5k Fitness

CIS 357 Fall 2019

Jesse David & Julia Hildebrand

# **Overview**

For this semester project, we chose to use AndroidStudio as our IDE for our mobile application. For our platform focus area, we decided to implement the Google Fit API. Based off of the information we could use from the API, we chose to create a running trainer application. The goal of this app is to connect your Google Fit information to the app and create a workout plan. To start using the application, the user is first prompted to sign in and enable their Google Fit information. They will then be directed to a page displaying today’s workout goal along with today’s workout history and the percent of today’s goal that has been completed. A menu will take the user to a future workouts page where they can view upcoming run distances for the next 10 days.

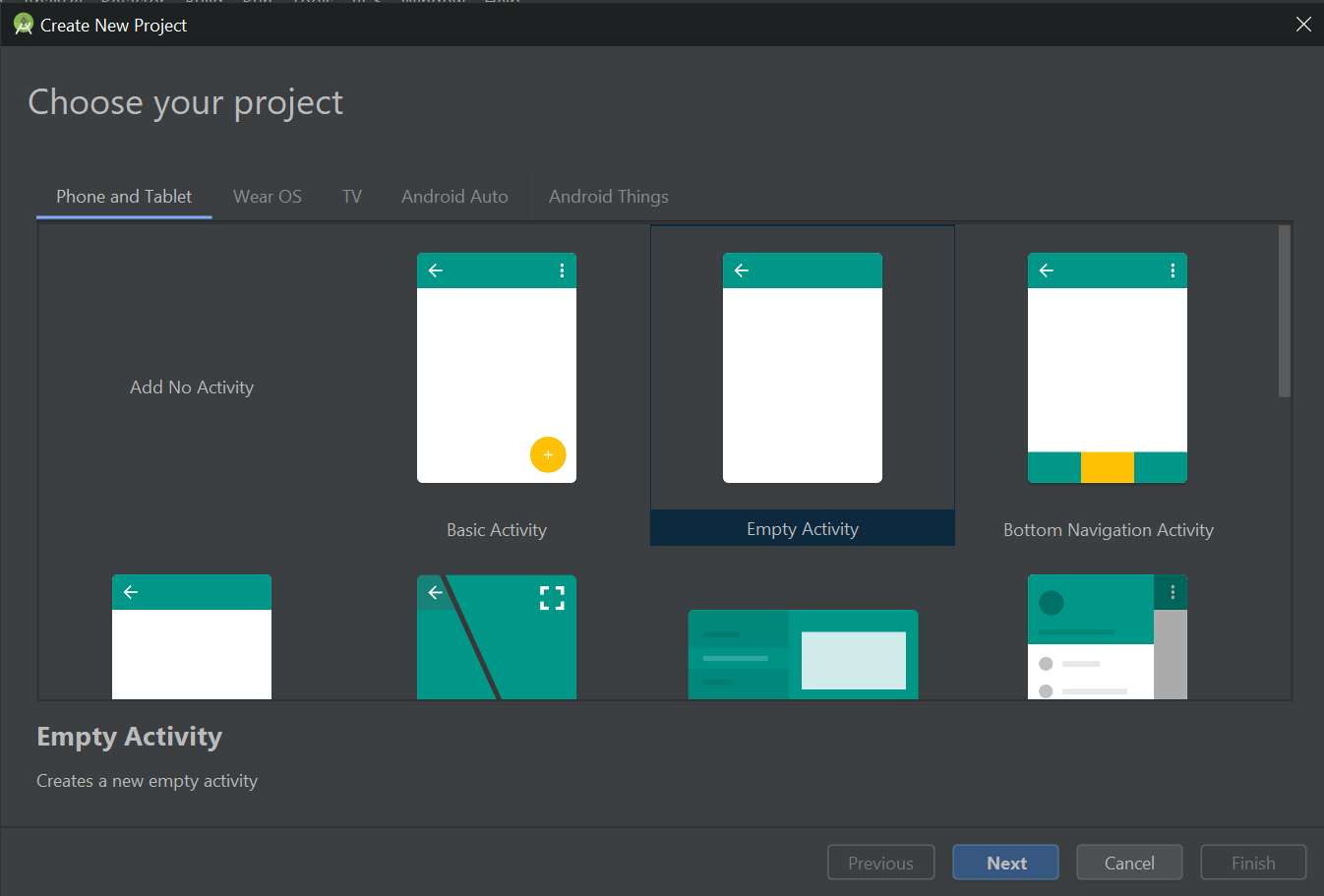
**Getting Started**

Before creating this application, note that you must download the Google Fit app and create an account. When developing this app, also note that you cannot use any emulator, and you must use your phone connected to the device running AndroidStudio.

