OneFitness

Muhammad Faizan Alam

OneFitness

By Muhammad Faizan Alam

Contents

Analysis

Identification of the Problem

Problem Description

Many people in today's world have started to become unhealthy due to an increase in desk jobs as well as eating out much more. Obesity is the second largest cause of cancer. Many governments are aware of this and have encouraged P.E in schools and a sugar tax on some items. There have also been many ad campaigns to promote fitness for adults to stay healthy. People however need a simple way to track all their fitness in one place easily without having to spend extra on new hardware and software. A simple fitness app that tracks the user's steps, calories burned, and their running/cycling sessions will allow many to improve their fitness and easily adjust their diet as they see how many calories they burned. People do not want multiple apps to track their different types of calories which is why I will be developing a simple app to track both steps and running/cycling sessions

I will make a mobile app that can be used to keep track of a user's steps. The user will set themselves a goal which they would like to reach daily. I will also work out the calories burned daily. The app will also allow a user to start a running or cycling session. These sessions will record the distance, time and calculate the pace/speed. After the session it will show the user their calories burned, distance, time and pace/speed. The app will have a map to show the route taken so the user can keep track of the different routes they have taken in the log. The app will keep track of the previous days/weeks/months for the user to check back on and see their progress. This will push people to improve and therefore work harder to keep their fitness level up.

This is suited for a mobile app because many people in today's world have a phone on them all day which they travel around with. The mobile can easily keep track of their steps without the user's input using its sensors. The program will also collect data for their running/cycling sessions through the mobile's GPS which can be used to calculate the distance and route travelled easily by API calls. The distance along with time can be used to calculate the speed and the calories burned. This will allow the user to easily keep track of their calories and steps which they can look back on to improve on in the future. This means the user will not have to manually calculate their calories for the distance travelled and their steps. The mobile app will give them a much more accurate estimate which they can use to improve their fitness and diet.

Muhammad Faizan Alam

Justification of Computational methods

Abstraction

Abstraction allows the program to be simplified so the programmer hides all, but the relevant data required to solve a certain problem which reduces the complexity.

The fitness app I will be making will allow the user to keep a track of their daily steps and their daily calories burned. The user will not have to worry about how the steps and calories are calculated. I will track the steps and not have to worry about how exactly it is done as a library will handle the reading the different sensors accurately for me which will allow my app to ignore details read from the sensor. I will ignore some features of the libraries I use because I will only be interested in some of the outputs from these libraries and APIs. I will also add running/cycling sessions to allow users to track their activities without having to use multiple apps. The distance, speed and route will only be tracked for running/cycling sessions so it can be omitted when the app is counting the steps and vice versa. Different libraries will handle these different sessions which means I will only have to worry about using the appropriate library. The program will calculate the calories using different algorithms depending on the user's activity because there will be different sensor data. I can isolate these algorithms and work on them separately, separating the program into multiple modules which can operate independently. This will allow me to take out certain parts of the program and not have to worry about them. This is suited for a mobile device which will be on the user all day and simplify their life.

Thinking Ahead

Thinking ahead allows us to plan which in turn allows us to be more efficient with our solutions such as re-using code and thinking about the inputs and outputs.

I will need inputs from the user for their weight, height and goal which will be used to calculate their calories and progress report. Most of the input will be from sensors which will be handled by libraries to calculate their steps or running/cycling sessions. The user will require outputs such as their daily calories burned as well as their steps. They will also have a progress report which will show previous activities as well as calories burned on those days.

I will also need libraries to read sensors such as a GPS and fitness library. This app will be developed for Android so it is likely I will be using the Google Fit API. I will have to plan how the API will be used to produce the required outputs for my app such as steps, speed and route travelled. Data has to also be tracked when the app is not running so the user can see their daily steps and calories. I can save the GPS and Google Fit services for future products that may require these modules which mean I don't have to re-write the same code multiple times.

Thinking Procedurally

Thinking procedurally allows a program to be written as a sequence of instructions executed in order which allow code to be repeated, broken down and called upon at certain times. Decomposition allows problems to be broken down into sub-programs which can easily be solved.

My program can be broken down into many sub-problems. A few of these sub-problems will be the log in, step-counter, user profile log and running/cycling sessions. The step-counter module of the program will not require the GPS module of the running/cycling sessions and can be broken down even more into counting steps and calculating the calories. The running/cycling sessions will not require the steps counter module which can be ignored. These smaller chunks for each sub-problem will allow me to work on each module separately for easier coding and future testing. I will need to consider events because my app will have a GUI. These events will include a progress report, a running/cycling session and settings. I will not know which one of these events the user will trigger in any order so I will have to activate their appropriate views when they are required.

Thinking Logically

Thinking logically allow algorithms to be written by making decisions, through iteration where code is repeated and branching points, so the correct output is generated from as little information as possible

I will have to think ahead and realise what parts of my code can be reused multiple times throughout the app. I can re-use the GPS service in both the running and cycling sessions of the app without having to re-write the same code to track the user. I can also reuse the map and tracking algorithms for both the cycling and running sessions as well as the same views. Services for the GPS and for the google fitness API will need to only be setup once and then reused multiple times to update the app frequently. My program will also have to branch to different modules as it is event driven and only certain parts of the app will be loaded when required. Decisions will also have to be made considering user data such as weight to calculate the appropriate data as.

Thinking Concurrently

Concurrency allows multiple things to be happening at the same time which mean multiple threads have to be used to execute the difference instructions. Concurrency allows the program to be more responsive and run algorithms in the background while the program is responsive to user input.

For my program I will require a service to always be active because it will need to keep track of the user for calculating an accurate step count. This service will never end and will always be on in the background even if other apps are running. If the fitness app is running this service will continue to have to run concurrently with the other modules of the program. The running/cycling sessions will also have to run a GPS service concurrently to keep an accurate location of the user as well as a google maps service to update their location. Many of the calculations will have to be done concurrently as more data is collected. The app will have to stay responsive to user input so the GUI will run concurrently with any background processes.

Stakeholders

The users for my fitness app will have a mobile device. These users include those that struggle with computers and others that are very familiar. Different users will use different features of the app more frequently than others. The ages can also vary quite a lot. For these reasons my app will have quite a variety of stakeholders.

A stakeholder for my app includes most people who are into their fitness and to improve their health. Some of my user's will be well versed with phones and into fitness. This demographic will mostly use the app to track their running/cycling sessions which they would like to improve on. The stakeholders for this demographic will be **Abytom** and **Harrison** who are both into fitness, young and well-versed with phones. They will likely use my app differently to an older demographic and will allow different features to be tested thoroughly. These two will use my app to track their calories burned from both step counter and exercises. Harrison bikes to and from college so he will thoroughly test out the tracking system and its precision. Abytom will use it to improve his running pace and distance.

Another demographic I will be targeting are those less familiar with phones. This demographic will use the app more to track their steps and calories as this demographic want to stay healthy by controlling their calories and are less into improving their times. The running/cycling sessions will likely be used less often. For this I will have **Jamshed** who is older and not as familiar with phones and fitness. He will likely use the app less often to check on his steps and calories throughout the week. The app will get data from the background which means he does not need to have it active all the time, making it suitable for him. He will rely on the logging feature less as he is not interested with exercising more. They will need this app to see if they are in a calorie surplus or deficit at the end of the day.

Research the Problem

Stakeholders

I will conduct a questionnaire because this app will be made for everyone and not just one person so I need quick thorough input from my stakeholders as to what they would like to see in the app. A questionnaire will allow me to gather a lot of information quickly and will be concise, allowing me to gather relevant data only. This will in turn allow me to process the information more easily and see what the most desired features of the app are, allowing me to prioritise certain features over others. However, a questionnaire will limit the choices to the stakeholders so I lose a personal approach that an interview could otherwise provide. The questionnaire will need to have well thought out questions and the information acquired needs to be precise and not too vague.

I will also interview Harrison. This will allow me to get some more personal feedback. Harrison will also be using the app more thoroughly and will explore every feature of the app, allowing me to get more accurate and personal feedback. This will allow me to find out what direction I should take with my app and I can ask more in-depth questions. Interviews however can be quite time consuming so that is why I will only be interviewing Harrison who will be using every part of my program. This will also allow me to ask any spontaneous questions that may arise throughout the development.

I gave them the questionnaire which they answered in front of me.

Questionnaire

1. How often do you exercise weekly?
    a. Rarely (once a month)
    b. Sometimes (once a week)
    c. Often (2 times a week)
    d. Most days (more than 3 times a week)
2. How do you currently use a fitness app to aid yourself? (multiple)
    a. N/A
    b. Track my calories intake
    c. Track my running
    d. Track my cycling
    e. Track my steps
3. What do you like about your current fitness app?
    a. N/A
    b. Intuitive
    c. Easy to use
    d. Precise tracking
4. What do you not like about your current fitness app?
    a. N/A
    b. Messy GUI
    c. Hard to use
    d. Tracking is wrong
5. What information is most important to you from a fitness app?
    a. Burned calories
    b. Steps
    c. Track running and cycling
    d. Logging (keep track of previous days)
6. What features would you like to be displayed on the main screen?
    a. Burned calories
    b. Steps
    c. Goals progress
    d. Distance travelled today
7. How would you like to control the app?
    a. Gestures
    b. Buttons
    c. Icons
8. Is there anything else you would like to add?

Question 1 is to find out their previous history with fitness. This information is required to see how the later questions will be answered differently by different types of people.

Question 2 will find out if they have previous experience with using a fitness app to aid their diet and fitness so I can find out which feature I need to prioritise the most

Question 3 will allow me to find out what features my stakeholders liked about previous solutions so I can prioritise them more

Question 4 will allow me to find out what features my stakeholders disliked about previous solutions so I can improve of disregard those features from my solution as to not clutter the app

Question 5 will allow me to record and display main information to the user more easily and visibly without having the user to dig in for the information they want.

Question 6 will show me what information is most vital to when the user opens the app so I can display that on important screens. This will allow the app to seem more intuitive and not cause the user to dig in for simple info.

Question 7 will allow me to find out how to layout the gestures for the app, so they are user friendly for my two demographics

Finally question 8 will allow me to take any extra input from the user that they may like to add so no interview has to be conducted saving time. I will use the answers to this question to aid the rest of my research so I can implement features I might not have thought of.

Muhammad Faizan Alam

Questionnaire Analysis

Questions summarised:

1. How often do you exercise weekly?
2. How do you currently use a fitness app to aid yourself? (multiple)
3. What do you like about your current fitness app?
4. What do you not like about your current fitness app?
5. What information is most important to you from a fitness app?
6. What features would you like to be displayed on the main screen?
7. How would you like to control the app?
8. Is there anything else you would like to add?

Answers:

Harrison:

1. most days
2. track cycling / track running / track my calories intake
3. precise tracking
4. messy gui
5. burned calories / track running and cycling
6. burned calories / steps / Goals Progress
7. gestures / icons
8. I would like all my required features to be in one app instead of multiple apps. I like the app you will be making and am interested in it

Abytom:

1. sometimes
2. track my running
3. easy to use
4. n/a
5. track running and cycling / logging
6. Goals Progress / distance travelled today
7. buttons / icons
8. Abytom wants a goal for steps and distance that he would like to reach daily

Jamshed:

1. rarely
2. track steps
3. easy to use
4. messy gui
5. steps / burned calories
6. steps / burned calories
7. icons
8. Show the steps and calories burned easily as well as reminders to reach my goal

Stakeholder requirements

My focus for the end-users are:

No.	Requirement	Justification
1.	Show daily steps, calories and distance on the main screen	So that the user can easily see their main stats
2.	Simple Intuitive Design	Keep the app simple to allow ease of use by using icons so they know the function of different buttons
3.	Navigation Menu	Allow the user to easily navigate the different views
4.	Track running and cycling sessions	Let the user start sessions to track info when they run and cycle
5.	Set daily goals	Let the user set their goals to reach daily
6.	Progress for daily goals	Progress for the daily goals should be shown on the main screen

With my questionnaire I have a clear understanding of what features my stakeholders find important and which ones must be prioritised. My stakeholders are quite varied as shown with question 1 which will allow me to aim the app at a large demographic. The main features are tracking calories, tracking running and steps and ease of use. My stakeholders all use a fitness app to track difference activities. This will allow me to develop an app that targets all these features right into one app.

I now have a good idea of what to prioritise for end users from this questionnaire for my stakeholders. However, I will need to carry out future questionaries' and an interview to get furthermore precise feedback.

Muhammad Faizan Alam

Existing Solutions

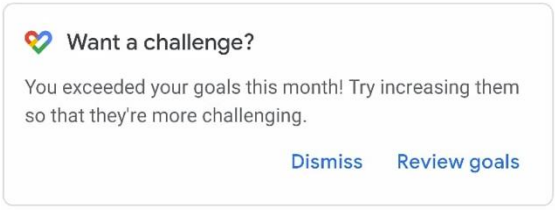
Google Fit

This is screenshots of Google Fit which uses the Google Fit API to keep track of the user's steps throughout the day. It keeps a log of this on the cloud which is connected to the users google account. The app allows the user to input their goal for the number of minutes they would like to move for, their gender, birthday, weight and height. Using this input alongside the number of steps tracked it also calculates the number of calories burned. The app keeps a log of previous activity which is viewable in charts.

I will also use the Google Fits API to implement the tracking like this app, allowing me to greatly simplify features

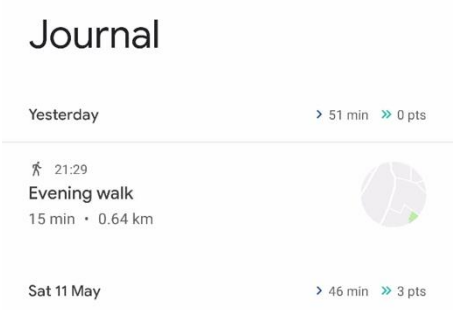
Google Fit's main screen has many components. The top has a points system for moving and a heart points system for fitness. I will not implement any similar systems because they are very confusing and do not have a set purpose. I will be aiming to keep my app simple and not cluttered for the older demographic so I will be ignoring these points systems.

However, it also calculates the steps, calories and distance. As requested by my stakeholders, these 3 features are very important and will be implemented. I will also display them on the top on the main screen for the user to easily see.

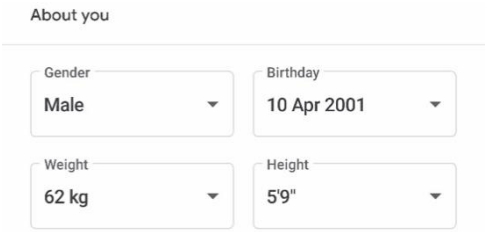


I will be implementing a similar feature to remind the user of their goals, however I will be implementation a progress bar that they can easily see how far away they are from their daily goal.

The bottom has a button with icons layout which is very simple and intuitive. This is one of the features my users liked about their current fitness apps so I will be implementing a similar design. This will also allow my app to fit in with the android echo system.



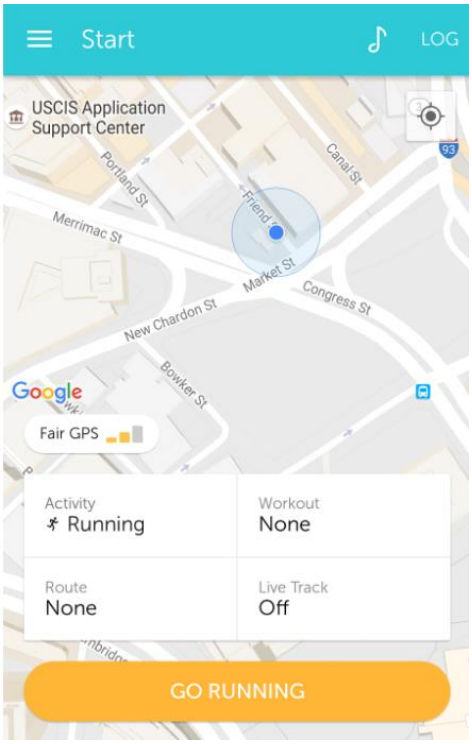
The app has a journal of previous days tracked. It shoes the time and distance. I will also put a log/ journal into my app for any previous work outs as well as steps, distance and time like this. This is a requested feature and will allow the user to improve on their goals by checking their log. This log will be able to be backed up too so it can be restored.



This app lets the user modify their details. I will also have a user's screen that will allow the user to change their details as well as set goals

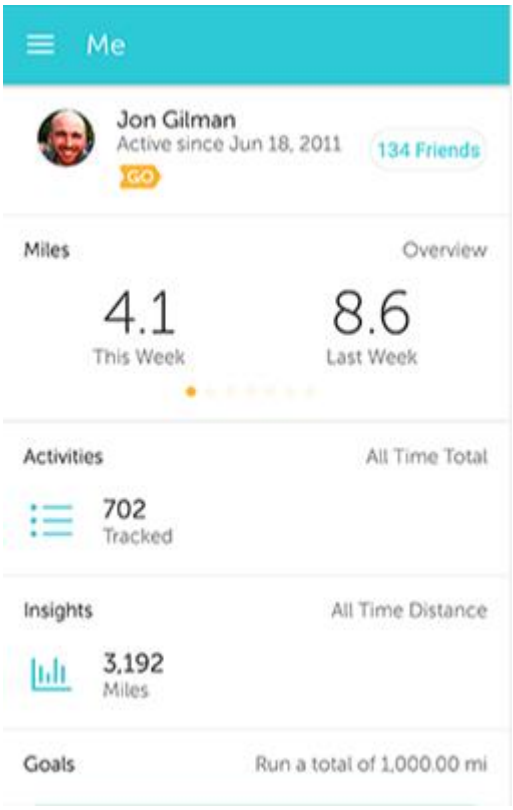
RunKeeper

RunKeeper is one of the most famous apps used by professional athletes to keep a track of their workout. The app is used mainly to keep track of the route the user took, their calories, distance and pace. Using this data, it also shows the history for the user to see their improvements. Like Google Fit it also has goals that can be set and broken.



The starting screen shows the GPS signal, a map in the background with the user's location. It then shows some options for the user to select from such as the activity, any saved routes and a preset for a workout. Finally, it has a large button to start the activity. I will have a large map view as well with the user's location being highlighted

I will be implementing the map with the user's location on the start screen like this for the running and cycling session. I will have chosen between the two like this. Finally, I will make the start button large and easy to hit for the user while moving.



The Me page is a review of what the user has achieved over the period of using the app. The miles compare the user's current weekly miles with their previous miles. It shows the activities for the user which can be expanded and allows the user to see how they tracked their miles.

I want to implement the expanded view of the activities. My logging page will be more like Google Fit than RunKeeper due to simplicity. I also will not be tracking more than cycling and running and do not have the time to implement tracking for all the other types of activities.

Essential Features

When the app is launched for the first time it will display a google log in page. This is essential to keep a record of the steps, calories and distance. The log in will be used alongside other apps to get the total steps, distance and calories saved. If this is the user's first time, they will be asked to give permissions to store the steps, calories and distance on their account. Sessions will also be able to be backed up to the phones storage so the user can transfer devices and also have backups in case anything wrong happens.

The app's main screen will display their total daily steps, burned calories as well as their goals progress. A progress bar will visualise the data so it can easily be interpreted by the user. The main screen will easily allow the user to start an activity for cycling and running. Finally, at the bottom there will be a menu to allow the user to easily navigate the app between the different screens. The reason I have chosen this is to keep the app simple for the older demographic as well as keep with the theme of mobile apps.

The cycling/running sessions will show a map with the route currently travelled by the user as well as their calories burned, speed/pace, distance and time. This data can easily be seen by the user when they are carrying out that activity, so they know how well they're currently performing. The map will be shown using Google Maps API and the relevant data will be recorded and updated regularly using GPS. This will allow the user to keep track of their route as well as an accurate speed and distance, allowing the user to improve their fitness more easily.

The log screen will show a scrollable calendar with the user's performed activities with basic distance and time for that session. This screen will allow the user to see their previous progress as well as expand a session individually to see more data such as their route and speed for activities. This will give the user an easy interface to see their progress so they can work on improving their session distance, calories and/or speed.

The profile screen will let the user see what account they have signed in with as well as set their goals. This screen will allow the user to change any personal details such height and weight. This is required so if any data becomes out of date such as weight the user can update it, so the calculations are more correct for the user. This screen will also have the goal settings so the user can modify any goals such as if they want to push themselves more in terms of daily steps and calories.

Limitations

Firstly, my application will be very complex as I will have to follow many rules of android and learn many libraries to achieve my results which will take time. I don't have much experience with Android and have no experience with the libraries required for many tasks this app will have to carry out. This means I have to go research and learn to use these libraries and techniques to achieve my desired behaviour. The libraries I will be using is mainly Google Fit as well as Google Maps API. I may have to use other libraries to provide a better UI for my design. I will also have to use SQL in the app to manage saved sessions. These libraries are a must because I do not have the time to be reading raw sensor data and working directly with files which can also be easily corrupted by the user. Reading and interpreting raw data is well above my knowledge scope and has been perfected over decades by experienced corporations so I will use their libraries. The libraries also simply many complicated equations for me and will save a lot of time. However, the libraries will have to be used within their limits meaning I may not be able to access some required information as easily. I also have limited experience with Java and Android Studio so I will have to get used to the syntax as well as learn to use a new IDE to debug my code. Android has many limitations due to some of the low powered devices it operates on so I will have to work with these limitations to achieve a solution. Overall, I will have to reach my deadline while working with these various limitations. It will help to separate different parts of the app form each other so the app can be developed in modules which will allow me to focus on different libraries.

Due to time constraints I will have to focus on implementing the important parts of the app first before adding other details. I will also have to focus my time on the analysis, design, development, testing and evaluation for my app which will further limit my time. I will have to use time daily to work on the write up which will limit my time on the development phase meaning I have to keep the app limited and cannot add any redundant features. I also don't have anyone to ask for help with particular issues which means I have to spend time online both researching and finding solutions for any problems I will encounter which will further limit my time to work on the development phase. I will not be able to implement features to record what food a user eats to keep track of their calories as this will be very time consuming. I won't have time to implement other exercises other than running and cycling because I do not have time to be writing custom algorithms for other them.

Android is very locked down and updates cannot be retrieved too many times due to battery consumption, because of this hardware limitation I will have less accurate data to work with in the app making calculations a bit off. I will have to work with this and take it into account when making the app as to not get any absurd values. Android is also run on many different types of hardware. I will have to make sure the functions and versions used for libraries are compatible with my version of Android on my phone. This means I may not be able to use the latest features as my phone does not have any support for those features Finally I will have to make sure to ask for permissions from the user and Google to use certain features which can take time to implement further decreasing the time I have for other features.

Lastly my app will have to keep user data safe on their google account. The details of the user must follow the data protection act so there is no unauthorised access. I will use Google services for this, so it stays encrypted without going through my app. This data will only be retrieved using the user's google account which will prevent unauthorised access. Using a google account will not only be intuitive but will also allow me to use Google's robust security to secure any data that may be private such as where a user has travelled. I will also export the data to the user's storage which I will ask for permission, because this storage is in the user's control, I don't have to worry about encrypting it.

Proposed Solution

Hardware & Software Requirements

The following are my hardware and software requirements to run and develop my program

Requirement	Justification
Hardware	
Android Device	App will run only on an Android device – these are cheap and widespread currently
Storage – Max 100 MB	A max of 100 MB storage space will be required. The app will not have many assets which will mean it will be lightweight
RAM – 2GB+	At least 2 GB of RAM will be required to run both the app's service and Android in the background
Touchscreen	Touchscreen of the phone will allow the user to easily navigate the app without having to carry any large devices
GPS + Accelerometer Sensors	These sensors will be required to run the program as they will keep track of the user and their speed throughout the day and during activities. Data from these sensors will be used to work out different information for the user
Software	
OS – Android 6/7/8/9	I will be using version 6 as my minimum requirement which means 70% of android devices will have access to the program as well as allowing me to use up-to date features of android
Android Studio	Android Studio is the only IDE capable of developing applications for Android and is maintained by Google. It will allow me to easily create user interfaces that work well with touch

Success Criteria

The success criteria are what functions and how my app will function and interact with the user and why I have chosen to go that route.

No.	Requirement	Justification	How will it be measured
1.	Menu with icons on the bottom	Allow the user to easily navigate the app	Check if clear icons are shown at the bottom that can be clicked on
2.	Main Screen – show main details	The main feature of the app is to show the user their steps and calories as well as distance, so the user can improve their fitness	Allow the user to easily see their steps, calories and distance on the main screen
3.	Track steps/calories/distance when the app is active	Tracks the user's steps/calories/distance while they're using the app	Check the counters are updated by walking and restarting the app
4.	Track the steps/calories/distance when the app is inactive	The user's steps also need to be tracked when they're not using the app and their phone	Check the steps counter is getting updated while the user walks and the app is not active
5.	Show user location before session is started	Let the user see what location they are going to start their session from	Check to see a map with the user's location highlighted is shown on the session screen
6.	Track running and cycling sessions	Allow the user to track their route and speed/pace when they go cycling/running	Check if the user can start a session
7.	Countdown for sessions	Let the user get ready to run/cycle before the session starts recording	Check to see a countdown begins when the start button is pressed
8.	Track speed, time and calories in sessions	Track the speed, time and calories as they run/cycle	Check if this data is displayed while a session is active
9.	Track route as the user travels	Lets the user see where they have travelled on the map as they perform their session	Check to see if the track is drawn on the map when a session is started
10.	Show a report after each session	Allow the user to see their performance after a session so they can improve on their goals	Check if a report screen shows all the relevant data
11.	Track the route	Track the route for the running or cycling sessions	See if the correct route is displayed after a running or cycling session
12.	Journal should have a vertical scrollable calendar	Easily see previous days that are scrollable like a webpage	Check if previous sessions are shown
13.	Show expandable cycling and running sessions in the journal	Allow separate tracking for the sessions so users can see their previous work outs and push themselves	Check if there are clickable buttons that show a summary of the sessions
14.	Export and import session data	Let the user make backups of their data and restore it – allowing them to transfer data between devices	Check to see if export and import buttons work with sessions being restored

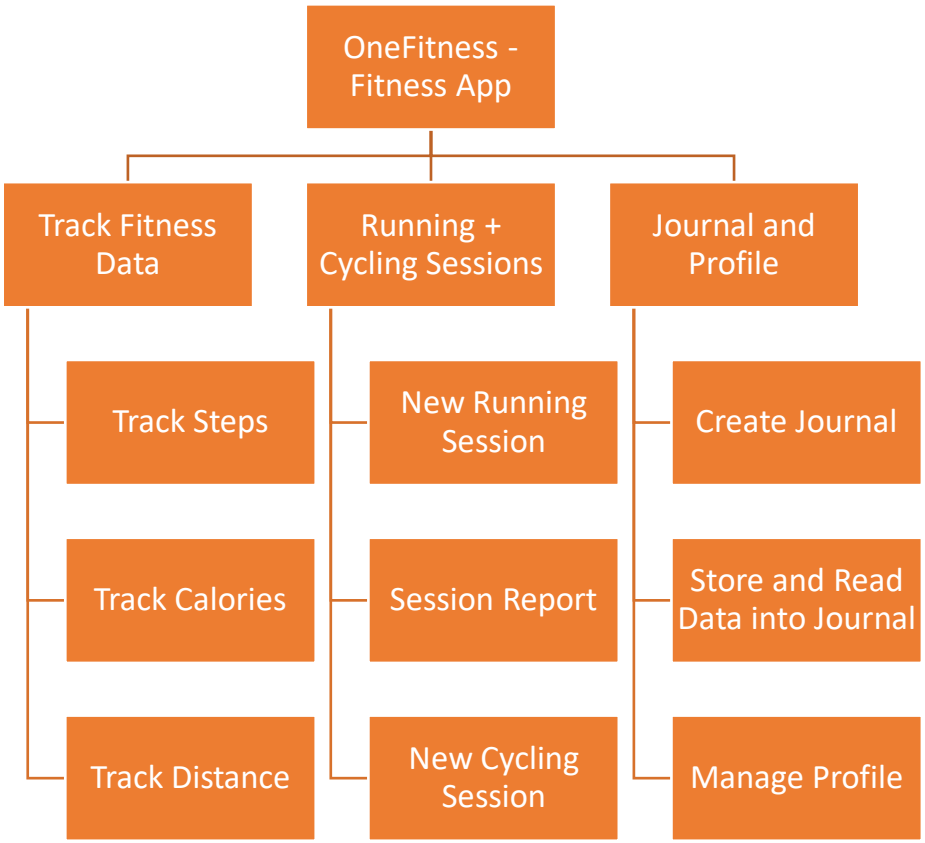
15.	Allow the user to easily change their height and weight	This will allow the user to update details about themselves	Check if changing the data is saved and reloaded and in range
16.	Set goals	Allows the user to set themselves goals in range so they can see how much is left	Check if goals are saved and shown after restart
17.	Show progress for goals	Allows the user to easily see their progress for the day	Check if progress is shown on the main screen
18.	Log in with google	Allow the user to easily sign in with google to keep track of their steps, calories and distance	Check if the user can sign in using their google account

Design

Decomposition

Decomposition allows problems to be broken down into sub-programs which are easier to understand, manage and program. The sub-programs can be solved individually making the tasks simpler. A 'Top-Down' design allows a problem to be split up into its different modules which can then be split up into sub-problems which are easier to solve. This repetitive process of decomposition is known as 'step-wise refinement' and will allow me to tackle one problem at a time making it easier to develop and improve on. It also allows my app to be modular and less dependent on other parts of the app, making it easier to tackle any problems.

Main Program



I have broken down the solution into 3 main parts: Tracking steps, distance and calories, running and cycling sessions, and finally the journal and profile.

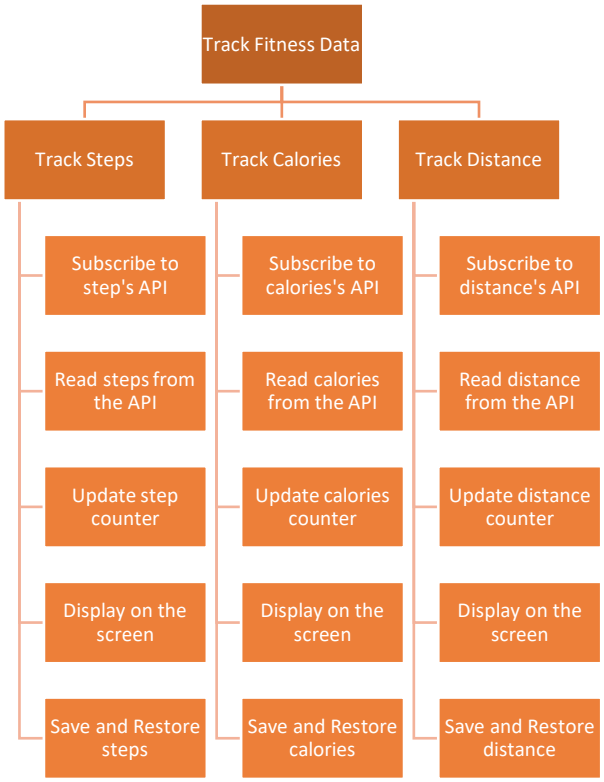
Tracking steps is the first main component and the main feature of my app.

The running and cycling session will add on to the app making it an all in one fitness app to track user's sessions

The journal will allow the user to see their previous data

Managing the profile will allow the user to change certain details and set goals as well.

Version 1 – Calculate and record main data



I will focus on the tracking steps and calories first as that's my app's main feature. This task is further broken down into 5 parts which are very similar but will have to be handled separately. I can use OOP to reduce code repetition.

Track Fitness Data - Iterative Plan

Track Steps:

Researching the solution, I have found out that I Andriod phones have Google services running in the background. I will have to use the API to subscribe and let the background service know I want the step data to be recorded. This will record all the steps of the user throughout the day as they move around. I will get this step counter, update, store and show it to the user. I will also restore the last saved counter when the app launches.

Track Calories:

I will also use the background service and subscribe to record the calories alongside steps. The same actions will be taken to record the calories which are updating, showing and saving the data. I will try to use OOP to increase code reusability as much as I can to allow easier modification in the future.

Track Distance:

The background service will also be used to track the user's distance they travelled throughout the day. Handling the distance for the day will be up to our app to reduce to the daily distance and not overall.

Version 2 – Record Cycling and Running Sessions

The running and cycling sessions will add on to the app, allowing the user to track more activities more precisely as the app is active.

Fitness Sessions - Iterative Plan

Button to begin a new running or cycling session as well as showing a report at the end

New Running Sessions:

Data for the running session will be tracked such as time, distance and calories.

Previous session's report

The report will show a map with the route tracked, the calories. distance, and time. It will show the pace for a running session and show speed for a cycling session

New Cycling Sessions:

Data for the cycling session will be tracked such as time, distance and calories as well as speed.

Track Route on map:

Tracking the route will be done on a map alongside the distance travelled so it can be saved

Track session data

The distance and time will be calculated which will be used to calculate the speed as well.

Version 3 – Journal and Profile

Journal and Profile - Iterative Plan

This version will focus on the user and their progress. It will also focus on setting their goal, weight and height.

Create Journal:

The journal will be a scrollable screen which will allow the user to see an overview of previous days as well as expand any running/cycling sessions

Store and Read Data Into Journal:

I will have to setup an SQL database to save data in the Session part and then load this data from this database into the journal, allowing the user to view old sessions.

Manage Profile:

The profile will allow the user to see their Google account name and profile picture. It will also allow them to set goals so they can improve their fitness and it will allow them to set their weight and height. It will also show the goals progress

Create Scrollable View:

Because views are static and normally not moveable, I have to figure out a way to make a scrollable view with all the sessions which can be scrolled through fast. This will provide an easy view to the user on mobile.

Show Summary of Session:

A summary has to be shown for when the user selects a session in the journal view. This will show more details such as the route, distance and calories to the user which is similar to the proposed solutions.

Show in Calendar View:

The sessions will be scrollable and shown in a view similar to a vertical calendar so the user can see exactly on which dates they ran/cycled

Export and Import Journal:

Allow the user to backup their database and let the import that database so they can transfer data between devices and have backups.

Save Sessions in SQL Database:

When sessions are completed, they have to be written into the SQL database so they can be loaded into the journal

Load Sessions in SQL Database:

Saved data from the database will be loaded into the journal view which will be scrollable.

Load name and profile picture:

Because the app works using the user's google profile, I will load their image and name in the profile view to show what account they are using.

Set Goals:

The user will have an option to set goals for their steps and calories.

Set weight and height:

The user should be able to set their new weight and height so they can see their progress as they workout.

Show Goals Progress

The user should easily be able to see their progress which will be shown on the main screen which should let them know how they are doing towards their goal.

Version 1: Track steps and calories

First version will track the user's steps in the background. It will also track the calories and distance in the background. When the app is launched it will show the user's daily steps, distance and calories on the main screen for easy viewing.

Version 2: Track running/cycling sessions

Allow running and cycling sessions to be started by the user in the app. It will track the route, distance and time for both sessions. Using this data, it will calculate the speed/pace for the session. After the session is finished it will show the route tracked on a map, calculate the calories and show the relevant data (distance, time, and speed/pace) as a summary. The pace and speed have to be calculated using the time and distance for the session. Finally, calories and distance will be accumulated with the daily steps data on the main screen.

Version 3: Journal and Profile

The last version will have a log to keep track of user's previous calories, distance and their sessions (running/cycling). The sessions will be able to be expanded so more details and the route can be seen. This screen will be scrollable so the user can see their sessions from previous days. These sessions can then be exported to the user's storage so they can create backups of it and then import these to restore their sessions. This will also allow the user to transfer data between apps. The profile will show the user their google account and set their goals, weight and height. The goals progress will also be shown on the main screen.

Solution Description

Usability Features

Version 1 – Track steps, calories and distance

The main screen will show 3 details that are most important to my user. I want to keep my app minimilistic therefore I will try keep the amount of detail on the screen minimal. The main screen will show the steps, calories and distance the user has travelled that day. These will be accompanied by icons as requested by the stakeholders to show a minimal but easy to read view.

> *Icon button to start running/cycling sessions*

This will allow the user to take a quick look at their current stats when lauching the app as well as start a session quickly without having too many screens.

Version 2 – Track running and cycling activities

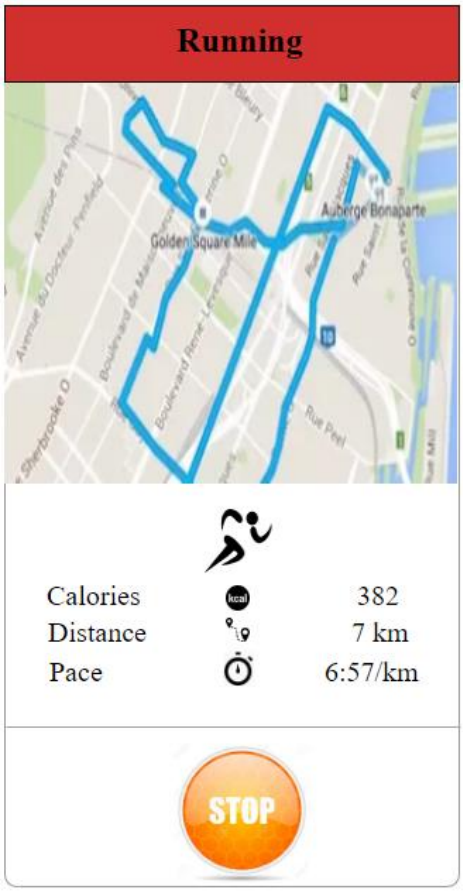
Show map with the user's location

The user's location will be shown in a map so they can see exactly where they are beginning the session and see their sourroundings.

Use icons to select type of activity

Large start button to easily begin activity

The activity/session window will allow the user to select between running and cycling to track an activity. The map will show them their current location highlighted. It will show a large start button to keep the app simple and easy to use because not all user's will want to worry about the details. I want to also make a counter when the start button is pressed so the user has time to get ready.

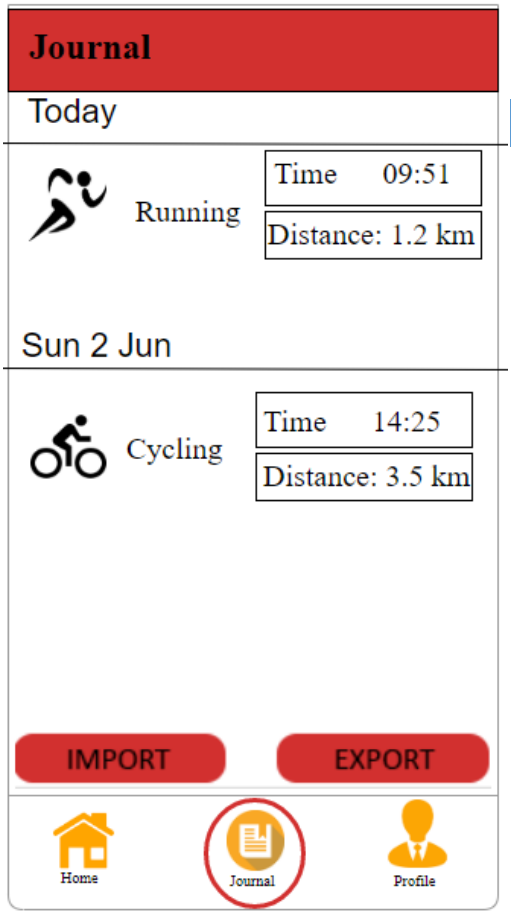


Map to track current location

When a running or cycling activity is started, a new activity will be launched; it will show a map that tracks the user's track and current position. The acitivity will also have calorie, distance and pace counters as well as a GPS signal tracker.

Details about session automatically being saved

Large stop button to end activity



Version 3 - Journal

Sorted using dates

The Journal screen will show the user their previous sessions. It will keep a log of relevant data to show the user a quick overview of their fitness. The app will keep the same layout and theme to allow easy navigation.

The running and cycling can be clicked on and expanded to allow the user to access more data such as the track, pace/speed and individual calorie burned.

Summary for each previous session underneath the date

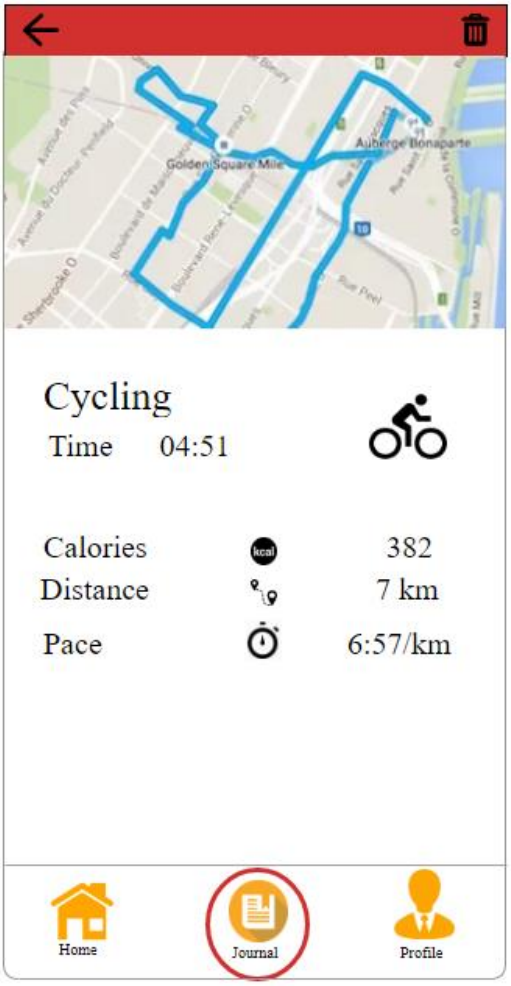
Clickable buttons to expand their session stats to view other details

Import and export buttons using same theme as app

Navigation Bar with icons to easily navigate the app

Some of my user's will not be well versed with phones therefore I have decided to stick with icons for my navigation bar as suggested by my stakeholders. The icons will also have labels to make it easier to distinguish what function it executes. This will allow the user to easily access the different screen of the app such as the log and settings and also keep with the theme of Android.

Import and export buttons will be distinguished with the app's theme. These will allow the user backup their sessions and import them in again.

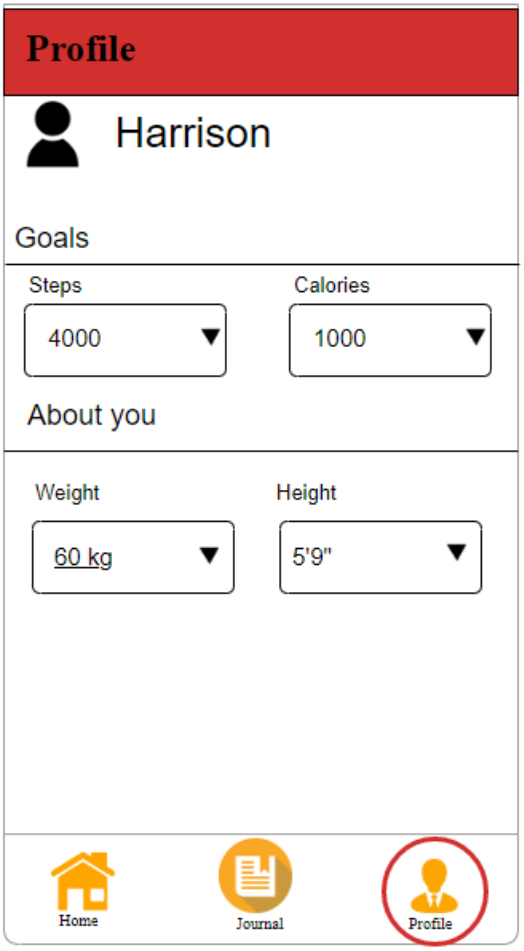


> *The map will show the route, with the details for the sessoins*

This screen will be used for both running and cycling sessions. Which will allow the user to easily see their route that they travelled as well as the relavant info. This screen will also allow the user to delete their activity from the journal.

Icons will be used throughout the sessions to keep with the theme of the app and make the app more accessible and easier to use.

A similar screen will be used for both walking and cycling to allow the user to see relavant data and delete it.

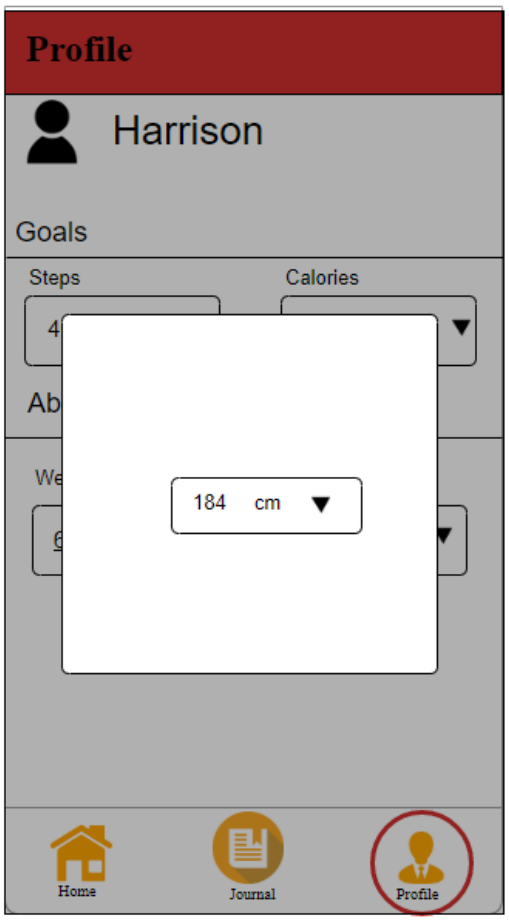


The profile section is the last main screen. This screen will allow the user to easily set their goals and change their personal details which are used to calculate calories. Using the same layout for all the options will allow the user to quickly change details while not having to learn anything new

Goals: Allows the user to set their desired daily goal for their steps and calories

Details of the user can be changed. These details are used to calculate the correct calories. This data can be changed such as if the weight or height changes.

Progress bars shown on the main screen



Drop downs will let the user select their weight and height so validation is limited. Drop downs will stay with the theme of the app as well as be more intuitive due to Android's design

Each drop down menu will open a dialog to change the details. For the height, weight and goals it will be a scrollable wheel which stays with the design of Android and is easy to use.

Stakeholder feedback on Design

As I now have a design for the app, I want to approach my stakeholders with it and get their feedback to anything they would like specifically and if it meets their requirements. I showed them how the mobile app would essentially work and how I have optimised the design to work well with touch.

Presented

I presented them printed copies of the design above and walked them through the app and how it will be used. I let them know the main information will be shown on the home page, allowing them to start a session easily. The journal will have all past sessions which can also be expanded and finally I showed them the profile page which will allow them to set their goals.

Harrison:

I enjoy the current design of the app being very simple with the main features in the app. This has sessions which is my main interest as I run and cycle daily. I like the map view and how it shows me where I have been. I personally don't mind the large buttons but if they weren't large, I would not mind as I know how to use fitness apps.

Abytom:

It's a very clean design. The goals are a welcome feature which is what I will be using mainly alongside maybe a few sessions a week. I like the calendar view you have which will show me how I am doing with my sessions.

Jamshed:

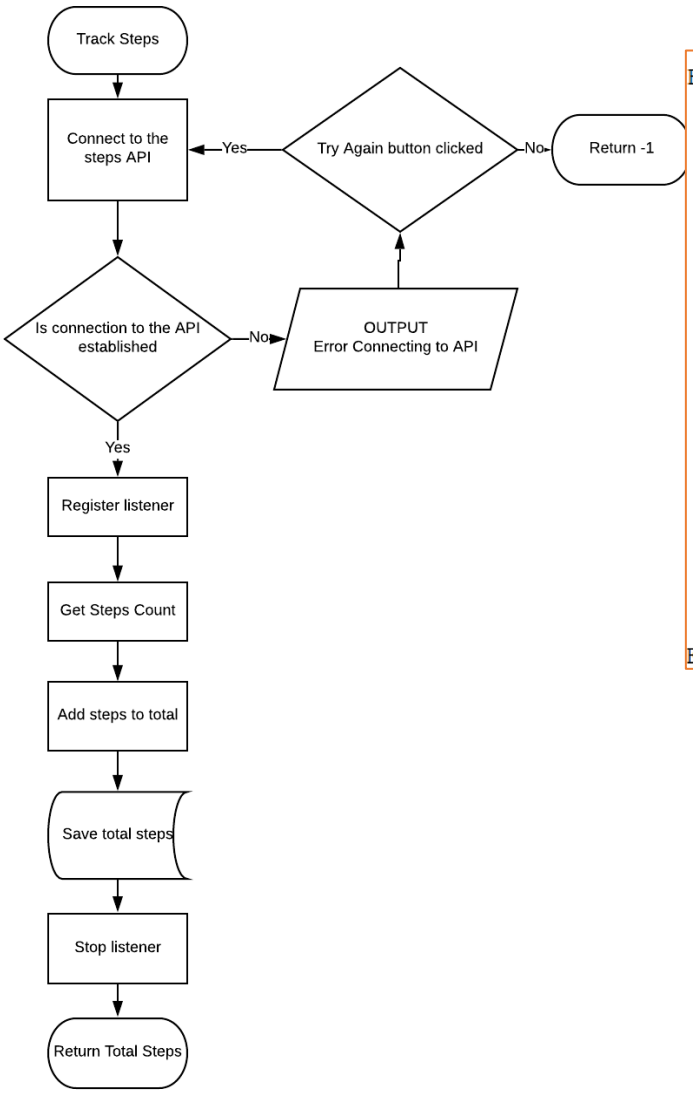
It's very clean and I love the icons included everywhere. I will definably use this app as it looks very simply to use after your explanation. I will not use it often for sessions but will focus on my steps and calories for the day. I love the progress bars implemented there.

Feedback:

I'm happy with all the responses as they seem to like the current design of the app. They liked the simplicity of the app and each had their own part which they liked. As all my stakeholders are satisfied, I can continue.

Algorithms

Track Steps

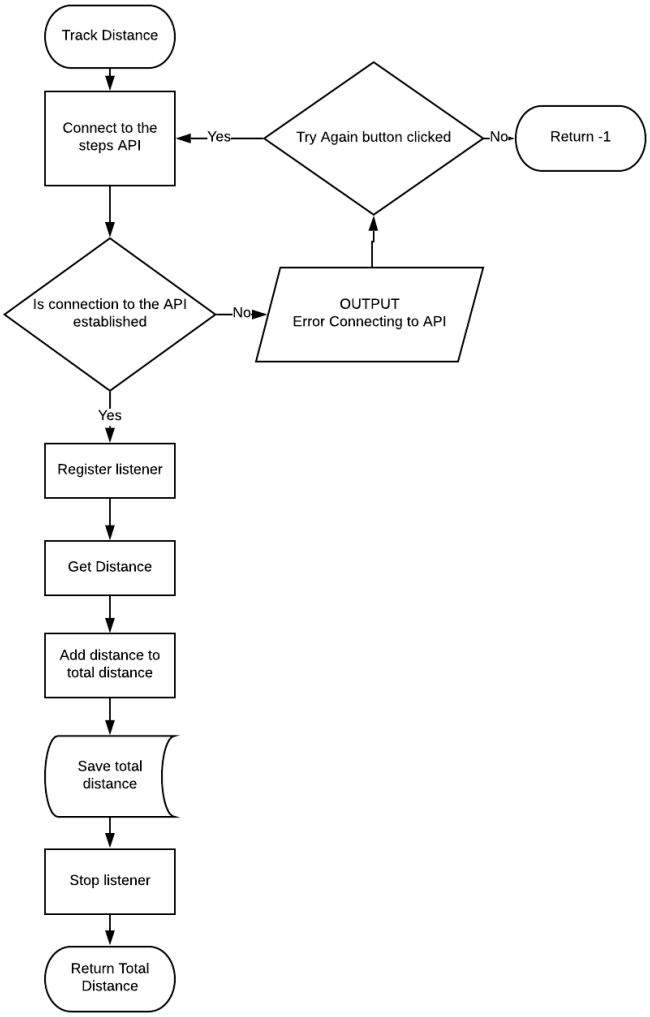



```
FUNCTION TrackSteps()

  status = Fitness.Subscribe(STEPS)
  WHILE status = false
    OUTPUT "ERROR Connecting..."
    tryAgain = INPUT
    IF tryAgain = TRUE THEN
      status = Fitness.Subscribe(STEPS)
    ELSE
      RETURN -1
    ENDIF
  END WHILE

  listener = Fitness.ListenRead(STEPS)
  steps = listener.read (STEPS)
  totalSteps += steps
  SAVE(totalSteps)
  listner.Unsubscribe(STEPS)
  RETURN totalSteps
END FUNCTION
```

Firstly, I try to connect to API until I either connect or the user decides to stop trying. The app will give feedback to the user instead of crashing and will let the user be in control. After connecting to the API, I register a listener for steps. Using the steps listener, I get the steps for a period which I then add to the total. The total is saved. Finally, I stop the listener and return the total steps counter. Stopping the listener will mean our app can run other tasks much faster without having to reserve resources to the listener.

Track Steps will be a function repeatedly on start-up and periodically throughout the execution of the app. This function will be used to update the counter on the main screen of the app.

Track Distance

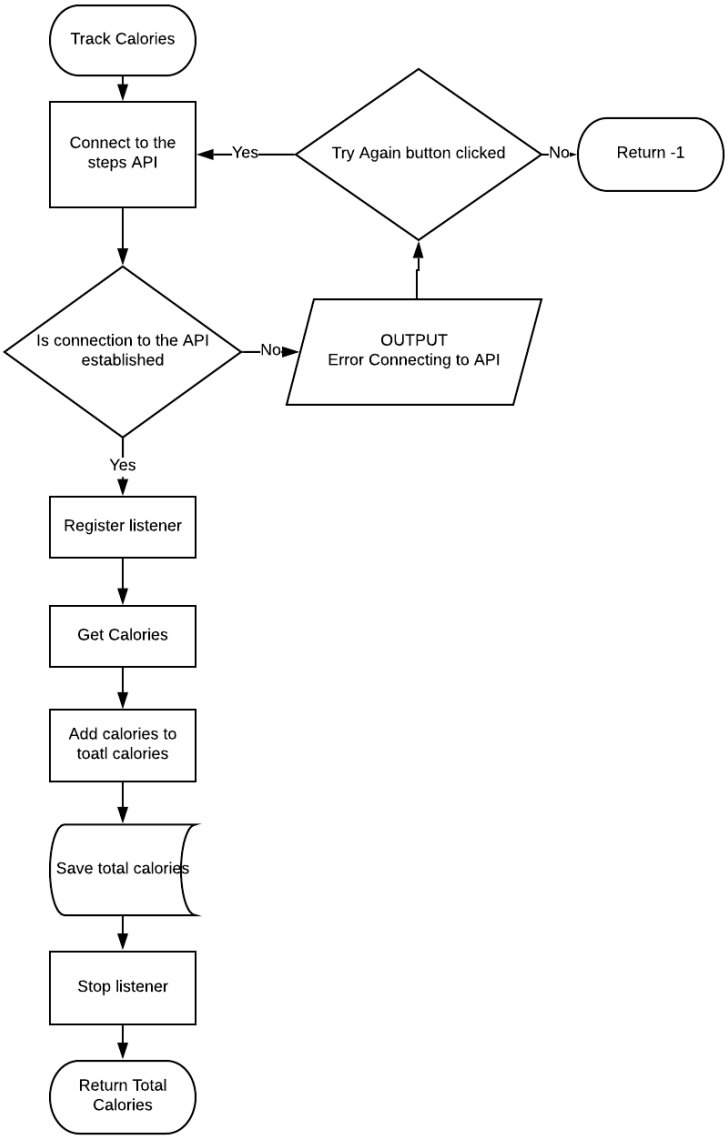


This function is very similar to the steps counter. However, I will be accessing different parts of the API so most of the code cannot be re-used as the two functions, while looking similar, will have to be treated very differently.

I first try connecting to the API, register a listener and finally get the distance and save it before stopping the listener and returning the total distance.

```
FUNCTION TrackDistance()
  status = Fitness.Subscribe(DISTANCE)
  WHILE status = false
    OUTPUT "ERROR Connecting..."
    tryAgain = INPUT
    IF tryAgain = TRUE THEN
        status = Fitness.Subscribe(DISTANCE)
    ELSE
        RETURN -1
    ENDIF
  END WHILE

  listener = Fitness.Listen(DISTANCE)
  distance = listener.read(DISTANCE)
  totalSteps += distance
  SAVE(totalDistance)
  listener.Unsubscribe(DISTANCE)
  RETURN totalDistance
END FUNCTION
```

Track Calories

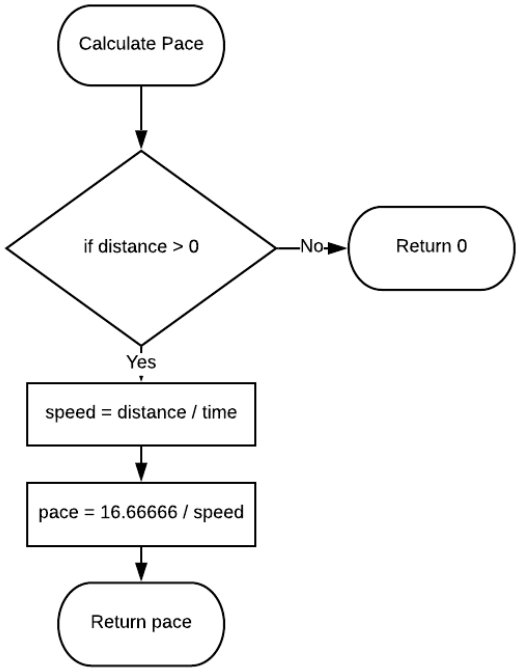



```
FUNCTION TrackCalories()
  status = Fitness.Subscribe(CALORIES)
  WHILE status = false
    OUTPUT "ERROR Connecting..."
    tryAgain = INPUT
    IF tryAgain = TRUE THEN
      status = Fitness.Subscribe(CALORIES)
    ELSE
      RETURN -1
    ENDIF
  END WHILE

  listener = Fitness.Listen(CALORIES)
  calories = listener.update(CALORIES)
  totalSteps += calories
  SAVE(totalcalories)
  Fitness.Unsubscribe(CALORIES)
  RETURN totalSteps
END FUNCTION
```

This algorithm will be similar to distance and steps as it will use the API to get the calories for the current user.

I first try connecting to the API, register a listener and finally get the calories and save it before stopping the listener and returning the total distance. If connection is failed, then I return -1 to let the program know.

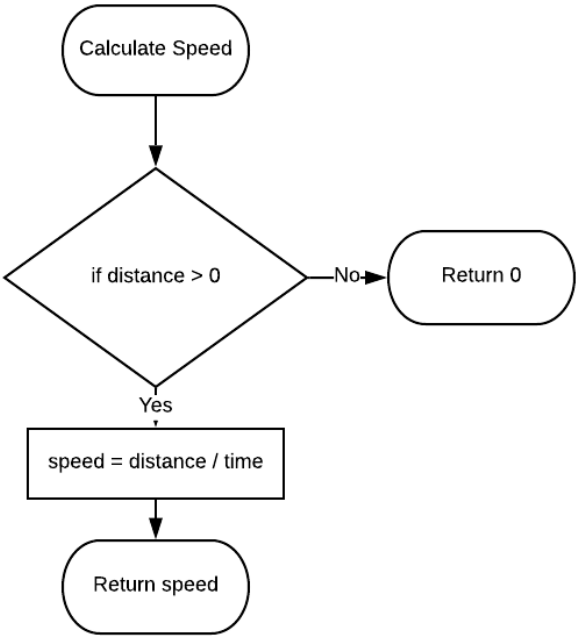
Calculate Pace




```
FUNCTION CalculatePace()
    IF (distance > 0) THEN
        speed = distance / time
        pace = 16.66666 / speed
        return pace
    ELSE
        return 0
END FUNCTION
```

The pace for running will have to be calculated. Because I will have the timer and distance, I can first calculate the speed using these two variables by dividing the distance by time. Once the speed is calculated I divide 16.66666 by the speed to get the pace which I then return. If no distance has been covered then the pace will be 0 so I return 0.

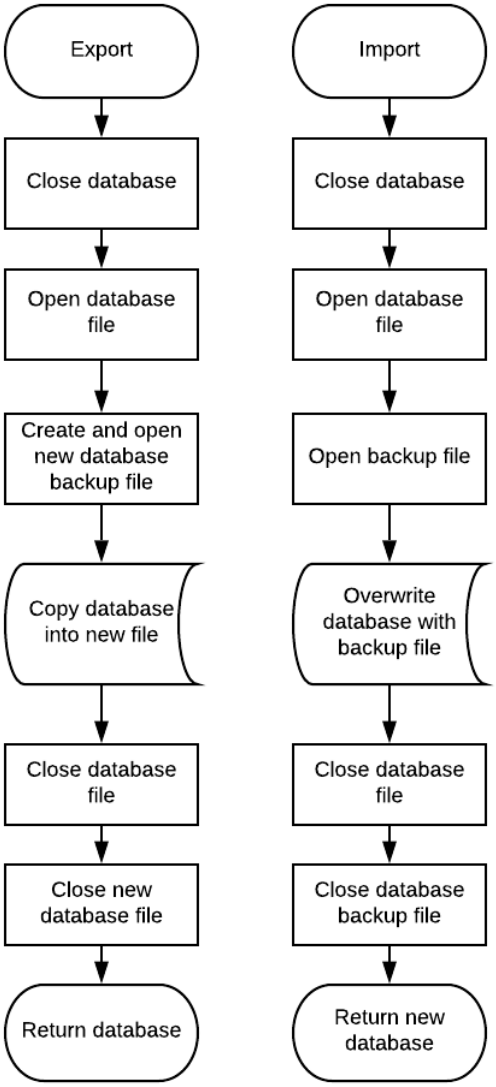
Calculate Speed




```
FUNCTION CalculateSpeed()
    IF (distance > 0) THEN
        speed = distance / time
        return speed
    ELSE
        return 0
END FUNCTION
```

The speed is calculated using the distance and time for the session. So first I make sure the distance is not 0, if it is then I return 0 as the speed is also 0 then. If it's not 0 then I divide distance by time to get the speed and return it.

Export and Import Database



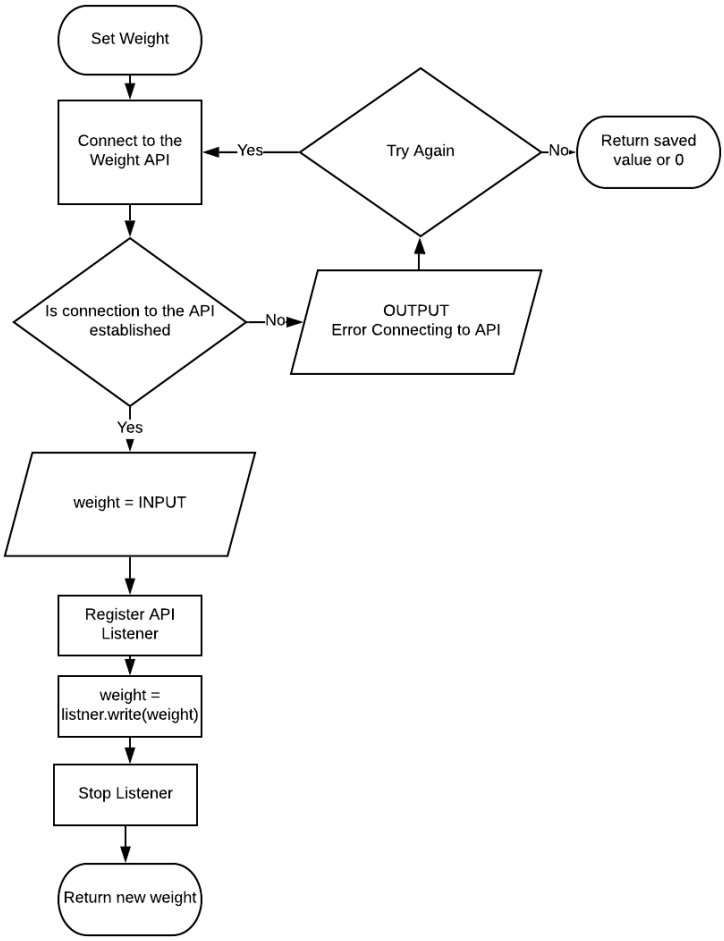

```
FUNCTION Export()
  database.close()
  instream dataFile = database
  outstream backupFile = File.NewFile()
  backupFile.write(dataFile)
  dataFile.close()
  backupFile.close()
  RETURN database
END FUNCTION

FUNCTION Import()
  database.close()
  instream dataFile = database
  outstream backupFile = backupfile
  dataFile.write(backupFile)
  dataFile.close()
  backupFile.close()
  RETURN database
END FUNCTION
```

The import and export methods are very similar because they open files and overwrite one with the other and close them. First the database has to be closed so it is not corrupted while being read and modified by the app concurrently.

For the export the database is opened, and a new file is created to copy the database to the new file. The new file and database are now closed, and the database is returned so it can be opened again.

For the import the database and backup database files are opened then the database is overwritten with the backup file before both files are closed. Finally, the database is returned so it can be opened again.

Set Weight/Height



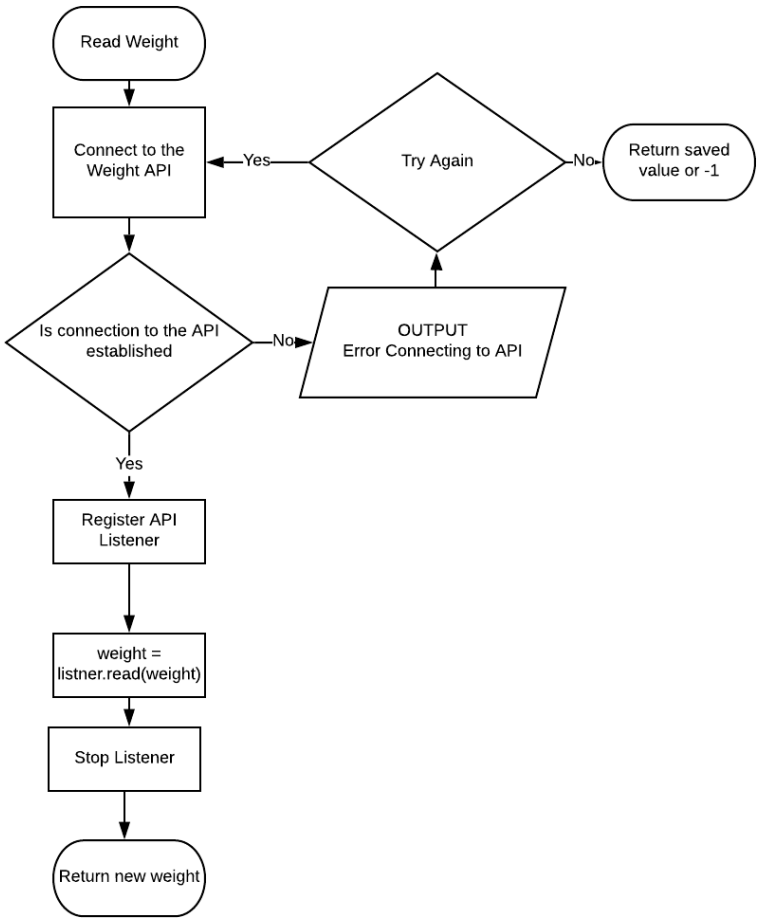

```
FUNCTION SetWeight()
    status = Fitness.Subscribe(WEIGHT)
    WHILE status = false
        OUTPUT "ERROR Connecting..."
        tryAgain = INPUT
        IF tryAgain = TRUE THEN
            status = Fitness.Subscribe(WEIGHT)
        ELSE
            RETURN -1
        ENDIF
    END WHILE

    weight = INPUT
    listner = Fitness.ListenWrite(WEIGHT)
    listener.write(weight)
    listener.Unsubscribe(WEIGHT)
    RETURN weight
END FUNCTION
```

```
FUNCTION SetHeight()
    status = Fitness.Subscribe(HEIGHT)
    WHILE status = false
        OUTPUT "ERROR Connecting..."
        tryAgain = INPUT
        IF tryAgain = TRUE THEN
            status = Fitness.Subscribe(HEIGHT)
        ELSE
            RETURN -1
        ENDIF
    END WHILE

    height = INPUT
    listner = Fitness.ListenWrite(HEIGHT)
    listener.write(height)
    listener.Unsubscribe(HEIGHT)
    RETURN height
END FUNCTION
```

Weight and Height will use the same algorithm as above but with the changed field.

