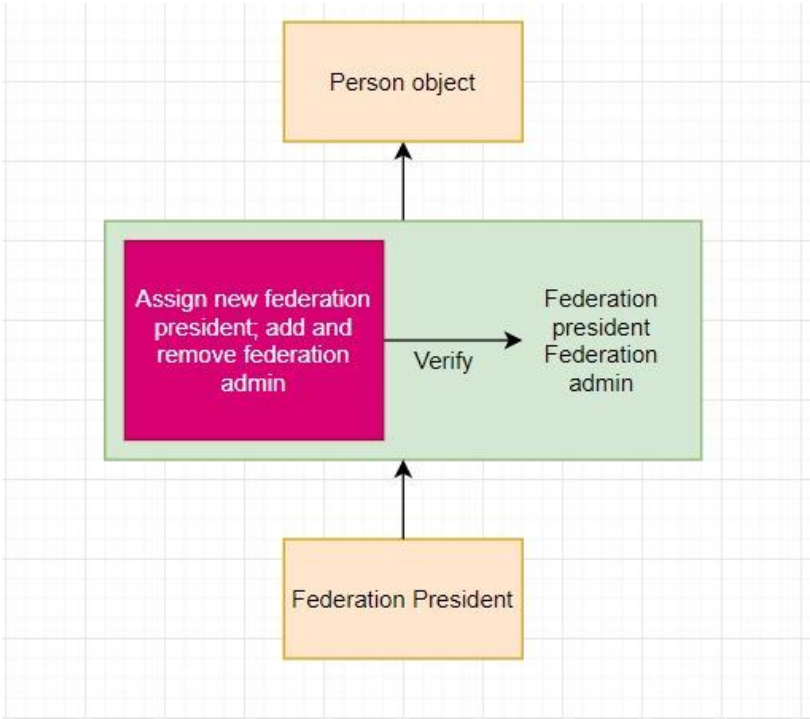
In this software, we firstly have a login page in order to authenticate the user. If the user does not sign in the system, it can see the only public information and does not have any privilege. There are privileged users in the system such as federation president, club president, coach, club admin, federation admin and system admin. All of the users have different works to do so there is no conflict in the system about works.

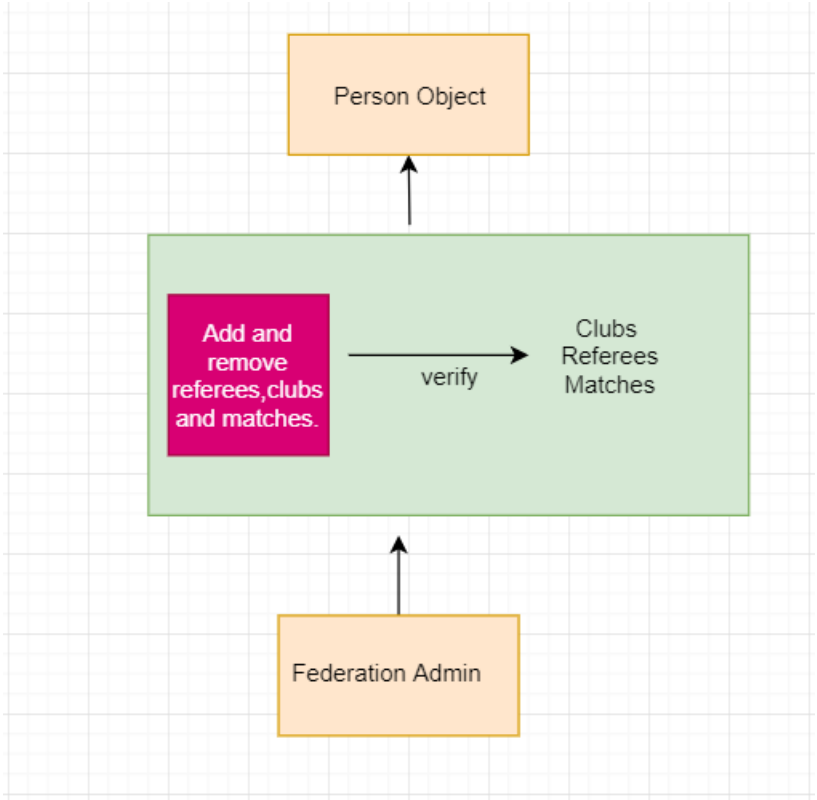
After the verification process done, firstly we have a main page to see standings of football clubs for a football league which user chooses and changes easily by navigating between countries (Football Federations) and football leagues. Then we have a page for tracking all the matches played and will be played in the latest football season. For the ones that are already played, user can see the details about the matches. If user wants to see information about previous seasons, it will be showed as an option to user. There will be another page to see all the information with details of a football club such as football players, stadium, club president etc.

