CS102	**Fall 2022/23**		
Instructor:	**Uğur Güdükbay**	Project Group	1I
Assistant:	**Osama Zafar**		

Criteria	TA/Grader	Instructor
Overall		

~ Bilkent Halısaha ~

The GOATS

Kaan AYDENİZ

Erdem AYDIN

Oğuzhan GENÇ

Furkan Emre KOLUKIRIK

Hasan Kutluhan ŞIPKA

Detailed Design Report

3 December 2022

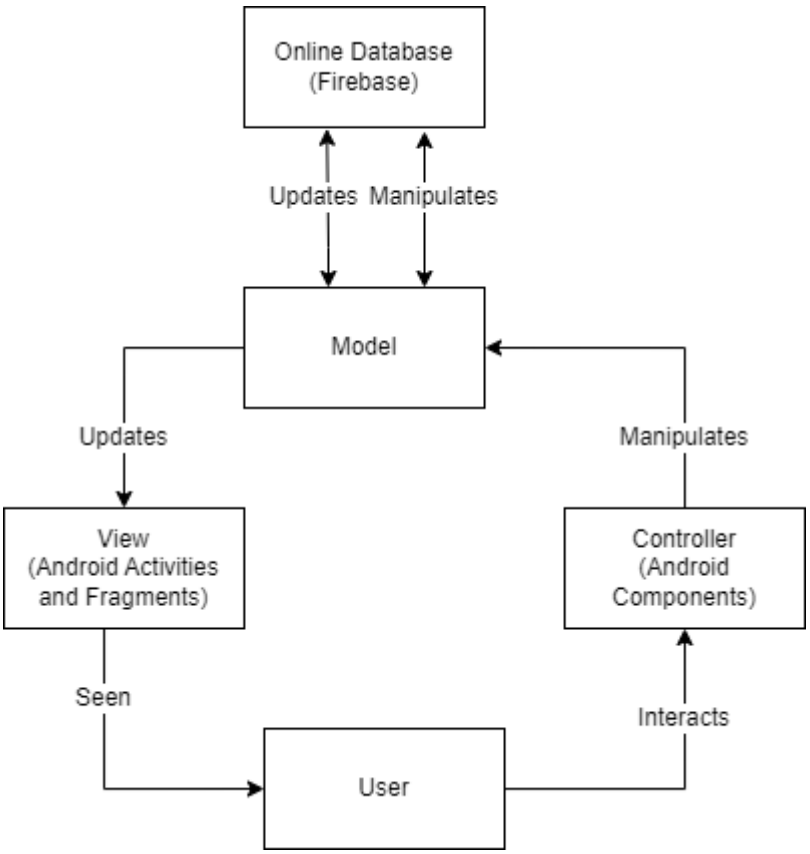
1. Introduction

Bilkent Halısaha is a mobile application that facilitates Bilkent University students to organize football matches and find players for these matches easier. Creating football matches with specific locations and dates; finding existing matches is possible with Bilkent Halısaha.