First, I try to connect to the API. I keep trying and if it fails return 0. I t have to setup a listener with the API which I then use to read the value for either weight or height. Finally, I can stop the listener and return weight/height

Read Weight/Height



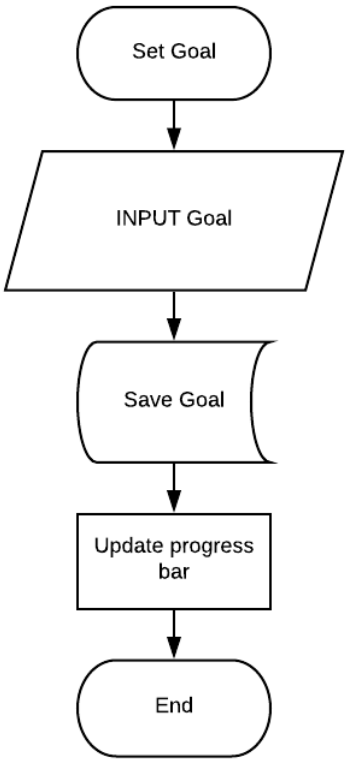

```
FUNCTION ReadWeight()
    status = Fitness.Subscribe(WEIGHT)
    WHILE status = false
        OUTPUT "ERROR Connecting..."
        tryAgain = INPUT
        IF tryAgain = TRUE THEN
            status = Fitness.Subscribe(WEIGHT)
        ELSE
            RETURN -1
        ENDIF
    END WHILE

    listner = Fitness.ListenRead(WEIGHT)
    weight = listener.read(weight)
    listener.Unsubscribe(WEIGHT)
    RETURN weight
END FUNCTION
```

```
FUNCTION ReadHeight()
    status = Fitness.Subscribe(HEIGHT)
    WHILE status = false
        OUTPUT "ERROR Connecting..."
        tryAgain = INPUT
        IF tryAgain = TRUE THEN
            status = Fitness.Subscribe(HEIGHT)
        ELSE
            RETURN -1
        ENDIF
    END WHILE

    listner = Fitness.ListenRead(WEIGHT)
    height = listener.read(height)
    listener.Unsubscribe(HEIGHT)
    RETURN height
END FUNCTION
```

Weight and Height will use the same algorithm as above but with the changed field.

First, I try to connect to the API. I keep trying and if it fails return -1. I then get the input from the user for their weight/height, update the data in my app. I then have to setup a listener with the API which I then use to write the new value to the API. Finally, I can stop the listener and return the new weight/height

Set Goal




```
PROCEDURE SetStepsGoal()
  goal = INPUT
  database.writeSteps(goal)
  updateStepsProgressBar(goal)
END PROCEDURE

PROCEDURE SetCaloriesGoal()
  goal = INPUT
  database.writeCalories(goal)
  updateCaloriesProgressBar(goal)
END PROCEDURE
```

The set goal method is very simple as I have to get valid input first, then save the goal so it can be restored on app restarts. The progress bar also has to be updated to show this new goal's progress.

Input and Outputs

These will be the main input and outputs of my program.

Input	Process	Output
Step, calories and distance counters	Save in database and update	Update main screen with new counters
Navigation menu	Will allow the user to switch between different screens	New views will open and old one will be closed
Session Start button	Will start a new view to which will show the map and allow the user to select a session type	Show session details for running and cycling
Stop button	Stops the session and saves the info in the database	Show the info for that session such as the distance, calories, speed and route
Map data	Save location coordinates	Show route on map
Export Button	Allow the user to select a directory to save database to	Saves backup database file to directory selected
Import Button	Let user select a file to overwrite the database with	Overwrites sessions with the sessions in the file
Steps Goal	Save goal and update progress bar	Update goal bar
Calories Goal	Save goal and update progress bar	Update goal value
Set weight and height	Open window to select their weight or height from	Send the new weight and height to their Google Account

Key variables and data structures

User

The user's information data will be stored on their Google account which means we do not have to store them in a local database but can store it in the application. The data about the user and the relevant application data will be stored in the app and will be attached to their Google account.

Field Name	Date Type	Description
Email	String (100)	Google email address of the user to save data to – will be saved in the application's data
First Name	String (20)	First name stored separately so it can be searched
Second Name	String (20)	Second name stored separately so it can be searched
Weight	Integer	User's weight is required to calculate the calories
Height	Integer	The height is required to calculate the calories

App Data

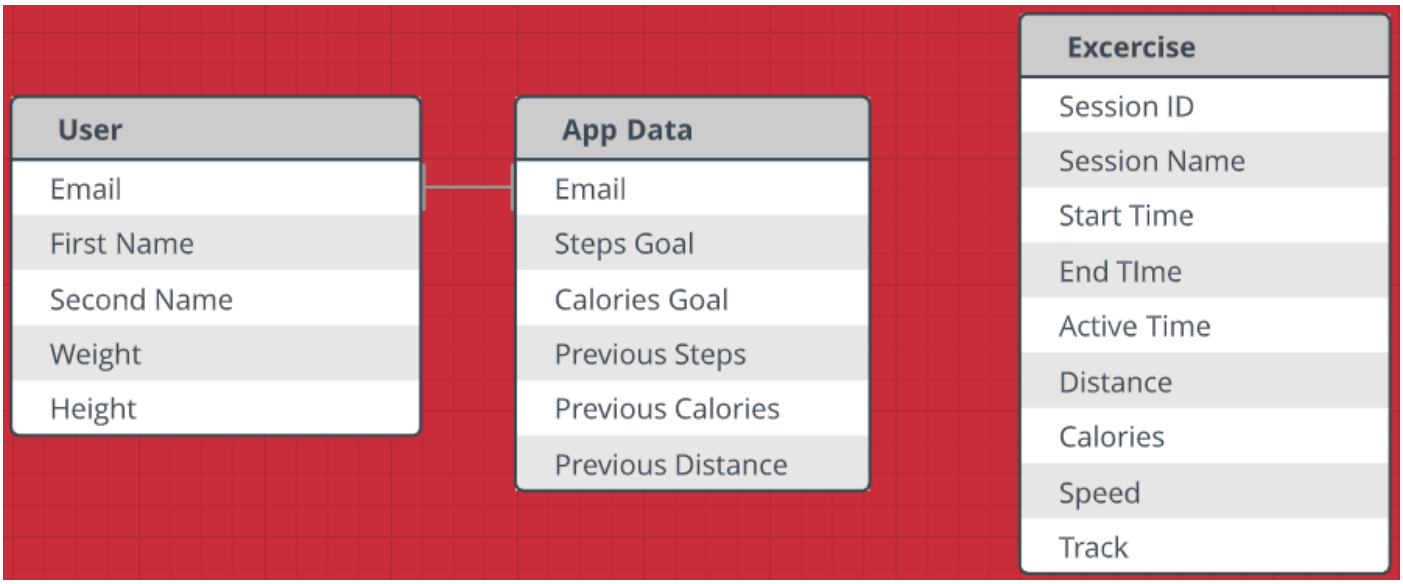
The app data will allow the program to load previously input data as well as any data retrieved from APIs to allow faster loading and reduce the number of times calls have to be made.

Field Name	Date Type	Description
Email	String (100)	Foreign key attached to the user's email to access their google account
Steps Goal	Integer	Will store the user's goal for their daily steps
Calories Goal	Integer	Will store the user's goal for their daily calories
Previous Steps	Integer	Hold the user's previously saved steps
Previous Calories	Integer	Will hold the user's previously saved calories
Previous Distance	Integer	Will hold the user's previously saved distance

Exercise

Each exercise will have its information stored. A class will be used to store this data so it can be displayed in the user's journal. All this data will be written to a database so it can be loaded into the program on start-up, backed up and restored by the user.

Field Name	Date Type	Description
Session ID	Integer	Primary key so the sessions can be sorted in order
Session Name	String (10)	Holds the type of the session whether it is Running or Cycling
Start Time	Integer	Will hold the start time in milliseconds
End Time	Integer	Will hold the end time in milliseconds
Active Time	String (5)	Will hold the session's active time in a string to show the user
Distance	Float	Will hold the distance – it's a float as distance is in km
Calories	Integer	Will hold the user's calories for that session
Speed	Integer	The speed will be stored as an integer for the session
Track	String	The locations that the user travelled through will be stored in a String format and will be converted into a custom location type when read, it will hold pairs of latitudes and longitudes



I will have 3 main entities to control the data in my program.

Firstly, the user entity will store all the data about the user, including their email and name, weight and height. The email and name will be handled by the Google Account and therefore will be stored in my apps cache so I can retrieve the data without having to request permission every time. The app data will allow store information that it requires to run such as the steps goal and calories goal. The previous steps, calories and distance will also be saved to allow the app to start up faster without having to make long API calls.

The exercise entity stores all the data for a specific exercise and data related to it. It will store the session ID so it can be sorted easily when being viewed in the journal. Every session will have the data as explained on the previous page. The track will hold the coordinates that the user travelled through.

Key variables

Field Name	Date Type	Description
Total steps/calories/ distance	Integer/Float	Transfer total steps/calories/distance between classes and display it.
steps	Integer	Will hold saved steps from previous session/launch. Will be used to retrieve data from the API for daily steps
cals	Integer	Will hold saved calories from previous session/launch. Will be used to retrieve data from the API for daily calories
distance	Integer	Will hold saved distance from previous session/launch. Will be used to retrieve data from the API for daily distance
tracker	Tracker	Will be used to track and record steps/calories/distance. This will hold data to make calls to the API for the relevant data and hold it so it can be used.
Session	Custom Session data type	Hold start and end time for a session for Fit API. This will be stored in an SQL database as well so it can be reloaded into the journal after the app has been closed. This class will transfer all the related data to display a summary as well. Using this I can save and restore sessions easily from the database
weight	Float	Will hold the weight for the user so it can be sent to the API to update calculations made for calories.
Height	Float	Will hold the height for the user. It will be sent to the API to update calculations for calories
stepsGoal	Integer	Will hold and transfer the user's steps goal around the program. This will be used to show progress to the user on the main screen and will also be saved so it does not have to be reset after every launch.
caloriesGoal	Integer	Will hold and transfer the user's calories goal around the program. This will be used to show progress to the user on the main screen and will also be saved so it does not have to be reset after every launch.
GPSLocations	Coordinates	Will be stored for each session to keep track of the route. These coordinates will be used to draw the route on the map when the user wants to view their session details.
Google Account	String	Will be saved to allow data from the user's account to be retrieved without requiring log in every launch

Validation Methods

To keep the app working and prevent crashing, the app must have validation to make sure the user's input data is correct data. There will be the inputs of the user and how I go about controlling the values to make sure they are valid.

To keep with the theme of a mobile app, most of my validation will be performed using dialog boxes with limited values. I will also limit the input to numbers for some inputs to make it not only easy for the user to input easily but to make sure any input data is valid. This will also make the app more intuitive to use as they know what input the app is expecting more easily and reduce wasted screen space. Allowing a more user-friendly interface as well as reduces any errors from input.

There are 3 types of validation: normal, extreme and erroneous. Normal data is expected data which is accepted. Extreme data is on the edge of acceptable data. Finally, erroneous data is invalid data which is not accepted.

Buttons

The buttons are linked to certain methods and will not allow any other data to be input as the user can only press and select them. This means there will be no validation required on that input as buttons will be shown when they can be used only.

Drop Downs

Drops downs will have predefined values which the user can select from so there is no risk in input that is not expected to be entered. This will reduce the amount of validation I have to carry out as the user will have limited valid options.

File/Directory Selection

The file or directory the user selects for export and import will be valid as Android will limit them to the directories that the app can have access to. For the read method I will limit the selection to only be of a certain file type so only valid databases are restored. This means the file opened can only be a valid one as the user will not be allowed to select and input a different type of file.

Weight, Height, Goals

The rest of the fields will have dialogs and will not have much validation as limits will be set on the values shown to the user. They will be simple.

Field Name	Validation
Weight	20 <= Weight <= 650
Height	30 <= Height <= 300
Steps Goal	0 <= Steps < 100000
Calories Goal	0 <= Calories < 100000
Data from API	Validate the data from API before trying to access and use it.
Saved data	Make sure saved data exists and is valid before using it

For the Weight and Height, I will have a dialog with a limited number of values, so the input does not have to be validated due to the user not being able to input anything illegal, this will also make it easier for the user to select. These values will be between 20 KG and 650 KG and 30cm and 300cm which is adequate for the user to input.

For the goals I will have a limit of 5 numbers and only allow digits to be inputted to make sure valid data is input. This will mean the user can not set a goal which is too large to store as well as a goal that would be completely unrealistic. Updating the progress bars will be easy as there is already a limit for the goals

One of my main validations will be making sure the data retrieved from the API. For API calls I will have to make sure I only access and work on the data when it is valid and exists. This means my app will not access data that it does not have permissions to access from the user which would otherwise cause the app to crash. Also, if it does not exist meaning there was no data to retrieve or there were any other issues, I have to make sure I don't access it as it could cause a crash.

Muhammad Faizan Alam

Development Testing

I will be testing the app thoroughly throughout development which I will document every run and fix any issues occurred. At the end of each version I will run a larger test to make sure the overall features for that version are working which I will plan below.

Version 1:

Permissions

Test Use	Justification	Expected Outcome
Permission Dialogue shown	Permission dialogue should appear over the activity and request permission to fitness data	Permission dialogue requests permission from the user for the app
Request Location Permission	Permission for location should be requested if not granted. If the user denies then show the user how to manually grant the permissions from phone settings	Location permission dialogue shown OR method to grant permission manually

Steps/Calories/Distance

Test Use	Justification	Expected Outcome
Subscription is made	Subscription to track the steps/calories/distance is made to the fitness API	Permission method executes successfully – prints message to log
Track steps/calories/distance	Steps/calories/distance are tracked and are accurate	Latest count is retrieved and shown in the log. Counts are the same as Google Fit app.
UI is updated	UI is updated for steps, calories and distance	The user is shown the counters on the main screen
Steps/Calories/Distance counters are restored	When the app is launched, the counters previously saved are loaded in again until they're updated	Counters saved from previous launch are shown on launch

Version 2:

Start Session

Test Use	Justification	Expected Outcome
Start Session View shown	A new view should be opened to allow the user to start a running/cycling session. This view must have a map view to show the user his location	View to select and start a session is shown with a map on top
Count Down before starting	Count down is shown to the user when they start a session, so they have time to prepare for it	Simple count down shown before actually recording the session
Track distance/calories /pace/speed	The start session should track the user's distance calories, pace/speed and show it to the user as the session is happening.	User is shown distance, calories and pace/speed as they run/cycle
Track location as user moves	The user's location should be tracked as they move and should be updated on the map	User is shown their location as they move
Show summary of session with details	To let the user, know how they performed in that session after they have completed it	Should show a summary of the session with their time, distance, calories and speed/pace.
Show outline of track on summary screen	Lets the user know where they travelled through on a map in the summary screen	The map should show a highlighted line of the user's movement throughout the session

End Session

Test Use	Justification	Expected Outcome
Show summary of session with details	To let the user, know how they performed in that session after they have completed it	Should show a summary of the session with their time, distance, calories and speed/pace.
Show outline of track on summary screen	Let the user know where they travelled through on a map in the summary screen	The map should show a highlighted line of the user's movement throughout the session

Version 3:

Journal

Test Use	Justification	Expected Outcome
Navigation Menu	Let the user easily switch between views with a navigation menu at the bottom as done in mobile apps	Icons in the navigation menu should change the screen when pressed
Sessions in journal	Let the user see their previous sessions which are sorted by date	Scrollable session screen showing the user's previous sessions
Session Summary	Let the user see their previous sessions in detail when clicked on	Open a new screen with the session in more detail

Database backup

Test Use	Justification	Expected Outcome
Export database	To allow the user to export a backup of their sessions which can be imported into another device	Should export sessions file to the directory the user selects
Import database	Allow the user to import sessions backup they have previously created	Let the user select a backup file and import the sessions from it into the app

Profile Screen

Test Use	Justification	Expected Outcome
User's name and picture	Shows the user what google account they are logged in with as they usually have multiple	Show the user's name and picture in the info screen
Set goals	Let the user set goals for themselves for steps and calories	Get input for the goal from the user and update the progress bars on the main screen
Set weight and height	Let the user update their weight and height so the app's calculations are more accurate	Get input for new weight/height from the user and update their overall weight/height with it

Approach to Testing

Version 1

The first version should have test runs to demonstrate the features below work as they intended.

- Steps are recorded and displayed accurately
- Calories are recorded and displayed accurately
- Distance is recorded and displayed accurately
- Steps/Calories/Distance are saved and reloaded on start-up
- Permission are requested and tested thoroughly with different user input

Test Use Case	Explanation	Expected Outcome
Steps are recorded and displayed accurately	Log in screen shown on first launch	Log in screen should only be shown on the first start-up
Values displayed	Values for steps, calories and distance shown on main screen	The main screen should have up-to-date values for steps, calories and distance
Update values when moving	Moving updates, the step, calories and distance values	Values are updated appropriately with movement. E.g. 3 steps should increment the value of steps by 3

Alpha Tests

The first tests that I am going to run are Alpha which test the basic system and its functionality. I will be using this testing rigorously before the app is ready. I will be testing each module separately in alpha testing to make sure they run as expected.

There are two types of alpha testing, white box and black box. White box testing is where I will know how the system is setup and therefore will know which values to input and check. I will be making sure:

- Data is accessed correctly from the database and files
- Sensor data is read correctly and as accurately as possible
- Correct calculations are made
- Code is commented and indented

In black box testing, I will focus more on the GUI of the app. I will check how the input and outputs are done. I will input various details to make sure that the correct results are calculated and outputted. I will also verify the values input by the user as well as checking the GUI for every input is setup correct.

- Check values for all the outputs on the main screen
- Make sure the main screen has all the info required
- Input valid and invalid data
- Correct output is shown

Test Use Case	Explanation	Expected Outcome
Log-in screen	Log in screen shown on first launch	Log in screen should only be shown on the first start-up
Values displayed	Values for steps, calories and distance shown on main screen	The main screen should have up-to-date values for steps, calories and distance
Update values when moving	Moving updates, the step, calories and distance values	Values are updated appropriately with movement. E.g. 3 steps should increment the value of steps by 3

Muhammad Faizan Alam

Beta Tests

I will use alpha tests to make sure my program is setup correct and everything is working as it should before refining the code and allowing a user to test the program. I will use **beta testing** to receive feedback from the user. I will use any feedback to add to the program, so the end-product meets user's requirements.

I will create some users to test outputs and calculations that I will be using in my program. This will allow me to check if outputs meet my expectations.

Test Data

User 1

Field Name	Details
Email	muhala181loreto@gmail.com
First Name	Faizan
Second Name	Alam
DOB	10/04/2001
Gender	Male
Weight	250
Height	210

User 2

Field Name	Details
Email	fitnessappTest@gmail.com
First Name	Fitness
Second Name	App
DOB	25/12/1975
Gender	Female
Weight	55
Height	160

User 3

Field Name	Details
Email	fitness@gmail.com
First Name	F1Tn
Second Name	ATS2
DOB	25/05/2003
Gender	Male
Weight	60
Height	195

Justification of Data

For user 1 the birthday I used is an extreme value which should be accepted but is close to something that would show an error. For User 2 it's normal data and should be accepted as well. For user 3 the birthday is out of range and should not be accepted.

I've also used extreme but valid data for weight and height for User 1 which should be accepted. This is to test the validation methods to accept data, especially extreme data.

If the user decides not to use their Google account, then they will be shown a screen with a log in button. This app unfortunately will not work without accessing their google account because one is required to get access to the user's sensors.

Acceptance Tests

Acceptance tests will be run after the development of the app. I will make sure the correct dialogs are shown when a user performs an unexpected or invalid action.

Test Use Case	Input	Expected Outcome
Don't log-in	Log-in dialog dismissed	Show main-screen with log-in button
Access Denied	User revokes permissions	Show error dialog and ask for permissions again.

Post Development Testing

Test No.	Description of Test	Test Data	Expected Result
Main Screen			
1.	Log-in with google account	**Valid Data:** Google Account signed in with valid email and password **Invalid Data:** Email and password are not valid for a google account	**Valid Data:** Google Account signed in **Invalid Data:** Ask for log-in info again until valid data and don't load any data to prevent crashing
2.	Ask for Google Account Permissions	**Valid Data:** Allow or deny pressed **Invalid Data:** Try to continue without response	**Valid Data:** **Allow**: Continue till next permission **Deny:** Ask for permissions again on next launch **Invalid Data:** Show message to grant permissions
3.	Ask for device permissions	**Valid Data:** Allow or deny pressed **Invalid Data:** Try to continue without response	**Valid Data:** **Allow**: Continue till next permission **Deny:** show prompt to give access to permission manually, don't load any data that requires the permission **Invalid Data:** Don't continue, not possible as user must either grant or deny permissions
4.	Update steps, calories and distance	**Valid Data:** Steps, calories and distance are read **Borderline Data:** No updated data is read **Invalid Data:** No data is read at all	**Valid Data:** Update steps, calories and distance counters **Borderline Data:** Restore previously saved values **Invalid Data:** Restore previously saved values or show it's loading
New Activity Session			
5.	Start Session Button pressed		Session is started with no errors. A map, session, type selection and the start button are shown
6.	Session Type Selected	**Valid Data:** Selects running or cycling **Invalid Data:** Not possible as there are only 2 options – don't accept any other type of input	Changes the session type to be started to the one chosen by the user
7.	Session is started		Starts a countdown which the session is started after
8.	Session data is updated as user runs/cycles	**Valid Data:** User moves around **Borderline Data:** User does not move **Invalid Data:** No invalid data possible	**Valid Data:** Map is updated with user's location. Distance, calories and pace/speed is updated as user moves **Borderline Data:** Everything is still updated but if the user doesn't move then everything except for the calories and timer will stay the same

9.	Session is stopped		Stop the session tracking and record all data of the session
10.	Session Summary		Summary is shown to the user with the relevant data such as time, distance, calories, pace/speed as well as their route
11.	Back button		Goes back to the previous session screen and then to the main screen
Navigation			
12.	Navigation Menu Buttons		Opens either Home, Journal or Profile screen when selected
13.	Navigation Menu hidden for new session		The navigation menu is hidden when a new session is being started and shown when back to home
Session's Journal			
14.	Journal Screen		When opened it should show the user's previous sessions sorted by date with most recent first
15.	Expanded Session		Session can be expanded to view more details such as the route taken, calories and speed/pace
16.	Export Session	**Valid Data:** User selects a directory **Borderline Data:** User does not select a directory	**Valid Data:** Sessions are exported to a new file in that directory **Borderline Data:** Nothing is exported, and the program continues to work as normal
17.	Import Session	**Valid Data:** User selects valid file **Borderline Data:** User does not select a file **Invalid Data:** Selects an invalid file	**Valid Data:** The sessions are overwritten with the sessions from the imported file **Borderline Data:** Nothing is imported, and the same data is used for sessions **Invalid Data:** User should not be allowed to select an invalid program as it will crash the program so selection of other files should be disabled
Profile Page			
18.	User's image is loaded	**Valid Data:** Image is retrieved **Invalid Data:** No image is retrieved	**Valid Data:** Image in profile view is updated with user's image **Invalid Data:** Image is set to default profile picture – no error should occur
19.	Set steps and calories goal	**Valid Data:** Reasonable number is typed in **Borderline Data:** Large number is typed in **Invalid Data:** Characters other than numbers are typed in	**Valid Data:** Step goal is set to the number input by the user **Borderline Data:** Goal is limited to a number of digits to keep the value realistic **Invalid Data:** Any character that's not a number is not accepted in input

20.	Progress bars are updated	**Valid Data**: User's steps/calories change, or user changes their goal **Borderline Data**: Goal is 0, reached or passed	**Valid Data**: Progress bar is changed to reflect new progress **Borderline Data**: If 0 then no progress is shown. If goal is reached or passed, then the progress bar is filled 100%
21.	Set Weight and Height	**Valid Data**: Reasonable number is typed in **Borderline Data**: Number too large or small is input **Invalid Data**: Characters other than numbers are typed in	**Valid Data**: Set's the weight to the new value **Borderline Data**: Numbers are limited so they continue to be realistic **Invalid Data**: The user is given a select and is not allowed to enter letters so the input will always be valid
22.	Steps, calories, weight and height are restored on restart		Make sure any data set by the user is saved and restored even after the app is restarted

Development and Testing

Tags to explain code, testing and write up in development and testing:

Why I carried out and solved a certain problem in the manner that I did.

What validation I included to make sure the data received into the app was valid.

Testing the validations to make sure they work correctly.

Algorithms I wrote to carry and return certain data.

Screenshot of the app being executed.

When the program ran into an error either at compilation time or run time.

When the app crashed with the message included.

The remedial code and action I took to fix either an error or crash.

Successful!

Included at the end of large test runs to make sure everything is working as intended.

Version 1

Setting Up

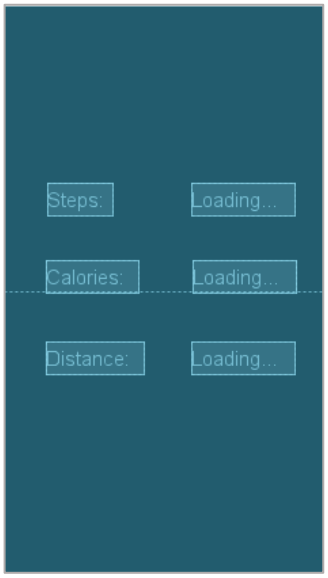
Before I start working on the first module, I need to setup a thing:

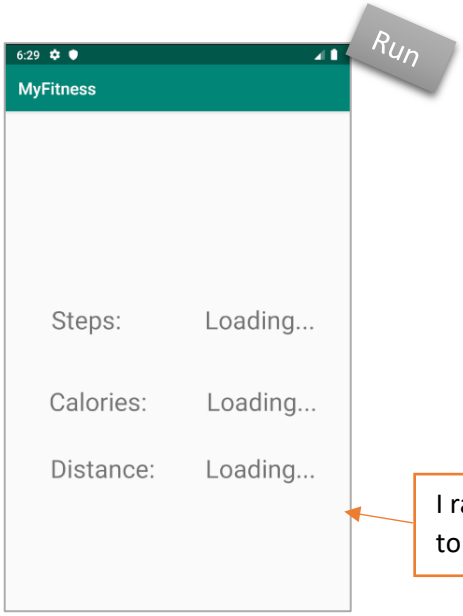
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.faizan.myfitness">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
```

My app will be accessing the devices precise location as well as using the internet to access Google Fit History API. I have to make Android and the user aware of this.

Next, I got authentication keys for the Google Fitness API for my app from Google API Console. This will allow the app to request information from it.

.





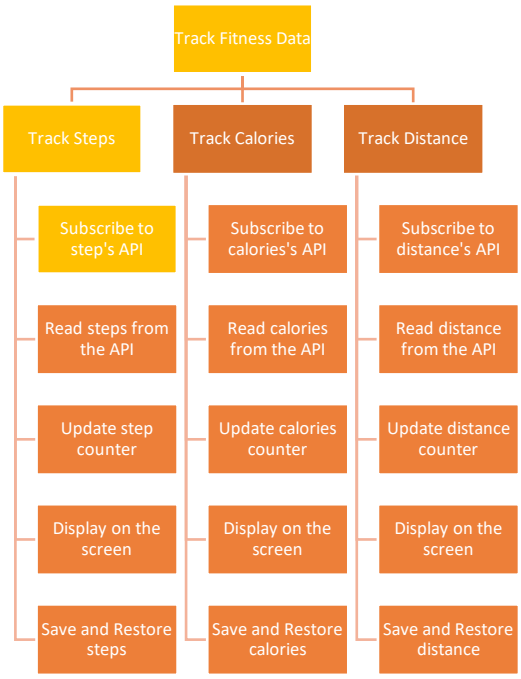
To start the development phase of my app I first had to setup a simple GUI to show the essential values. For now, I set the TextViews to have a placeholder of Loading. These will be replaced with the actual value when it's retrieved.

This layout then has 6 TextView widgets. Three of the left ones are used to describe the values which I will be changing dynamically as the app is run on the left. For now, these TextViews have a 'Loading…' placeholder. I will replace them with progress bars in the future when I refine the app.

The default screen for now consists of a ConstraintLayout which is used to align the widgets. The reason I use ConstrainLayout is because Android is run on many different phones which means I must deal with many different types of resolutions. ConstraintLayout automatically aligns the widgets with the constraints.

Any strings for the text and the text size were stored in resource files so they can easily be modified in the future.

Module 1 – Track Steps



The first module I will be focusing on is tracking steps. For this I have to connect to the Google API first. I researched how to record steps and get steps history. I used the official documentation [1] to research methods to create a subscription.

[1] https://developers.google.com/fit/scenarios/record-steps

Justification

I have two choices to getting the data. I could inherit an AsyncTask and override its function to carry out my subscription in the background or I could use an AsyncTask subclass. For now, I will use an AsyncTask subclass as I will not need many of the features inherited from AsyncTask. The reason I need to use an AsyncTask is because the subscription and connection to the API needs to be made in the background so the UI is not frozen which could suggest a crashed app to the user.

Subscribe Method

Subscribe to Recording - StepCounter

To handle the step counter, I will create a new class called StepCounter which will handle the subscription and updating the counter as the application is run.

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
implementation 'com.google.android.gms:play-services-fitness:18.0.0'
```

I had to add the following implementations into the gradle properties file to access the APIs

```
import android.content.Context;
import android.util.Log;

import androidx.annotation.NonNull;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.fitness.Fitness;
import com.google.android.gms.fitness.data.DataType;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
```

The following imports were added into the StepCounter class file as well. These were required for the subscribe() function to access the fitness and sign in APIs.

```
// function will subscribe to get a live counter for steps, the subscription will run even when the app is stopped
void subscribe() {
    Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context))    // register a client with the API
            .subscribe(DataType.TYPE_STEP_COUNT_CUMULATIVE)                              // subscribe to record all steps through the day
            .addOnCompleteListener(
                    new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                Log.i(TAG,  msg: "Successfully subscribed");             // has completed successfully, so log message
                            } else {
                                Log.w(TAG,  msg: "Problem subscribing", task.getException()); // has failed, so log message
                            }
                        }
                    });
}
```

Algorithm

I have created a subscribe() method which is project-private because it has no modifier. From the documentation I found out I could subscribe to the TYPE_STEP_COUNT_CUMULATIVE type from the recording client. This subscription will allow the device to track the user's steps even when the app is stopped and not running. I need this subscription feature because users will be using other apps and only check their fitness app a few times a day or week. I add a listener to the subscription as well, so I know when the subscription succeeds and fails.

Validation

I check if the subscription was successful before using the data retrieved by the subscription. If it succeeds, I just log the message for now but will handle the data returned later to update the steps. If it fails, I log the error as well as the exception that occurred. I will test this validation below when the MainActivity has been implemented.

```
private Context context;    // get parent activities context to pass into methods


StepCounter(Context context) {
    this.context = context; // initialise the member variable
}
```

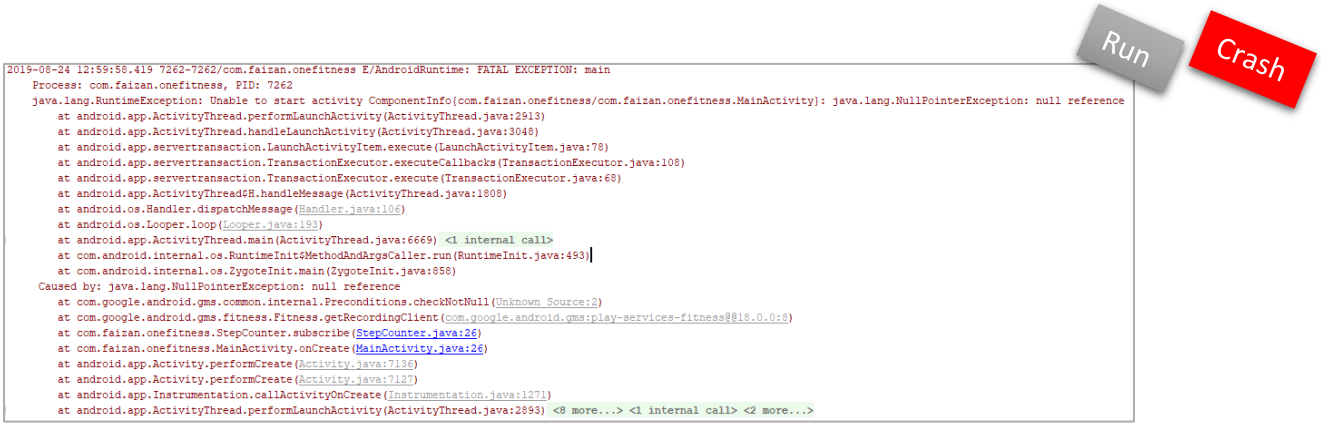
I had to get the context of the MainActivity for the subscribe() method. I get this value using a constructor because it will be used in the future by other functions. I keep the member variable context private, so it is encapsulated from being accessed and changed by other classes.

MainActivity

```
public class MainActivity extends AppCompatActivity {
    private StepCounter stepCounter;                        // used to make subscription and read steps

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                 // create superclass' activity(basic)
        setContentView(R.layout.activity_main);             // set layout for this activity

        stepCounter = new StepCounter( context: this);      // create an object of stepCounter
        stepCounter.subscribe();                            // subscribe to steps tracking
    }
}
```

The onCreate() method in the MainActivity class is the first method called. Here I will be initialising the UI and other classes. For now, I want to test the subscribe function.

I first created a member variable for the stepCounter. The onCreate() method is called first to populate the UI. Here I also initialise the stepCounter and pass it the MainActivity() as context. The context is used to return to this activity after the log-in activity finishes executing.



Running the app caused a crash. Looking at the logs it seems to have come from the subscribe function, more precisely the sign-in line. The log shows that I called a method on a null object.

```
void subscribe() {
    Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context))    //
            .subscri
            .addOnC
```

Argument 'GoogleSignIn.getLastSignedInAccount(context)' might be null more... (Ctrl+F1)

I am told that it might be null when running the program. I need to now figure out a way to create a sign-in activity for the first launch. I had previously believed calling this method would trigger that activity.

I learned from the documentation that GoogleSignIn class has two methods, hasPermission() and requestPermission(). I can use these methods to check for the permissions first before requesting them. This will allow me to only request these permissions on the first launch only.

Fix

```
// type of data we want to request
FitnessOptions fitnessOptions = FitnessOptions.builder().addDataType(DataType.TYPE_STEP_COUNT_CUMULATIVE)
                                                        .build();

if (!GoogleSignIn.hasPermissions(GoogleSignIn.getLastSignedInAccount( context: this), fitnessOptions)) {
    // request permission to data if we don't have access
    GoogleSignIn.requestPermissions( activity: this,
                                     REQUEST_CODE_STEP_COUNT,
                                     GoogleSignIn.getLastSignedInAccount( context: this),
                                     fitnessOptions);
    Log.i(TAG,  msg: "Requested permission");
} else {
    stepCounter.subscribe();                                    // subscribe to steps tracking
}
```

I first made an options object with the permission I wanted to request. I can add future permission requests to this line as well which will be passed onto the requestPermission() function. Next in the if statement I check if the app has the required permissions using the hasPermission() function, if not then the permissions are requested using requestPermission(). I pass the MainActivity as the context, so the Sign-in activity returns back to the current app. I will be adding many Log lines which I use when debugging. These lines are omitted outside of debug mode.

```
private static final String TAG = "MyFitness-MainActivity";
private static final int REQUEST_CODE_STEP_COUNT = 4444;
```

I added these two member variables which are constant and cannot be changed. They are placeholders to make the code more readable.

Justificati

There are two reason why I request permission from the MainActivity and not from the StepCounter. Firstly, it is good practice in Android to call new activities from an existing activity class. The second reason is because in the future I will be requesting permission to other data pints from the Fitness API and therefore I want to encapsulate the step tracking from the rest of the program to keep the program modular.

Validation

Permissions from the google account signed in are retrieved and requested if they are were not previously granted. This allows the program to fetch the steps. If the data was retrieved immediately, it would cause a crash.

Run

```
com.faizan.onefitness I/MyFitness-MainActivity: Requested permission
```

Error

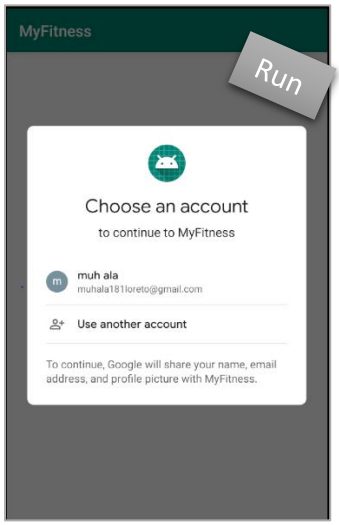
Running the program, it requests permission as expected. However, the subscribe() function is never executed. Looking at the code I quickly realise it is only called when the program already has permission from a previous launch. Running my app again confirms my suspicion.

Fix

```
// run after the sign-in activity closes
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == Activity.RESULT_OK) {     // if the log-in acitivity exited successfully
        if (requestCode == REQUEST_CODE_STEP_COUNT) { // and the request code matches
            stepCounter.subscribe();            // then subscribe to the step counter
        }
    }
}
```

Justificatio

I could call the subscribe method straight after requesting permissions but if the sign-in activity returns with an error then the subscribe() method will be called with a null object, just like the issue I had in the beginning. To fix this I used the onActivityResult() method to check that the sign-in activity exited successfully. I then checked if the activity was called using my request code before calling the subscribe() method. This means the subscribe() method is never called with a null object as it checks the sign-in dialog.




```
com.faizan.onefitness I/MyFitness-MainActivity: Requested permission
com.faizan.onefitness I/MyFitness-StepCounter: Successfully subscribed
```

Successful! Running the app again, I can see that permissions are requested, and the subscription is successful!

Validation Testing

Because this is the first launch the menu to log-in is shown before the steps are requested from the user as shown on the right. This prevents a crash if I were to use the user's account without their permission. I verify the permissions have been granted and the subscription has been successful before continuing. If the user does not log-in, then the subscription is never executed which would normally cause a crash.

Test Use	Explanation	Actual Outcome	Works?
Permission Dialogue shown	Permission dialogue should appear over the activity and request permission to fitness data	Dialogue appears and requests permissions from the user for their fitness data	Yes
Subscription is made	Subscription to track the steps/calories/distance is made to the fitness API	The log shows the subscription has been made successfully to track steps.	Yes

```
import android.content.Intent;
import android.util.Log;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.fitness.FitnessOptions;
import com.google.android.gms.fitness.data.DataType;
```

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

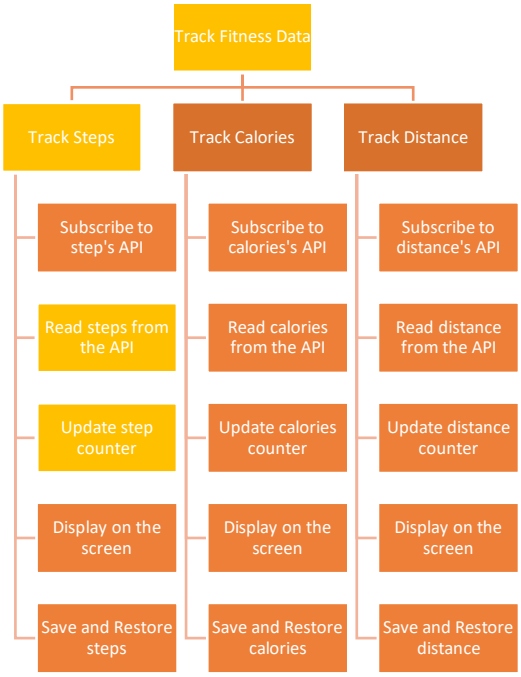
The authentication service had to be implemented to access the permission activity. I also added the following imports to MainActivity to access the sign-in activity and fitness options.

```
} else if (resultCode == Activity.RESULT_CANCELED) {
    // show message to user to grant permissions on restart
    Toast.makeText( context: this,  text: "Error: Restart app to grant permissions", Toast.LENGTH_LONG).show();
}
```

I added the following in the onActivityResult to let the user know the program will not work without permissions and a restart is required to grant permissions.

To summarise this section, what I have created is a subscription to let the Google Fit API know that the app will be tracking steps throughout the day. This means the API will track the user's steps even when the app is not running. Most people will be using other apps on their phone throughout the day or will have them off. Making a subscription to this data will allow me to continue to retrieve up-to-date results for the steps.

Read Daily Steps



Now that I have the subscription set up, I need to actually make a request to read the data that is logged by the subscription. I will read the data which then needs to be sent to the MainActivity so the activity can update the UI. I will do this using an intent which I will cover below.

I will use the documentation [1] for the API to help aid me in retrieving the daily steps. I have to use the HistoryClient to request the delta count and then send that count back to the MainActivity

[1] https://developers.google.com/fit/scenarios/read-daily-step-total

Permission - MainActivity

```
// type of data we want to request
FitnessOptions fitnessOptions = FitnessOptions.builder().addDataType(DataType.TYPE_STEP_COUNT_CUMULATIVE)
                                                        .addDataType(DataType.TYPE_STEP_COUNT_DELTA)
                                                        .build();
```

To begin with I have to add the TYPE_STEP_COUNT_DELTA permission to the fitnessOptions which are requested from the user in the MainActivity:onCreate() method.

Get Daily Total - StepCounter

```
// will retrieve daily steps
void updateSteps() {
    Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context))// register a HistoryClient with account
            .readDailyTotal(DataType.TYPE_STEP_COUNT_DELTA)                         // read daily steps using COUNT_DELTA
            .addOnSuccessListener(                          // listener executed when method succeeds
                    new OnSuccessListener<DataSet>() {
                        @Override
                        public void onSuccess(DataSet dataSet) {    // get the total on success
                            long total = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_STEPS).asInt();
                            Log.i(TAG, msg: "Total steps: " + total);    // log the steps
                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {   // if it failed to retrieve then log a warning
                            Log.w(TAG, msg: "Could not get the step count", e);
                        }
                    });
}
```

Algorithm

Justification

I added a separate updateSteps method to the StepCounter class to handle reading the steps from the server. The method firstly creates a HistoryClient using the last signed in google account. Using the client, it retrieves the total daily. I add two listeners, one if the reading was successful and one to handle if it failed. For now, if it succeeds then the total is read, and I print that to the log. The total is read from the dataset that is retrieved from the API. I use the conditional operator (? :) to first check if the dataSet is empty and if it is then set the total to 0. If it's not empty then I get the value of the step's field and convert it to an integer. I use a conditional operator instead of an if else statement because it makes the code much clearer to read.

For now, I will only output the total steps to the log so I can check the system is working before implementing a system to update the MainActivity.

Validation

The OnFailureListener is executed when there is an error. For now, the exception is sent to the log using the warning label. I do not retrieve any data from the client if it failed to connect, otherwise the steps are accessed.

```
./com.faizan.onefitness I/MyFitness-MainActivity: Requested permission
./com.faizan.onefitness I/MyFitness-StepCounter: Successfully subscribed
./com.faizan.onefitness I/MyFitness-StepCounter: Total steps: 0
```

Run

Validation Testing

Running the app printed the following log. The app has run exactly how I expected. However, it was run on an emulator so there was no real data which is why it read 0. Log message from our validation listeners return success!

Run

```
com.faizan.onefitness I/MyFitness-StepCounter: Successfully subscribed
com.faizan.onefitness I/MyFitness-StepCounter: Total steps: 1722
```

I ran the app again on a test device. It ran exactly as expected and reported the total steps for the day.

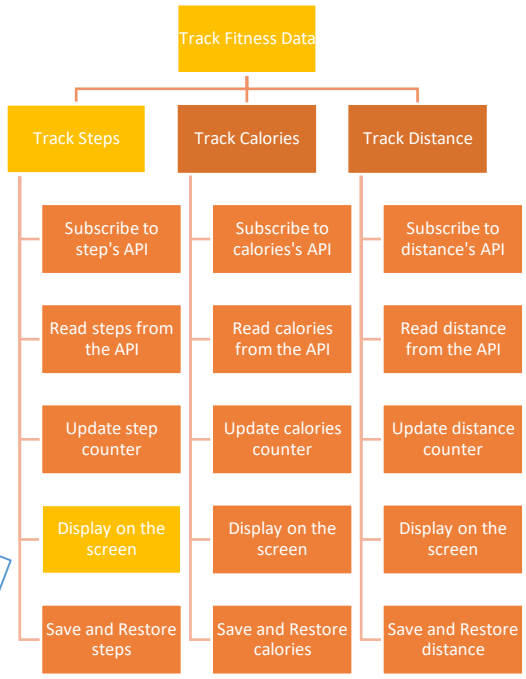
After permission was granted, no errors were occurred, and the log showed the app ran successfully. Next step is to update the UI!

Justificatio

I then compared the total steps to Google's official Fitness app to compare the step count. The steps count was the same so I can conclude tracking steps is both working and is accurate!

Update UI

My next step, after subscribing to recording the steps and retrieving the steps is to update the UI. It is much safer to handle all the UI updates in the MainActivity which is the enclosure to the screen. For this reason, I am going to send all the data back to the MainActivity, which will then update the UI. Keeping the UI in one place will also make is easier to modify in the future instead of having to search of the line/s in multiple classes.



OnReadStepsInterface

```
package com.faizan.onefitness;

public interface OnReadStepsInterface {
    // declaration, needs to be defined by MainActivity
    public void sendSteps(long total);
}
```

Justification

There are many different types of way to send data from the StepCounter class to the MainActivity. Many of these methods, including directly has their advantages and disadvantages. The most appropriate one for me to use is a java interface class. I don't want to use direct because firstly sending data directly means I have to make public methods in the MainActivity which means there will be more names for me to handle in that namespace. Secondly calling direct methods is sometimes not possible, including in our case where the update method will be called from a sub-class. Interfaces is also a type of **polymorphism** in Java. There allow many classes to have the same interface.

Another advantage of an interface is that StepCounter does not have to worry about how MainActivity implements the interface. The interface can be used to send the data to other classes such as a web interface in the future. This means the StepCounter does not have to be changed as much.

For now, the interface has one declaration for a sendSteps method which accepts the total using a long integer.

Send to UI – MainAcitivity

```
private TextView stepsCountText;   // used to update steps counter on UI
```

Create a private member variable to be a reference to the UI TextView element.

```
stepsCountText = (TextView) findViewById(R.id.stepsCnt);    // assigned UI element stepsCnt
```

In the onCreate() method, I initialise the variable. I pass the id that I signed the UI element earlier to the findViewById method which retrieves the TextView. The cast to the TextView is redundant but I put it there to make it clear that the object is a TextView.

```
@Override
public void sendSteps(long total) {
    stepsCountText.setText(String.valueOf(total)); // update UI element with total
}
```

I then implement the method in the MainActivity. It's best to read the line from right to left. First I convert the total to a string using the String::valueOf() Method. I then pass that to the setText method on the stepsCountText.

Send Steps to MainActivity – StepCounter

```
private OnReadStepsInterface updateCntInterface;     // used to send update to the UI
```

Private interface variable will be used to make the send call.

```
StepCounter(MainActivity activity) {
    this.context = activity; // initialise context
    this.updateCntInterface = activity; // initialise interface
}
```

I made a slight change to the constructor. Instead of the passed in variable being of type Context, I changed it to MainActivity. I did this so I can initialise both the context and the interface from the MainActivity reference. This means I do not have to pass two variables to the constructor and can leave the MainActivity call as is.

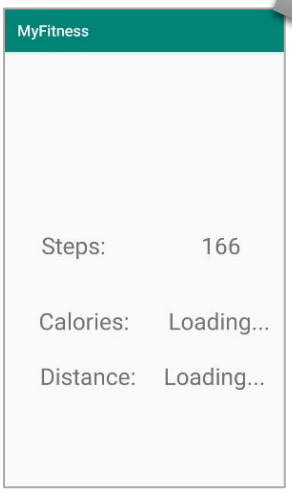
```
(OnSuccessListener) (dataSet) → {     // get the total on success
        long total = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_STEPS).asInt();

        updateCntInterface.sendSteps(total);        // send total steps to MainActivity, to display
        Log.i(TAG,  msg: "Total steps: " + total);     // log the steps
})
```

Finally, I added the highlighted line to the OnSuccessListener in the updateSteps method. This line calls passes the total to the sendSteps method I declared earlier in the interface. The updateCntInterface is initialised from the MainActivity so the MainActivity::sendSteps method will be executed – updating the counter on the UI.

Validation

I make sure the data is only retrieved when the API returns data that is valid to work with. I implement a successListener which will be called when there are no exceptions in returning the data. At this point I can assume the data is valid and retrieve the steps from the data returned from the API call.

Run



Successful! Running the app updates the steps counter as expected.

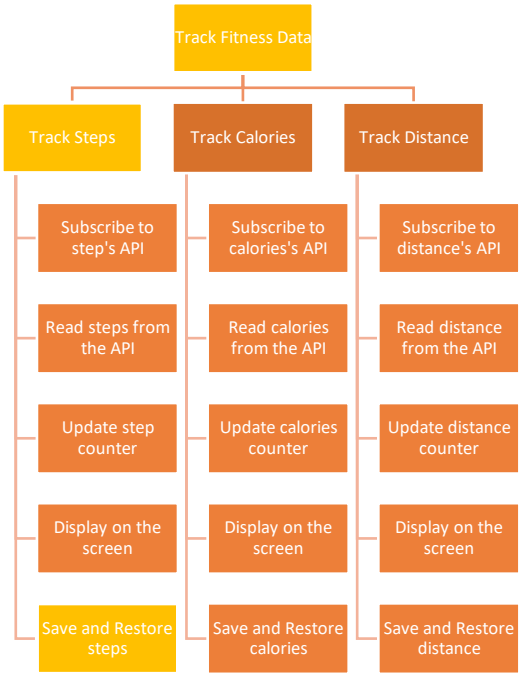
Validation Testing

As I can see the steps are updated only when a valid value is returned. The validation for the data has been successful as the app has no crashes when retrieving them.

Save and Restore Steps

Earlier I skipped the updating and saving the step counter. I first wanted to make sure all the required methods to update the UI were implemented before trying to set them on start-up.

There are many different methods of saving the values of the variables which need to be reloaded after a restart. I could use the OnInstanceSave() method to save the variables data before the app exits but that method is mostly used for managing activities. It also means I have to store the latest total of the steps as a member variable. Using SharedPreference is the recommended method and will also mean I can save it whenever the UI is updated.



Retrieve Steps – MainActivity

```
private static final String STEPS_TAG = "STEPS";
```

```
SharedPreferences sharedPrefs;      // used to retrieve stored variables
```

```
// initialise sharedPrefs using my apps name and using the private context
sharedPrefs = getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);
// retrieve the steps value, if it doesn't exist get -1
long steps = sharedPrefs.getLong(STEPS_TAG,  l: -1);
if (steps != -1) { // only set the value if it isn't saved
    stepsCountText.setText(String.valueOf(steps));
}
```

Create a SharedPreferenece member variable and initialise it for the app using private mode. This means other app's and activities can't retrieve the stored values. Then use the getLong method and the STEPS_TAG value to get the saved value. If there is no previous saved value then set steps to -1. If the value of steps variable is not -1 then set the UI TextView element to the previously saved value. This **validation** makes sure the value exists and is not set to -1 if there is no value saved.

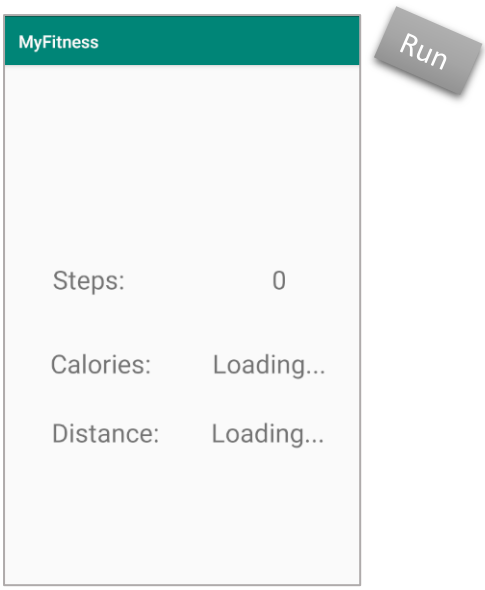
Validation

As the value is set to -1 if no previous value can be loaded I make sure that a invalid data is not shown to the user or a crash from appearing by using an if statement. If a value is returned it will be positive and not -1.

Save Steps – MainActivity::sendSteps

```
@Override
public void sendSteps(long totalSteps) {
    stepsCountText.setText(String.valueOf(totalSteps)); // update UI element with total
    sharedPrefs.edit().putLong(STEPS_TAG, totalSteps).apply(); // save the steps
}
```

The sendSteps method is updated with the following line so the total steps are saved and stored at the end. First I call the edit method and use the putLong method to save the total steps using the STEPS_TAG. Finally the apply() method is called so our changes are saved.



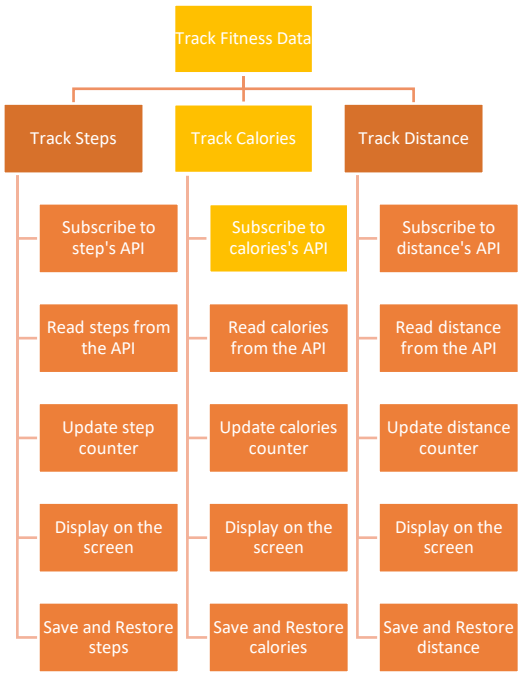
Running the app for the second time restores the 0 on the UI without the Loading… ever appearing. This suggests the app worked as expected, updating the UI on launch instead of waiting for the calls to the API to be successful!

Validation Testing

As the value 0 is displayed this means an invalid value has not been returned from the cache of the app. The validation has worked correctly and an invalid int value has not been assigned to the steps value which would cause a crash.

To summarise, the app first creates and initialises the StepCounter and UI element for the counter. Then it checks for permissions, if the permissions are already granted then the subscribe() method is executed. If those permissions were not granted from a previous session, then the google sign-in activity is launched where it requests those permissions. After permissions have been handled, the subscribe() method is executed. If the subscribe method fails then it is logged. However if it succeeds then the updateSteps() method is called. That method gets the daily steps. If it succeeds, then an interface is used to send the steps to the MainActivity where the TextView is updated. If it fails then it's logged.

Module 2 – Track Calories



My next step is to record and calculate the calories. For now, I want to handle the calories for the steps only. In future modules when I add running and cycling, I will have to handle the calories for that as well.

Before continuing I need to request extra permission from the Android system to record the calories that are used [1].

Justification

I already used interfaces which is a type of polymorphism, but I am going to create an abstract class instead. I am going to derive classes to track steps, calories and distance. Using an abstract class will allow me to keep a similar interface for these similar classes. I could use an interface but I need to have member variables which cannot be declared in an interface.

[1]https://developers.google.com/android/reference/com/google/android/gms/fitness/data/DataType.html#TYPE_CALORIES_EXPENDED

Restructuring

```
<uses-permission android:name="android.permission.ACTIVITY_RECOGNITION" />
```

First, I request permission to activity recognition. This allows my app to recognise what activity the user is performing such as running, cycling and driving.

```
// type of data we want to request
FitnessOptions fitnessOptions = FitnessOptions.builder().addDataType(DataType.TYPE_STEP_COUNT_CUMULATIVE)
                                                        .addDataType(DataType.TYPE_STEP_COUNT_DELTA)
                                                        .addDataType(DataType.TYPE_CALORIES_EXPENDED)
                                                        .build();
```

Next I add the following line to the FitnessOptions so I can request the relevant permissions to record and retrieve the calories used by the user.

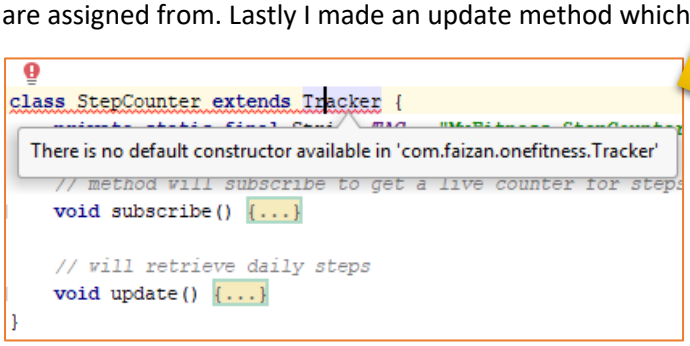
MainActivityInterface

I also renamed the OnReadStepsInterface to MainActivityInterface because I will be using the interface to send other data to the MainActivity, not just the steps.

Abstract Class - Tracker

```
package com.faizan.onefitness;

import android.content.Context;

public abstract class Tracker {
    protected Context context;    // get parent activities context to pass into methods
    protected MainActivityInterface updateUIInterface;    // used to send update to the UI

    public Tracker(MainActivity activity) {
        this.context = activity; // initialise context
        this.updateUIInterface = activity; // initialise interface
    }

    abstract void update();
}
```

I created a new abstract class called Tracker. I created two member variables to hold the context and the interface to update the UI in MainActivity. These variables are protected so they can be accessed by sub-classes which inherit the class but not by other classes. Next I created a constructor like the StepCounter one where the context and interface are assigned from. Lastly I made an update method which is declared abstract so it must be inherited and defined.




```
class StepCounter extends Tracker {
    // method will subscribe to get a live counter for steps
    void subscribe() {...}

    // will retrieve daily steps
    void update() {...}
}
```

I inherited the Tracker class from StepCounter and renamed updateSteps to update. However I am now getting an error that there is no default constructor in the Tracker class.

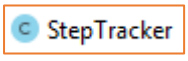
Researching online it seems this is because Java is declaring a default constructor for StepCounter which calls the default Tracker constructor but that does not exist.

Fix

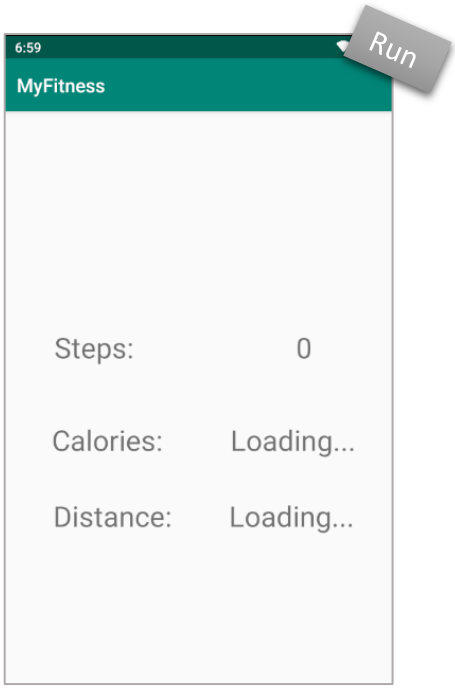
```
public StepCounter(MainActivity activity) {
    super(activity);
}
```

Justification

To fix this issue I explicitly defined a constructor that takes MainActivity and passes it to the Tracker constructor. This resolves the issue because Java no longer declares a default empty constructor that tries to call the default Tracker constructor which doesn't exist. This also means StepCounter cannot be used without passing MainActivity.



It was also appropriate I rename StepCounter to StepTracker as the name is more appropriate. Android Studio's refactor handles all the renaming, including in files.



I ran the app in the emulator just to confirm there were no errors.

Subscribe Method

CalorieTracker Class

```
abstract void subscribe();
abstract void update();
```

First I declared the subscribe() method in the abstract class, Tracker. I had forgotten to do this earlier. This will force any sub-class to define that method.

```
public class CalorieTracker extends Tracker {
    private static final String TAG = "MyFitness-CalorieTrackr";  // used for debugging

    public CalorieTracker(MainActivity activity) {
        super(activity);
    }

    void subscribe() {

    }

    void update() {

    }
}
```

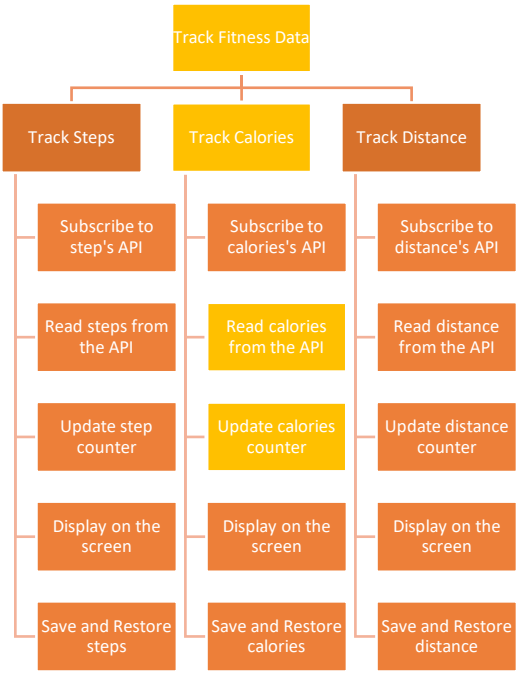
Next I crated a CalorieTracker class which will inherit the Tracker class, this **reduced repeated code** as I use **classes** along with **polymorphism** to execute the same code as other classes. I then defined the constructor, subscribe and update method.

Algorithm

Subscribe Method

```
void subscribe() {
    Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context))   // register a client with the API
            .subscribe(DataType.TYPE_CALORIES_EXPENDED)                                  // subscribe to record calories throughout the day
            .addOnCompleteListener(
                    new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                Log.i(TAG, msg: "Successfully subscribed to calories");            // has completed successfully, so log message
                                update();                                                          // get daily total steps after success
                            } else {
                                Log.w(TAG, msg: "Problem subscribing to calories", task.getException()); // has failed, so log message
                            }
                        }
                    });
}
```

The subscribe method is exactly the same as the one from StepCounter apart from one line. I can re-write the code to remove this repeating of code but for now I will use a separate subscribe method to test the application out before removing any duplicated code.



Read Daily Calories

Next step is to read the calories from the API for the day. I need to make a request similar to the StepTracker::update() method and handle the different type of result received.

I am going to use TYPE_CALORIE_EXPENDED[1] to retrieve the calories used by the user. Later when adding cycing and running, I will have to handle those calories separately.

[1] TYPE_CALORIE EXPENDED – displayed in kcals

Update - TrackCalories

Algorithm

```
// will retrieve daily calories expended
void update() {
    Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context))// register a HistoryClient with account
            .readDailyTotal(DataType.TYPE_CALORIES_EXPENDED)      // read daily calories using CALORIES_EXPENDED
            .addOnSuccessListener(                          // listener executed when method succeeds
                    new OnSuccessListener<DataSet>() {
                        @Override
                        public void onSuccess(DataSet dataSet) {    // get the total on success
                            Float total = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).asFloat();

                            Log.i(TAG,  msg: "Total calories: " + total);    // log the calories
                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {   // if it failed to retrieve then log a warning
                            Log.w(TAG,  msg: "Could not get the calorie count", e);
                        }
                    });
}
```

The update method is also very similar so some of the code can likely be used. For now, I will just use the update method. The first highlighted line retrieves the DataSet for the calories used. The second highlighted line has many changes. Firstly the returned type of FIELD_CALORIES is a float so I changed the type to float. For testing purposes, I just logged the total instead of updating the UI.

Validation

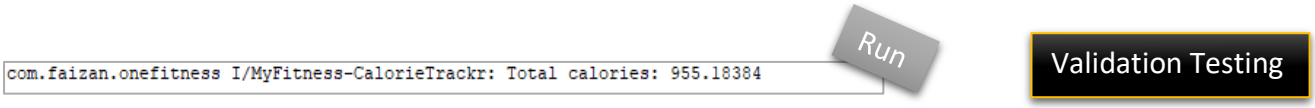
The success and failure listeners are added to validate data before it is used by the main application like the steps class and tracker class. I will perform the testing later by calling it from the MainActivity.

```
private CalorieTracker calorieTracker;
```

```
stepTracker = new StepTracker( activity: this);                // create an object of stepTracker
calorieTracker = new CalorieTracker( activity: this);          // and CalorieTracker
```

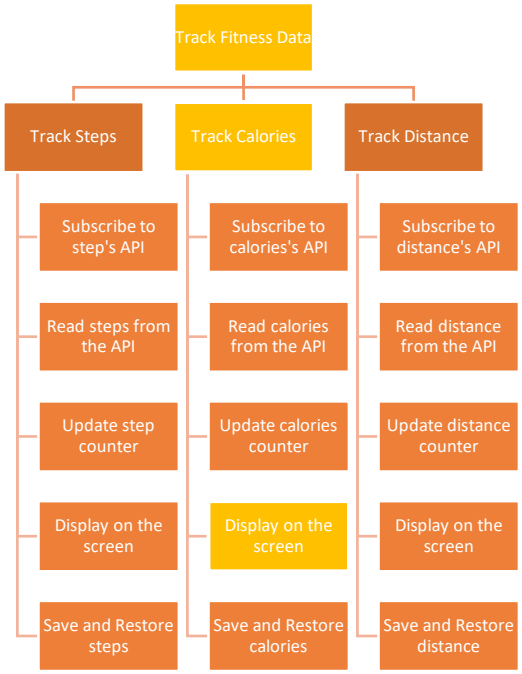
```
stepTracker.subscribe();            // subscribe to steps tracking
calorieTracker.subscribe();         // and calorie tracking
```

I declare a CalorieTracker object and then initialise it in the onCreate() method. StepTracker and CalorieTracker inherit the same abstract class and therefore have the same interface so I call the same method on CalorieTracker that is called on StepTracker.



Running the app shows the correct number of calories for the day. Executing the correct listener!

Update UI



I will be taking a similar approach to updating the UI in the MainActivity so UI related methods are kept together. This will also make the code easier to read as it's persistent.

I'm going to add to the MainActivity interface to have the code more readable and easier to modify in the future.

Send to UI - MainActivity

```
public void sendCalories(float totalCalories);
```

First, I declared the sendCalories method in MainActivityInterface, similar to sendSteps. It takes a float and returs nothing.

```
private TextView stepsCountText;    // used to update steps counter on UI
private TextView caloriesCountText; // used to update calories counter on UI
```

I declared a private TextView to point at the UI element which I are going to update with the calorie count.

```
caloriesCountText = (TextView) findViewById(R.id.caloriesCnt); // assign UI element caloriesCnt
```

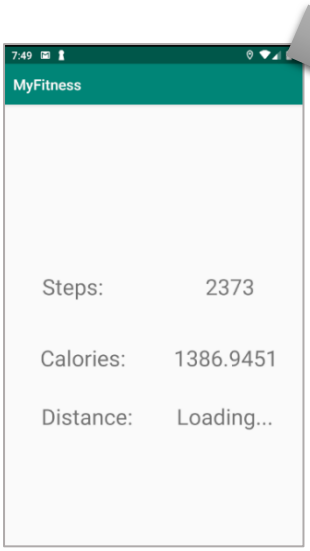
I then initialise the variable with the UI element by using the findViewByID method along with the caloriesCnt ID.

```
@Override
public void sendCalories(float totalCalories) {
    caloriesCountText.setText(String.valueOf(totalCalories)); // update UI element with total
}
```

I can define the sendCalories method in MainActivity to update the UI element using the setText method. I also use the String.valueOf method to convert the float into a string which can be displayed.

```
updateUIInterface.sendCalories(total);
```

Finally the method is called from CalorieTracker::update() with the daily calories to update the UI.



Running the app works as expected however there is one issue. The number for the calories is quite large. I have two options I can either truncate the number or round it to the next whole number.

Fix UI output

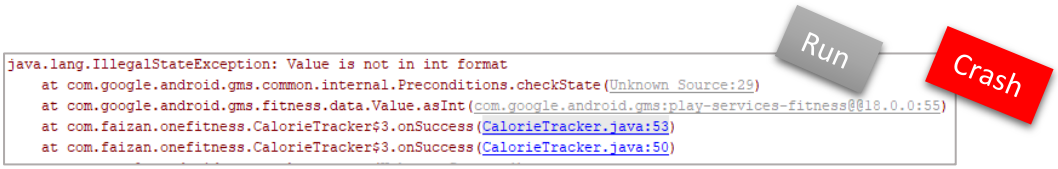
```
int total = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).asInt();
```

Instead of returning the value as a float, I changed it to return an int using the asInt() method. I also changed the type of total to int.

```
public void sendCalories(int totalCalories);
```

```
@Override
public void sendCalories(int totalCalories) {
    caloriesCountText.setText(String.valueOf(totalCalories)); // update UI element with total
```

I then changed anywhere a float was used to an int so there are no unnecessary conversions.



When running the app, it caused a crash. Looking at the crash log I saw the error was with the value returned by FIELD_CALORIES. The value was a float and could not be returned as an int.

Justification

With some research I found that I could either work with the float and then discard the decimal point using String.format or I could cast the float to an int. From my research I found that casting to an int was about 10x faster so that will be the method I will use. [1]

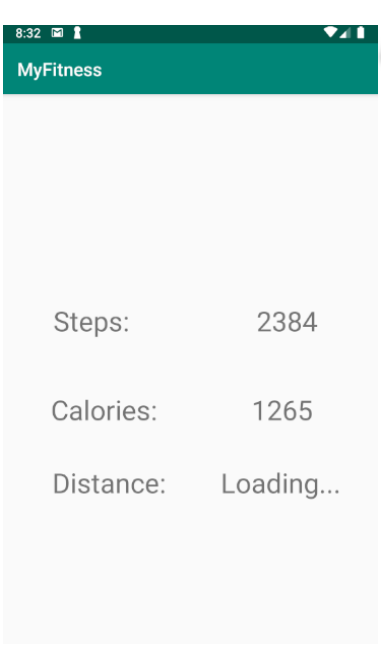
[1] https://www.baeldung.com/java-double-to-string

Fix

```
// get the float value and then cast it to an int
int total = dataSet.isEmpty() ? 0 : (int) dataSet.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).asFloat();
```

Justification

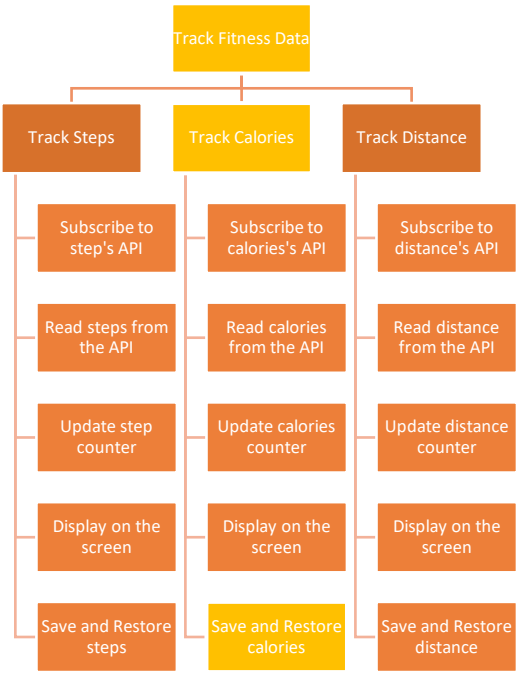
I changed it back to return a float using asFloat() method at the end but before assigning the value I added (int) to cast the float value to an int before it is assigned to total. This does truncate the value instead of rounding it like String.format() would have done but these values are already an **estimate**, so it makes no difference here.



