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

24 December 2022

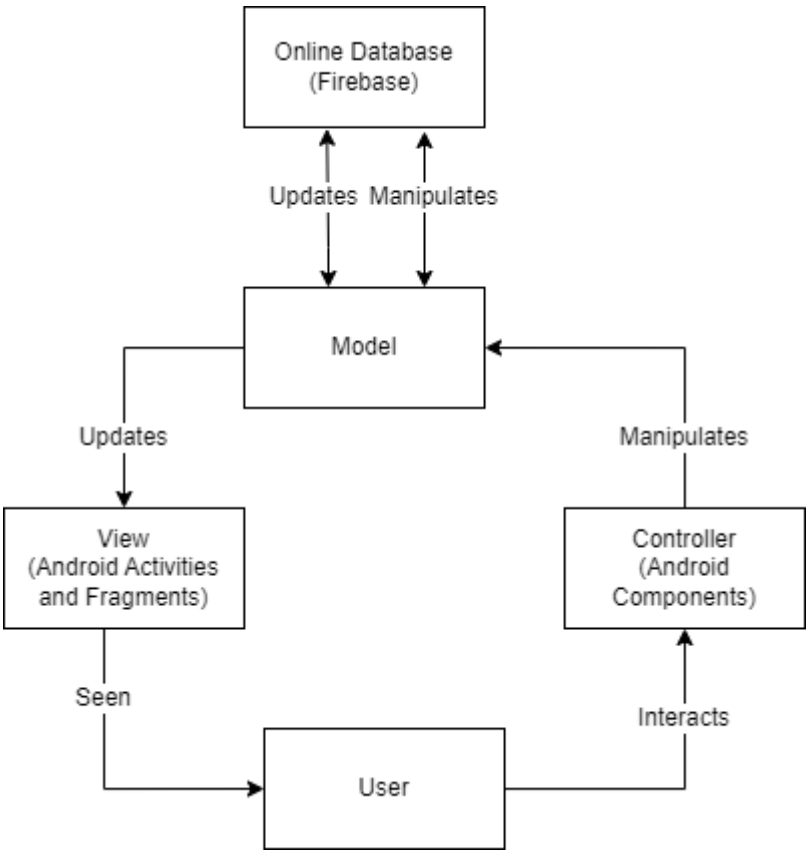
1. Introduction

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

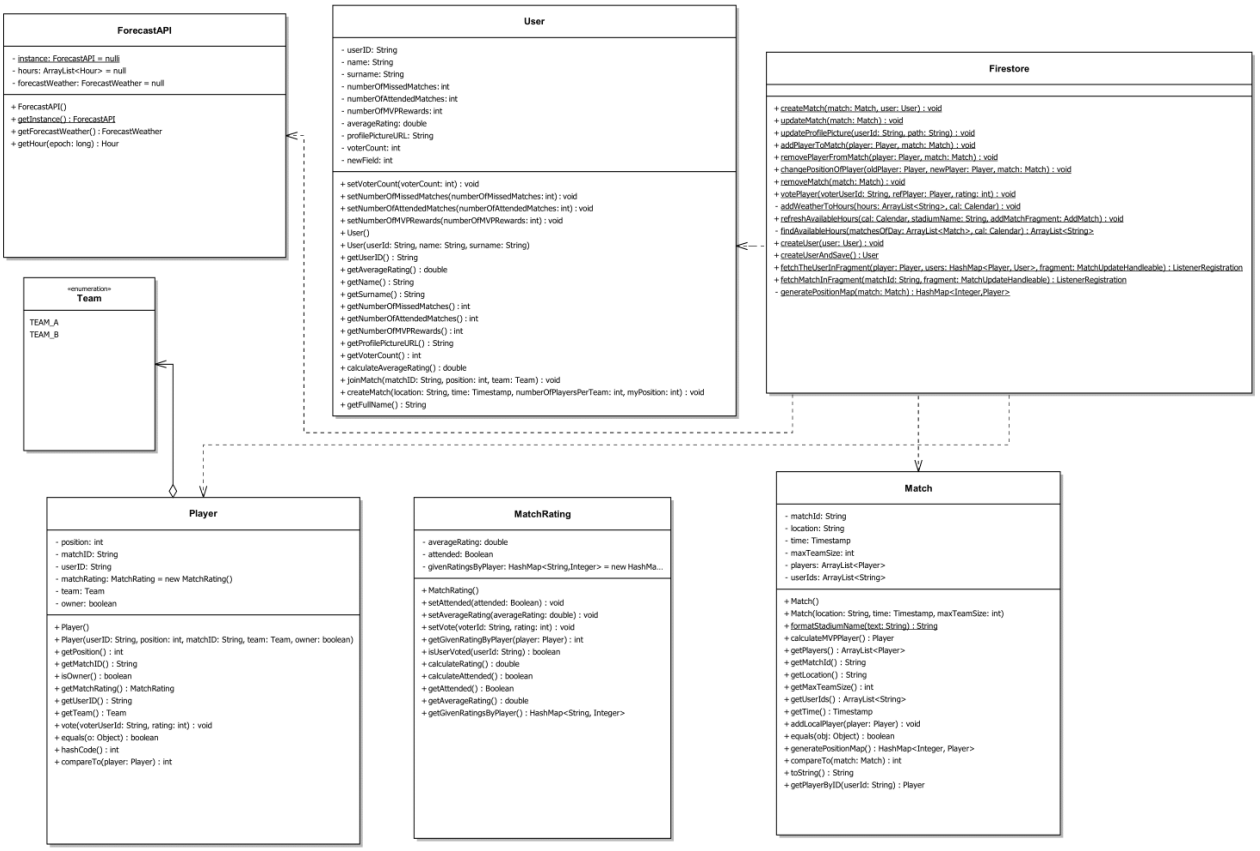
2.1 System Overview

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



BilkentHalısahaUML.