COMPUTER SCIENCE AQA A-LEVEL NEA

EXERCISE DETECTION PROJECT

CONTENTS

[1 ANALYSIS 3](#_Toc88126921)

[1.1 Introduction 3](#_Toc88126922)

[1.2 Sensors Overview 4](#_Toc88126923)

[1.3 Exercises and Activities 10](#_Toc88126924)

[1.4 Current Market and Applications 11](#_Toc88126925)

[1.5 Third-Party Feedback 13](#_Toc88126926)

[1.6 Prototyping and Spiking 15](#_Toc88126927)

[1.7 Objectives and the Critical Path 19](#_Toc88126928)

[2 DESIGN 22](#_Toc88126929)

[2.1 High Level Overview 22](#_Toc88126930)

[2.2 Pedometer Algorithm 24](#_Toc88126931)

[2.3 GPS Tracking 30](#_Toc88126932)

[2.4 Computer Vision and ML Kit Library 31](#_Toc88126933)

[2.5 User Defined Classes 32](#_Toc88126934)

[2.6 Database Design 33](#_Toc88126935)

[2.7 User Interface Design 40](#_Toc88126936)

[3 TECHNICAL SOLUTION 42](#_Toc88126937)

[3.1 Issues and Logs: 42](#_Toc88126938)

[3.2 XML Files and Design 46](#_Toc88126939)

[3.3 Fragments and Activities 64](#_Toc88126940)

[3.4 Custom Java Classes 87](#_Toc88126941)

[3.5 Miscellaneous Files 91](#_Toc88126942)

[4 TESTING 95](#_Toc88126943)

[5 EVALUATION 96](#_Toc88126944)

[6 APPENDIX 97](#_Toc88126945)

[7 REFERENCES 102](#_Toc88126946)

# ANALYSIS

## Introduction

In an age of ever-increasing reliance on technology, with more and more people **adopting a sedentary lifestyle**, health and fitness have become an aspect that in many cases have become neglected. However, there is now an increasing pressure on governments and on society to change old habits of an unhealthy lifestyle, to a more active one.

Countless studies suggest that exercising not only improves physical health but also has a significant impact on improving mental health and relieving stress. One study suggests that those who exercised had **43.2% fewer days of poor mental health** in a month than those who did not. [1]. This shows to us the importance of exercise, especially as one of the leading causes of death in the US and as well as for men in the UK [2] is heart disease, which can be preventable through exercise and a healthy diet.

Despite the majority of the public know the great benefits of exercising, **only 63.3% of people aged over 16 consider themselves physically active** doing 150 minutes or more of moderately intensive activity in a week, according to a UK government survey. [3] There remains a large portion of the public who do not exercise, for many different and respective reasons. Various reasons may include not having enough time during the day to exercise, finding a gym that is affordable or simply not having enough motivation to work out.

Whilst a large chunk of the population remains unactive, the proportion of the population, especially among young adults, who have access to mobile smartphones continues to increase. One study suggests, that for those aged 16-24 years old, roughly 99% of respondents say they have a smartphone, for those living in the UK [4]. This is a rather stark contrast to society just 10-20 years ago, where smartphones had barely begun to break into the common consumer market. This shows to us the **commonality of the smartphone** and how much of the UK’s population use smartphones daily, especially younger people.

My objective in this NEA is to investigate **how physical activities can be tracked** and detected through a device’s onboard sensors. Furthermore, at the end of this project, I aim to have a working app that gives users the interface to **access exercise recognition software**, to help more young people become active through their mobile phones.

## Sensors Overview

Between devices, the physical hardware and sensors on a given device vary, depending on various factors. Over the past decade, the general progression of these sensors tends to improve over time as computational power increases with more and more devices receiving more and more sensors.

Before motion sensors were widely adopted in mobile phones, they were often used for devices such as Wii remotes, airbag deployment, aircraft, missiles, etc. During the period between about 2005-2012, the adoption of onboard sensors started to break through to mobile phones, most notably within Apple and Samsung phones. The very first phone with 3-dimensional movement recognition was the Samsung SCH-310 [5], with Apple’s first iPhone also using accelerometer technology [6]. Nowadays, it is expected of manufactures to have various sensors, including the **gyroscope, accelerometer, and magnetometer.**

As I am focusing my project on Android applications, I will mainly be discussing the sensors on a typical Android phone, rather than one on an IOS phone.

Accelerometer Sensor:

As suggested by the title an accelerometer is a part of the phone which measures the **acceleration on a device on a three-axis** reflecting real-world movement. These axes are in the x, Y, and Z direction.

The data provided by an event from an Android device is given in meters per second squared (unit for acceleration). This provides a force along the X, Y, and Z vectors. [7]

The accelerometer is often used to detect motion within a given axis, especially to measure the translation of a device.

I am in this investigation, to research more on how I can use the accelerometer to track the distance of a given motion exercise, like walking or treadmilling.

Sparkfun Triple Axis Accelerometer https://www.sparkfun.com/products/13926

Gyroscope Sensor:



GK10A MEMS die (oscillating plate) https://www.ifixit.com/Teardown/iPhone+4+Gyroscope+Teardown/3156

A Mechanical Gyroscope https://guide-images.cdn.ifixit.com/igi/nmNv4u3uHqZ5VNIR.large

A gyroscope measures the **rate of rotation around the three axes**, often known as yaw, pitch, and roll. [7] Early versions of a gyroscope included three spherical axes that span around a rotor, being able to rotate freely in three axes. The gyroscope nowadays in phones is much more compact and consists of a tiny vibrating plate in a chip that is pushed around and is detected by the device processor. [8] can be used for a variety of different applications for a mobile phone. The most common is determining the orientation of a phone, whether it is being held in landscape or portrait mode, which was first well utilized in Apple’s iPhone 4. [9]

To my investigation, I am more interested in how a gyroscope can be used to determine an exercise being done by a user. For example, when a user puts a mobile phone in their pocket and goes for a run, how can I use the data from the gyroscope sensor to determine whether the user is running, based on the rate at which the device rotates in a repeated pattern of harmonic motion.

Magnetometer Sensor:



First Magnetometer https://nationalmaglab.org/education/magnet-academy/history-of-electricity-magnetism/museum/magnetometer

The magnetometer is responsible for measuring the **strength and direction of magnetic fields**, often used to figure out the spatial position of a device in a given space. They utilize the Earth’s magnetic orientation to calibrate a given device to a specific position. [10]

It is often used as a device in spacecraft measuring magnetic fields and metal detectors. More specifically, mobile phones, it is used to help judge a device’s position, relative to the north pole of the Earth. [9]

In my investigation, I hope to be able to utilize the gyroscope as means to calibrate the device correctly to the north, as a reference to the direction of travel of a device. Furthermore, by using the gyroscope of a device in tandem with the GPS, I hope to be able to show how such data can be used to map out the exact route of an exercise (one which the user travels).

Global Positioning System Receiver:



Source: National Coordination Office for Space-Based Positioning, Navigation, and Timing

GPS is a radio navigation system developed and owned by the US government, which uses radio waves between satellites in space and receivers on a device **to triangulate a device’s position on Earth**. The original GPS used in earlier mobile phones required multiple satellites to pinpoint the position of a phone, which often took a lot of power and was often very slow as radio waves often became obstructed between satellites and receivers.