Run

Running the app this time worked perfectly. The decimal point is removed by truncating the float and displayed. This looks much cleaner than the value before, making the app more user friendly.

Save and Restore Calories



To finish off the calorie's module, I'm going to implement saving the calories so it can restore quickly when the app is launched again.

As I did earlier with the steps, I am going to use SharedPreferences to save the variables to keep the app consistent and reuse code. As discussed earlier it is the recommended and most widely used method as well.

Retrieve Calories – MainActivity

```
private static final String CALORIES_TAG = "CALORIES";
```

Create a static final which is a constant to just hold a tag. This makes sure calories are not saved and retrieved using different tags – meaning different values are not retrieved (like 0).

```
// retrieve the values, if they don't exist get -1
long steps = sharedPrefs.getLong(STEPS_TAG, -1);
int cals = sharedPrefs.getInt(CALORIES_TAG, -1);
```

I then used the already defined sharedPrefs to get the value of the saved calories count. If no value has previously been saved then -1 is returned which is assigned to cals.

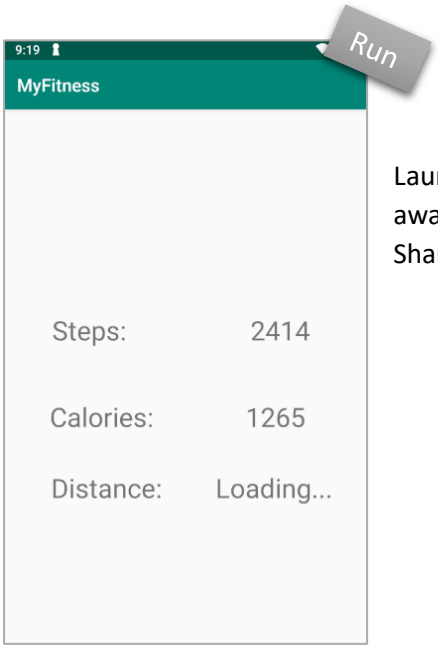
```
if (cals != -1) { // only set the value if it is saved
    caloriesCountText.setText(String.valueOf(cals));
}
```

If cals is not -1, meaning if there was a value previously saved then I set the UI text to that value using setText method. This makes sure I don't set -1 and the UI shows "Loading…" on first launch.

Save Calories – MainActivity

```
@Override
public void sendCalories(int totalCalories) {
    caloriesCountText.setText(String.valueOf(totalCalories)); // update UI element with total
    sharedPrefs.edit().putInt(CALORIES_TAG, totalCalories).apply(); // save the calories
}
```

The sendCalories method is also updated so when the UI is updated, the new calories count is also saved in SharedPreferences using the putInt() method. I use the CALORIES_TAG here like before so there is no syntax error that would be hard to debug. Finally the apply() method is called to write the data to storage.



Launching the app for the second time, the calorie and steps are shown straight away without any delay meaning they have been restored from the saved SharedPreferences before being updated by other tasks.

Optimise code

Before continuing I want to **remove some redundant code** and replace it wither better, **more reusable modules**. This is one of the advantages of **OOP**. It makes **reading** and **adding** code to it **easier**. I will be adding tracking distance after so it's best I make the reusable code now before having too much to handle.

Subscribe Method – Tracker

```
private RecordingClient recordClient; // used to subscribe to record data
```

In the Tracker abstract class I created a member variable to hold the recording client so it doesn't have to repeatedly be retrieved which takes time. The variable is private because I will only be accessing it from the Tracker class and not any sub-classes.

```
// initialise the recordingClient and historyClient using the context and last signed in account
recordClient = Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context));
```

The recordClient is then initialised in the only constructor for the Tracker class. It is initialised using context and the last signed in google account just as it was before.

Algorithm

```
protected void subscribeToRecordingClient(final DataType dataType, final String TAG) {
    recordClient.subscribe(dataType)                          // subscribe to record data type throughout the day
            .addOnCompleteListener(
                    new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                Log.i(TAG,  msg: "Successfully subscribed!");          // has completed successfully, so log message
                                update();                                             // call the update method to retrieve daily data
                            } else {
                                Log.w(TAG,  msg: "Problem subscribing", task.getException()); // has failed, so log message
                            }
                        }
                    });
}
```

Justification

I created a new protected method which will be called from the subscribe methods in the sub-classes. It takes a DataType and a String to hold the TAG for debugging. These two variables are final because they will not be changed. TAG must be final because it is used in onComplete() method which is run as a separate task so it should not be changed in that time or it will cause a crash.

Validation

Again, I add a listener to make sure I have successfully subscribed before retrieving the calories. There is now only 1 set of listeners instead of separate ones in each class **reducing repeated code** by using **class inheritance.**

```
// method will subscribe to get a live counter for steps, the subscription will run even when the app is stopped
void subscribe() {
    subscribeToRecordingClient(DataType.TYPE_STEP_COUNT_CUMULATIVE, TAG);
}
```

```
// method will subscribe to get a live counter for calories, the subscription will run even when the app is stopped
void subscribe() {
    subscribeToRecordingClient(DataType.TYPE_CALORIES_EXPENDED, TAG);
}
```

The subscribe methods from StepTracker and CalorieTracker now call the subscribeToRecordingClient method from the Tracker class with the relevant DataType and TAG.

Justification

One question that arises now is why not just call the recordingClient with these two DataTypes in the Tracker class. The reason I do this is because the subscriptions start a separate task from the main thread of the app. If they were called from the Tracker class then the calories will only be updated after the steps have been updated. This would slow the program down. Instead I use separate classes so 2 separate tasks(similar to threads) are started to executed and retrieve the two pieces of data separately, not relying for the other to complete first.

Update Method – Tracker

```
protected HistoryClient historyClient; // used to get daily data
```

In the tracker class I created a member variable to hold a HistoryClient so it doesn't have to be repeatedly retrieved in sub-classes. This variable is protected so it can be accessed by subclasses.

```
// initialise the recordingClient and historyClient using the context and last signed in account
recordClient = Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context));
historyClient = Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context));
```

In the constructor I then initialise the historyClient to retrieve the fitness history client, which uses the last signed in google account.

Justification

While the update() methods look quite similar in our StepTracker and CalorieTracker classes, they cannot be reused like the subscribe method because they return results that I handle quite differently. The update methods start tasks(threads) which return different results for both the steps and calories of different types. If I want to handle these results then new methods will have to be defined which would make the code less readable.

```
historyClient.readDailyTotal(DataType.TYPE_STEP_COUNT_DELTA)     // read daily steps using COUNT_DELTA
```

```
historyClient.readDailyTotal(DataType.TYPE_CALORIES_EXPENDED)     // read daily calories using CALORIES_EXPENDED
```

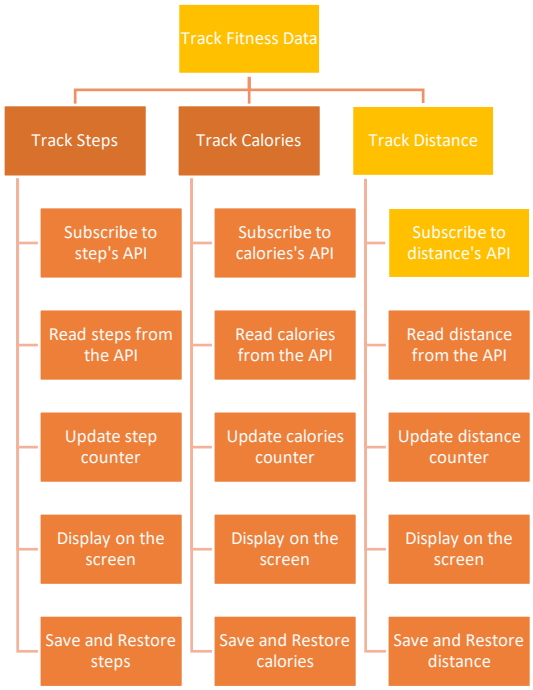
The only change I made to the update methods are using the historyClient which I initialised in the Tracker class.

```
restoreValues(); // restore previously saved values for UI
```

```
private void restoreValues() {
    // initialise sharedPrefs using my apps name and using the private context
    sharedPrefs = getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);

    // retrieve the values, if they don't exist get -1
    long steps = sharedPrefs.getLong(STEPS_TAG,  i: -1);
    int cals = sharedPrefs.getInt(CALORIES_TAG,  i: -1);
    if (steps != -1) { // only set the value if it is saved
        stepsCountText.setText(String.valueOf(steps));
    }
    if (cals != -1) { // only set the value if it is saved
        caloriesCountText.setText(String.valueOf(cals));
    }
}
```

I also grouped some code together to make the onCreate method easier to read and not as bloated

Module 3 – Track Distance



Justification

I're going to go through a similar process for the distance as tracking the steps. Because I optimised our subscribe method earlier, I can just call that method with our data type.

Following the reference, I will be interested in com.google.distance.delta data type. It looks like this will be returning a float like calories, so I need to handle the data similarly.

Permissions

```
// type of data we want to request
FitnessOptions fitnessOptions = FitnessOptions.builder().addDataType(DataType.TYPE_STEP_COUNT_CUMULATIVE)
                                                        .addDataType(DataType.TYPE_STEP_COUNT_DELTA)
                                                        .addDataType(DataType.TYPE_CALORIES_EXPENDED)
                                                        .addDataType(DataType.TYPE_DISTANCE_CUMULATIVE)
                                                        .addDataType(DataType.TYPE_DISTANCE_DELTA)
                                                        .build();
```

Firstly in the MainActivity::onCreate method, I add the DISTANCE_CUMULATIVE and DISTANCE_DELTA data so I can request permission from the user for that as well.

```
private static final int REQUEST_CODE_TRACKER = 4444;
```

```
private static final String TAG = "MyFit-CalorieTracker";  // used for debugging
```

I also renamed REQUEST_CODE_STEP_CODE to REQUEST_CODE_TRACKER because it is more accurate.

I changed the TAG prefix from MyFitness to MyFit because it cannot be more than 20 characters.

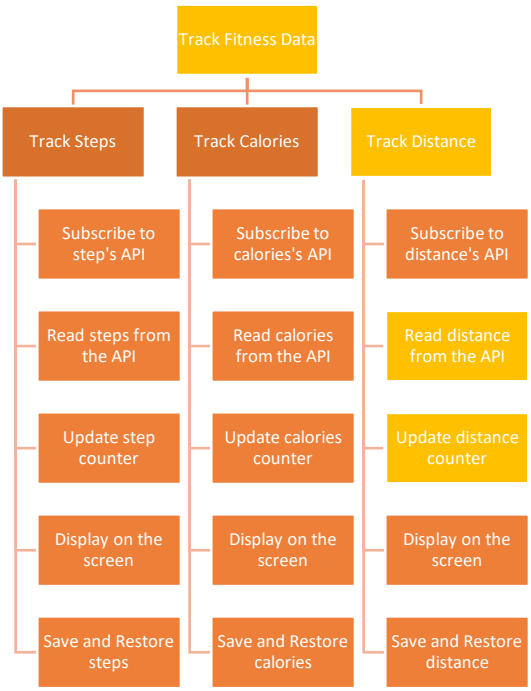
Subscribe to Recording – DistanceTracker

```
public class DistanceTracker extends Tracker {
    private static final String TAG = "MyFit-DistanceTracker";  // used for debugging

    public DistanceTracker(MainActivity activity) {
        super(activity); // initialise context and interface
    }

    // method will subscribe to get a live counter for distance, the subscription will run even when the app is stopped
    void subscribe() {
        subscribeToRecordingClient(DataType.TYPE_DISTANCE_CUMULATIVE, TAG);
    }

}
```

I created a class named DistanceTracker which is a sub-class of Tracker. It has a TAG for debugging log lines. The constructor is simple and used to stop Java generating an implicit constructor which would cause errors because Tracker does not have a no parameter constructor. This constructor just passes the activity to the super constructor.

The subscribe method is very simple as well due the reusable method I made earlier. It calles the subscribeToRecordingClient with the data type to record all the distance travelled by the user throughout the day. That method as I seen previously starts a task to record all data for distance in the background even when the app is not running.

Read Daily Distance



Next step after subscribing to the API is to read the total daily distance travelled by the user. I'll do this in the update method similar to the other two classes and use the HistoryClient::readDailyTotal method.

Update method - DistanceTracker

Algorithm

```
// will retrieve daily calories expended
void update() {
    historyClient.readDailyTotal(DataType.TYPE_DISTANCE_DELTA)      // read daily calories using DISTANCE_DELTA
            .addOnSuccessListener(                                  // listener executed when method succeeds
                    new OnSuccessListener<DataSet>() {
                        @Override
                        public void onSuccess(DataSet dataSet) {    // get the total distance on success
                            // get the float value and then cast it to an int
                            int total = dataSet.isEmpty() ? 0 : (int) dataSet.getDataPoints().get(0).getValue(Field.FIELD_DISTANCE).asFloat();

                            Log.i(TAG, msg: "Total distance: " + total);    // log the calories
                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {   // if it failed to retrieve then log a warning
                            Log.w(TAG, msg: "Could not get the distance count", e);
                        }
                    });
}
```

The update method is similar to that of the calorie's and step's one. I use the historyClient from the super-class(Tracker) to call readDailyTotal. It retrieves the daily distance travelled by the user. The returned value is a float similar to the calorie one. I aren't interested in the precise reading so I just truncate it, after all it is an estimate.

Validation

For now, the distance is logged so I can test the app. I will also send the distance to the activity to update the view as I have validated the data to be correct here and not out of range/

Create object - MainActivity

```
private StepTracker stepTracker;    // used to make subscription and read steps
private CalorieTracker calorieTracker; // and calories
private DistanceTracker distanceTracker; // and distance
```

```
stepTracker = new StepTracker( activity: this);                // create an object of StepTracker
calorieTracker = new CalorieTracker( activity: this);          // and CalorieTracker
distanceTracker = new DistanceTracker( activity: this);        // and DistanceTracker
```

```
stepTracker.subscribe();            // subscribe to steps tracking
calorieTracker.subscribe();         // and calorie tracking
distanceTracker.subscribe();        // and distance tracking
```

```
stepTracker.subscribe();            // then subscribe to the step counter
calorieTracker.subscribe();         // and calorie tracking
distanceTracker.subscribe();        // and distance trackingXX
```

Finally, to have the code run, I repeated the same lines that I had for StepTracker and CalorieTracker in MainActivity. This includes creating an object of that type, initialising it in onCreate and calling the its subscribe() method in onCreate and onActivityResult.

Error

```
W/MyFit-DistanceTracker: Problem subscribing
SecurityException: com.google.distance.cumulative requires android.permission.ACCESS_FINE_LOCATION
```

After running the app a run-time error occurred. The app did not crash but there was an error subscribing to record distance. Looking at the exception error, it seems I do not have access to fine-location which is required by the app to record distance as it gives the app a more precise location.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACTIVITY_RECOGNITION" />
```

Looking at the permission, it seems I had already granted ACCESS_FINE_LOCATION. I believe I may have missed something or there was an error granting the permissions.

```
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.faizan.onefitness/com.faizan.onefitness.MainActivity}: java.lang.NullPointerException: null reference
```

```
at com.faizan.onefitness.Tracker.<init>(Tracker.java:30)
at com.faizan.onefitness.StepTracker.<init>(StepTracker.java:26)
at com.faizan.onefitness.MainActivity.onCreate(MainActivity.java:56)
```

Crash

I uninstalled the app and re-installed it as that may solve the problem however that caused the app to completely crash.

```
historyClient = Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context));
```

Looking at the crash log it seems this doesn't have anything to do with our DistanceTracker. The issue seems to be in the Tracker constructor. Looking at the code again I quickly realise I tried to get the HistoryClient without first allowing the user to sign in.

Fix

```
protected void subscribeToRecordingClient(final DataType dataType, final String TAG) {
    Fitness.getRecordingClient(context, GoogleSignIn.getLastSignedInAccount(context))
```

```
void update() {
    Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context))
```

- in all tracker classes

Unfortunately, I couldn't reuse the HistoryClient and RecordingClient like I had believed. Each task must retrieve it separately. So I went and removed historyClient and recording member variables form the Tracker class and changed the above lines to retrieve separate clients.

```
W/MyFit-DistanceTracker: Problem subscribing
SecurityException: com.google.distance.cumulative requires android.permission.ACCESS_FINE_LOCATION
```

The program now launches with the same error as before

Justification

With some research [1] it turns out I need to explicitly ask for GPS permission from the user. It's best I use the standard and recommended method of requesting permissions in the app [2]. For this I will need to create several methods in MainActivity. It's recommended to handle the permissions in the activity and not in any subclasses. This makes it easier to pass the activity as well allow UI changes. It also allows me to prevent other modules from being executed, not just distanceTracker if required in the future.

[1] https://stackoverflow.com/questions/46915957/access-fine-location-permissions

[2] https://developer.android.com/training/permissions/requesting#java

Requesting Permission – MainActivity Dialogs

```
private static final int MY_PERMISSIONS_REQUEST_LOCATION = 4541;
```

Member variable to hold a static value – a constant as a code.

Algorithm

```
private void subscribeToDistancePerms() {
    // check if we already have permission
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {

        // Permission is not granted
        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
                Manifest.permission.ACCESS_FINE_LOCATION)) {
            // handle using seperate dialgue
        } else {
            // No explanation needed; request the permission
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSIONS_REQUEST_LOCATION);
        }
    } else {
        // Permission has already been granted
        distanceTracker.subscribe();
    }
}
```

This method is called when I want to subscribe to retrieve the distance. Firstly, it checks if I do not have permissions for the device's fine location. If I do then just execute distanceTracker::subscribe method.

Justification

If I don't then I need to first check if I have been declined the permissions before (i.e. this isn't the first launch or permissions have been revoked). If I have been declined the permissions before then for now, I leave it blank but I am going to be adding a dialog to let the user know why I need the permissions and how to enable them through settings. If this is the first launch, then just simply ask the user for the permissions. I use the MY_PERMISSION_REQUEST_LOCATION so I can check if the permissions were granted or not.

Validation

I cannot assume that the permissions are always granted to see the user's location when the app starts up as they can be removed from the app through settings. For this reason I carry out a number of steps to check for the permission and grant it before updating the distance which requires the location to calculate. If I call the distance without this permission it will cause a crash

```
private static final int REQUEST_SETTINGS_CODE = 4712;
```

Another random int to handle the result when activity returns (the settings activity).

```
// create a dialogue to manually get permission from settings
private void manualPermissionDialog() {
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder( context: this);
    alertDialogBuilder.setTitle(R.string.manual_permission_title); // set dialogue title
    alertDialogBuilder
            .setMessage(R.string.manual_permission) // set the message
            .setCancelable(false)                   // can't be ignored
            .setPositiveButton( text: "SETTINGS", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {  // run when the SETTINGS button is clicked
                    Intent intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS); // create an intent to run settings
                    Uri uri = Uri.fromParts( scheme: "package", getPackageName(), fragment: null); // with this apps page open
                    intent.setData(uri);                                                    // set the uri(settings page)
                    startActivityForResult(intent, REQUEST_SETTINGS_CODE);   // start the activity and handle result
                }
            })
            .setNegativeButton( text: "LATER", new DialogInterface.OnClickListener() {    // if LATER button is clicked
                public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();                                                    // cancel the dialogue
                }
            });
    AlertDialog alertDialog = alertDialogBuilder.create(); // create the dialogue
    alertDialog.show();                                    // activate it
}
```

I use an AlerDialog.Builder to create a dialog to show the user how to enable permissions in the settings and why I need them. I set the title, message and make it non-cancellable so it cannot be ignored. Then I set the positive button to open the settings page for the app. The settings activity is started with REQUEST_SETTINGS_CODE so I can handle the result from it later. I want to check for the permissions again after the user has finished with the settings activity.

I also set a LATER button which just removes the dialog for now.

```
<string name="manual_permission_title">Change Permissions in Settings</string>
<string name="manual_permission">Go to Settings to grant permission to track distance.</string>
```

I also made two string values in the strings file which are used above to set the text for the dialog. This makes it easier to modify the UI in the future.

```
// called when the permission dialogue returns with a result
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_LOCATION:
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted
                distanceTracker.subscribe();
            } else {
                // permission was denied, ask to manually set it
                manualPermissionDialog();
            }
            break;
    }
}
```

This method is called when the user either grants or denies permission for the app when it is launched for the first time. This first confirms the requestCode with that sent by the permission dialog using the switch statement and the constant MY_PERMISSIONS_REQEUST_LOCATION. If they are the same then it checks whether the user granted or denied permission. If the user granted permission then distanceTracker subscribe method is called. If the user denies permission, then I manually ask for the permission like above.

```
// run after an activity closes
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQUEST_CODE_TRACKER:              // if the log-in activity finishes
            if (resultCode == Activity.RESULT_OK) {
                stepTracker.subscribe();            // then subscribe to the step counter
                calorieTracker.subscribe();         // and calorie tracking
                subscribeToDistancePerms();         // request location permission if it has not been granted
            }
            break;
        case REQUEST_SETTINGS_CODE:                 // if settings activity finishes
            subscribeToDistancePerms();              // check for the permissions again
            break;
    }
}
```

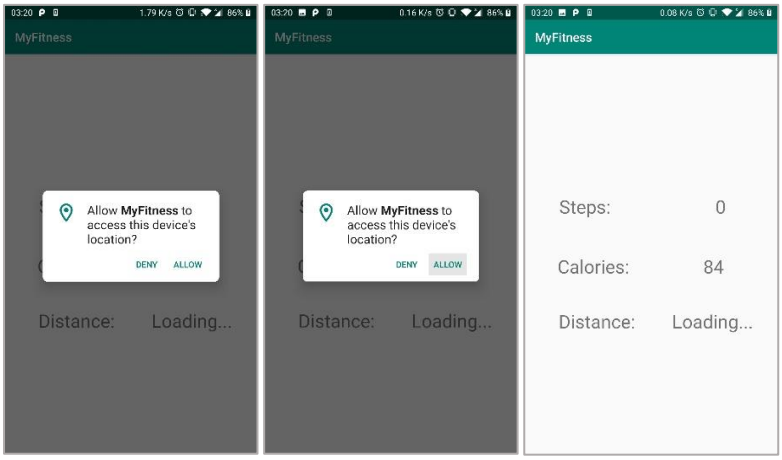
I re-wrote the onActivityResult method with a switch statement because I are testing an integer and can **easily add** more cases in the future instead of use many if statements. It also makes it easier to read the code

For now wanted to check whether permissions have been granted after the settings method had completed (i.e. the user had backed out of it). I don't check if the activity exited successfully as it will always be exited/cancelled by the user. This basically creates an endless loop to keep the manual dialog open until the user clicks later or grants permission.

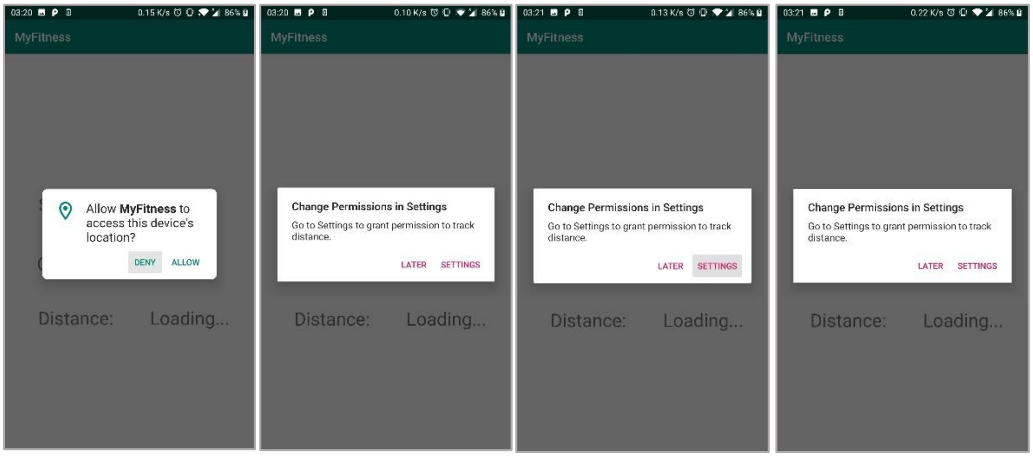
Muhammad Faizan Alam

Permission Testing

Before continuing I want to test the permission dialogs rigorously so there are no errors as I planned to in version 1 testing.

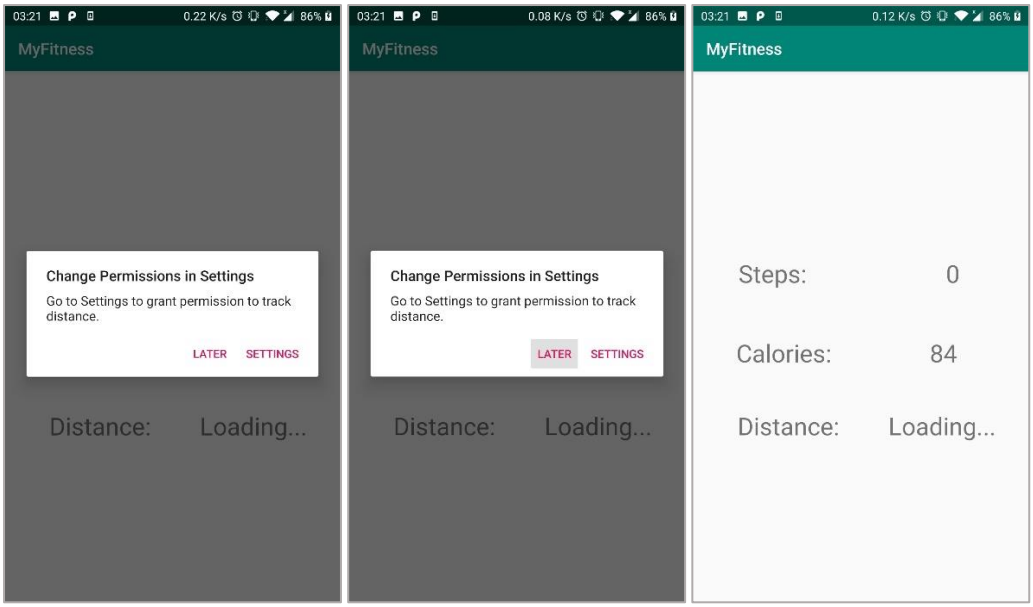


I ran the program after clearing storage and it asked for google sign in as well as permissions for location. I granted it and no other dialogues appeared or errors in the log.



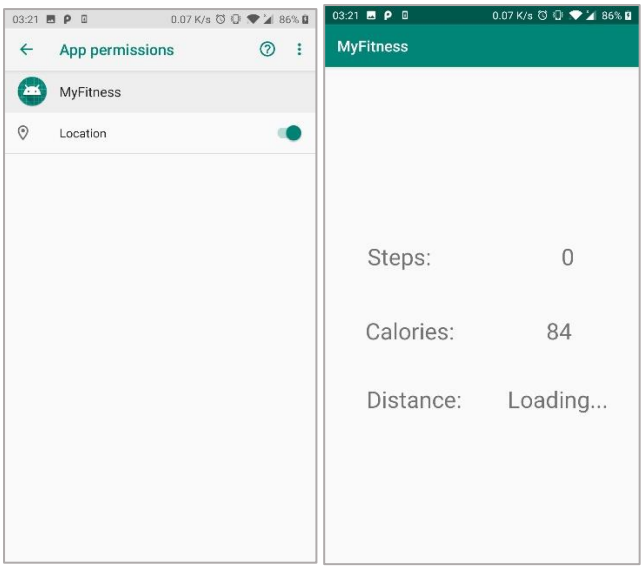
Validation Testing

Denying the permission popped up the manual dialog where I clicked settings and backed out without granting permissions. The dialogue appeared to have stayed but what happened was the permissions were checked and a new dialog was shown.



Validation Testing

Clicking the later button made the dialog disappear but did not execute the DistanceTracker subscribe method. I used validation to prevent that so the app would not crash when the user did something unexpected.



Lastly, I granted permissions manually and it also caused the manual dialog to disappear and not reappear which is exactly what I wanted!

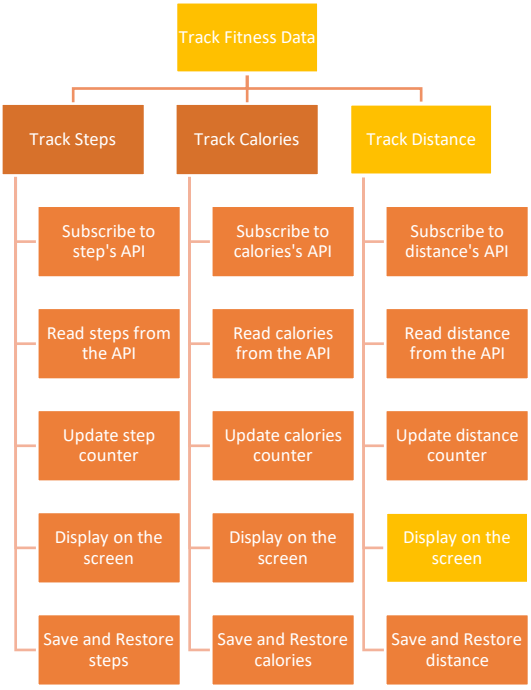
Validation Testing

I performed every user action possible with the permissions and everything was successful as I got the feedback and output, I had expected exactly.

Now the distance will only be updated when location permissions are granted, this will prevent a crash due to lack of permissions.

Test Use	Explanation	Actual Outcome	Works?
Request Location Permission	Permission for location should be requested if not granted. If the user denies then show the user how to manually grant the permissions from phone settings	Location permission are requested and if they are not granted, they are requested manually	Yes

Update UI



Now that I have read the data I need, I can update the UI and show the user their daily distance. I'll do this by adding to MainActivity interface and sending the data to the MainActivity so it can update it. I am also going to handle the UI when permissions are not granted for the location. All the UI updates will be done in the MainActivity.

Send to UI - MainActivity

```
public void sendDistance(int totalDistance);
```

I declared sendDistance method in MainActivityInterface. It takes an integer and is public so it can be called from DistanceTracker.

```
private TextView distanceCountText; // used to update distance counter on UI
```

I declared a private TextView to point to the UI element to update the distance count.

```
distanceCountText = (TextView) findViewById(R.id.distanceCnt); // assing UI element distanceCnt
```

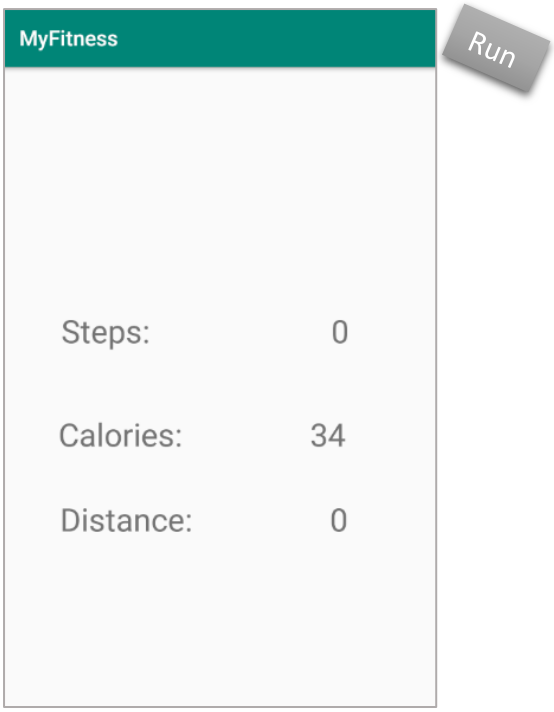
I then initialise it in onCreate() using the findViewByID and the id of the UI element.

```
@Override
public void sendDistance(int totalDistance) {
    distanceCountText.setText(String.valueOf(totalDistance)); // update UI element with total
}
```

Finally I defined the sendDistance method in MainActivity to set the UI element to the counter from DistanceTracker.

```
updateUIInterface.sendDistance(total);
```

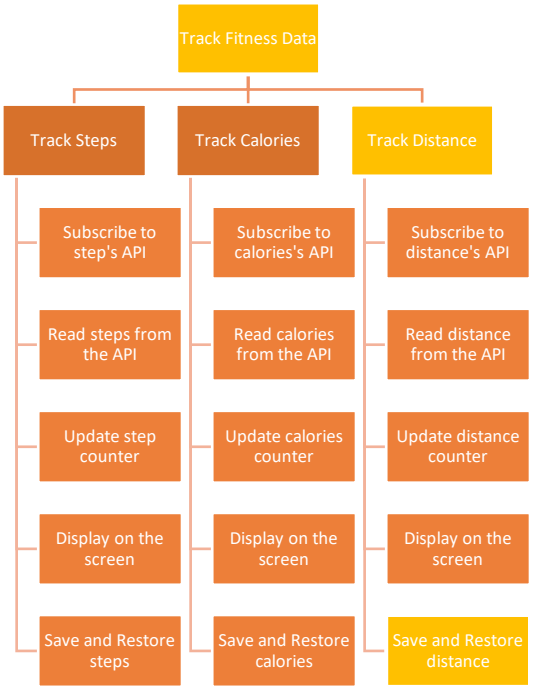
I finally call the method from DistanceTracker::update() with the total distance to update the UI.



Running the app updated the distance counter exactly as expected!

Test Use	Explanation	Actual Outcome	Works?
Track steps/calories/distance	Steps/calories/distance are tracked and are accurate	Each element is tracked and retrieved correctly	Yes
UI is updated	UI is updated for steps, calories and distance	The user is shown the counters on the main screen	Yes

Save and Restore Distance



To finish of this whole module I need to implement saving and restoring the distance on launch like calories and steps.

I am going to use SharedPreferences to keep the app consistent. This is also the recommended method.

Retrieve Distance – MainActivity

```
private static final String DISTANCE_TAG = "DISTANCE";
```

I crated a constant using static final to hold the tag to save and retrieve the values saved for distance in SharedPreferences.

```
// retrieve the values, if they don't exist get -1
long steps = sharedPrefs.getLong(STEPS_TAG, -1);
int cals = sharedPrefs.getInt(CALORIES_TAG, -1);
int distance = sharedPrefs.getInt(DISTANCE_TAG, -1);
```

Next in the restoreValues method, I get the saved integer using the tag if it exists. If it doesn't then -1 is assigned to distance

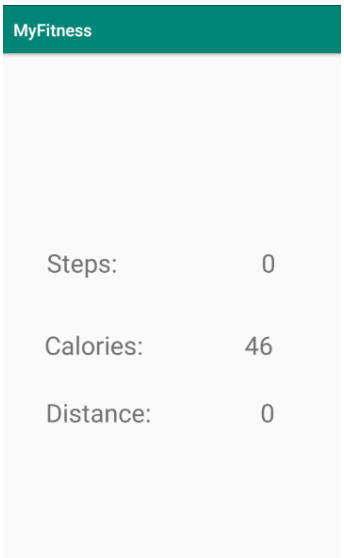
```
if (distance != -1) { // only set the value if it is saved
    distanceCountText.setText(String.valueOf(distance));
}
```

If distance is not -1 then set the text to the saved value.

Save Distance – MainActivity

```
@Override
public void sendDistance(int totalDistance) {
    distanceCountText.setText(String.valueOf(totalDistance)); // update UI element with total
    sharedPrefs.edit().putInt(DISTANCE_TAG, totalDistance).apply(); // save the calories
}
```

Finally, the sendDistance method uses the SharedPreferences putInt method to store the integer when it is assigned to the UI. This way the most up-to-date value is saved.



Run

Error

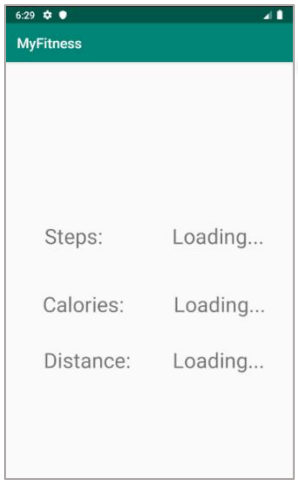
The app restores the values for all 3 elements, however there is one problem. I don't want the values of 0 to be restored, I want it to continue to show "Loading…" until up-to-date values are retrieved.

Test Use	Explanation	Actual Outcome	Works?
Steps/Calories/Distance counters are restored	When the app is launched, the counters previously saved are loaded in again until they're updated	Counters saved from previous launch are shown on launch	Yes

Fix

```
private void restoreValues() {
    // initialise sharedPrefs using my apps name and using the private context
    sharedPrefs = getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);

    // retrieve the values, if they don't exist get -1
    long steps = sharedPrefs.getLong(STEPS_TAG,  l: 0);
    int cals = sharedPrefs.getInt(CALORIES_TAG,  i: 0);
    int distance = sharedPrefs.getInt(DISTANCE_TAG,  i: 0);
    if (steps != 0) { // only set the value if it is saved
        stepsCountText.setText(String.valueOf(steps));
    }
    if (cals != 0) { // only set the value if it is saved
        caloriesCountText.setText(String.valueOf(cals));
        Log.i(TAG,  msg: "calories restored");
    }
    if (distance != 0) { // only set the value if it is saved
        distanceCountText.setText(String.valueOf(distance));
    }
}
```

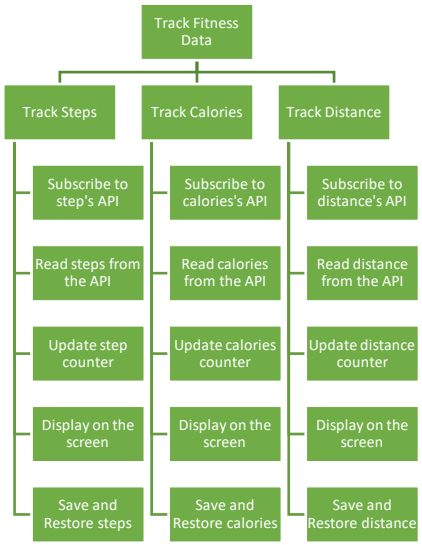
I updated the get methods to retrieve the value 0 if no value exists (as a default value). This will make sure that when the value saved is 0 or there is no saved value then the "Loading…" remains until the up-to-date methods are retrieved.


6:29
MyFitness
Steps:
Loading...
Calories:
Loading...
Distance:
Loading...


Run

Successful!

Conclusion



This concludes the first major modules of the app. In this module, I used OOP to take advantage of encapsulation and polymorphism to simplify and make the app easier to read and modify in the future. I used the Google API to subscribe, retrieve, save and update the UI for steps, calories and distance on the main screen.

I used many features of Java including interfaces and abstract classes to separate classes and write easier to read code. An interface was used to send data from "data" classes to the MainActivity so the UI could be updated. Each "data" class started a separate task to retrieve the counters separately which were then updated on the main thread as to not cause and unexpected crashes. I also asked for permission for google log-in, fitness details and location.

Throughout the development phase I've tested each step using the tests I made in the design phase. Here is a summary of what actually happens as documented above:

Permissions Test Summary

Test Use	Actual Outcome
Permission Dialogue shown	Permission dialogue appears on first launch to request permission to the user's fitness data
Request Location Permission	Permission for the location is shown. If granted the distance is retrieved. If it is denied, then the user is requested to grant the permission manually using settings. If they choose not to then distance is not retrieved at all – to prevent a crash.

Steps/Calories/Distance Test Summary

Test Use	Actual Outcome
Subscription is made	Subscription is made to steps, calories and distance as shown in the debug log
Track steps/calories/distance	Steps calories and distance are tracked and retrieved as the counters are shown in the debug log
UI is updated	UI with the latest counters are updated whenever they are retrieved which are shown on the main screen
Steps/Calories/Distance counters are restored	If there are any saved values from the previous launch, they are restored on launch until they can be updated by the API

The app was further tested by comparing the data retrieved to that of Google Fit App. This allowed me to see the data recorded by the app was up-to-date and as accurate as possible

Stakeholder's Feedback

Me:

I have completed all the requirements for version 1. Each module has been implemented and works as it should. The app is also broken up into separate classes and modules. I took advantage of encapsulation and polymorphism to keep the app simple. This allows the app to be written in modules which are easily readable and can be modified more easily.

The app launches and asks for permission for the account details and location. It then displays the either 'Loading…' or the counters from the previous launch if any are saved. It takes a few seconds to update the counters with the latest data. The latest data is saved in the background so it can be reloaded again.

I use separate tasks to retrieve data from the fitness API which is recording the data in the background. This allows my program to continue to run and not hang while it tries to retrieve data.

I will interview and show the app to each stakeholder to review and give me any feedback they have with what they like and dislike.

Harrison:

I like that the app launches very fast and loads the data quickly. The program is very simple but I feel a few pictures could improve the way the app looks. Our minds work in pictures so it will be easier to recognise by newer users. There seems to be an error on my phone as well where the lines are not aligned correctly. The app should have the lines aligned right.

Abytom:

The app displays and tracks all the data for version 1. It works fast however I would like if the data was displayed in separate boxes for better aesthetics. It also makes the app look more interactive. The new Android version is very dark theme focused so adding a dark theme would attract more users and look more modern.

Jamshed:

The app is very basic and tracks nearly everything I need. I like the simplicity of it. One thing I would improve is add some pictures to make it easier to read the values and their labels. The app does not update the steps and calories after updating my android version

Improve:

My main goal is to fix the issues with the new version of Android 10 causing issues with the app on Jamshed's phone.

Also, from feedback from my stakeholders the main thing I need to work on is the UI of the app. I am going to add icons as suggested by Harrison and Jamshed. I am also going to fix the issue of the alignment issue on Harrison's device. Finally, I will add a dark theme to suit the app so it can provide a very similar UI to the new Android 10 Material design.

Version 1 Improvements - Stakeholders

Android 10 Issues

Run

```
com.google.step_count.cumulative requires android.permission.ACTIVITY_RECOGNITION
```

From the error above and some research there are a few changes with Android 10. Google has documented the changes here. I can see that I will have to ask for extra permissions due to the new security system added in Android 10.

I switched *com.google.android.gms.permission.ACTIVITY_RECOGNITION* in the manifest file with *android.permission.ACTIVITY_RECOGNITION.* This solved the issue after clearing the cache on new updated devices.

This fixed Jamshed's issue with the app due to the new update and the app performs all the actions of version 1.

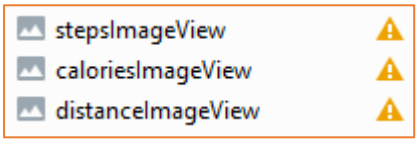
Run

```
I/MyFit-CalorieTracker: Successfully subscribed!
I/MyFit-DistanceTracker: Successfully subscribed!
I/MyFit-DistanceTracker: Successfully subscribed!
```

User Interface Updated

Next, I want to update the UI with some icons. I will also fix the alignment issue on Harrison's device which is because the original xml file did not have every constraint on the text views.

Firstly, I imported 3 images into the project so I can use them.



I added 3 ImageView widgets to the main_activity.xml which will represent the images in the GUI. I named them appropriately.

I then reset all the constraints I had previously set as they were not correct. This is what caused the issue on Harrison's device.

```
<dimen name="small_textview_size">15sp</dimen>
<dimen name="main_imageview_size">65dp</dimen>
```

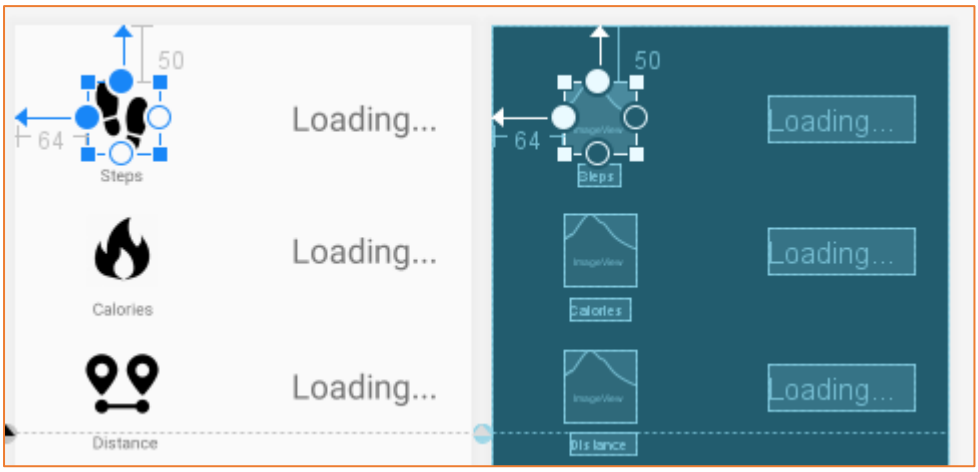
Justification

I created two dimen values to hold the size values for the text and the image. These will allow me to change the size of all the text and images easily in the future, allowing easy modification instead of use hard values for each image/text.

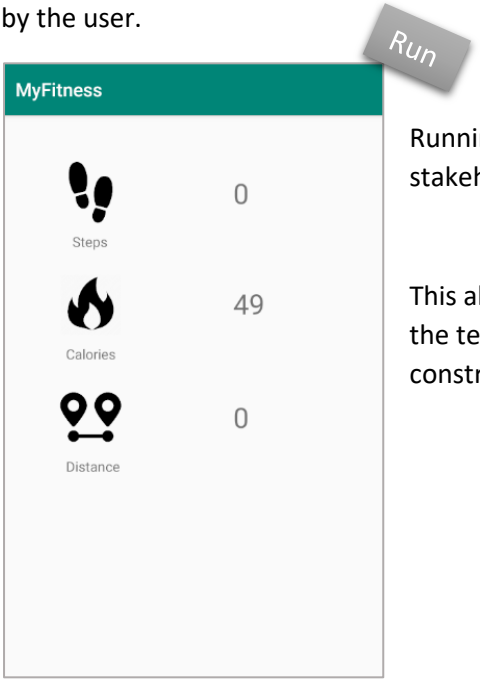
layout_width	@dimen/main_imageview_size
layout_height	@dimen/main_imageview_size

textSize	@dimen/small_textview_size

I then set the attributes to these values in the activity_main.xml file. This resized the text and images to the same size.



Lastly, I set the constraints, making sure to only have one vertical and horizontal line so issues would not occur on some devices like Harrison's. I made the images larger than the text to allow easier glance and detail when required by the user.



Running the app shows a cleaner, more polished look as requested by the stakeholder. This look keeps the minimal feel of the app.

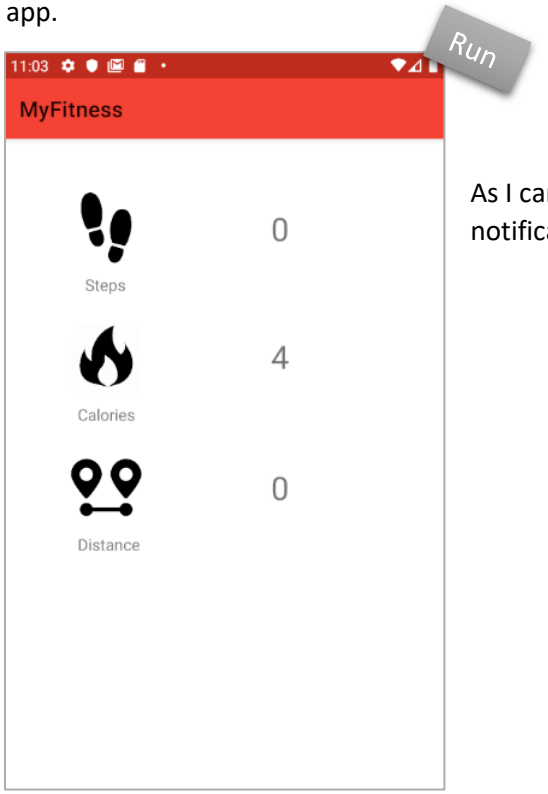
This also fixed Harrison's issue as when I had previously created the constraints for the text views, they were incorrect which I had not realised at the time. The constraints were also not accurate, so text alignment was off.

Dark Theme and Colour Changes

To further make the app more user friendly, I'm going to change the theming colours for the app.

```
<resources>
    <color name="colorPrimary">#F44336</color>
    <color name="colorPrimaryDark">#C02A1C</color>
    <color name="colorAccent">#F3E923</color>
</resources>
```

This is very simple to do due to the way I made the UI using values instead of hardcoded values. This allows me to just change the primary, primary dark and accent colours directly and they will change wherever they are used in the app.



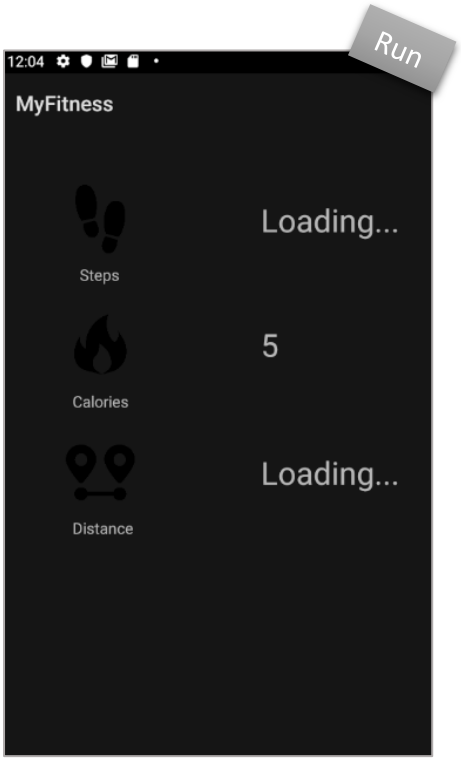
As I can see the primary colour is changed in the title bar while the notification bar is changed to the colorPrimaryDark.

```
implementation 'com.google.android.material:material:1.2.0-alpha01'
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
```

For the dark theme, I first changed the theming of the app to Material. I implemented the alpha version because the new Android Q methods are not implemented in the normal version. In the style file I changed the parent from default Android theme to the Material one.

```
<resources>
    <color name="colorPrimary">#AA0C0B</color>
    <color name="colorPrimaryDark">#000000</color>
    <color name="colorAccent">#F44336</color>
</resources>
```

I then created a folder called values-night which I then added a colors xml file. I used the primary colour as the accent and changed the primary colours to black and dark orange. This should create a black background with the app being black.



Running the app, I can see the title and notification bar are now black with the background being a very dark grey, allowing shows to be displayed. However, I have one problem. The icons are black which are hard to see and not in contrast to the background.

I reverted the image colours so they were white using photoshop and added them to the project folder under the same names but with the suffix of _dark.

res.xml:

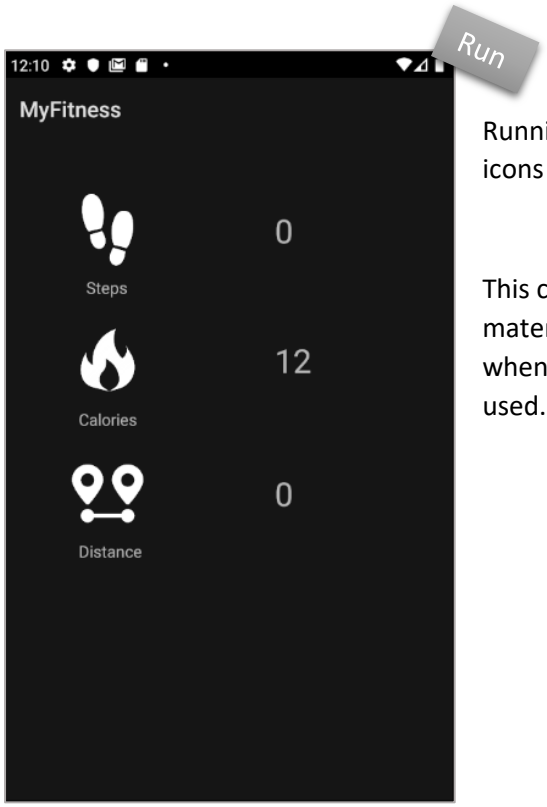
```
<resources>
    <drawable name="distanceImage">@drawable/distance</drawable>
    <drawable name="caloriesImage">@drawable/calories</drawable>
    <drawable name="feetImage">@drawable/feet</drawable>
</resources>
```

res(night).xml:

Fix

```
<resources>
    <drawable name="distanceImage">@drawable/distance_dark</drawable>
    <drawable name="caloriesImage">@drawable/calories_dark</drawable>
    <drawable name="feetImage">@drawable/feet_dark</drawable>
</resources>
```

I then added the above values to the res.xml file and the res.xml file in the night folder respectively. These are reference values I will use in the xml file for the main activity so the appropriate image can be retrieved depending on the theme.



Running the app now shows the light icons when in dark mode. The text and icons now contrast the background as required.

This concludes the request by Abytom to implement a dark theme. Using material design has allowed to easily switch to the dark theme in the app when the system-wide dark theme is activated. Otherwise the light theme is used.

BLACK PAGE – GO TO NEXT PAGE

Version 2

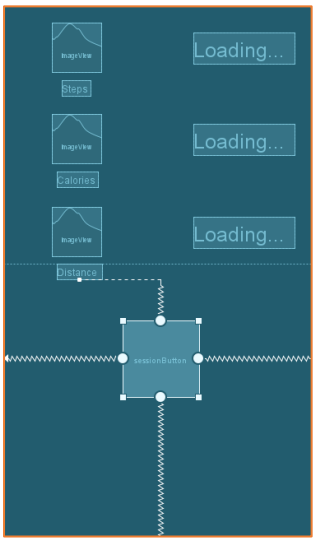
Setting Up

Before I start working on the first module, I need to setup a few items for the running and cycling session

Main Screen Button GUI

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape android:shape="oval">
            <stroke android:color="?android:colorPrimaryDark" android:width="7dp" />
            <solid android:color="?android:colorPrimary"/>
            <size android:width="150dp" android:height="150dp"/>
        </shape>
    </item>
</selector>
```

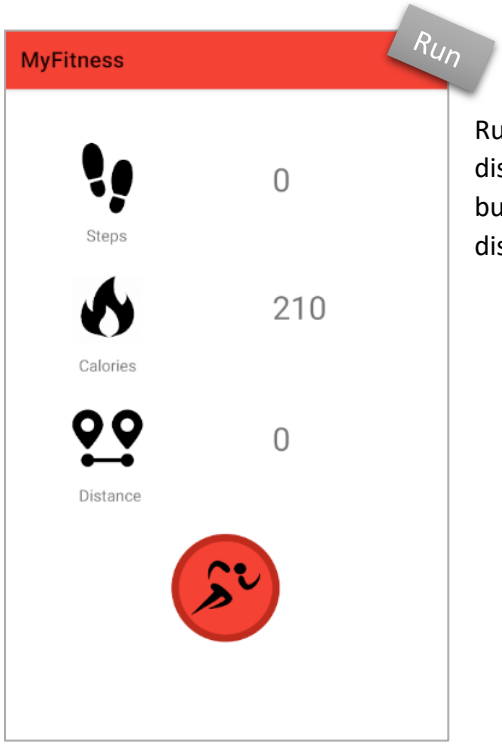
First, I created an xml file to represent our round button on the main screen. For this I created a shape and overrode the colours to the primary dark colour for the border and the primary colour for the background of the button.



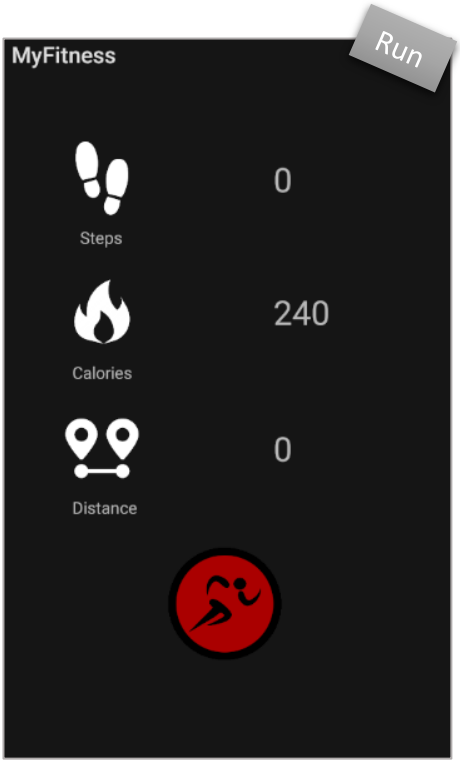
I added an ImageButton to the acitivity_main.xml file and set the name to sessionButton so I can call it from my code. I also set the constraints, so it is displayed correctly with any resolution phone. This will keep the same layout on every android phone.

```
<ImageButton
    android:id="@+id/sessionButton"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:adjustViewBounds="true"
    android:background="@xml/rounded_corner"
    android:padding="18dp"
    android:scaleType="fitCenter"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ditanceTextView"
    app:layout_constraintVertical_bias="0.22"
    app:srcCompat="@drawable/running" />
```

I added a running icon, which I then set to this button. I set the width and height to 100dp exactly and set a padding of 18 so its displayed in the centre. I also set the background of the button the oval button I designed above. I also set the ID to sessionButton so I can access it dynamically in my code. The image is a png icon which I imported into the project.



Running the app in light and dark mode displays the correct colours which make the button stand out – making it easier to distinguish it is an actual button and not icon.

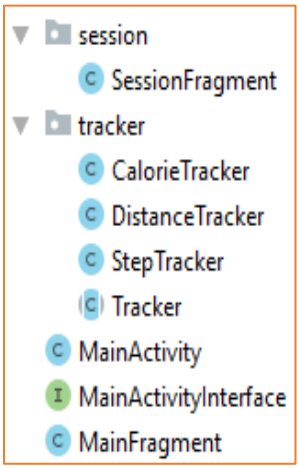


Using Fragments

Justification

Before I continue, I want to make the foundation for the running and cycling session. My application activity will need to use multiple different views which work separately called **fragments**. Fragments separate different views and their GUI logic, making tests and future modification easier. Because I am going to be using fragments, I must re-write my MainActivity to be a holder for all these fragments and move the current MainActivity into a new fragment view.

Much of this code will not be new but will be re-arranging the previous classes so they are more modular and easier to add on-to using OOP with fragments.



I sorted the tracker classes into a new tracker package. I had to change the access of the classes from protected to public so they can be accessed from outside the package by MainActivity.

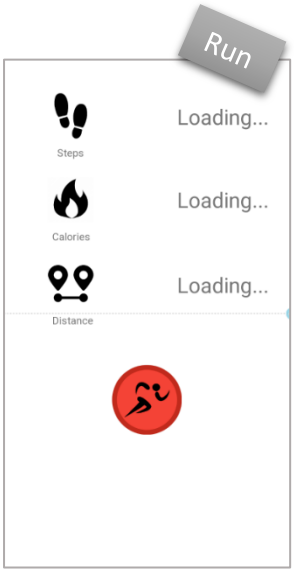
I also created another package for the session fragment which for now is just a placeholder fragment so I can call it from the main activity.

Fragments represent a user interface in the same application. Fragments make it easier to start new views and stack views on top of each other so the user can go back to the previous view with ease. Due to me using fragments I am going to have to create a large change to the MainActivity. It will now only consist of a frame holder for the fragments it holds. All logical steps will still be carried out by the same methods, but UI updates will have to be changed to the fragment classes.

Update main activity - Activity_main.xml

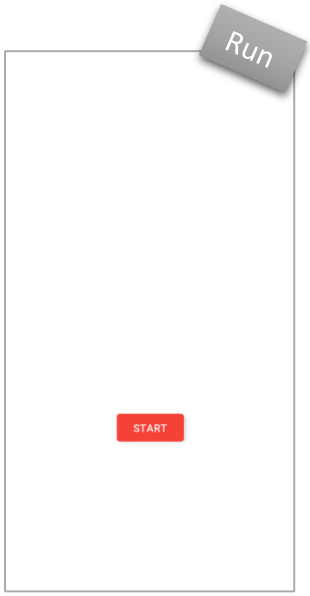
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">


    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </FrameLayout>
</LinearLayout>
```

First, I change the activity_main.xml file to just hold a FrameLayout. This will be changed dynamically while the program runs to update fragment that will be in view. I set the FrameLayout id to fragment container which I will use from the activity and other fragments to change the view of the main activity. I no longer need a constraint layout, so I change the layout to a vertical LinearLayout because it only holds one item.



Main View - Fragment_main.xml

I copied and pasted the previous activity_main.xml layout to the fragment_main.xml as the fragment will now control the main view. It is the same as before.



Session Start View - Fragment_session.xml

Finally, for the session fragment I set a placeholder button for now. This is to allow me to distinguish which fragment is being shown when I test the app for now. I will add to this later

Transfer View over - MainFragment.java

```
private TextView stepsCountText;    // used to update steps counter on UI
private TextView caloriesCountText; // used to update calories counter on UI
private TextView distanceCountText; // used to update distance counter on UI

private ImageButton sessionButton; // used to start the session fragment
```

I need to transfer all the view related items from MainActivity to the MainFragment class. First I cut over the private variables used to access the UI elements because the MainActivity no longer has access to them as they are held in MainFragment. I also added a sessionButton variable which will be used to access the new button I added above.

```
//stepsCountText.setText(String.valueOf(totalSteps)); // update UI element with total
```

I commented every UI update in MainActivity. All of them were setText calls to the UI elements. I commented these out because these elements no longer exist in the MainActivity view. These would have caused a compile time error. I will remove them later when I implement them into the fragment.

Populate Fragment View - MainFragment.java

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // inflate the class with the xml view and get a reference to that view
    View view = inflater.inflate(R.layout.fragment_main, container,  attachToRoot: false);

    stepsCountText = (TextView) view.findViewById(R.id.stepsCnt);     // assigned UI element stepsCnt
    caloriesCountText = (TextView) view.findViewById(R.id.caloriesCnt); // assign UI element caloriesCnt
    distanceCountText = (TextView) view.findViewById(R.id.distanceCnt); // assing UI element distanceCnt

    sessionButton = (ImageButton) view.findViewById(R.id.sessionButton); // assign the button to this element

    // start the session fragment when button is clicked
    sessionButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
                    .replace(R.id.fragment_container, new SessionFragment(),  tag: "session") // replace the current fragment
                    .addToBackStack(null);  // add this fragment to the stack

            fragmentTransaction.commit(); // execute the fragment call created above
        }
    });

    // required to return the fragment when called by activity
    return view;
}
```

For fragments all UI related actions must be done in the onCreateView instead of the onCreate method. First, for the MainFragment I get the view of the current fragment layout which I inflate the class with. I then refer to this view every time I need to access one of the UI elements. This is very similar to the MainActivity class; I set all the private varaibles to their respective UI elements.

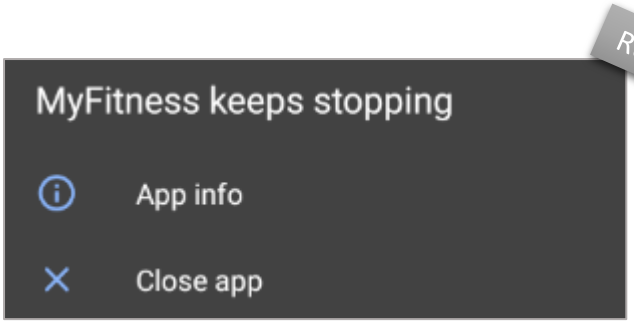
I then create a click listener for the sessionButton. This button creates a call for the current fragment to be replaced with session fragment. I first use Android Fragment Manager to replace the current fragment_container view with a new SessionFragment. I assign a tag “session” so it can be accessed later if required. Finally, I add the current fragment to the stack so it can returned when the user decides to back out of the SessionFragment fragment. Finally I execute the call.

In the end I return the view the fragment just created in the method so it can be accessed by the parent activity.

Start Fragment - MainActivity.java

```
FragmentTransaction fragmentTransaction;     // used to call fragments

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);                          // create superclas
    setContentView(R.layout.activity_main);                      // set layout for t

    // create the MainFragment view by adding it to the FrameLayout
    fragmentTransaction = getSupportFragmentManager().beginTransaction()
            .add(R.id.fragment_container, new MainFragment()).addToBackStack(null);
    fragmentTransaction.commit();
```

I created a Fragment Transaction variable which will be used to create calls to different fragments for the app activity so different views can be shown.




```
com.faizan.onefitness.MainActivity@f395e87 must implement OnFragmentInteractionListener
```

Running the app causes it to crash and looking at Logcat I are told that I must implement OnFragmentInteractionListener.

Fix

For now, I do not require the Interaction Listener for both of my fragments, so I have completely removed the interfaces and the methods used by them. This fixed the crashing issue.

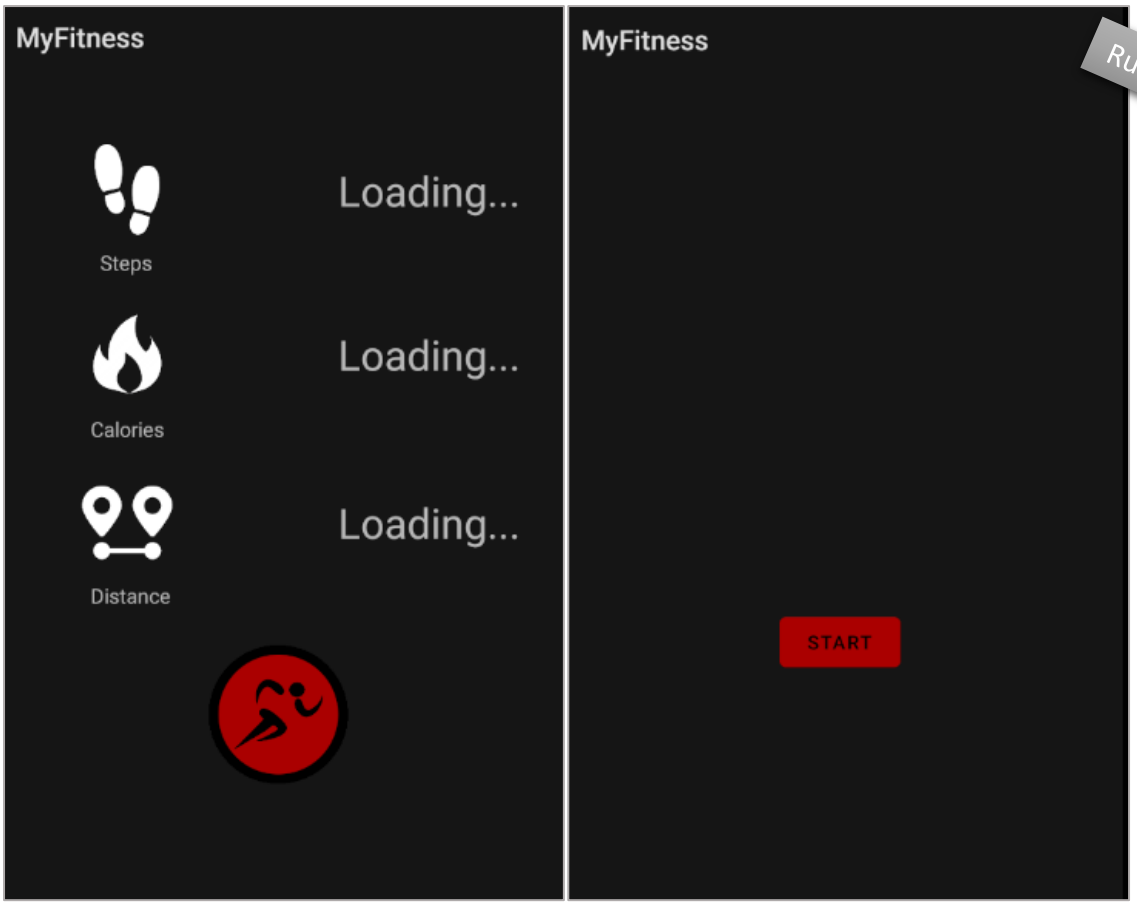
Running the app does not cause a crash anymore but the back button does not exit the app, it goes to the main activity view which is just black.

After some research I figured out I was adding the main view to the stack.

```
.add(R.id.fragment_container, new MainFragment()).addToBackStack(null);
```

Fix

I removed the .addToBackStack call in the MainActivity::onCreate() method which solved the issue.



Running the app now shows the two appropriate fragments being displayed correctly without any crashes or other errors. As I expected, the values for steps, calories and distance are not updated as I commented out the lines earlier in MainActivity. I need to re-write the updating methods for the UI now to fix this issue.

Get Totals - MainFragment

```
public void receiveStepsUpdate(String steps) {
    stepsCountText.setText(steps); // set updated steps
}

public void receiveCaloriesUpdate(String cals) {
    caloriesCountText.setText(cals); // set updated calories
}

public void receiveDistanceUpdate(String distance) {
    distanceCountText.setText(distance); // set updated distance
}
```

In the MainFragment.java file I created 3 public methods in the class that I will simply use to receive the updated counters from the Main Activity and it will update the appropriate UI element.

