SPORTMATE:

FIND YOUR SPORTS PARTNER ONLINE

by

JIANRUI HU

A report submitted to the faculty of

The University of Utah

in partial fulfillment of the requirements for the degree of

Masters

in

Software Development

School of Computing

The University of Utah

December 2018

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I like sports, however most of the sports are require two or more people. Sometimes it become pretty hard to find people to play with, special with someone who have closer skill. So, I want to build an android application that help me find their sports partner.

Here are the features that I am looking for in my app:

1. Sports search/match
2. Location search/match
3. Shill level search/match
4. Schedule search/match

In order to achieve those features, each user can pose their own ticket which contains sports type, skill level, location and the time. All the user can view the current exist ticket and pick the ticket that you want to join. By this way, people who don’t know each other can play the sports together.

EXIST APPLICATION

Before I star to design my application, I went to the Google Play Store and search for the existed that can provide similar feature.

**Basketball:**

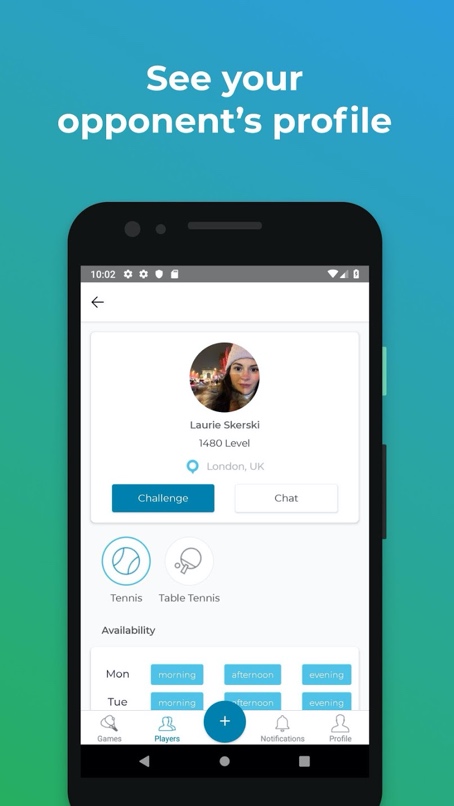
First, I try to find the application that can help people to find partner to play basketball together. However, there is no such application on the Google Play store, I think the reason for that is most of the basketball court are indoor, and usually there is a lot of people playing the basketball all the time. So, people can just come to the basketball anytime, and not worry about there is not enough people to have a game. And another reason is that basketball is not like other sports, even you are the only one in the basketball court you can still play the basketball alone, like practice shooting skill.

**Tennis:**

Next, I searched for the application the help people find tennis partner. In the Google Play Store, there is more then ten different application that can help people to searching for the tennis partner. Here I will choose an application that called “RacketPal: Find local racket sports partners today” (<https://play.google.com/store/apps/details?id=com.racketpal>). This application not only help people to find the partner in tennis, it also helps people find partner in Badminton, Table Tennis, Squash and Padel. Base on the information that provide by Google Play Store, there is more than one thousand people have installed this application. Here are some screenshots for this application (See the figure on next page). From the screenshots we can easily know that the application somehow gives everyone a level number. Base on ether you won a game or lose a game with other user in this application, you will increase or decrease the level number. In my opinion, people should more enjoy the sports itself, but not beat someone else. I do agree that win or lose is part of the sports charm, however, to measure in a number skill is not the right way. It will make people only looking for the player who have a lower level number then their selves. Also, it is not a good thing to know the people that you play with are better than you base on the application level number. So, in my application I do want to have some way to measure the skill level for each sport, but I won’t use the system like this level number. I am more willing to use “beginner, intermediate, advanced, professional” to range all the users. In this way, people can play with the people that have the similar skill level, but no idea who is actual better than other. I believe people will more enjoy the sports in this way.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated 