However, nowadays most mobile phones use AGPS (Assisted Global Positioning System), which is an improved version of GPS.

AGPS works by introducing data from **cellular services and cell towers to “ping” the location of a device**. This entirely depends on how many cell towers are near a device, but it is often quite reliable in triangulating a device’s position.

In tandem with the magnetometer, AGPS can be used to track the distance, location, and route of an exercise.

For Androids, this data is available through the Location data class, which has various attributes and methods, such as getSpeed, getLatitude, getLongitude etc. I aim to investigate this further and be able to manipulate such data in my Android app.

Camera Sensor

The camera sensors found within phones is a complex feat of engineering, refined through the decades. We started with cameras using light and physical films, to now being able to use digital cameras in our mobile phones.



Example of camera sensor – Techspot https://www.techspot.com/guides/850-smartphone-camera-hardware/

The sensors found within most mobile phones today works by using CMOS (complementary metal-oxide-semiconductor) technology. Photodetectors separate a given image into individual pixels and measure analogue information to determine a value for each pixel. As there are millions of these photodetectors on the sensor, real-life images can be replicated digitally by stitching up the respective pixels into one single image. The role of the lens is to focus light onto the area of the sensor so that images are crisp and clear. [11]

For my investigation, I will focus on how we can use a camera on a mobile phone to detect patterns in the live image data to determine what exercise is being performed by the user. This will help me to understand the realm of computer vision and how we can make computers become able to recognise patterns within data.

Six degrees of freedom:

In my research of these sensors, one concept that kept being mentioned was the notion of 6 degrees of freedom. This is the idea of combining both translation and rotation in all 3 dimensions, to make a rigid body freely move in a given space. [12]



By GregorDS - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=38429678

It is by using the gyroscope and accelerometer in person, that a mobile phone can fully track the exact movements and rotations that a user makes with a mobile device. Furthermore, I hope to be able to use this idea of 6 degrees of freedom in my investigation, to make exercise detections more accurate and defined.

It is due to all these various sensors, that we can now use algorithms to track and trace various activities and I hope to be able to use them as part of my investigation in exercise detection.

Pedometer Algorithms

A pedometer is a simple device that combines the sensors mentioned above into a single device. A usual pedometer will output a basic user interface, displaying the step count with functions to reset the step count etc. Despite the simplicity, there is much more to a pedometer than meets the eye, especially concerning the algorithms which determine the number of steps.



A digital Omron HJ-112 pedometer

As mentioned before, the accelerometer measures acceleration within 3 dimensions (x, y and z) so to count the steps that a person takes we have to consider whether they are moving in the z-direction, bounces in the y-direction and if the phone maintains orientation throughout the walk. [13]

Furthermore, we can plot the acceleration in the y axis of a typical person walking onto a graph. In a perfect scenario it would look like this:

A graph of acceleration in the y axis over time of someone walking https://www.aosabook.org/en/500L/a-pedometer-in-the-real-world.html

What makes a step a distinct step is whether or not at any point the acceleration in the y-direction reaches a peak. In this case, as we can see three peaks, these are three steps.

Alternatively, we can consider the number of troughs. Similarly, this represents the point at which the person decelerates in the y axis (when their body accelerates downwards towards Earth when taking a step.)

Mathematically, the acceleration graph of a perfect step can be modelled as a sine graph.

Total acceleration consisting of user and gravitational. Figure 16.4 Component Signals - https://www.aosabook.org/en/500L/a-pedometer-in-the-real-world.html

Another thing to note is how the raw data from the accelerometer sensor is a combination of both the user acceleration and the gravitational acceleration of the Earth. To get a more accurate graph of the user acceleration, we must take away the factor of gravitational acceleration from the total acceleration.

In producing a working pedometer algorithm to measure steps, I will need to consider some sort of filtration which can determine the gravitational acceleration and remove this to output the user acceleration for analysis. Furthermore, I will need to research more about 3-D space and vectors, especially on using dot product in comparing the magnitude and direction of the user’s acceleration.



http://datahacker.rs/dot-product-inner-product/

## Exercises and Activities

There are many different exercises and activities out there. From within Google Fit’s activity tracking app alone, there are over 50 different types of activities that can be recorded on their app. Some can be easily tracked, whereas others are harder to detect by Google’s algorithm. For example, Google Fit can easily detect and track someone going for a run, with the specific step count, distance etc. However, when it comes to tracking activity such as rowing, Zumba or even scuba diving, this becomes significantly more complex.



Some Activity types found within Google Fit

For simplicity’s sake, I will split the different types of exercises into two main categories: aerobic exercise, and strength exercises. For aerobic exercises, these are ones in which motion is needed, and the use of the accelerometer, gyroscope and GPS is required.

For strength exercises, these are activities such as push-ups, planks and pull-ups. I aim to use the camera sensor on a mobile device and MLKit Vision Library by Google for computer vision and detection of poses/exercises. As MLKit Vision does not entirely provide exercise detection immediately, I may have to develop a simple computer vision algorithm, geared towards exercises.

## Current Market and Applications

In the current market, various businesses and competitors offer a wide assortment of fitness and exercise apps. Some include activity and exercise tracking, whilst others simply display exercise for people to follow. Here are some of the most noticeable apps:

**Leap Fitness Group / ABISHKKING LIMITED.:**

Currently, as of 2021, one of the most common and prominent companies producing fitness/health apps in Android and IOS platforms is the “Leap Fitness Group”. Their apps have a consistent design, UI, and theming that makes them recognizable, especially amongst the Google Play Store. Furthermore, they do not have a niche market, but produce apps that are wide-ranging with various activities and demographics – there is always an app made by them that covers the wide appeal of the general public. They have millions and millions of downloads.

This incentive of producing as many apps as they can for fitness/health be one of the reasons why they are so successful in the Google Play Store because they can cover so many different aspects of fitness and exercise. From this, they use advertisements and monetization to gain profits.



Apps by Leap Fitness Group on the Google Play Store https://play.google.com/store/apps/developer?id=Leap+Fitness+Group&hl=en\_GB&gl=US

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Able to cover a wider audience and hitting every part of the market demographic, whether male or female, strength, or fitness etc.  All apps are free to install for users and can run on most Android versions.  Apps normally include a large range of exercises that people can choose from. Includes tracking information such as calorie burn, in each exercise done. Text to speech directions is also included for exercises.  Most of their apps receive 4.4+ out of 5 ratings, so is a reputable fitness app company. | App experience does not feel unique between apps, so it feels like they have been copied and pasted.  Too many apps in production and development mean it may be hard for the company to manage.  Updates and resolving issues in apps may be slow, causing a bad user experience.  Users may get annoyed at ads being featured in their menus unless they purchase a pro version that removes the ads. |

**Google Fit:**



Google Fit App by Google LLC on the Google Play Store https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=en\_GB&gl=US

Google Fit is a widely used tracking app that mainly focuses on detecting what activity you may have done. The app displays to users their step count, their activity history, calorie burnt as well as data from other devices, such as smartwatches.