Send Totals - MainActivity

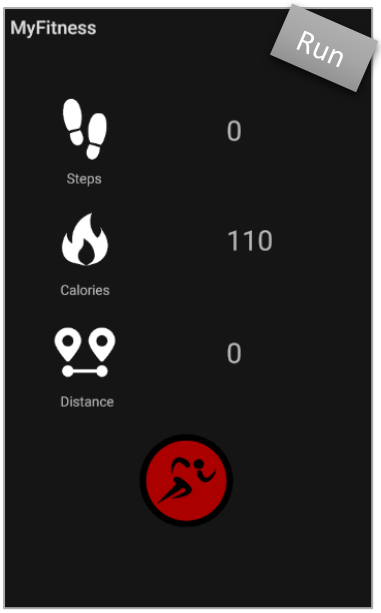
```
MainFragment mainFragment;    // hold reference to mainFragment
```

```
mainFragment = (MainFragment) new MainFragment();
```

Firstly, I created a variable in the class which I set to a new MainFragment. I will use this reference to send the totals to the mainFragment instead of having to retrieve it multiple times.

```
@Override
public void sendSteps(long totalSteps) {
    mainFragment.receiveStepsUpdate(String.valueOf(totalSteps));  // update UI element with total
    sharedPrefs.edit().putLong(STEPS_TAG, totalSteps).apply(); // save the steps
}

@Override
public void sendCalories(int totalCalories) {
    mainFragment.receiveCaloriesUpdate(String.valueOf(totalCalories)); // update UI element with total
    sharedPrefs.edit().putInt(CALORIES_TAG, totalCalories).apply(); // save the calories
}

@Override
public void sendDistance(int totalDistance) {
    mainFragment.receiveDistanceUpdate(String.valueOf(totalDistance)); // update UI element with total
    sharedPrefs.edit().putInt(DISTANCE_TAG, totalDistance).apply(); // save the calories
}
```

In the send methods I call the appropriate receive method of the mainFragment with the number value converted into a string so it can be assigned to the UI.



Running the app gives us a successful run in updating the UI appropriately using the fragment now instead of the activity.

However, I need to still restore the values saved when the app is restored like before.

```
if (steps != 0) { // only set the value if it is saved
    //stepsCountText.setText(String.valueOf(steps));
    mainFragment.receiveStepsUpdate(String.valueOf(steps)); // update UI element with total
    Log.i(TAG,  msg: "Steps restored");
}
if (cals != 0) { // only set the value if it is saved
    mainFragment.receiveCaloriesUpdate(String.valueOf(cals)); // update UI element with total
    Log.i(TAG,  msg: "calories restored");
}
if (distance != 0) { // only set the value if it is saved
    mainFragment.receiveDistanceUpdate(String.valueOf(distance)); // update UI element with total
    Log.i(TAG,  msg: "Distnace restored");
}
```

Run

Crash

I remove the commented lines and replace them with the appropriate fragment update methods for steps, calories and distance.

```
Attempt to invoke virtual method 'void android.widget.TextView.setText(java.lang.CharSequence)' on a null object reference
ceiveCaloriesUpdate(MainFragment.java:71)
storeValues(MainActivity.java:207)
Create(MainActivity.java:67)
```

Running this app causes a crash. From the crash I see that the receive method is being called on the Fragment before it is initialised on recieveCaloriesUpdate() method.

Fix

I'm going to transfer all the counters for the steps, calories and distance to the fragment. The fragment should manage these values and save them so they can be done at the right time without causing a crash due to having the MainActivity running without waiting for the MainFragment to finish executing.

```
/*
 Interface used to send updated values from
 Tracker classes to the Main Fragment
 */
public interface MainFragmentInterface {
    void sendSteps(long totalSteps);

    void sendCalories(int totalCalories);

    void sendDistance(int totalDistance);
}
```

```
public class MainFragment extends Fragment implements MainFragmentInterface
```

First, I created a MainFragmentInterface.java which is the same as MainActivityInterface which I removed. I then implemented this interface into the MainFragment class.

```
private SharedPreferences sharedPrefs; // used to save and retrieve values
```

```
private void restoreValues() {
    // initialise sharedPrefs using my apps name and using the private context
    sharedPrefs = getActivity().getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);

    // retrieve the values, if they don't exist get -1
    long steps = sharedPrefs.getLong(STEPS_TAG,  defValue: 0);
    int cals = sharedPrefs.getInt(CALORIES_TAG,  defValue: 0);
    int distance = sharedPrefs.getInt(DISTANCE_TAG,  defValue: 0);
    if (steps != 0) { // only set the value if it is saved
        //stepsCountText.setText(String.valueOf(steps));
        stepsCountText.setText(String.valueOf(steps));  // update UI element with total
        Log.i(TAG,  msg: "Steps restored");
    }
    if (cals != 0) { // only set the value if it is saved
        caloriesCountText.setText(String.valueOf(cals)); // update UI element with total
        Log.i(TAG,  msg: "calories restored");
    }
    if (distance != 0) { // only set the value if it is saved
        distanceCountText.setText(String.valueOf(distance)); // update UI element with total
        Log.i(TAG,  msg: "Distnace restored");
    }
}
```

Justification

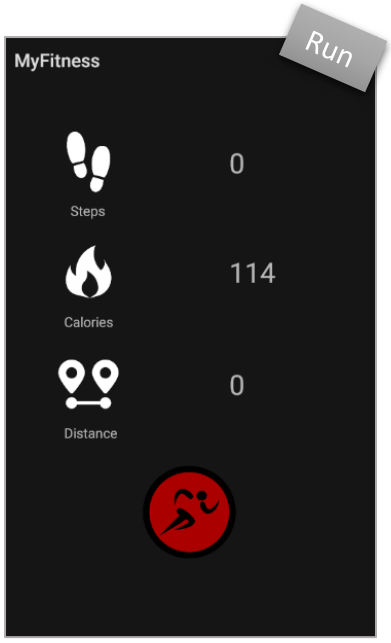
In the MainFragment class I created brought over the sharedPreferences private variable and restoreValues method from the MainActivity. I removed that from the MainActivity as it was no longer required there. This is so the restoreValues is handled after MainFragment has been created so the app does not crash like above. The validation here used has already been discussed and implemented in Version 1.

```
@Override
public void sendSteps(long totalSteps) {
    stepsCountText.setText(String.valueOf(totalSteps));  // update UI element with total
    sharedPrefs.edit().putLong(STEPS_TAG, totalSteps).apply(); // save the steps
}

@Override
public void sendCalories(int totalCalories) {
    caloriesCountText.setText(String.valueOf(totalCalories)); // update UI element with total
    sharedPrefs.edit().putInt(CALORIES_TAG, totalCalories).apply(); // save the calories
}

@Override
public void sendDistance(int totalDistance) {
    distanceCountText.setText(String.valueOf(totalDistance)); // update UI element with total
    sharedPrefs.edit().putInt(DISTANCE_TAG, totalDistance).apply(); // save the calories
}
```

I then brought over the send methods from the MainActivity but replaced the receive method calls with directly changing the text as I now have access to those UI methods in the fragment. This also means the receive methods in the fragment are not required anymore so I removed those as well.

```
private static final String STEPS_TAG = "STEPS";
private static final String CALORIES_TAG = "CALORIES";
private static final String DISTANCE_TAG = "DISTANCE";
```

Finally I moved over the static variables to save the values in the SharedPreferences and removed them from MainActivity.

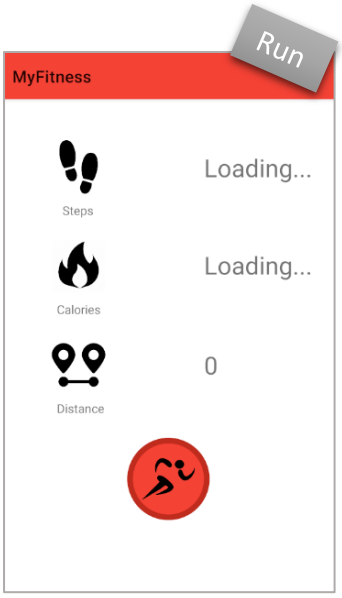


Running the app now works without any crashes.

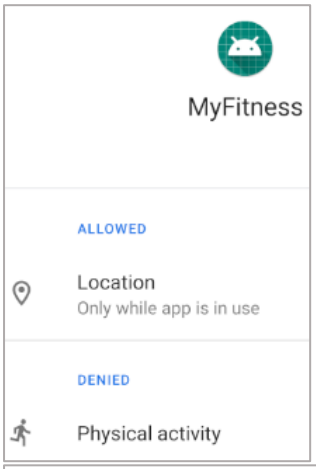
What caused the crash was that the MainActivity would try to update the UI elements before the MainFragment had a chance to create the relevant UI elements. Moving the restore method to the MainFragment and having that handle all the UI updates directly solved the problem because MainFragment will run on its own thread and create the UI elements before trying to restore the values. This solved the crashing issue trying to restore the calories.

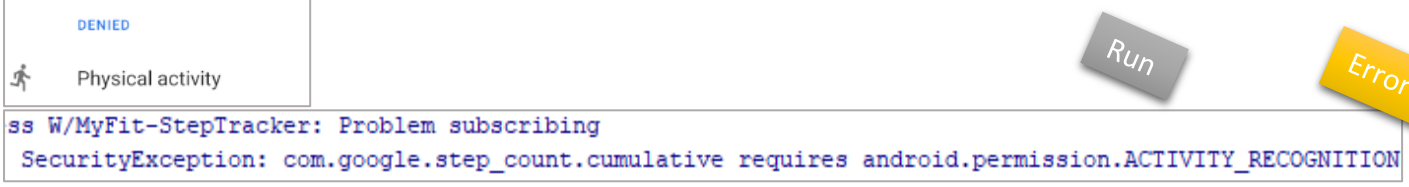
Due to using an emulator no steps or distance was recorded. I now have converted our whole app to use fragments so I can easily switch views.

Fixing Android 10 Permission Issues – again…



Running the app on clean updated phones has caused larger issues with the app no longer able to retrieve the calories and steps anymore. This is because the new android 10 version now requires the permission to be requested from the user unlike previous editions where it just had to be declared in the manifest. This means I must request the permission manually just like the location.





Justification

Copying and pasting the same code but for the activity tracking caused an error because different threads would request the permissions separately. However, I can only request the activity permissions after the location permissions are requested as I soon learned. This meant I had to restructure the MainActivity of the app. It cannot subscribe to the steps because I don't have the permission by the time the tracking methods are called.

```
private void grantPermissions() {
    // check if we already have location permission first
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {

        // Permission is not granted
        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
                Manifest.permission.ACCESS_FINE_LOCATION)) {
            manualPermissionDialog();
        } else {
            // No explanation needed; request the permission
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSIONS_REQUEST_LOCATION);
        }
    } else {
        // Permission has already been granted
        distanceTracker.subscribe();
        grantActivityPerms();
    }

}
```

Firstly, I renamed the subscribeToDistancePerms to grantPermissions as the name was more appropriate because I now have multiple permissions to request and tracked steps, calories and distance now.

Validation

Next, I call the grantAcctivityPerms method which I created which will be called if the location permissions are already granted. I validate that permissions have already been granted before getting the activity permission because the app would crash otherwise

```
case MY_PERMISSIONS_REQUEST_LOCATION:
    // If request is cancelled, the result arrays are empty.
    if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // permission was granted
        distanceTracker.subscribe();
        grantActivityPerms();
    } else {
```

Validation

I also call this method when the location permissions are granted for the first time in the result method after the dialogue is finished for the location permissions.

```
private void grantActivityPerms() {
    // check if we need to request the permission as it's only required after API 28
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {

        // check if we already have permissions
        if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACTIVITY_RECOGNITION)
                != PackageManager.PERMISSION_GRANTED) {
            // have permissions been denied last time
            if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
                    Manifest.permission.ACTIVITY_RECOGNITION)) {
                manualPermissionDialog();
            } else {
                // No explanation needed; request the permission
                ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACTIVITY_RECOGNITION},
                        MY_PERMISSIONS_REQUEST_ACTIVITY);
            }
        } else {
            // retrieve data if we do have permissions
            calorieTracker.subscribe();
            stepTracker.subscribe();
        }
    } else {
        // retrieve data if we have a previous version of android
        calorieTracker.subscribe();
        stepTracker.subscribe();
    }
}
```

The grantActivityPerms is executed to retrieve permissions for to track user tracking activity. I used validation in this method to first check for the SDK version that the app is currently running on. If the version is lower than Android 10(Q) then I do not have to request these permissions and therefore can call the tracking methods directly. However, with the new version of android I must request the permissions from the user. This is done exactly as the location permissions are requested except, I request the activity permissions instead of the location. If I already have these permissions, then I just call the subscribe methods on the trackers. These methods are not called otherwise as they would cause a crash

Validation

```
case MY_PERMISSIONS_REQUEST_ACTIVITY:
    // If request is cancelled, the result arrays are empty.
    if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // permission was granted
        calorieTracker.subscribe();
        stepTracker.subscribe();
    } else {
        // permission was denied, ask to manually set it
        manualPermissionDialog();
    }
    break;
```

Fix

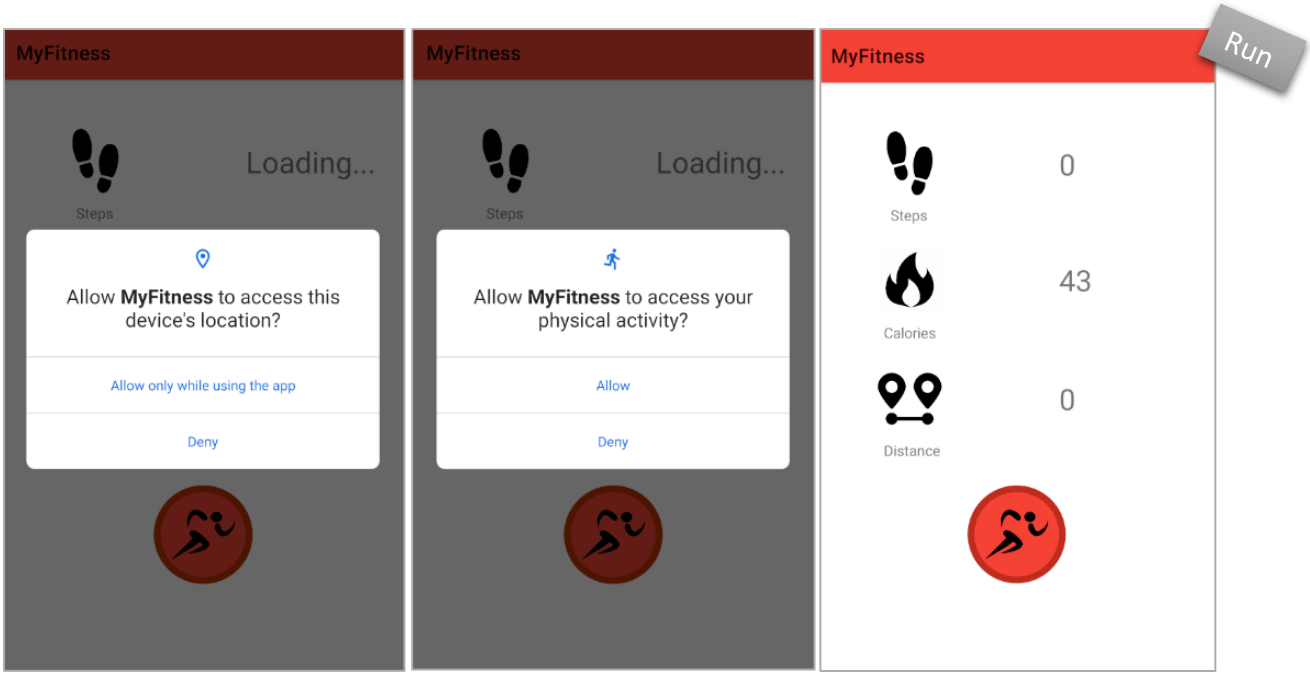
I also add a case to the onRequestPermissionResult method which is called after a user exits a permission dialogue. This is like the location case as it requests the permissions from the user manually if they decline or have them ignore it. If the permissions are granted then I subscribe to the tracking classes.

```
private static final int MY_PERMISSIONS_REQUEST_ACTIVITY = 5551;
```

The static variable is used to check for the result of the activity permission above.

```
<string name="manual_permission">Go to Settings to grant permissions for tracking.</string>
```

I also change the string in the dialogue to request the permissions to be appropriate for multiple permissions.

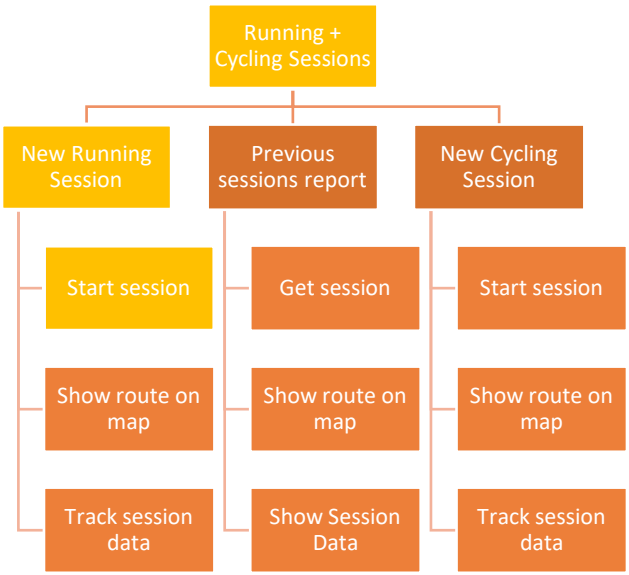


Running the app now allows us to retrieve all this data on Android 10 and any future versions as well as allow the app to continue working on previous versions. This stops the screen from just forever showing Loading… as the subscribe methods are now called.

Validation Testing

Using the permission dialog allows me to prevent a crash by retrieving data that is not accessible because the user has denied permissions. This means the app will only continue to function when the user grants physical activity permissions. A dialog is shown to the user to show the steps to grant these permissions if they decline them.

Module 1 - Running Session

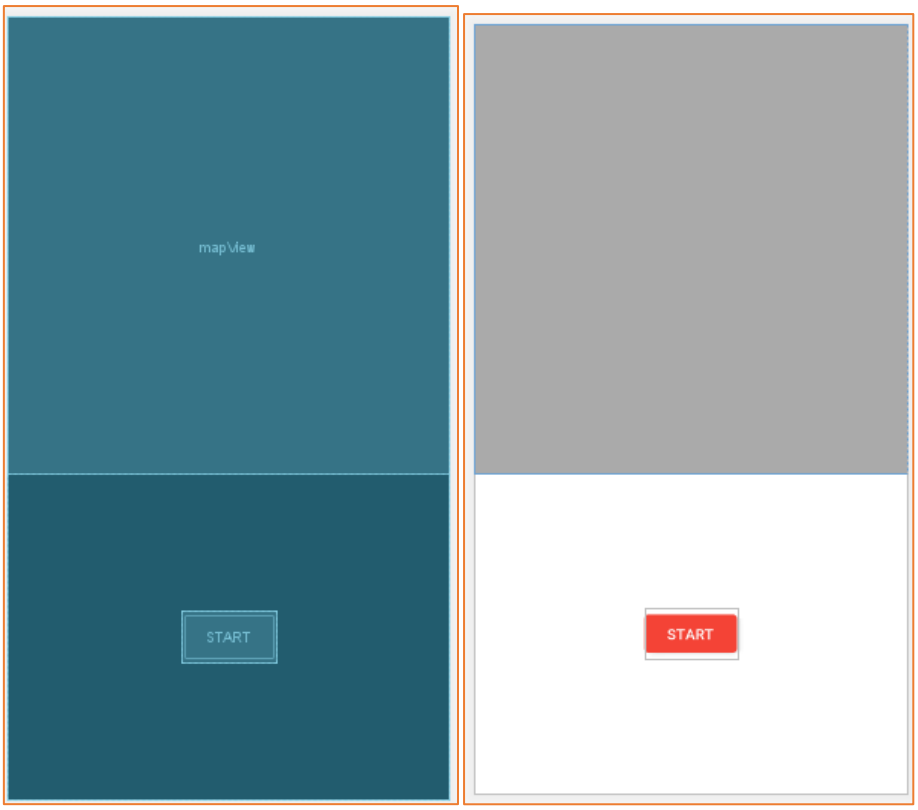


I will now be creating the actual fragment for the running session. This will be reused for cycling as well but I will implement that later. First, I want to create a map that shows the current location of the user. This will show the user their surroundings so they can choose the right spot to start the run.

I've had to visit my developer console for google and add another key for using the Google Map's API within my app. I implemented this key into my project to allow access. Without this key I would not be allowed to make calls to the API.

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

In Android Studio I've had to download Google Play Services as Google Maps is a part of those services. I've then had to implement the maps into the project using gradle so the project was linked to the maps API.



Before initialising the map I have to create the view for it. I inserted a MapView and conveniently assigned it an ID of mapView in the fragment_session.xml file. I set the constraints to the top and sides as well as setting the bottom to the START button. Finally, I added a gap in between the two to create a more user-friendly interface. I also set the ID for the button to startButton so I can use it in my code later.

Create Map View – SessionFragment

I need to track the user's location and show it on the map so the route can be tracked and shown later as well.

```
MapView mapView; // point to the UI element
GoogleMap map;   // control access to the Maps API
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_session, container,  attachToRoot: false);

    mapView = (MapView) view.findViewById(R.id.mapView); // init mapView fragment
    mapView.onCreate(savedInstanceState);
    mapView.onResume();                                  // if there is a previous state
    mapView.getMapAsync( onMapReadyCallback: this);

    // set current location on the map and zoom in
    getCurrentLocation();

    return view;
}
```

First, I inflate the view. From the view I retrieve the map view I placed in the previous section. I restore the instance of this map and resume it. I do this so if the user decides to switch between this app and other apps, it resumes the map to the previous location. Finally, it starts the map in a different thread so it does not hang the GUI.

I call another method to retrieve and set the current location before returning the view to the activity.

```
public class SessionFragment extends Fragment implements OnMapReadyCallback {
```

```
// This method is called automatically when the maps view is ready
@Override
public void onMapReady(GoogleMap googleMap) {
    map = googleMap;        // called when GoogleMap is ready
    map.setMapType(GoogleMap.MAP_TYPE_NORMAL);  // Make the retrieved terrain normal
    map.getUiSettings().setAllGesturesEnabled(false);   // Disable any movement by user
    map.setMyLocationEnabled(true);         // enable the user to retrieve their location
}
```

Next I implemented the map ready interface and implemented the onMapReady method. In this method I set the GoogleMap member variable. This method is called when the map api and view are all ready to be used. This way I can make some API changes to modify the view of the map. First I set the map to show the normal terrain. After I disable all the gestures the user can use. This way the map is only controlled by the app. Finally, I set the current location to true so the user can click the button to centre their location.

```
private void getCurrentLocation() {
    // client run in the background to not slow down GUI when getting location
    FusedLocationProviderClient fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(getActivity());

    // start background tast
    Task<Location> task = fusedLocationProviderClient.getLastLocation();
    // validation: wait for task to complete before moving camera
    task.addOnSuccessListener((OnSuccessListener) (location) → {
            Log.i(TAG,  msg: "Updated Location");
            moveCamera(location);
    });

}
```

Validation

This method is called when the map view is ready. It registers a client with the Android GPS provider to retrieve the location of the user. This method is called using a Task so it is run in the background without having to slow down the GUI of the app. I use validation to check the task has completed successfully and if it has I move the camera of the map to that location using the moveCamera method. The validation of the location retrieval is required before I can use the data. If I directly use the data then it would likely cause a crash as the location might be null.

Validation Testing

```
Attempt to invoke virtual method 'double android.location.Location.getLatitude()' on a null object reference
```

As shown by this error if I were not to validate our data then the app would crash by trying to retrieve from a null object. I ran into this problem before adding the validation above. Now that the validation is added it will move the camera only when the location is valid and not null, meaning after the phone GPS has replied with the location.

```
private void moveCamera(Location location) {
    // get the coordinates of the current location
    LatLng coordinates = new LatLng(location.getLatitude(), location.getLongitude());
    // animate the camera to zoom in on the coordinates
    map.animateCamera(CameraUpdateFactory.newLatLng(coordinates));
    // zoom in onto the user at level 15
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coordinates, 15));
}
```

The moveCamera method gets the coordinates, both latitude and longitude from the location retrieved by the client. It uses these coordinates to centre the map on the user's location and zoom in.

```
public SessionFragment() {
    // Required empty public constructor
}
```

The class also has an empty constructor which is required or the class would crash because the default constructor will cause the map task to fail.

To use maps, I have to check that the API is ready before can access it and use the Google Maps data in my map.

For this I created another class that verifies the maps activity is ready in a separate task so the GUI can continue to run without causing it to hang and crash because Google Maps cannot be displayed.

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
```

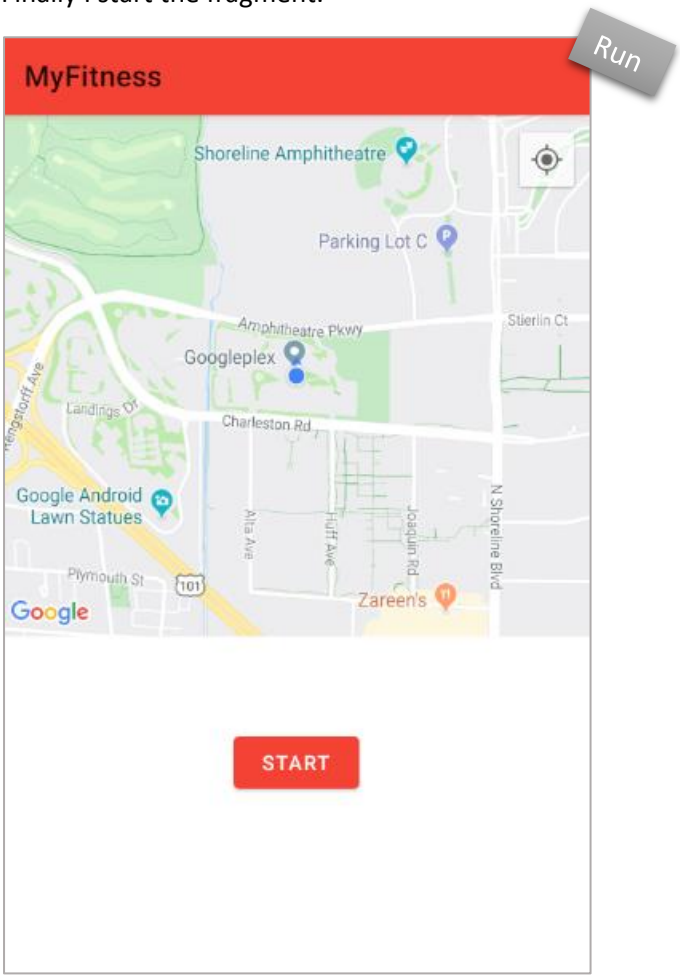
Firstly, I get a MapFragment which I initialise to the fragment I| use in our application to display the maps. This fragment will be used to retrieve all the maps data without us having to worry about how it is done by the API. I however have to make sure the map data is retrieved before executing any commands on the fragment.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

I override the onMapReady method in this class and implement it by initialising the map UI element in my app. This will start to show the map and allow the rest of my app to execute methods on the map to show the user's current location. This will also create a moving animation on the map

```
// start the session fragment when button is clicked
sessionButton.setOnClickListener((v) → {
        FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container, new SessionFragment(), tag: "session") // replace the current fragment
                .addToBackStack(null);  // add this fragment to the stack

        fragmentTransaction.commit(); // execute the fragment call created above
});
```

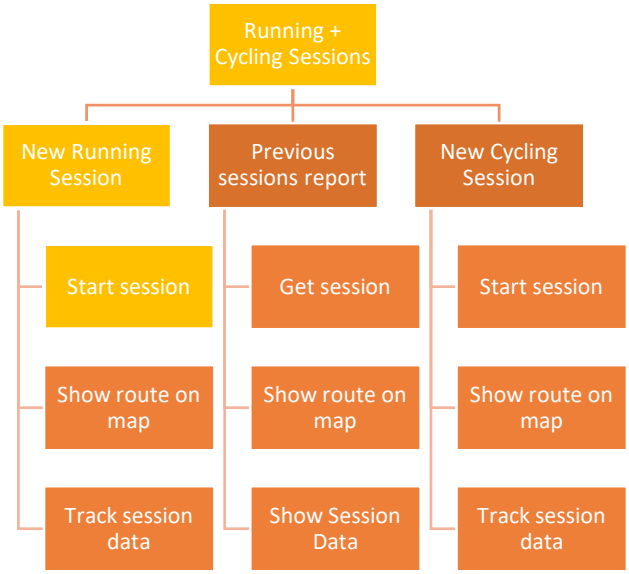
Finally, to start this map fragment, I set the listener on the session button on the main fragment to start a transaction for the new fragment above. This transaction will replace the user with the new view as well as allow the use to use their phones back button to return to the main screen as I add the current fragment to the current stack. Finally I start the fragment.



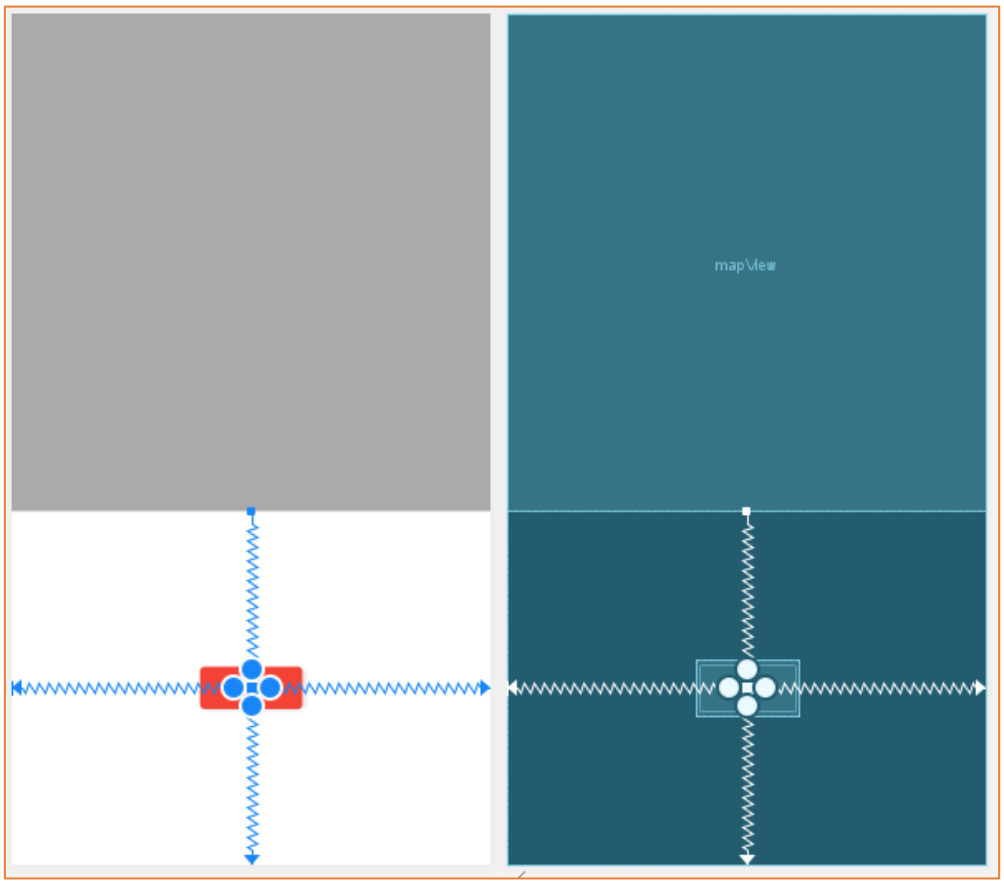
As this is a virtual machine it shows the location as Googleplex in America. However, testing this on my own device shows my current location and zooms in with the animation I implemented to provide a nicer interface to the user.

This screen is shown after clicking the run button and the back button on android returns you to the main screen.

Implement start button – SessionFragment



I want the start button to start a timer before starting the session. This will allow the user to put their phone in a safe spot before they begin their run instead of counting the time that they are stationary. For this I will use a simple TextView.



I inserted a TextView and set the ID to countDownText. I place this in the exact position as the button. I then set the text of this to 3 as that will be the first digit shown as it counts down. Finally I set the visibility to GONE which means it will not take up any space at all and will have it completely hidden until I set it to visible programmatically.

```
private Button startButton;      // used to set behaviour of button
private TextView countDownText; // used to show a counter to the user

private boolean started = false; // keeps track of if session has started
```

In SessionFragment class I create three private variables. The first two will refer to items in the view and allow us to change it while the started Boolean will allow me to track of when the session has been started and when it is stopped.

```
// assign UI items from view
startButton = (Button) view.findViewById(R.id.startButton);
countDownText = (TextView) view.findViewById(R.id.countDownText);

// create a listener which will be called when the button is clicked
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // if the session has already been started
        if (started) {
            runningSession.stopSession(); // then stop it
            startButton.setText("Start"); // update the UI
            started = false; // able to start a session again
        } else {
            startButton.setVisibility(View.INVISIBLE);  // if its not been started then hide the button
            countDownText.setVisibility(View.VISIBLE); // and begin the count down
            startSession(); // start the session as well
            started = true; // session is started
        }
    }
});
```

In the onCreateView method I retrieve the UI elements form the view and assign them to the private variables.

Next I set a listener on the startButton to allow a session to be started as well as stopped. I use the started Boolean to check if a session has started. If it has started then I stop recording the session and set the button text to start so another session can be started. I will likely change this behaviour in the future but for now it will be enough. Next I set the Boolean to false so another session can be started. I will implement the RunningSession class later to handle all the background activities for tracking the user.

If the Boolean is false and no session has been started then I hide the startButton by setting the visibility to INVISIBLE. I used INVISIBLE instead of GONE because I still want the button to take up room on the UI so the map is not unexpectedly resized after the button is pressed. The countDownText was set to GONE because it takes up less space than the button and will not cause and resizing issues. This will also save memory while the app is running.

Finally, I call the private method startSession which I will implement next.

```
private void startSession() {
    // start a timer for 4 seconds with updates every second
    CountDownTimer counterDownTimer = new CountDownTimer( millisInFuture: 4000,  countDownInterval: 1000) {
        // this method will be called every second
        @Override
        public void onTick(long millisUntilFinished) {
            int seconds = (int) millisUntilFinished / 1000 % 60; // retrieve the amount of seconds remaining
            countDownText.setText(String.valueOf((seconds))); // update the UI with the seconds remaining
        }

        // this method will be called when the count down has finished.
        @Override
        public void onFinish() {
            countDownText.setVisibility(View.GONE); // remove the counter from the UI
            startButton.setText("Stop");            // set the text for the button to stop
            startButton.setVisibility(View.VISIBLE);// set the button to be visible again
            runningSession.startSession();          // start the running session
        }
    }.start(); // start the counter straight after it is created
}
```

The startSession method will handle the count down for the user as well as actually start the session to record. First I created a CountDownTimer which I set to 4 seconds (4000 ms) and have it tick every second (1000 ms). This will allow me to update the UI every second instead of having to check for updates multiple times a second.

I have earlier decided to set the count down from 3 however here I have set it to 4. The reason for this is because the first second passes very quickly and does not show the number 3 to the user at all. If I start counting down from 4, then a 3 will be shown for a second. The value 4 will never be set to the TextView because I are going down in an interval of one whole second.

The onTick method is called whenever a second passes. This method retrieves the number of seconds remaining by dividing the milliseconds by 1000 and using a modulus of 60. I use this little equation to convert the milliseconds to a whole second. Using the modulus means I don't have to worry about any decimals appearing. I assign this to an int which then is set the text of the countDownText variable I defined above.

The onFinish method is called when the countdown has completed. Firstly, it makes the countdown TextView disappear and has the button reappear but with the text as Stop instead of start now. Finally, the session is started.

The start method is called on this class directly after it is created to start the timer immediately.

```
public class RunningSession {
    private static final String TAG = "MyFit-RunningSession";  // used for debugging

    protected Context context;     // get parent activities context to pass into methods

    private Session session;

    public RunningSession(Context activity) { this.context = activity; }
```

Next I created a class to track all the data for the running session when it is started by the user. I have a TAG for debugging just like every other class. Then I have a Context variable to hold where the class has been started from. The constructor takes in the context and assigns it. This is the only constructor for this class. The Session variable will be used to access the current running session.

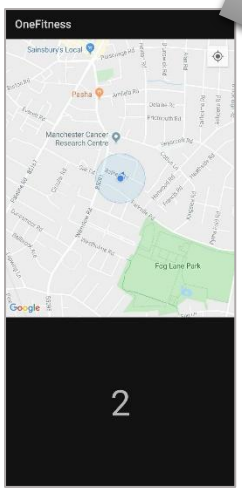
```
public void startSession() {
    // get the current time
    Date startTime = Calendar.getInstance().getTime();
    session = new Session.Builder()
            .setName("Running") // set the name
            .setIdentifier("OneFitness" + String.valueOf(startTime.getTime())) // set unique identifier using time
            .setDescription("Running Session") // set description of the session
            .setStartTime(startTime.getTime(), TimeUnit.MILLISECONDS) // set the start time
            .setActivity(FitnessActivities.RUNNING) // session is running type
            .build();
    // start the session by registering it in the API
    Task<Void> response = Fitness.getSessionsClient(context, GoogleSignIn.getLastSignedInAccount(context))
            .startSession(session);
}
```

The startSession method is used to register a new session with Google Fit. First I use a Date class to get the current time in milliseconds. I need this to pass to the session when I start it so there is a start time. The session for now has a title of Running and the identifier is the current time in milliseconds. I set the start time and set the activity to a RUNNING and then build it. I start this session in a separate task because it will be running in the background, so it does not freeze the UI. I log a message at the end for debugging purposes.

```
public void stopSession() {
    Task<List<Session>> response = Fitness.getSessionsClient(context, GoogleSignIn.getLastSignedInAccount(context))
            .stopSession(session.getIdentifier());
    Log.i(TAG,  msg: "Stopped Session");
    response.addOnSuccessListener(new OnSuccessListener<List<Session>>() {
        @Override
        public void onSuccess(List<Session> sessions) {
            Log.i(TAG, sessions.toString());
        }
    })
    .addOnFailureListener((e) -> { Log.i(TAG, e.toString()); });
}
```

The stop session does the opposite. It calls the API to stop recording the session and has it return all the results. I print these results the log for debugging for now. If it fails I print out the exception. These listeners act as validation methods and allows us to only use valid data or cause an error otherwise.

Run



Running the app and clicking the start button shows the count down as expected and starts the session. However, after stopping the session the validation I implemented returns an exception instead of a result.

Error

```
/MyFit-RunningSession: Started Session
/MyFit-RunningSession: Stopped Session
/MyFit-RunningSession: com.google.android.gms.common.api.ResolvableApiException: 5000: Application needs OAuth consent from the user
```

From the error I can see that there is some error in terms of accessing data that I do not have permission to. The error seems to be for not asking for consent for something I require from the user. I have already asked to access user's activity and location data so I believe this error is more to do with accessing the user's fitness data.

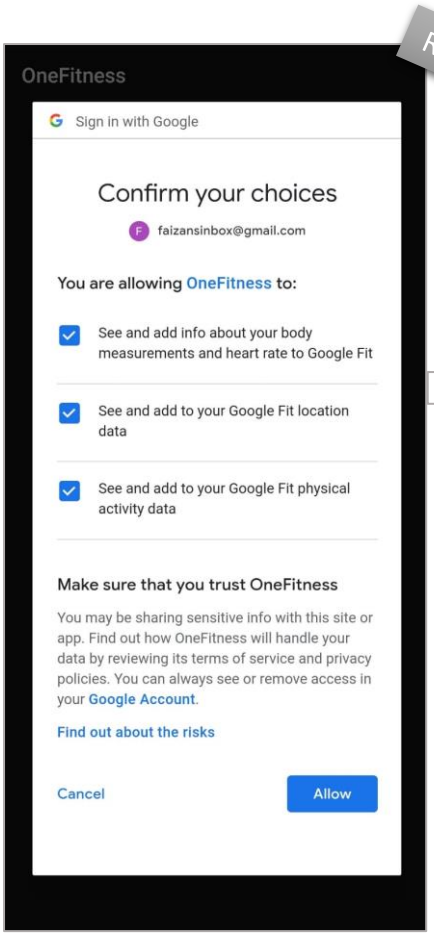
```
.addDataType(DataType.TYPE_ACTIVITY_SEGMENT)
.addDataType(DataType.TYPE_LOCATION_SAMPLE)
.addDataType(DataType.TYPE_SPEED)
.addDataType(DataType.TYPE_POWER_SAMPLE)
.addDataType(DataType.TYPE_MOVE_MINUTES)
.addDataType(DataType.TYPE_LOCATION_TRACK)
.addDataType(DataType.TYPE_WORKOUT_EXERCISE)
.addDataType(DataType.TYPE_CYCLING_PEDALING_CADENCE)
.addDataType(DataType.TYPE_CYCLING_PEDALING_CUMULATIVE)
.addDataType(DataType.TYPE_CYCLING_WHEEL_REVOLUTION)
.addDataType(DataType.TYPE_CYCLING_WHEEL_RPM)
.addDataType(DataType.TYPE_WEIGHT)
```

I decided to ask for more data for sessions from the Google Fit API. Resetting the app and launching it again caused the same error.

Fix

```
.addDataType(DataType.TYPE_ACTIVITY_SEGMENT, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_LOCATION_SAMPLE, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_SPEED, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_POWER_SAMPLE, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_MOVE_MINUTES, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_LOCATION_TRACK, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_WORKOUT_EXERCISE, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_CYCLING_PEDALING_CADENCE, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_CYCLING_PEDALING_CUMULATIVE, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_CYCLING_WHEEL_REVOLUTION, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_CYCLING_WHEEL_RPM, FitnessOptions.ACCESS_WRITE)
.addDataType(DataType.TYPE_WEIGHT, FitnessOptions.ACCESS_WRITE)
```

Upon further research I found out that the default permission only asks to read data. Now that I are recording data and writing it to the API I also need write access for all our permissions.



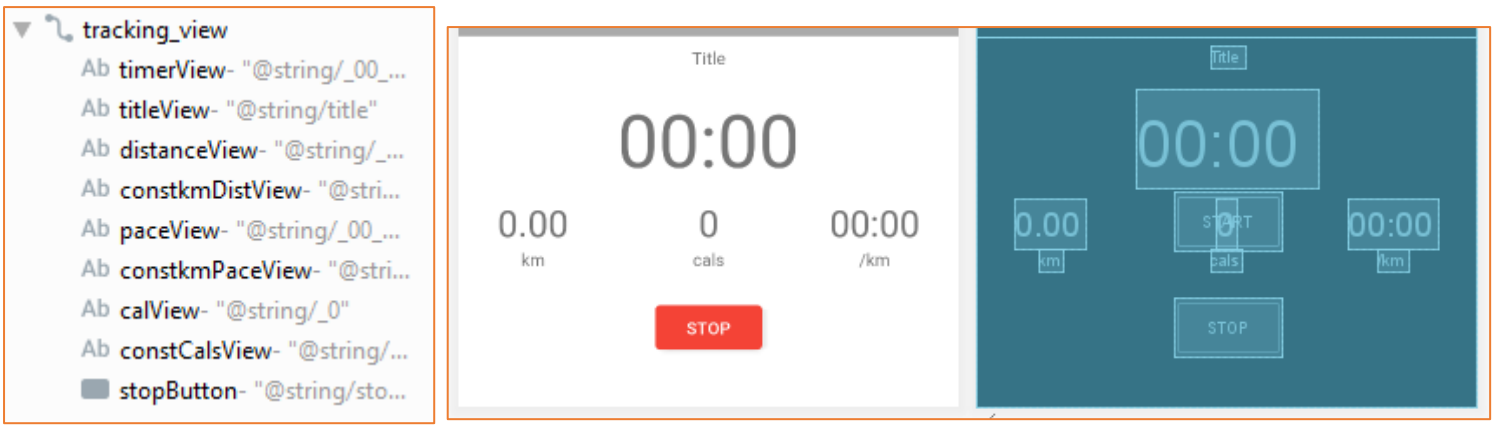
Running the app now asks for write permissions as well as recording data about the user's body such as weight. I will need these permissions for later but also will allow us to fix the exception issue above!

After running this, I now get a response from the session instead of an exception.

```
I/MyFit-RunningSession: [Session{startTime=1577914751545, endTime=1577914762272,
```

Successful! The log now returns the session data instead of an exception, so the issue has been fixed!

Tracking Data View – SessionFragment



I created a constraint layout to the SessionFragment which will consist of the UI for reporting data back to the user as they carry out their session. This is a constraint layout because I want all items to be displayed correctly no matter what the screen size of the user's device is.

The layout consists of a title which will either be Running or Cycling, and a timer which will start as soon as it's displayed. I also put in a distance, calories and pace counter to report back to the user. These are important details for running which I'm focusing on for now. For cycling I will implement speed instead of pace.

Finally I put in a stop button which will be used to stop the session, this will also mean I have to change the behaviour of my app as before I used the same button to stop and start the session.

I set this layout to INVISIBLE and set the start button to visible again. I will show this layout when the user starts the session after the countdown.

```
private ConstraintLayout trackingView; // layout to hold all tracking
private TextView titleText;            // set title
private TextView timerText;            // show timer to user
private TextView distanceText;         // show distance to user
private TextView paceText;             // show pace to user
private TextView caloriesText;         // show calories to user
private Button stopButton;             // allows to stop session
```

Firstly, I made private variables to allow me to access these UI elements.

```
trackingView = (ConstraintLayout) view.findViewById(R.id.tracking_view);
titleText = (TextView) view.findViewById(R.id.titleText);
timerText = (TextView) view.findViewById(R.id.timerText);
distanceText = (TextView) view.findViewById(R.id.distanceText);
paceText = (TextView) view.findViewById(R.id.paceText);
caloriesText = (TextView) view.findViewById(R.id.calText);
stopButton = (Button) view.findViewById(R.id.stopButton);
```

In the onCreateView I get the UI items from the view and assign them to these private variables respectively.

```
// create a listener which will be called when the button is clicked
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startButton.setVisibility(View.INVISIBLE);  // if its not been started then hide the button
        countDownText.setVisibility(View.VISIBLE); // and begin the count down
        startSession(); // start the session as well
    }
});
```

I removed the started private variable as it is no longer needed. I changed the startButton behavior when it is clicked to just start the countdown and hide the start button instead of performing any checks.

```
public void onFinish() {
    countDownText.setVisibility(View.GONE); // remove the counter from the UI

    titleText.setText("Running");    // set title to Running
    startTimer();                    // start timer for session
    trackingView.setVisibility(View.VISIBLE);
    runningSession.startSession();          // start the running session
}
```

I changed the onFinish method on the countdown now to not set the start button to visible again. For now I set the title to Running and start the timer as well as make the layout visible.

```
stopButton.setOnClickListener((v) -> {
        runningSession.stopSession();               // stop tracking session
        trackingView.setVisibility(View.INVISIBLE); // hide the tracking layout
        startButton.setVisibility(View.VISIBLE);    // show the start button

});
```

I set a listener for when the stop button is pressed to stop showing the tracking layout I just implemented and show the start button again. This will be changed to show an overview later as I develop the app. I also call the runningSession stopSession method to stop tracking the actual info for the session.

Track Information

This part of the algorithm in the session will record, calculate and present the user with every relevant data for the session. This includes the distance, calories and time.

For running it will also include the pace which has to be calculated using the distance and time.

Time

Next, I want to implement tracking the time using a timer and also track the distance when the session is started. I will also add a few more things such as tracking the calories and pace of the user which I had not initially thought of when designing the program.

```
private void startTimer() {
    // custom class to start count down
    CountUpTimer timer = new CountUpTimer( durationMs: 30000) {
        public void onTick(int second) {
            timerText.setText(String.valueOf(second)); // update the timer every second
        }
    };
    // start timer
    timer.start();
}
```

The startTimer private method will start the timer for how long the session continues for. This will also show the user how long this session is running for. I had to create a custom class which I called CountUpTimer to handle. This class counts up every second and updates the text to the user. At the end I start this timer.

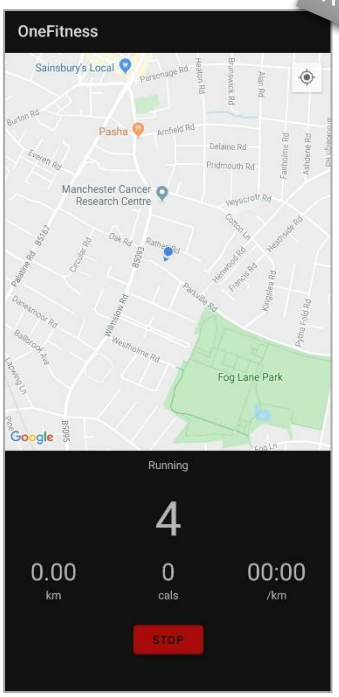
```
public abstract class CountUpTimer extends CountDownTimer {
    private static final long INTERVAL_MS = 1000; // every second
    private final long duration;    // for how long

    protected CountUpTimer(long durationMs) {
        super(durationMs, INTERVAL_MS); // call CountDownTimer with values
        this.duration = durationMs; // set the duration
    }

    public abstract void onTick(int second); // don't implement behaviour for this class

    @Override
    public void onTick(long msUntilFinished) {
        // implement CountDownTimer onTick to keep calling infinitely
        int second = (int) ((duration - msUntilFinished) / 1000);
        onTick(second); // call this methods onTick
    }

    @Override
    public void onFinish() {
        onTick( msUntilFinished: duration / 1000); // call the onTick method again
    }
}
```

I implemented this class as an abstract class so it can be used with different behaviors. Abstract classes force us to implement one or more methods I set using polymorphism. I used this with the Tracker class and use it again here.

This class inherits the CountDownTimer to track the actual time but makes sure there is no duration by infinitely calling the onTick method. One of these methods I override to reset the timer in the parent class and the other will be used to implement the behavior of updating the text in the fragment like shown in startTimer().

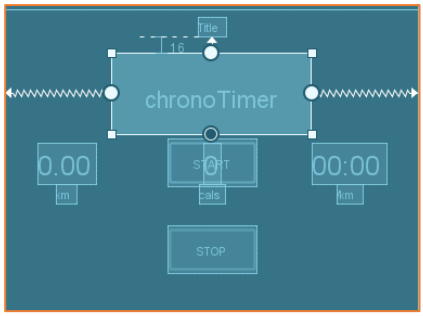
Muhammad Faizan Alam

Run



Starting this timer works just as expected however I have one issue. I want to show the output like a clock and not just the seconds. Seconds will be very hard for the user to interpret after it gets in the many minutes and hours even. Doing further research I found I could use the Chronometer class implemented in Android which will handle a lot of this behavior for me instead of having to implement all that behavior in the CountUpTimer. This means I no longer need that abstract class and so can remove it.

Fix



I replaced the TextView for the timer to a Chronometer in the view.

```
private Chronometer timer;                    // show timer to user
```

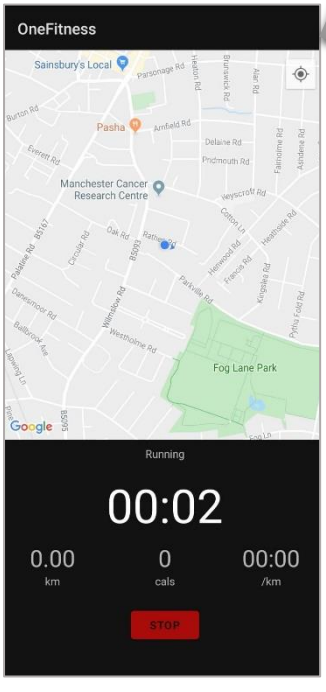
```
timer = (Chronometer) view.findViewById(R.id.chronoTimer); // initiate a chronometer
```

I then replaced the TextView variables in the class with the chronometer.

```
private void startTimer() {
    timer.setBase(SystemClock.elapsedRealtime()); // reset the timer to current time
    timer.start(); // then start from 0
}
```

Finally, I changed the startTimer method to start this chronometer. The chronometer starts the second the fragment starts. To fix this I set the base to the current time so it starts at 0 again when the startTimer is called.

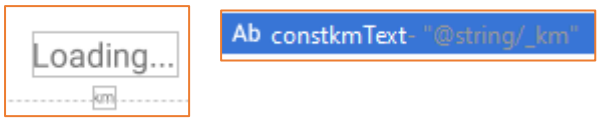
Run



Now I can see the timer is displayed correctly when running for the user to easily see. This is much easier to work with as well and will allow me to easily store it without having to carry out many conversions from seconds like before.

Distance

Next I want to track the distance the user has travelled. Initially I showed the distance in total meters, but I want to change this to kilometers as most of the users using my app will be traveling a lot. I want to show this in the format of 0.00 so will have to figure out how to convert it as well.



First I added a new TextView to show the user that the distance is being shown in kilometres by adding a small text showing “km”

Next I did some research to figure out a way to turn my long output for the distance to the correct format. For this I’m going to use DecimalFormat which can also handles rounding up and down a the number.

```
void sendDistance(float totalDistance);
```

In the MainFragmentInterface I changed the int to a float as I will now be working with the whole distance.

```
float total = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_DISTANCE).asFloat();
```

This meant I had to remove the casting in the DistanceTracker update method. I were no longer passing an int and are now passing a float so I’m going to get the original float value.

```
final private DecimalFormat df = new DecimalFormat( pattern: "0.00");
```

In the MainFragment class I made a private variable to handle the formatting of this value when I assign it to the TextView. I pass it a “0.00” so if there are no numbers that will be the default format it will take. This means it will always be rounded to two decimal points. This variable is final which means it can only be assigned to once. This means I cannot change the formatting which is what I want so the format stays consistent. This is one of the features of encapsulation and OOP.

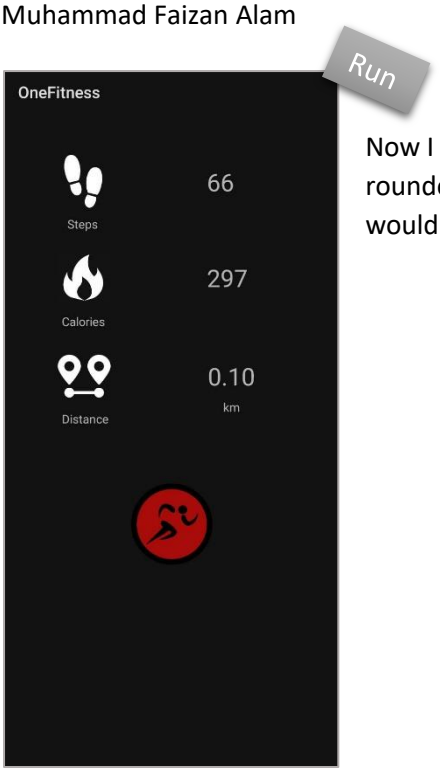
```
@Override
public void sendDistance(float distance) {
    distanceCountText.setText(df.format( number: distance / 1000)); // update UI element with total
    sharedPrefs.edit().putFloat(DISTANCE_TAG, distance).apply(); // save the distance
}
```

In the MainFragment class I have to change the way the text is changed. First I get the distance and divide it by 1000 to convert the meters to kilometres. Next this is passed to the DecimalFormat class which rounds it for us and puts it in the format of 0.00. Now instead of storing the int I are storing a float so putInt is changed to putFloat.

```
float distance = sharedPrefs.getFloat(DISTANCE_TAG,  defValue: 0);
```

```
distanceCountText.setText(df.format( number: distance / 1000)); // update UI element with total
```

To finish it off I also have to getFloat and not use the getInt as now I are storing a floating point number in memory. This means my test devices have to remove all storage as it will crash trying to get a float from an integer that’s been stored.



Now I can see the distance is 97.5 meters in the log. This is converted to 0.10 as it is rounded up to two decimal points. This now looks much cleaner than before when I would have a large value. It also shows the user what unit I use for distance.

```
private void startTimer() {
    timer.setBase(SystemClock.elapsedRealtime()); // reset the timer to current time
    timer.start(); // then start from 0

    // update info every second
    timer.setOnChronometerTickListener(new Chronometer.OnChronometerTickListener() {
        @Override
        public void onChronometerTick(Chronometer chronometer) {
            // calculate the seconds elapsed by getting difference since started
            int elapsedSeconds = (int)(SystemClock.elapsedRealtime() - chronometer.getBase()) / 1000;

            if (elapsedSeconds % 10 == 0) { // every 10 seconds
                // get distance
                runningSession.getDistance();
            }

        }
    });
}
```

Next, I want to track the distance when a running session is started. In the SessionFragment I set the listener for the chronometer to update the distance every 10 seconds. I do this by checking when the timer was started and then checking the current time. Taking these away shows how many milliseconds has elapsed. I divide this by 1000 next and convert it to an integer so it is truncated to the second.

The modulus sign returns the remainder so I use it to check if it is divisible by 10 and if so I update the distance using the getDistance method. This means every 10 seconds the getDistance method is called. This allows us to save on resources instead of updating the distance every second and making API calls as well as provide a smoother experience to the user.

```
public interface SessionFragmentInterface {
    void sendDistance(float totalDistance);

    //void sendPace(long totalSteps);

    //void sendCalories(int totalCalories);


}
```

Because I want to handle all the UI updates in the Fragment class which will lower the risk of crashes when the session is running in the background I setup an interface class. For now I only declare the sendDistance method and comment the other two as I will implement them later.

```
public class SessionFragment extends Fragment implements OnMapReadyCallback, SessionFragmentInterface
```

```
final private DecimalFormat df = new DecimalFormat( pattern: "0.00"); // convert distance
```

```
@Override
public void sendDistance(float totalDistance) {
    distanceText.setText(df.format( number: totalDistance / 1000)); // update UI element with total
}
```

First I make the SessionFragment implement this interface. I then make a private variable which is final so it cannot be assigned to again. This DecimalFormat variable will be used to convert the distance to a “0.00” format so the user gets a consistent layout throughout the app.

The sendDistance method simply updates the distance TextView UI element with the distance covered that is tracked by the RunningSession class which I will implement next. I divide the distance by 1000 so it is converted from meters to kilometres.

```
public void getDistance() {
    // make a read request to get get the distance between when the session was started and the current time
    DataReadRequest readRequest = new DataReadRequest.Builder()
            .setTimeRange(session.getStartTime(TimeUnit.MILLISECONDS), System.currentTimeMillis(), TimeUnit.MILLISECONDS)
            .read(DataType.TYPE_DISTANCE_DELTA)
            .build();
```

In the getDistance method I first create a readRequest type which holds the information for what data I want to get. I set the time range between when the session was started to the current time so it only gets the distance covered in that period by the user. It reads the distance change between those two time periods and finally I build the request. I pass the TimeUnitMilliseconds so all the times are handled precisely using milliseconds.

```
Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context)) // get the historyClient
        .readData(readRequest)     // read distance covered since session was started
        .addOnSuccessListener(                          // listener executed when method succeeds
                new OnSuccessListener<DataReadResponse>() {

                    @Override
                    public void onSuccess(DataReadResponse dataReadResponse) {
                        // get the distance data from the response from the API
                        DataSet dataSet = dataReadResponse.getDataSet(DataType.TYPE_DISTANCE_DELTA);
                        // get the distance covered as a float. If it's empty get the value 0
                        float distance = dataSet.isEmpty() ? 0 : dataSet.getDataPoints().get(0).getValue(Field.FIELD_DISTANCE).asFloat();

                        updateUIInterface.sendDistance(distance); // update UI

                        Log.i(TAG, msg: "Distance response " +  distance);    // log the distance
                    }
                })
        .addOnFailureListener( // when method fails
                new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Log.i(TAG, msg: "Error: " +  e); // show error
                    }
                });
```

Validation

Straight after I use the HistoryClient once again to retrieve our data. I pass it the readRequest created above to get the distance between those two time periods. I use validation to make sure I don't use any invalid data as that could cause a crash. If the response is successful I get the data for the distance that I are interested in and convert that data to a float. If the data set however is empty I assign the float a 0 so there is no crashes once again and just set it to 0 as no distance has been covered. This validation keeps the app from crashing with an exception as no distance returns an empty DataSet.

Finally, I update the UI similar to the Tracker class and call the and log it for debugging purposes. I need to implement the interface for this call:

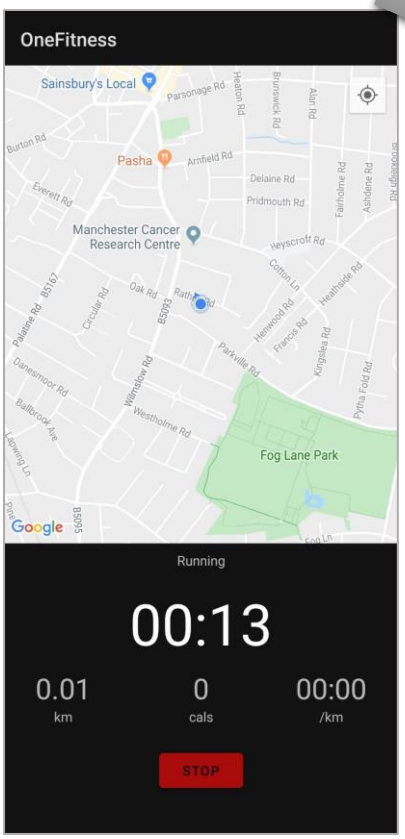
```
private SessionFragmentInterface updateUIInterface;
```

```
public RunningSession(Context activity, SessionFragment fragment) {
    this.context = activity;
    this.updateUIInterface = fragment;
}
```

```
runningSession = new RunningSession(getContext(),  fragment: this);
```

This was simple to do. I declared a private variable for the interface class. I then assigned this in t he constructor and finally passed the SessionFragment when instantiating the RunningSession.

Run



Now when I run the app and walk around a little, I are shown the distance updating every 10 seconds as I just implemented. It is also shown in the correct format so my app is working exactly how it should be this far with no errors. The distance update methods are only called every 10 seconds and I confirmed this through debugging so slower devices are not slowed down because of it.

Run

Error

```
MyFit-RunningSession: Distance response 0.0
MyFit-RunningSession: Distance response 12.52416
MyFit-RunningSession: Distance response 18.500528
MyFit-RunningSession: Distance response 18.500528
MyFit-RunningSession: Distance response 18.500528
MyFit-RunningSession: Distance response 18.500528
```

Justification

However, there is one issue. After 4-5 updates the distance returned no longer changes as you can see it freezes at 18 meters. After extensive research the issue seems to be with the API and not any of my code. I want to fix this issue and have tried multiple solutions. One of these solutions was getting the longitude and latitude location directly from the GPS every 10 seconds and working out the distance with that. However, this solution was not always accurate and also was very resource intensive. Another solution I tried was just getting the distance every 10 seconds and accumulating it. Above I get the distance from when the session starts but if I just store two times in milliseconds and update them as time passes by, I can essentially get the distance travelled in that time and accumulate it. This solution solved all the issues with my GPS solution and solved the issue with the API returning the same value endlessly.

```
private float totalDistance = 0; // hold total distance by accumulating it
private Date lastUpdateTime;     // hold the time from last api request
private Date newUpdateTime;      // hold the current time
```

First, I created three private variables to hold the total distance, the time when the last update occurred and the current time for when an update is to occur. I will explain how these variables are used below.

```
lastUpdateTime = startTime;
```

At the end of the startSession() method I have to set the lastUpdateTime to the start time of the session. This is so I can get the distance from the beginning of the session when I are updating the distance.

Fix

```
// get the current time
newUpdateTime = Calendar.getInstance().getTime();

// make a read request to get get the distance between the two time intervals
DataReadRequest readRequest = new DataReadRequest.Builder()
        .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
        .read(DataType.TYPE_DISTANCE_DELTA)
        .build();

// update the last update to current time
lastUpdateTime = newUpdateTime;
```

The getDistance method is updated to use these new variables. Firstly I assign the current time to the newUpdateTime. This variable is then used in the readRequest as the end time for when I want to get the data. The lastUpdateTime is used as the start time for the update. This means I get the distance update in 10 second intervals now instead of from the start of the session which was giving us issues above. Lastly, I update the lastUdpateTime to the newUpdateTime as this is when the last update for the distance occurred.

```
totalDistance += distance; // update the total distance
updateUIInterface.sendDistance(totalDistance); // update UI with total distance

Log.i(TAG,  msg: "Current Distance response " +  distance);    // log the distance
Log.i(TAG,  msg: "Total Distance response " +  totalDistance);    // log the total distance
```

I can no longer update the UI with the distance the call returns as it will only return the distance for 10 second intervals only. I use the totalDistance member variable to hold the total distance by adding the distance every time it is fetched. I pass this to the UI interface so the total distance is displayed now.

Run

```
Current Distance response 0.0
Total Distance response 26.91445
Current Distance response 2.4484518
Total Distance response 29.362902
Current Distance response 0.36609662
Total Distance response 29.728998
Current Distance response 2.3909783
Total Distance response 32.119976
Current Distance response 3.9744534
Total Distance response 36.09443
```

Now when I run the app and look at the log it does not freeze at a certain value and continues to update forever!

Fixed!

Calories

Next, I want to track the calories. This should be like distance tracking above with a few changes. I believe I can optimise the API calls by joining the distance and calories calls together.

```
public interface SessionFragmentInterface {
    void sendDistance(float totalDistance);

    //void sendPace(long totalSteps);

    void sendCalories(float totalCalories);
}
```

In the SessionFragmentListener I uncomment the method to update the UI which I will call from the RunningSession class.

```
@Override
public void sendCalories(float totalCalories) {
    caloriesText.setText(String.valueOf(Math.round(totalCalories))); // update UI element with total
}
```

I want to show the calories whole number and not in the format of a decimal, so I use Java's Math class to round it to the nearest integer and then convert that integer to a string by using String.valueOf() method. This is then assigned to the UI element for calories.

```
private float totalCalories = 0; // hold the total calories
```

Like totalDistance I declare a private variable to hold the total calories which is initialised to 0.

```
// make a read request to get get the distance between the two time intervals
DataReadRequest readRequest = new DataReadRequest.Builder()
        .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
        .read(DataType.TYPE_DISTANCE_DELTA)
        .read(DataType.TYPE_CALORIES_EXPENDED)
        .build();
```

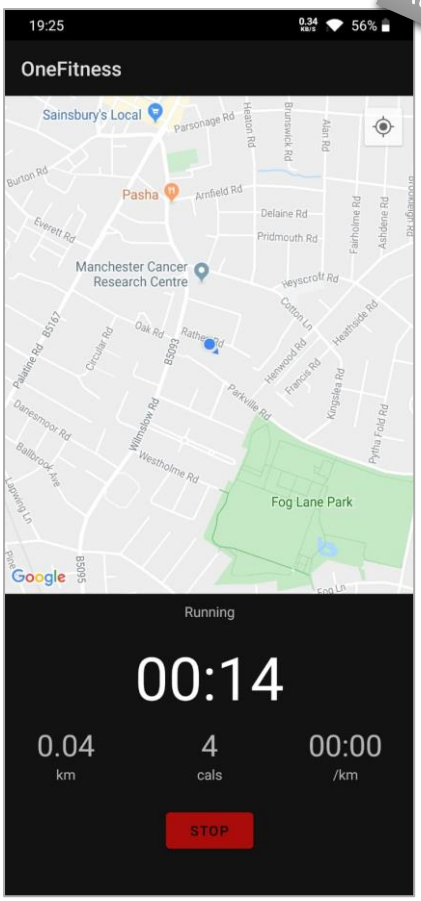
The read request to the phone's sensor API is updated to retrieve the CALORIES_EXPENDED type as well so I can parse that data for total calories.

```
// get the calories data from the response from the API
DataSet caloriesData = dataReadResponse.getDataSet(DataType.TYPE_CALORIES_EXPENDED);
// get the caloires covered as a float. If it's empty get the value 0
float calories = caloriesData.isEmpty() ? 0 : caloriesData.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).asFloat();

totalCalories += calories; // update the total calories
updateUIInterface.sendCalories(totalCalories); // update UI with total calories
```

In the onSuccess listener I add a very similar code that I used to retrieve the distance. First, I get the data set for all the calories expended by the user in that time frame. Then I check if there were any calories burned, if not I assign it the value 0 else, I get the calories field from the type as a float and assign it to the calorie's variable.

I update the totalCalories by adding calories to it using the += operator.Then I update the UI with the calories so the user can see their expenditure.

Run



Error

```
Current Distance response 0.0
Total Distance response 35.45438
Current Calories response 1.2301251
Total Calories response 5.3976626
```

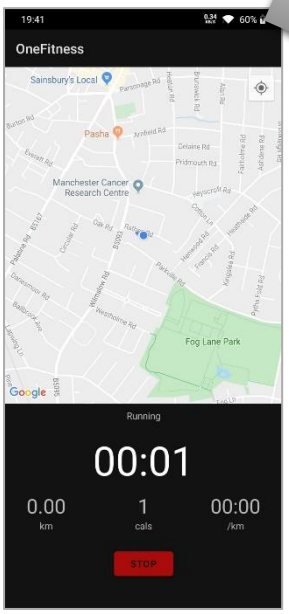
Running the app now updates the calories and they're tracked correctly just like distance now. However from starting and stopping sessions the UI is updated with the last session's distance and calories again. I quickly realise that I do not reset the values of totalDistance and totalCalories to 0 after a session has ended.

Fix

```
totalCalories = 0;
totalDistance = 0;
```

I add the following to the startSession() method so the two values start with 0 when starting a session. This solved the issue as I review the log again and start another session

Run



The distance and calories are reset now and the issue with them staying persistent is fixed! The calories and distance from the previous session are not loaded as you can see the values are 0.

Pace

To calculate the pace I first have to get the speed in meters per second and then divide 16.66… by the speed which will give me the number of minutes it takes to cover a kilometre. This equation is has to be implemented into my algorithm to work out the correct data and update it every time the distance is updated, resulting in accurate data.