A close up of a computer

Description automatically generated A screenshot of a cell phone

Description automatically generated A close up of a computer

Description automatically generated

*Figure 1.1 RacketPal Screenshot*

**Badminton:**

When I search for the application that help people find badminton partner, I found an application call “Playo - Find Players, Book Venues, Manage Groups” (<https://play.google.com/store/apps/details?id=com.techmash.playo>). I believe it is the most popular application that provide this feature. Base on the information that provide by Google Play Store, there is more than five hundred thousand people have installed this application. This application not only provide partner searching for badminton, tennis and basketball, it also provides partner search for other fifty-nine different kinds of sports. Here are some screenshots for this application (See the figure on below and next page). From the review and the number of people using this application we can easily know that this application is really good. However, from the screenshot we can know that this application not only provide the feature to find sports partner, but it also has a lot of build in social features that help people communicate to each other. It for sure will bring users closer and easy to become friend. However, I want my application to be simple. We already have lots of application provide communication, such as Facebook, WeChat, Line. There is no need to have communication in this application. If they want to become friends, they can have connection in other more popular communication application.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

*Figure 1.2 Playo Screenshot*

CLASS DESIGN

The main idea for my application is that every user can submit their ticket online. And each ticket will contain sports type, skill level, location, date, time and the people that involve in the ticket. At home page everyone can look at the exist tickets and pick the ticket that they are interested. By this way people can join the game together.

There are two main class using in this application, one is the User class, and another is the Ticket class. The User class will contain the information for each user like a profile, and the Ticket class will contain the information for each ticket, which will help to filter the ticket display and help user to search for the ticket.

**User Class:**

There are eight different variables in User class, there are

1. private String id;
2. private String name;
3. private String gender;
4. private String zipCode;
5. private String badminton;
6. private String basketball;
7. private String tennis;
8. private List<String> ticketID;

The string id is the key for this entire class. This id is generated by the firebase Authentication which I will talk about later (See Page X). All the information for this user is store at the firebase cloud storage under this key which is the string id. By using this string id, the user is allowed to have access to their user information, either they can change their profile and write to the cloud storage or read their profile from the cloud storage. String name, gender, and zipCode are working as their naming, their store the user’s name, gender and the zip code. String badminton, basketball, tennis is working slight differently. They represent the skill level for these three different kinds of sports. There are four different selection that you can chose for the skill level, they are “beginner, intermediate, advanced, professional”. The List<String> ticketID contains all the ticket id that created by this user. In this way, the pose section (See Page X) can easy show all the tickets that involve by current user. Because in this way, the application doesn’t need search for the entire database to find all the tickets that related to this current user.

In all these eight different variables, string name, gender, zipCode, badminton, basketball, tennis are the front end information which will show in the android application to the current user; the string id and List<String> ticketID are the back end information which will not show in the android application to the user. They are the information that help the application to extract other information. Here is how the User class looks like in the firebase cloud storage (See the figure below).

A screenshot of a cell phone

Description automatically generated A screenshot of a social media post

Description automatically generated

*Figure 2.1 User Class in firebase cloud storage Figure 2.2 Ticket Class in firebase cloud storage*

**Ticket Class:**

There are seven different variables in Ticket class, there are

1. private String id;
2. private String sports;
3. private String level;
4. private String zipCode;
5. private String date;
6. private String time;
7. private List<String> userID;

The string id is the key for this entire class. All the information for this ticket is store at the firebase cloud storage under this key which is the string id. By using this string id, the user can extract the ticket information, more important, the user can searcher through all the ticket by this string id. This id is combining by the current time stamp and the current user id. When a user is creating a new ticket, the id for this ticket will be current DATE + TIME + USERID. If you look at the figure 2.2 Ticket Class in firebase cloud storage, an example will be “2019106124143710KxjofP3KWZdb5E62iiklQaP4r853”. When this ticket created, the date was 2019 Oct 6th, the time is 12:41:43 and 710 milliseconds. The rest string is the user id. The reason I add the user id to the ticket id is that if we have huge amount of user for this application, their will have a high chance that create conflict when people try to create new ticket at excited same time. Adding the user id to the end of the ticket id can easy fix the conflict. Because one user can never submit two tickets online at same time.

String sports, level and zipCode are working as their naming, their store the sports name, skill level and the zip code which will be location. String date and time are also working as their variable name, they represent the event date and time when this game will setup. The List<String> userID contains all the user id that involve by this ticket. In this way, the list section (See Page X) can easy keep track how many people have join this event. Because in this way, the application doesn’t need search for the entire database to find all the users that related to this ticket.

In all these seven different variables, string sports, level, zipCode, date, time are the front end information which will show in the android application to all the user; the string id and List<String> userID are the back end information which will not show in the android application to the user. They are the information that help the application to extract other information. Here is how the Ticket class looks like in the firebase cloud storage (See the figure above).