Google fit is very well integrated with Google Play Services, through Google’s Fitness API. This API is easily accessed by developers who also want to extract and use data from a person’s device for their android app, or certain web applications.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Google Fit is well integrated into Android, and especially Google services, so all data is available through the cloud.  Tracking algorithms are surprisingly accurate and may even detect what type of activity you are doing without the user’s input.  Google Fit API can be easily used by other apps to update the user’s activities on their Google account.  Receives regular updates and support from Google’s developers, leading to a generally good user experience. | Only serves as the main storer of data and tracking, rather than giving user’s exercises to follow etc.  Most common for Android users to use this, as Apple users are more likely to use Apple Fitness+ for their fitness tracking. |

## Third-Party Feedback

My third party will be focused on young people who are generally fit and able to do exercises. Here is a list of subjects who have agreed to give me feedback throughout this course:

**Joey Hui, 31 Male**

Joey Hui is currently job-searching for jobs within London. He previously has experience in the fashion sector and worked for a tailor. Now he is looking to enter the IT sector and learn programming languages.

Here is a little background to Joey’s workout routine:

I exercise 5 to 6 times a week at Pure Gym Finchley, however I am flexible if I feel I am too fatigued, I will rest more.

I prefer exercising in a place other than the home because this helps me concentrate during my workout, and the equipment required for a good full body workout would be unfeasible for the home.

I mainly do a bodybuilding style workout and train every muscle. My aim is to grow so I usually choose a light manageable weight and do higher repetitions, focusing on maintaining good technique and a mind-muscle connection. As I become more familiar with an exercise, I gradually increase the weight I use whilst maintaining good technique/form. If the weight is too heavy to maintain good form, then I will reduce the weight until my body has grown to a point that it can handle more.

Joey’s workout routine can be found at appendix 1.

Key takeaways from Joey’s workout routines/fitness:

1. For Joey, exercising is a big part of his daily life and can be considered to be the upper quartile of people, exercising more than the average person.
2. Joey values muscle strength training over cardio.
3. Naturally, fatigue is the biggest hurdle for Joey, especially as he exercises 5 to 6 times a week.



I usually start with running followed by a mix of either upper body or lower body weight training. My workouts include cardio, weights. i.e., running, weights, squats, deadlifts, and leg extensions.

I exercise because since I sit at home majority of the time, it makes me feel mentally/physically better if I exercise. I have felt the health benefits of exercising as I never used to which is why I try to exercise (although not as often now)

I don’t exercise as much as I used to mainly because I don’t commute into the office anymore. Before, there was a gym near my office, so I’d go there before going to work. But since working from home the gym is a bit further for me, so I don’t go as often.

**Keziah Zhou, 28 Female**

Keziah Zhou is a full-time support Engineer at a finance firm. She has over 5 years experience working in the technology and finance sector.

Here is a little background to Keziah’s workout routines:

I exercise 1-2 times a week and usually in the evening at either 6pm or 8pm (normally after dinner). I go to the gym to exercise because there is more space and equipment and I feel more motivated if I’m at the gym compared to at home.

Key takeaways from Keziah’s workout routines/fitness:

1. Keziah values overall fitness over gaining muscle strength, including how exercising helps her mental health.
2. Keziah’s biggest hurdle in exercising is being busy with other things, as well as finding the motivation to exercise.
3. Finding a suitable place to work out is another hurdle, as the home may not be the ideal place to exercise, considering how much space a person has, as well as having the right equipment.

## Prototyping and Spiking

For my investigation, I aim to be able to apply my findings from exercise detection, into a working Android app. To do this, I must first learn how to code an Android app from scratch. Because of the wider documentation, I will aim to use Java in my final project, thus I need to learn Java.

I also aim to include a client-server model to parse data to and from devices, thus I will need to learn SQL and how to implement SQL into a Java program. This will most likely mean I will have to use an API of some sort to do this.

Android Development Spiking:

As I have had no previous experience in programming in Java as well as Android Studio, I will learn all of this from scratch. Thanks to freeCodeCamp.org I followed their comprehensive Android App Development course to learn both Java and the workings of Android.

These are some screenshots of my very first applications which were basic and used basic Java logic as well as basic XML code.



Prototyping with Android Sensors:

As previously mentioned, data from an android device is available through sensor events of each sensor. Depending on the sensor type, there were a few values that can be extracted. For example, the accelerator type has sensor event data as three values within the SensorEvent.values array.



Motion sensors supported by the Android platform https://developer.android.com/guide/topics/sensors/sensors\_motion

However, the SensorEvent.values array for the uncalibrated accelerometer had 6 values, which also held data about drift (Drift is when the gyroscope sensor values change despite being stationary). As I have had no previous experience using these sensors, I thought I should develop a simple app using the gyroscope and the accelerometer sensors to test them out.

Text

Description automatically generated

Firstly, I tested out the gyroscope sensor with the help of a YouTube video. I had to create three main objects, the gyroscope of class Sensor, the sensorManager of class SensorManager and the gyroscopeEventListener of class SensorEventListener.

The sensor manager acts as the main interface between the application and the sensors. It allows an application to access a sensor, controlling the registering and unregistering of a sensor.

The SensorEventListener allows us to detect and manipulate data from a sensor. Whatever motion is detected is stored as a certain event and it is the role of the SensorEventListener to detect this.

In combination with all these three objects, we can override the method onSensorChanged in the event listener to act. In this simple application, I’ve made it so that when the user rotates to the left (in the z-axis) the background will turn blue. When the user rotates to the right (in the z-axis) the background will turn yellow. As an indication of data, I have also made it to print out the float value from the gyroscope sensor to the screen as a TextView.

Interestingly, this data from the gyroscope sensor is recorded as radians per second, which means that I will have to work with radians rather than degrees. Another thing for me to note is that the sensor values for the gyroscope are available in 3 dimensions (SensorEvent.values[0], SensorEvent.values[1], SensorEvent.values[2]), so I will need to grasp the concept of 3-dimension rotation.

Furthermore, if I were to use the uncalibrated gyroscope sensor, I will need to consider drifting and how to counteract that to improve accuracy.

Text

Description automatically generated

I created a similar application to the gyroscope prototype but using the accelerometer. Like before, I had to create three objects and a TextView to help visualise the live data that was coming through the sensor. In the case of the accelerometer, I wanted to just measure the acceleration in the z-axis. When the phone was still on a flat surface, the accelerometer measured roughly 9.8, which is the force of gravity on the phone. Then when I moved the phone up and down, the values quickly shifted in the positive and the negative, reflecting how the phone was accelerating.

Here is a video demonstrating both the simple gyroscope and accelerometer app:

<https://youtu.be/ZrCl8taJ7SY>

## Objectives and the Critical Path

**Investigation and Further Research:**

1. Investigate methods in which data from a camera sensor can be used to track and detect exercises. In doing this, I should research more about the MLKitVision library and experiment with basic computer vision.
2. Investigate methods in which data from the accelerometer and gyroscope sensors can be used to detect moving exercises, such as running and walking. I should research more on calorie specifics and how to calculate calories burnt.
3. Investigate more on the different exercises that my third-party users do, that I should include in my detections, i.e. pull-ups, sit-ups, bench presses etc. I should try to find patterns in those exercises as well as research calorie usage, intensity etc.
4. Investigate more on how to implement a GPS tracking system in an Android app, so that when a user tracks an exercise that is not static to one location, the distance and specific route can be tracked.