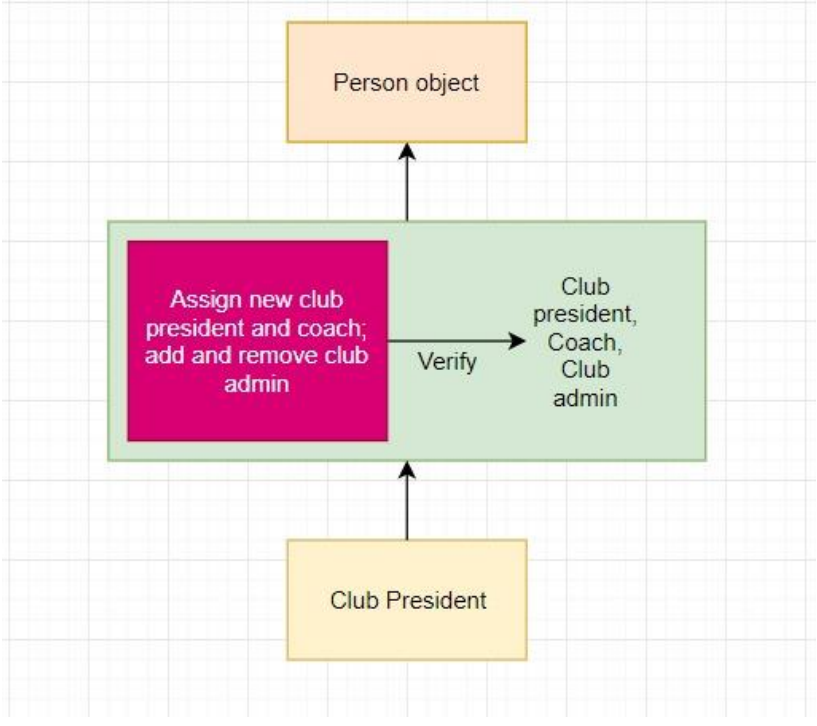
The program will have another page to see detailed information about players such as name, surname, age, height, weight etc. when information of a player is chosen. Finally there are pages for users who are privileged such as admins, presidents.

