1. Introduction

1.1 Marketing Blurb

Tracking ONLY your step too dull? Forget about Google Fit / iOS Health!
Tired of looking at maps with only cold icons? Forget about standard Google Map!
Want to make friends with healthier people? Forget about Facebook and Tinder!

Try Mosso!

An app that can track your steps and healthy status.
An app that can make friends and see your friends' steps.
An app that can see your friends running and walking near you realtime!

Because you could not find an app in the real world so far like Mosso;
that could track your steps anywhere and anytime,
that could track others steps anywhere and anytime,
The app that could make competition with your friends by working steps!

1.2 Purpose

We want to build a health and step tracking app.

Currently in the market there are apps that can track steps (Google Fit, Apple Health), but they do not have any friends functions or a well built map. In the meantime while there are many social media apps that can make friends with each other, there are none to make healthy friends and show steps / health stats with each other.

We are targeting this combination area.

We want to build an app that

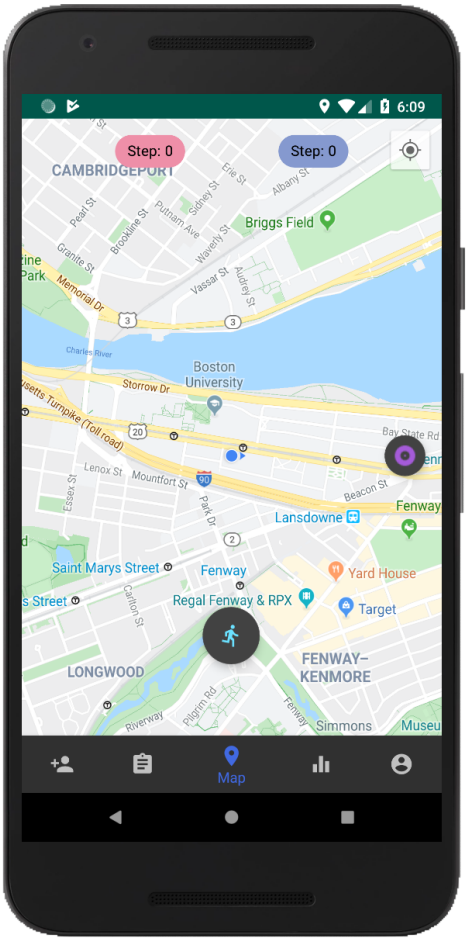
1. Can track the user's steps and health status, can be used as a regular step tracking app.

2. Can also see where the user is, can see a map, and can track the location history of the walking
3. Can make friends, see friends' steps, and see friends' real time locations when they are walking.
4. Can participate in teams to compete by the number of steps walked.

In summary we are seeking to build an app that utilizes multiple technologies and targets the combined area of these technologies.

2. App Walkthrough

2.1 Main Page



This is the main page of our app.

The main page first shows the map, and the user will be shown by a blue dot -- identical to Google Map.

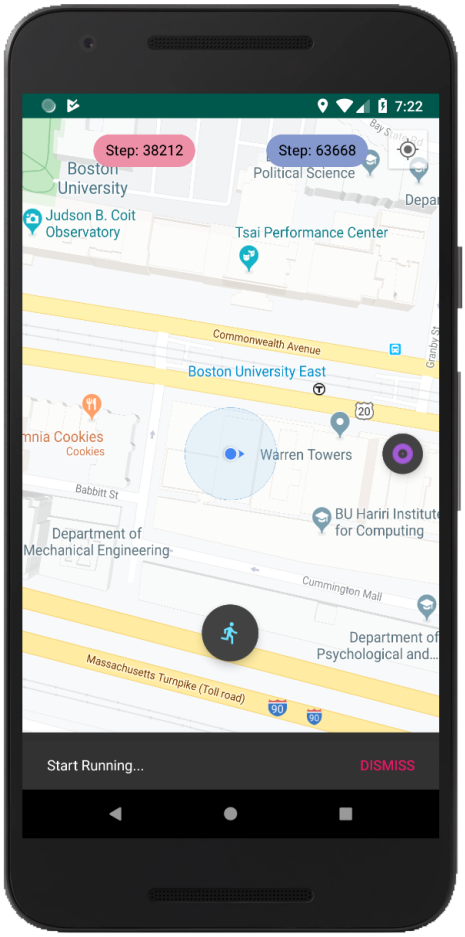
On the bottom of the app is the navigation bar, which can be used to switch different pages (Implemented by fragments).