```
private float pace = 0;           // hold the pace
```

```
// make sure distance is not 0 so we don't have 0 in a division
if (totalDistance > 0) {
    // get the time since the session has started
    long time = newUpdateTime.getTime() - session.getStartTime(TimeUnit.MILLISECONDS);
    // calculate the speed by dividing total distance by the total time
    float speed = totalDistance / (time / 1000);
    // calculate the pace by dividing 16.66 by the speed(m/s)
    pace = 16.66666666666f / speed;
    updateUIInterface.sendPace(pace); // update UI
}
```

Validation

In the success listener for retrieving distance I first check that the total distance is greater than 0 so I do not divide anything by 0 as the speed will also be 0. This validation avoids run time crashes and errors from occurring.

First, I get the time since the session has started by taking away the session's start time from the current time. Next I get the speed by dividing the total distance covered by the time in seconds by dividing it by 1000. To get the pace I simply must divide 16.66… by the speed which is in meters per second. I add the postfix of f to convert the 16.6… to a floating number so it can be divided without any conversion to a double. This is then assigned to the private variable pace. Finally I update the UI by sending it the pace.

```
public interface SessionFragmentInterface {
    void sendDistance(float totalDistance);

    void sendPace(float pace);

    void sendCalories(float totalCalories);
}
```

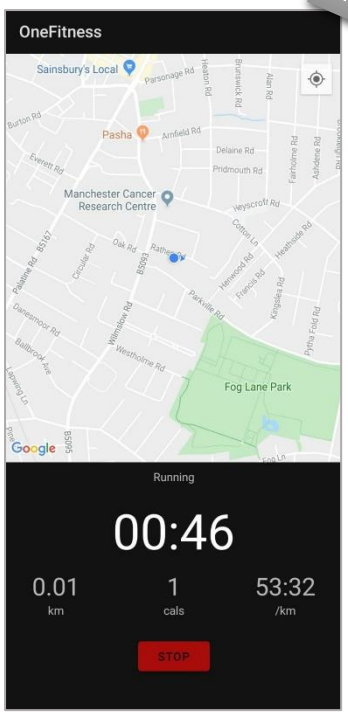
The interface now has all three methods uncommented so I have to implement the sendPace method now.

```
final private SimpleDateFormat sdf = new SimpleDateFormat( pattern: "mm:ss"); // convert time
```

```
@Override
public void sendPace(float pace) {
    // make sure pace is less than an hour
    if (pace < 60) {
        // convert the pace to milliseconds
        long timeInMilliSeconds = (long) Math.floor(pace * 60 * 1000);
        // create a date with the time in ms
        Date date = new Date(timeInMilliSeconds);
        // set pace in correct format mm:ss
        paceText.setText(sdf.format(date));
    }
}
```

In the SessionFragment class I first make sure the pace is smaller than 60 so I ignore whole hours. I then convert the pace to milleseconds which is used to assign a Date type. This Date type is easier to work with and display on screen. I then set this paceText with the time in the format mm:ss by using the sdf private variable.

Run



Running the app now records the pace as well and shows it to the user just like I implemented above. I now have all three pieces of information being tracked and updated accordingly. The next step is to track the GPS location of the user so I can draw a track on a map for the session.

Validation Testing

As the pace is updated and shown correctly, it means the validation for the data has worked correctly as no crashes have occurred. The value is also in range meaning the validation has not calculated something out of range and invalid.

Route

To track the route of the user I need to store the longitude and latitude of the user every 10 seconds when the other information is updated. This will allow me to make a rough path of the user to show in the summary as well as store it to be viewed later on.

With some research I found that the maps can draw a polyline using longitude and latitudes stored in a PolylineOptions data type:
https://developers.google.com/android/reference/com/google/android/gms/maps/model/PolylineOptions.html

```
private PolylineOptions track; // hold coordinates to draw
```

In the RunningSession I created a private variable to allow longitude and latitude to be added as the session

```
track = new PolylineOptions();
track.color(Color.rgb( red: 244,  green: 67,  blue: 54));
track.width(20);
track.endCap(new RoundCap());
```

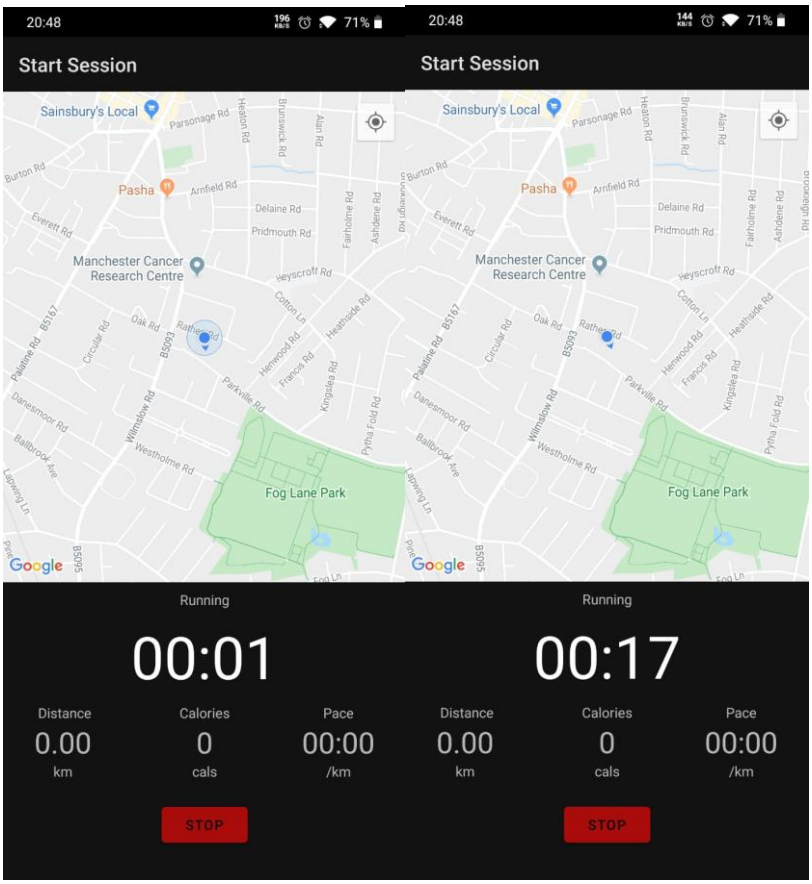
In the startSession method I assign the track a new PolylineOptions so there is a new track being stored with every new session. I set the track colour to the main colour of the app using the RGB values. Then I increase the width of the line drawn on the map so it is easier to view. I set the end bit of the tracked line to round so it looks nicer without having a jagged boxed cut off.

```
private void trackLocation() {
    // client run in the background to not slow down GUI when getting location
    FusedLocationProviderClient fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(context);

    // start background task
    Task<Location> task = fusedLocationProviderClient.getLastLocation();
    // validation: wait for task to complete before moving camera
    task.addOnSuccessListener((OnSuccessListener) (location) → {
            // get coordinates of the current location
            LatLng coordinates = new LatLng(location.getLatitude(), location.getLongitude());
            track.add(coordinates);
            Log.i(TAG,  msg: "Location Tracked: " + coordinates.toString());
    });

}
```

I created a private method to handle the location tracking when it is called in the RunningSession. I use a location provider to get the current longitude and latitude values of the user. I use validation to wait for when the location is retrieved before processing it so no null values are accessed. This prevents the app from crashing. When the location is retrieved it gets the coordinates of the location and stores them in a LatLng type. These coordinates are then added to the track variable I created above.

```
trackLocation();
```

This method is then called in the UpdateData method, so it retrieves the user's location every 10 seconds along with all the other data and then stores the location.



As you can see above the direction of the user changes when I point the device in a different direction. This means the user's location is being tracked and saved.

I now have the user's location being tracked, the next step is to create a fragment to show the summary and their tracked data when a session has been ended.

Module 2 - End Session Summary

I have already tracked calories above so do not need to do that when the session ends. The focus for the summary fragment view will be to pass all the tracked info and show the user a summary of the session alongside a map view with the user's previous locations while the session as running.

First, I created a new Fragment class alongside a view which is shown above. This view consists of a map view at the top half of the screen. This map will be used to show the user's track they travelled during a session. Next I show the Title of the session with the data which will show the start and end time. I also will show the total distance, calories and the pace of the user. This summary fragment will be used later in version 3 with the log so I want to make it as modular as possible. I will also use this view to show a cycling session that I have not implemented yet.

Retrieve Session

I now need to get the session data from the SessionFragment when it has ended. The data is must be passed so it can be used to display the review page for the user with distance, calories and time as well as pace. I will change it to speed later for the cycling session.

```
public static final String ARG_SESSION = "SESSION";
```

```
public static SessionSummaryFragment newInstance(RunningSession ses) {
    SessionSummaryFragment fragment = new SessionSummaryFragment();

    Bundle args = new Bundle();
    args.putSerializable(ARG_SESSION, ses);

    fragment.setArguments(args);
    return fragment;
}
```

I only want one instance of the summary fragment so there is no clashing in terms of what data and session is passed when showing the view. This allows us to pass arguments from the parent fragment so I can access data from it.

First I create a new instance of the SessionSummaryFragment which will returned when the data is received.

The runningSession will be a Serializable class as I will implement that interface later to allow the whole class to be passed by the Fragment Manager. I get the session from the arguments by passing a constant and assign that to the RunningSession ses.

Justification

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        // assign the RunningSession to the private varaible
        session = (RunningSession) getArguments().getSerializable(ARG_SESSION);
    }
}
```

In the onCreate method I must make sure I have arguments that have been passed by the parent fragment before assigning them to the private vairaible. This validation prevents any crashing if a null session were to be passed.

```
ssion implements Serializable {
```

To allow the RunningSession to be passed by the FragmentManager as an argument it must implement Serializable

```
Bundle args = new Bundle();
args.putSerializable(SessionSummaryFragment.ARG_SESSION, session);
//args.putString(SessionSummaryFragment.ARG_DISTANCE, session.getDistance());
SessionSummaryFragment fragment = new SessionSummaryFragment();
fragment.setArguments(args);
FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
        .replace(R.id.fragment_container, fragment, tag: "summary") // replace the current fragment
        .addToBackStack(null);  // add this fragment to the stack

fragmentTransaction.commit(); // execute the fragment call created above
```

In the SessionFragment I need to pass the sesion from this fragment to the SessionSummaryFragment so the data can be shown and worked on.

To do this I create a new Bundle which essential holds different values that it passes when a new fragment is started. I add the running session to the bundle as a serializable so it can be converted after in the summary fragmnent. Afterwards I initiate a framgentTransaction similar to the first fragment but call the SessionSummaryFragment with the arguments passed through. I can now work with the session in the new fragment.

When displaying the data I want it to be handled by the Session instead of the SessionFragment so it is consistent between different fragments.

```
@Override
public void sendDistance() {
    distanceText.setText(session.getDistance()); // update UI element with total
}

@Override
public void sendSpeed() {
    speedText.setText(session.getSpeed()); // update the speed/pace
}

@Override
public void sendCalories() {
    caloriesText.setText(session.getCalories()); // update UI element with total
}
```

I created get_() methods for the data which are then used in fragments to retrieve the string in correct format. This will allow me to show the same data for both fragments.

```
final protected DecimalFormat df = new DecimalFormat( pattern: "0.00"); // convert distance
final protected SimpleDateFormat sdf = new SimpleDateFormat( pattern: "mm:ss"); // convert time
```

```
public String getDistance() {
    return df.format( number: totalDistance / 1000); // return distance
}
```

The formatting for the distance, speed and calories is now handled in the runningSession. Please note I changed the pace to speed as I will be using polymorphoism to implement cycling later.

I also implement other get methods to retrieve data for the fragments.

```
public PolylineOptions getTrack() {
    return track;
}

public String getName() {
    if (session != null) { // make sure a session has started
        return session.getName();
    }
    return "";
}

public long getStartTime() {
    if (session != null) { // make sure a session has started
        return session.getStartTime(TimeUnit.MILLISECONDS);
    }
    return -1;
}

public long getEndTime() {
    if (session != null) { // make sure a session has started
        return session.getEndTime(TimeUnit.MILLISECONDS);
    }
    return -1;
}
```

These get methods validates that data is available before returning data which allow safe access to private varaibles.

Map

I want to show the route the user travelled when they finish a session which means I have to setup a Google Map view first. Once it is setup I can use related methods to draw the track on it.

```
// Inflate the layout for this fragment
mapView = (MapView) view.findViewById(R.id.mapView); // init mapView fragment
mapView.onCreate(savedInstanceState);
mapView.onResume();                                  // if there is a previous state
mapView.getMapAsync( onMapReadyCallback: this);
```

In the onCreateView method I need to initialise the UI to a start state. I set the mapView to that of Google Maps. If there is a previous state of this map I retrieve that else a new one is created.

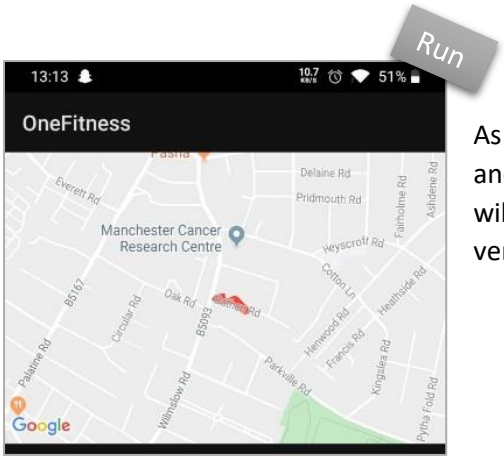
```
public class SessionSummaryFragment extends Fragment implements OnMapReadyCallback {
```

```
@Override
public void onMapReady(GoogleMap googleMap) {
    map = googleMap;
    map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
    // make map constant
    map.getUiSettings().setAllGesturesEnabled(false);
    showTrack();
}
```

I Implement OnMapReady interface so I can use the above method to modify the map when it is ready to use. This is a form of validation so the map is not accessed while it is initialising.

```
private void showTrack() {
    // move camera to first location tracked
    moveCamera(session.getTrack().getPoints().get(0));
    // draw a line of all the locations tracked
    map.addPolyline(session.getTrack());
}
```

The showTrack method moves the camera to the first location of the user in the session and then draws a line representing the locations tracked during the session.



As I can see the tracking of the user now works. I walked around the street and came back which you can see on the map when the session is ended. I will do further testing at a different location at the end of this version to verify everything is working, which I believe should be!

Show Session Data Review

Now that I have the session as well as the route drawn on the map, I can show update and show the rest of the data in the session. This will consist of the distance, calories and also calculating the time using the start and end time.

```
// get all UI items
titleText = (TextView) view.findViewById(R.id.titleText);
dateText = (TextView) view.findViewById(R.id.dateText);
distanceText = (TextView) view.findViewById(R.id.distanceText);
speedText = (TextView) view.findViewById(R.id.speedText);
caloriesText = (TextView) view.findViewById(R.id.calText);
```

I need to update the rest of the UI so first I have to get the UI elements so I can work with them programmatically.

```
// update the UI texts with session info
titleText.setText(session.getName());
distanceText.setText(session.getDistance());
speedText.setText(session.getSpeed());
caloriesText.setText(session.getCalories());
```

I then set all this data by retrieving the relevant data stored in the session by calling the methods which return a consistently formatted data.

```
private final SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy");
private final SimpleDateFormat timeFormat = new SimpleDateFormat( pattern: "hh:mm");
```

```
private void setDate() {
    // gets the start and end time in the format dd/MM/yyyy, hh:mm - hh:mm
    String date = dateFormat.format(session.getStartTime()) + ", "
                + timeFormat.format(session.getStartTime()) + " - " + timeFormat.format(session.getEndTime());
    dateText.setText(date);
}
```

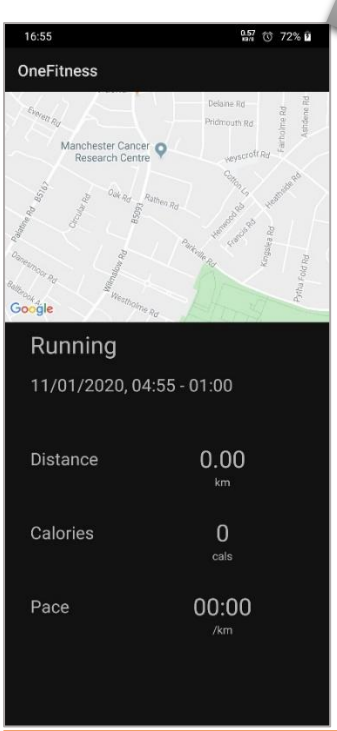
The setDate method retrieves the start time and end time and formats it to show it in the format shown below. It shows the start time and end time. The SimpleDateFormat class handles the format I pass to it for us without having to work directly with milliseconds making it easier to show in the correct format to the user. This class will simplify my code and reduce the number of errors that otherwise might be present with handling time.

Justification

Format it will be shown in is:

dd/MM/yyyy, hh:mm – hh:mm

Muhammad Faizan Alam



Run

Error

Running the app and starting a session and stopping it shows the summary fragment. Everything in the summary fragment is correct except the date. It shows the start time as AM instead of the time in PM as is the mobile. It then shows the end time as 1 AM which is also wrong. I need to fix this issue.

Fix

```
private final SimpleDateFormat timeFormat = new SimpleDateFormat( pattern: "HH:mm");
```

To fix the start time it was just a matter of requesting the time in 24 hours by changing the formatting to capital HH.

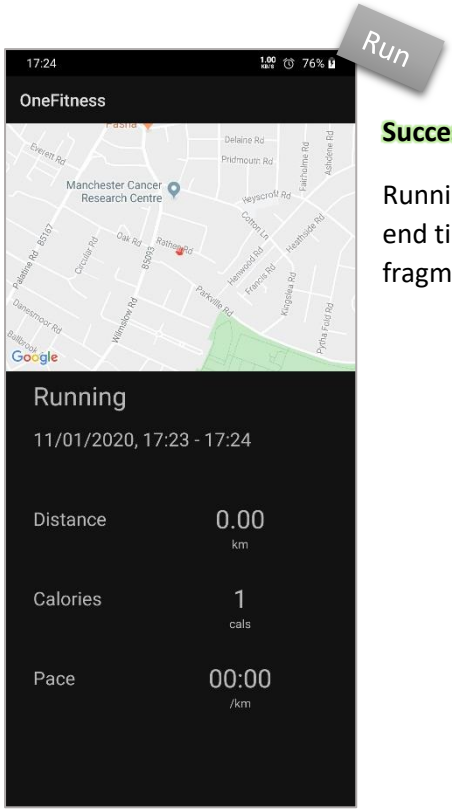
```
MyFit-SessionSummary: 17:07
MyFit-MySession: 0
MyFit-MySession: [Session{startTime=1578762432394, endTime=1578762457158,
```

From debugging the app I see the end time is always returning 0. This is because the UI is updated before the session can end.

Fix

```
private void setDate() {
    // get end time from session
    long endTime = session.getEndTime();
    // if session is just ending
    if (endTime == 0) {
        // set the end time to the current time
        endTime = System.currentTimeMillis();
    }
    Log.i(TAG,  msg: "End Time: " + String.valueOf(endTime));

    // gets the start and end time in the format dd/MM/yyyy, hh:mm - hh:mm
    String date = dateFormat.format(session.getStartTime()) + ", "
                + timeFormat.format(session.getStartTime()) + " - " + timeFormat.format(endTime);
    dateText.setText(date);
}
```

To fix the end time I researched and debugged a few methods. The issue was I could not update the UI from the stop listener in the session and therefore had to handle the end time in the SessionSummaryFragment. I fixed it by checking if the end time was 0. This shows me the session is just ended now and therefore I can use the current time as the end time. This algorithm simply converts the start and end time to a range.



Successful!

Running the app now and starting a session for 1 minute shows the correct start and end time as I fixed them above. This means I are now finished with the summary fragment and can also use it in the future to show previous sessions.

Module 3 - Cycling Session

Justification

Because I have implemented many of the features for starting and stopping a session as well as tracking information, I can use OOP's polymorphism to create a cycling session which will handle any of the differences between these two sessions. The distance and calorie tracking will stay the same with both as well as how they're shown on the screen.

One of the changes will be to show the speed in km/h instead of the pace which was shown in m/km. This means I will need to update the constant labels as well.

MySession Class

```
public abstract class MySession implements Serializable {
```

Class is abstract as it will have abstract methods.

```
private static final String TAG = "MyFit-MySession";  // used for debugging
final protected DecimalFormat df = new DecimalFormat( pattern: "0.00"); // convert distance
final protected SimpleDateFormat sdf = new SimpleDateFormat( pattern: "mm:ss"); // convert time

protected Context context;    // get parent activities context to pass into methods

private String sessionName;
protected SessionFragmentInterface updateUIInterface;

protected Session session; // tracking session data

protected Date lastUpdateTime;    // hold the time from last api request
protected Date newUpdateTime;     // hold the current time

protected float totalDistance = 0; // hold total distance by accumulating it
protected float totalCalories = 0; // hold the total calories
```

First I copied over many of the private variables from the RunningSession to a new class called MySession. These private variables will hold the data which is shared by both running and cycling sessions. I added a variable to hold the name of the session.

```
public MySession(Context activity, SessionFragment fragment, String name) {
    this.context = activity;
    this.updateUIInterface = fragment;
    this.sessionName = name; // get session type using name
}
```

The constructor is the same except it also assigns the name to either Running or Cycling

```
session = new Session.Builder()
        .setName(sessionName) // set the name
        .setActivity(activityType) // session is from current app
```

```
String activityType = FitnessActivities.RUNNING; // set default activity to running
// if session is cycling
if (sessionName == "Cycling") {
    activityType = FitnessActivities.BIKING; // change session type to cycling
}
```

The name in Google Fit API is set to either Cycling or Running however I have to set the type for the activity as well. I do this my initially having a string hold the Running activity type but change it to cycling if the name of the session is cycling as shown above.

The stopSession, trackLocation and other get methods stay exactly the same and the startSession is the same as well except for the line above. Instead of the hardcoded name of “Running” I set it to the value that is retrieved in the constructor.

```
protected void getBaseData(DataReadRequest readRequest) {
    Fitness.getHistoryClient(context, GoogleSignIn.getLastSignedInAccount(context)) // get the historyClient
            .readData(readRequest)     // read distance covered since session was started
            .addOnSuccessListener(                          // listener executed when method succeeds
                    (OnSuccessListener) (dataReadResponse) → {
                        // get the distance data from the response from the API
                        DataSet distanceData = dataReadResponse.getDataSet(DataType.TYPE_DISTANCE_DELTA);
                        // get the distance covered as a float. If it's empty get the value 0
                        float distance = distanceData.isEmpty() ? 0 : distanceData.getDataPoints().get(0).getValue(Field.FIELD_DISTANCE).asFloat();

                        totalDistance += distance; // update the total distance
                        updateUIInterface.sendDistance(); // update UI with total distance


                        // get the calories data from the response from the API
                        DataSet caloriesData = dataReadResponse.getDataSet(DataType.TYPE_CALORIES_EXPENDED);
                        // get the calories covered as a float. If it's empty get the value 0
                        float calories = caloriesData.isEmpty() ? 0 : caloriesData.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).asFloat();

                        totalCalories += calories; // update the total calories
                        updateUIInterface.sendCalories(); // update UI with total calories

                        updateCustomData(dataReadResponse);


                        Log.i(TAG,  msg: "Current Distance response " +  distance);    // log the distance
                        Log.i(TAG,  msg: "Total Distance response " +  totalDistance);    // log the total distance
                        Log.i(TAG,  msg: "Current Calories response " +  calories);    // log the calories
                        Log.i(TAG,  msg: "Total Calories response " +  totalCalories);    // log the total calories

                    })
            .addOnFailureListener( // when method fails
                    (e) → {
                        Log.i(TAG,  msg: "Error: " +  e); // show error
                    });

    trackLocation();
}
```

I renamed the getData class to getBaseData for the MySession class. This class handles retrieving the data shared by both running and cycling classes. This method is called with a readRequest which will be created in the other two classes so different data can be tracked as well. It is the same as that of the RunningSession but the pace and it’s relevant data are removed.

```
public abstract void updateData();
```

The update method is abstract which will be implemented by the running and cycling sessions so

```
protected abstract void updateCustomData(DataReadResponse dataReadResponse);
```

I created an abstract method in the MySession class so it must be implemented by any children classes such as the running and cycling classes. This will allow the two classes to handle retrieving specific data that is not shared between the two such as the speed and pace.

```
void sendSpeed();
```

```
public abstract String getSpeed();
```

I changed the pace names in the MySession class and SessionFragment Interface to speed so it is more appropriate as I will be handling the speed for cycling now as well.

Updating RunningSession

```
public class RunningSession extends MySession {
```

First thing I want to do is update the RunningSession as I can now remove a lot of duplicate code. I first extend the MySession class so it's now a child of that class using polymorphism allows us to simplify our code usage.

```
private float pace = 0;          // hold the pace
```

The only extra piece of data I will need to track in cycling is the pace so I create a new variable for that.

```
public RunningSession(Context activity, SessionFragment fragment) {
    super(activity, fragment,  name: "Running");
}
```

The RunningSession has to let the parent know what type of session it is so I pass the string Running in the constructor.

```
@Override
public void updateData() {
    // get the current time
    newUpdateTime = Calendar.getInstance().getTime();

    // make a read request to get get the distance between the two time intervals
    DataReadRequest readRequest = new DataReadRequest.Builder()
            .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
            .read(DataType.TYPE_DISTANCE_DELTA)
            .read(DataType.TYPE_CALORIES_EXPENDED)
            .build();

    // update the last update to current time
    lastUpdateTime = newUpdateTime;


    getBaseData(readRequest);
}
```

The updateData method will create the read request to the API. This method is not implemented in the parent class because it running and cycling may have different data I might want to request in the future. For now it requests the same exact data as the previous RunningSession and class the getBaseData to retrieve the data and handle it in the parent class.

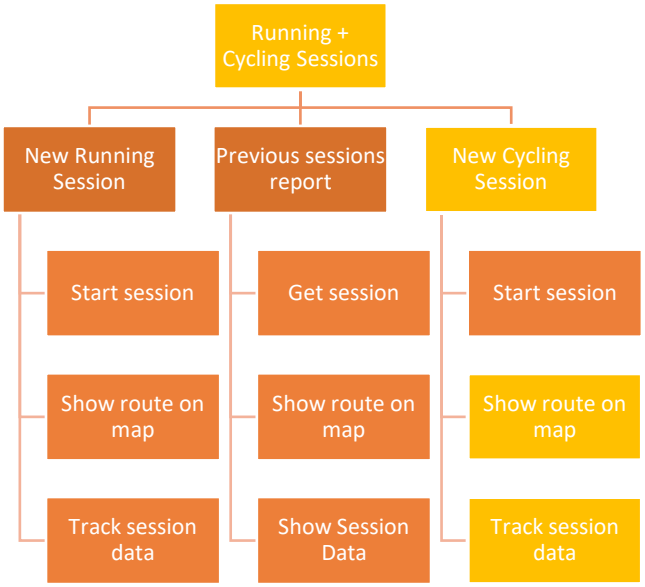
```
@Override
protected void updateCustomData(DataReadResponse dataReadResponse) {
    // make sure distance is not 0 so we don't have 0 in a division
    if (totalDistance > 0) {
        // get the time since the session has started
        long time = newUpdateTime.getTime() - session.getStartTime(TimeUnit.MILLISECONDS);
        // calculate the speed by dividing total distance by the total time
        float speed = totalDistance / (time / 1000);
        // calculate the pace by dividing 16.66 by the speed(m/s)
        pace = 16.66666666666f / speed;
        updateUIInterface.sendSpeed(); // update UI
    }

    Log.i(TAG,  msg: "Pace response " +  pace);     // log the total calories
}
```

The updateCustomData will handle the response from the API and handle any extra data for the running session. I have pace which will need to be handled so I use the exact same code as before which is not placed in MySession but use it in this method to get the pace. This methods implementation is described above.

```
public String getSpeed() {
    // make sure pace is less than an hour
    if (pace < 60) {
        // convert the pace to milliseconds
        long timeInMilliSeconds = (long) Math.floor(pace * 60 * 1000);
        // create a date with the time in ms
        Date date = new Date(timeInMilliSeconds);
        // return pace in correct format mm:ss
        return sdf.format(date);
    } else {
        // if more than an hour then return pace as 0
        return "00:00";
    }
}
```

Finally, the last method I must implement is the getSpeed method which returns the pace to update on the screen using the correct formatting. **Again, this was implemented in the previous sections and explained, so I won't explain it here again.**

CyclingSession



Because most of the work has been done by using polymorphism, I can extend the MySession class I implemented above to handle most of the data except the speed.

Using polymorphism and inheritance allows me to handle tracking the route and data. The route will be tracked just like the RunningSession so will not have to be implemented in again. For the data most of it will be the same except the speed will be calculated separately as it's not the pace like running sessions.

Justification

```
public class CyclingSession extends MySession {
```

The CyclingSession extends the MySession class to track distance, calories and location.

```
private float speed = 0;            // hold the pace
```

I will be handling the speed in the cycling class.

```
public CyclingSession(Context activity, SessionFragment fragment) {
    super(activity, fragment,  name: "Cycling");
}
```

In the constructor I call the parent class and pass it the String Cycling to show the type of session.

```
@Override
public void updateData() {
    // get the current time
    newUpdateTime = Calendar.getInstance().getTime();

    // make a read request to get get the distance between the two time intervals
    DataReadRequest readRequest = new DataReadRequest.Builder()
            .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
            .read(DataType.TYPE_DISTANCE_DELTA)
            .read(DataType.TYPE_CALORIES_EXPENDED)
            .build();

    // update the last update to current time
    lastUpdateTime = newUpdateTime;


    getBaseData(readRequest);
}
```

The updateData method is the same for now for the cycling session however this type of polymorphism allows us to add new data and handle it easier in the future.

```
@Override
protected void updateCustomData(DataReadResponse dataReadResponse) {
    // make sure distance is not 0 so we don't have 0 in a division
    if (totalDistance > 0) {
        // get the time since the session has started
        long time = newUpdateTime.getTime() - session.getStartTime(TimeUnit.MILLISECONDS);
        // calculate the speed by dividing total distance by the total time
        speed = totalDistance / (time / 1000);
        updateUIInterface.sendSpeed(); // update UI
    }

    Log.i(TAG,  msg: "Speed response " +  speed);     // log the speed
}
```

The updateCustomData method handles calculating the speed by first making sure the distance isn't 0. If I have a distance value then I use it with the active time of the session to calculate the speed. Finially I update the UI with the speed.

```
public String getSpeed() {
    return String.valueOf(Math.round(speed)); // return speed whole number
}
```

The getSpeed method method simply rounds the speed to the nearest whole number and returns that value as a string.

SessionFragment Update

I need to update the way the sessions are handled in the SessionFragment class so the relevant information can be shown for each type of session.

I added a Spinner which simply is a drop down menu. I assigned two values to the spinner when it is created which are Running and Session by creating an array of the two in Strings.xml file.

I also renamed the pace to speed so I can update the String shown to either Speed or Pace depending on the type of the session.

```
private Spinner sessionType;     // used to get type of session
```

```
private TextView speedLabel;              // update label to speed or pace
```

I create a variable for this UI element, so I know what type of session the user started. The speedLabel will be used to change the label to either km/h or /km for pace

```
// assign UI items from view
sessionType = (Spinner) view.findViewById(R.id.sessionType);
```

```
sessionType.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        sessionName = parent.getItemAtPosition(position).toString();

        if (sessionName.equals("Cycling")) {
            updateSessionTexts( spTxt: "0",  spLbl: "km/h");

            session = new CyclingSession(getContext(), (SessionFragment)getParentFragment());
        } else {
            updateSessionTexts( spTxt: "00:00",  spLbl: "/km");
            session = new RunningSession(getContext(), (SessionFragment) getParentFragment());
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

In the onCreateView method I set a listener on the spinner type so I can update the type of session when the user selects one from the drop down or changes it.

Simply it changes the format of the pace to 0 and km/h for the speed or changes it to 00:00 and /km for a running session. I created a method called updateSessionTexts to reduce repeated code. Lastly I assign the correct type of session to the session variable.

```
private void updateSessionTexts(String spTxt, String spLbl) {
    titleText.setText(sessionName);
    speedText.setText(spTxt);
    speedLabel.setText(spLbl);
}
```

The updateSessionTexts method updates the title, the format of the speed/pace and the unit label for it.

Error

```
session = new CyclingSession(getContext(), getParentFragment());
```

CyclingSession (Context, com.faizan.onefitness.session.SessionFragment) in CyclingSession cannot be applied to (Context, androidx.fragment.app.Fragment)

I get the following error because the type for the Fragment is not the one expected by the method.

Fix

```
session = new CyclingSession(getContext(), (SessionFragment) getParentFragment());
```

Crash

I cast the fragment now to the one expected

```
java.lang.NullPointerException: Attempt to invoke interface method 'void com.faizan.onefitness.session.SessionFragmentInterface.sendDistance()' on a null object reference
```

p fr = null

Running the app causes a crash. Looking at the debug and log I see the fragment being passed to my session is null.

Fix

```
    session = new CyclingSession(getContext(), fragment: SessionFragment.this);
} else {
    updateSessionTexts( spTxt: "00:00", spLbl: "/km");
    session = new RunningSession(getContext(), fragment: SessionFragment.this);
```

I changed the getParentFragment to SessionFragment.this to pass a direct pointer to the current fragment. This has fixed the crashing issue and the log confirms this:

p fr = {SessionFragment@12500} "SessionFragment{b677edb (5c14415e-b5a6-4379-9811-6b6c60eee08b) id=0x7f090096 session}"

Crash

```
java.lang.IndexOutOfBoundsException: Index: 0, Size: 0
    at java.util.ArrayList.get(ArrayList.java:437)
    at com.faizan.onefitness.sessionSummary.SessionSummaryFragment.showTrack(SessionSummaryFragment.java:114)
    at com.faizan.onefitness.sessionSummary.SessionSummaryFragment.onMapReady(SessionSummaryFragment.java:108)
```

Another issue now occurred with my app: Starting a session and stopping it straight away caused the summary fragment to crash. I realised this was because the tracked location was empty. This happened because the first location tracked was 10 seconds into the session.

Fix

```
track.endCap(new RoundCap());
trackLocation();
```

To fix this issue I tracked the location of the user at the start of the session when it began by calling the trackLocation method in the startSession method.

```
java.lang.ClassCastException: com.faizan.onefitness.session.CyclingSession cannot be cast to com.faizan.onefitness.session.R
    at com.faizan.onefitness.sessionSummary.SessionSummaryFragment.onCreate(SessionSummaryFragment.java:60)
```

Crash

I tested starting a cycling session, but the summary fragment crashed while it worked for a running session. I looked through the log file and realised I was casting the cyclingSession to runningSession.

```
session = (RunningSession) getArguments().getSerializable(ARG_SESSION);
```

```
session = (MySession) getArguments().getSerializable(ARG_SESSION);
```

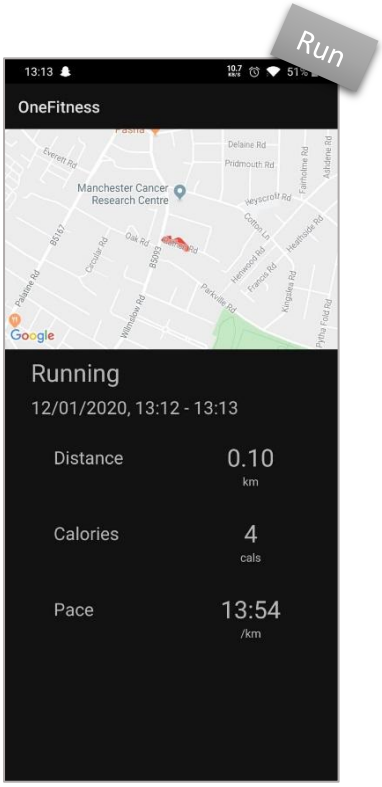
Fix

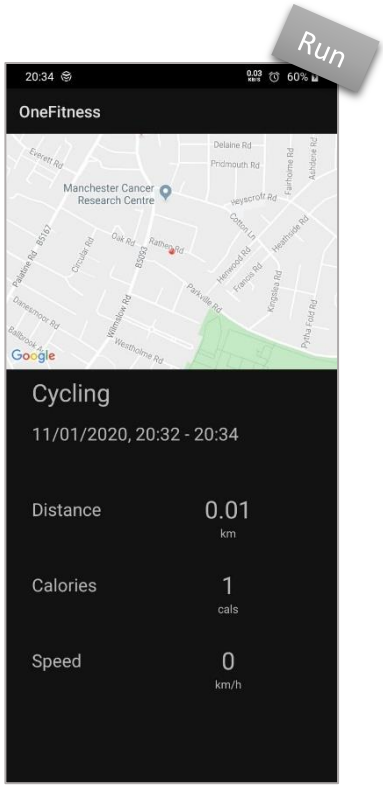
I fixed this by casting the argument type to a MySession type which is a parent of both sessions.

```
private TextView constSpeed;
private TextView speedLabel;
```

```
if (session.getName().equals("Cycling")) {
    constSpeed.setText("Speed");
    speedLabel.setText("km/h");
}
```

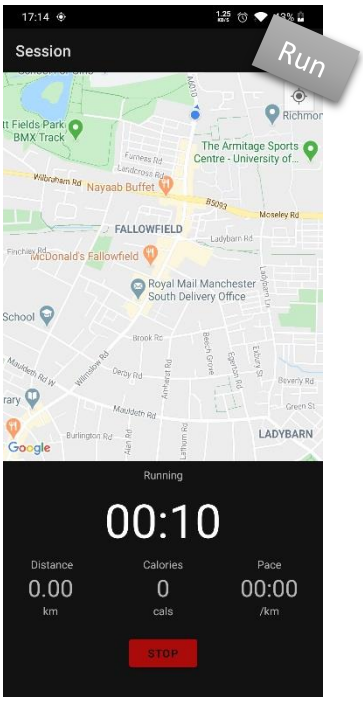
I had to change the summary fragment to show either Speed or Pace and the relevant unit. I initiated it with for a running session and check in the onCreate() method if the session type is cycling and change it accordingly.

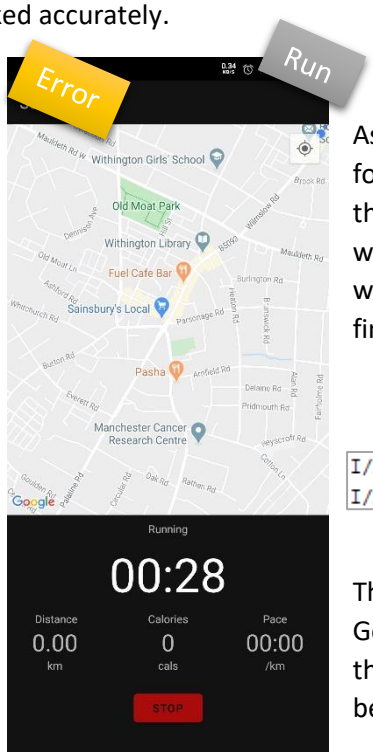




Running the app now has no crashes so these issues were fixed. I tested out the two types of sessions which are running and cycling.

However, there was two issues. Firstly, when the user travelled outside of the map they were no longer seen, and the distance was not being tracked accurately.





As you can see in both screenshots the map is not following where I travelled and therefore if I got out of the bounds, the app will continue to work but the user won't know where they are easily. The path tracked will also end up behind outside of the view. I need to first fix these two map issues.

Error

```
I/MyFit-MySession: Current Distance response 0.0
I/MyFit-MySession: Total Distance response 0.0
```

The next issue was that the distance call from the Google Fit was returning empty at times and therefore the speed and pace were much lower than they should be. I am going to fix the second issue as well.

Map View Fix – SessionFragment.java

I first removed requestLocationUpdate as it was no longer required because I was going to be using a different method to get the location in a fast interval.

```
private FusedLocationProviderClient locProvider;
private LocationRequest locRequest;
private LocationCallback locCallback;
```

I first created 3 new variables that will handle the request to get the location in a fast interval.

```
locProvider = LocationServices.getFusedLocationProviderClient(getActivity()); // get location provider
```

Next I initialise the locProvider which we will use to retrieve the location from in the onCreate class.

`requestLocationUpdates();` - in onCreateView()

Algorithm

```
private void requestLocationUpdates() {
    locRequest = LocationRequest.create();
    // create request for high priority
    locRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    locRequest.setInterval(5 * 1000); // get location atleast every 20 seconds

    locCallback = new LocationCallback() { // called when location is retrieved
        @Override
        public void onLocationResult(LocationResult locationResult) {
            super.onLocationResult(locationResult);

            if (locationResult == null) { // if there is no location then do nothing
                return;
            }
            for (Location location : locationResult.getLocations()) {
                // make sure location is retrieved
                if (location != null) {
                    // get coordinates of the current location
                    LatLng coordinates = new LatLng(location.getLatitude(), location.getLongitude());

                    moveCamera(coordinates); // move camera to centre the user

                    Log.i(TAG,  msg: "Updated Location " + coordinates.toString());
                }
            }
        }
    };

    locProvider.requestLocationUpdates(locRequest, locCallback, Looper.myLooper());
}
```

I removed the requestLocation method earlier and now replaced it with a new method called requestLocationUpdates. This method first initialises the request private variable. Its set's it to a high accuracy and priority so the location is retrieved fast with high accuracy. Next I set the interval, so it is retrieved at least every 5 seconds. I next initialise the callback variable which will be called whenever the location is retrieved. In this method I have to carry out a lot of validation to make sure the data retrieved from the device is valid before using it as it would cause a crash. First I make sure the request is not empty otherwise I end the call. If the result has data, then I get each location stored in it and go through them. A request could have multiple locations as the user moves when it's retrieved. Next I update the camera just like before with those coordinates by creating a LatLng value.

Error

```
SessionFragment: Updated Location lat/lng: (53.4278108,-2.2268453)
SessionFragment: Updated Location lat/lng: (53.4278108,-2.2268453)
```

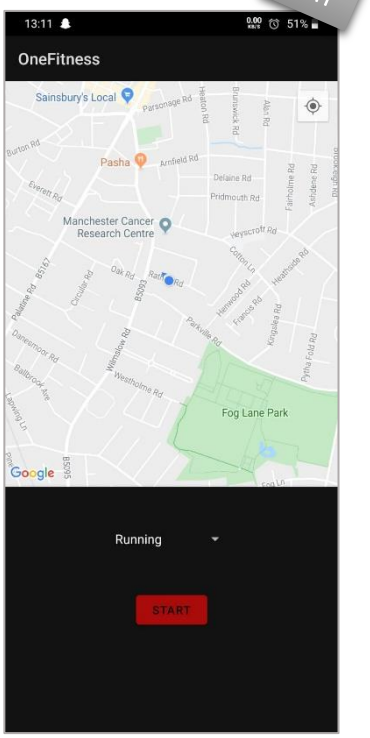
Running my app now updates the map as the user travels however there looking at the log it seems the location is retrieved twice together which is useless and just adds load to the device. I realised this was because I forgot to unsubscribe to the location when the fragment was ended. This would mean the location was retrieved in the background even when the session fragment wasn't being viewed.

Fix

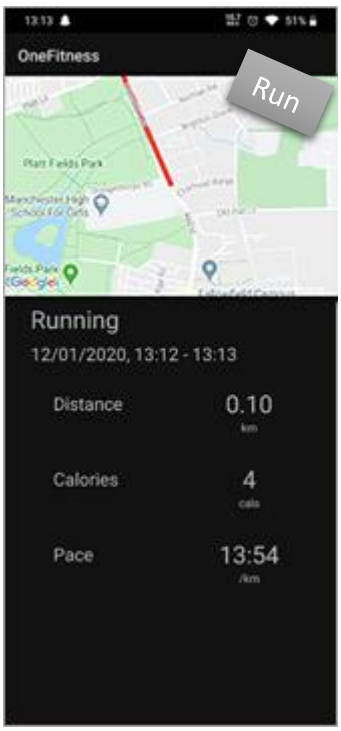
```
@Override
public void onResume() {
    super.onResume();
    if (locProvider != null) {
        // set current location on the map and zoom in - update every 2 seconds
        requestLocationUpdates();
    }
}

@Override
public void onPause() {
    super.onPause();
    if (locProvider != null) {
        // stop tracking location when fragment is not being shown
        locProvider.removeLocationUpdates(locCallback);
    }
}
```

To fix this I made sure to call the method in the onResume method instead so if the app is paused by the user exiting then it will request the location again when started. In the onPause is where the fix is. This method is executed whenever the user does not have the fragment on the screen, like being on the main screen of the app. This calls the removeLocationUpdates with the callback variable I made to stop getting the location when not required. This now fixed my issue. Now when I travelled, the map updated with me every 5 seconds and the location stayed centre of the map.

Run



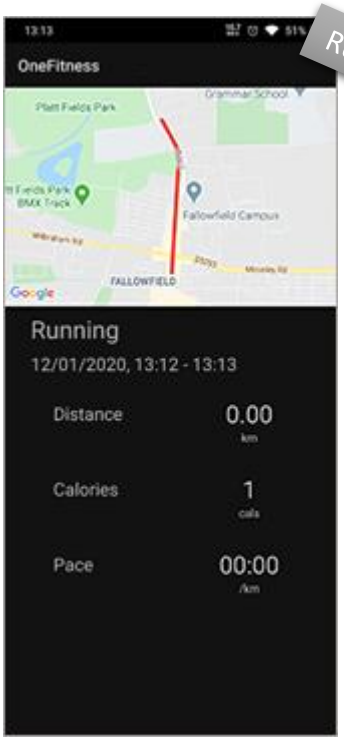
Route Track Fix – SessionSummaryFragment.java



In the screenshot on the left you can see the track has been tracked and drawn correctly however the map is cantered on the start point of the map and therefore the end is cut out. I need to fix this by moving the camera in the summary fragment to show the whole route no matter how large it is. This means we need to adjust the zoom and centre depending on the route dynamically instead of using fixed zoom levels.

```
// used to centre map on coordinates
LatLngBounds.Builder builder = new LatLngBounds.Builder();

// add all track coordinates to builder
for (int i = 0; i < track.getPoints().size(); i++) {
    builder.include(track.getPoints().get(i));
}

// get the devices density
float scale = getContext().getResources().getDisplayMetrics().density;
int padding = (int) (40 * scale + 0.5f); // used to add padding depending on screen size
// centre camera to include the whole of track with padding
CameraUpdate cu = CameraUpdateFactory.newLatLngBounds(builder.build(), padding);
// animate the camera to move to the right location
map.animateCamera(cu);
```

The above code first gets the max bounds of the track by adding all the coordinates. This will be handling getting max values at each end. Next we create padding to the camera depending on the screen's size so the track's surroundings can be viewed as well. We then move the camera to the location encompassing the track with padding.



The track is now enclosed in the view and showed correctly without being outside of the map and the map's view is zoomed in or out depending on the track now.

I have now fixed all the issues with the map's view and it works correctly. As you can see on the right the distance wasn't tracked which happens at times which I need to fix next.

Distance Fix – MySession.java

As I talked about earlier the distance sometimes does not get tracked for unknown reasons. I believe this is more of an issue with how I am using the Fit API to request the distance constantly. This does not work well with me so I am going to calculate the distance myself using the GPS coordinates of the user when I get them. This way we get not only a much more accurate reading, but it will never fail to be retrieved.

```
if (elapsedSeconds % 2 == 0) { // every 2 seconds
    // update user's location and data
    session.updateData();
}
```

Before I started, I made the update cycle every 2 seconds instead of 10 so the screen is updated more often with the track being more accurate as well as when the distance, speed and time shown to the user when the session is active. This does not affect the values when the session ends as that considers the totals.

```
//totalDistance += distance; // update the total distance
```

```
// make a read request to get get the distance between the two time intervals
DataReadRequest readRequest = new DataReadRequest.Builder()
        .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
        //.read(DataType.TYPE_DISTANCE_DELTA)
        .read(DataType.TYPE_CALORIES_EXPENDED)
        .build();
```

I first removed anything to do with updating and handling the distance in all three session classes that I had previously created with some examples showing the commented-out lines above.

Justification

With some research I figured out that the best way to calculate the distance between two GPS locations is to use the Location.distanceBetween() method. We could do it manually, but our calculations might have errors or not be as accurate. This method requires the coordinates of two locations and a float array to input the results into. We will only be interested in the first value in the float as we are only interested in the distance and not the degree to travel that route.

```
private Location lastLocation; // holds user's previous location
```

```
lastLocation = null;
```

I created a variable to hold the last location so I can compare the new coordinates to the old coordinates to work out the distance between the two. I set this to null in the startSession() method. This way it will be null whenever a new session is started so we don't calculate anything with previously saved values.

Algorithm

```
private void updateDistance(final Location newLocation) {
    float[] result = new float[1]; // used to hold the distance result
    // make sure the new location and old location are valid
    if (newLocation != null && lastLocation != null) {
        // get the distance between the two and store is result
        Location.distanceBetween(newLocation.getLatitude(), newLocation.getLongitude(),
                                 lastLocation.getLatitude(), lastLocation.getLongitude(), result);
    }

    totalDistance += result[0]; // update total distance by adding the current distance
    updateUIInterface.sendDistance(); // update UI with total distance

    Log.i(TAG,  msg: "new distance " + result[0]); // log for dubug
    Log.i(TAG,  msg: "Total Distance " +  totalDistance); // log the total distance
}
```

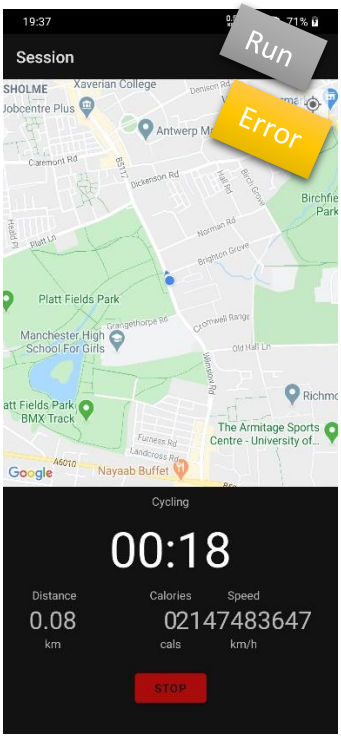
The updateDistance method takes in a Location variable which will hold the user's most up to date location. It has a result float array which will be used to store the distance. Next I make sure none of the locations are null. The new location might be null if there was an error retrieving the location and the lastLocation will be null if the session has just begun. Once we make sure both have valid values then we get the distance between the two locations and store it in the result array by passing it to the Location.distanceBetween() method which will do the calculations for us.

Next we update the total distance by adding the distance between the two points. We then update the user interface for the user to see the new distance.

Lastly, we just log to make sure everything is correct when running.

```
// update route and set new lastLocation
updateDistance(location);
lastLocation = location; // update last location
```

I call this method method from trackLocation so it's run whenever a new location is retrieved. Once it's completed I set the lastLocation to this new updated location.



Now the distance is being calculated correctly but I realised the speed is incorrect.

Algorithm

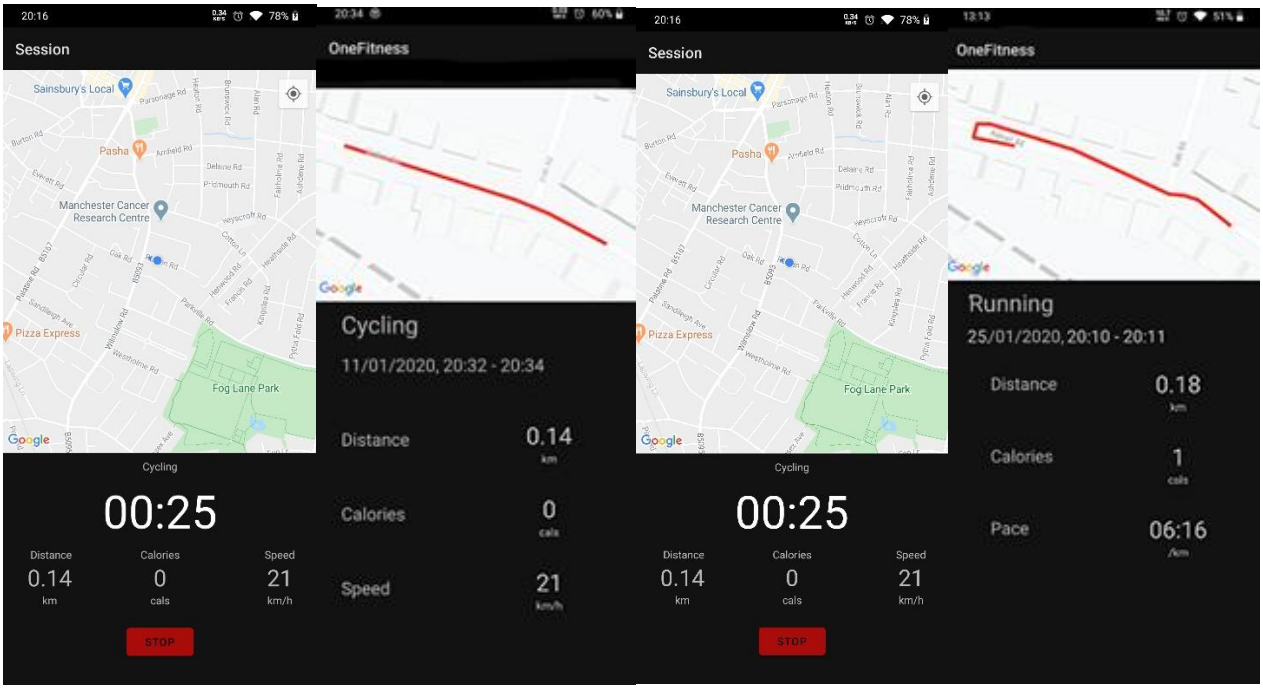
Fix

```
// calculate the speed by dividing total distance in km by the total time in hours
float timeInHours = ((float) time) / 1000f / 3600f;
float distanceInKm = totalDistance / 1000f;
speed = distanceInKm / timeInHours;
```

This was simply because my calculation was wrong which I corrected as seen above. I first turn the time from milliseconds to hours by dividing by 1000 and then by 3600. I store this in a float so it's an accurate value. I convert the distance from meters to km as well. Then I calculate the speed km/h by dividing the distance by time. This fixed the issue as I test everything out next.

Full Version Testing

Runs



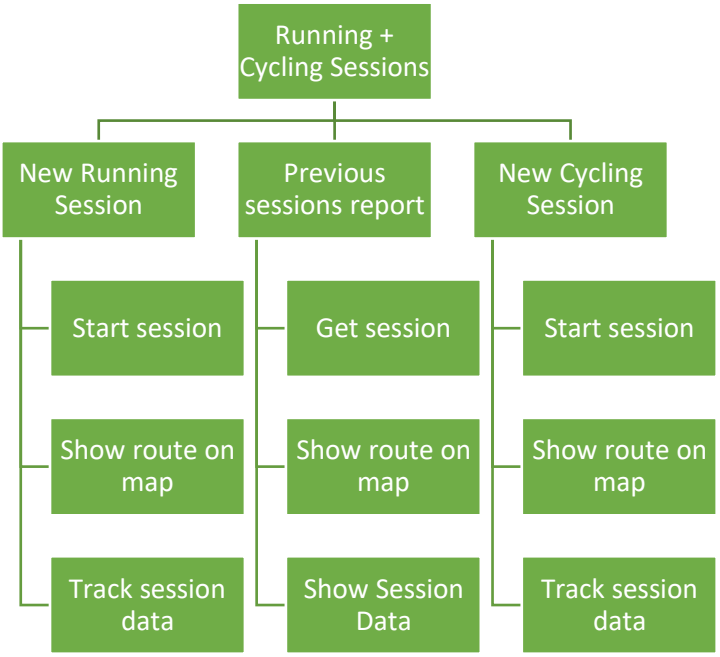
Successful!

I now tested out every feature I implemented in this version to make sure everything was working. I checked the distance and speeds against Google Maps to make sure they were accurate and they were 100% the same.

For both running and cycling the correct values were shown as they were tracked. The map shows the user's positions which is now updated as they move. The track is also shown with the correct zoom level now. The speed and distance now work completely 100% of the time instead of being a hit and miss with the API calls.

The time is shown in accurately and in the correct format as well. Everything is working and running successfully!The user's location was tracked as they moved when the session had startedI then took a walk around the street and tracked my running session. As you can see the correct route was tracked and drawn as well as all the details with it. I did the same with cycling to make sure the speed was being calculated correctly.

Conclusion



I have concluded and completed the second module of the app. In this module I used a key feature of OOP, polymorphism, to reduce repeated code as well as reduce the time it took me to implement the running and cycling sessions by allowing the two to share a parent that handled their common data. I also used new features within the UI handling of Android such as showing and hiding different UI elements on the same fragment view. This allowed me to create more pleasing UI's which were more interactive. I also continued to work with the Google Fit API but handled other data myself.

I handled the speed and pace by using my own equations alongside with the time in milliseconds. I had to work with the time in milliseconds multiple times to calculate the date, time and time difference in multiple methods.

I continued to work with Java features such as abstract methods and interfaces making my app easier to modify and understand in the future.

Start Session Summary

Test Use	Actual Outcome
Start Session View shown	The session button starts a view with a map that lets the user select the type of session they would like to start
Count Down before starting	The start button starts a count down to let the user know when the session will begin, allowing them time to put their phone away
Track distance/calories /pace/speed	The distance, calories and pace/speed are recorded and updated as the session is running
Track location as user moves	The user's location is updated on the map with their track being highlighted

End Session Summary

Test Use	Actual Outcome
Show summary of session with details	The summary is shown to the user after they have ended the session with all their details including distance, calories and pace/speed
Show outline of track on summary screen	A map is shown on the summary screen to let the user know where they have travelled

Throughout the development phase I've tested each step to make sure everything was working and fixed any issues with remedial action as highlighted by the 'Fix' tags. The above tests I ran at the end of the version to make sure this version is complete and working overall together with the tests I planned in the design.

Stakeholder's Feedback

Me:

All the requirements for version 2 are completed. Everything is working as it should. I have the session module split up into multiple modules so I can re-use the code easily in the future. I took advantage of encapsulation to allow parent and children classes to access the data they need and nothing else. I used polymorphism to keep the code easily readable and reusable.

When the user clicks the start session button, a new fragment is created and opened. This fragment allows the user to first select the type of activity they want to track. One they have either chosen running or cycling it allows the user to click the start button which begins a count down. The count down allows the user to prepare before recording data. Once the session is started it tracks the distance, calories and pace/speed. This is updated live as well as their location. A timer also shows the user how long their session has been active for. Once the user decides to end the session a summary is shown with the date time, distance, calories, pace/distance and a map view with the user's track they travelled.

I will interview and show the app to each stakeholder to review and give me any feedback they have with what they like and dislike.

Harrison:

The additions to the app provide me features I look for in a fitness app. I like the count down before a session starts. I also like that the track is shown to you when you end an activity. The app accomplishes everything I would need. It would be nice to have a steps counter for the running bit.

Abytom:

The app is nice. I like the way you have displayed the stats when you start the running. It's very simple and does everything I would need to track my running and cycling. I would like the total time for the session displayed at the end in the summary.

Jamshed:

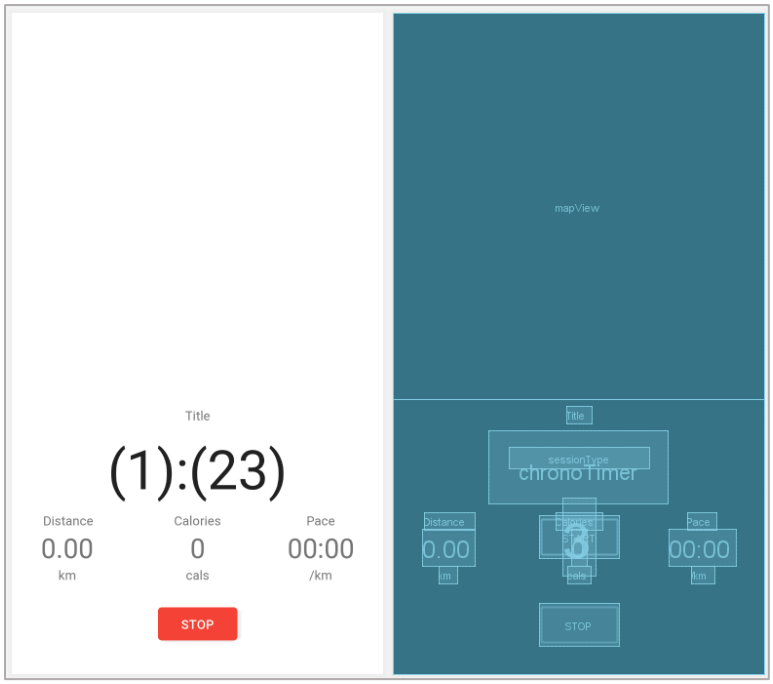
I like how basic the app is currently but would like to see some labels and symbols to quickly work out the representation of data. Some symbols for the pace and distance would especially be helpful. I like how you have handled the summary being very large texts and easy to view.

Improve:

I want to first prioritise making labels and adding icons to the app to make it more user friendly. I then want to add a step counter for the running sessions. I can implement the timer in the summary fragment that Abytom requested.

Version 2 Improvements - Stakeholders

Add Icons and Labels



I added the labels for distance, calories and pace. The pace label must be handled programmatically so it can be changed to Speed if it's a cycling session.

```
private TextView constSpeed;            // update label to speed or pace
```

```
constSpeed = (TextView) view.findViewById(R.id.constSpeedText);
```

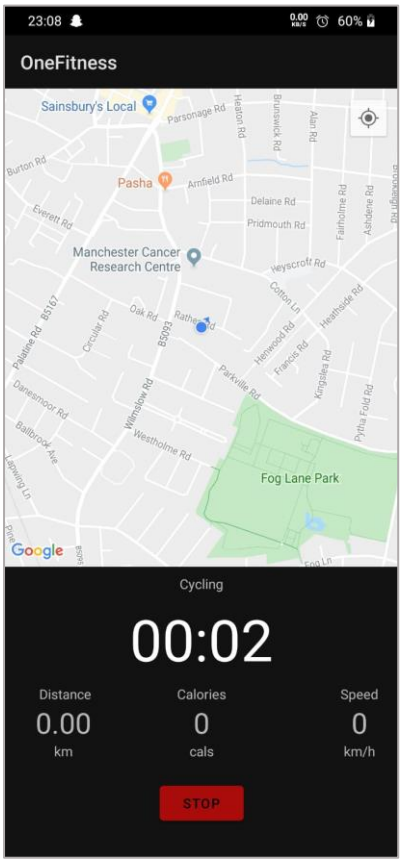
First, I get the constSpeedText from the view and store it in a private variable

```
private void updateSessionTexts(String spTxt, String spLbl, String spl) {
    titleText.setText(sessionName); // set to Running or Cycling
    speedText.setText(spTxt); // update to 0 or 00:00
    speedLabel.setText(spLbl); // update to km/h or /km
    constSpeed.setText(spl); // update label to Speed/Pace
}
```

I added a line to update the text in the updateSessionTexts method.

```
updateSessionTexts( spTxt: "0",  spLbl: "km/h",  spl: "Speed");

session = new CyclingSession(getContext(),  fragment: SessionFr
lse {
updateSessionTexts( spTxt: "00:00",  spLbl: "/km",  spl: "Pace");
```

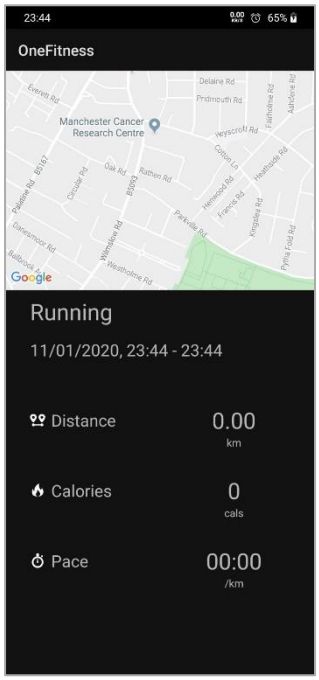
To these methods I pass an extra variable either "Speed" or "Pace" depending on the type of session chosen which updates the labels on the screen.



Running the app, I can see the labels are shown above the counters which are now easier to understand than looking at the units.

```
<drawable name="paceImage">@drawable/pace</drawable>
```

```
<drawable name="paceImage">@drawable/pace_dark</drawable>
```



I added two icons one for dark mode and one for light mode. I then added strings to refer to these images so the relevant one is used when required.

In the summary fragment I added icons for distance, calories and now speed/pace.

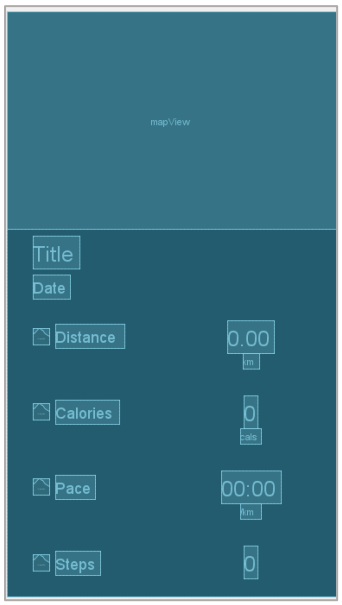
Running the app shows the correct icons when in dark mode.

This has satisfied Jamshed and now the app looks much more user friendly.

Track steps with running

Another piece of feedback from Harrison is to see his total steps after a running session.

For this I will have to change the UI and add relevant steps data.



I've setup a similar layout to the others for steps. This has an icon beside it, steps label and finally the counter for steps. I set these to INVISIBLE so they cannot be seen on start-up and are only shown when the user has a running session.

```
private int totalSteps = 0; // hold the total steps
```

In the RunningSession class I need to make a few additions. Firstly I have a private variable to hold the total steps in the session.

```
// make a read request to get get the distance between the two time intervals
DataReadRequest readRequest = new DataReadRequest.Builder()
        .setTimeRange(lastUpdateTime.getTime(), newUpdateTime.getTime(), TimeUnit.MILLISECONDS)
        .read(DataType.TYPE_DISTANCE_DELTA)
        .read(DataType.TYPE_CALORIES_EXPENDED)
        .read(DataType.TYPE_STEP_COUNT_DELTA)
        .build();
```

Next the readRequest in the updateData method is updated to include calls to the API for the steps.