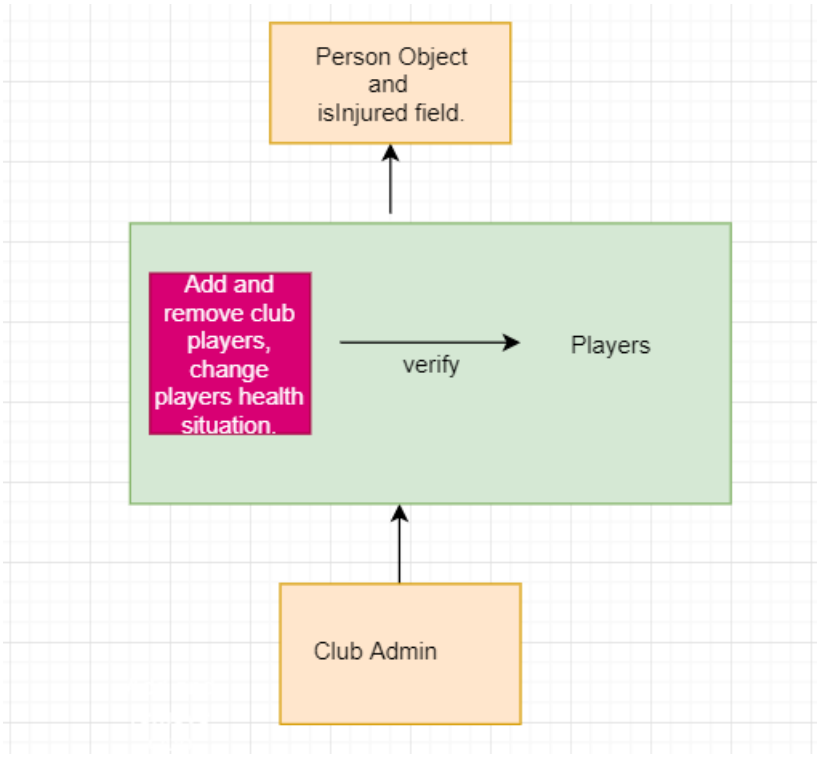
We kept the information about the system in the file such as federations, clubs, seasons, players and users. For this reason, we have made the process of reading the file and saving this information to the file private. Thus, we ensured that only those who had access to this information were accessed. We restricted access to other information by dividing the program into pages. In this way, we showed users only transactions that have access.