**Pedometer Algorithm:**

1. Implement and develop a pedometer algorithm in Java, that has at least an 80% accuracy rate in tracking the step count of a mobile phone during a brisk walk.
2. Using Android Studio, Java and XML, begin to extract data from the accelerometer (for step counting) and initially save it on a local file. This data then can be graphed and analysed for patterns.
3. Create a filtration algorithm to sanitise the raw data from the accelerometer to just receive the user acceleration. (Remove the effect of gravitational acceleration)
4. Implement an analysing algorithm to determine the number of peaks or troughs in the acceleration data, to count the number of steps. Take into consideration the different types of peaks.

**Android App:**

1. Produce an Android app that applies my findings from exercise detection algorithms, in a way that is user friendly for them to use, whilst also being able to efficiently run on most Android devices.
2. This app should have a modern/minimalistic user design, suited more for young adults. The interfaces should also be simple and easy to use, whilst adhering to Material Design.
3. Introduce a client-server model for my Android app so that information about exercises can be accessed through a database (i.e. step counts, calories burnt). This will most likely be using SQL, and a JSON file to parse information between the client and the server. Such information should then be displayed within the app within the menu of the app.
4. Within the app, there should be a scrollable menu with different types of exercises that the user can track. Once an exercise is selected, there should be a counter interface to display live information about the exercise and the duration of the exercise.
5. Once an exercise is complete, the information such as duration, calorie burnt, and distance (where applicable) should then be sent as SQL to a database and updated, so that information is not stored on the device itself, but rather on a server.
6. Get proactive feedback from my third party on exercise specific queries as well as eventual user design of the app, to ensure that my investigation remains focused on my objectives and that the project satisfies the needs of my users.
7. When dealing with data such as location, and personal information, the data should remain secure and should only be accessed by the user itself. The app should support at least two security measures to achieve this.

**Critical Path:**

# DESIGN

## High Level Overview

The app solution, which combines all my findings from my investigation about exercise detection will include 4 main components: Pedometer tracking and detection, GPS route tracking, Computer Vision detection and a client-server model. This is a very high-level and basic overview of how the app will function, with where each process will be applied.

**Languages and Libraries:**

As mentioned previously, my solution will be developed upon the Android system. The two available languages for Android development are Java and Kotlin. As Java is much more widely supported and documented then Kotlin I will use Java to develop the app. With the help of Android studio, certain functions and dependencies in my program will be imported. Here is a list below:

1. **ML Kit Vision**

Used to track and detect objects within an image, sent via an API to Google’s machine Learning services. Will aid me in developing an exercise detector using a phone’s camera. Specifically, the use of PoseDetector class and its API.

1. **CameraX (Part of Android Jetpack Library)**

Used to make it easier to use and collect data from the camera sensor, without dealing with depreciated APIs of previous camera class in Android. Also can integrate with MLK effectively.

1. **Java.SQL and PHP**

Used to set up a remote database and allow data to be transferred between devices and a main server hosting the database. Java.SQL allows me to use SQL commands within the Java application, acting as the backend of the app.

1. **Java.Math**

Used to perform mathematical functions not available with the standalone Java program. This will be needed for functions like square rooting, trigonometric functions for the pedometer algorithm.

1. **Material Design (com.google.android.material dependency)**

Used to import user interface components without having to code individual components. Each individual component conforms to the Material Design guidelines which adapt a modern and simple design standard.

## Pedometer Algorithm

Before beginning to develop a pedometer algorithm, I first needed to get a grip on the raw data of the accelerometer from an Android device. I began experimenting with the accelerometer more and investigating how I would be able to save the data into an excel graph and plot it into a graph.

Figure 1 Initial walking acclerometer tests

This graph shows the raw data that I exported into a .csv file from the accelerometer of an Android phone, with acceleration data from all three axes (relative to the phone). We can see the first major problem of developing a pedometer algorithm, in that the peaks and troughs are hard to distinctly spot. In this test, I walked 5 steps, which is hard to see in this graph. Also, there is a lot of noise, considering that in this quick test I had recorded over 2000 data points. We do observe significant troughs which do represent my exaggerated leg movements going up and down.

However, for an accurate pedometer algorithm to be developed, I must consider all three dimensions, and the algorithm must be able to accurately detect the count of steps.

From my research of other pedometer algorithms which people have developed, they follow a very universal and common structure, which is outlined in this basic flow diagram.

**Pre-Processing:**

A phone’s orientation can be at any random direction; thus, we need to isolate the acceleration in the line of gravity alone, by considering the resultant acceleration from all three data points (X, Y and Z), relative to the phone. This can be done by value of sensor TYPE\_LINEAR\_ACCELERATION, to be left with purely the acceleration done by the user’s motion.

Then, by calculating the magnitude of the resultant acceleration for each data point, we can then take comparisons of the data over time. This is done by the following equation:

*Magnitude (acceleration) = √ (x2 + y2 + z2)*

*Where X, Y and Z is the acceleration in the X axis, Y axis and Z axis respectively.*

An alternative to finding out the magnitude is to use dot product, which gives us a singular value of the acceleration in the direction of gravity (which is always down towards the Earth), telling us a good value to determine the number of steps as a step is always going up and down the line of gravity.

*Where Auser = [X1, Y1, Z1] and Bgrav = [X2, Y2, Z2]*

*DOT PRODUCT = X1 \* X2 + Y1 \* Y2 + Z1 \* Z2*

Furthermore, I will need to decide a suitable sampling rate of the acceleration. In my initial tests, I sampled the acceleration at every possible change of acceleration. However, the app would not be able to load the Android activity file for some reason and was solely focused on writing data to the .csv file. This proved to be rather impractical.

To solve this, I will need to set a suitable sampling rate of the acceleration, so that a typical mobile smartphone can handle the app, whilst not hogging up resources and files with an extremely large csv file and have enough data points to accurately record steps. (I plan to initially hold the data locally, then commit it to a server)

Graphical user interface, text, application, chat or text message

Description automatically generatedAndroid has a simple solution in the onResume() function, where you can the sampling rate. There are 4 pre-set sampling rates: fastest, game, normal and UI. Appendix 3 entails my results from testing each individual delay types plotted on a graph, by taking 20 steps each time. From this screenshot, we can see a drastic file size difference of about 30KB between the fastest sampling rate (SENSOR\_DELAY\_FASTEST), and the slowest rate.

Upon broad inspection of the varying graphs in appendix 3, the slower sampling rates can still somewhat accurately record the peaks and troughs of a walk, even though the highest sampling rate had more data points. Considering this pedometer algorithm will be running for at least 20-30 minutes in an average jog/walk, space efficiency is also a priority. Therefore, I will use the sampling rate under the attribute SENSOR\_DELAY\_NORMAL for my algorithm.

This also acts as a passive filter, in removing the jitter observed when using the higher sampling rate, caused by minute movements of the phone.

[https://developer.android.com/reference/android/hardware/SensorEvent#values](https://developer.android.com/reference/android/hardware/SensorEvent%23values)

**Filtering:**

As the data from the accelerometer introduces a lot of noise, we can implement something called a “Low-pass filter”. As there are various different applications of low pass filtering, such as usage in music and optics, I should clarify that the low pass filter that I am referring to here is one which allows data from a frequency lower than a specific frequency.