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. Firestore 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 Strings.

5. 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.

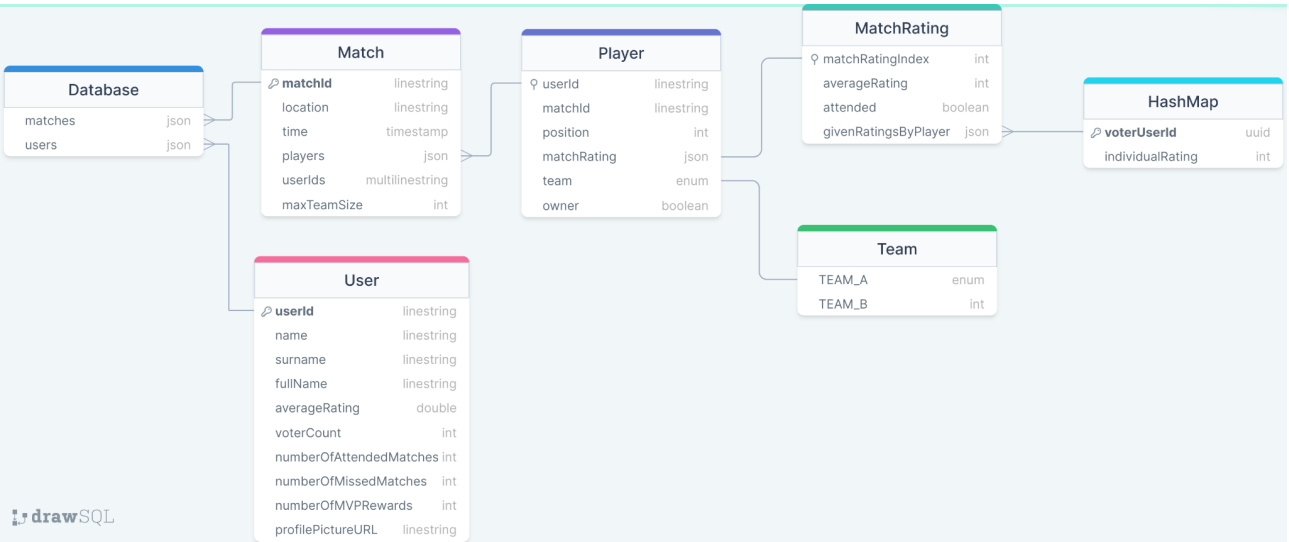
6. ForecastAPI and Weather Classes

ForecastAPI returns the current weather information and forecast weather information according to time. It is returned as a weather object.