LAYOUT DESIGN

The overall workflow is show as the figure below.



*Figure 3.1 overall layout workflow*

**Welcome Page:**

When the user opening the application, I will first load a welcome page (See figure 3.2 below). The welcome will stay for display for 3 second to 5 second. The reason that I want a welcome page at very beginning of the application is that I am using firebase (See page X) as my back end, and it usually take about 10 to 20 second to build connection between the application and firebase cloud storage. Rather than spend time to waiting in the login page, I separate the waiting time to two part. A little bit waiting in welcome page, and some waiting time in login page. In this way, the user experience will be much better than waiting in the login in page for 10 seconds. Here is what my welcome page looks like.

A screenshot of a computer

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a social media post

Description automatically generated

*Figure 3.2 welcome page Figure 3.3 login page Figure 3.4 sign up page*

**Login Page:**

In the login page (See figure 3.3 above), user just need enter they email address and the password. All that information is store in the firebase Authentication, all the information is encrypted and hidden for everyone. Even as the administrator for this application and severe, like me still can’t know the user’s password. I can see all the user’s email and delete their account but cannot achieve their password, which is good for program and server security. Firebase Authentication will auto check for the password, if the password is match to the server, the application will jump to the home page and show a “successful login” task at the bottom of the screen. If the password is not match to the server, the application will stay in this login page, the email will remain what the user have typed, but the password part will wrap out and become blank. At the same time, it will show a “successful fail, please try again” task at the bottom of the screen.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

*Figure 3.5 profile create/edit page Figure 3.6 home page Figure 3.7 profile page*

**Sign-Up Page:**

For the people who is using this application the first time, they will need to click the “sign up” button to jump to the sign-up page (See figure 3.4 above) to create their account. It requires their email address and the password. Because everything typing in the password will not be display, in order to make sure the user is typing what they want without any mistake I request a second time to enter the password. Only the two passwords enter are match each will allow you to create the new account, otherwise it will show a “the password won’t matching” task at the bottom of the screen and wrap out the second line of password make it become blank.

**Profile Create/Edit Page:**

After the sign up, the application will jump to the profile create page (See figure 3.5 above). The user will enter their name, gender, zip code, badminton skill level, basketball skill level, and tennis skill level. Submit button will grab all the information that user has entered and using that information to create a new User class. And then send this class to the firebase cloud storage (See page X) and jump to the home page (See figure 3.6 above). If you already have an account, after login, you will also jump to the home page.

**Home Page:**

There are three section in home page (See figure 3.6 above), first is the list recycler view for all the ticket, second is the pick button, and the last one is the navigation bar.

The list recycler view can be scrolling up and scrolling down, it will display all the tickets after time filter and the skill level filter. The time filter will filter out all the tickets that are already expired. User only interested on the game that will happen in the future by not the game that already passed. At the same time the skill level filter will choose the game that match your profile skill level. For example, if you set your badminton skill to “intermediate” in the profile, then it will only show the badminton ticket which are also intermediate. In this way, the application will help the user to ignore the extra information that they don’t need. It will help the user looking through all the ticket easily and find the ticket that matching their schedule.

The list recycler view is selectable, user can select each ticket by click it. And then clicking the “pick” button below will add the current user to the ticket that you selected. Technically, it will get the ticket id that your selected, and pull out all the information from this ticket class. By click the “pick” button, it will add your user id to the List<String> userID variable under this ticket class. In this way, the user has joined the game that they want. Last, at very bottom of the screen, there is the navigation bar, which help user to jump into different pages.

**Ticket:**

In each ticket, there is five pieces of information showing up. On the left side is an image that will represent the sports type. The reason I am using the image rather than the text words is because that people absolve information from the image is faster and easily than from the text words. On the right side, it shows the event date, time, skill level, and the number of people is currently involved in this event. The date, time and skill level are directly extract from the ticket class and the number of people involved in this event are calculated by the length of the member variable List<String> userID inside the ticket class.

**Navigation Bar:**

The navigation bar will display all the time on the very bottom of the screen. There are four buttons on the navigation bar, home, profile, ticket and logout. Each button will help user to jump into different pages. The “home” button will take the user to the home page (See figure 3.6 above). The “profile” button will take the user to the profile page (See figure 3.7 above). The “ticket” button will take the user to the ticket page (See figure 3.8 below). And the “log out” button will allow the user to log out their account and jump back to the login page (See figure 3.3 above).