In the case of an accelerometer, a low pass filter will allow data in which the change in acceleration is measured by the frequency of how quickly the acceleration is changing. Conversely, a high pass filter will allow data from where the change in acceleration is above a certain frequency. [14]

CLASS Filter

PRIVATE ATTRIBUTE minThreshold

PRIVATE ATTRIBUTE maxThreshold

PRIVATE ATTRIBUTE LIST filteredData

CONSTRUCTOR (minThreshold, maxThreshold)

SELF.minThreshold = minThreshold

SELF.maxThreshold = maxThreshold

SELF.filteredData = new LIST

END CONSTRUCTOR

METHOD filter(unfilteredData) THEN

FOR index in unfilteredData THEN

IF unfilteredData[index] == 0 THEN

filteredData[index] = 0



Ideal Filter Response Curves https://www.electronics-tutorials.ws/filter/filter\_2.html

ELSE IF unfilteredData[index]>maxThreshold THEN

filteredData[index] = maxThreshold

ELSE IF unfilteredData[index]<minThreshold THEN

filteredData[index] = minThreshold

ELSE THEN

filteredData[index] = unfilteredData[index]

ENDIF

ENDFOR

THIS.filteredData = filteredData

END METHOD

**Detection of Peaks:**

Because the motion of a human does not follow a perfect pattern which can be easily detected, we have to consider the different types of peaks that appear in the acceleration data. This ties into how we filter our peaks, to ignore peaks which do not account to steps.

From figure 1, we can see how beside the big, large peaks, there a less noticeable little peaks. This then brings the question; do we count all the individual peaks? Do we count the medium size peaks? Do we count the peaks that are made up of smaller sized peaks?

Visually, humans can already distnict what is a peak, because relative to the graph’s pattern, we know that on average a point must be at a certain height for it to be considered a peak. However, for computers it is not that sample. To solve this issue, I will use mathematical comparisons between peaks in order to decide whether or not a peak is a peak. This can be done through a **running mean and the standard deviation** of the peaks.

If a peak is at an average height or more than the running mean, then we know that that peak is a peak.

The detection of peaks will come as the last stage, as we will need the filtred data beforehand to be inputed into the detector. Here is some pseudocode on how I imagine this will be implemented.

CLASS Detector

PRIVATE ATTRIBUTE Threshold

CONSTRUCTOR (Threshold)

#set the initial threshold when instantiating detector class

SELF.Threshold = Threshold

END CONSTRUCTOR

METHOD stepcount (filteredData as Array) THEN

Integer stepCount = 0

For index in filteredData THEN

IF filteredData[index] >= SELF.Threshold THEN

stepCount+=1

ENDIF

ENDFOR

RETURN stepCount

END METHOD

METHOD updateThreshold (Threshold) THEN

#When running mean changes by significant margin, then the threshold of detection needs to change

SELF.Threshold = Threshold

END METHOD

## GPS Tracking

In order to programmatically retrieve the GPS location of an Android device, I can use the LocationManager class and the LocationListener interface, to be left with 2 values, the latitude, and the longitude of a device.

As part of Android’s permissions, I also need to request access to the device’s internet and cellular services in order to do so. This is done through the android manifest file in XML, which specifies all the minimum requirements that the app needs.

Here is a simple pseudocode that shows how this will be implemented:

object LocationManager as var locationManager

object LocationListener as var LocaitonListener

//initiate GPS services for app

locationManager.requestLocationService()

//overriding onLocationChanged in class “LocationListner” function

String longitude = getLongitude()

String latitude = getLattitude()

String entry = longitude + “,” + latitude

EXPORT entry to array

**Haversine Formula**:

Due to the spherical shape of the Earth, the calculation of two points on an Earth is not as simple as drawing a straight line. We need to consider the angles between two points, as well as how the Earth curves. A formula that was developed to calculate the distance between two points of longitude and longitude is the Haversine formula which is:

Where d is the distance between two points, r is the radius of the Earth (6,371km), φ is the latitude and λ is the longitude. Φ1 λ1 is one point on the Earth and Φ2 λ2 is another point on the Earth.



Gade, K. (2010). A Non-singular Horizontal Position Representation. Journal of Navigation, 63(3), 395-417. doi:10.1017/S0373463309990415

## Computer Vision and ML Kit Library

Depending on the complexity of computer vision in tracking exercises which are done in one place, such as squats or push-ups, I will either try to develop a simple algorithm in Java or try to use Google’s machine learning kit library to do so. Either way, I will need to do a lot of research and learning as I have never really dealt with computer vision before.

<https://developers.google.com/ml-kit/vision/pose-detection/android>

<https://developers.google.com/ml-kit/vision/pose-detection/classifying-poses>

## User Defined Classes

As my app is developed in Java, all the functionality is built around OOP, where methods and attributes are called via objects/classes, rather than procedurally. Therefore, naturally I would have to deal with objects and classes, as well as developing my own classes.

For each activity and fragment that is called, they each need a separate class to handle the logic behind each visual layout of the app, i.e. the main menu would need it’s own activity class, the running activity would also need its own class etc. They all would inherit from Android’s AppCompatActivity class.

For dealing with handling data and methods specific to exercise data, I would need to create classes for them. I.e. the user would need its own class where it can store information statically about the user’s height, age, etc. For a given route, it would need a class to hold information about all the coordinates from the gps etc.

Icon

Description automatically generated with medium confidence

## Database Design

As my app will be dealing with multiples pieces of data and will use a client server model for data to be accessed by users, I will need to create a database that is optimal for multiple clients and learn how to implement these transactions in Java.

**Quick Overview of the Data Involved:**

From a high-level perspective I have identified two main “things” or nouns which the data in my app exist around. These are the:

* User (Person using the app)
* The exercise

Further data about the user:

* Personal information (name, age, weight, height)
* User’s past exercise history of 30 days
* User’s friend network

Further data about the exercise:

* Calories
* Duration
* Route
* Distance
* Step count

Other pieces of data would be the current leader board with the points associated with each user.

**Calorie Count:**

My app should be able to use data of an activity/exercise that has been tracked, to then calculate the calories burnt by the user. By definition, a calorie is the amount of energy needed to raise the temperature of 1 kilogram of water by 1 degree Celsius. One calorie is equal to 4.184 kilojoules of energy. [15]

But how does the calorie relate to the exercise? Are there also not variations of calories burnt depending on how hard the person worked out? Or how long the person worked out? Or how fit the person is, in terms of the weight, height etc. To solve this, a measure was calculated for each different activity, which is known as the **Metabolic Equivalent of Task (MET).** It is the ratio between the total calories burnt and the mass of the person doing the exercise over a period of time.

*MET = Calories burnt (kcal) / (mass (kg) \* time (hr))*

Rearranging this equation gives us a calculation that can give us an approximation of the number of calories burnt by a person doing an exercise over a period of time.

*Calories Burnt (kcal) = MET \* mass (kg) \* time (hr)*

This MET value is calculated through a collective study of people’s oxygen intake whilst doing an activity. As I do not have the resources and time to do such a large-scale study, I will simply use MET values online that are widely accepted. I will base my values of this website:

<https://golf.procon.org/met-values-for-800-activities/>

In my database, I should include this value of MET, as part of an exercise, so that the calorie burnt of an exercise can be derived from the weight of the user, and this MET value, as well as the duration of the exercise.

**Database Relationships:**

Text

Description automatically generatedUpon quick inspection, this is a simple entity relationship diagram of my database.