```
// get data for steps
DataSet stepsData = dataReadResponse.getDataSet(DataType.TYPE_STEP_COUNT_DELTA);
// retrieve steps value from data set
long steps = stepsData.isEmpty() ? 0 : stepsData.getDataPoints().get(0).getValue(Field.FIELD_STEPS).asInt();
totalSteps += steps; // add to total
Log.i(TAG,  msg: "Steps " +  steps);     // log the steps
```

In the updateCustomData method I first read the step data from the API call and store it in a DataSet. Then I read that DataSet and retrieve steps if any exist as an Integer. Finally I add those steps to the total as this call is made approximal every 10 seconds.

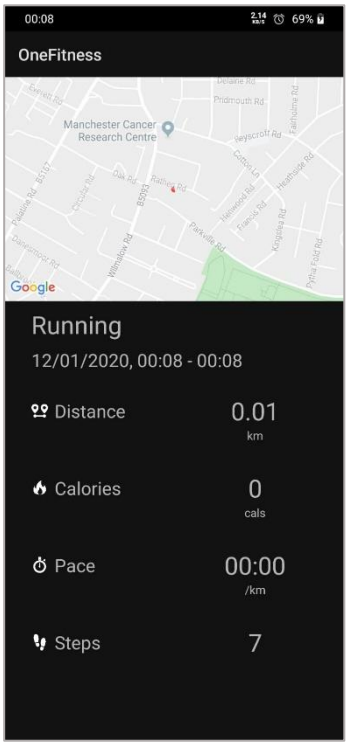
```
public int getSteps() {
    return totalSteps;
}
```

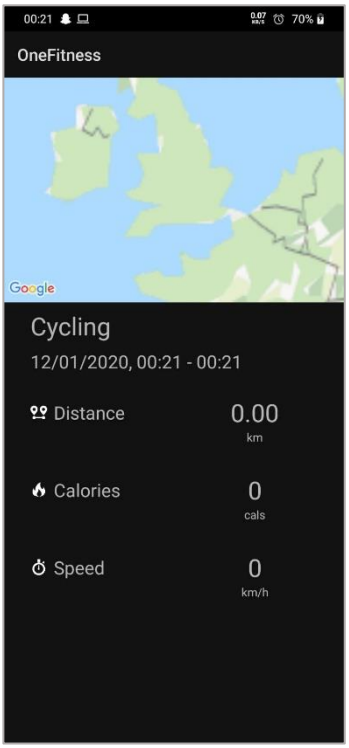
I set a simple getter method for the speed to be used later

```
private ImageView stepsImage;
private TextView constSteps;
private TextView stepsText;
```

```
if (session.getName().equals("Cycling")) {
    constSpeed.setText("Speed"); // update the UI elements
    speedLabel.setText("km/h");
} else {
    // get UI elements for steps data
    stepsImage = (ImageView) view.findViewById(R.id.stepsImageView);
    constSteps = (TextView) view.findViewById(R.id.constStepsText);
    stepsText = (TextView) view.findViewById(R.id.stepText);
    // cast to call getSteps method
    RunningSession runningSes = (RunningSession) session;
    // set UI elements visible
    stepsImage.setVisibility(View.VISIBLE);
    constSteps.setVisibility(View.VISIBLE);
    stepsText.setVisibility(View.VISIBLE);
    stepsText.setText(String.valueOf(runningSes.getSteps()));
}
```

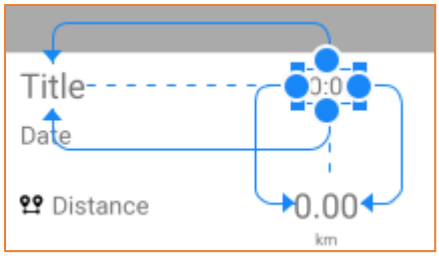
I added an else clause to the if statement that checked if the session was a cycling or running session. If it's not a cycling session then it must be a running session so I retrieve the UI elements from the fragment I made above and make them visible. I cast the session to a RunningSession so I can call the getSteps() method





When I run the app now the steps counter is only shown when a running session is run. Otherwise it is not visible and not updated. This gives Harrison an accurate step counter for his sessions now.

Next with Abytom's feedback I'm going to add the total time a session has run for in the summaryFragment.



First I added a TextView to hold the time and had this constrained to the right of Title.

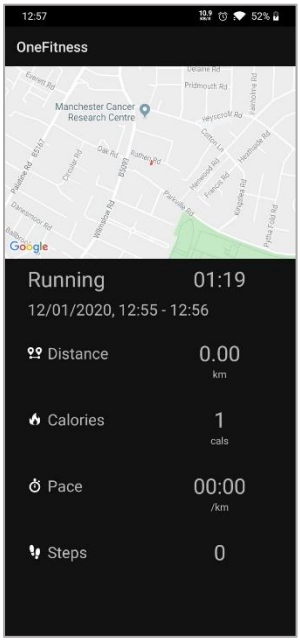
```
private TextView timeText;
```

```
timeText = (TextView) view.findViewById(R.id.timeText);
```

```
// get the time the session has been running for
long timeDiff = endTime - session.getStartTime();
// convert this time to the correct format of mm:ss
String time = String.format(Locale.UK,  format: "%02d:%02d",
                            ...args: TimeUnit.MILLISECONDS.toMinutes(timeDiff),
                            TimeUnit.MILLISECONDS.toSeconds(timeDiff) -
                            TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(timeDiff))
        );
// set the new time
timeText.setText(time);
```

I first initialise the timeText with the UI element.

Then in the end of setDate method I add the above code. First it gets the time difference between the start of the session and the end of the session. Then it uses this time difference in milliseconds and converts it to minutes and seconds. I do this my converting the time to whole minutes and then get the seconds but removing the nearest minute so it shows the minute and second in the correct format. Finally this string is set on the UI element.



Running the app now shows the time the session has been active for in the top right corner just like Abytom had requested. This app now has integrated all the features requested by the stakeholders and they are happy with it now. I can now move onto the final module.

BLACK PAGE – GO TO NEXT PAGE

Version 3

For this version I will be focusing on two main features; the journal and the profile screens.

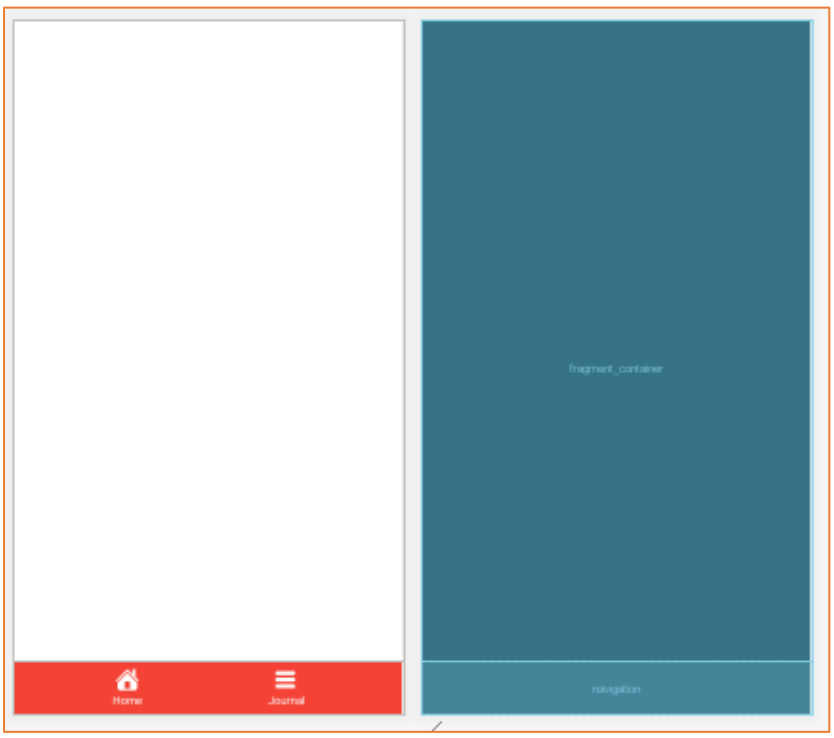
The journal will show the user their previous sessions, graphs of their performance over the week and allow them to expand sessions and view them in the SessionSummaryFragment.

The profile page will allow the user to set their weight, height, birthday and gender as well as set their daily goal for steps and calories.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/navigation_home"
        android:icon="@drawable/homeImage"
        android:title="@string/home" />

    <item
        android:id="@+id/navigation_journal"
        android:icon="@drawable/journalImage"
        android:title="@string/journal" />
</menu>
```

Muhammad Faizan Alam

First, I create a new menu named navigation.xml where I add two items. In these two items I added the Home item and the journal item. I implemented icons for these two items as well as set their text. I also set the IDs here which I will use later.



Next in activity_main.xml file I added a BottomNavigationMenu item and set it to the menu I made above. I also made attached this to the bottom of the screen. I set the colour to the primary colour of the app.

```xml
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@android:color/holo_orange_dark" android:state_checked="true"/>
    <item android:color="@android:color/white"/>
</selector>
```

I made another color xml file to change the colours when an item is selected and deselected. I set this color on the navigation menu which changes it to be more pleasing.

Next, I created an empty fragment for the journal which I will add to later for now I just want to make sure the navigation menu is implemented and working.

```
private BottomNavigationView navigation; // update UI element
```

```
// get UI element and assign it
navigation = (BottomNavigationView) findViewById(R.id.navigation);
// set the listener for when the item selected is updated to this class
navigation.setOnNavigationItemSelectedListener(this);
```

Next, I retrieve the BottomNavigationView and assign it to a private variable in the MainActivity class. I want this navigation bar to change the fragment when a different item is selected so I need to set a listener for when it changes.

```
public class MainActivity extends AppCompatActivity implements BottomNavigationView.OnNavigationItemSelectedListener {
```

Because I set the listener to the MainActivity I need to implement the listener for when the change occurs. This means I have to override the listener method.

```
private boolean loadFragment(Fragment fragment) {
    if (fragment != null) {
        // create the fragment view by replacing it onto the FrameLayout
        FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container, fragment);
        fragmentTransaction.commit();
        // fragment loaded successfully
        return true;
    }
    // fragment passed was null
    return false;
}
```

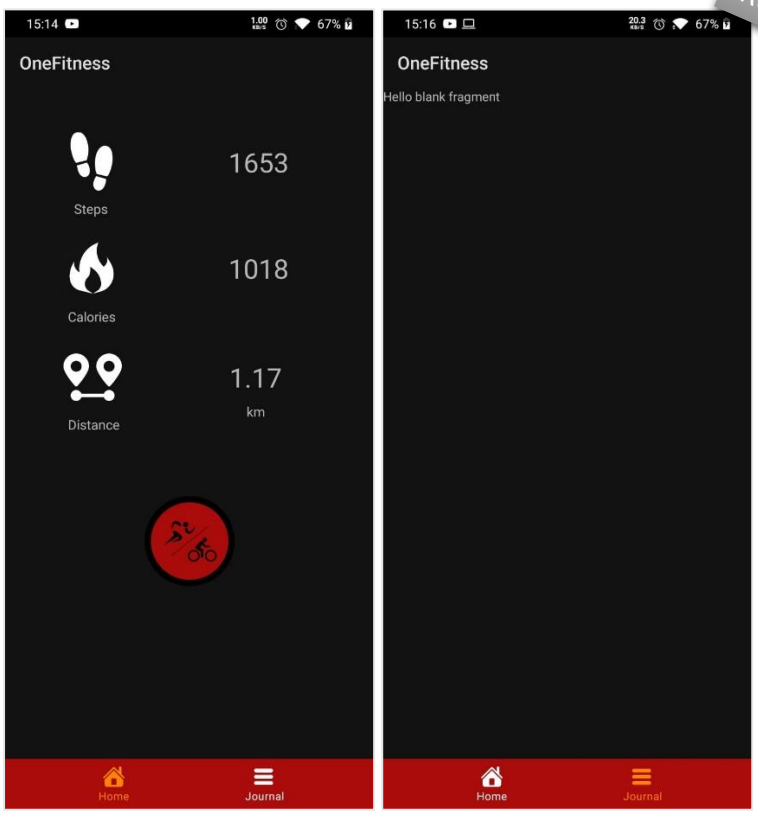
First I make a private method to handle the code of replacing the fragment in the fragment container. This method takes in a fragment and starts a Fragment Transaction to replace the view to the new fragment.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    Fragment fragment = null;

    switch (item.getItemId()) {
        // if the home was selected
        case R.id.navigation_home:
            // change to main fragment
            fragment = mainFragment;
            break;
        case R.id.navigation_journal:
            // change to new journal fragment
            fragment = new JournalFragment();
            break;
    }
    // start the new fragment
    return loadFragment(fragment);
}
```

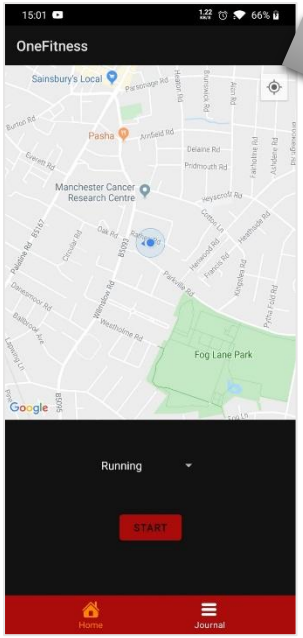
This is the method I must override. This method is called when a different item is selected in the navigation meu.

Firstly I create a null fragment. Using a switch statement I check whether the home or journal option was selected. If it was the home fragment then I assign the mainFragment which I already have initialised. If it was the journal option then I create a new journalFragment which for now is empty.

Lastly I start the fragment using the private method above by passing the fragment.



The app now has a navigation menu which works when switching between the home and journal fragments. For now, the journal fragment is empty, but I will add to it when I implement it.



Run

Error

One of the issues is that the navigation bar is persistent and does not disappear when a session is being started. I want to hide the navigation bar when I start a session because it is not required and should only be accessible from the main screen.

Fix

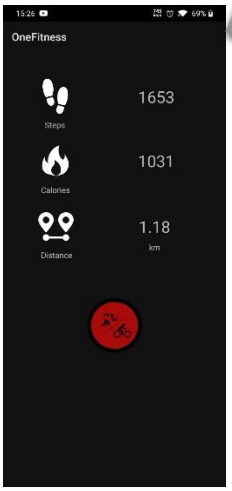
```
// start the session fragment when button is clicked
sessionButton.setOnClickListener((v) → {
        FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container, new SessionFragment(), tag: "session") // replace the current fragment
                .addToBackStack(null);  // add this fragment to the stack
        // remove the navigation bar
        ((MainActivity)getActivity()).setNavigationVisibility(false);
        fragmentTransaction.commit(); // execute the fragment call created above
});
```

When the sessionButton is pressed in the mainFragment it gets the main activity and casts it to MainActivity so I can call the setNavigationVisibility method which I pass false to, so it disappears.

```
public void setNavigationVisibility(boolean visible) {
    if (navigation.isShown() && !visible) {
        // remove the navigation bar
        navigation.setVisibility(View.GONE);
    }
    else if (!navigation.isShown() && visible){
        // show the navigation bar
        navigation.setVisibility(View.VISIBLE);
    }
}
```

I implement the setNavigationVisibility method to either remove the navigation menu or add/show it. I use a simple if statement to check it is being shown before trying to remove it and I do the opposite to show it.

Muhammad Faizan Alam

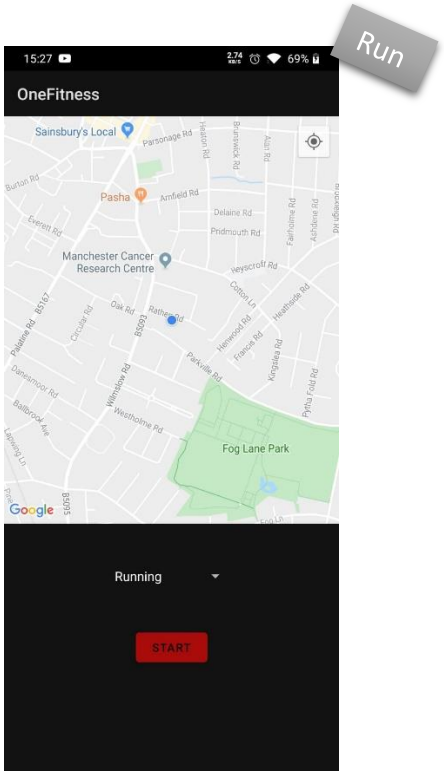


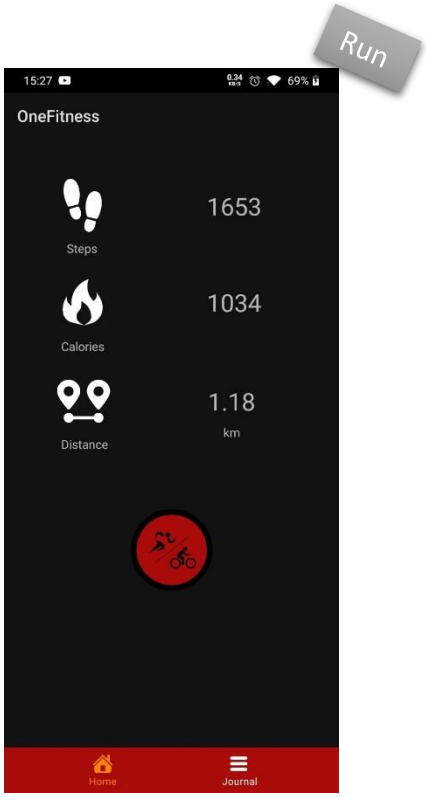
However now when I run it and start a session and return to the main menu the navigation menu is not shown at all as you can see in the screenshot. I must make it reappear once the MainFragment is shown again.

Fix

```
// show the navigation menu if not shown
((MainActivity)getActivity()).setNavigationVisibility(true);
```

To fix this I add the above line to the onCreateView for the MainFragment class. This will make sure the navigation menu is always shown when the fragment is active.





Now when I run the app it correctly displays the navigation bar on the main screen and removes it when a session is being started. When I back up to the main screen it shows the navigation menu again.

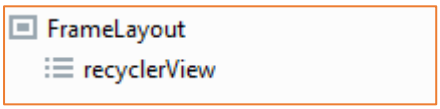
Module 1a – Journal

The journal will have a scrollable view with all the sessions the user has performed. This will allow the user to view each session and expand it so they can view a details about it.

Setup View

I want to create a scrollable view for the calendar so old sessions can be scrolled through it. I will achieve this using RecyclerView which will allow me reuse the same items so memory is not wasted. It is efficient for large lists such as our sessions.

First, I created a layout which will be the view for each item in the RecyclerView. This item view holds the image, title, time and distance for each session. This will be updated and assigned for each session the user has performed.



In the fragment_journal.xml I simply add a recyclerView.

```
public class RecyclerSessionAdapter extends RecyclerView.Adapter<RecyclerSessionAdapter.ViewHolder> {
    private static final String TAG = "MyFit-RecyclerAdapter";  // used for debugging
```

I need to make an adapter class for each item. This adapter class handled the view for the recycler class and it is a must. It extends the RecyclerView.Adapter View Holder class because it will implement the view for it.

RecyclierView

I will have to use a recyclerview for the scrolling so new items are loaded in from the top/bottom as the user scrolls with the correct info.

```
public class ViewHolder extends RecyclerView.ViewHolder {
    // hold the view elements
    ImageView image;
    TextView titleText;
    TextView timeText;
    TextView distanceText;

    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        // get the elements from the UI
        image = (ImageView) itemView.findViewById(R.id.imageView);
        titleText = (TextView) itemView.findViewById(R.id.titleText);
        timeText = (TextView) itemView.findViewById(R.id.timeText);
        distanceText = (TextView) itemView.findViewById(R.id.distanceText);
    }
}
```

Inside the Adapter class I make a ViewHolder class which simply holds the items inside the sessionitem view. This will allow us to update those items in the main adapter class.

```
private ArrayList<MySession> sessions = new ArrayList<>();
private Context context;

public RecyclerSessionAdapter(Context cntx, ArrayList<MySession> ses) {
    sessions = ses;
    context = cntx;
}
```

For now, I create a simple constructor to hold the session and context of the adapter.

```
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    // inflate the adapter with items
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.layout_sessionitem, parent,  attachToRoot: false);
    ViewHolder holder = new ViewHolder(view);

    return holder;
}
```

I need to make the view holder which inflates the view with items. I will add to this later.

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    Log.d(TAG,  msg: "onBindViewHolder: called");

    // update for each item
    holder.titleText.setText(sessions.get(position).getName());
    holder.timeText.setText(sessions.get(position).getActiveTime());
    holder.timeText.setText(sessions.get(position).getDistance());
}
```

This class is called to update the UI elements and their content for each item by using the position in each item.

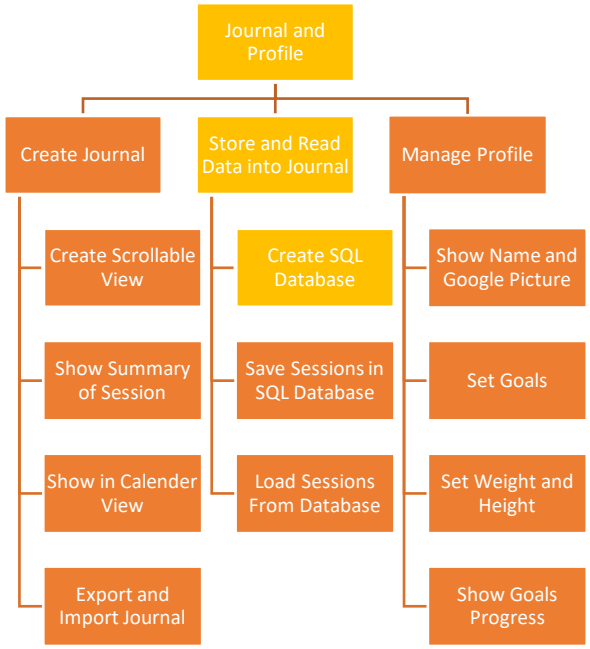
```
@Override
public int getItemCount() {
    return sessions.size();
}
```

This gets a counter for how many items there are, which will be used by the fragment later.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_journal, container,  attachToRoot: false);
    // get Recycler UI element
    RecyclerView recyclerView = view.findViewById(R.id.recyclerView);
    // Create a new adapter to hold all the items
    RecyclerSessionAdapter adapter = new RecyclerSessionAdapter(getContext(), sessions);
    // set the adapter on the UI element
    recyclerView.setAdapter(adapter);
    // Use simple linear layout
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    return view;
}
```

The journalFragment simply creates the RecyclerView and assigns a new adapter to inflate the view using all the items. For now, there are no items so the view will be empty. Our next step is to save data and load it into this view afterwards.

Module 1b – 1. Store and Read Data



I need to save all the session data when they are completed so they can be loaded into the Journal later. For this I will create an SQL database using SQLite. I will use Room which is an interface for SQLite that prevents runtime errors by verifying data at compile time. This is safer to use when working with data in Android as to prevent any issues that would otherwise be present with handling SQL directly such as reducing boilerplate code. Room also verifies calls at compile time unlike SQL which would cause errors at runtime. This allows me to catch errors before I ship the app, making my overall program more robust

Justification

```
maven { url 'https://maven.google.com' }
```

```
implementation 'android.arch.persistence.room:runtime:1.1.1'
annotationProcessor "android.arch.persistence.room:compiler:1.1.1"
```

I need to implement Rooms from Google's maven compiler so I can use them with my app. The above lines implement the latest versions.

Declare Data Type

```
@Entity(tableName = "session") // set table name
public class SessionData implements Serializable {
    @PrimaryKey(autoGenerate = true)
    private int id;
    // title for each column
    @ColumnInfo(name = "name")
    private String name;

    @ColumnInfo(name = "startTime")
    private long startTime;

    @ColumnInfo(name = "endTime")
    private long endTime;

    @ColumnInfo(name = "activeTime")
    private String activeTime;

    @ColumnInfo(name = "distance")
    private String distance;

    @ColumnInfo(name = "calories")
    private String calories;

    @ColumnInfo(name = "speed")
    private String speed;

    @ColumnInfo(name = "track")
    private LatLngs track;

    @ColumnInfo(name = "steps")
    private String steps;
```

To use Rooms I need to use tags for each class and it's data and methods. This allows maven to verify my calls to the database at compile time instead of run time which allows me to capture any issues at compile time.

Firstly I created a new data class called SessionData which will hold all the data for each session in the database. I will use an instance of this class each time I want to write or retrieve data from the database. The above varaibles hold the headings for each column with the first id being the primary key for the table.

```
public SessionData(int id, String name, long startTime, long endTime, String activeTime, String distance, String calories,
                   String speed, LatLngs track, String steps) {
    // set value from constructor as new field in table
    this.id = id;
    this.name = name;
    this.startTime = startTime;
    this.endTime = endTime;
    this.activeTime = activeTime;
    this.distance = distance;
    this.calories = calories;
    this.speed = speed;
    this.track = track;
    this.steps = steps;
}
```

I then have 4 similar styled constructors for the data type that directly initialise the private variables.

```
// getters abd setters
public long getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public long getStartTime() {
```

I then made public getters and setters for the data so they can be retrieved or modified individually.

Declare SQL Data Queries

```
@Dao
public interface SessionDataDao {
    @Query("Select * from session") // queries to table to get all items
    List<SessionData> getSessionDataList();

    @Query("Select * from session Where id = :id")
    SessionData loadSessionByID(int id);

    @Insert
    void insertSessionData(SessionData session);

    @Update
    void updateSessionData(SessionData session);

    @Delete
    void deleteSessionData(SessionData session);
}
```

I need to make a Dao class as required by Rooms. This class handles the table. It handles all the querying, inserting, deleting and updating instructions. For now I will need the selecting every item from the table.

Declare Database for Data

```
@Database(entities = SessionData.class, exportSchema = false, version = 1)
@TypeConverters({LatLngsConverter.class})
public abstract class SessionDatabase extends RoomDatabase {
    private static final String DB_NAME = "session_db";
    private static SessionDatabase instance;

    // make sure theres one one instance so database is not corrupted
    public static synchronized SessionDatabase getInstance(Context context) {
        if (instance == null) {
            instance = Room.databaseBuilder(context.getApplicationContext(), SessionDatabase.class, DB_NAME)
                                        .fallbackToDestructiveMigration()
                                        .build();
        }
        return instance;
    }

    public abstract SessionDataDao sessionDataDao();
}
```

Lastly I need the actual database for Rooms. I use the Database annotation to declare this.

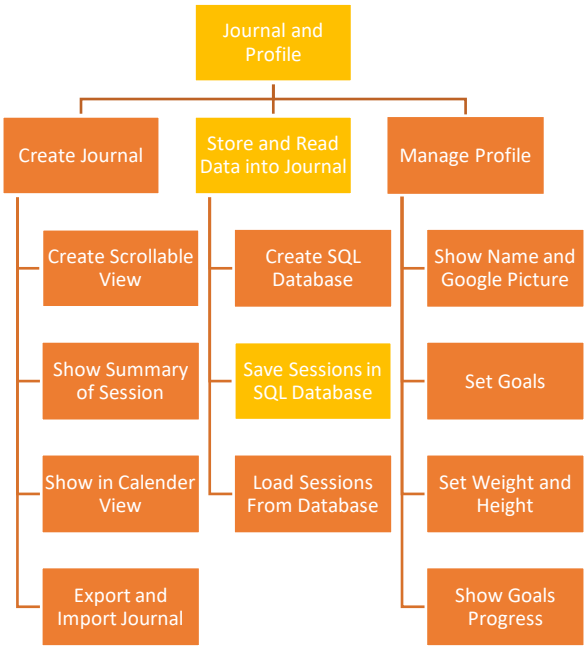
I must make sure this database is being accessed from only one place(called a singleton) at a time and is not accessed with multiple instances so I implement a synchronized method which retrieves one instance. It will create an instance if one does not exist already.

Validation

If an instance already exists then return it else create a new one, so the database is never corrupted by being accessed twice together.

I have a TypeConverters annotation which is a class I implemented to convert the user's location to string and back so it can be shown on a map after restoring the values.

2. Store Location Type in Database



Now I need to save the actual session data into the SQL database I just created. For this I will need to convert an custom type of data such as location to something primitive like String.

This data will be stored in the database so it can be retrieved later using queries.

One of the issues I run into is that I can only store primitive types in the Rooms database. This means I can not store PolygonOptions or LatLng types to store the track of the user. One solution to this is to convert the data to a string and then convert it back to the data type required.

One of the other issues I run into is that I cannot store a list so I need a class to be the interface for the list.

```
public class LatLngs {
    private List<LatLng> coordinates; // hold the user's locations

    public LatLngs(List<LatLng> coord) { coordinates = coord; } // initialise locations

    public List<LatLng> getCoordinates() { return coordinates; } // return the locations

    public void setCoordinates(List<LatLng> coordinates) {...} // set  new locations
}
```

The LatLngs class simply makes an interface to handle a List<latLng> class. This way I can convert this class directly to a string and back without having to go through two conversions.

Convert String to Location data type

```
@TypeConverter
public LatLngs toLatLngs(String coord) {
    // will hold all the coordinates
    List<LatLng> latlngs = new ArrayList<>();

    // split the string into seperate locations first
    List<String> coordinates = Arrays.asList(coord.split( regex: "\\s*:\\s*"));
    for (String crd : coordinates) {
        // split locations into latitude and longitude
        List<String> location = Arrays.asList(crd.split( regex: "\\s*,\\s*"));
        // add new location to list
        latlngs.add(new LatLng(Double.valueOf(location.get(0)), Double.valueOf(location.get(1))));
    }
    // return the locations in the LatLngs data type
    return new LatLngs(latlngs);
}
```

Firstly the toLatLngs will convert a string retrieved from the database back into the original data type which is LatLngs.

First it splits the string by colon so it gets a list of pairs of latitudes and longitudes. Then it goes through these pairs and splits the two coordinates up to retrieve the LatLng value which is added to a list. Finally the list of locations is returned.

Convert Location to String

```
@TypeConverter
public String storeCoordinates(LatLngs coordinate) {
    String value = ""; // holds the string value for the locations

    for (LatLng crd : coordinate.getCoordinates()) {
        // get the latitide
        value += String.valueOf(crd.latitude);
        // seperate lat and long by comma
        value += ",";
        value += String.valueOf(crd.longitude);
        // seperate each locatoin by colon
        value += ":";
    }
    return value; // return the whole data as string
}
```

This method converts the LatLngs into a String to store in the database. This carries out the opposite operation as the method on the page above. First it gets the latitude, adds a comma and then adds the longitude. Between each pair it adds a colon. This allows us to now store the locations of the user in the database!

One of the requirements for Rooms and SQL is to carry out the operations on a separate thread so the UI is not slowed down and the data is retrieved regardless.

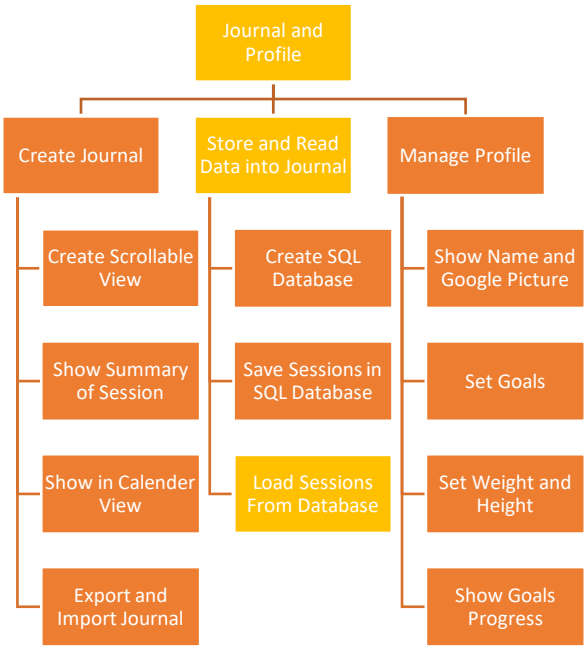
```
public class AppExecutors {

    // For Singleton instantiation so there's only one instance
    private static final Object LOCK = new Object();
    private static AppExecutors sInstance;
    private final Executor diskIO;
    private final Executor mainThread;
    private final Executor networkIO;

    private AppExecutors(Executor diskIO, Executor networkIO, Executor mainThread) {
        this.diskIO = diskIO;
        this.networkIO = networkIO;
        this.mainThread = mainThread;
    }

    // makes sure there is only one instance
    public static AppExecutors getInstance() {
        if (sInstance == null) {
            synchronized (LOCK) {
                sInstance = new AppExecutors(Executors.newSingleThreadExecutor(),
                        Executors.newFixedThreadPool( nThreads: 3),
                        new MainThreadExecutor());
            }
        }
        return sInstance;
    }

    // carry out diskIO in background
    public Executor diskIO() {
        return diskIO;
    }
    // carry out mainThread in background
    public Executor mainThread() { return mainThread; }
    // carry out networkIO in background
    public Executor networkIO() { return networkIO; }

    private static class MainThreadExecutor implements Executor {
        private Handler mainThreadHandler = new Handler(Looper.getMainLooper());

        @Override
        public void execute(@NonNull Runnable command) { mainThreadHandler.post(command); }
    }
}
```

For this I are going to use a helper class which will easily allow us to start a new thread without having to re-write the boilerplate code every time. This makes sure there is only one instance of the thread and runs it in the background using a Runnable type.

Validation

There is no validation run directly here but I will run methods in this class to prevent from any GUI locking up as it will validate when to run the method.

3. Retrieve Data – RecycleViewAdapter



Now that the SQL database has been made which stores the relevant data, I need to retrieve it using the queries I have already defined above. Once retrieved I need to handle the data in the app so the journal can populate and show them all as well as allow them to be expanded and more details be viewed.

I now need to setup the adapter to inflate each item in the recyclerView.

```
private Context context;
// hold the data for sessions
private List<SessionData> sessionData;

public RecyclerSessionAdapter(Context cntx) {
    context = cntx;
}
```

First I have a context private variable I will use in the class which is initialised in the constructor.

I cannot initialise the sessionData in the constructor as that would use the main thread so I need to do it later.

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    Log.d(TAG,  msg: "onBindViewHolder: called");

    // update for each item
    holder.titleText.setText(sessionData.get(position).getName());
    holder.timeText.setText(sessionData.get(position).getActiveTime());
    holder.distanceText.setText(sessionData.get(position).getDistance());
    // if the data type is cycling then update the image
    if (sessionData.get(position).getName().equals("Cycling")) {
        holder.image.setImageResource(R.drawable.cyclingImage);
    }
}
```

This method is called to populate the whole list for the view. It goes through each piece of data and assigns the values to a new item.

```
@Override
public int getItemCount() {
    if (sessionData == null) {
        return 0; // return 0 if no data is loaded
    }
    return sessionData.size(); // else return number of items
}
```

This method is called by the RecyclerView to optimise the app. It allows the view to make a scroll view when required for different sized screens. It will return 0 if there is no data.

```
public void setTasks(List<SessionData> sessions) {
    sessionData = sessions; // update data
    notifyDataSetChanged(); // update view
}
```

This method simply sets the data to a new list and notifies the view that the data has changed so it can redraw and update.

```
public List<SessionData> getTasks() { return sessionData; }
```

Getter method to retrieve the data in the view.

```
private RecyclerView recyclerView; // hold scrollable view
private RecyclerSessionAdapter adapter; // adapter to handle view
private SessionDatabase mDb; // database to retrieve data
```

In theJournalFragment I removed the Session type and replaced it with the database type. I also declared the recyclerView and an adapter of my custom type for it to populate it.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_journal, container,  attachToRoot: false);

    // get Recycler UI element
    recyclerView = view.findViewById(R.id.recyclerView);
    // Use simple linear layout
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    // Create a new adapter to hold all the items
    adapter = new RecyclerSessionAdapter(getContext());
    // set the adapter on the UI element
    recyclerView.setAdapter(adapter);

    mDb = SessionDatabase.getInstance(getContext());

    return view;
}
```

I then initialise the recyclerView and set the adapter for it so it can populate with our custom data types.

I also get an instance of the database. I do not have to worry about retrieving the data incorrectly as Rooms will handle it all for us and make sure the data is valid at compile time.

```
@Override
public void onResume() {
    // called when fragment is shown
    super.onResume();
    // get all the sessions from the database
    retrieveTasks();
}
```

Crash

```
java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time.
    at androidx.room.RoomDatabase.assertNotMainThread(RoomDatabase.java:209)
    at androidx.room.RoomDatabase.query(RoomDatabase.java:237)
    at com.faizan.onefitness.SessionData.SessionDataDao_Impl.getSessionDataList(SessionDataDao_Impl.java:176)
    at com.faizan.onefitness.journal.JournalFragment.onCreateView(JournalFragment.java:78)
```

I have to make sure I get the data whenever the view is shown to the user. It must be updated every time it is paused and resumed again with new data so I put the data retrieval in the onResume class which is called by Android. Adding the method here prevents the crashing caused when data is not updated after resuming.

```
private void retrieveTasks() {
    // start a new thread
    AppExecutors.getInstance().diskIO().execute(() -> {
            // update the data in the background thread
            final List<SessionData> sessions = mDb.sessionDataDao().getSessionDataList();
            // update the view in the UI thread
            getActivity().runOnUiThread(() -> {
                    adapter.setTasks(sessions);
            });
    });
}
```

Fix

This method starts a new thread in the background which retrieves the data from the database. This way our UI is not slowed down. Next the view is updated with the data I the UI thread of the activity.

Save Data

Now I have everything implemented to handle the data and retrieve it. I now must store it whenever a session is stopped by making the actual query call. I have already setup everything to save the data above.

```
// hold database and data to store
private SessionDatabase mDb;
protected SessionData sessionData;

protected abstract void updateSaveData();
```

In the MySession class I created a new updateSaveData which is abstract so it has to be implemented by any children classes

```
@Override
protected void updateSaveData() {
    sessionData = new SessionData(getName(), getStartTime(), getEndTime(), getActiveTime(), getDistance(), getCalories(), getSpeed(), new LatLngs(getTrack().getPoints()));
}
```

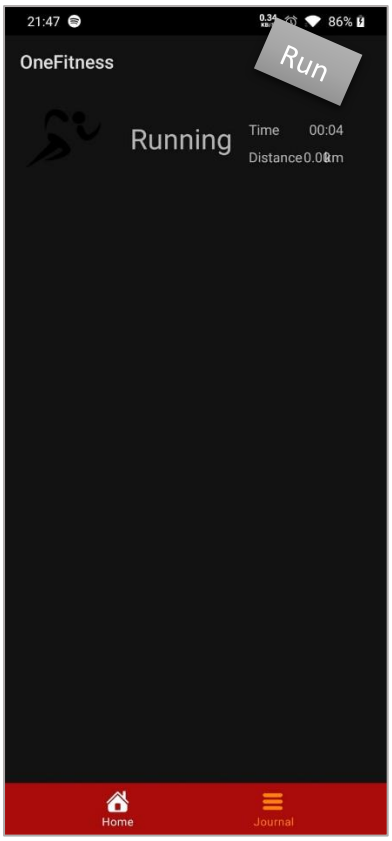
```
@Override
protected void updateSaveData() {
    sessionData = new SessionData(getName(), getStartTime(), getEndTime(), getActiveTime(), getDistance(), getCalories(), getSpeed(), new LatLngs(getTrack().getPoints()), getSteps());
}
```

These two methods are implemented nearly identically except RunningSession adds steps as well to the data that has to be stored. These two methods store the name, start time, end time, active time, distance, calories, speed, locations, and steps.

```
response.addOnSuccessListener((OnSuccessListener) (sessions) → {
        Log.i(TAG, sessions.toString());
        Session session = sessions.get(0);

        updateSaveData();

        AppExecutors.getInstance().diskIO().execute(() → {
                mDb.sessionDataDao().insertSessionData(sessionData);
        });
})
```

When a session is successfully ended it calls the classes updateSaveData method to store either the data for running or cycling. It then starts a background thread to store the data in the database using the insertSessionData method I implemented above. This prevents crashing and hanging the UI.

I now have both writing data and reading data implemented. Next thing to do is test it.

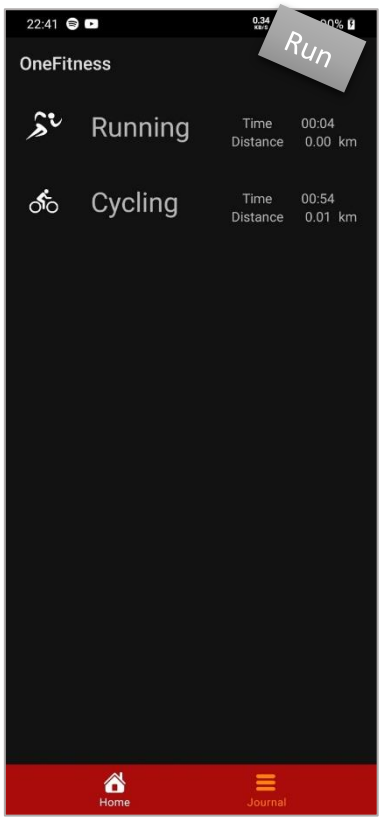


Testing

Validation Testing

Successful!

It works as expected after starting a running session and stopping it. I went back to the journal and the data shown was accurate but was not aligned and represented correctly. I had a dark theme so I need to change the icon to white and also reduce the size of the title. I also need to align the km unit using constraints.



I added white versions of the icons and set it in the night mode references. I also realigned the icon, title and unit for distance. This way the view now looks much more user friendly.

Module 2 - Show Summary of Session

Now that I have created the view as well as loaded data from the SQL database and shown it on the Journal view, I can allow the user to expand it to view more details when clicked on so they can see a map as well as more precise info about the sessions performed.

I did this out of order because I needed the SQL database before I could implement this.

Update Summary Fragment

Because I are working with Rooms and working with data directly instead of sessions I need to update the SessionSummaryFragment to instead show SessionData and not MySession types. This will allow me to reuse that view for the calendar view.

```
private SessionData sessionData; // hold current sessions data
```

```
public static SessionSummaryFragment newInstance(SessionData ses) {
```

```
sessionData = (SessionData) getArguments().getSerializable(ARG_SESSION);
```

I changed all instances of MySession to SessionData as well as the variable names.

```
stepsText.setText(sessionData.getSteps());
```

I no longer need to cast to RunningSession and can get the steps directly.

```
args.putSerializable(SessionSummaryFragment.ARG_SESSION, session.getSessionData());
```

In the SessionFragment I pass the SessionData instead of the actual session class.

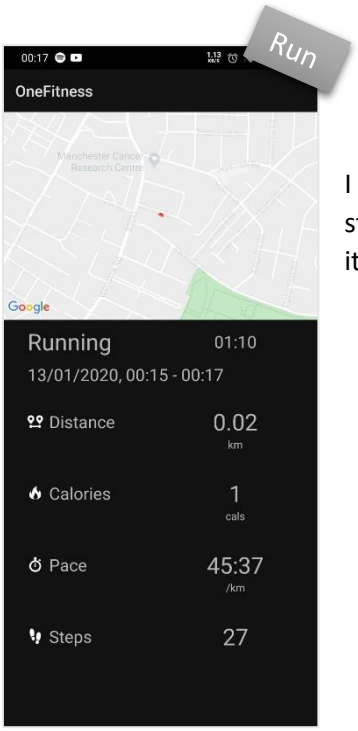
```
public class SessionData implements Serializable {
```

This also means I have to make SessionData Serializable so it can be passed in fragment intents.

```
java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.String com.faizan.onefitness.SessionData.SessionData.getName()' on a null object reference
    at com.faizan.onefitness.sessionSummary.SessionSummaryFragment.onCreateView(SessionSummaryFragment.java:102)
```

Running the app now causes a crash because I started and stopped the session before the first 10 seconds was up. So I added the updateSaveData to the end of the startSession which fixed this problem.

```
updateSaveData();
```




```
Total Distance response 25.898712
```

I was having inconsistency with this fix because the summary fragment was being started before the session was ended giving the distance to be 0.02 instead of 0.03 as it should have been by looking at the debug log.

To fix this I'm going to start the SessionSummaryFragment only after the session has ended

```
public void startSummaryFragment() {
    trackingView.setVisibility(View.INVISIBLE); // hide the tracking layout
    startButton.setVisibility(View.VISIBLE);    // show the start button
    sessionType.setVisibility(View.VISIBLE);

    Bundle args = new Bundle();
    args.putSerializable(SessionSummaryFragment.ARG_SESSION, session.getSessionData());
    //args.putString(SessionSummaryFragment.ARG_DISTANCE, session.getDistance());
    SessionSummaryFragment fragment = new SessionSummaryFragment();
    fragment.setArguments(args);
    FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
            .replace(R.id.fragment_container, fragment, tag: "summary") // replace the current fragment
            .addToBackStack(null);  // add this fragment to the stack

    fragmentTransaction.commit(); // execute the fragment call created above
}
```

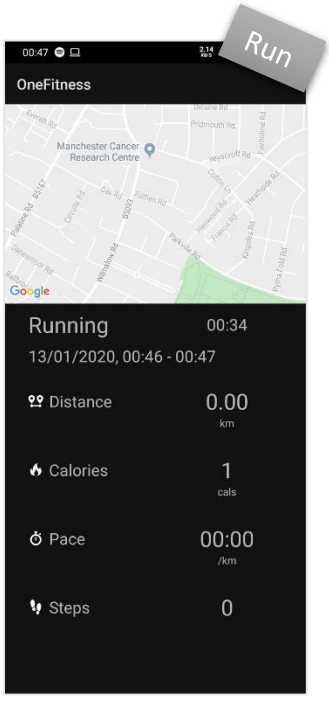
I moved the session UI updates to a new method from the stop button listener.

```
void startSummaryFragment();
```

I then declare this in the SessionFragmentInterface so I can call the method from the MySession class.

```
updateUIInterface.startSummaryFragment();
```

I then call this method from after the saved data has been updated. This means I don't have to keep calling the save data method and can just call it once at the end. This gives me good consistency with the data.



Now the data shown is correct as its shown after it has been saved!

Show Summary Of Calendar Items

Now that I have everything implemented, I can implement expanding the items when they are selected by the user in the journal. This will open a SessionSummaryFragment to show the same exact information.

```
public interface OnSessionListener {
    void onSessionClick(int position);
}
```

First I made an interface which will allow me to call a method in the JournalFragment to show the summary fragment.

```
implements RecyclerSessionAdapter.OnSessionListener{
```

I implement this interface in the Journal fragment so I it can be called by the adapter.

```
@Override
public void onSessionClick(int position) {
    Bundle args = new Bundle();
    args.putSerializable(SessionSummaryFragment.ARG_SESSION, sessions.get(position));

    SessionSummaryFragment fragment = new SessionSummaryFragment();
    fragment.setArguments(args);
    FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction()
            .replace(R.id.fragment_container, fragment,  tag: "summary") // replace the current fragment
            .addToBackStack(null);  // add this fragment to the stack

    fragmentTransaction.commit(); // execute the fragment call created above
}
```

I implement the listener method onSessionClick to start a SessionSummaryFragment which I pass the argument of the session at the position.

```
public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
```

In the ViewHolder I implement the OnClickListener so I it can be called when the view is clicked.

```
OnSessionListener onSessionListener;
```

```
this.onSessionListener = onSessionListener; // assign click listener
itemView.setOnClickListener(this); // implement click listener
```

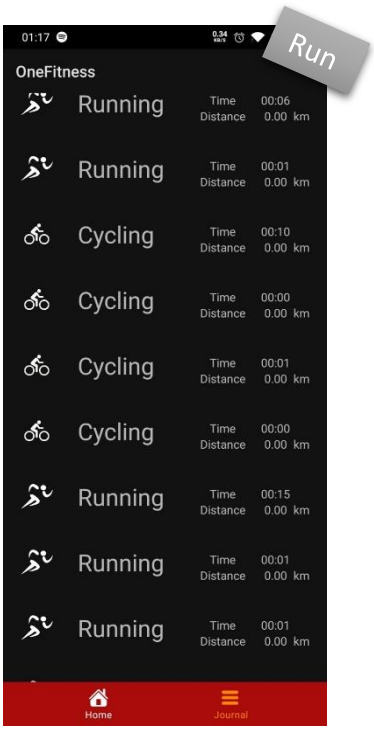
I then implement the interface and assign it in the constructor as well as make the OnClickListener active on the view for the current item.

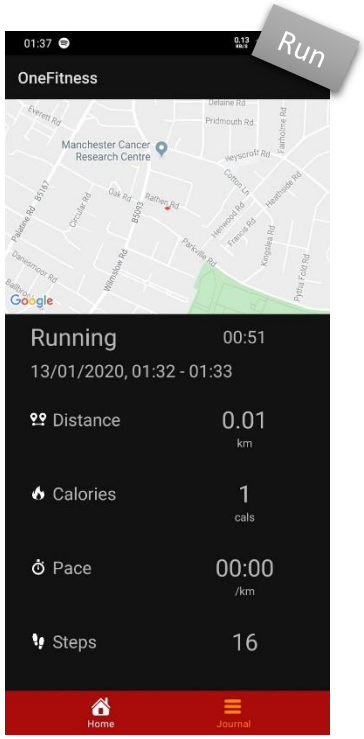
```
@Override
public void onClick(View v) {
    Log.d(TAG,  msg: "onClick: called");
    // call the interface method when item is clicked
    onSessionListener.onSessionClick(getAdapterPosition());
}
```

Finally I override the method onClick to call the interface method that will start a summary fragment on the current item. The getAdapterPosition will get the position for the item currently clicked.

```
List<SessionData> sessions;
```

Finally, I have a reference to the data in the JournalSession so it can be passed to the summary fragment.





Now when I click on an item it shows it in the summary fragment with all the relevant data!

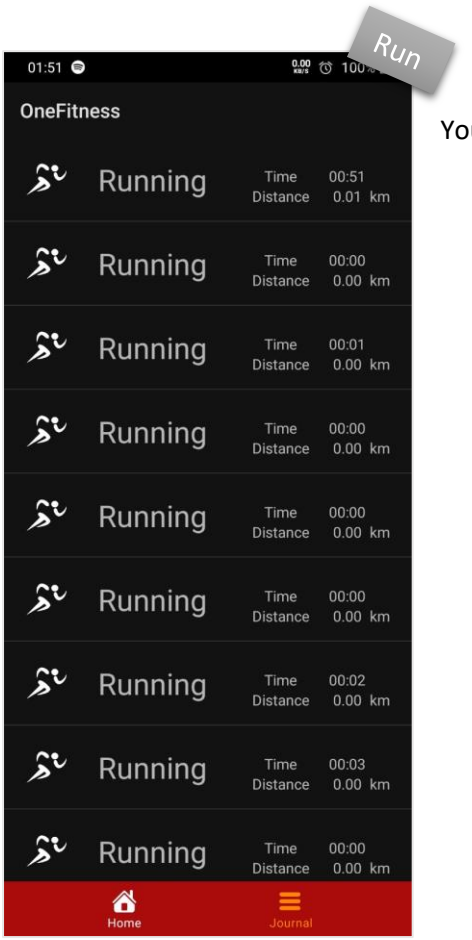
However there is one issue which is the last item cannot be seen as it's covered by the navigation bar.

```
android:paddingBottom="64dp" />
```

I added padding to the bottom of the RecyclerView which fixed the issue and now all items are viewable!

```
DividerItemDecoration dividerItemDecoration = new DividerItemDecoration(recyclerView.getContext(),
        getResources().getConfiguration().orientation);
recyclerView.addItemDecoration(dividerItemDecoration);
```

To finish the view I added horizontal dividers between every item to make it look more pleasantly appealing.

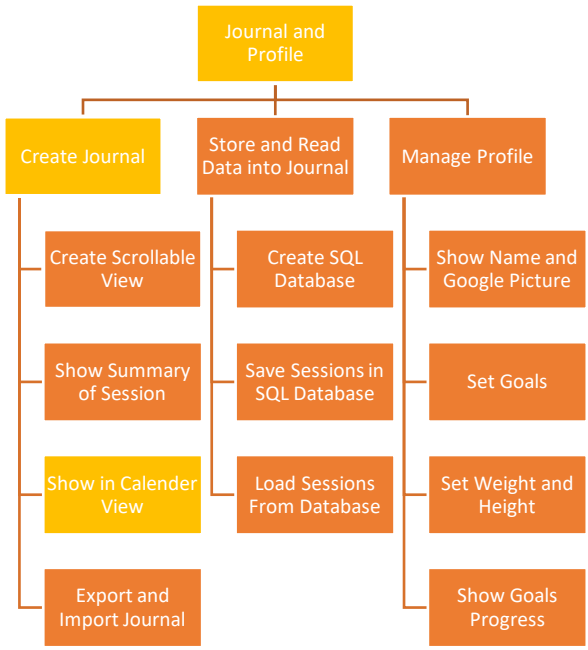


You can now see the grey lines separating each session now.

Muhammad Faizan Alam

Successful!

Show Date In View



I want to show the sessions ordered from newest to oldest in a scrollable calendar view as I designed. This is the final step for the journal.

To show the user a timeline I must sort the sessions loaded into the scrollable view by date starting with the most recent date.

To achieve this I need to retrieve the data from the SQL database by descending order using the start time so the largest number (the recent time) is loaded in first.

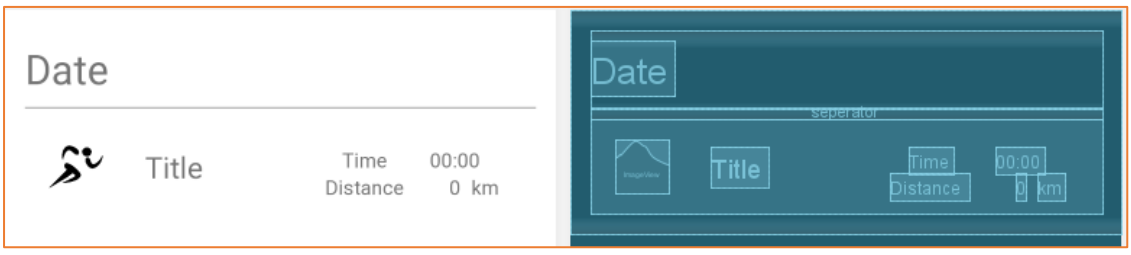
Justification

```
// get in order starting from most recent session
@Query("Select * from session ORDER BY CAST(startTime AS Long) DESC")
List<SessionData> getSessionDataListSorted();
```

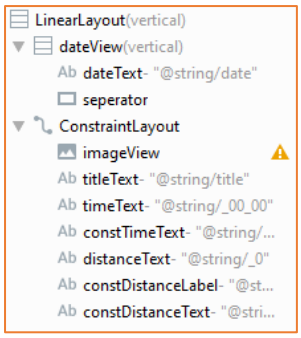
To do this using SQL I add another Query method. This method selects every item from the session table just like before except this time it gets the most recent session from the database. It does this by first making show I have the value as a long so it can be ordered by descending order using DESC.

```
sessions = mDb.sessionDataDao().getSessionDataListSorted();
```

I need to update the call to the database to now use this new call, so the newest sessions are loaded in instead of the oldest.



Next I want to handle showing the date for each session. However the date should only be shown when it's the first session for that day else it will hide the Date.



To achieve this I added the sessionitem layout into a LinearLayout so it can be resized when the dateView is not shown. Constraint layout does not support this so I put that layout inside a LinearLayout to move the items when they're not visible.

```
// variables to handle Date
LinearLayout headingView;
TextView dateText;
```

```
headingView = (LinearLayout) itemView.findViewById(R.id.dateView);
dateText = (TextView) itemView.findViewById(R.id.dateText);
```

The Date view is retrieved from the layout as well as the text for the date so they can be shown and assigned to in the holder sub-class.

```
private final SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy");
```

```
private String lastDate; // hold the last date
```

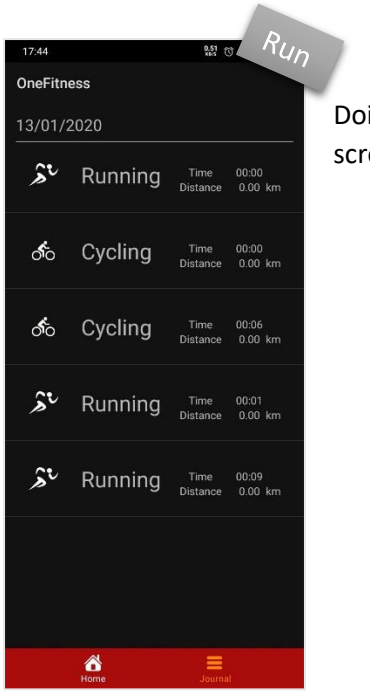
Class variables to hold data and handle conversion.

```
// get the time from the current session
long startTime = sessionData.get(position).getStartTime();
// format the time to retrieve the date
String date = dateFormat.format(startTime);
// Only show the date if it does not already exist
if (!date.equals(lastDate)) {
    lastDate = date; // update the lastDate
    // Show the header for the date
    holder.headingView.setVisibility(View.VISIBLE);
    // set the date text to the session's date
    holder.dateText.setText(date);
} else {
    // remove the date if it's already shown
    holder.headingView.setVisibility(View.GONE);
}
```

In the BindViewHolder method I get the session being populated and get it's start time. I then convert this start time to a date using the dateFormat variable which is final so it cannot be changed.

Next I compare the date retrieved from this session to the last date. This is so I only show the date for the current session if it's a new day to be shown that way the same date isn't shown with every session view item. This then sets the visible of the date view to visible so it can be shown and then set's the text of it to the date that was retrieved.

Finally, I hide the view if the date has already been shown with another session item.

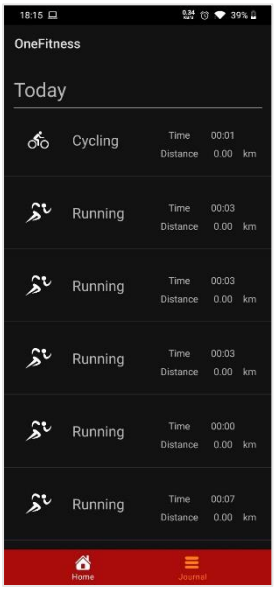


Doing this now shows the sessions underneath a heading with the date as shown in the screenshot.

```
// if date is today's date then set it to 'Today'
if (date.equals(dateFormat.format(System.currentTimeMillis()))) {
    date = "Today";
}
```

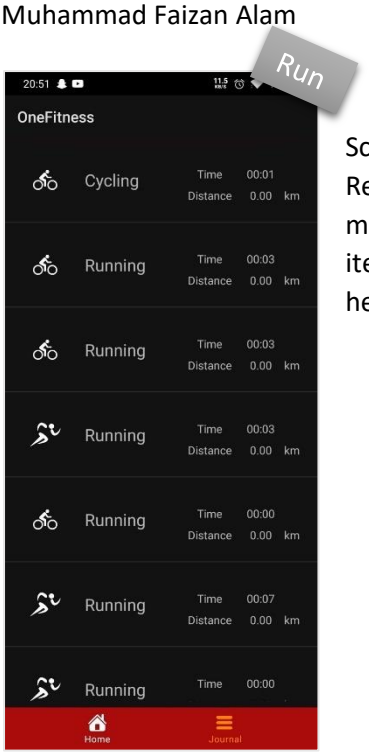
I added the following code to set the Date to “Today” when the sessions have been performed today.

I also updated the layout to make it more easily readable.



Now the app shows “Today” in larger text because it’s the current date for those sessions.

The layout is also updated reduce the size of the sessions’s titles and make the date larger as well as making distance more easily viewable by adding a margin to the top of it.



Scrolling down and then back up would make the date disappear. This is because the RecyclerView reuses the holder items instead of destroying and recreating them. This means the items hold their state from the previous instantiation which also means the item which should show the title is assigned to a new item which does not have the header item active.

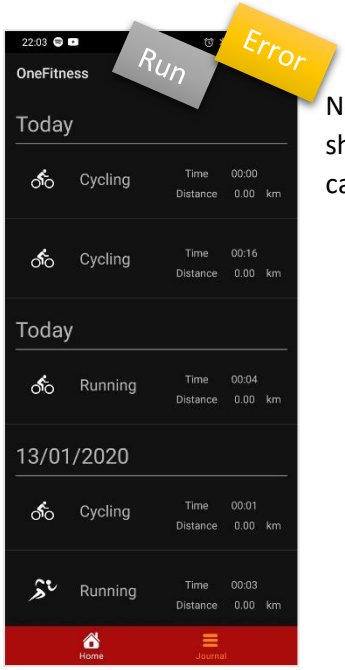
I need to keep track of what items will have the date active above them. I can do this by creating a list of every position with the title on it.

```
// hold items that should always show the date
private List<Integer> headerItems = new ArrayList<>();
```

I create a list and assign it a new empty list. This list will hold integer values which are the positions for the list of SessionData items I have for the RecyclerView.

```
// Only show the date if it does not already exist
// or item previously showed the date
if (!date.equals(lastDate) || headerItems.contains(position)) {
    lastDate = date; // update the lastDate
    headerItems.add(position); // item should show date everytime
```

I updated the if statement to now also execute when the item has previously shown the date. That way when an item that goes off the screen and is shown again, it will show the date for that item.



Now when I scroll down and back up instead of the date disappearing, it is repeatedly shown. This is because the lastDate variable switches between Today and the 13th as you can see on the screenshot.

To solve this issue previously I used only one temporary variable to only hold the last date. However this wouldn't work because what if I have three dates displayed at once. To solve this I will store all the dates that have been shown in a list.

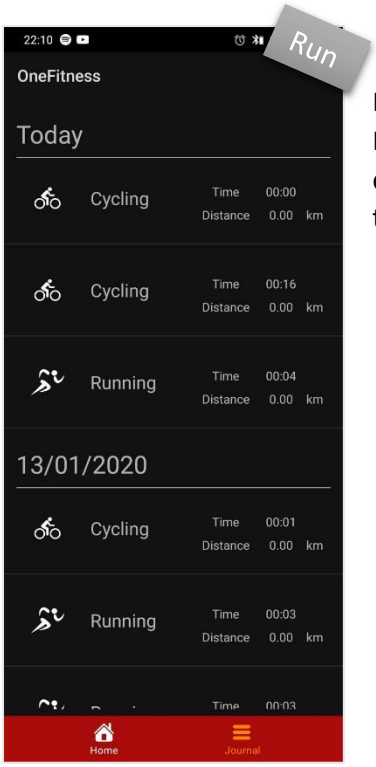
```
// hold all previous dates, will hold unique dates
private List<String> dates = new ArrayList<>();
```

I create another member list but this time it's of Strings. This is initialised as empty and I will add a date every time this list does not have it.

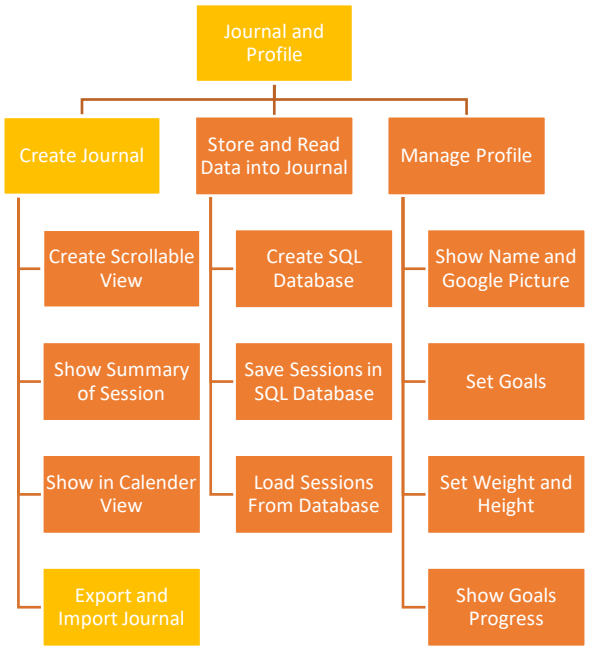
Fix

```
// Only show the date if it does not already exist
// or item previously showed the date
if ((!dates.contains(date) || headerItems.contains(position))) {
    dates.add(date); // add because it is a unique date
    headerItems.add(position); // item should show date everytime

    // Show the header for the date
    holder.headingView.setVisibility(View.VISIBLE);
    // if date is today's date then set it to 'Today'
    if (date.equals(dateFormat.format(System.currentTimeMillis()))) {
        date = "Today";
    }
    // set the date text to the session's date
    holder.dateText.setText(date);
} else {
    // remove the date if it's already shown
    holder.headingView.setVisibility(View.GONE);

}
```

I updated the if statement again so it checks if the date exists in the list or if the item should show the header like before. This means every item which has a new unique date will have the date shown above it else it won't.



Now only unique dates are shown and stay there when I scroll up and down. This way I have fixed the two issues I ran into. The Journal view is now complete. It shows the dates, with the sessions underneath it and those sessions can be clicked to expand them.



Justification

Next step is to implement the export and import methods to allow the user to create backups of their sessions. This will allow the user to transfer data between devices as well as make regular backups in case the application is deleted so it can be restored back with their data. For this I will have to do file handling as well as managing the files using bytes as that's the most efficient and best way to transfer the files. I will follow Android's recommended way to handle files to reduce the risk of corrupt files as well as crashes.

https://developer.android.com/training/data-storage

Exporting Database

Permissions – MainActivity

To be able to access the user's files I must ask for permissions from the user which will be similar to the location permissions.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

First I have to let the Android system know that my app requires external storage access by adding the above to the manifest file.

```
private static final int MY_PERMISSIONS_REQUEST_FILE_ACCESS = 2494;
```

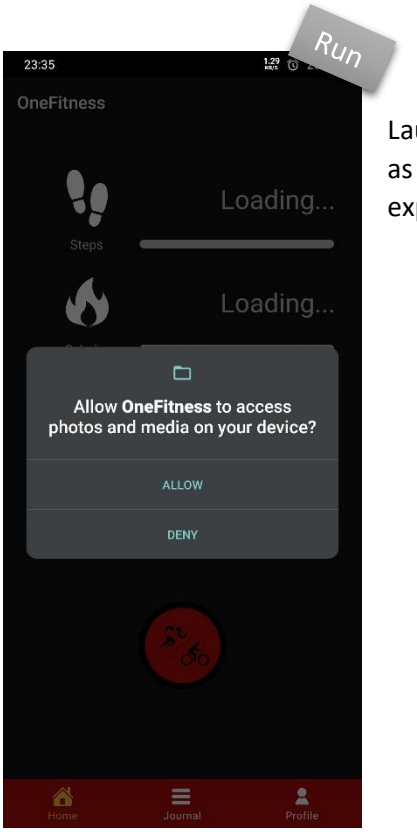
```
private void grantFilePermissions() {
    // check if we already have file permission first
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {

        // Permission is not granted
        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
                Manifest.permission.WRITE_EXTERNAL_STORAGE)) {
            manualPermissionDialog("Go to Settings to grant permissions for importing and e...");
        } else {
            // No explanation needed; request the file permission
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                    MY_PERMISSIONS_REQUEST_FILE_ACCESS);
        }
    } else {
        // make sure to get location permissions next
        grantLocationPerms();
    }

}
```

I used a static variable to hold the return code from the permission activity first. I created a new method in the MainActivity to handle getting external write permissions, this will also get permissions to read the data as writing cannot be done without reading it. This method is very similar to the location method I created in version 2. It first checks if the permissions were already granted and if so it requests the location permissions. Otherwise it'll request the permissions if they have not been requested before. If they have it manually requests them from the user by guiding them to settings just like before.

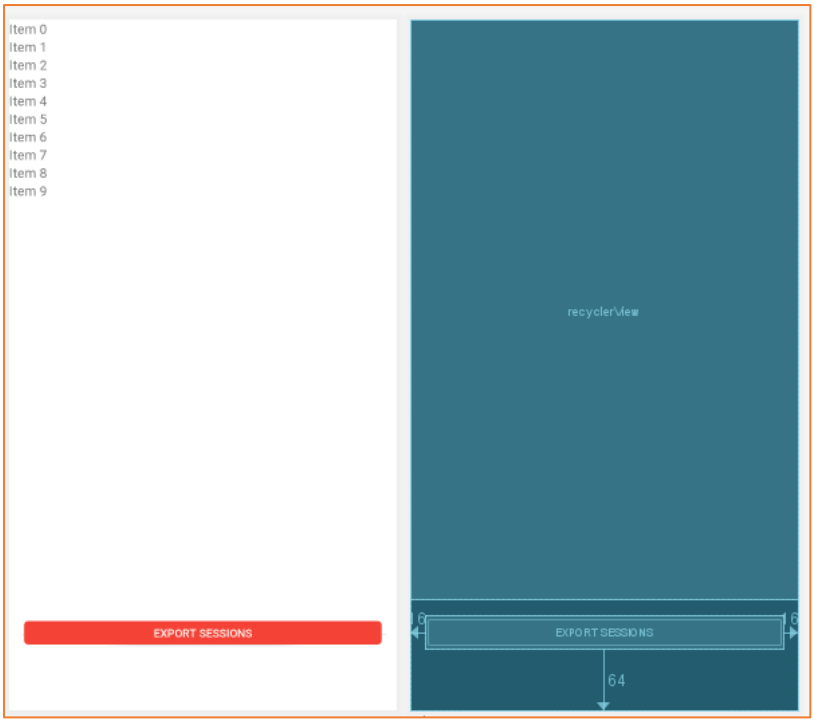
I also renamed grantPermissions to grantLocationPerms() as it was more appropriate.

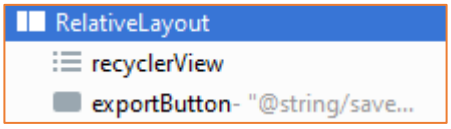
```
case MY_PERMISSIONS_REQUEST_FILE_ACCESS:
    // if permission has not been granted
    if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        grantLocationPerms(); // get location permissions next
    } else {
        // ask the user to grant permissions manually
        manualPermissionDialog("Go to Settings to grant permissions for importing and e...");
    }
    break;
```

I add the following to the switch statement inside the onRequestPermissionsResult method. It carries out a similar method to the location permissions. If the file access permissions are granted then it will request the location permissions otherwise it will guide the user to the settings to grant them manually.



Launching the app for the first time now asks for access to the user's files on launch as implemented above. This now allows me to access the user's storage files to export data.





JounralFragment

Firstly I added a button to the journal fragment xml file. I named it appropriately, exportButton. I had to change the layout to a relative layout so it would be shown relative to the amount of items in the journal. I added a gap between the button and the bottom because that's where the navigation menu is. Otherwise the button would be hidden behind the menu.

```
private Button exportButton; // allow user to save data
```

```
// retrieve UI element
exportButton = (Button) view.findViewById(R.id.exportButton);
```

The exportButton is then initialised in the JournalFragment class. It will carry out a function when pressed by the user.

```
private static final int REQUEST_DOCUMENT_CODE = 7846; // used to identify intent
```

```
// when the export button is presssed
exportButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // start an intent to ask for permission to write to user's files
        // this will show the user a tree view
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT_TREE);
        startActivityForResult(intent, REQUEST_DOCUMENT_CODE);
    }
});
```

Next I setup the action that's executed when the button is pressed by the user. This will start an intent for a new activity to get permission to write to the user's files directory in their phone. I pass it a custom code so I know what to process when it returns.

```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {
    if (resultCode == Activity.RESULT_OK) { // activity exited with success
        Uri treeUri = resultData.getData(); // get the Uri for the selected directory

        Log.d(TAG,  msg: "save location: " + treeUri.getPath()); // log it for debug

        mDb.save(treeUri); // copy the database over to the user's directory
    }
}
```

Handling the result will happen in the JournalFragment::onActivityResult. Here I first make sure the intent completed successfully by comparing it to Activity.RESULT_OK. Once it's successful I can get the Uri which is the path to store the database at. I log this for debugging purposes and then pass it to the database's save method where I will handle the copying of the file.

Copy Database out of App

```
public void destroyInstance() {
    if (instance != null && instance.isOpen() == true) {
        instance.close();
    }

    instance = null;
}
```

The destroyInstance method makes sure to close the database if it is open in the app. If I don't close it then I will get a corrupted database without the data of the app. It first checks make sure the instance is not null so it is not previously destroyed before calling the isOpen method. If it's open then I close it and set the instance to null.