To begin, first download AndroidStudio. This can be found here.

<https://developer.android.com/studio?gclid=CjwKCAiArJjvBRACEiwA-Wiqq3unHU7KIC5vnoD1OqLyjWoxPoxbl53-Rlq4CSZo5oNtgPoOBYnaVBoC-ioQAvD_BwE>

Open the application and create a new project with an empty activity.



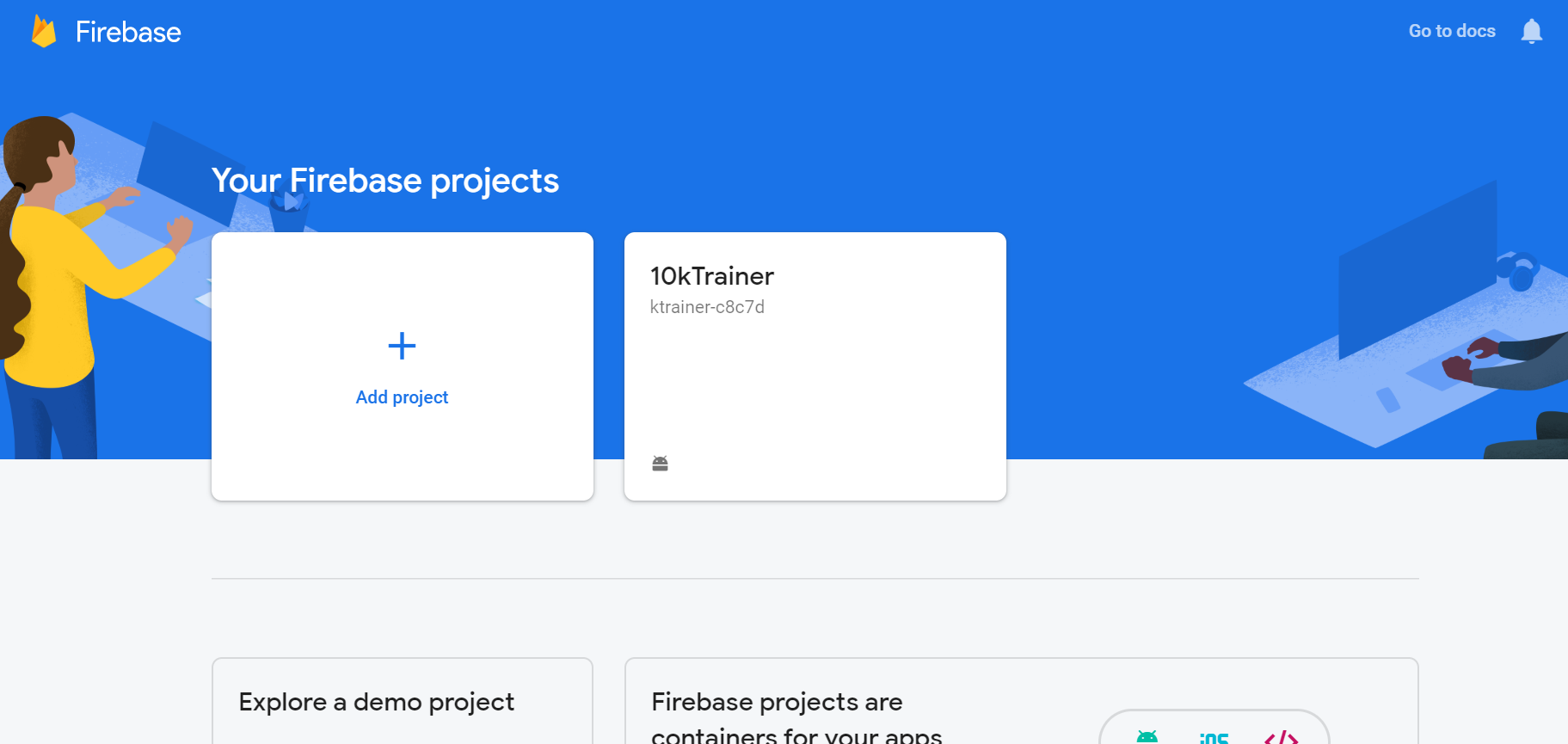
At this point, you can choose to hook up your GitHub account. Here is a link to do this.