The user entity holds data about the user, the exercise entity holds data about an exercise, and the activity entity links a user with the exercise.

Here are my initial entity descriptions:

User (userID, firstname, surname, dateOfBirth, weight, height)

Activity (activityID, *exerciseID, userID*, date, timeStarted, duration)

Exercise (exerciseID, exerciseName, exerciseType, metValue)

However, these entity descriptions will not be sufficient for my application, as I have missed out on relationships such as user’s friends and distance and routes for specific exercises.

Graphical user interface, application

Description automatically generated

My thought process for this is to store friend connections and routes as separate entities. For connections, I should store individual connections between two users under a separate table. For the route, I should store key coordinates, along with the corresponding activity. Multiple entries within the Route table should then make a whole route of an activity. This is my new amended entity descriptions:

User (userID, firstname, surname, dateOfBirth, weight, height)

Friendship (friendshipID, userID, userID)

Activity (activityID, *exerciseID, userID*, date, timeStarted, duration, *route*)

Exercise (exerciseID, exerciseName, exerciseType, metValue)

Route (routeID, *activityID*, longitude, latitude, timestamp)

With these key entities, I will aim to set up a normalised database that conforms to third normal form.

**Normalisation:**

For a database to be in first normal form, it must have no repeating groups of data and all the attributes of each table must be at the atomic level.

Upon inspection, I can safely say that all my entity tables are at 1NF and in 2NF, because all my entity tables do not include any composite keys where there are partial key dependencies.

For a database to be in third normal form, there must be no non-key dependencies. This is an issue for the exercise table, where the exerciseType depends on the exerciseName and the metValue also depends on the exerciseNam

For the Friendship table, I envision a lot of repeated values for the UserID, where by a user has multiple friendships and a new instance would have to be created for each friendship. However, as the alternative option would be to add another attribute to the User table that enlists all the UserIDs of each user’s friends, it is more suitable to use a friendship table linking two users together for each instance as each instance will still be unique.

For the exercise table, I was unsure about the attribute ExerciseType, as many exercise instances may have the same type, causing a lot of repeated data. A solution to this would be to create a separate table linking an ExerciseTypeID to an exercise type. In doing so, I could also store more general information about exercise types which links more information to the exercise table.

ExerciseType (TypeID,typeName)

Exercise (exerciseID, exerciseName, *TypeID*, metValue)

Graphical user interface, application

Description automatically generatedUsing the website previously on the METs, I can also use their classification of the exercises for the different types of exercises i.e. home activities, conditioning etc. [16]

**Final Database Design:**

CREATE TABLE User (

UserID int NOT NULL,

firstname varchar(50) NOT NULL,

surname varchar(50) NOT NULL,

dateOfBirth DATE NOT NULL,

weight FLOAT(24),

height int,

PRIMARY KEY (UserID)

);

CREATE TABLE Friendship(

FriendshipID int NOT NULL,

userID1 int NOT NULL,

userID2 int NOT NULL,

PRIMARY KEY(FriendshipID),

FOREIGN KEY (USERID1) REFERENCES User(UserID),

FOREIGN KEY (USERID2) REFERENCES USER(UserID)

);

CREATE TABLE Activity(

ActivityID int NOT NULL,

ExerciseID int NOT NULL,

UserID int NOT NULL,

Date DATE NOT NULL,

timeStarted time NOT NULL,

duration int NOT NULL,

PRIMARY KEY (ActivityID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ExerciseID) REFERENCES Exercise(ExerciseID)

);

CREATE TABLE Exercise(

ExerciseID int NOT NULL,

Name varchar(255) NOT NULL,

TypeID int NOT NULL,

metValue FLOAT(24) NOT NULL,

PRIMARY KEY (ExerciseID),

FOREIGN KEY (TypeID) REFERENCES ExerciseType(TypeID)

);

CREATE TABLE ExerciseType(

TypeID int NOT NULL,

Name varchar(255) NOT NULL,

PRIMARY KEY (TypeID)

);

CREATE TABLE Route(

RouteID int NOT NULL,

ActivityID NOT NULL,

Longitude varchar(255) NOT NULL,

Latitude varchar(255) NOT NULL,

Timestamp Time NOT NULL,

PRIMARY KEY (RouteID),

FOREIGN KEY (ActivityID) REFERENCES Activity(ActivityID)

);

**Example Data:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User | | | | | |
| UserID | firstname | surname | dateOfBirth | Weight | height |
| 1 | John | Smith | 19/10/1999 | 64 | 170 |
| 2 | John | Birling | 19/11/2000 |  |  |
| 3 | Eva | Daisy | 19/12/2001 | 56 | 174 |

|  |  |  |  |
| --- | --- | --- | --- |
| Exercise | | | |
| ExerciseID | Name | TypeID | metValue |
| 1 | Running | 2 |  |
| 2 | Walking | 2 |  |
| 3 | Push-Up | 1 |  |

|  |  |
| --- | --- |
| ExerciseType | |
| TypeID | Name |
| 1 | Conditioning |
| 2 | Cardiovascular |
| 3 | Miscellaneous |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Activity | | | | | |
| ActivityID | ExerciseID | UserID | Date | timeStarted | Duration |
| 1 | 1 | 2 | 21/10/2021 | 10:22:12 | 402 |
| 2 | 2 | 3 | 21/10/2021 | 11:30:01 | 512 |
| 3 | 2 | 1 | 22/10/2021 | 11:45:21 | 700 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Route | | | | |
| RouteID | ActivityID | Longitude | Latitude | timestamp |
| 1 | 1 | 2 | 21/10/2021 | 10:23:00 |
| 2 | 1 | 3 | 21/10/2021 | 10:45:00 |
| 3 | 1 | 1 | 22/10/2021 | 10:46:00 |

|  |  |  |
| --- | --- | --- |
| Friendship | | |
| FriendshipID | UserID1 | UserID2 |
| 1 | 1 | 2 |
| 2 | 1 | 3 |
| 3 | 3 | 2 |

## Data Flow Diagrams

## User Interface Design

As my target audience are young adults, it would make more sense for the user interface design to be modern and simple. Moreover, it should be easy to navigate on the phone, especially one-handed. For this reason, all the main functions and buttons of the app should come at the bottom portion of the device, whilst other information should come at the top, similar to the Samsung One UI.

Moreover, there should be generally rounded rectangular buttons to feel more natural with simple graphic favicons to represent different functions within the app. This becomes easier to implement in an Android device, through Material Design, which allows the specific styling of components in an app to be imported into the app with ease.



Samsung’s One UI: a Singular Smartphone Experience https://news.samsung.com/global/samsungs-one-ui-a-singular-smartphone-experience

I have identified 6 main user screens that will be displayed: Log-in screen, live running, leader board, route history, tracking an exercise and a setting screen. They will be accessible by the user through a bottom navigation menu, representing each screen (apart from the log-in screen which is initiated at the start-up).

**Log in screens upon start-up:**

A picture containing bar chart

Description automatically generatedA picture containing diagram

Description automatically generatedUsers prompted with log in screen, consisting of simple UI with textbox inputs to either create an account or log into an existing account. Integration of “sign in with Google” to simplify and ease the process.