In the middle of bottom there is a Run button.

To the right of the screen, the purple button toggles the layers of the map.

In the end the total steps of both team Red and team Blue could be seen on the top.

2.2 Main Page - Run Button



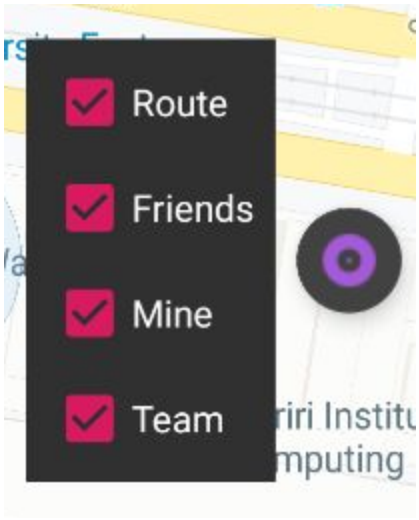
After the Run button is pressed, the app will go into the Running mode.

In the running mode,

1. Route will be tracked and shown by line, steps are tracked all the time
2. Now the user's running data (location, steps etc) will be uploaded to the server every 10 seconds. AWS Lambda and Redis will record these data.
3. At the same time, other users will be able to see you running on their map.

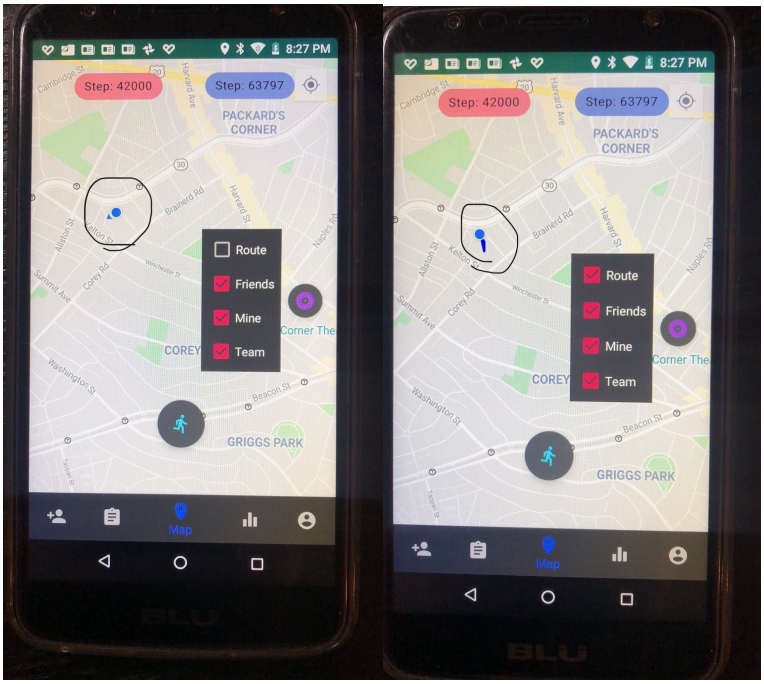
The running feature is the main feature of the app. All other features are built to support the user experience of this feature.

2.3 Main Page - Map Layers



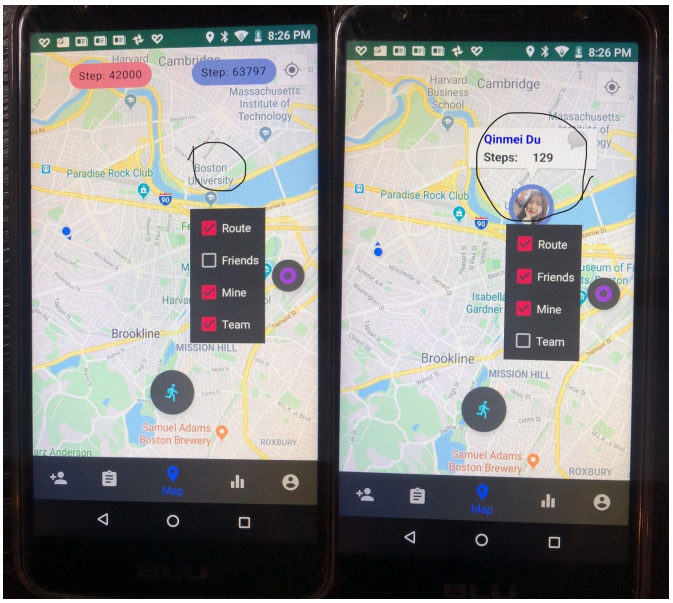
The purple button to the right of the screen controls layers of the map. We provide this button to users to toggle some display preferences, so that users can choose what to be presented on their screen.