<https://www.youtube.com/watch?v=-dAr6VnmomM>

Now, you will want to create a Firebase account with your Google credentials. Here is the link to this website.

<https://firebase.google.com/>

Once you do this, click add project.



You will follow the set up directions for this project that Firebase provides. This will include adding code snippets into several files, downloading and including a json file into your project, and running your application to verify it with Firebase. You will need to change the min SDK version to 16. The default is 15 which is no longer supported. After this, go into Firebase, develop, database, rules, and change the code to the following.

{

/\* Visit https://firebase.google.com/docs/database/security to learn more about security rules. \*/

"rules": {

".read": true,

".write": true

}

}

**Step-by-Step**

**Main Activity**

Let’s start by working on the main activity. We will use just one xml file for this activity because we do not need a tab bar. Delete your activity\_main.xml. In your content\_main.xml, remove the link to activity\_main.xml. Change any other instances of activity\_main.xml to content\_main.xml. In the content\_main.xml, we will create a TextView for the title of the app. This can be whatever font, color, size, etc that you want it to be. We will make that centered at the top. Here is the code for our title.

<TextView

android:id="@+id/appTitle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:fontFamily="@font/audiowide"

android:text="@string/title"

android:textColor="#FF9800"

android:textSize="35sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.025" />

<TextView

android:id="@+id/appTitleColor"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:fontFamily="@font/audiowide"

android:text="@string/title"

android:textColor="#000000"

android:textSize="35sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintHorizontal\_bias="0.480"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.022" />

Here is what that title looks like.



**Google Fit API**

We will now be connecting our app to the Google Fit API. To use and gain an understanding of Google Fit go to this link and read the documentation.

<https://developers.google.com/fit/overview>

At the very bottom of that page you will find another link detailing how to access firebase in android. Here it is in case you can’t find it: <https://developers.google.com/fit/android>.

Google Fit has six sub API’s that you can work with, for this tutorial we will specifically be using the [Sessions API](https://developers.google.com/fit/android/using-sessions) and the [History API](https://developers.google.com/fit/android/history). In order to use Google Fits api’s we first need to get the SHA-1 fingerprint for our certificate, Google provides great [documentation](https://developers.google.com/fit/android/get-api-key) on how to do this. On that same page, it walks you through; setting up a project in the Google API Console, enabling Fitness API, and using your SHA-1 fingerprint to create new credentials for your app. This process generates an OAuth 2.0 Client ID for your project.

After this, implement two dependencies in your build.grade: module.

dependencies {

implementation 'com.google.android.gms:play-services-fitness:18.0.0'

implementation 'com.google.android.gms:play-services-auth:17.0.0'

}

Next, we will be providing our app with the proper authorization for accessing the Fit API. Start by creating the method *requestOAuthPermission()* this will launch the Google Sign in activity to request OAuth permission for the user.

private void requestOAuthPermission() {

FitnessOptions fitnessOptions = getFitnessSignInOptions();

GoogleSignIn.requestPermissions(

this,

REQUEST\_OAUTH\_REQUEST\_CODE,

GoogleSignIn.getLastSignedInAccount(this),

fitnessOptions);

}

Throughout the process of implementing this code you are going to run into several errors due to missing dependencies. If you are using Android Studio you can click the red text and then use alt+enter to import those dependencies. After requestOAuthPermission, we are going to create *getFitnessSignInOptions()* this is used to check or request OAuth permission for the User.

private FitnessOptions getFitnessSignInOptions()

return FitnessOptions.builder().addDataType(DataType.TYPE\_ACTIVITY\_SEGMENT,

FitnessOptions.ACCESS\_READ).addDataType(DataType.TYPE\_DISTANCE\_CUMULATIVE

,FitnessOptions.ACCESS\_READ).addDataType(DataType

.AGGREGATE\_DISTANCE\_DELTA, FitnessOptions.ACCESS\_READ).addDataType(

DataType.TYPE\_STEP\_COUNT\_CUMULATIVE, FitnessOptions.ACCESS\_READ)

.addDataType(DataType.TYPE\_CALORIES\_EXPENDED, FitnessOptions.ACCESS\_READ)

.build();

}

These data requests are then used to request permissions from the user to access Google Fit data.