```
if (!destFile.getParentFile().exists())
    destFile.getParentFile().mkdirs();

if (!destFile.exists()) {
    destFile.createNewFile();
}

FileChannel source = null;
FileChannel destination = null;

try {
    source = new FileInputStream(sourceFile).getChannel();
    destination = new FileOutputStream(destFile).getChannel();
    destination.transferFrom(source,  position: 0, source.size());
} finally {
    if (source != null) {
        source.close();
    }
    if (destination != null) {
        destination.close();
    }
}
```

I will go through the save method step by step. I tried many methods with using Java's file system however that result in no access to the files as show below:

Run

Crash

```
I/MyFit-SessionDatabase: save: java.io.IOException: No such file or directory
```

This was because Android has changed the way in which it handles files due to security. To save files the recommended method is to ask the user to select a directory and allow access and then use the new data type DocumentFile to write the file. To do this I tried the following:

Justification

```
public boolean saveCopy(final Uri savePath) {
    File db = context.getDatabasePath(DB_NAME); // get the database file
    Log.d(TAG,  msg: "save: " + db.getPath()); // log the file's name

    // open directory as DocumentFile
    DocumentFile saveDirectory = DocumentFile.fromTreeUri(context, savePath);
    // create new file in the directory, file type is an .sqlite3
    DocumentFile saveFile = saveDirectory.createFile( mimeType: "application/x-sql",  displayName: "OneFitnessData");
```

Fix

First I got the current database from the app's folder. I can access this directly without using DocumentFile as it's part of my app. Next I log it for debugging purposes.

I create a DocumentFile which opens the directory the user chosen using the passed Uri parameter. Next I create a new file in the directory using the createFile method. This expects a MIME datatype and the name of the file. I set the type to a sql file which will add the appropriate extension to the data.

```
// streams to copy files
private static FileInputStream inStream = null;
private static OutputStream outStream = null;
```

Algorithm

```
// due to file handling it's best to add exceptions so all files are closed appropriately
try {
    inStream = new FileInputStream(db); // read data from the app's database
    // output the data to the file we just created
    outStream = context.getContentResolver().openOutputStream(saveFile.getUri());

    // read 2048 bytes each time
    byte[] buffer = new byte[2048];

    int bytesRead;
    // keep reading the data until the end is reached
    while((bytesRead = inStream.read(buffer)) != -1) {
        // save each piece of data in the new file
        outStream.write(buffer,  off: 0, bytesRead);
    }
```

I create member variables to hold the input and output streams. Due to carrying out file handling and input and output it is best I carry this out in a try clause so I can handle any errors occurred without having the app crash each time. Firstly I open the input stream with the database file. I then assign the new DocumentFile I created to the outstream. I use the ContentResovler so I can get access to the application output stream letting Android know where the data is coming from.

To output it's best to do it in buffers so it can be done in chunks and allow the CPU to handle other tasks. I choose to do it in 2048 bytes. The while loop continues copying the data until the end of file is reached which is -1. This way I copy the whole file.

```
} catch (FileNotFoundException e) { // if database was not found
    e.printStackTrace(); // log message but don't crash
} catch (IOException e) { // if there was input with reading and writing
    e.printStackTrace();
} catch (Exception e) { // handle other exceptions
    e.printStackTrace();
```

The catch clauses handle any file management, input and output exceptions and other exceptions. This is to prevent any crashing if there is an error with the files or the input and output buffer. Without this Android would complain as I should be handling these. I print the error out to debug if any of these issues occur.

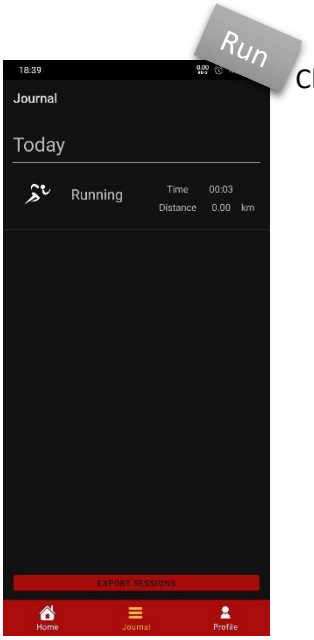
```
        } finally { // make sure to close files
            try {
                // if files have been opened
                if (inStream != null && outStream != null) {
                    // close files
                    inStream.close();
                    outStream.close();
                    return true; // return success
                }

            } catch (IOException e) { // make sure files close properly
                e.printStackTrace();
            }
        }
        return false; // return failure if files were not opened and closed appropriately
}
```

Now the main reason I use the try clause is to make sure the files are closed when either an exception occurs or the file is completely copied over. If I don't do this there is a very high chance the files will corrupt. For this reason I use the try clause so they are closed if they were opened and true is returned to suggest a successful write. Otherwise false is returned.

```
// let user know it saved
Toast.makeText(context, text: "Database exported: " + saveFile.getName(), Toast.LENGTH_SHORT).show();
return true; // return success
```

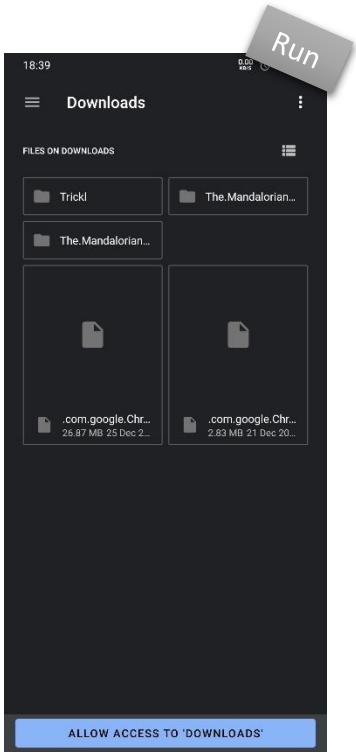
```
// let user know an error occurred
Toast.makeText(context, text: "Failed to export!", Toast.LENGTH_SHORT).show();
return false; // return failure if files were not opened and closed appropriately
```

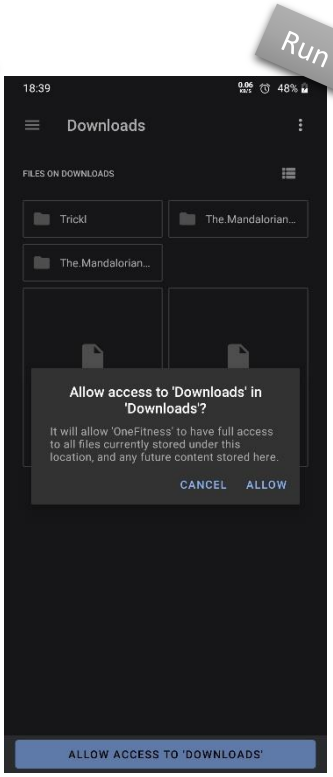
I added two toats which are grey pop-up messages to let the user know that the database has either be exported alongside the files name or it has failed to export.

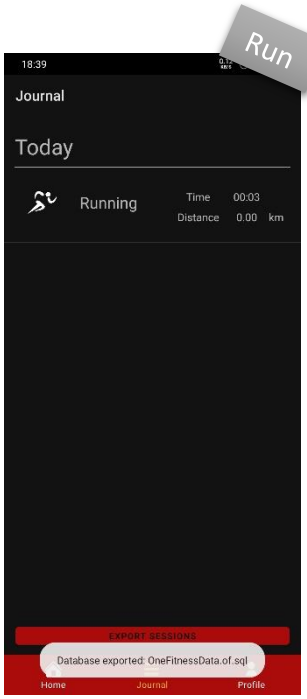


Checked to see the export button was shown as intended in our view so it can be pressed.

When the button was pressed a view popped up to show the user an activity to select the save location as shown in the first screenshot. The user is asked to click the Allow access to downloads which shows a popup asking for permissions.

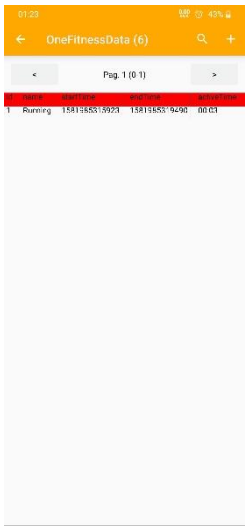






When the permission is granted the save method is executed and the user is shown a toast message letting them know that the file has been saved in their folder.

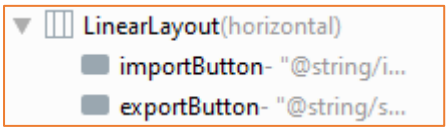
I verified the database had saved correctly by checking the data in another app. As you can see it shows the one entry I had in the database.

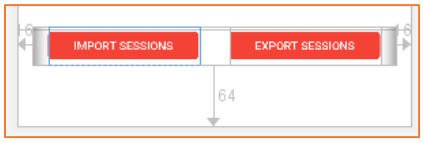


Now that the user can export their data, I need to add an import function.

Importing Database

JuornalFragment – MainActivity





First I created the view in the fragment_journal file. I added a horizontal linear layout and put a new button inside that. I added the relevant strings to show what the new button will do. I then readjusted the height to have the buttons slimmer so they are not as intrusive and do not take up much space as they will now be used as often.

```
private Button importButton; // allow user to import data
```

```
importButton = (Button) view.findViewById(R.id.importButton);
```

The import button is created in the fragment's class and initiated similar to the export button.

```
// when the import button is presssed
importButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // start an intent to select the database file
        // this will show the user a tree view
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        startActivityForResult(intent, REQUEST_READ_CODE);
    }
});
```

I then set the click listener to create a new intent that opens up an activity to select the file saved by my app. This will let the user select the file exported by the export button so they can restore any data they previously had saved and transfer it between devices.

```
private static final int REQUEST_WRITE_CODE = 7846; // used to identify intent
private static final int REQUEST_READ_CODE = 3845; // used to identify intent
```

I added these codes which are sent when the new intents are started so I can differentiate between which activity is returned and how to handle the relevant result.

```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {
    super.onActivityResult(requestCode, resultCode, resultData);
    switch (requestCode) {
        case REQUEST_WRITE_CODE:              // if the save location is chosen
            if (resultCode == Activity.RESULT_OK) { // location chosen with success
                Uri treeUri = resultData.getData(); // get the Uri for the selected directory

                Log.d(TAG,  msg: "save location: " + treeUri.getPath()); // log it for debug

                mDb.save(treeUri); // copy the database over to the user's directory
            }
            break;
        case REQUEST_READ_CODE:                 // if file is opened
            if (resultCode == Activity.RESULT_OK) { // file is read successfully
                Uri fileUri = resultData.getData(); // get the URI of the file

                Log.d(TAG,  msg: "load file: " + fileUri); // log for debugging

                mDb.load(fileUri); // load the file into the database
            }
            break;
    }

}
```

This is now the new modified onActivityResult. First it calls the classes parent onAcitivtyResult to figure out what activity was returned. If it was the write method I do the same thing as before so there is no change there. However now I have a new option which is reading the file. Here I simply get the file's URI, log it and then send it to the database so it can load the file.

Run

Crash

```
android.content.ActivityNotFoundException: No Activity found to handle Intent { act=android.intent.action.OPEN_DOCUMENT }
```

Justification

Running the app now and pressing the import button causes a crash. Through some research and reading over the documentation online I found out that I have to set the type of file I would like the user to select. For this I have to change the save method to save a file in a custom form format. This would allow the user to only select a file that is exported by my app which means I don't have to carry out as much validation with the input file, preventing crashes.

Validation

```
DocumentFile saveFile = saveDirectory.createFile( mimeType: "multipart/form-data",  displayName: "OneFitnessData");
```

Now the file is saved in a custom type as I have changed the mimeType shown above. I can not start an intent to look for this file.

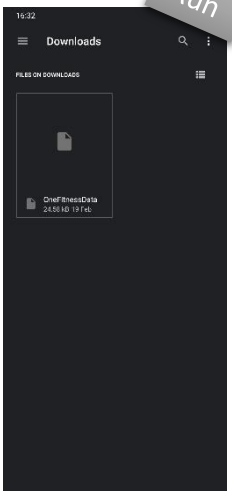
Fix

```
intent.setType("multipart/form-data"); // set the type of file to open
intent.addCategory(Intent.CATEGORY_OPENABLE); // make sure there is read permissions
```

I added the above to the intent in the import button click listener to set the type of file I will be looking for as well as making sure the user has read access. This will then allow my app to read the file only if permission is given otherwise it will prevent a crash.

Now running the app shows the view without crashing as seen on the right.

Run



Copy Database to App

Next I need to implement the load method

```
public boolean load(final Uri fileUri) {
    destroyInstance(); // make sure to close the database
    File db = context.getDatabasePath(DB_NAME); // get the database file to overwrite
    DocumentFile userDatabase = DocumentFile.fromSingleUri(context, fileUri); // get the user chosen file
```

First in the load method I make sure to destroy the database instance so it is closed. Next I open the database file that was closed so I can overwrite it with the file the user has chosen.

```
// due to file handling it's best to add exceptions so all files are closed appropriately
try {
    inStream = context.getContentResolver().openInputStream(userDatabase.getUri()); // read data from database
    // output data to the database file to overwrite
    outStream = new FileOutputStream(db,  append: false);
```

Next I open the files in streams so I can directly overwrite the files with the bytes I read. I do this in a try clause so there is no unexpected crashes if there is a file read/write error just like the save method. I cannot reuse the save method however because the files are handled in the opposite direction.

Algorithm

```
private void copyBytes() throws IOException {
    // read 2048 bytes each time
    byte[] buffer = new byte[2048];

    int bytesRead;
    // keep reading the data until the end is reached
    while((bytesRead = inStream.read(buffer)) != -1) {
        // save each piece of data in the new filemim
        outStream.write(buffer,  off: 0, bytesRead);
    }
}
```

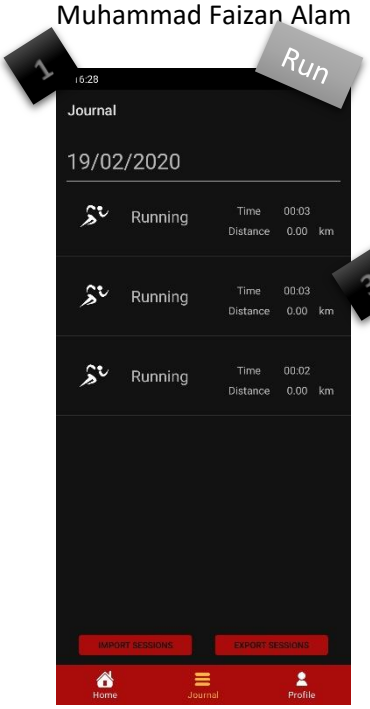
I created a copyBytes method which will be called by save and load to increase code reusability as the code is the same. It throws an IOException which is captured by the save and load method.

`copyBytes();` - inside the save()

```
    copyBytes();
} catch (FileNotFoundException e) { // if database was not found
    e.printStackTrace(); // log message but don't crash
} catch (IOException e) { // if there was input with reading and writing
    e.printStackTrace();
} catch (Exception e) { // handle other exceptions
    e.printStackTrace();
} finally { // make sure to close files
    try {
        // if files have been opened
        if (inStream != null && outStream != null) {
            // close files
            inStream.close();
            outStream.close();
            // let user know database was imported
            Toast.makeText(context, text: "Database imported: " + userDatabase.getName(), Toast.LENGTH_SHORT).show();
            return true; // return success
        }

    } catch (IOException e) { // make sure files close properly
        e.printStackTrace();
    }
}
// let user know an error occurred
Toast.makeText(context, text: "Failed to import!", Toast.LENGTH_SHORT).show();
return false; // return failure if files were not opened and closed appropriately
```

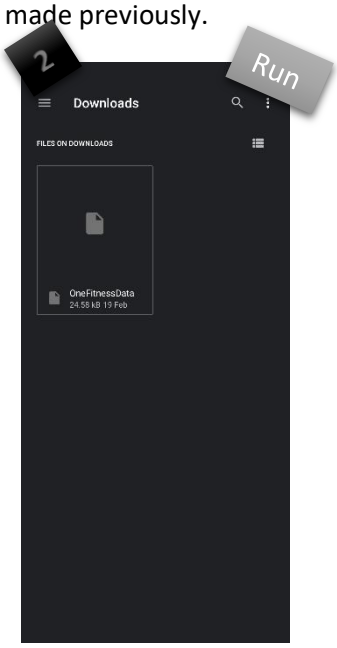
I simply call copyBytes now to use the streams to copy data over exactly. Similar to the save method I catch any exceptions and output them to the log stack. Finally, I make sure the files are closed so there is no corrupted files.

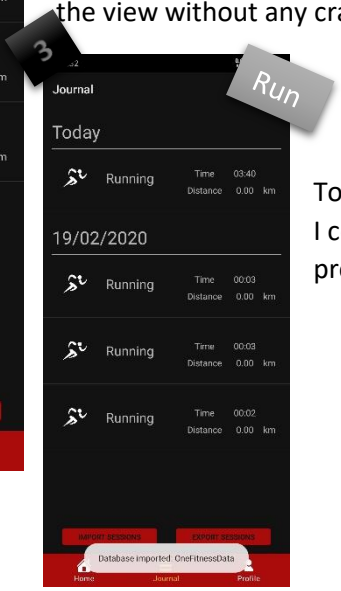
Muhammad Faizan Alam



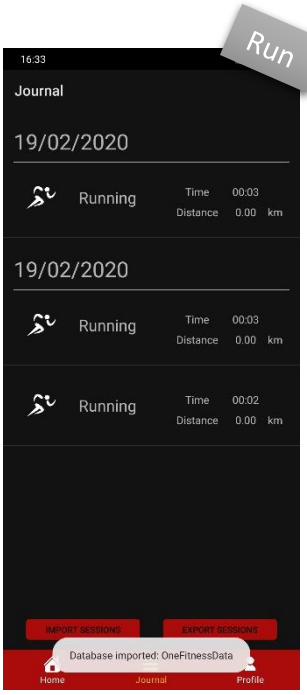
I first exported a file with the following sessions using the button I made previously.

I then clicked the import button which now successfully opened the view without any crashes and showed the file I just exported.





To test the app, I added another session so I could try overwrite it with the backup previously created.



Error

Now when I imported the previous file it overwrote and removed that session just added which is the correct behaviour however there is a duplicate of dates. I presume this is because the RecyclerView is not being reset properly after having the new database created from the backup.

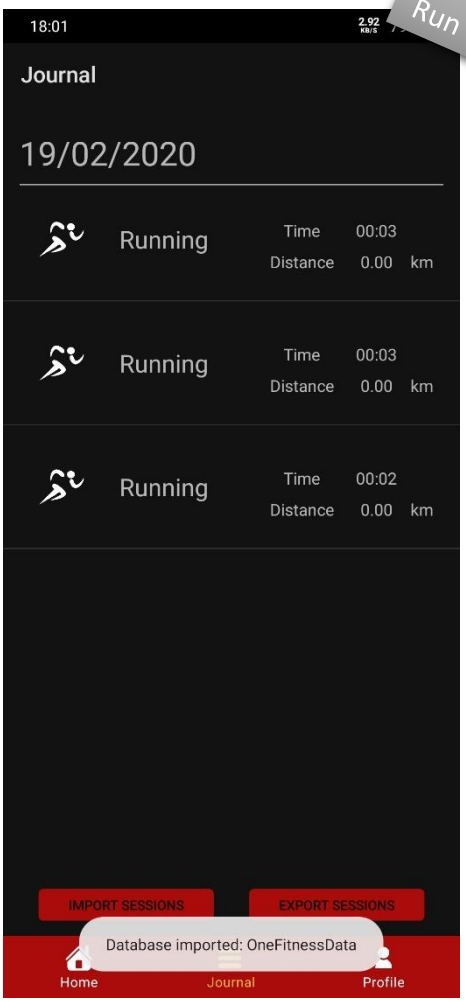
```
mDb.load(fileUri); // load the file into the database
updateView(); // refresh view
```

Fix

```
private void updateView() {
    // get the new database
    mDb = SessionDatabase.getInstance(getContext());
    // remove the recyclerView
    recyclerView.setAdapter(null);
    recyclerView.setLayoutManager(null);
    // setup a new recyclerView
    adapter = new RecyclerSessionAdapter(getContext(), onSessionListener: this);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    // update/redraw the view in the recyclerView
    adapter.notifyDataSetChanged();
    // force a redraw of the whole fragment
    getView().invalidate();
}
```

I first made sure to get the new database. The database class will initialise itself after being recreated by the load method.

To fix this I created a new updateView class that essentially deletes the recyclerView which shows the sessions and dates and then recreated it. I then forced the fragment to redraw the view. This will not delete the whole fragment so memory is not wasted. It will only force the view to be completely redrawn so it can update with the correct items of the new database.



Successful!

Now when I load the previous database the view is shown correctly without any duplicate dates like above. The issue is now fixed and now both importing and exporting has now been implemented allowing the user to transfer data between devices and save it between app deletion and reinstallations!

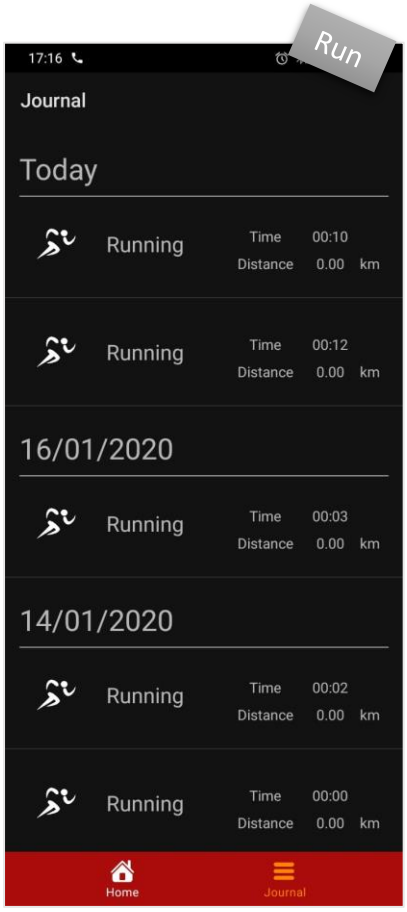
BLANK PAGE

Set Title for Views

I want to the titles for each fragment to be relevant instead of showing the ‘OneFitness’

```
getActivity().setTitle("OneFitness"); // set title
```

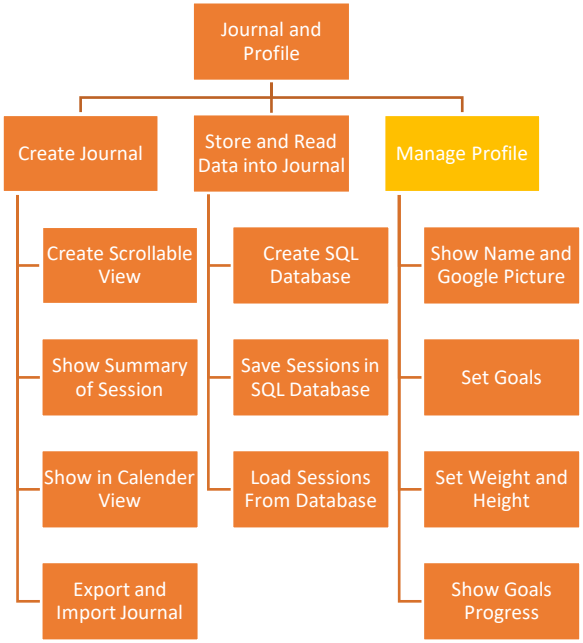
I added the following line with the relevant line for each session in the onCreateView method.



As you can see the top left now has “Juornal” written now instead of the app’s name.

Module 3 - Profile View

Justification



My last focus will be creating the profile view. This will allow the user to see their logged in profile picture and name. It will let them set their goals, weight and height. The goals will be used to show progress on the main screen while the weight and height will be sent to the Google Fit API to update how calculations for calories are made.

Navigation Menu

```
<item
    android:id="@+id/navigation_profile"
    android:icon="@drawable/profileImage"
    android:title="@string/profile" />
```

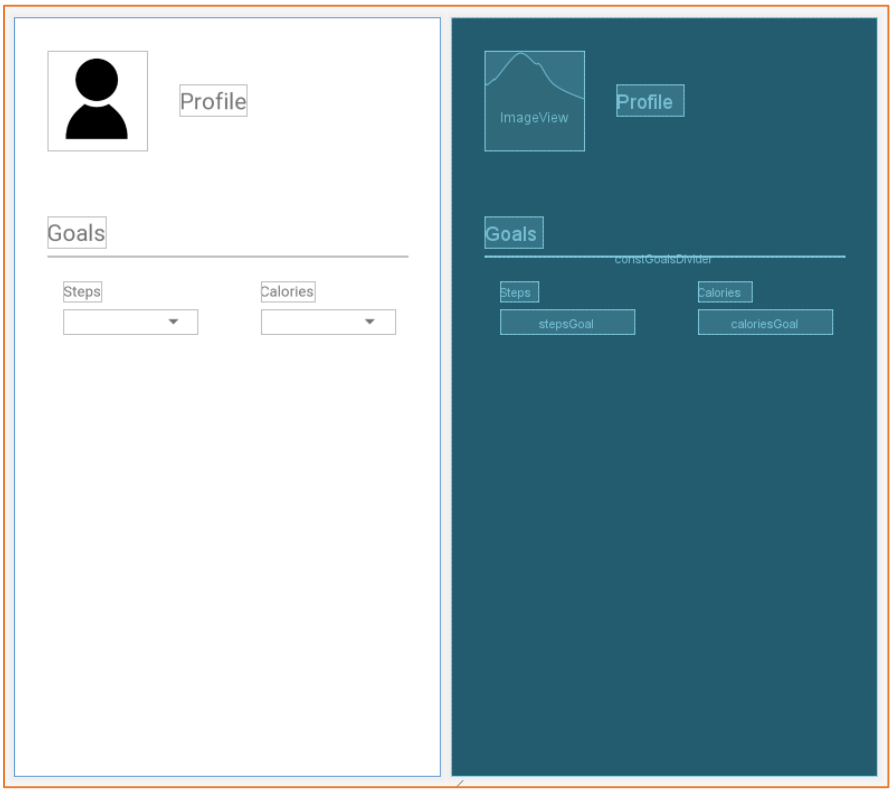
I added another item to the navigation bar with the profile image.

As I can see this adds a third entry to the menu

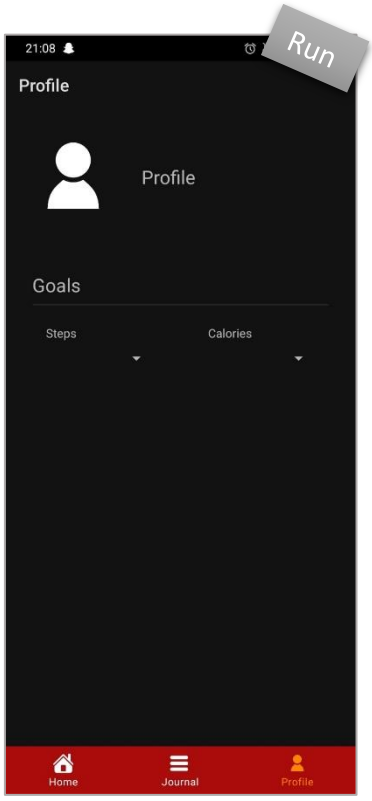
```
case R.id.navigation_profile:
    // change to profile fragment
    fragment = new UserProfile();
    break;
```

To have this work I need to set the case for when it's chosen to start up the fragment for the user's profile which is conveniently called UserProfile.

Profile View – Set up

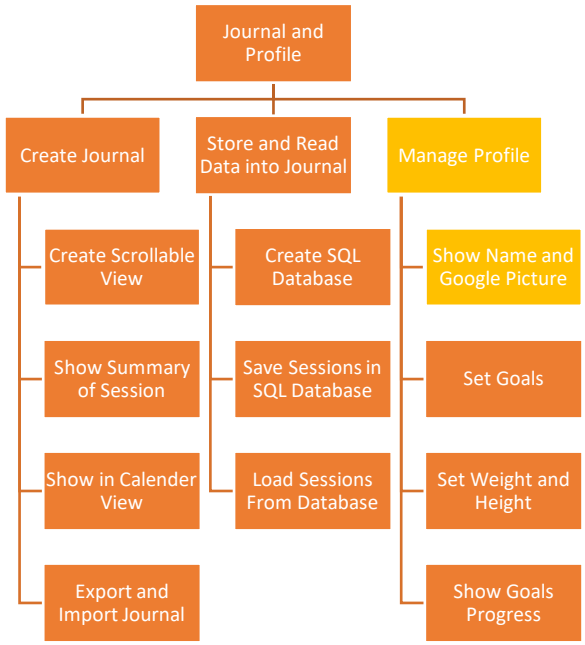


I setup the view in the xml file by adding an ImageView, TextViews and Spinners as well as a horizontal separator to result in the above view in a ConstraintLayout.



Running the app shows the following view. This is my starting view until which I can add onto to result in the final product.

Set Uesr's Info



Next I want to update the above UI with the user's name and their google profile picture they have set which will easily allow them to see what account they have signed in.

```
// access UI elements
private TextView profileName;
private ImageView profileImage;
private Spinner stepsGoal;
private Spinner caloriesGoal;
```

```
// get UI elements
profileName = (TextView) view.findViewById(R.id.profileTextView);
profileImage = (ImageView) view.findViewById(R.id.profileImageView);
stepsGoal = (Spinner) view.findViewById(R.id.stepsGoal);
caloriesGoal = (Spinner) view.findViewById(R.id.caloriesGoal);
```

As usually I make private variables to access the UI elements and then initialise them in the onCreateView method.

I want to change the text size of the spinner and to achieve this I have to set an adapter for the text on the spinners.

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@android:color/darker_gray"
    android:textSize="30sp" />
```

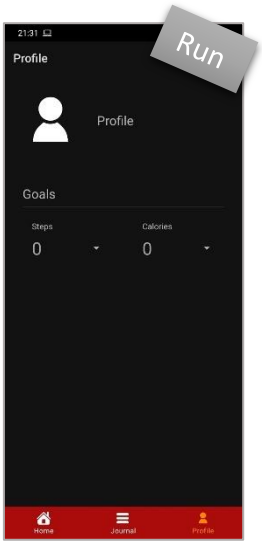
I created a new layout with the larger text at 30sp for now which I will set on the spinner adapters.

entries	@array/number

I set the spinners with an array of the value 0 for placeholder text.

```
// set text format for spinners
ArrayAdapter adapter = ArrayAdapter.createFromResource(getContext(), R.array.number, R.layout.spinner_item);
stepsGoal.setAdapter(adapter);
caloriesGoal.setAdapter(adapter);
```

I then set the adapter on the two spinners.



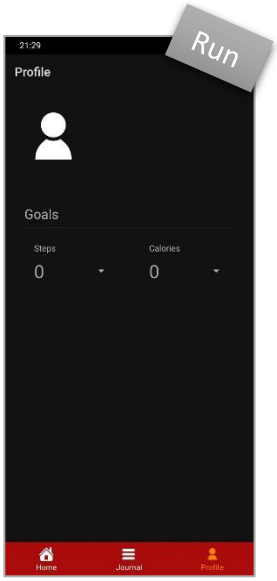
Running the app now shows the larger text on the spinners with the placeholder text of 0.

I want to now update the profile image and their name.

```
// update UI with user's info
setUserData();
```

```
private void setUserData() {
    // sign in with last account
    GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(getContext());
    // get the accounts name and picture
    String personName = acct.getDisplayName();
    Uri personPhoto = acct.getPhotoUrl();
    // set the name and image on the UI elements
    profileName.setText(personName);
    profileImage.setImageURI(personPhoto);
}
```

I declared a new method in the UserProfile class to handle retrieving information from the user's google account. This for now retrieves the user's name and their profile picture so it can be set on the UI elements.



Running the app however does not update the username or profile image.

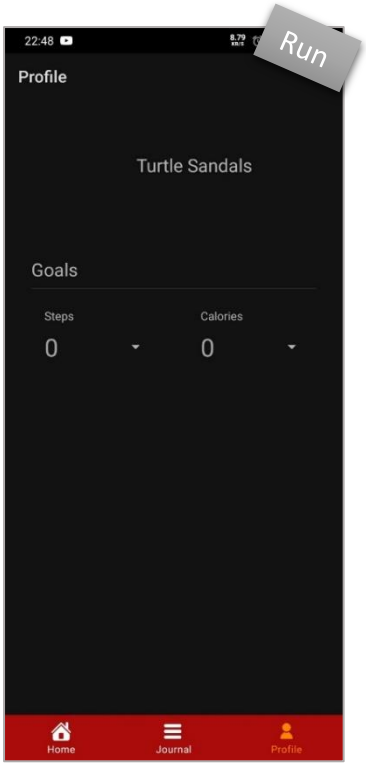
personName = null
personPhoto = null

Debugging the app shows the name and photo data are not retrieved at all as the values are null.

Researching online I found out it was because I had not requested permission for the user's information and had only requested fitness data.

```
// get info for user's profile data
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .requestProfile()
        .addExtension(fitnessOptions)
        .build();
// get permission from user's sign in
GoogleSignInClient signIn = GoogleSignIn.getClient( activity: this, gso);
// start the activity to sign into google
Intent signInIntent = signIn.getSignInIntent();
startActivityForResult(signInIntent, REQUEST_CODE_TRACKER);
```

I fixed this by adding another call to google in the MainActivity onCreate method which requested the user's profile. Google will now let the user know that I are requesting their profile data.



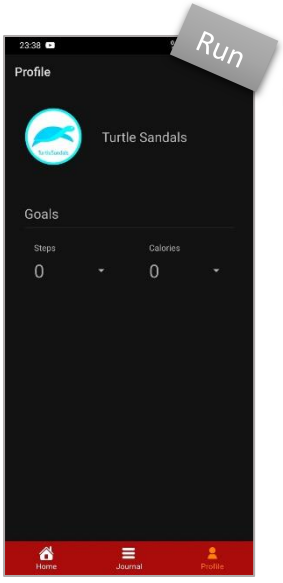
Running the app now showed the name but not the image. This was because ImageView can only show local URI images and not ones stored on a server such as the user's image.

```
implementation 'com.master.android:glideimageview:2.1'
```

I am going to use GlideImageView to fetch and show the image. This is recommended by Google themselves in their documentation.

```
<FrameLayout
    android:id="@+id/profileImageFrame"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <com.master.glideimageview.GlideImageView
        android:id="@+id/profileImageView"
        android:layout_width="96dp"
        android:layout_height="96dp"
        android:src="@drawable/profileImage"
        app:error_res="@drawable/no_image"
        app:placeholder_res="@drawable/profileImage"
        app:show_progress="true"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</FrameLayout>
```

I changed the ImageView to a GlideImageView but I ran into the problem of it not working with ConstraintLayout. As a workaround I tried putting it inside a FrameLayout as shown above and that worked!



Running the app now fetches the image. Marking this part finished.

Now testing with an account that didn't have a profile picture caused the following error:

Crash Run

```
java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.String android.net.Uri.toString()' on a null object reference
    at com.faizan.onefitness.profile.UserProfile.setUserData(UserProfile.java:87)
```

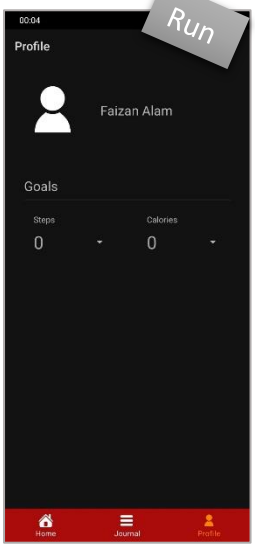
I fixed this by adding some validation to the data with an if statement to make sure the data was valid and existed before trying to access it.

Fix Validation

```
// set the name and image on the UI elements
profileName.setText(personName);
if (personPhoto != null) {
    profileImage.loadImageUrl(personPhoto.toString());
}
```

This checks that the photo URI is not null before trying to retrieve the URL and set the image.

Validation Testing



Running the app now shows the user's name without crashing due to not having an image. It continues to show the default profile image.

Set Goals

Next I want to handle saving and restoring the goals using dialogs. These goals will be sent to the main screen as well to show but I will implement that at the very end. I will have to setup validation here as well as to not get any out of bounds data.

```
<TextView
    android:id="@+id/caloriesGoal"
    android:layout_width="130dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:drawableEnd="@drawable/dropdown_arrow"
    android:drawablePadding="5dp"
    android:singleLine="true"
    android:text="@string/_0"
    android:layout_marginTop="8dp"
    android:textColor="@android:color/darker_gray"
    android:textSize="30sp"
    app:layout_constraintStart_toStartOf="@+id/constCalories"
    app:layout_constraintTop_toBottomOf="@+id/constCalories"/>
```

I had to switch the spinners to TextViews with an arrow at the end of it to represent a spinner. I want it to be clicked and open a custom dialog instead of the spinner dialog.

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">#
    <item>
    <bitmap android:src="@android:drawable/arrow_down_float"
        android:gravity="center" android:tint="@color/colorPrimary" />
    </item>
</selector>
```

I also made an arrow and coloured it the primary colour orange of the app to use in the TextViews.

```
private static final int STEPS_DIALOG = 5494;
private static final int CALORIES_DIALOG = 8494;
```

```
// show dialog when goals are clicked
stepsGoalText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openAlertDialog(STEPS_DIALOG, title: "Set Steps Goal", positive: "Set", negative: "Cancel");
    }
});

caloriesGoalText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openAlertDialog(CALORIES_DIALOG, title: "Set Calories Goal", positive: "Set", negative: "Cancel");
    }
});
```

To reduce code repetition, I will make the same dialog with custom titles and positive and negative buttons.

```
// get data from sharedPreference
sharedPrefs = getActivity().getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);
// retrieve data for steps and calories if stored or 0
int steps = sharedPrefs.getInt(STEPS_TAG, defValue: 0);
int calories = sharedPrefs.getInt(CALORIES_TAG, defValue: 0);
// set the steps and calories text
stepsGoalText.setText(String.valueOf(steps));
caloriesGoalText.setText(String.valueOf(calories));
```

I then added the code above to the restoreUserData method so it can retrieve the steps and calories for the gaols if they were previously saved.

```
private void openAlertDialog(final int result, String title, String positive, String negative) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle(title);
    // get input from the user
    final EditText input = new EditText(getContext());
    // make dialog only accept
    input.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_TEXT_VARIATION_PASSWORD);
    input.setFilters(new InputFilter[] {new InputFilter.LengthFilter( max: 5)});
    // open alert dialog
    builder.setView(input);

    // set the text to the correct goal
    switch (result) {
        case STEPS_DIALOG:
            input.setText(stepsGoalText.getText());
            break;
        case CALORIES_DIALOG:
            input.setText(caloriesGoalText.getText());
            break;
    }
```

Algorithm

The alert dialog is first formatted to have an EditText view which will be used to get the user's input. This EditText is set to only accept numbers and not allow input past 5 numbers.

Validation

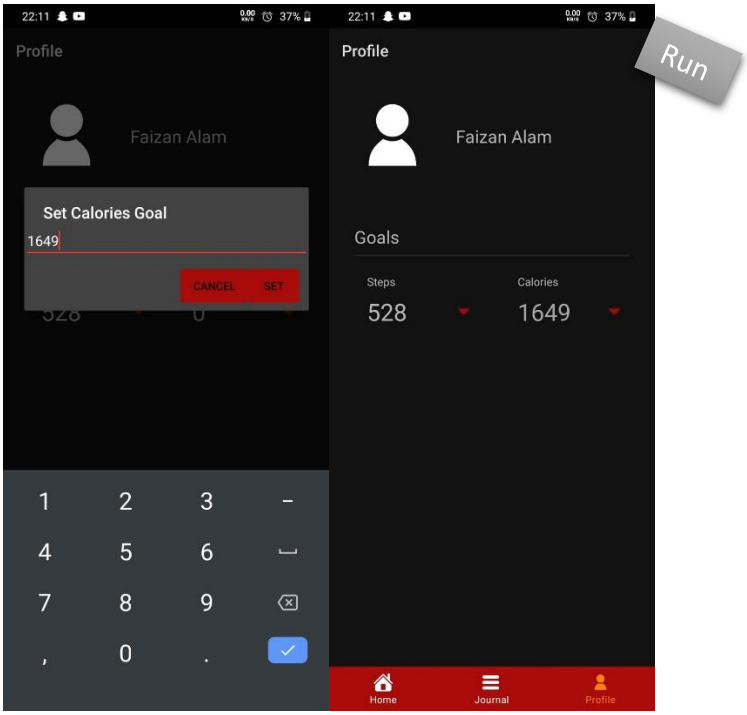
This is a form of validation to make sure the user does not enter data that my app cannot convert to an integer as well as set a limit because the goal will never be more than 99999 steps/calories. The EditText is set to the previously saved values for the goals

```
// Set up the buttons
builder.setPositiveButton(positive, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // set the new steps/calories goal
        alertDialogResult(result, input.getText().toString());
    }
});
builder.setNegativeButton(negative, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
    }
});

builder.show();
```

At the end of the method I then set a listener for when the positive button is pressed. This calls another method to handle setting the new data. The negative button just dismisses the dialog.

```
private void alertDialogResult(final int result, String input) {
    // if the input is not empty then update it
    if (!input.isEmpty()) {
        // handle result from dialog for step or calories
        switch (result) {
            case STEPS_DIALOG:
                Log.d(TAG, msg: "Set steps dialog");
                stepsGoalText.setText(input);
                sharedPrefs.edit().putInt(STEPS_TAG, Integer.parseInt(input)).apply();
                break;
            case CALORIES_DIALOG:
                Log.d(TAG, msg: "Set calories dialog");
                caloriesGoalText.setText(input);
                sharedPrefs.edit().putInt(CALORIES_TAG, Integer.parseInt(input)).apply();
                break;
        }
    }
}
```

Finally, the alertDialogResult handles the result for the two different dialogs to set the steps/calories goal as well as save them in Shared Preferences for the app.

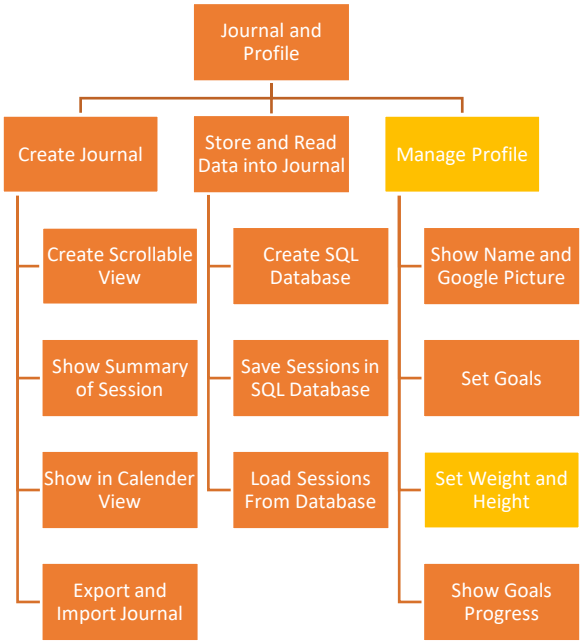


Now when the app is run, the calories and steps are set and finally the app is restarted it shows the previous steps and calories by restoring them as shown above.

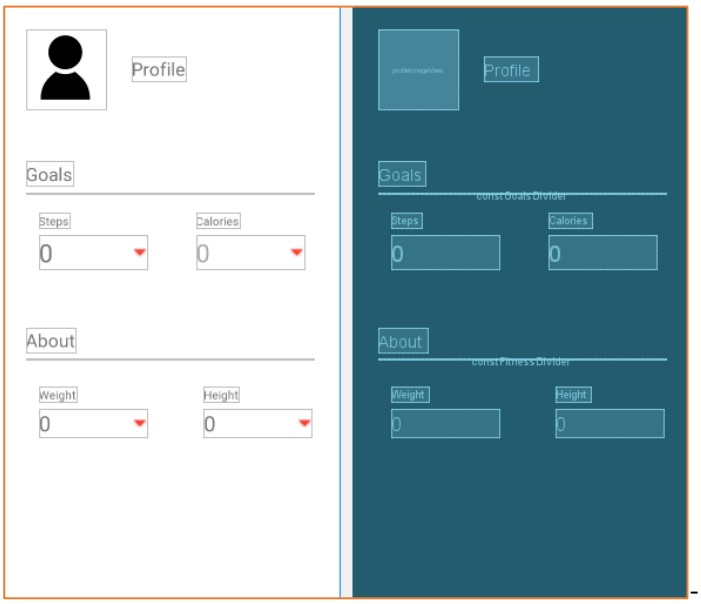
Validation Testing

As you can see only a keybad with numbers is shown meaning it will only accept numbers. There is also a limit so anything above 99999 will not work but even if it was accepted there would be no errors in the app.

Weight and Height

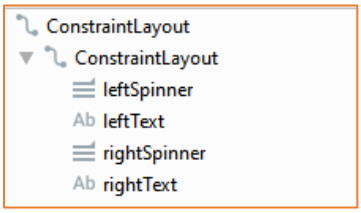


Next for the profile I want to allow the user to set their weight and height which means it first has to be retrieved from their Google account. Once it is set in the dialog it needs to be sent to the API.



- fragment_user_profile.xml

First, I set up the UI two display the user's weight and height with the labels under a heading "About" as shown above. This will allow the user to see their set weight and height and change it.



- spinner.xml

When each TextView is clicked on it will open a custom spinner view with two spinners and a text view after it as show above.

Retrieve from Google API

```
private TextView weightText;
private TextView heightText;
```

```
weightText = (TextView) view.findViewById(R.id.weightText);
heightText = (TextView) view.findViewById(R.id.heightText);
```

I initialise two TextViews to hold the items from the view.

Algorithm

```
private void readFitData() {
    // make a read request to get get the weight and height
    DataReadRequest readRequest = new DataReadRequest.Builder()
            // the start time is set to 1 and end time is now
            .setTimeRange( l: 1, Calendar.getInstance().getTimeInMillis(), TimeUnit.MILLISECONDS)
            .read(DataType.TYPE_WEIGHT)
            .read(DataType.TYPE_HEIGHT)
            .build();

    Fitness.getHistoryClient(getContext(), GoogleSignIn.getLastSignedInAccount(getContext())) // get the historyClient
            .readData(readRequest)      // read weight and height from FIT
            .addOnSuccessListener(                          // listener executed when method succeeds
                    new OnSuccessListener<DataReadResponse>() {

                        @Override
                        public void onSuccess(DataReadResponse dataReadResponse) {
                            // get the weight data from the response from the API
                            DataSet weightData = dataReadResponse.getDataSet(DataType.TYPE_WEIGHT);
                            // get the weight converted as a float. If it's empty get the value 0
                            float weightResponse = weightData.isEmpty() ? 0 : weightData.getDataPoints().get(0).getValue(Field.FIELD_WEIGHT).asFloat();

                            // get the height data from the response from the API
                            DataSet heightData = dataReadResponse.getDataSet(DataType.TYPE_HEIGHT);
                            // get the weight converted as a float. If it's empty get the value 0
                            float heightResponse = heightData.isEmpty() ? 0 : heightData.getDataPoints().get(0).getValue(Field.FIELD_HEIGHT).asFloat();

                            // round the values to two decimal places
                            weight = Double.valueOf(twoDForm.format(weightResponse));
                            height = Double.valueOf(twoDForm.format(heightResponse));
                            // update textViews with data
                            weightText.setText(weight + " kg");
                            heightText.setText(height + " m");
                            // log data
                            Log.d(TAG,  msg: "weight: " + weight);
                            Log.d(TAG,  msg: "height: " + height);
                        }
                    })
            .addOnFailureListener( // when method fails
                    (e) -> {
                        Log.i(TAG,  msg: "Error: " +  e); // log error
                    });
}
```

The readFitness data is called when the view is launched to retrieve the user's current weight and height. This call is similar to other calls to the API except this time I request the weight and height and handle them in the success listener using validation.

One other difference is the start and end time I use. The start time is set to 1 while the end time is set to the current time the app is run. This means the average weight input by the user ever since they registered on the app is shown.

Next in the onSuccessListener I get the two DataSets for both the weight and height. I then reduce the values to two decimal points so they can be displayed to the user. The user does not want exact values and two decimal places should do.

Finally I update the TextViews on the UI and log the messages as well as any errors if it fails.

```
readFitData(); // get weight and height from api
```

This method is then called at the end of restoreValues method, so it is called whenever the fragment is created.

Open AlertDialogs

```
private static final int HEIGHT_DIALOG = 1848;
private static final int WEIGHT_DIALOG = 2494;
```

```
// open dialog when textviews are clicked
weightText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        scrollAlertDialog(WEIGHT_DIALOG);
    }
});

heightText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        scrollAlertDialog(HEIGHT_DIALOG);
    }
});
```

I create two int values to pass to my scrollAlertDialog to distinguish between the two types of dialogs when they are pressed. I set the onClickListeners on the TextViews to call the dialog with the appropriate static int value.

```
// dialog used for weight and height
private void scrollAlertDialog(final int result) {
    AlertDialog.Builder builderSingle = new AlertDialog.Builder(getContext());

    // inflate the dialog with a custom view
    View holder = View.inflate(getContext(), R.layout.spinners,  root: null);
    // retrieve the UI elements
    Spinner leftSpinner = holder.findViewById(R.id.leftSpinner);
    Spinner rightSpinner = holder.findViewById(R.id.rightSpinner);
    TextView leftText = holder.findViewById(R.id.leftText);
    TextView rightText = holder.findViewById(R.id.rightText);

    // use to populate spinners
    List<String> leftSpinnerArray = new ArrayList<>();
    List<String> rightSpinnerArray = new ArrayList<>();
```

The following method is very complex and large so I will break-it down bit by bit. Firstly I setup a new AlertDialog as usual and set our custom spinner.xml view on it that I created above. Next I retrieve the UI elements which are spinners and TextViews. I then create two lists of strings to hold the values that will be inside the spinners.

Algorithm

```
// check if dialog was called on weight or height
switch (result) {
    case WEIGHT_DIALOG:
        builderSingle.setTitle("Set Weight"); // set the title
        // populate the first spinner with values from 20 to 650
        for(int i = 20; i <= 650; i++){
            leftSpinnerArray.add(String.valueOf(i));
        }
        // populate the second spinner with values from 0 to 99
        for(int i = 0; i <= 99; i++){
            rightSpinnerArray.add(String.valueOf(i));
        }
        // set the first text to a decimal point
        leftText.setText(".");
        rightText.setText("kg"); // and the second to kg to show unit
        break;
    case HEIGHT_DIALOG:
        builderSingle.setTitle("Set Height"); // set title
        // populate left spinner from 30 to 300
        for(int i = 30; i <= 300; i++){
            leftSpinnerArray.add(String.valueOf(i));
        }
        // set left text to cm for unit
        leftText.setText("cm");

        // remove both right spinner and TextView
        rightSpinner.setVisibility(View.GONE);
        rightText.setVisibility(View.GONE);
        break;
}
```

Next I check whether the dialog was called on the weight or the height so the spinners can be populated with different values and appropriate titles set.

Validation

The weight populates the first spinner from 20 to 650 which will hold the whole number of KGs. The decimal part runs from 0 to 99 as expected. I also set the decimal point on the TextView as well as the unit to be “kg”.

For the height I populate only the leftSpinner with values from 30 to 300 and set the unit to “cm”. I then hide the rightSpinner and right Text view as they are not required.

```
// Use an adapter to populate the spinner with the leftSpinnerArray and rightSpinnerArray
ArrayAdapter<String> leftAdapter = new ArrayAdapter<~>(
        getContext(), android.R.layout.simple_spinner_item, leftSpinnerArray);
ArrayAdapter<String> rightAdapter = new ArrayAdapter<~>(
        getContext(), android.R.layout.simple_spinner_item, rightSpinnerArray);
// spinners should be drop down view's
leftAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
rightAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// finially set the adapter so the values are set on the spinner
leftSpinner.setAdapter(leftAdapter);
rightSpinner.setAdapter(rightAdapter);
```

Now that I have the list of strings I have to create ArrayAdapters to populate the spinners with them. This is required because the adapter converts the list values into that which the Spinner view can show to the user.

I have two adapters, once for each spinner and have them set as default spinner dialogs. I set these on the respective spinners to populate them.

Algorithm

```
// set the dialog values to those stored in the weight and height variables
switch (result) {
    case WEIGHT_DIALOG:
        // round to two decimal points without whole number
        double roundOff = (double) Math.round((weight.doubleValue() - weight.intValue()) * 100) / 100;
        // get the two decimal points as a whole integer i.e 0.43 -> 43
        int decimalPart = (int) (100 * roundOff);

        // get right and left spinner positions using the values from the weight variable
        int rightSpinPos = leftAdapter.getPosition(String.valueOf(decimalPart)) + 1;
        int leftSpinPos = leftAdapter.getPosition(String.valueOf(weight.intValue()));

        // finally set the spinners to show the above values as default
        leftSpinner.setSelection(leftSpinPos);
        rightSpinner.setSelection(rightSpinPos);
        break;
```

Now that I have the spinners populated with the relevant values I need to set the starting values to that of the user's current weight and height. For example if the user's weight is 62.5 kg then the dialog should start with the value 62 on the left spinner and 5 on the right spinner so the user can see their weight and change accordingly.

Justification

To handle this first I remove the whole number part of the value by subtracting the whole part from it by converting it to an integer. The table below shows what exactly happens

Value	Calculation	New Value
62.52	62.52 – 62	0.52

Next I need to convert this 0.5 to 5 so it can be shown in the next spinner so it is multiplied by 100 to convert it into a whole number:

Value	Calculation	New Value
0.52	0.52 * 100	52

Now I can find the whole value (62) on the left spinner and the decimal value (52) on the right spinner so it shows the two values.

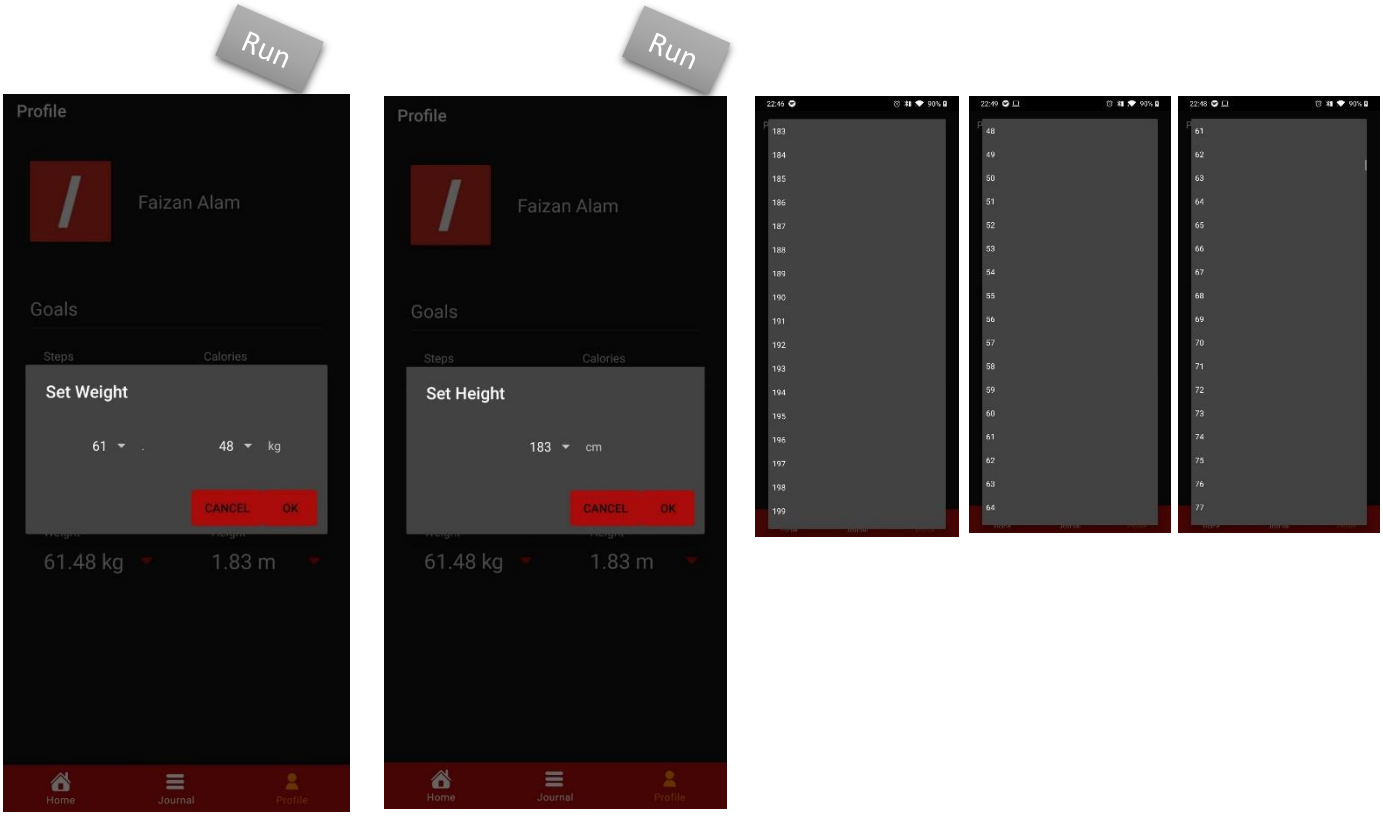
Now that I have handled the dialog for the weight I need to do the same if the AlertDialog is shown for the height.

```
    case HEIGHT_DIALOG:
        // convert the height from meters to centi-meters
        int heightInCm = (int) (height * 100);
        // get the string value of the integer
        int pos = leftAdapter.getPosition(String.valueOf(heightInCm));
        // set leftSpinner to show the height as default value
        leftSpinner.setSelection(pos);
        break;
}
```

The other part of the switch statement get's the height in centimetres by multiplying the height which is stored in metres by 100. I then convert this into a string so it can be found in the spinner and set as the default value.

```
    builderSingle.setView(holder); // set the view on the alert dialog
    // set the ok and cancel buttons
    builderSingle.setNegativeButton( text: "cancel", (dialog, which) → {
            dialog.dismiss();
    });

    builderSingle.setPositiveButton( text: "Ok", (dialog, which) → {
            dialog.dismiss();
    });
    builderSingle.show();
}
```

Finally I set the view on the dialog and set the positive and negative buttons which both only dismiss the dialog for now as I have not implemented the insertion of weight and height.



I ran the app, clicked on the weight text which showed the AlertDialog. The AlertDialog had the spinners set to the correct values of the user's current weight. I did the same with the height which also had the correct value shown as the default one.

Next, I clicked on the spinners to check if the spinner dialogs were shown. As the screenshots show they are shown above.

Validation Testing

As the dialogs only show a range of values it means no validation has to be performed directly on the input. If I were to accept user typing then I would have to make sure the values were valid but here I set up a range of valid values for weight and height so the user cannot choose anything that's invalid.

Set Weight and Height

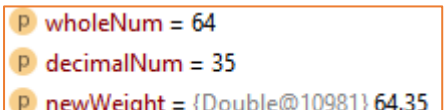
```
final Spinner leftSpinner = holder.findViewById(R.id.leftSpinner);
final Spinner rightSpinner = holder.findViewById(R.id.rightSpinner);
```

I need to set the Spinner's as final so they are not changed and can be changed in the Listener class inside the method. I would get an error without this.

```
private void scrollAlertDialogResult(final int result, final Spinner leftSpinner, final Spinner rightSpinner) {
    switch (result) {
        case WEIGHT_DIALOG:
            // get the whole number, left spinner value as int
            int wholeNum = Integer.parseInt(leftSpinner.getSelectedItem().toString());
            // get decimal part of weight/rigth spinner
            int decimalNum = Integer.parseInt(rightSpinner.getSelectedItem().toString());
            // convert the value to double before dividing by 100 and add whole weight
            weight = wholeNum + (((double) decimalNum) / 100d);
            // set new weight in FIT API
            insertFitData(DataType.TYPE_WEIGHT, weight.floatValue());

            // log new weight
            Log.d(TAG,  msg: "new weight: " + weight);
            break;
        case HEIGHT_DIALOG:
            // get height selected by user
            int heightInCm = Integer.parseInt(leftSpinner.getSelectedItem().toString());
            height = ((double) heightInCm) / 100; // convert cm to metres by dividing by 100
            // set new weight in FIT API
            insertFitData(DataType.TYPE_HEIGHT, height.floatValue());

            // log new height
            Log.d(TAG,  msg: "new height: " + height);
            break;
    }
    readFitData(); // set new weight/height on profile screen
}
```

This method will be called when the user has selected a new weight or height. First I check whether it's a new weight or height being set.

For the weight I do the reverse of the algorithm above by getting the two whole values from the two spinners as integers. The second value is cast to a double before being divided by 100 so I can convert the two digit number to a decimal value. I then add the whole number to get the new weight which is assigned to the weight value.



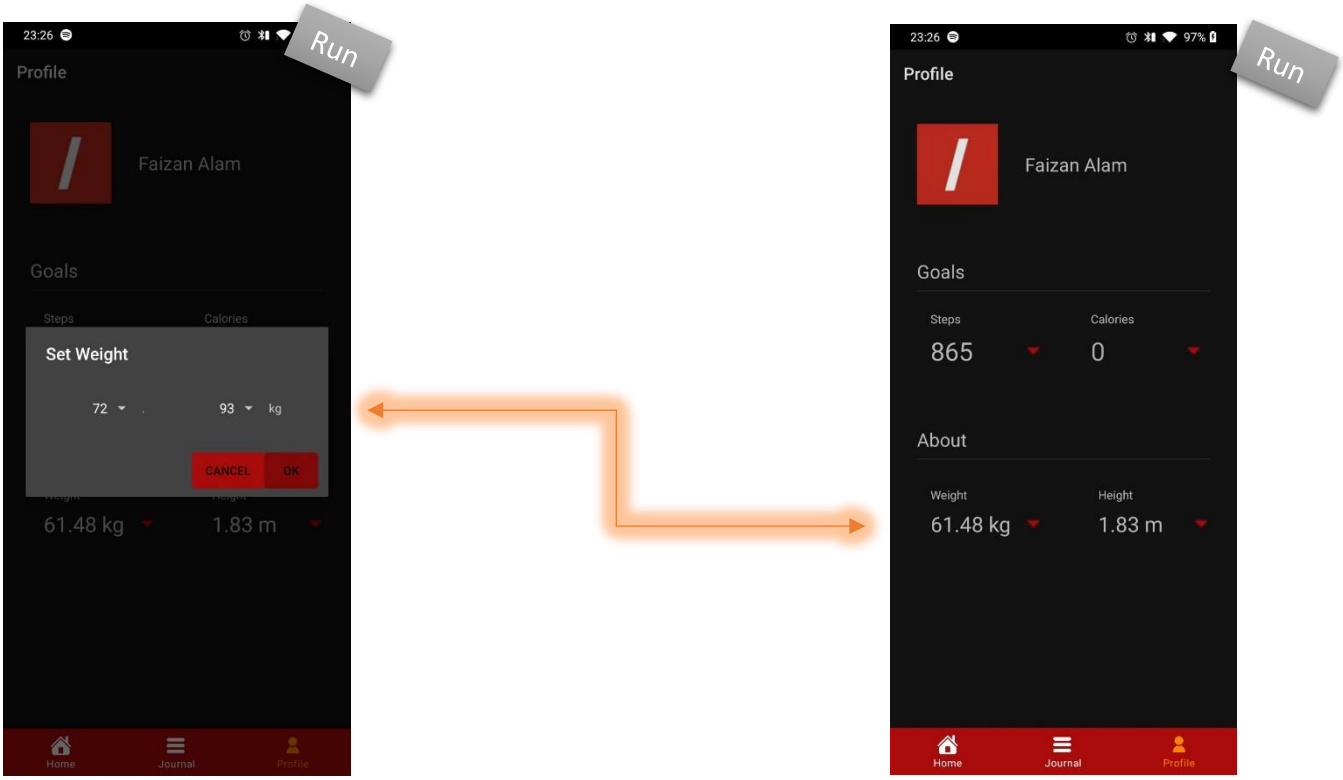
Here the algorithm is shown in action in debug mode.

For the height I simply first cast to a double before dividing by 100. This converts the centimetre to metres.

For both I insert the data into the API before logging the value and updating the profile screen by retrieving the new weight and height from the API. The value is convert from a double to a float to be inserted into the API as it expects a float type.

```
private void insertFitData(DataType dataType, final float value) {
    long time = Calendar.getInstance().getTimeInMillis(); // get current time
    // build data type to add new api data to
    DataSource dataSource = new DataSource.Builder()
            .setAppPackageName(getContext())
            .setDataType(dataType)
            .setType(DataSource.TYPE_RAW)
            .build();
    // create data pint with the type of data and the value
    DataPoint dataPoint = DataPoint.builder(dataSource)
            .setTimeInterval(time, time, TimeUnit.MILLISECONDS)
            .setFloatValues(value)
            .build();
    // make the DataSet the API expects
    DataSet dataSet = DataSet.builder(dataSource)
            .add(dataPoint)
            .build();
    // finially get the history client and add the data to it
    Fitness.getHistoryClient(getContext(), GoogleSignIn.getLastSignedInAccount(getContext()))
            .insertData(dataSet) Task<Void>
            .addOnSuccessListener((OnSuccessListener) (aVoid) → {
                    Log.i(TAG,  msg: "Insert " + value); // log success
            }) Task<Void>
            .addOnFailureListener((e) → {
                    Log.d(TAG,  msg: "Failed to insert " + value); // log failiure
            });
}
```

The insertFitData works in reverse to the API calls I have made so far. It first builds a DataSource to add data to. This DataSource holds the context of the application so the API knows which app the call comes from. Next I make the DataPoint to assign a new value to the DataType that I pass into the method such as weight or height. Finally I can insert this data from my app into the API by calling the historyClient and it's insertData() method. I log the success or failure of it.



Running the app now and setting a new weight shows that it is assigned to the weight value as well as inserting it into the API as show by the log below.

```
ProfileFragment: new weight: 71.29
ProfileFragment: Insert 71.29
```

```
ProfileFragment: new height: 1.95
ProfileFragment: Insert 1.95
```

Not an error

Justification

However, what I notice is that the weight is not changed even though the user set a new weight. This however is not an error. There are multiple reasons for this. Firstly, it sets a new weight which means when I retrieve the user's weight from the API it gets the average weight since they have set it. This means new changes do not affect the value straight away. Secondly the FIT API takes time to update in the cloud before retrieving the new value. None of these are issues as this is how Google FIT handles both the weight and height to calculate other data.

Save and restore Weight and Height

To solve the issue above as the user may believe something is wrong with the app, I will save the value they set in the app and use that. It will take time to update in the cloud so it's best I handle the UI updates in the app.

```
private static final String WEIGHT_TAG = "WEIGHT_PROFILE";
private static final String HEIGHT_TAG = "HEIGHT_PROFILE";
```

I first made two tags to use with SharedPreferences to save the data in the app's memory.

```
// save weight in app data
sharedPrefs.edit().putFloat(WEIGHT_TAG, weight.floatValue()).apply();
```

```
// save height in app data
sharedPrefs.edit().putFloat(HEIGHT_TAG, height.floatValue()).apply();
```

I saved the weight or height in the app's data when it is updated using the tags above.

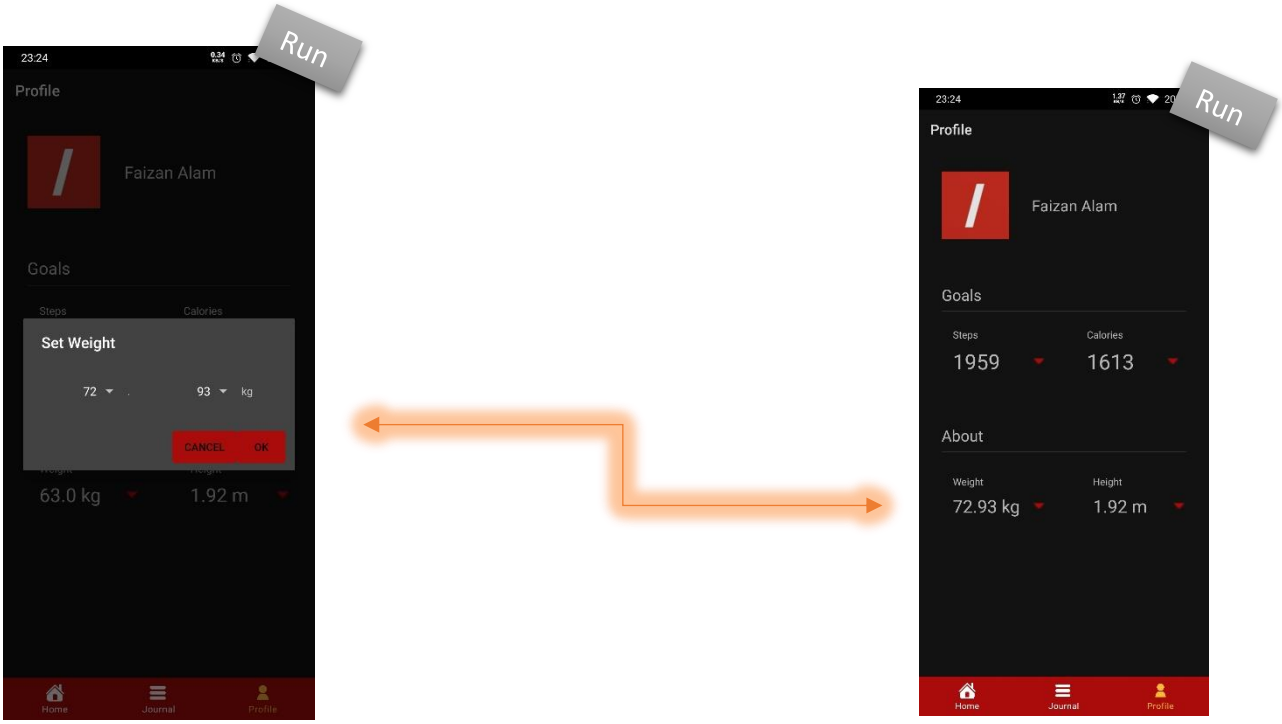
```
private void restoreFitData() {
    // retrieve the values, if they don't exist get -1
    float savedWeight = sharedPrefs.getFloat(WEIGHT_TAG,  defValue: 0);
    float savedHeight = sharedPrefs.getFloat(HEIGHT_TAG,  defValue: 0);

    if (savedWeight != 0) { // only set the value if it is saved
        weight = Double.valueOf(twoDForm.format(savedWeight));
        weightText.setText(weight + " kg"); // update UI
```

I renamed the readFitData to restoreFitData and modified it to get the saved values. First I check if the value is not 0, if it's not 0 then we get the saved value and set it to the weight in the app.