1. Route layer



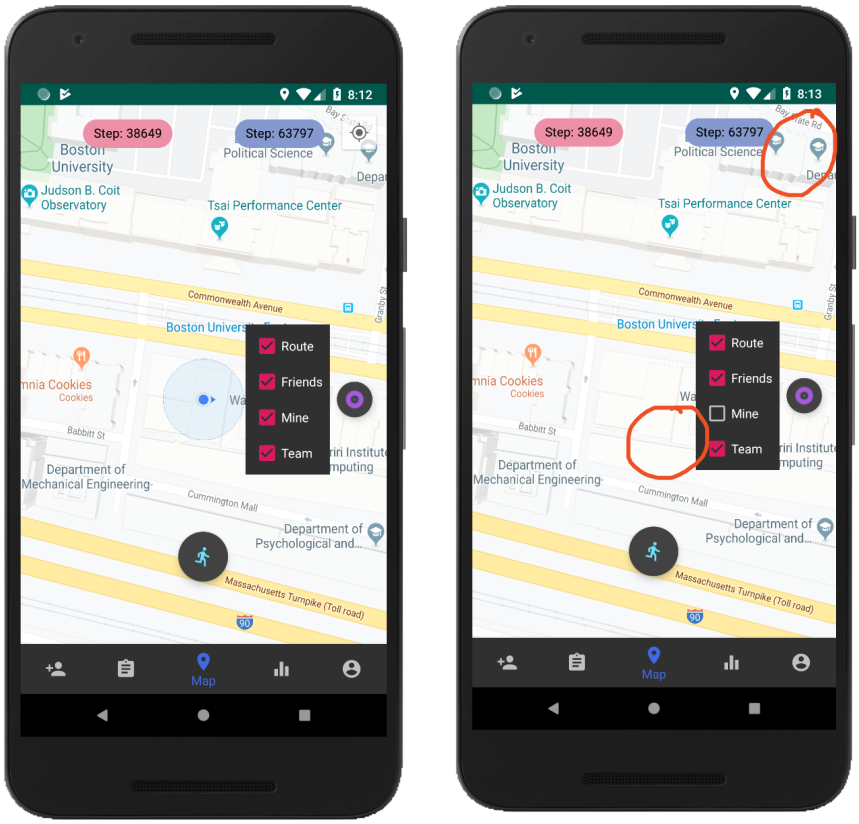
Route Layer toggles the display of the user's running routes.

2. Friends layer



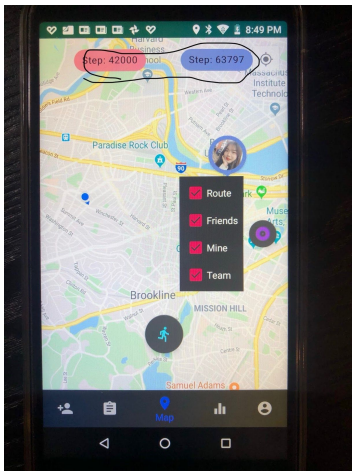
Friends layer toggles the display of other friends.

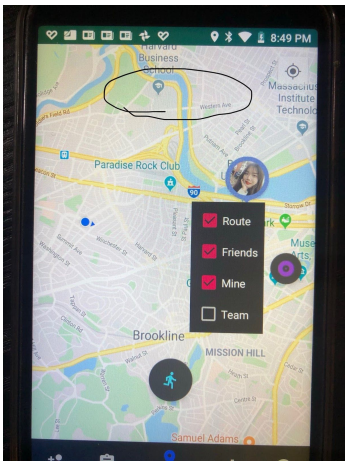
3. Mine layer



The ‘Mine’ layer toggles the display of the user him/herself, and also the basic functions of Google Map.