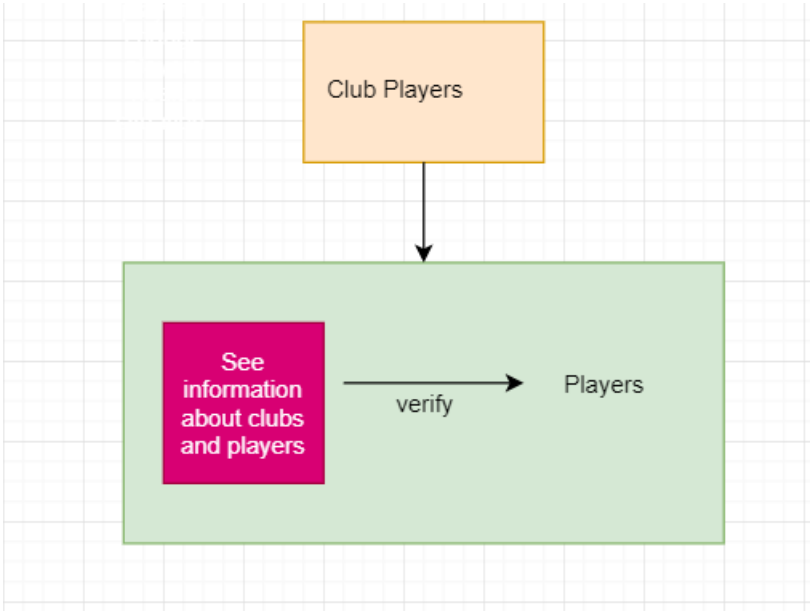
5. Detailed system modules

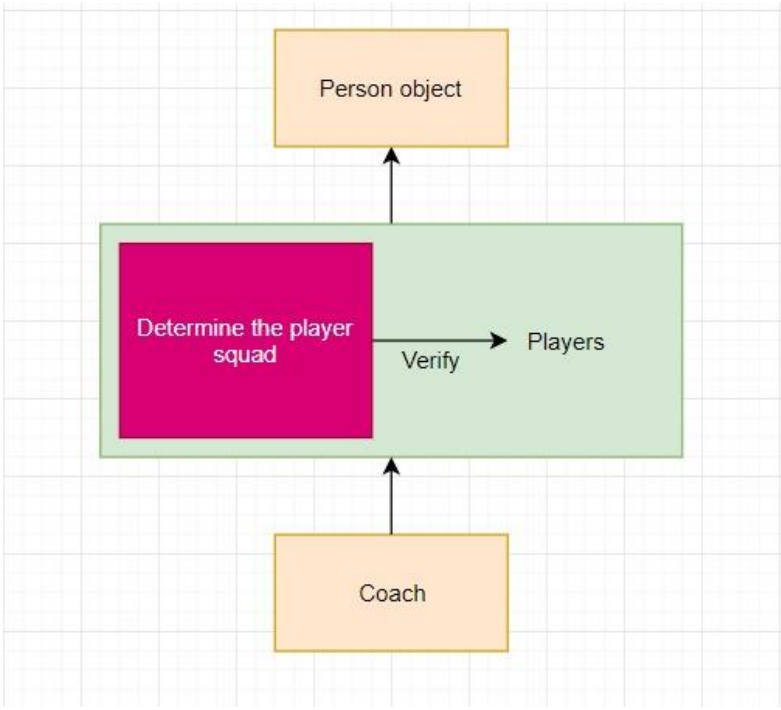


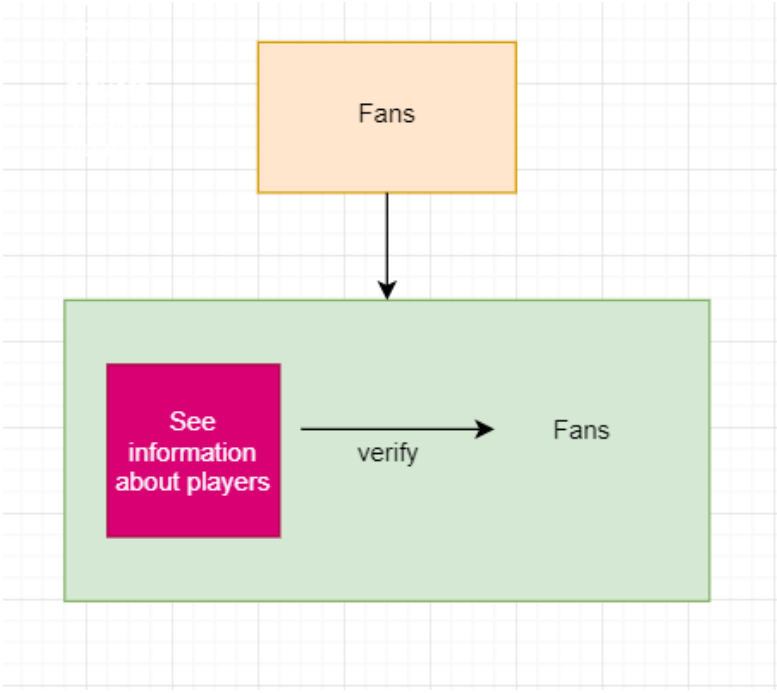

Person Object
Add and remove referees,clubs and matches.
verify
Clubs
Referees
Matches
Federation Admin


Person object
Assign new club president and coach; add and remove club admin
Verify
Club president, Coach, Club admin
Club President