```
} else { // else get the height from their google account
    // make a read request to get get the weight
    DataReadRequest readRequest = new DataReadRequest.Builder()
            // the start time is set to 1 and end time is now
            .setTimeRange( l: 1, Calendar.getInstance().getTimeInMillis(), TimeUnit.MILLISECONDS)
            .read(DataType.TYPE_WEIGHT)
            .build();

    Fitness.getHistoryClient(getContext(), GoogleSignIn.getLastSignedInAccount(getContext())) // get the historyClient
            .readData(readRequest)     // read weight and height from FIT
            .addOnSuccessListener(                          // listener executed when method succeeds
                    (OnSuccessListener) (dataReadResponse) → {
                        // get the weight data from the response from the API
                        DataSet weightData = dataReadResponse.getDataSet(DataType.TYPE_WEIGHT);
                        // get the weight converted as a float. If it's empty get the value 0
                        float weightResponse = weightData.isEmpty() ? 0 : weightData.getDataPoints().get(0).getValue(Field.FIELD_WEIGHT).asFloat();

                        // round the values to two decimal places
                        weight = Double.valueOf(twoDForm.format(weightResponse));
                        // update textViews with data
                        weightText.setText(weight + " kg");
                        // log data
                        Log.d(TAG,  msg: "weight: " + weight);
                    })
            .addOnFailureListener( // when method fails
                    (e) → {
                        Log.i(TAG,  msg: "Error: " +  e); // log error
                    });
}
```

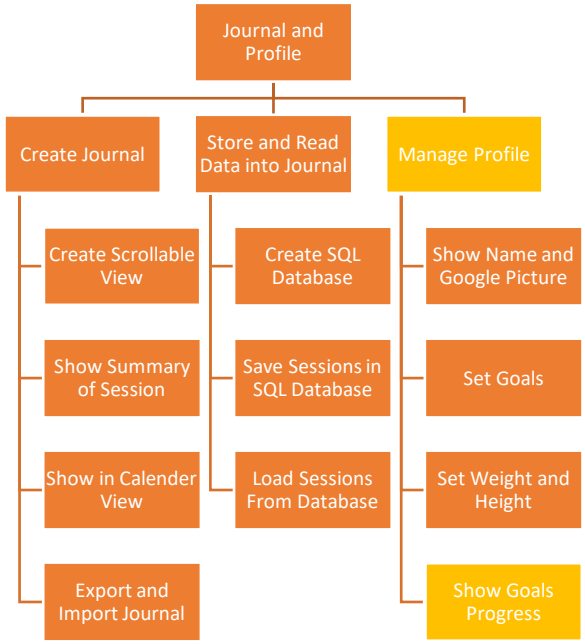
If the app does not have a saved weight then it calls the API just like before with no changes as you can see. It will get the weight from the cloud and update the app with this new value. I do the exact same with weight right after this if statement. I cannot make a new method to reduce code repetition here because they deal with different data and different parts of the FIT API.

Muhammad Faizan Alam



As you can see now the user is getting feedback quicker when they set the weight and height instead of having to wait for updates to their Google Accounts being verified. The value from the app's own data is now retrieved instead of calling the API every time thus increasing performance of the app as well. This now works as intended!

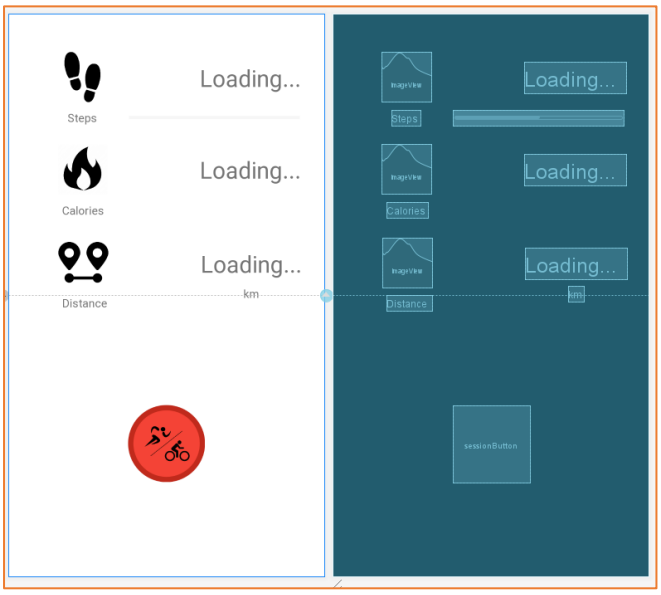
Show Goal Progress

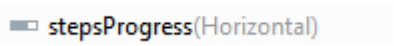


Finally, the last bit will not focus on the profile view but the main view. This means I have to retrieve the goals in the main view fragment and set on the progress bars so the user can see their progress for their set goals.

Steps Progress Bar

Next, I want to setup a progress bar to show the user a visual representation of how far they are from their daily goal for both steps and calories.





First in the fragment_mail.xml file I add a progress bar underneath the steps counter. I set it up so it is aligned to the "Steps" text. I set the ID to stepsProgress for now and will modify the look of it later.

```
private ProgressBar stepsBar; // used to show progress to goal
// store the goals set in the UserProfile
private int stepsGoal = 0;
private int caloriesGoal = 0;
```

```
stepsBar = (ProgressBar) view.findViewById(R.id.stepsProgress);
```

As usual I set created a private variable to hold the ProgressBar so I can update it as the app runs. In the onCreateView I get the UI element and assign it.

I also created two other int variables which are private because I will only need them in this class. These two variables will be used to get the saved values stored in the UserProfile by the user.

```
private void getGoals() {
    // get data from sharedPreference
    sharedPrefs = getActivity().getSharedPreferences( name: "com.faizan.myfitness", Context.MODE_PRIVATE);
    // retrieve data for steps and calories if stored or 0
    stepsGoal = sharedPrefs.getInt(STEPS_TAG,  defValue: 0);
    caloriesGoal = sharedPrefs.getInt(CALORIES_TAG,  defValue: 0);
}
```

```
getGoals(); // get goals set by the user
```
-- called from restoreValues()

The getGoals method is called in the restoreValues method which is called whenever the view is refreshed. This will allow the most recent goal to be retrieved whenever the user changes it and comes back to the main fragment. I already have a sharedPrefs initialised so I use it to get the goals stored in it by the UserProfile class.

```
java.lang.ClassCastException: java.lang.Long cannot be cast to java.lang.Integer
    at android.app.SharedPreferencesImpl.getInt(SharedPreferencesImpl.java:307)
    at com.faizan.onefitness.MainFragment.getGoals(MainFragment.java:164)
    at com.faizan.onefitness.MainFragment.onCreateView(MainFragment.java:83)
```

However, running this program causes the following error. This is because the value stored in the SharedPreferences a long and not an integer which I store in the UserProfile. I realise this is because I am using the local class's tags

```
public static final String STEPS_GOAL_TAG = "STEPS_GOAL";
public static final String CALORIES_GOAL_TAG = "CALORIES_GOAL";
```


```
stepsGoal = sharedPrefs.getInt(UserProfile.STEPS_GOAL_TAG, defValue: 0);
caloriesGoal = sharedPrefs.getInt(UserProfile.CALORIES_GOAL_TAG, defValue: 0);
```

I make the tags public in the UserProfile class and change them in the getGoals method so they are saved to these specific tags in the app's data and not the daily tags I used previously. Now the app runs without crashing, next I need to handle updating the ProgressBar.

```
private void updateStepsGoal(final long totalSteps) {
    // make sure we don't divide by 0
    if (totalSteps != 0 || stepsGoal != 0) {
        // get progress from 0 to 100
        int progress = (int) (((double) totalSteps) / ((double) stepsGoal) * 100);

        if (progress < 100) {
            stepsBar.setProgress(progress); // set the progress value to a relevant value
        } else {
            stepsBar.setProgress(100); // its max so set it to 100
        }
    } else {
        // no steps travelled or goal set
        stepsBar.setProgress(0);
    }
}
```

Algorithm

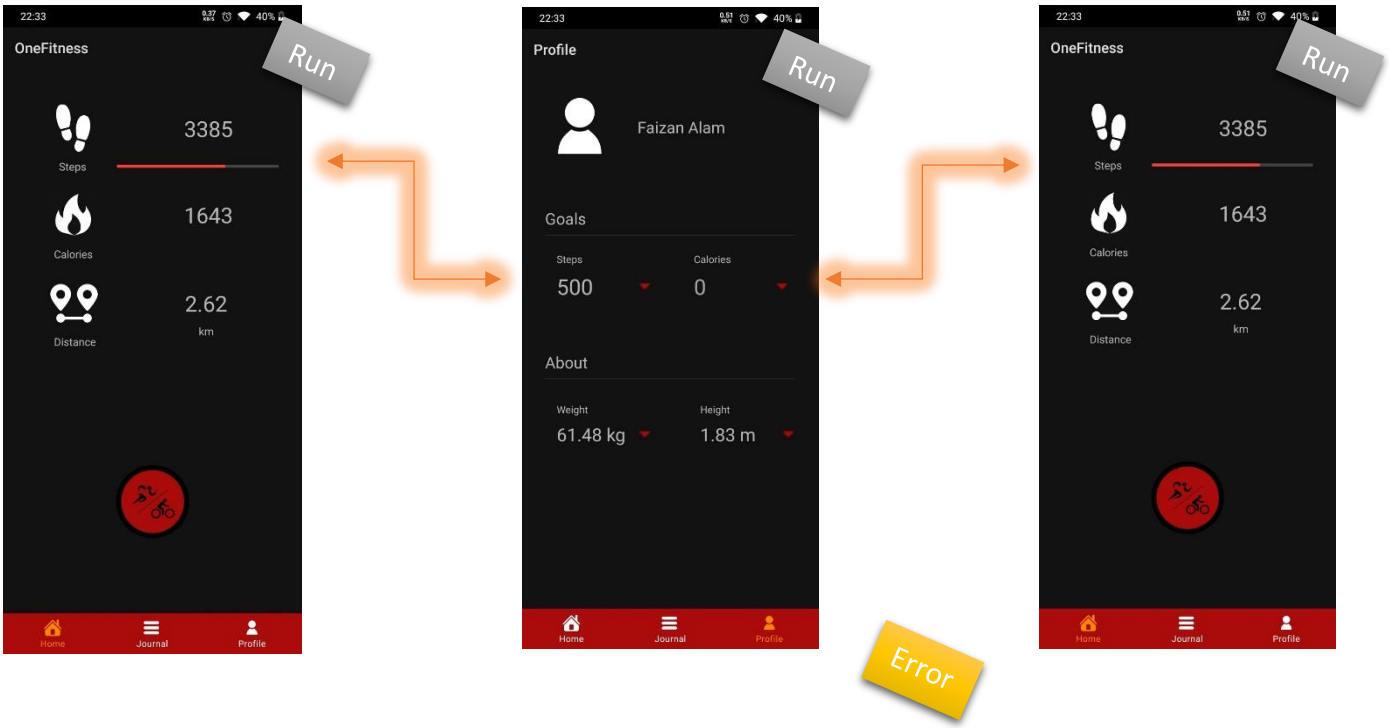
```
updateStepsGoal(totalSteps);
```
- called from sendSteps() method

Justification

The updateStepsGoal method will be called whenever the steps are updated or the goal is updated by the user. This method first checks that none of the values are 0 because I do not want to be dividing by 0. If the steps are 0 or the goal is 0 then the progress value is set to 0 so no progress is shown.

The progress is calculated by first casting the values to a double so I can get a percentage and then multiply it by 100 before casting it back into an integer. This allows the value 0.43 to be set to 43. The progressBar.setProgress() method requires a value from 0 to 100 so I need to do this.

Next I check the value is less than 100 so I can set the progress. This means the value is between 0 and 100. If it's 100 or above, then I simply set it to 100 as the progress cannot exceed the max value.



The progress bar is set to the correct value when the goal is 5000 on launch however when changing it back to 500 and then going back to the Home screen it does not update like I had programmed.

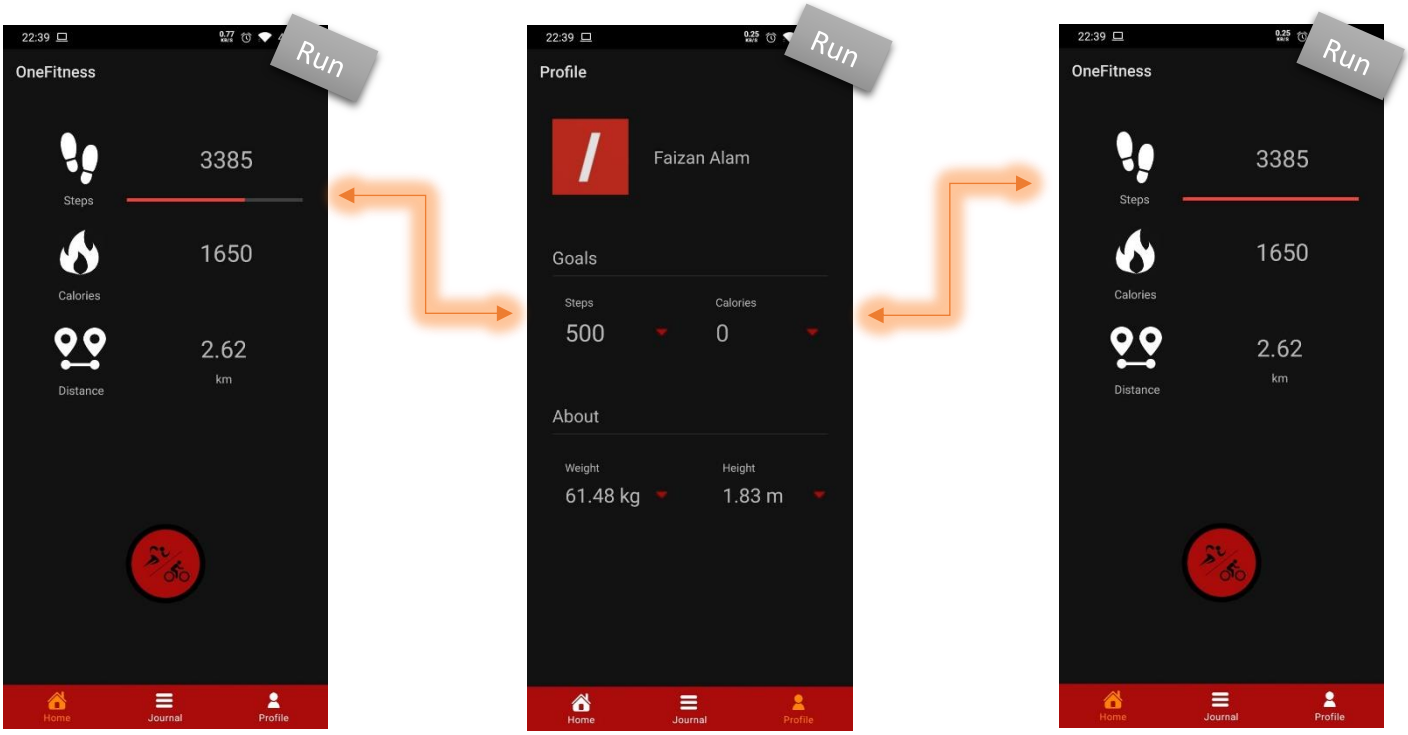
With some research I found out that I need to update the progress bar in a separate thread because of the way it is implemented. It does not hold up the UI thread, so it needs to run in a separate thread.

Fix

```
private void updateStepsGoal(final long totalSteps) {
    // run this in a background thread so the UI is updated when it runs
    // post() gets a handler to run a new thread for the progressBar
    stepsBar.post(new Runnable() {
        @Override
        public void run() {
            // make sure we don't divide by 0
            if (totalSteps != 0 || stepsGoal != 0) {
                // get progress from 0 to 100
                int progress = (int) (((double) totalSteps) / ((double) stepsGoal) * 100);

                if (progress < 100) {
                    stepsBar.setProgress(progress);
                } else {
                    stepsBar.setProgress(100);
                }
            } else {
                stepsBar.setProgress(0);
            }
        }
    });
}
```

I updated the method to retrieve a new Handler for a separate thread from the progress bar itself. The run method is called to update the UI of the progress bar when it is ready.

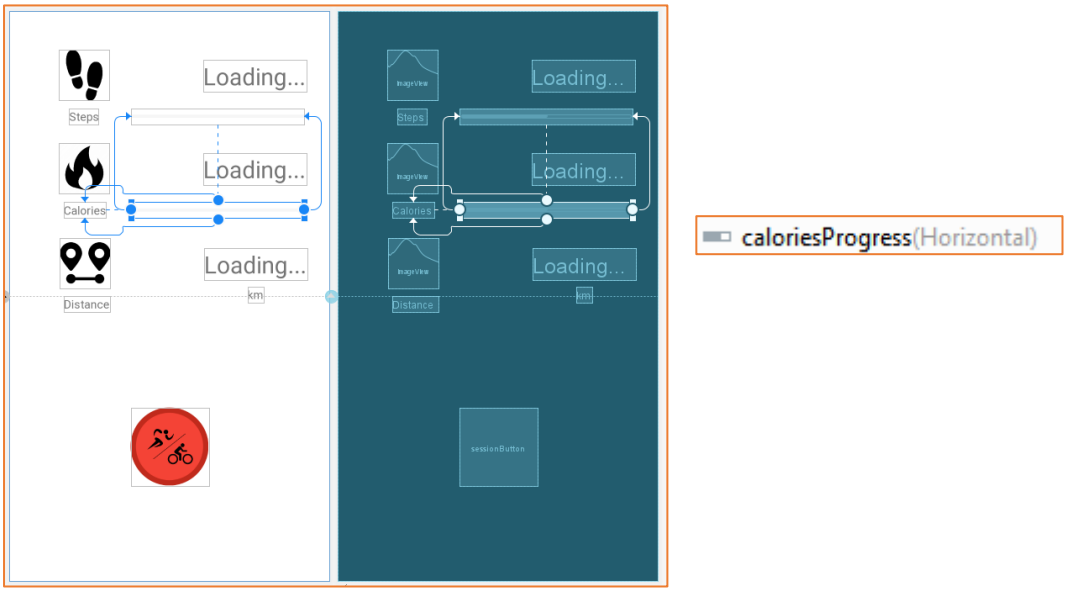
Muhammad Faizan Alam



Successful!

I ran the app again, changed the step's goal from 5000 to 500 and looked at the progress bar again. This time it was changed!

Calories Progress Bar



Next I added another ProgressBar widget for the calories goal. The width is set to that of the stepsProgress and the height is set to that of the Calories text as shown above.

```
private ProgressBar caloriesBar; // used to show progress of goal
```

```
caloriesBar = (ProgressBar) view.findViewById(R.id.caloriesProgress);
```

Next I initialise a private variable to access the ProgressBar.

```
private void updateGoalProgress(final ProgressBar bar, final double total, final double goal) {
    // run this in a background thread so the UI is updated when it runs
    // post() gets a handler to run a new thread for the progressBar
    bar.post(new Runnable() {
        @Override
        public void run() {
            // make sure we don't divide by 0
            if (total != 0 || goal != 0) {
                // get progress from 0 to 100
                int progress = (int) (((double) total) / ((double) goal) * 100);

                if (progress < 100) { // if progress is in range
                    bar.setProgress(progress);
                } else { // else set to max
                    bar.setProgress(100);
                }
            } else { // set progress to 0 if any values are 0
                bar.setProgress(0);
            }
        }
    });
}
```

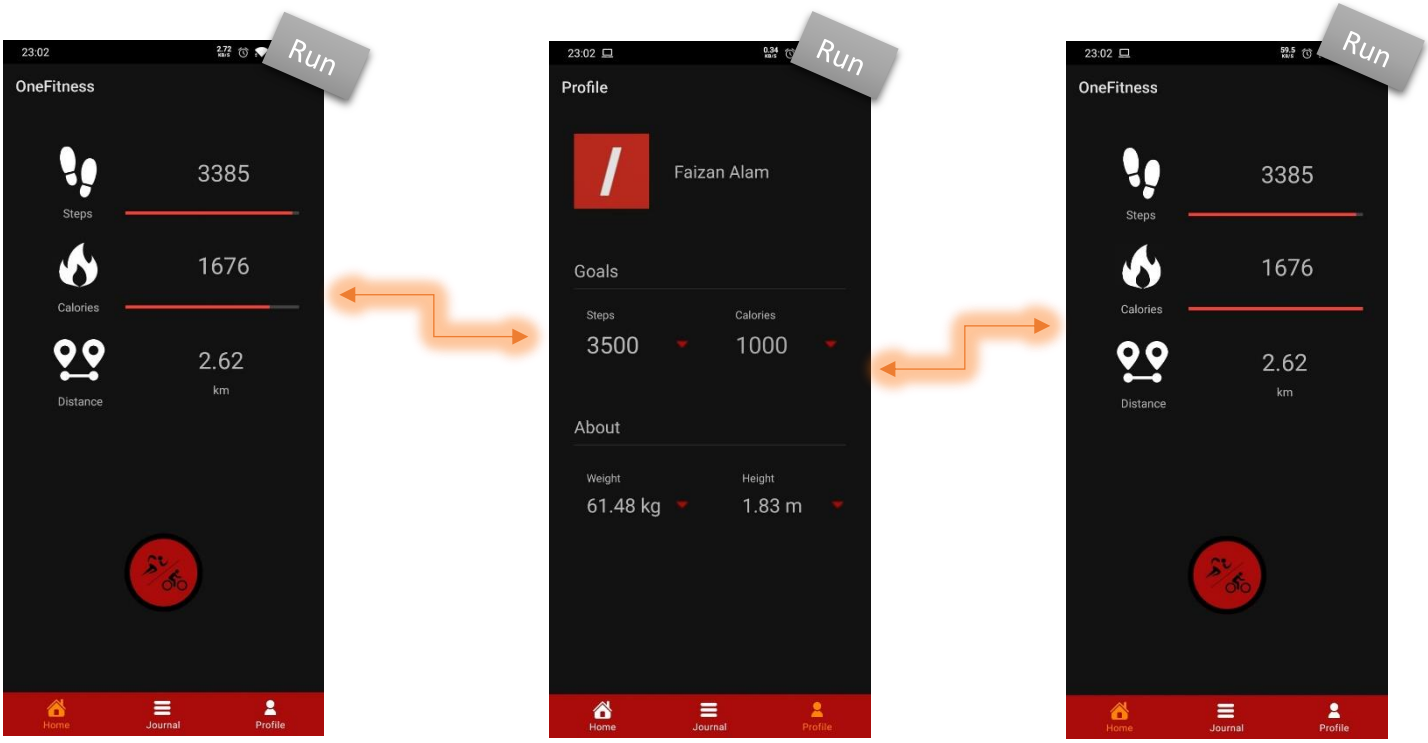
To take advantage of modular code I am going to copy the updateStepsGoal into a new method and change the variables to use those of the parameters. This way I can pass arguments from updateStepsGoal and updateCaloriesGoal to reduce repetition of code.

```
private void updateStepsGoal(final long totalSteps) {
    // update the step's progress
    updateGoalProgress(stepsBar, totalSteps, stepsGoal);
}


private void updateCaloriesGoal(final int totalCalories) {
    // update the calorie's progress
    updateGoalProgress(caloriesBar, totalCalories, caloriesGoal);
}
```

Now that I have the above code I can simply call the method with different arguments to update the relevant progress bar with the relevant data as shown above.

```
updateCaloriesGoal(totalCalories); // update progress bar
```

The method is then called from sendCalories and restoreValues methods, so it is updated on launch, resume and when new data is received from the API.



Running the app loads the correct value for the progress bar when the goal is set to 2000. I then changed it to 1000 and it updated when going back to the home screen!

Customise Progress Bar

I want to change the look of the progress bar to look more appealing and clearer to the user.

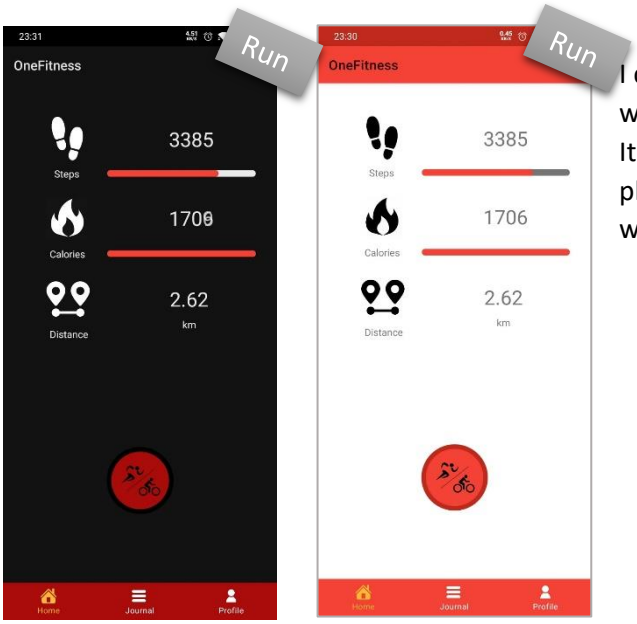
custom_progressbar.xml

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background"
        android:gravity="center_vertical|fill_horizontal">
        <shape android:shape="rectangle"
            android:tint="?attr/colorControlNormal">
            <corners android:radius="8dp"/>
            <size android:height="10dp" />
            <solid android:color="#90caf9" />
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress"
        android:gravity="center_vertical|fill_horizontal">
        <scale android:scaleWidth="100%">
            <shape android:shape="rectangle"
                android:tint="?attr/colorControlHighlight">
                <corners android:radius="8dp"/>
                <size android:height="10dp" />
                <solid android:color="#90caf9" />
            </shape>
        </scale>
    </item>
    <item android:id="@android:id/progress"
        android:gravity="center_vertical|fill_horizontal">
        <scale android:scaleWidth="100%">
            <shape android:shape="rectangle"
                android:tint="?attr/colorControlActivated">
                <corners android:radius="8dp"/>
                <size android:height="10dp" />
                <solid android:color="#b9f6ca" />
            </shape>
        </scale>
    </item>
</layer-list>
```

I created a new drawable xml file to set the look of the progress bars. I created rectangles with a gay background with rounded ends.

```
android:progressDrawable="@drawable/custom_progressbar"
```

I set the drawable on the progress bars.



I checked the new progress bars which are now easier to view as well as see the progress left due to the highlighting background. It also looks better due to the rounded corners. Finally, I set my phone to light mode to test the app running with a light theme which still looks good.

Conclusion

I have completed the third and final module of the app using many features of OOP to verify data is correct as well as simplify future increments to the app. It also allowed me to manage the memory well and only have objects required active as memory is limited on mobile. I reduced a lot of repeated code by using polymorphism alongside encapsulation to prevent issues and allow easy debugging. I added an SQL database that stored all the sessions the user performs. It loads these when the app is launched. I created the journal to let the user see their session which can be expanded when clicked on. I showed the sessions sorted by date to allow the user easy management of them. Finally, I added the feature to export and import the database by the user using file handling and copying the bytes directly for the most efficient method. Finally, I created the profile page which I used a lot of validation to make sure the user's input was always valid. It showed their image and name as well as let them set goals. It also showed dialogs to set their weight and height. Finally, I let the user see their progress on the home screen towards their goals.

I copied the files using the bytes because it was the most efficient method and produced the least error. I also made sure to limit inputs and show dialogs where appropriate to the user, so the input is always valid and not out of range. I broke down each step and tested it through different runs and fixed any errors and crashes I ran into. I also got to use try clauses to make sure no files were corrupted and were always closed. I got to work with bytes directly here and learned that I had to be very careful whenever reading and writing files as they can easily be corrupted.

Journal Summary

Test Use	Actual Outcome
Navigation Menu	There is a menu at the bottom that lets the user change between home, journal and profile screens.
Sessions in journal	In the journal screen, the user's sessions are shown which is scrollable so they can view as many sessions as saved.
Session Summary	When the user clicks on a session, it expands that session and shows it in more detail including the map, distance, calories and speed/pace.

Database backup Summary

Test Use	Actual Outcome
Export Database	When the user pressed the export button, they are prompted to select a directory in their storage where the file is then exported to.
Import Database	The import button lets the user select a backup file which sessions are read from and into the app.

Profile Screen Summary

Test Use	Actual Outcome
User's name and picture	The user's name and picture are loaded on the profile page, so the user knows which google account is logged in
Set goals	The user is asked to enter numbers for their calories and steps goal which then updates the progress bar on the main screen.
Set weight and height	The user sets their weight and height by selecting the value from a dialog.

I tested that each feature worked throughout the development of version 3. I fixed any issues that I ran into and everything works as intended.

I have now completed this app thoroughly and it has ever feature that was designed. I will get the stakeholders feedback and add any features they may request.

Stakeholder's Feedback

Me:

Everything for version 3 has been implemented and works as designed. I have the journal view show a scrollable calendar with the user's sessions which can be expanded. These sessions can be exported to the user's storage and then imported allowing the app's data to be backed up and transferred between devices. I also have the profile view with the user's basic information as well as allowing them to set goals, weight and height. I use dialogs for this and add limits to validate all the data. Finally, there are progress bars which are updated in a separate thread, so they do not slow down the GUI to show the user's their progress for the day. The journal and profile were added to the navigation bar which is how most mobile apps allow multiple views that carry out separate functions.

I will interview and show the app to each stakeholder to review and give me any feedback they have with what they like and dislike.

Harrison:

The apps great and simple, I like it! More specifically I like the goals and how the progress is shown on the main screen. As I have said I will be using the app to view my progress multiple times, so I like it being on the main screen. I also like the simple view of the journal you have setup. One thing I would like is a sound when the countdown starts for a session. Thanks

Abytom:

Amazing app. I like the journal and how I can expand on my previous sessions. This session is very important for me as I want to improve how well I perform week by week. One thing I would like are some animations in between the screens when I change them. Most apps have nice animations making them appear professional and nicer.

Jamshed:

The apps simple and has all the icons which are simple to detect without reading the text underneath it which can be hard as the screen is small. I like the dialogs you have made for the weight and height, so it is easy to select without having to manually type it.

One improvement would be to add swiping to delete sessions like deleting emails.

Improve:

Firstly, I want to implement the sound for the countdown when a session is being started. This will let the user know about the progress without having to look at their phone so they can put their phone away. I want to then add the animations Abytom has suggested which will make the app more user friendly as well as appear more fluid. Finally, I will implement the swiping action as well as add some theme to it to fit well with my app. Overall everyone is happy with the app and it works great for them!

Muhammad Faizan Alam

Version 3 Improvements – Stakeholders

Play sound on start-up

SessionFragment.java

```
ToneGenerator beepTone; // to play sounds
```

```
// will be used to play sound for the count down
beepTone = new ToneGenerator(AudioManager.STREAM_MUSIC,  volume: 65);
```

I will be using a ToneGenerator to play simple sounds for the countdown. I'll set the volume at 65 so it's not too intrusive but still loud enough to hear outdoors.

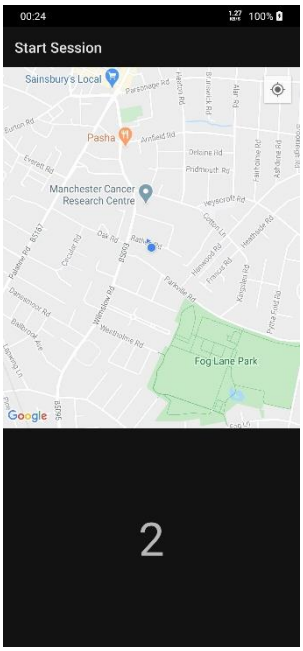
```
public void onTick(long millisUntilFinished) {
    beepTone.startTone(ToneGenerator.TONE_CDMA_PIP,  durationMs: 100); // play short beeps
```

I will play a very short beep every second the timer ticks by to represent the countdown being started. This is done in the onTick method which has other pieces of code in it as well.

```
public void onFinish() {
    beepTone.startTone(ToneGenerator.TONE_CDMA_ANSWER,  durationMs: 800); // play long sound
```

In the onFinish method I'll play a longer beep to let the user know that the session has started which is a different tone to differentiate it between the countdown sounds.

Successful! This satisfies what was requested by Harrison. Unfortunately, I cannot show this in the word document but it does work when the count down is started every second.



Animations for Fragments

As Abytom suggested, I added a few animations between the different fragments. This removes the feeling of the app feeling slow as the views are changed with a nice animation and allows it to also load data from the API as well.

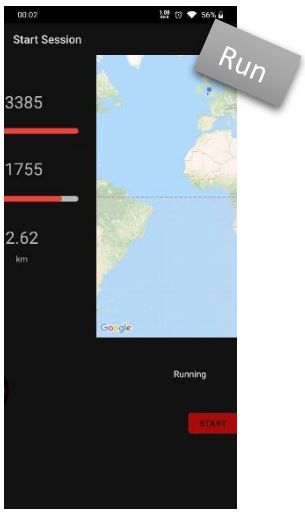
The animations are pre-made by google; they basically modify the screen which the view is shown on.

The right to left and left to right animations will be used for the session and session summary. The expand in and shrink in will be used to hide and show different fragments in the navigation menu.

```
.setCustomAnimations(R.anim.enter_right_to_left, R.anim.exit_right_to_left, R.anim.enter_left_to_right, R.anim.exit_left_to_right)
```

```
.setCustomAnimations(R.anim.expand_in, R.anim.shrink_in, R.anim.expand_in, R.anim.shrink_in)
```

I added the animation from the xml files to the FragmentTransactions. They will be called when the fragment is started and when it is ended. That is why there is two animations which are essentially opposite of each other.



Successful! Animation shows one fragment disappearing(shrinks) while another one appears(grows).

This animation shows a session being started, the main fragment is pushed to the left and the new fragment is pulled in from the right.

These animations now work exactly as designed. The views above are **not glitched** but show the animations in the middle working.

Delete Sessions

Optimise Code

```
private void initRecyclerView() {
    // Use simple linear layout
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    // Create a new adapter to hold all the items
    adapter = new RecyclerSessionAdapter(getContext(), onSessionListener: this);
    // set the adapter on the UI element
    recyclerView.setAdapter(adapter);
    // add divider lines
    DividerItemDecoration dividerItemDecoration = new DividerItemDecoration(recyclerView.getContext(),
            getResources().getConfiguration().orientation);
    recyclerView.addItemDecoration(dividerItemDecoration);
}
```

One of the things I changed was to initialise the recyclerView code in the onCreateView and updateView() were similar so I put the code in a new method to reduce code repetition and allow easier debugging. This method is now called from those two separate methods when required and completely sets up the recyclerView again.

Setup swipe action

```
// we will get a callback when an item is swipred left or right
ItemTouchHelper.SimpleCallback itemTouchHelperCallback = new ItemTouchHelper.SimpleCallback( dragDirs: 0,
        swipeDirs: ItemTouchHelper.RIGHT | ItemTouchHelper.LEFT) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false; // implement this later
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
        int swipedPos = viewHolder.getAdapterPosition(); // get the item swiped
        mDb.sessionDataDao().deleteSessionData(sessions.get(swipedPos)); // delete the item from the database
        sessions.remove(swipedPos); // remove the item from the list of sessions in the view
        adapter.notifyDataSetChanged(); // update the recyclerView adapter so the item is removed in the GUI
    }
};
```

To setup the swipe action on the items in the RecyclerView I attached an ItemTouchHelper that will detect the actions for me and allow me to run my code when the user carries out the swipe. Firstly I allow both left and right swipes to be detected. The onMove I will keep empty for now as I don't need to execute anything in that. In the onSwiped method I first get the position of the item that was swiped. I then get the relevant SessionData item and pass it to the deleteSessionData so it can be removed from the database. Then I remove it from the list that is displayed in the RecyclerView before notifying the adapter to update the view, so the user is shown the new items.

```
// set left and right swipe action
new ItemTouchHelper(itemTouchHelperCallback).attachToRecyclerView(recyclerView);
```

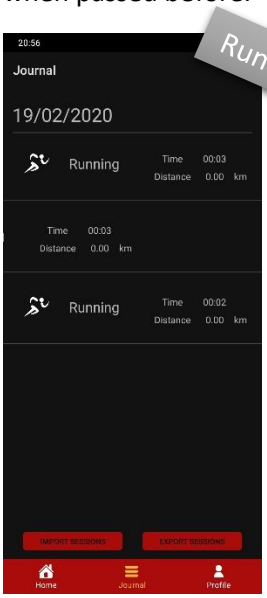
I then set the action to execute on the RecyclerView in my class.

```
java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time.
```

Running it caused a crash however. From what I can see from the crash stack the main issue seems to be that I was trying to access the database from the UI thread. Because this is being executed on the JournalFragment class I need to run it in a separate thread so the app doesn't hang.

```
@Override
public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
    final int swipedPos = viewHolder.getAdapterPosition(); // get the item swiped
    Log.d(TAG,  msg: "swiped item: " + swipedPos); // debug purpose

    AppExecutors.getInstance().diskIO().execute(new Runnable() {
        @Override
        public void run() {
            mDb.sessionDataDao().deleteSessionData(sessions.get(swipedPos)); // delete the item from the database
        }
    });
    sessions.remove(swipedPos);
    adapter.notifyItemRemoved(swipedPos); // update the recyclerView adapter so the item is removed in the GUI
}
```

Like before I get an AppExecutors helper class to run the code in a separate thread. I had to make swipedPos static to be able to pass it to the AppExecutor class else it would give an error at compile time as the value is not guaranteed when passed before.



Crash

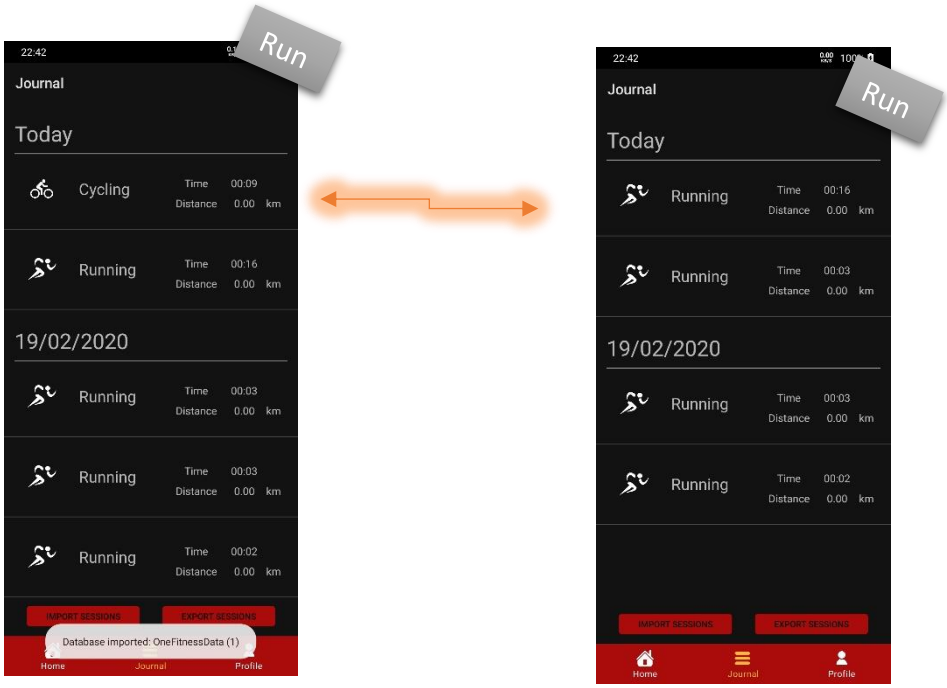
```
E/AndroidRuntime: FATAL EXCEPTION: pool-1-thread-1
    Process: com.faizan.onefitness, PID: 19191
    java.lang.IndexOutOfBoundsException: Index: 2, Size: 2
        at java.util.ArrayList.get(ArrayList.java:437)
        at com.faizan.onefitness.journal.JournalFragment$4$1.run(JournalFragment.java:209)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1167)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:641)
        at java.lang.Thread.run(Thread.java:919)
```

I ran the app and swiped away two items which caused a crash shown above. This is because the List of items continues to have the same number of items even though I remove one of the items. The index still stays the same which causes the adapter to believe there is an extra item. This then causes an out-of-bounds crash as I try to access an index that does not exist.

Fix

```
retrieveTasks(); // update the adapter with the new list
adapter.notifyItemRemoved(swipedPos); // update the recyc
```

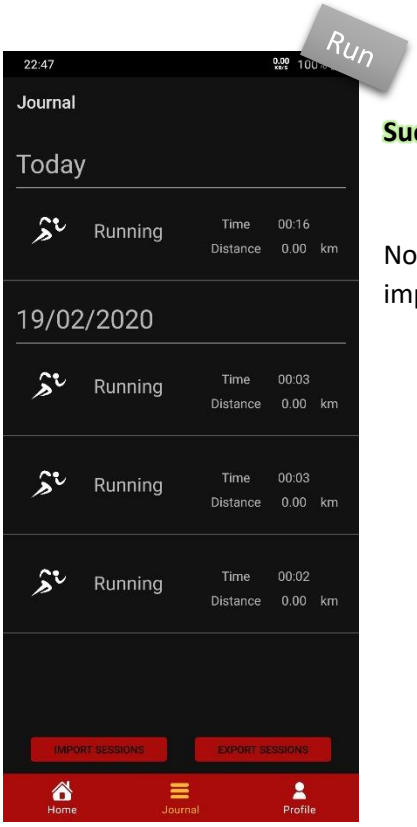
I simply called the retrieveTasks method I previously made to update the list in the adapter, so it knows that there is no longer an item at that position and reduces the index total by 1 when an item is removed.



Now when run, and I remove the top cycling session, the running from the previous date is inserted.

```
initRecyclerView(); // force redraw
//adapter.notifyItemRemoved(swipedPos);
```

To fix this I reset the whole view by calling the initRecyclerView I made above which will reinitialise the adapter to update the view with the relevant icons and data in the correct order.



Successful!

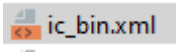
Now when I remove the cycling item it updates with the correct format. I have now implemented deleting sessions now as well.

Muhammad Faizan Alam

I want to add a red background with a bin icon similar to how other mobile apps work when an item is being deleted to let the user know what action they are carrying out.

```
implementation 'it.xabaras.android:recyclerview-swipedecorator:1.2.1'
```

For this I implemented a premade library that will allow me to add colour to the background as well as an icon



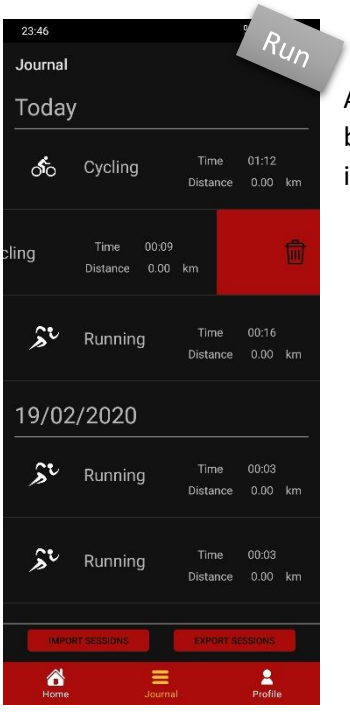
I also added a rubbish bin black icon to use with it.

```
// draw the background when item is swiped
@Override
public void onChildDraw (Canvas c, RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
                         float dX, float dY, int actionState, boolean isCurrentlyActive){

    // set the background to an orange red
    new RecyclerViewSwipeDecorator.Builder(getContext(), c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive)
            .addBackgroundColor(ContextCompat.getColor(getContext(), R.color.colorPrimary))
            .addActionIcon(R.drawable.ic_bin) // draw a bin icon
            .create()
            .decorate(); // add it to view
    // draw parent's items implmented by android
    super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive);
}
```

I used the example provided by that app to add the above code to the ItemTouchHelper class I used in my Journal Fragment.

It sets the background colour to the primary colour of the app which is a dark red. It then draws a bin in the background as well to show the user what action they are about to carry out.

I also added a horizontal bar above the buttons to show a cleaner UI with the buttons being separated from the sessions.



As you can see now, the red background alongside the bin icon is shown when an item is being swiped. There is also a horizontal bar at the bottom to differentiate between the import and export buttons and the sessions, providing a cleaner UI.

Evaluation

Post Development Testing

All my tests are performed in the videos with both voice and captions added. The test's thoroughly check every little step in each test to be thorough and make sure there are no errors. In the table below I will summarise each video below each test.

Test No.	Description of Test	Test Data	Expected Result
		Main Screen	
1.	Log-in with google account	**Valid Data:** Google Account signed in with valid email and password **Invalid Data:** Email and password are not valid for a google account	**Valid Data:** Google Account signed in **Invalid Data:** Ask for log-in info again until valid data and don't load any data to prevent crashing
For test #1 I first tried an invalid email which did not let me login and continue, then I tried a valid email but an invalid password which also did not let me login. I then logged in with a valid email and password which let me continue to the app which is the expected outcome.			
2.	Ask for Google Account Permissions	**Valid Data:** Allow or deny pressed **Invalid Data:** Try to continue without response	**Valid Data:** **Allow**: Continue till next permission **Deny:** Ask for permissions again on next launch **Invalid Data:** Show message to grant permissions
For test #2 I first dismissed the dialog without a response which showed a message that let me know that I had to grant the permissions by restarting the app to continue. Next I denied the permissions which showed the same message to restart the app. I finally restarted the app and allowed the permission which allowed the me to continue to the next step. This is the expected outcome for this test which shows it works thoroughly here.			
3.	Ask for device permissions	**Valid Data:** Allow or deny pressed **Invalid Data:** Try to continue without response	**Valid Data:** **Allow**: Continue till next permission **Deny:** show prompt to give access to permission manually, don't load any data that requires the permission **Invalid Data:** Don't continue, not possible as user must either grant or deny permissions
For test #3 I denied the permissions which caused a dialog to pop up to guide the user to grant the permissions manually. When I dismissed this dialog it did not load any data which otherwise would cause a crash. Next I granted the permissions which loaded the steps, calories and distance as expected!			
4.	Update steps, calories and distance	**Valid Data:** Steps, calories and distance are read **Borderline Data:** No updated data is read **Invalid Data:** No data is read at all	**Valid Data:** Update steps, calories and distance counters **Borderline Data:** Restore previously saved values **Invalid Data:** Restore previously saved values or show it's loading
For test #4, for borderline data, I removed permissions from the app to see if the saved data was loaded which it was. For invalid data, I removed access to permissions completely which caused the app to not get retrieve any data. Finally I granted all the permissions to make sure no problems were run into. This loaded the steps, calories and distance which is the expected outcome. This completes this test and it works as expected.			

New Activity Session			
5.	Start Session Button pressed		Session is started with no errors. A map, session, type selection and the start button are shown
For test #5, I pressed the activity button which opened a new view with the map, session type selection and start button. The main screen was hidden while this was shown. This is the expected outcome and the app works as expected for this step!			
6.	Session Type Selected	**Valid Data:** Selects running or cycling **Invalid Data:** Not possible as there are only 2 options – don't accept any other type of input	Changes the session type to be started to the one chosen by the user
For test #6, when selecting the session the user has the option to switch between running and cycling. I selected both options to test it changed as expected which it did. Invalid data could not be selected as the user is forced to select from the two types or else Running is chosen by default. This means test 6 works as expected!			
7.	Session is started		Starts a countdown which the session is started after
For test #7, I pressed the start button on the start session screen. This started the countdown from 3 which is the expected outcome.			
8.	Session data is updated as user runs/cycles	**Valid Data:** User moves around **Borderline Data:** User does not move **Invalid Data:** No invalid data possible	**Valid Data:** Map is updated with user's location. Distance, calories and pace/speed is updated as user moves **Borderline Data:** Everything is still updated but if the user doesn't move then everything except for the calories and timer will stay the same
For test #8, I first tested it by moving with a running session. All the details were updated as shown in the video. I then tested it with me not moving. The distance and speed were not updated at all. I then repeated these two tests for the cycling session which had the same exact outcome. Invalid data was not possible as the user can either start a session or not start a session. These tests show that test #8 shows the expected results.			
9.	Session is stopped		Stop the session tracking and record all data of the session
For test #9, After starting a session I clicked the stop button which stopped that session. This will record all the data in the database which can be viewed later which I will test later.			
10.	Session Summary		Summary is shown to the user with the relevant data such as time, distance, calories, pace/speed as well as their route
For test #10, I started a cycling session and cycled around the street. I then stopped the session to see the data in the summary. The summary showed the same data which was accurate for time, distance and speed as well as the route that was drawn. This is the expected outcome for this test!			

11.	Back button		Goes back to the previous session screen and then to the main screen
For test #11, I pressed the back button to go from the summary to the start session screen and then I pressed it again to go to the main screen. The back button works and loads a different view like planned!			
Navigation			
12.	Navigation Menu Buttons		Opens either Home, Journal or Profile screen when selected
For test #12, I now tested the navigation buttons by pressing them in different orders. It changed the views while still showing the navigation menu. This is the expected outcome to open Home, Journal or Profile screens when they are pressed.			
13.	Navigation Menu hidden for new session		The navigation menu is hidden when a new session is being started and shown when back to home
For test #13, I pressed the start activity button to test that the navigation menu was hidden when it was pressed. The video shows it was hidden. I then went back to the main menu and it showed the navigation menu again. This is the expected outcome.			
Session's Journal			
14.	Journal Screen		When opened it should show the user's previous sessions sorted by date with most recent first
For test #14, I clicked the journal button to open the journal in the navigation menu. I then checked to see if the sessions were ordered by the most recent at the top which they were. The dates were shown for each day that sessions were recorded for. This screen was scrollable like planned with the sessions being shown in a list. This is the expected outcome for the journal screen.			
15.	Expanded Session		Session can be expanded to view more details such as the route taken, calories and speed/pace
For test #15, I clicked on a session in the journal and it showed an expanded view of it with all the details such distance and calories as well as the track. This is just like the summary screen. This is the expected outcome for expanding a session from the journal.			
16.	Export Session	**Valid Data:** User selects a directory **Borderline Data:** User does not select a directory	**Valid Data:** Sessions are exported to a new file in that directory **Borderline Data:** Nothing is exported, and the program continues to work as normal
For test #16, I first exported the current sessions by granting permission to the downloads folder. This worked and the sessions were exported. I then tried not selecting a directory which caused nothing to happen as expected. The user either selects or doesn't select a directory so all the tests are done here. The expected outcome is shown in the video.			

17.	Import Session	**Valid Data:** User selects valid file **Borderline Data:** User does not select a file **Invalid Data:** Selects an invalid file	**Valid Data:** The sessions are overwritten with the sessions from the imported file **Borderline Data:** Nothing is imported, and the same data is used for sessions **Invalid Data:** User should not be allowed to select an invalid program as it will crash the program so selection of other files should be disabled
For test #17, I first deleted all the sessions so I can import them again to show it works. I first imported a valid file which showed the journal fill up with those sessions. Next I tried not selecting a file which meant nothing happened which is expected as well. Finally I tried selecting another file that wasn't valid however like I have implemented, it did not allow me to select those files to import meaning only valid data is ever imported. This is the expected outcome for this test.			
Profile Page			
18.	User's image is loaded	**Valid Data:** Image is retrieved **Invalid Data:** No image is retrieved	**Valid Data:** Image in profile view is updated with user's image **Invalid Data:** Image is set to default profile picture – no error should occur
For test #18, I first logged in with an account with no picture. In the video I went to the profile screen and as shown no picture was loaded and the default profile picture is shown. This is the expected outcome for invalid data. Next I signed in with an account with a picture, which was then shown in the profile screen like expected.			
19.	Set steps and calories goal	**Valid Data:** Reasonable number is typed in **Borderline Data:** Large number is typed in **Invalid Data:** Characters other than numbers are typed in	**Valid Data:** Step goal is set to the number input by the user **Borderline Data:** Goal is limited to a number of digits to keep the value realistic **Invalid Data:** Any character that's not a number is not accepted in input
For test #19, I set the steps and calories to a reasonable number such as 2500. This was accepted and updated. I then tried borderline data with a large number. This was accepted as well but I had a limit on it to make sure only a realistic number is entered. Finally I tried putting in invalid data such as other characters. However none of these inputs was accepted. I limit the keyboard to numbers so no other data is input into this other than numbers. Therefore completing this test.			
20.	Progress bars are updated	**Valid Data:** User's steps/calories change, or user changes their goal **Borderline Data:** Goal is 0, reached or passed	**Valid Data:** Progress bar is changed to reflect new progress **Borderline Data:** If 0 then no progress is shown. If goal is reached or passed, then the progress bar is filled 100%
For test #20, I set the steps and calories goal to 2500 and then I checked the home screen which showed the progress bars updated with the new values. This is the expected outcome for valid data. For borderline data I tested 0 which meant the progress value was filled fully. I then set a goal that was lower than the steps and calories which also caused the bar to be filled. Finally I set the values to exactly the steps and calories values which also had them filled in fully. All of this is the expected outcome and it works as planned.			

21.	Set Weight and Height	**Valid Data**: Reasonable number is typed in **Borderline Data**: Number too large or small is input **Invalid Data**: Characters other than numbers are typed in	**Valid Data**: Set's the weight to the new value **Borderline Data**: Numbers are limited so they continue to be realistic **Invalid Data**: The user is given a select and is not allowed to enter letters so the input will always be valid
For test #21, I set the weight and height with valid data which updated the values that were selected. I then tested out borderline data which was large and small numbers for weight and height. This also worked as I have a limit on either end to make sure the data has a range. I cannot input invalid data as the user has to select their weight a height instead of typing it out with their keyboard. Setting a range also means these values are accurate. This is all now the expected outcome as planned.			
22.	Steps, calories, weight and height are restored on restart		Make sure any data set by the user is saved and restored even after the app is restarted
This test shows that the steps, calories goals set by the user are restored even when the app has been stopped and restarted. This is the same with the weight and height. This is the expected outcome to make sure the user's data is always restored as the app will be closed multiple times throughout the day.			

I performed every test in my videos and walked through it by considering every step possible. All of the tests worked as planned with no errors or crashes whatsoever. The videos have been labelled using the test number and have both subtitles and voice over to walk through the tests. The test results are then summarised above.

This concludes the development testing as being complete and successful.

TEvaluation of Success Criteria

I will be referring back to the success criteria I wrote in the analysis to see if I have met every criterion. If I have not met any criteria I will explain why. I will copy the table and show if it's been met and how below.

I will use the key to the right in my table to make it easier to read. I have used a tick if the criteria were met, and an X if it was not met which are logical. For the partially met I have used a square to represent it as there is no logical symbol to represent that normally. This key table will be used to illustrate how I have met the success criteria.

✔	Criteria was successfully met
☐	Criteria was partially met
✘	Criteria was not met

I will break down the success criteria and talk about each one separately using a column below it similar to the post development testing done above. The test's mentioned below were performed above and the related video has to be watched for the test to see the evidence and a walkthrough.

No.	Requirement	Justification	✔ Met?
1.	Menu with icons on the bottom	Allow the user to easily navigate the app	✔
Navigation menu with icons representing each screen is shown as seen in test 12. This menu allows the user to go between the different views in whatever order they decide to and is robust. This allows the user to carry out tasks much quicker as they do not have to go through a sequence.			
2.	Main Screen – show main details	The main feature of the app is to show the user their steps and calories as well as distance, so the user can improve their fitness	✔
The home screen shows the steps, calories and distance to the user as seen in test 4. The permissions must be granted and the user must log into their google account as shown in test 1-3 above before their main details are updated.			
3.	Track steps/calories/distance when the app is active	Tracks the user's steps/calories/distance while they're using the app	✔
The user's steps, calories and distance are tracked when they are using the app and are updated when verified for next update as shown in test 4.			
4.	Track the steps/calories/distance when the app is inactive	The user's steps also need to be tracked when they're not using the app and their phone	✔
The user's steps, calories and distance are tracked even if the app is not launched. The app is updated when it is launched with the recorded values. The user logs into their google account which lets us to subscribe to the services and get updates when the app is launched for the time period when the app was not active.			
5.	Show user location before session is started	Let the user see what location they are going to start their session from	✔
The user is shown their location on a map when they go to the session start screen as shown in test 5 where the whole view is shown. This lets the user know where they currently are and their surroundings easily.			
6.	Track running and cycling sessions	Allow the user to track their route and speed/pace when they go cycling/running	✔
The user can select between running and cycling as shown in test 6. Once the user has selected a type of activity they can press the start button which will start a countdown as shown in test 7. After the countdown the session is started and tracked.			

7.	Countdown for sessions	Let the user get ready to run/cycle before the session starts recording	✔
Pressing the start button starts a countdown from 3 as shown in test 7.			
8.	Track speed, time and calories in sessions	Track the speed, time and calories as they run/cycle	✔
Once the session has been started the session data is updated as the user moves. The distance, speed, time and calories are updated frequently. When the user is not moving then the distance and speed remain at 0 as they are not moving. I have thoroughly tested this in test 8 which shows it works and is robust so I believe I have bet this success criteria as it's very accurate data.			
9.	Track route as the user travels	Let's the user see where they have travelled on the map as they perform their session	☐
I have partially met this success criteria. I track the user's location as they travel and move which is shown in the summary once the session is ended so this has been met. However I do not draw this route while the session is being carried out for multiple reasons. Firstly it would cause more demand on the phone while it's trying to update the location that already is very intensive as the phone will have to make many more API calls to draw the route. Secondly if the user was to travel the same path multiple times in the same session then it would be drawn very wrong and cover details that they might otherwise need such as the street's name. Other well-known fitness apps do not have this feature for this reason so throughout my development I have decided not to implement this and keep the app more fluid and easier to use. Test 10 shows that the route is still tracked as it is drawn when the session is ended however I do not draw it actively while the session is active.			
10.	Show a report after each session	Allow the user to see their performance after a session so they can improve on their goals	✔
When the user ends a session(test 9) they are shown a summary as shown in test 10. The data here is very accurate and is the same as tracked by the session. I have thoroughly tested this to make sure it worked for both running and cycling sessions. The route is drawn and the user is shown their speed, calories and distance in the summary.			
11.	Track the route	Track the route for the running or cycling sessions	✔
As shown in test 10 the route is tracked and drawn in the summary. This is done for both running and cycling sessions so it has been met.			
12.	Journal should have a vertical scrollable calendar	Easily see previous days that are scrollable like a webpage	✔
Test 14 shows the journal screen being scrolled. The sessions are ordered from the most recent at the top as shown in that test. This allows the user to easily see previous sessions and will save sessions forever. The journal view is scrollable to view older sessions as shown there. It is a vertical calendar with the sessions being underneath each date.			
13.	Show expandable cycling and running sessions in the journal	Allow separate tracking for the sessions so users can see their previous work outs and push themselves	✔
As shown in test 15, the sessions in the journal can be clicked on to view the session's more detailed summary. This shows the user the route they travelled for that session, they calories, distance and start and end time.			
14.	Export and import session data	Let the user make backups of their data and restore it – allowing them to transfer data between devices	✔
I first test the export feature in test 16. I first have a few sessions in the app recorded to export. I then export this to a file which is successful. I thoroughly test this feature as file handling can have a lot of errors. For all the tests the export method works as expected. I then deleted these sessions from the app and used the import method to load them back in again which also worked. I tested this thoroughly as well to make sure no invalid files could be imported as shown in that test.			

15.	Allow the user to easily change their height and weight	This will allow the user to update details about themselves	✔
As shown in test 21, the user can update their height and weight in the profile screen. I made the input a selectable value and added a range so all the values selected are valid and within range for a normal human. This lets them update their details easily.			
16.	Set goals	Allows the user to set themselves goals in range so they can see how much is left	✔
In test 19 I made sure that the goals that were set were valid by limiting the input to numbers only and with a certain range as well. These values are saved and reloaded whenever they launch the app again. This will let them push themselves daily			
17.	Show progress for goals	Allows the user to easily see their progress for the day	✔
In test 20 I tested all the values and the updates to the progress bar to thoroughly make sure nothing would break. The progress bars were always filled in the correct amount for different values such as 0. This lets the user either set goals or ignore them completely.			
18.	Log in with google	Allow the user to easily sign in with google to keep track of their steps, calories and distance	✔
As shown in test 1, the user is asked to log into their google account to continue to use the application. Without an account the user cannot continue using the app and will be asked on start-up until they sign in.			

I've now explained that my success criteria were met nearly 100% other than the change I decided on for criteria 9. The path is still tracked as it's drawn in the summary of the session however it is not drawn while the session is active to reduce load on the phone. This was also not an important feature either and was not requested by any of the stakeholders.

Success of Essential Usability

Some of my early usability features I wanted to implement were from my stakeholder in the analysis.

I'm going to use the same key to make check if the stakeholder's criteria for the general usability of the application were met. The key to the right as explained earlier shows a tick and cross for whether the criteria was met or not met respectively while the square is used if it was met partially.

✔	Criteria was successfully met
☐	Criteria was partially met
✘	Criteria was not met

The criteria below are focused mainly on the look as well as how the app is used by the end user. I based my initial design on the constraints below. I will talk about each criterion in the cell directly below it like before. The app can be seen in the testing videos provided with each test highlighted in the development testing table above.

No.	Requirement	Justification	✔ Met?
1.	Show daily steps, calories and distance on the main screen	So that the user can easily see their main stats	✔
These requirements were made before the success criteria in the analysis and therefore this requirement is very simply to success criteria number 2 shown in the table above. However I will talk about the presentation of the data here. The daily steps, calories and distance are shown on the main screen in large texts. This was one of my early requirements to make the app simply and easy to use. Using large texts allows the user to easily see their most important data. This large texts is shown in test 4.			
2.	Simple Intuitive Design	Keep the app simple to allow ease of use by using icons so they know the function of different buttons	✔
I have achieved the simply intuitive design by using large texts, labels and icons around my app. The labels explain the icons and as the user get's use to the app they do not have to read the labels as they get familiar with the icons. I've also separated the app's in sections with different views to make sure it is not cluttered and the user only has access to what they need. This provides the simple intuitive design while still carrying out many functions. This can bee seen throughout tests 4-22 with different parts of the app being split up in sections.			
3.	Navigation Menu	Allow the user to easily navigate the different views	✔
I've implemented the navigation menu which is used by most mobile apps today to provide a similar feel. This makes sure the user does not have to learn anything new as the design of the app stays consistent with the design of the system and other apps. The navigation menu allows them to access different screens without having all the information cluttered onto one screen. This can again be seen in test 12.			
4.	Track running and cycling sessions	Let the user start sessions to track info when they run and cycle	✔
This is similar to the success criteria above where I make sure the user can track their running and cycling activity by starting a session. Above I talked about the detail of what should be carried out and this has been implemented as seen in test 7.			
5.	Set daily goals	Let the user set their goals to reach daily	✔
One of the most important features was allowing the user to set goals for themselves. In test 19, I show the user can set goals for steps and calories that they can try to achieve daily.			
6.	Progress for daily goals	Progress for the daily goals should be shown on the main screen	✔
The progress is shown and updated on the main screen as test 20 shows. This allows the user to see their progress as they get closer to their goal.			

I've met all the essential usability features mainly for the UI of the app as shown above and therefore the design was completed exactly as expected with proof being shown in the tests.

Maintenance

As I've highlighted through the development of my app, my main focus to use object-oriented approach(**OOP**) was to allow easily future maintenance of the app as well as reduce the complexity of it. Firstly I made a tracker class which I then had three other classes **inherit** it to track the steps, calories and distance. This was because these classes had very similar methods to track data and update it on the user's screen. Using **abstraction** I can reduce the main methods these classes will have in common. This allows me to add future pieces of data to be tracked and updated without having to worry about how it is done in the background and when the app works. I used this to add calories and distance tracking very fast and efficiently as shown in the development. This same method can be used for any other data the user might be interested in.

I also used **inheritance** to simplify how a session is tracked. The running and cycling classes have a lot in common as they inherit a session class which handles all the main features such as starting and stopping the sessions and tracking the user's location. The cycling session then calculates the speed while the running session tracks the steps. This allows me to reduce repeated code which and therefore the complexity of the app is reduced so bugs and errors can be found more easily and fixed in one spot instead of many. This also allows me to simply add a new type of activity by simply inheriting the session class and recording any session specific data after.

I have also used OOP to handle the data transferred between the different classes. Instead of having to pass many items throughout the classes I can centralise the session data to a class and allow my program's classes which have access to read and modify it if required. This means the classes can work independently of one another instead of having to rely on data to be sent from another class first. This further allows new features to be added to the app which can be isolated from the other classes and therefore implement them much easier in the future.

I have used **sensible** variable, class and method **names** throughout my app as well as used OOP's **encapsulation** to allow me to change and modify them with ease. Using encapsulation means only the right methods and variables which are required are able to be accessed in that range. This means my classes can handle variables and use private functions without having other future classes worry about how they are used and implemented. Every section of my app is split up into multiple classes which themselves have multiple methods to allow the right functions to be found easily and modified easily as well. Using sensible names in the correct scope as well as splitting the code allows me to add new features to it more easily in the future as the code is in separate blocks.

Lastly I have documented my code with **comments** explaining what each bit of code does to programmers other than myself. This will allow them to figure out what part of the code does what and allow them to easily modify the correct code. Allowing others to maintain the code also means maintenance can be performed much quicker as it will not only rely on me. Commented code has and will continue to help me figure out what parts of previous code I have written do. This makes it easier for me to either increase the performance or add to it in the future much easier.

Limitations

I have implemented all except one feature that I had planned on. The feature I didn't implement was drawing the route as the user travelled due to the stress this would put on the APIs and the phone. I also did not implement any other types of sessions other than running and cycling but I have highlighted not being able to do that in my analysis and design due to the time constraints I have.

Draw route as session is active

I did not implement this feature due to the issues it would cause with performance on the phone. Phones are not nearly as powerful as computers and most android phones are even slower than my phone. For this reason I have decided not to implement this feature. Another reason I did not implement it was because it would draw over street names which I did not have a fix for due to my limited knowledge with the Google Map's API. Many other fitness apps did not draw the route either and so I decided throughout the development that I would exclude this feature as it is not important. The route would still be drawn in the session summary after the session was complete which I felt was satisfactory for the end-user. I believe I could have found many workarounds to this if I had more experience but many developers who make such apps work in large teams and have years and decades of experience as well as more time to implement it. I was also running short on time as I had to progress onto the next version of my app so I could implement everything else. Overall I believe I made the right decision due to all the constraints I had.

Delete animations

The animations for the delete swipe are not as smooth due to not being able to implement the RecyclerView in my app correctly due to lack of knowledge. This meant I have used a workaround I figured out myself to be able to figure out the solution to the swiping for delete. I believe if I had more experience and knowledge with Android development as well as much more time I could implement this more smoothly. The lack of knowledge was also one of my issues in the limitations in the analysis.

Overall I believe I've implemented everything and it works as it should. There are many things I can think about which I could add however I have implemented what I initially designed and due to time constraints I cannot add new features that I've thought of.

Improvement for further development.

My solution is quite successful and implements all the features based on my testing, success criteria and stakeholder feedback, I can still think of many features to add and improve the app with if I had more time and experience.

Start-up Guide Dialog

I had not planned on this but I believe it would be a welcome feature especially to newer users. If I had included a start-up dialog that guided the user throughout the app to show them the different features of the app. I had not thought of this in the beginning but as I saw the app bigger I believe it would have been a welcome feature. For now the start-up dialogs that are shown are for signing into their google account and granting permissions. I did not think of this in my design as I had walked through the app overall design with the stakeholders and they had no suggested it and as well as me not being able to think of it ahead. I believe implementing this feature would cost me quite a lot of time as it would be more complicated due to having to interact with the user. Due to the time constraints and lack of knowledge I have not implemented it and it was not in my initial design.

View progress in graphs

One other feature I can think of which would be helpful to the user would be to show the user their progress for steps and calories. The progress would be shown in a graph by week, month or year to allow the user to see their progress over time. Graphs could also be shown for the user's weight to see how they are performing over time. This feature would be great to implement if more time was available as well as the knowledge to implement graphs.

Track Food In-Take

This feature would be one of the most helpful but would be quite hard to implement due to the time constraints. This feature would allow the user to track their calories intake by logging what food they had. This could be used alongside their calories burned for that day to set a goal for whether they want to increase or decrease weight. This feature would be quite helpful but would require a lot of work and time to implement. I believe if I had more time I could implement this feature.

Google Backups

One of the features would be to backup the data from the app to the user's google account. This would allow the user to not worry about exporting and backing up their data and having to then import it again as all the data would be stored on their google account. The app would restore and save all data to the account and provide a seamless experience over different devices as well. If I had time I believe I could implement this as I have already implemented logging into the user's account.