Chart

Description automatically generated**Other user screens:**

# TECHNICAL SOLUTION

## Issues and Logs:

In order to be able to perform dot product on values for user linear acceleration and values for gravity, to get the acceleration in the direction of gravity, I needed to access from both SENSOR\_TYPE\_LINEAR\_ACCELERATION and SENSOR\_TYPE\_GRAVITY at the same instance. However, as accessing android sensor data is asynchronous, I needed to find a solution to this.

From this stack overflow thread, they suggested to generalise the sensor type, and create if statements of which filter either the sensor data from the acceleration or gravity sensor.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated



Pre-Processing data, duration of the whole second when data is updated, thus passes through if statement if (seconds%5), resulting in small, segmented sets of data being sent to filter. Solved by introducing Boolean variable to check if for a 5 second interval, whether or not a chunk of data has been processed. If not then processing occurs, else processing does not occur

Text

Description automatically generated

Handling Permissions:

Diagram

Description automatically generatedIn order to use GPS location services, it is necessary for to ask the user beforehand for their permission to access the device’s location, especially when accessing their exact location in the background. To handle the result of requesting a permission, the onRequestPermissionResult() is overridden in the activity. Initially, the event flow of when I accessed and asked for their permissions, kept causing crashes or bad user experiences breaking functionality.

Initial Permissions Flow

This flow diagram shows how I initially asked for permissions. We see the fundamental issue of looping forever, forcing the user to grant permissions, or they would be stuck in a loop.

Amended Permissions Flow

The flow diagram above details the amended permissions flow, to prevent looping and define at what point should a permission be requested and checked.

Distance calculation:

When doing exercise on a treadmill, it would not make any sense to track the distance using the GPS location, therefore, it is based on the average pace of the user, which is derived from their height.

MLK Library

Issues with learning how to use camera and with various deprecated libraries, unsure of which one to use. Still looking through documentation on how to initialise a camera feed in the app and send images to MLK to analyse. Solved by using the CameraX library specifically designed to ease process of handling different camera sensors and configs

Reading and writing activities to phone device

Would like to use phone storage to store an activity’s details, after an activity is completed. Used as a buffer place, before sending to the server for cloud storage. Will be faster for activities of a user to be stored locally initially.

First solution would be to use .csv file. Some fields are mandatory, such as type, time started, calories and duration, whereas distance and route is not mandatory.

<https://stackoverflow.com/questions/68508321/android-java-lang-noclassdeffounderror-failed-resolution-of-lcom-mysql-cj-mys>

You cannot use recent versions of MySQL Connector/J on Android, because it uses types and features not available on Android. The specific reason here is that the type com.mysql.cj.MysqlType implements java.sql.SQLType (introduced in Java 8 / JDBC 4.2), and judging by the error this type does not exist in Android. In the past, I have also seen errors related to using named groups in regular expressions, which are (or were) also not supported on Android.

Things to implement:

Audio motivation to keep pushing users on.

Does the ML library support multiracial people/ different lighting? If not can talk about ethical use of it.

## XML Files and Design

In Android, XML files are used to declare the layout and visuals of any components used in an activity/fragment. By nature of XML, the syntax is followed by tags which contain any component declared.

Moreover, layout files in android can either be done through constraints, linear layouts, or relative layouts. In my project I have mainly used constraint and linear layouts.

**Activity XML Files:**

These XML files detail the layout and design of all the Android activities, which are the main menu, running activity, walking activity, treadmill activity and push-up activity.

activity\_running.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/main\_colour"  
 >  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:text="Running Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothic"  
 android:textSize="30sp"  
 android:layout\_marginTop="30dp"  
 />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:textSize="80sp"  
 android:fontFamily="@font/gothicbb"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:layout\_marginBottom="150dp"/>  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/timerText"  
 app:layout\_constraintVertical\_bias="0"  
 android:gravity="center\_horizontal"  
 >  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/stepText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Steps:\n0"  
 android:textSize="20dp"  
 android:textAlignment="center"  
 android:layout\_margin="10dp"/>  
  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:textAlignment="center"  
 android:id="@+id/calText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Calories:\n0"  
 android:textSize="20dp"/>  
  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:textAlignment="center"  
 android:id="@+id/distText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Distance:\n0m"  
 android:textSize="20dp"/>  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/row2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="@id/row1"  
 app:layout\_constraintTop\_toBottomOf="@+id/row1"  
 app:layout\_constraintVertical\_bias="0"  
 android:gravity="center\_horizontal"  
 >  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/paceText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Pace:\n0"  
 android:textSize="20dp"  
 android:textAlignment="center"  
 android:layout\_margin="10dp"/>  
  
 </LinearLayout>  
  
  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="400dp" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/startStopBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="80dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Pause"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.172"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/finishBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="80dp"  
 android:fontFamily="@font/gothicbb"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:textColor="@color/main\_colour"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.831"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

activity\_main.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/parent"  
 tools:context=".MainActivity">  
  
 <FrameLayout  
 android:id="@+id/fragmentContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/bottom\_navigation" />  
  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/bottom\_navigation"  
 style="@style/Widget.MaterialComponents.BottomNavigationView.Colored"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 app:menu="@menu/bottom\_navigation\_menu" />  
  
</RelativeLayout>

bottom\_navigation\_menu.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <item  
 android:id="@+id/page\_1"  
 android:enabled="true"  
 android:icon="@drawable/ic\_exercise"  
 android:title="Exercise"/>  
 <item  
 android:id="@+id/page\_2"  
 android:enabled="true"  
 android:icon="@drawable/ic\_history"  
 android:title="History"/>  
 <item  
 android:id="@+id/page\_3"  
 android:enabled="true"  
 android:icon="@drawable/ic\_friends"  
 android:title="Friends"/>  
 <item  
 android:id="@+id/page\_4"  
 android:enabled="true"  
 android:icon="@drawable/ic\_settings"  
 android:title="Account"/>  
</menu>

activity\_treadmill.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/main\_colour"  
 >  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:text="Treadmill Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothic"  
 android:textSize="30sp"  
 android:layout\_marginTop="30dp"  
 />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:textSize="80sp"  
 android:fontFamily="@font/gothicbb"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:layout\_marginBottom="150dp"/>  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/timerText"  
 app:layout\_constraintVertical\_bias="0"  
 android:gravity="center\_horizontal"  
 >  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/stepText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Steps:\n0"  
 android:textSize="20dp"  
 android:textAlignment="center"  
 android:layout\_margin="10dp"/>  
  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:textAlignment="center"  
 android:id="@+id/calText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Calories:\n0"  
 android:textSize="20dp"/>  
  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:textAlignment="center"  
 android:id="@+id/distText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Distance:\n0m"  
 android:textSize="20dp"/>  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/row2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="@id/row1"  
 app:layout\_constraintTop\_toBottomOf="@+id/row1"  
 app:layout\_constraintVertical\_bias="0"  
 android:gravity="center\_horizontal"  
 >  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/paceText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Pace:\n0"  
 android:textSize="20dp"  
 android:textAlignment="center"  
 android:layout\_margin="10dp"/>  
  