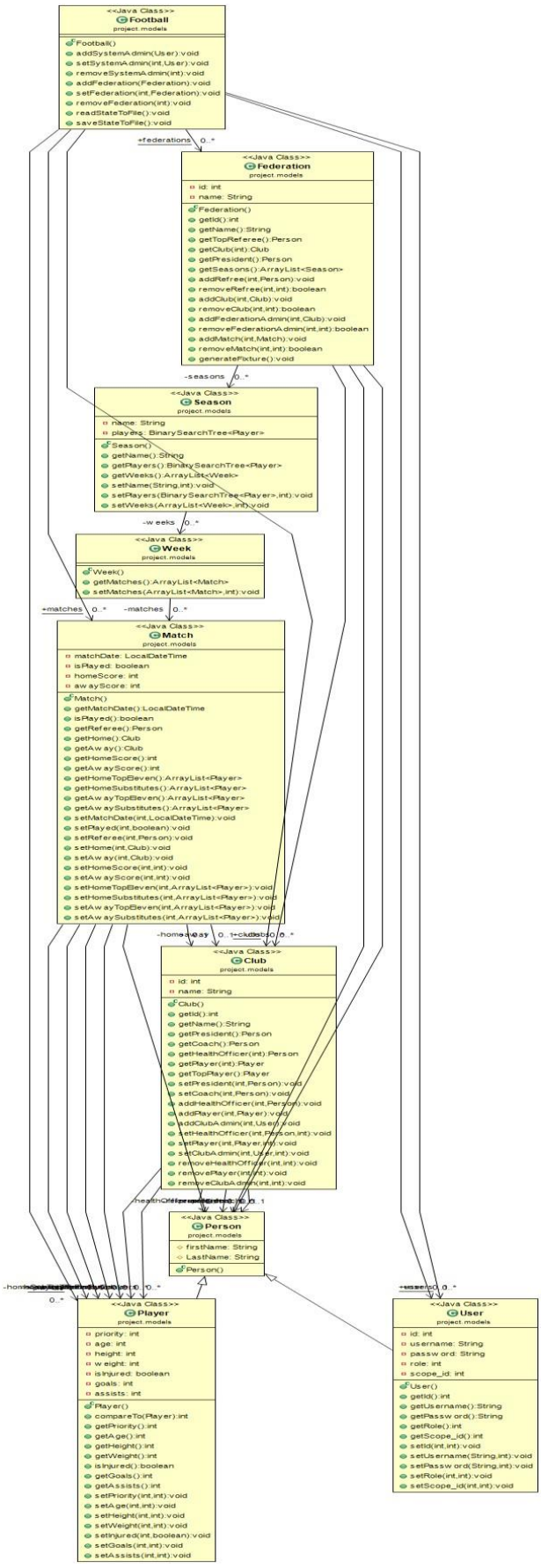

Person Object
and
isInjured field.
Add and
remove club
players,
change
players health
situation.
verify
Players
Club Admin