7. Team

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

Database Structure

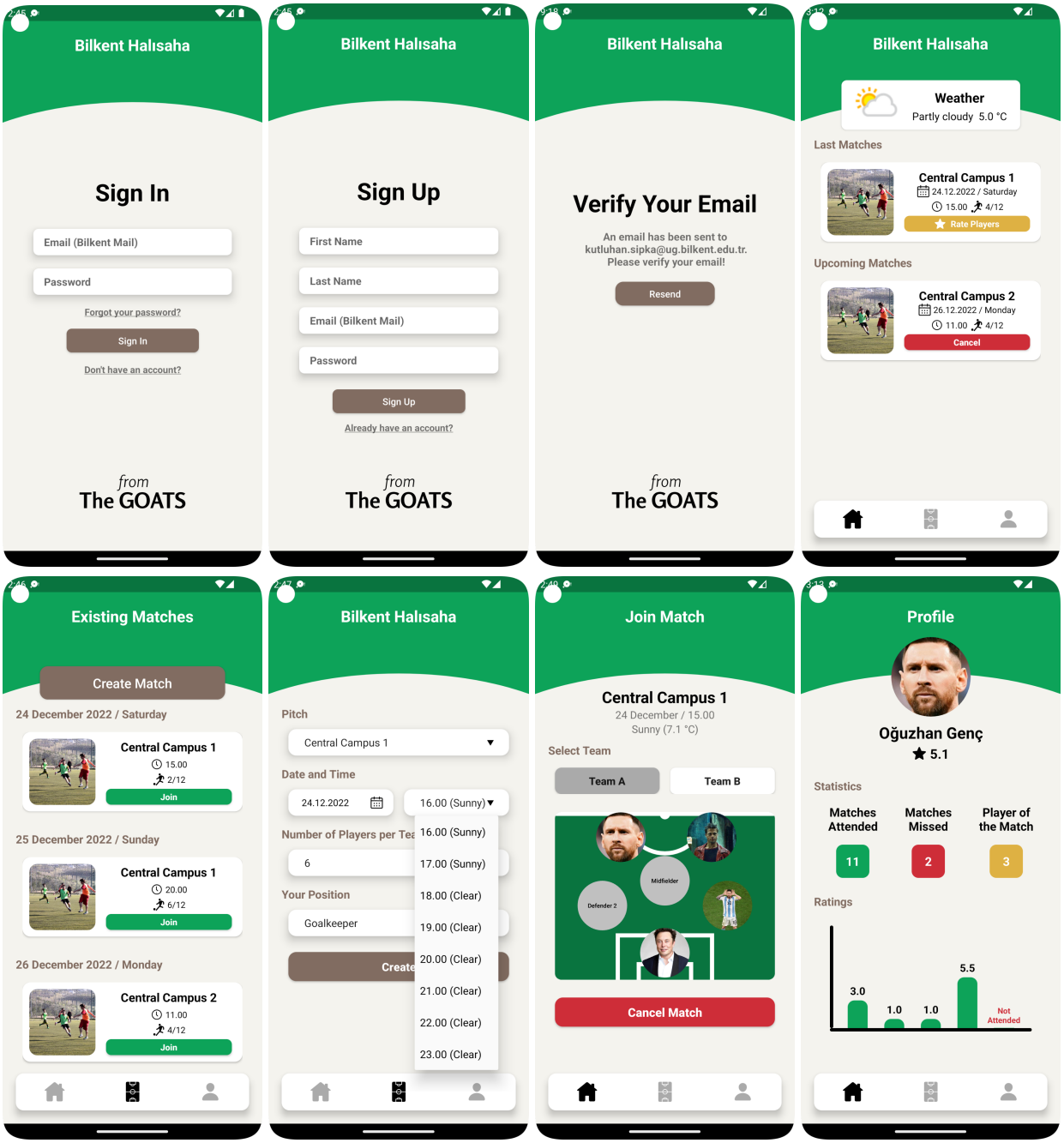


DatabaseDiagram.png

Also, here is our database structure. Database stores User and Match objects and other objects are 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 IDs is stored in the database. In contrast, our client, which is the Android application, stores these objects as nested objects. We have used realtime updates, pagination and querying while fetching data from the database and used Firestore for these technologies. Thanks to that, our app is scalable and can work with big number of matches.

We have used Firebase Storage for storing profile pictures and used Glide to cache them in local devices and decrease the bandwidth usage of clients.

2.3 Screenshots



You can use this link to see the demo video of our app: [Link]

2.4 Achievements

We have almost achieved all of our goals. To illustrate, we successfully implemented weather API, firebase integration, the rating system for matches, and graphs for the last matches in the profile. We planned to implement a request-sending system. However, while developing our app we thought that it would be difficult to implement that. That's why, we included a player-kicking system instead of a request-sending system, which enables the owner of the match to kick any of the players if he does not want to see him in the match. Also, we added a new feature that is forecast weather information while creating or joining a match which is not stated in the our requirements report. In addition, we tried to create a scalable structure by using Firestore rather than using Firebase Realtime Database. Thanks to Firestore, we use queries, realtime updates, pagination and

composite indexing and thanks to these technologies our system can work big number of matches and users without using too much network bandwidth.

2.5 Tasks

1. Kaan Aydeniz: Worked on Firestore class and integrated with Firestore in synchronous structure to all fragments of the app. Implemented recycler view adapters for the some recycler views and integrate these recycler views with Firestore. Integrated Firebase Storage with Glide. Implemented adding, selecting and uploading image for profile picture of players and integrate with these with Firebase Storage. Implemented some methods to object classes like MatchRating and Player. Integrated time spinner of AddMatch class with ForecastAPI and Firestore.