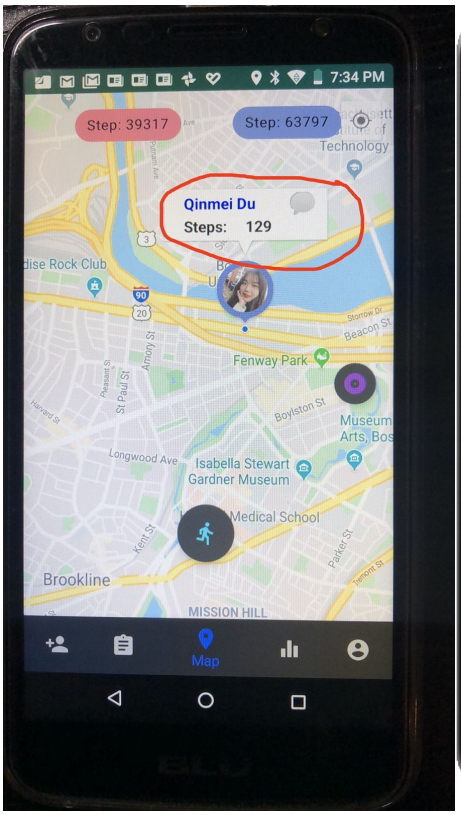
4. Team layer

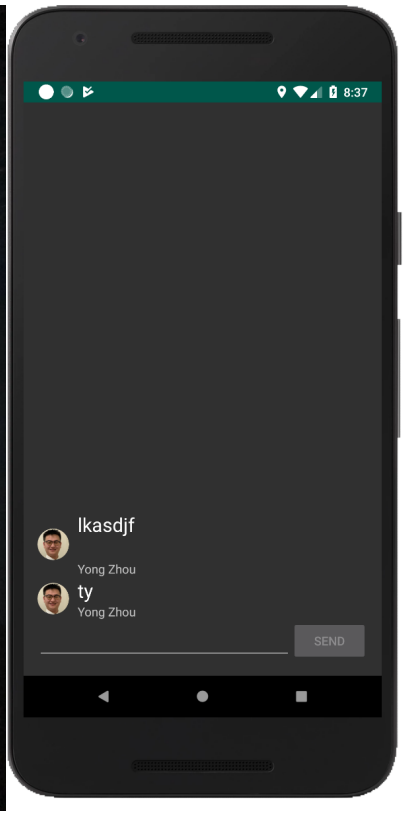




Controls the display of the team label and teams' steps.

2.4 Main Page - Showing and Interacting with Other Users





One important feature of our app is the ability to make friends and interact with them during running.

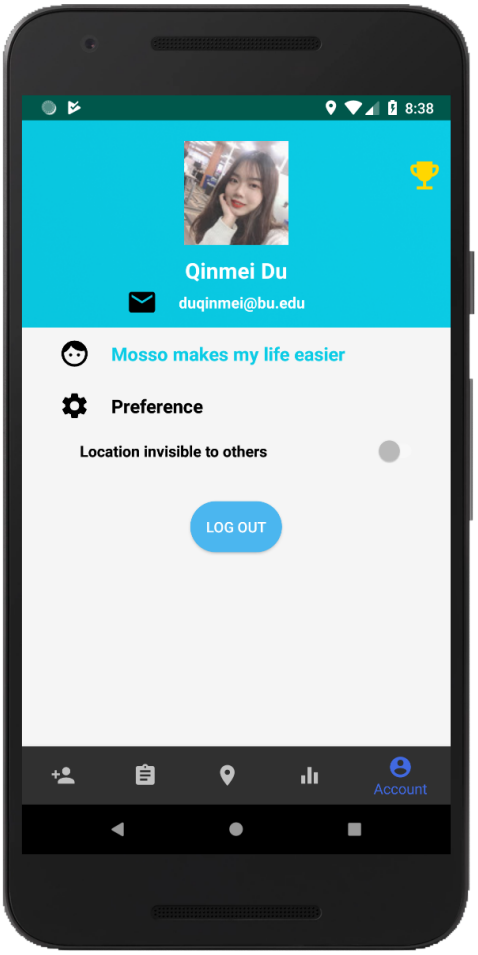
During the run, user can see their running friends in real time on their map. The friend's name and step are shown and updated.

Users could directly chat with the friend by clicking the info window

Details of the chatting feature will be introduced below.

In the meantime, the team steps are also updated in real time. Each user will contribute to their team's total steps. The red/blue team's steps will be updated as the runners are running.