**Profile Page:**

The profile page (See figure 3.7 above) shows all the user information that they entered when they create the account. The information including their name, gender, zip code, badminton skill level, basketball skill level, and tennis skill level. At the bottom of this page, there is a “edit” button. By clicking this button will allow user to update their profile information, so it will take the user to the profile create/edit page.

**Ticket Page:**

The ticket page (See figure 3.8 below) is pretty similar to the home page. There are also three section in this page, first is the list recycler view for all the tickets that involve with the current user, second is the pose button, and the last one is the navigation bar. In this list recycler view, there is also two filters, one is the time filter which will be exactly the same with the time filter in home page. It will filter out all the tickets that are already expired. Another filter is the user id filter. This filter will searcher though all the tickets and pick the tickets that involve by the current user. The tickets can be either created by the current user or the current user had joined the ticket. Below the list recycler view is the “pose” button, which will bring the user to the pose page.

**Pose Page:**

The pose page (See figure 3.9 below) is how the user can create new ticket online. In this page, it will ask the user to enter the sports type, skill level, zip code, date and time. The application will check each link to make sure that all the input is valid. After clicking the “submit” button, the application will use that information to create a new ticket class and send to the firebase cloud storage. Now the new ticket will show on both home page list recycler view and the ticket page list recycler view.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

*Figure 3.8 ticket page Figure 3.9 pose page*

FIREBASE

Firebase is a Backend-as-a-Service (BaaS) developed by Firebase, Inc. in 2011. The company was acquired by Google in 2014, then the firebase grew up into a mobile and web application development platform on Google Cloud Platform. As of October 2018, the Firebase platform has 18 products [2], which are used by 1.5 million apps. In my application, I am using two features from the firebase. There are Authentication and Realtime Database.

**Set Up:**

Before adding Firebase to my Android app, I need to create a Firebase project and then connect it to my Android app. When we create a new Firebase project in the Firebase console, we're actually creating a Google Cloud Platform (GCP) project behind the scenes. A GCP project is a virtual container for data, code, configuration, and services, and a Firebase project is a GCP project that has additional Firebase-specific configurations and services.

A Firebase project is the top-level entity for Firebase. It provides a Firebase configuration file (Android/iOS) or a configuration object (web) that need adding directly into the local project.

For iOS, we need add a “GoogleService-Info.plist” configuration file;

For Android, we need add a “google-services.json” configuration file;

For web, we need add a Firebase configuration object for our initialization scripts;

In this case, we are only need add an android application to the firebase project. After add the “google-services.json” configuration file to “app/src/” folder, the application now is connecting to the firebase project that I created.

Next, I need add Firebase SDKs to my application by implementation dependencies into the “build.gradle” file. After sync, my application will have all the environment that allow it to interactive with the firebase. Now everything is set up and ready to go.

**Authentication:**

My application needs to know the identity for the user. Knowing a user's identity allows my app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to my app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. In my case, I am using the password.

To sign a user into my app, it first gets authentication credentials from the user. The credential is the user's email address and password. Then, my application passes these credentials to the Firebase Authentication SDK. Firebase backend services will then verify those credentials and return a response to my application.

After a successful sign in, the user can access their basic profile information, and the data stored in Firebase Realtime Database.

**Realtime Database:**

Firebase realtime database is a NoSQL cloud database. Data is synced across all user in realtime and remains available when the application goes offline. Data is stored as JSON and synchronized in realtime to every connection. All of my application user share one Realtime Database instance and automatically receive updates with the newest data.

My realtime database have two children, one is the user and other is the ticket. The user child will store all the user profile information which is the user class that I created (See figure 2.1). The user class can automate format as a JSON file and push into the realtime database. Similarly, the ticket child will store all the ticket information which is the ticket class that I created (See figure 2.2). The ticket class can also automate format as a JSON file and push into the realtime database. In this way the user can extract and update the information from the realtime database back and forward.

The firebase makes up my entire back-end server. It provides the identity for the user and the data cloud storage.

REFERENCE

1. Tamplin, James. "Firebase is Joining Google!". Firebase, Inc. Retrieved October 22, 2014.
2. "Firebase Products". Firebase, Inc. Retrieved October 31, 2018.