2. Erdem Aydın: Implemented some methods in MatchRating class and adapter, created profile, rate players and player profile fragments.

3.Oğuzhan Genç: Worked on UI fragments such as sign in, sign up, homepage. Implemented some navigation parts of app and implemented adapters for home page recycler view part. Moreover, worked on UI part of weather functionality. Finally, worked on app icon.

4. Furkan Emre Kolukırık: Created model classes such as User, Player, MatchRating, Match, Team Enum, worked on fragment classes, added dialog boxes to application and prepared the application for firebase integration.

5.Hasan Kutluhan Şıpka: Worked on Sign In, Sign Up, and Email Verification activities. Implemented AddMatch class, ForecastAPI class, Profile class, MatchInfo class and Weather classes. Also, created UI fragments such as display matches, create match, and profile. Moreover, handled the auth operations, bottom navigation, and splash screen. Finally, worked onobi all app designs.

2.6 Final Reflections

We all enjoyed the project process. We had difficult times but handling the problems we confronted was very entertaining to solve. The most difficult aspect of the project was assigning tasks and merging the different works. All of us have experienced difficulties ourselves; yet, we overcame all the obstacles that we have faced by helping each other. If we have a chance to start over again, we would probably start learning Android Studio earlier and understand the basic concepts of Android Studio and Firebase. We spent 2.5 weeks on our project implementation. The reports we have written during the course helped us very much. We are very proud of the final product. We think that we developed a very complete, useful, scalable, and user-friendly app.

3. Summary & Conclusions

To sum up, our project is a mobile application that allows football players to create and join matches. We 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. All of our members completed their tasks and we have created our app and fulfilled our requirements successfully.