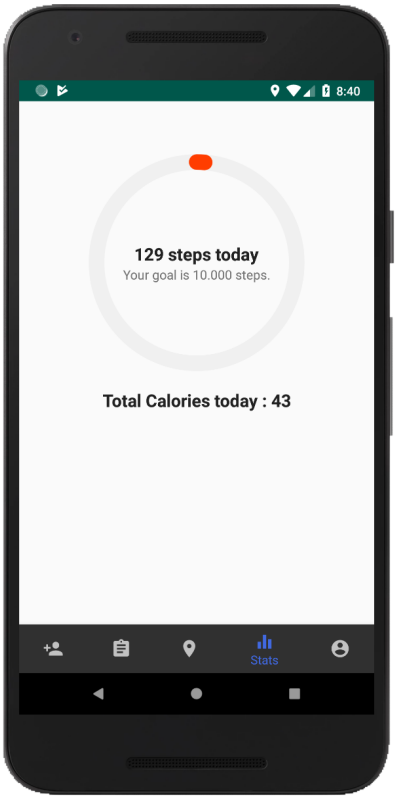
2.5 Profile Page



This is the profile page showing the user's basic information, and preferences.

We read user's Google account's profile photo (if available) to be the user's photo.

2.6 Stats Page



The other main feature of our app is step tracking.

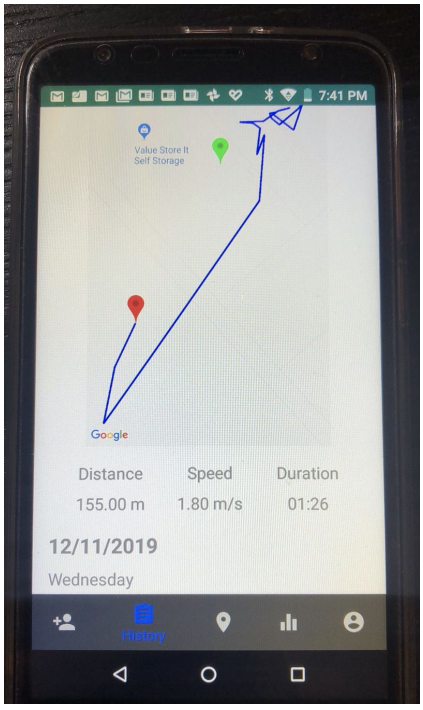
The Stats page shows the user's steps and calories today, data obtained from Google Fit API.

The whole circle indicates the total step goal for today, giving the user an obvious animation of how many steps are achieved today.

Below also shows the calories of today.

Any steps recorded by Google Fit can be shown here. All the steps are connected by Google services.

2.7 History Page

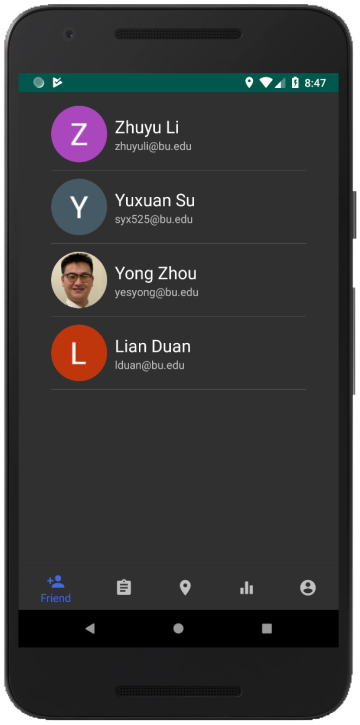


After each running, the running track and running information are automatically recorded.

The information includes the running routes, distance, average speed and running duration.

User can switch to the History Page to check each of the running histories.

2.8 Friend and ChatPage

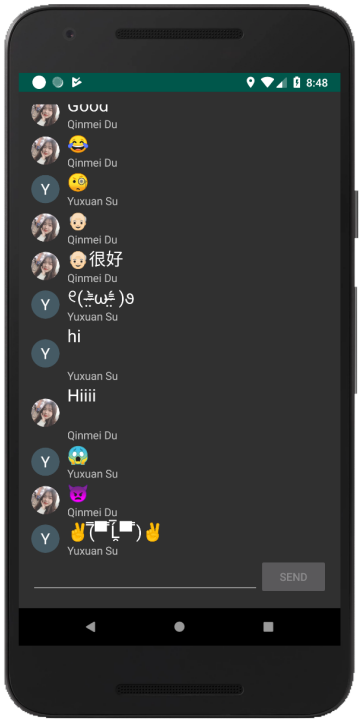


One another important feature is friends.