Club Players
See
information
about clubs
and players
verify
Players

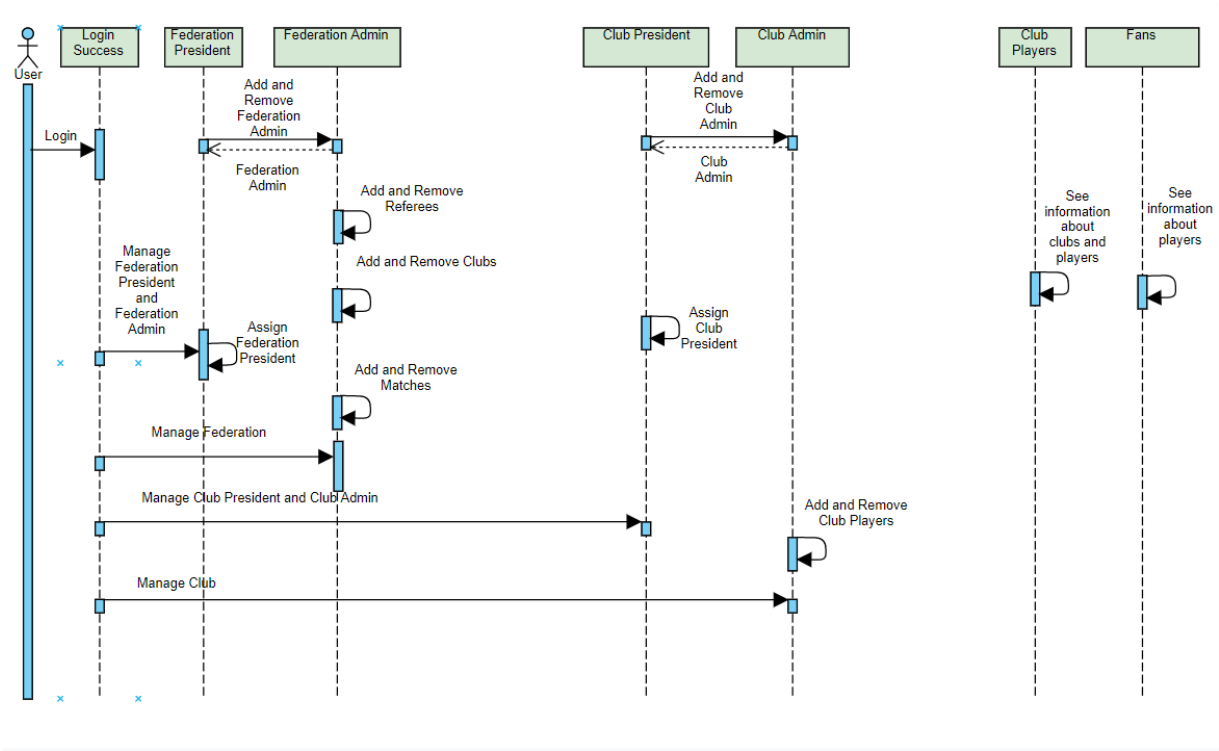

Person object
Determine the player squad
Verify
Players
Coach

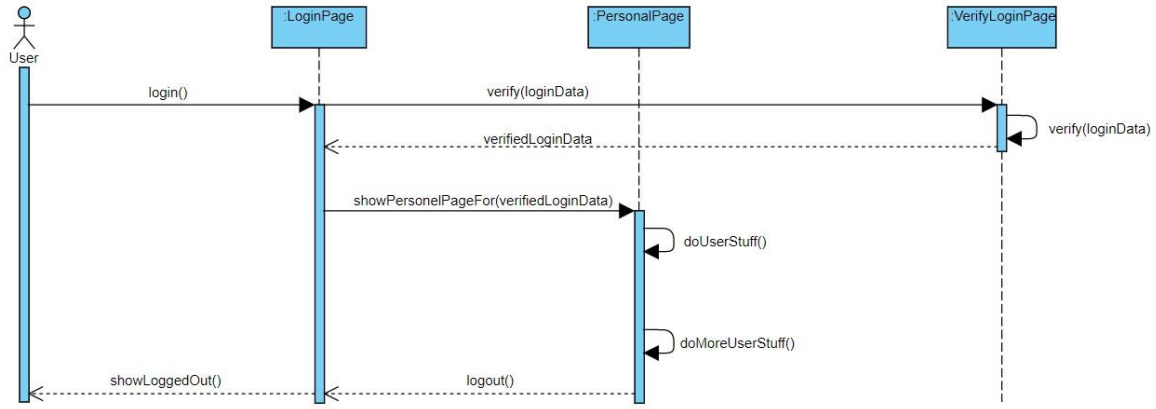

Fans
See information about players
verify
Fans

6. Class diagram



7. Sequential diagrams





8. Other diagrams

-

9. Implementation details

In this Project, we got inspired by MVC architecture. We did not exactly apply this architecture but we partly did it. Firstly, we divided program to pages. Some users have some different options in same pages. For example, club president sees the chance club president,chance coach, add club admin etc. But regular guest cant see these options, they can just see some of the basic options such as players, fixture etc. To do that, we have some indicators. We checked those indicators, we understood what the user is and then we showed to user all of the options that they can do. For every option, we created the aimed page's object and call them to show the user chosen page. And we use these data structures:

ArrayList : Seasons, clubs, weeks, matches and when holding the players in matches class, ArrrayList is used as the data structure. These informations must be accessed randomly so that is why ArrayList is used.

PriorityQueue : Every player has its key priority to be in the play. For indicated keys, top eleven, substitutes and players who are out of the team are determined. Players who have the highest priority plays the match for the team. That is why priority queue is used here.

Balanced Binary Search Tree : In this system, user can see the goal amount higher than the indicated amount. In this balanced binary search tree, a player, who has got the maximum amount of goals, is chosen from every team and is pushed to the balanced binary search tree. The point is every team has only one player in this tree. Because of search algorithm takes O(n) time in balanced binary search tree, it is appropriate to use.