2. Details

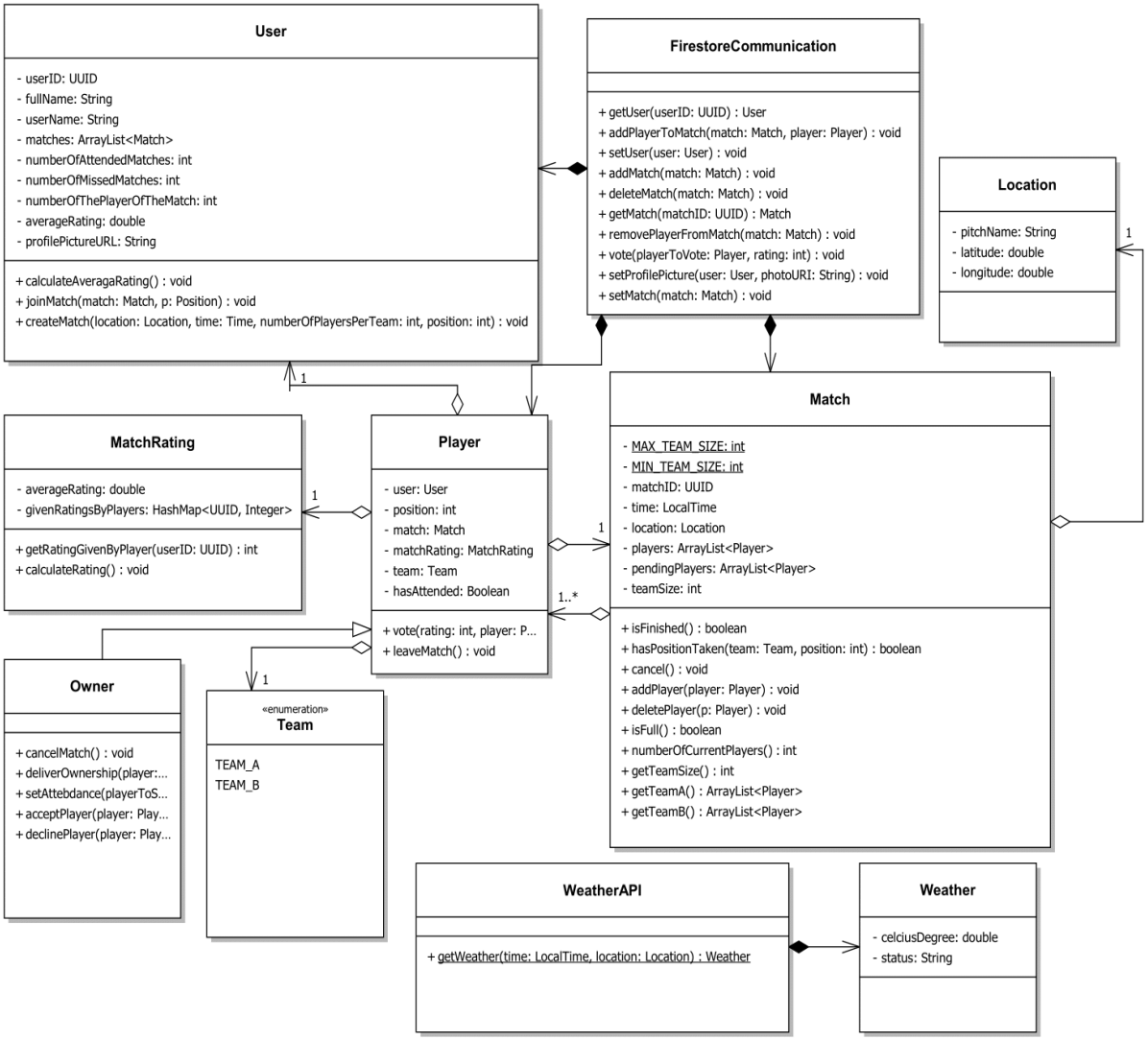
2.1 System Overview

Our application will be available on the Android operating system. We will develop the application by using the Android Studio development environment. Because our application needs interaction with other users, we will have an online database and we have selected some Google Firebase tools which are Cloud Firestore, Cloud Storage, and Firebase Authentication.



We use a model-view-controller design architecture in our project. Through this architecture, users can interact with the application and by the controllers such as buttons or combo boxes, the models can be changed. There is two-way interaction between model and online database. After a change in the model, the model either manipulates or updates the database. Then, the database either also updates or manipulates the model. After this procedure, the model updates the view, so the user can see the changes in the view.

2.2 Core Design Details



BilkentHalisahaUMLDiagram.pdf

1. User Class

User class contains information about users such as fullname, username, userId, profile picture and some statistics about the user. Briefly, the user class represents the attributes of users. Also, the user class has some methods that give them the ability to join or create matches.

2. Match Class

When the user creates a match, the match object will be instantiated. Match class contains information about the location of the match, team size, players in the match and methods to manipulate these fields. Basically, match class is a representation of a football match. Furthermore, the match class has some methods to give functionality to a football match.