In the Friends page the user can view all his/her friends. Friends' names, emails and their photos are shown in this page.

By clicking the friend, a chatting page will pop out. The two users are able to chat in this private chatting room.

The chatting room not only supports basic English, but also support emoji and even Chinese language characters.



As mentioned above, the chatting window can not only be triggered in the Friend page, but can also be triggered in the running window. If the user sees a friend's running and click the chat icon beside the name, they will also be directed to this screen and start or continue the chat.

3. Implementation Details

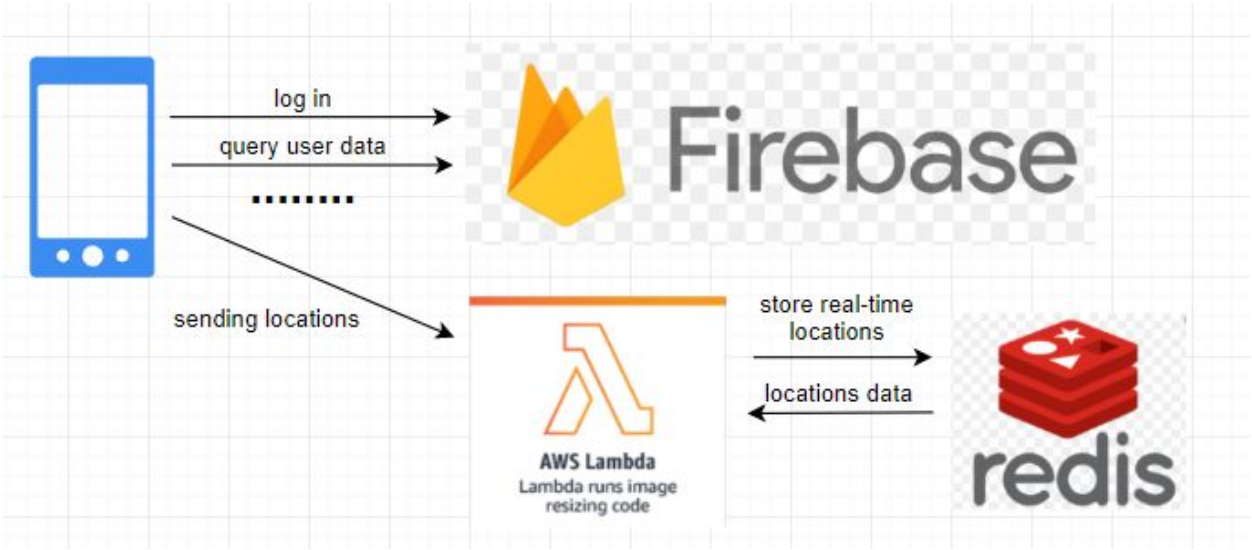


Figure: Backend Structure Overview

Backend: Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

In our application, we store all the real-time locations to the redis database. Redis allows us read&write real-time locations fast and easy. Besides, redis support geocode, it's an efficient way to store geo-location data.

Backend: AWS Lambda

Function as a Service (FaaS), or Serverless computing is a cloud-computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. One characteristic of FaaS is that pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.

Different from Firebase of usual Cloud services, where usually the cloud service will give user exact number of virtual machines, FaaS does not need to give the machine resources to the user all the time. It will be based on functions, and have elasticity for usage on the virtual machines. If there are no usages on the cloud services we do not need to rent the whole machine, and when there is a large demand on resources the FaaS platform can automatically scale up the service for the users.

AWS Lambda is the first and the most famous FaaS service now on the market. And we are using Lambda together with Redis to implement the service for saving and broadcasting the running stats (steps and location).

