Hiking/Fitness/Running App

Anusha Bhattacharyya and Callum Bruton

For our mobile programming final project, we decided to create a running application. The original idea was to create an app that would show trails local to the user general location but issues with finding an API for hiking trails forced us to have a change in direction. The current application will create a local profile and track that person's runs. The data created will track the time spent running, distance, calories burned, and a picture of the route they ran. All this data is saved into a local database so the user can view their progress overtime.

The methodology behind making application was to have a simple local based application that was user intuitive and lightweight. We opted to keep everything locally on the device so the issue of not having networking would not impact the basic functionality. Having a key focus on having everything being local allowed us to quickly prototype and test our changes.

The outcomes we are expecting is firstly having a working application that can consistently record data about the uses runs. Having the basic layout and design of the application created in order to give a fair demonstration of how the application is expected to perform. Lastly, we hope to showcase the skills for mobile development we gained over the semester.

I. Introduction

A problem with most popular fitness applications are they are targeted to the enthusiast rather than the novice. Our goal is to create a run logging app that is easy to use while not giving the user information overload. A key concept we wanted to stay with was keeping the app as simple as possible. That is why during development we focused on building a strong foundation that can be extended overtime. We believe that with technology we can encourage people to live healthier lives by giving them that matches the actions they performed. Thus, the data being the real sense of encouragement to keep working out.

Our solution key goal is to provide an easy user experience. We wanted to make sure our app was not dependent on any online service. Our reason behind that is when someone is starting out trying to build something into their routine actions will be spontaneous. Thus, having issues with an online service might cause them to remove the application out of frustration. Other thing we did was only show the relevant information while the user is working out to avoid them being overloaded. This again was to help them focus on the task at and not worried about the overall status during that time. The results screen correctly communicates the status of the run they just completed. All the result information is then saved into a database so the user can review their previous rans any time. Lastly the focus on a single activity means the user won't get sidetracked by other features.

As stated above the app will take a user's name and weight. After that information is proved they will be sent to the main application screen. On that screen they will have the given option to start a new run, and view past run they have completed. If they select to run that activity screen features a map and timer showing the total time spent running along with the path one took. The post run screen which is saved to the database includes a lot more interesting stats such as distance, calories burned, and time spent working out.

II. Related Work

Similar projects to this one is found in apps that have integrations with smart watches. Such as the Garmin connect app, Fitbit app, and Apple's own fitness app built-into IOS. While Apple's version does not force the user to use one of their smartwatches the other two does make that a requirement.

Since I happen to be more familiar with Garmin's version, I will be focusing on that application more. Firstly, as stated above they require the use of one their smart watches to get any data. This blocks a user from spontaneously downloading the app without buying into the echo system. This adds an extra barrier of entry that prevents a user from building a fitness routine. Another issue is that home screen interface is busy with extra information making the user feel overwhelmed by the amount of different data points they are seeing at a given time.

Other apps in the same category also try to have their application follow Software as a Service (SaaS). Meaning that only the enthusiast would be more interesting in the additional premium features. The issue is that some that is a novice would feel pushed away from the in-app purchases that are constantly popping up on their screen.

Our application on the other hand is fully local the persons device. We hopped that keeping things simple will help someone push themselves to get into a better weekly routine. Plus making sure they user is overwhelmed with data when first opening the app gives a sense of calm and peacefulness.

III. Application Design

This app followed a MVVM design ideology. Despite having to take a different approach with the project than we originally had, the Model aspect of the design remained mostly the same as planned. For the Model aspect of the design, we mainly implemented the database components for the Runs that we were storing in the application. In the model aspect, there was a Room Class where we defined the variables we collect during the user's run like the run's bitmap (this is the user's run route image), timestamp, average speed, distance, time, calories. We also have a RunDAO and the RunningDatabase where we respectivly use to interact with the data_table created in the Room class and then convert the data_table to a database. After this, we moved on to the View aspect of the Application. For