Going back to the onCreate method, we then call the *verifySession()* method within a try catch statement. Verify session is where the bulk of the Google Fit code comes in to play.



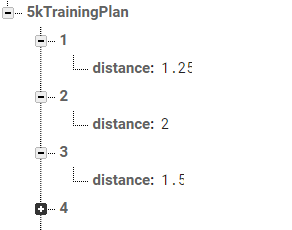
In verify session we then create a session read request using Google Fit Sessions API.

|  |  |
| --- | --- |
|  | private SessionReadRequest readFitnessSession() { |
|  | // [START build\_read\_session\_request] |
|  | // Set a start and end time for our query, using a start time of 1 week before this moment. |
|  | Calendar cal = Calendar.getInstance(); |
|  | Date now = new Date(); |
|  | cal.setTime(now); |
|  | long endTime = cal.getTimeInMillis(); |
|  | cal.add(Calendar.DAY\_OF\_YEAR, -1); |
|  | long startTime = cal.getTimeInMillis(); |
|  |  |
|  | // Build a session read request |
|  | SessionReadRequest readRequest = new SessionReadRequest.Builder() |
|  | .setTimeInterval(startTime, endTime, TimeUnit.MILLISECONDS) |
|  | .read(DataType.TYPE\_DISTANCE\_CUMULATIVE) |
|  | .read(DataType.AGGREGATE\_DISTANCE\_DELTA) |
|  | .read(DataType.TYPE\_STEP\_COUNT\_CUMULATIVE) |
|  | .read(DataType.TYPE\_CALORIES\_EXPENDED) |
|  | .readSessionsFromAllApps() |
|  | .enableServerQueries() |
|  | .build(); |
|  | // [END build\_read\_session\_request] |
|  |  |
|  | return readRequest; |
|  | } |

We then loop through theses sessions to get the dataSets associated with them. Through this, we are then able to extract our requested data from Google Fit and display it in our app however we see appropriate. For more best practice examples of connecting to Google Fit, take a look at this Github account: <https://github.com/android/fit-samples>.

**Firebase Workouts**

Now that we have connected our app to the Google Fit API, we need to give our users a goal to meet. By creating a table in Firebase Real Time Database we are able to query specific days of the month to then request the distance to be run that day.