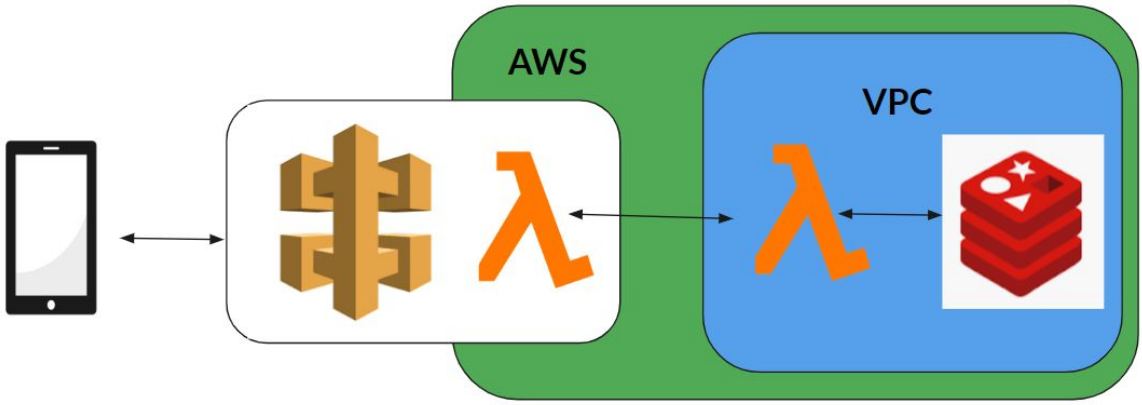


Figure: AWS - Redis Double Layer Trigger Pattern

The real time display mechanism in the main page are all using this technique. And by using Lambda with Redis, we ensure:

1. The service can handle a large quantity of requests : when lots of users start running at the same time. And we do not waste computing resources when at night no one is running.
2. The updating of location is optimized. It may not be felt from using the app, but by using Lambda with Redis, we ensure that useless stale data are removed in a well designed manner.

In a word, we implemented our services to be both scalable and well designed. We do not simply rely everything on Firebase, which is easy to start, but focus on building something durable and trying some cutting edge technologies.

Location updating: Android service

We utilize the Android Service to build a stable and accurate location tracking system in our application.

Android Service is an application component that can perform long-running operations in the background, and it doesn't provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

These are the three different types of services:

Foreground
A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a Notification. Foreground services continue running even when the user isn't interacting with the app.

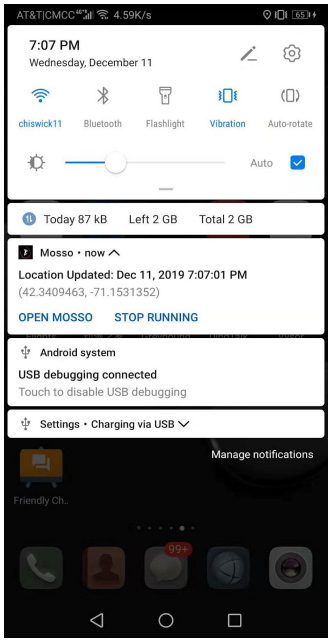
Background
A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

Bound
A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application

component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

In our application, we implement a location tracking service, it will first become a background service, when our application goes to background, the service becomes a foreground service and keep tracking users' locations.

When the application goes to the background, our service will keep tracking the locations:



Database: Firebase

Firebase is a mobile and web development platform that provides developers with a lot of tools and services to help them develop high-quality apps, grow their user base, and earn more profit. Among all the tools that firebase provide to the developer, we mainly used cloud firestore. Firestore is actually a database, we used it to store and sync data between users and devices. Cloud firestore gives us live synchronization and offline support along with efficient data queries. We used it to store users' profile data and users' running data.Also, firebase provides a service called firebase authentication, by using it, users can use their google account to log into our application. So they can get rid of the redundant work of creating and maintaining a new account. Besides, firestore allows the device get

real-time update when the data in the database is changed. So we implement a chat room based on it. Users can send messages and talk to their friends in real-time now.

Step Tracking : Google Fit

We use Google Fit API to track the steps of our app.

Google Fit is an open ecosystem that allows developers to upload fitness data to a central repository where users can access their data from different devices and apps in one location:
Fitness apps can store data from any wearable or sensor.
Fitness apps can access data created by other apps.
User's fitness data is persisted when they upgrade their fitness devices.