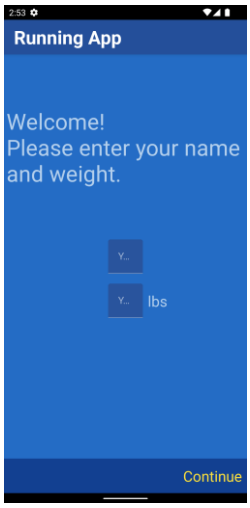
this, we defined the Fragment classes (Run, Settings, Setup, and Tracking) with each Fragment defining a different screen that is displayed. The two main fragments were the RunFragment and the TrackingFragment where in each fragment respectively, we get location permissions from the user and then connect to our Google Maps API which we then used collected updated user location points to draw connecting lines determining the user's path and drew on the map indicated by a red line. After the implementation of the Model and View aspect of the code, we then focused on the View Model implementation. This included classes Main Repositories and TrackingService where repositories and user interactions are defined and then MainViewModel and StatsViewModel which act as connectors between the Fragment classes (View) and the Database classes (Model).

When working on this app, the Architecture of the project was heavily influenced by a provided GitHub by Phillip Lackener[2] with the application layout. Along with this, his video series[1] on his application was very helpful with the learning and implementation of the Model and View classes in our app. We also used the online guides provided by the android developer site in order to learn how to work with Dagger-Hilt Injections which helped tremendously with the MVVM format[4] that we used and also drawing the path on the Map using Polylines[3].

IV. Results

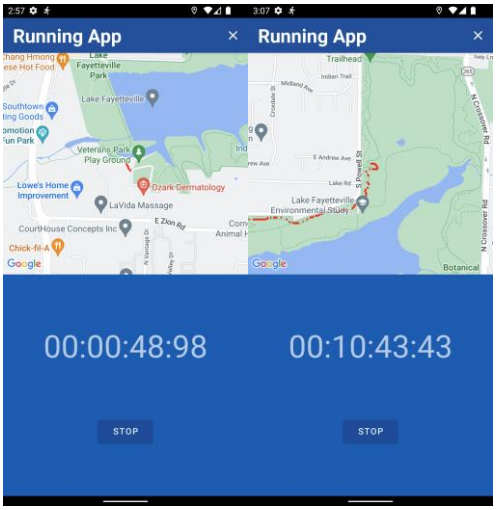
For the final project we needed to make massive changes to the project the application preforms within expectations. The app can take in a user's name and weight. Allow them to start a workout and record where the user went during their run the amount of time it took to complete. Lastly once the run was finishing all stats are shown for that run in the past runs list.

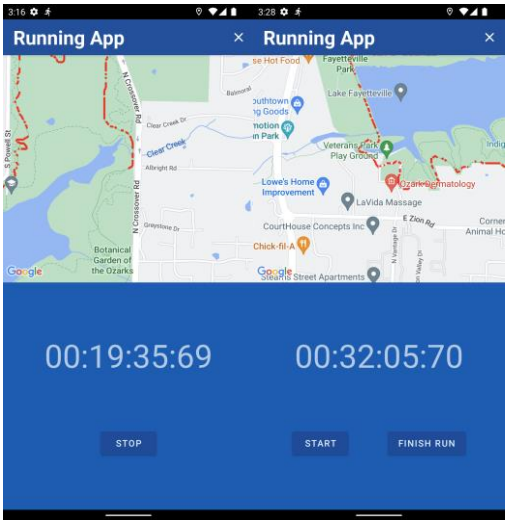
Let's start by moving though a common workflow a use might have when they open the application. The first screen has you entering your name and weight. Clicking continue will send you to the home screen.



The main screen will display your past runs. Clicking the yellow plus floating action button will take you to the new run recorder.

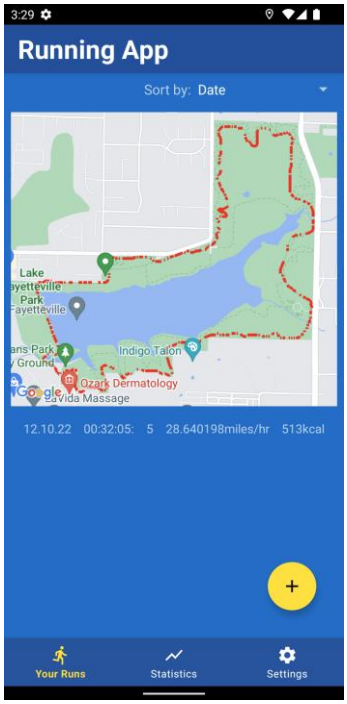
The run recorder will open to start a new run or to finish a run screen. Clicking the x on the top right till cancel the activity. The flowing set of shows what will it look during a run.