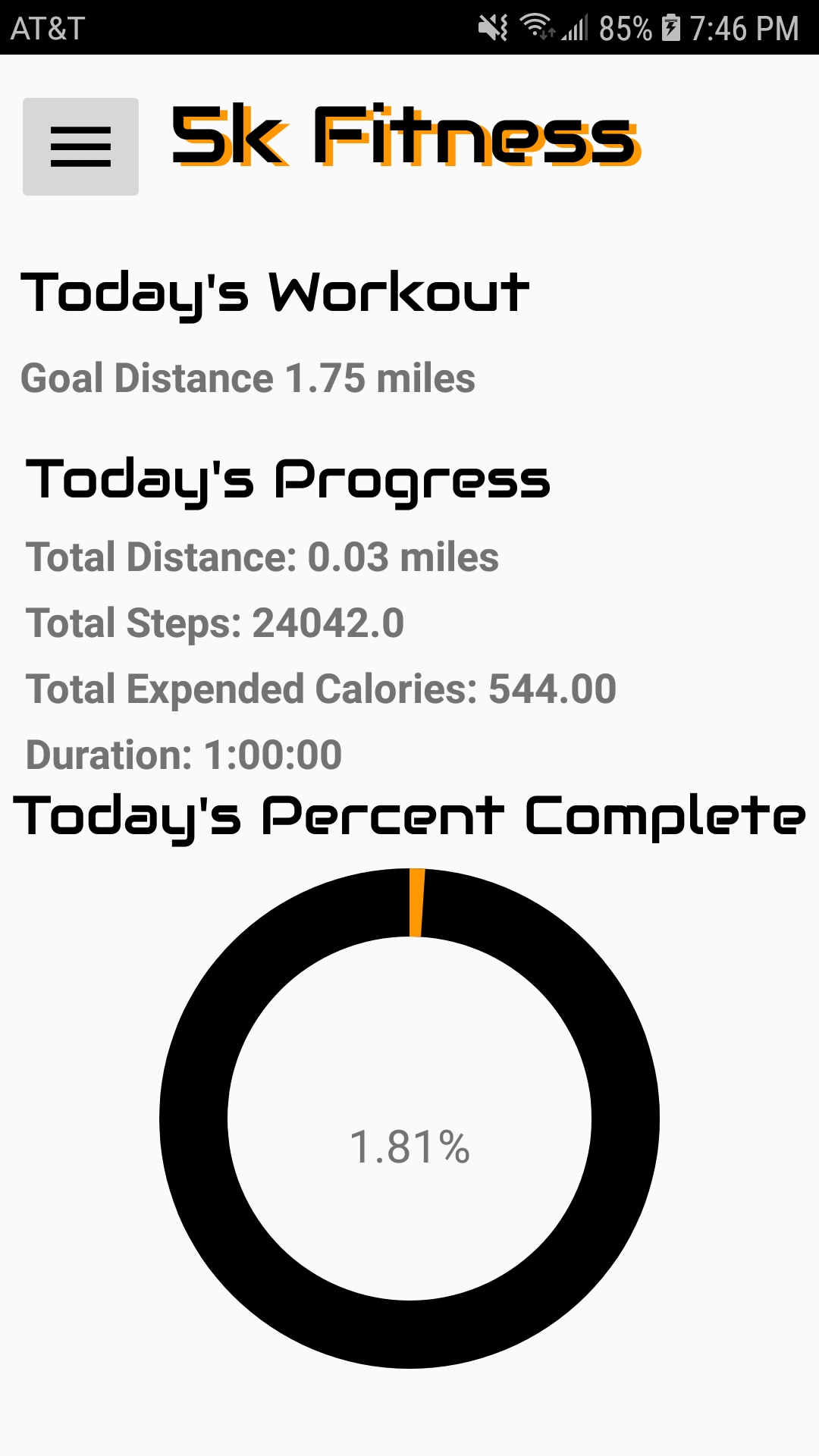
**Displaying Data**

We will now create TextViews within the content\_main.xml as headers for today’s planned workout, today’s progress, and percent complete as well as TextViews for the fitness data we want to display.

For the percent complete, we will use a progress bar. Go ahead and follow the tutorial below. We will be changing the number that is used in the middle to be our percent of progress complete.

<https://medium.com/@evanca/android-tutorial-for-beginners-create-a-pie-chart-with-xml-36e67dabe67f>

Here is an example of a completed progress bar.



Here is the code to set the values of the progress bar.

double percentCom = 100 \* (*metersToMiles*(dist) / goalDistance);

TextView textPercent = findViewById(R.id.*percentComplete*);

String text = decimalFormat.format(percentCom) + "%";

textPercent.setText(text);

ProgressBar progressBar = findViewById(R.id.*stats\_progressbar*);

int progress = (int) percentCom;

progressBar.setProgress(progress);

You can find all other methods to set the data in the MainActivity.java class in the GitHub repository as we are basically done with the majority of this class!

**Future Workouts Activity**

Now that we have completed the bulk of the application, we decided to create a new activity to display future workouts for the next few days. We will start by adding a pop up menu. Here is a link to be able to do this.

<https://www.youtube.com/watch?v=s1fW7CpiB9c>

We will make the menu display a page that contains future workouts for the next 10 days. Create TextViews for a title and the data. Add another Java class that will contain the methods for setting the data to these TextViews. Within the MainActivity.java, add the following method to transition us from the main activity to the future workout activity when we select it from the menu.

@Override

public boolean onMenuItemClick(MenuItem menuItem) {

if (menuItem.getItemId() == R.id.*futureWorkouts*) {

Intent intent = new Intent(this, FutureWorkoutsActivity.class);

startActivity(intent);

return true;

}

return false;

}

Make sure to implement PopupMenu.OnMenuItemClickListener in the main activity and add the proper import.

# **Final Code**

Here is a link to our GitHub repository with our finished code.

<https://github.com/davidjes97/5kFitness>

**Conclusion**

Now that you have completed this tutorial, you should have a fitness application to track your runs and give you goals. An alternative approach to give goals to the users would be to replace the use of Firebase with an Enum that has goal data.

**Stretch Goals**

Other ideas that we had for the application that we wanted to include were the following…

* Create an activity to display workout information for previous days and whether or not the goal of the day was reached.
* Create a settings activity, with a link in the menu, that allows users to disconnect their Google Fit information.