3. Player Class

Player class is specific to each match. It contains required data for the match-specific information which are position, team and has attended information and MatchRating object. It contains methods for giving votes to other players in that match and leaving the match.

4. Owner Class

Owner class is inherited from the player class. It is basically the admin of the match. Therefore, it has some additional functionality in addition to the player that enables the owner to organize the match.

5. FirestoreCommunication Class:

FirestoreCommunication class establishes connection and interaction between the online database which is Cloud Firestore. The class has methods to fetch current data about past or incoming Match objects and up-to-date data of users. It also provides methods to add, update or remove Match and User objects. This class interacts with the object using deterministic userId and matchId attributes which are UUID.

6. MatchRating Class

MatchRating class contains information about the rating of the player object. This class enables our system to find players and their given ratings to players. This class also has functionality of calculating average rating of players for specific matches. Since the average rating will be displayed on the profile page, our system will use this class' functionality.

7. WeatherAPI and Weather Class

WeatherAPI returns the weather information according to location and time. It is returned as a weather object.

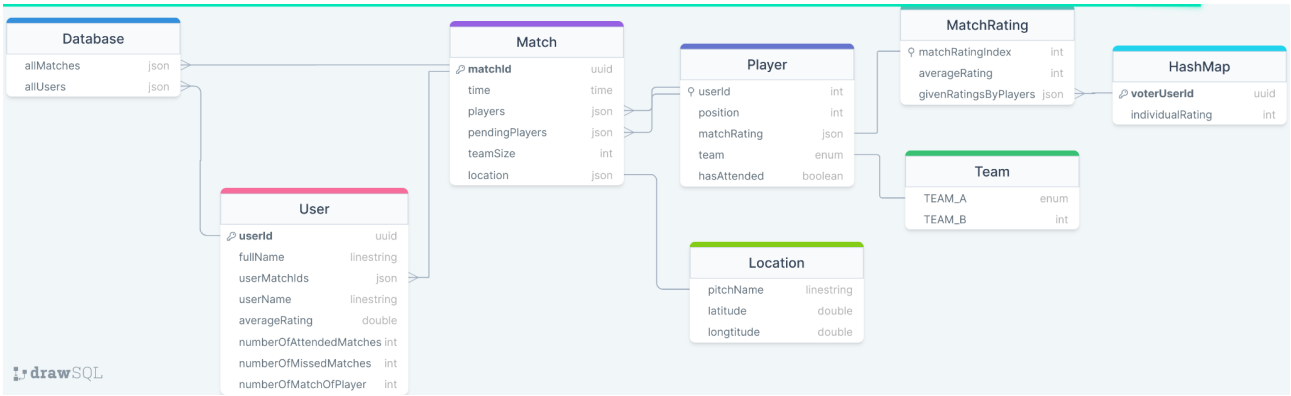
8. Location

Location class includes the name of the pitch and also it includes the latitude and the longitude of the pitch.

9. Team

Team is an enumeration that contains team A or team B enums.

Database Structure



databaseDiagram.png

Also, here is our database structure. Database will store User and Match objects and other objects will be nested objects inside of User and Match objects. Because User and Match objects might be updated independently of each other, when User and Match objects need references to

each other, their UUID will be stored in the database. In contrast, our client which is the Android application will store these objects as nested objects.

2.3 Task assignment

1. **User Interface**

   Oğuzhan and Erdem will implement the UI components, the basic event listeners the connection between the UI and the model components.

2. **Database**

   Kaan will create the connection between the app and firebase. He will also handle the FirestoreCommunication class that contains the methods about firebase.

3. **Model Classes**

   Kutluhan and Furkan Emre will implement model classes such as Match, User, Player, etc. which constitute the main functionality of the application.

   (Note that each member can help each section if some sections become overburdened)

3. Summary & Conclusions

To sum up, our project will be a mobile application that allows football players to create and join matches. We will use model-view-controller design architecture in our project. Furthermore, we have organized our classes and the relationship between them. The organization can be seen in the UML diagrams. Finally, we divided the work into smaller parts and all our members took part in the assignments.