Queue : Every federation has referees. This referees can be hold in a queue. In this structure, every referee got a game to rule fairly. When a referee ,who is at the front of the queue, is assigned to a match, it is removed from the queue and goes to the match, that provides federation to a referee can be assigned a match at every week if there are enough referee to matches. After the match, referee goes back and is pushed to the end of the queue. In this way referees have been given to chance to rule the games in order.

Set-Map : We use set for keeping the home's and away's top elevens and substitues. In this structure datas stored unordered. For this reason we keep this data into the set. Because we don't want their orders. Also, set structure can not achive its contents by index we didn't need it either. So, we thought that the use of this structure makes sense.

Skip List : federationAdmins and ClubAdmins SkipLists is used as the data structure. We used this data structure to make it easier and faster to access and thus perform operations on admins and we don't need to reach out to the admins one by one.

10. Test cases

Test Case Id	Test Scenario	Test Steps	Test Data	Expected Results	Pass/Fail
T01	Check login with invalid username password	1.Open program 2.Enter username 3.Enter password 4.press login	username=admin password=admin	As Expected	Pass
T02	Check login with valid username password	1.Open program 2.Enter username 3.Enter password 4.press login	username=admin password=admin	As Expected	Pass
T03	Check readStateFromFile method.	Call methods with this order: *readFederationsStateFromFile*() *readClubsStateFromFile*() *readSeasonsStateFromFile*() *readPlayersStateFromFile*() *readUsersStateFromFile*()	No parameter	As expected	Pass
T04	Check saveStateToFile method.	Call methods with this order: *saveFederationsStateToFile*(); *saveClubsStateToFile*(); *saveSeasonsStateToFile*(); *savePlayersStateToFile*(); *saveUsersStateToFile*();	No parameter	As expected	Pass
T05	Check addReferee method.	Call method with valid parameter.	Person Object	As Expected	Pass
T06	Check addReferee method.	Call method with unauthorized user.	Person Object	As Expected	Pass
T07	Check removeReferee method.	Call method with valid parameter.	Person Object	As Expected	Pass
T08	Check removeReferee method.	Call method with unauthorized user.	Person Object	As Expected	Pass
T09	Check addClub method.	Call method with valid parameter.	Club Object	As Expected	Pass
T10	Check addClub method.	Call method with unauthorized user.	Club Object	As Expected	Pass
T11	Check removeClub method.	Call method with valid parameter.	Club Object	As Expected	Pass
T12	Check removeClub method.	Call method with unauthorized user.	Club Object	As Expected	Pass

T13	Check addFederationAdmin method.	Call method with valid parameter.	Person Object	As Expected	Pass
T14	Check addFederationAdmin method.	Call method with unauthorized user.	Person Object	As Expected	Pass
T15	Check removeFederationAdmin method.	Call method with valid parameter.	Person Object	As Expected	Pass
T16	Check removeFederationAdmin method.	Call method with unauthorized user.	Person Object	As Expected	Pass
T17	Check addPlayer method.	Call method with valid parameter.	Player Object	As Expected	Pass
T18	Check addPlayer method.	Call method with unauthorized user.	Player Object	As Expected	Pass
T19	Check addClubAdmin method.	Call method with valid parameter.	User Object	As Expected	Pass
T20	Check addClubAdmin method.	Call method with unauthorized user	User Object	As Expected	Pass
T21	Check removePlayer method.	Call method with valid parameter.	Player Object	As Expected	Pass
T22	Check removePlayer method.	Call method with unauthorized user.	Player Object	As Expected	Pass
T23	Check removeClubAdmin method.	Call method with valid parameter.	Player Object	As Expected	Pass
T24	Check removeClubAdmin method.	Call method with unauthorized user	Player Object	As Expected	Pass
T25	Check addMatch method.	Call method with valid parameter.	Match Object	As Expected	Pass
T26	Check addMatch method.	Call method with unauthorized user.	Match Object	As Expected	Pass
T27	Check removeMatch method.	Call method with valid parameter.	Match Object	As Expected	Pass
T28	Check removeMatch method.	Call method with unauthorized user.	Match Object	As Expected	Pass