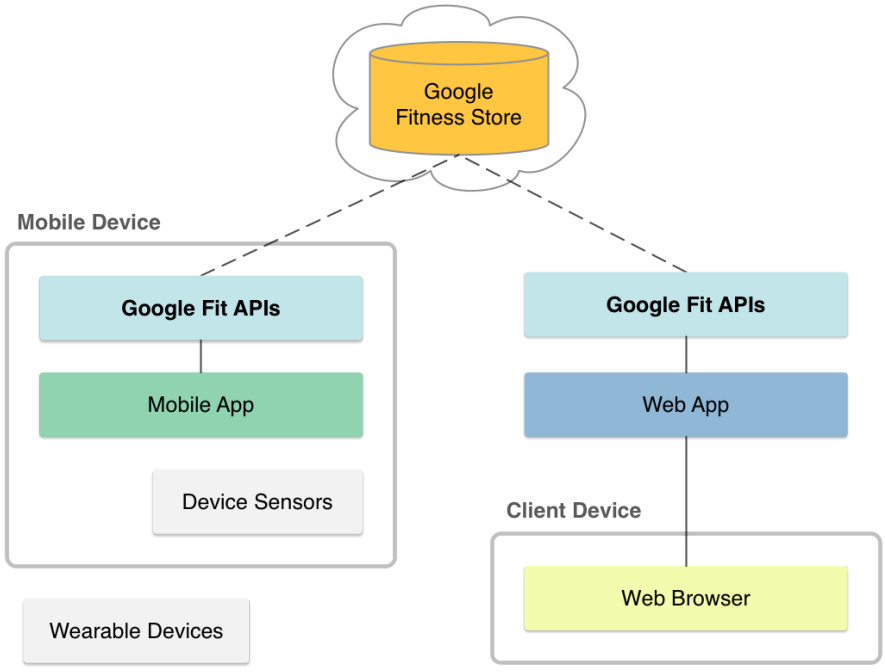


Figure : Structure of Google Fit API

Using Fit API enables the app to read the history data connected to the user's account. In other words, all the Google Fit app's data are connected and saved in the user's Google account. No matter when the user is using the Google Fit's app in an Android phone, iPhone, Android watch or Apple watch, if Google Fit is used and enabled, the history data will be recorded and uploaded to Google's server.

On the other hand, on Android devices, Google Fit API is embedded in the Google service, and therefore reading the steps can reduce the used resources. In this way the app can make full use of the Android framework.

Our app's strategy is to utilize the Fit API in this way: we ask the user the give us the permission to read their Fit data, and we load the detailed history data using Google Fit API, and then analyze the data to generate the step data we are looking for within a day.

MVVM

Front-end refreshing: MVVM

We used a relatively new technology in this project called MVVM (Model-View-ViewModel), which is a data-driven structure. We'll not cover too much knowledge about MVVM, just talk about how MVVM benefits us at location synchronization process.

The logic of location updating is in a specific Android service, when data in service is updated, we also updated UI in MapFragment. There might be NullpointerException for traditional pattern since MapFragment may be destroyed. By using MVVM, we can let MapFragment observe the location data and let MapFragment refresh the view when location data is changed. By doing that, when location data is updated, if MapFragment is created, it observes the change and update the UI, if MapFragment is destroyed, then no one observes the change of location data and nothing changes.

In summary, using MVVM helps us avoid some leakage issues which usually cause NullpointerException error.

4. Conclusion

4.1 Reasoning

We used the most advanced techniques from the front-end and the back-end as we were not only to write an application but also it was a research-based project. The idea behind

this app that is we did not find any other app that has the same functionalities as ours. Our app could track your own steps and other steps as well. Moreover, this app applies to a game like competition based on steps tracked. One of the goals of the app is that the newer technology it includes, the more precision and interesting it will be. In other words, it would advertise itself by its own interests to gain more users.

To the Front-end: we use Model View ViewModel to design and show different layers for better visualization and personalization. The Back-end: we use Lambda(refer above) to access the database so that our app becomes more developed and sustainable in the future.

4.2 Lessons Learned

Challenge ourselves and edges cutting are the best practices for a software developer. Through this project, we were scrum masters each week and work together many times a week to learn each other's skills. The project overall provided me a pre-working environment.

Not only we learned edges cutting, debugging and especially the new skills when we explored new knowledge and applied to this project, but also we applied all received from this course to our app - Mosso.

4.3 Future Work

Next steps, we want to implement and complete a few of the functionalities. For example, put more functionalities on team fragment and optimize the UI design. Since we expect our app to represent itself by its usability rather than spending time to express what our app is good for.

We will publish this app in the Play Store and Apple store in the coming winter break. In addition, we expect to use our app on I-watch.

4.4 Other Misc

I strongly suggest this course to future students because I realize that this course not only for an Android application but also strengthen my abilities to be a software developer in the future.

5. References

Google Fit API
https://developers.google.com/fit

Fit chat view from Github
https://github.com/txusballesteros/fit-chart

Firebase
https://firebase.google.com/

AWS Lambda
https://aws.amazon.com/lambda/

Redis
https://redis.io/

Material Design (inspired us the most)
https://material.io/design/