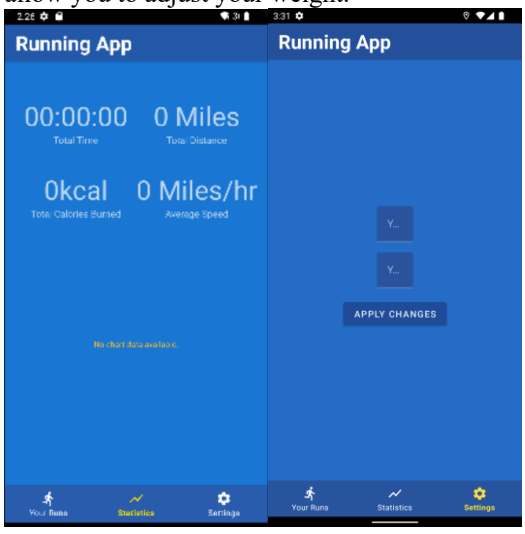


As you can during a run activity your route is being recorded to a red line on the map. While you're running a timer will

tick up to show how long one has been working out for. Once your run is over you select the stop button and brings you back to the start run screen where you can select to have the run by clicking finish run. After clicking finish run you will be taken back to the main screen where you will see the latest run added to the recycler view.



Under the photo of the map of the trail you just ran you will see the date, time, distance, speed, and calories burned. The other tabs shown give your life-time figure since using the app by clicking the 'Statistics' tab. While the setting tab just only allow you to adjust your weight.



V. Future Work

Ways that we can expand the app are firstly adding online storage or a way to export data so the user can switch between devices easier. We could add additional activities in a sub menu under the floating action button. An example activity would be biking since we would have to use a different formula to calculate the calorie burn. We also could integrate a form of social media sharing so people could share their favorite runs to their friends and family. We could also create an achievements tab that would mark key milestones. This would have the user checking back to see how close they are. Thus, allowing use to still encourage the healthy life due to the data while the user has all the control. The idea us statistics page to be linked to the achievements page just so milestones are acknowledged the uses accomplishments of using the app over time.

Sections where the app fell short include user inclusivity, proper stats implementation, our inability to add a user points system, and our shaky implementation of user location tracking on the Google Maps API. While a working database has been completely implemented to the application, we were able to integrate a user input system to get accurate calculated data for the user. On the setup page, despite asking the user for a name and weight input, this isn't taken in account when determining the user's run stats. One good example of this is the caloriesBurned variable which only assumes that the user is 120 lbs. We also wanted to implement a statistics aspect to the app. In fact we went to the extent of adding the classes where we could take the sum of the data stored in the database and then display it in the Statistics section of the App. Unfortunately, we weren't able to get to that point. We also wanted to integrate a user points system which was based on the distance the user walked. This would have been simple to integrate since it would only take the distance variable into account. The reason we didn't immediately integrate this aspect is due to us wanting to add a method to check whether the user was cheating the system by using a car rather than walking or biking a trail. This would be similar to the popular game, PokemonGo. In order to draw the path at which the user took, we used Polylines to connect the longitude and latitudes at which the user was located at. This method turned out to be incredibly glitchy which can be seen when running a mock route in the emulator.

References

[1] Lackener, Phillip. "MVVM Running Tracker App (Dagger-Hilt) Playlist."*YouTube*,YouTube,youtube.com/playlist?list=PLQkwcJG4YTCQ6emtoqSZS2FVwZR9FT3BV.

[2] Philipplackner. "Philipplackner/RUNNINGAPPYT." *GitHub*, github.com/philipplackner/RunningAppYT.

[3] Googlemaps. "Android-Samples/Polylinedemoactivity.kt at Main · Googlemaps/Android-Samples." *GitHub*, https://github.com/googlemaps/android-samples/blob/main/ApiDemos/kotlin/app/src/gms/java/com/example/kotlindemos/polyline/PolylineDemoActivity.kt.

[4] "Dependency Injection with Hilt : Android Developers." *Android Developers*,https://developer.android.com/training/dependency-injection/hilt-android.