 </LinearLayout>  
  
  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="400dp" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/startStopBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="80dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Pause"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.172"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/finishBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="80dp"  
 android:fontFamily="@font/gothicbb"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:textColor="@color/main\_colour"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.831"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

activity\_pushup.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/main\_colour"  
 >  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:text="Push-Up Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothic"  
 android:textSize="30sp"  
 android:layout\_marginTop="30dp"  
 />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:textSize="80sp"  
 android:fontFamily="@font/gothicbb"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 app:layout\_constraintVertical\_bias="1"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:layout\_marginBottom="80dp"/>  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintVertical\_bias="1"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/timerText"  
 android:gravity="center\_horizontal"  
 >  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/repText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Reps:\n0"  
 android:textSize="20dp"  
 android:textAlignment="center"  
 android:layout\_margin="10dp"/>  
  
 <TextView  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:textAlignment="center"  
 android:id="@+id/calText"  
 android:fontFamily="@font/gothic"  
 android:textColor="@color/black"  
 android:text="Calories:\n0"  
 android:textSize="20dp"/>  
 </LinearLayout>  
  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="250dp" />  
  
 <androidx.camera.view.PreviewView  
 android:layout\_width="match\_parent"  
 android:layout\_height="350dp"  
 android:id="@+id/tv"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintVertical\_bias="0">  
 </androidx.camera.view.PreviewView>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/startStopBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="50dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Pause"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.172"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/finishBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="50dp"  
 android:fontFamily="@font/gothicbb"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:textColor="@color/main\_colour"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textSize="30dp"  
 app:cornerRadius="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.831"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Scatter chart

Description automatically generated with medium confidenceA picture containing shape

Description automatically generatedXML Design Previews:

Graphical user interface

Description automatically generated with medium confidenceA picture containing graphical user interface

Description automatically generated

**Fragment XML Files:**

fragment\_exercise.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="60dp"  
 android:text="Exercise"  
 android:fontFamily="@font/gothicbb"  
 android:textAlignment="center"  
 android:textSize="40sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.126" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
 <ScrollView  
 android:scrollbarThumbVertical="@color/grey"  
 android:layout\_width="match\_parent"  
 android:layout\_height="500dp"  
 app:layout\_constraintVertical\_bias="1"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline2"  
 app:layout\_constraintBottom\_toBottomOf="parent">  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
 <com.google.android.material.card.MaterialCardView  
 android:id="@+id/card1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="3dp"  
 app:cardCornerRadius="20dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="125dp"  
 android:layout\_margin="10dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/runningman" />  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:gravity="center\_horizontal"  
 android:text="Running exercise"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/runBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 android:fontFamily="@font/gothicbb"  
 android:onClick="sendOnChannel1"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false"  
 android:textAlignment="center" />  
 </LinearLayout>  
 </LinearLayout>  
 </com.google.android.material.card.MaterialCardView>  
  
  
  
 <com.google.android.material.card.MaterialCardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="3dp"  
 app:cardCornerRadius="20dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="125dp"  
 android:layout\_margin="10dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/walkingimg" />  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:gravity="center\_horizontal"  
 android:text="Walking"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/walkBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 android:fontFamily="@font/gothicbb"  
 android:textAllCaps="false"  
 android:text="@string/start\_tracking"  
 android:textAlignment="center" />  
 </LinearLayout>  
 </LinearLayout>  
 </com.google.android.material.card.MaterialCardView>  
 <com.google.android.material.card.MaterialCardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="3dp"  
 app:cardCornerRadius="20dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="125dp"  
 android:layout\_margin="10dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/treadmill" />  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:gravity="center\_horizontal"  
 android:text="Treadmill"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/treadmillBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 android:fontFamily="@font/gothicbb"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false"  
 android:textAlignment="center" />  
 </LinearLayout>  
 </LinearLayout>  
 </com.google.android.material.card.MaterialCardView>  
 <com.google.android.material.card.MaterialCardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="3dp"  
 app:cardCornerRadius="20dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="125dp"  
 android:layout\_margin="10dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/pushupimg" />  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:gravity="center\_horizontal"  
 android:text="Push-Up"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/pushUpBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 android:fontFamily="@font/gothicbb"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false"  
 android:textAlignment="center" />  
 </LinearLayout>  
 </LinearLayout>  
 </com.google.android.material.card.MaterialCardView>  
  
 </LinearLayout>  
 </ScrollView>  
</androidx.constraintlayout.widget.ConstraintLayout>

fragment\_friends.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="60dp"  
 android:text="Your Friends"  
 android:fontFamily="@font/gothicbb"  
 android:textAlignment="center"  
 android:textSize="40dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.126" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

fragment\_history.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="60dp"  
 android:text="Activity History"  
 android:fontFamily="@font/gothicbb"  
 android:textAlignment="center"  
 android:textSize="40dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.126" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
 <ListView  
 android:id="@+id/activityList"  
 android:layout\_width="match\_parent"  
 android:layout\_height="500dp"  
 app:layout\_constraintVertical\_bias="1"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline2"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="@id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"/>  
  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

fragment\_settings.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="60dp"  
 android:text="Settings"  
 android:textAlignment="center"  
 android:textSize="40dp"  
 android:fontFamily="@font/gothicbb"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.126" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="500dp"  
 app:layout\_constraintVertical\_bias="1"  
 android:gravity="center\_horizontal"  
 android:orientation="vertical"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline2"  
 app:layout\_constraintBottom\_toBottomOf="parent">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/weightField"  
 android:hint="Weight (kg)"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_width="300dp"  
  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content">  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:inputType="numberDecimal" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/DOBfield"  
 android:hint="Date Of Birth (dd/mm/yy)"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_width="300dp"  
 android:layout\_marginTop="30dp"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content">  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:id="@+id/dobText"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:inputType="date" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/heightField"  
 android:hint="Height (cm)"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_width="300dp"  
 android:layout\_marginTop="30dp"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content">  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:id="@+id/heightText"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:inputType="number" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/UpdateButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_marginTop="150dp"  
 android:layout\_height="wrap\_content"  
 android:text="Update My Details"  
 android:fontFamily="@font/gothicbb"  
 android:textAllCaps="false"  
 android:textSize="20sp"  
 app:cornerRadius="10dp" />  
  
  
 </LinearLayout>  
  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Graphical user interface

Description automatically generated with medium confidenceA picture containing graphical user interface

Description automatically generatedGraphical user interface, text

Description automatically generated

## Fragments and Activities

These java classes detail the logic behind the layout files, handling how the text views, buttons, constraints etc. In my application I use fragments which allow me to interchangeably switch what is being displayed on the screen via a navigation menu. Activities which are focused can then be called to be drawn on top of the app.

My fragment classes are called ActivityFragment, ExerciseFragment, HistoryFragment and SettingsFragment, each representing the 4 different menus that can be selected within the main app (not including the log in screen)

MainActivity.java

package com.example.exercisetracker;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
  
import android.os.Bundle;  
import android.view.MenuItem;  
import com.google.android.material.bottomnavigation.BottomNavigationView;  
import com.google.android.material.navigation.NavigationBarView;  
  
import java.util.Date;  
  
public class MainActivity extends AppCompatActivity{  
 private BottomNavigationView bottomMenu;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 getSupportActionBar().hide();  
 bottomMenu = findViewById(R.id.*bottom\_navigation*);  
 bottomMenu.setSelectedItemId(R.id.*page\_1*);  
 Fragment fragment = new ExerciseFragment();;  
 getSupportFragmentManager().beginTransaction().replace(R.id.*fragmentContainer*,fragment).commit();  
 bottomMenu.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {  
  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
 Fragment fragment = new ExerciseFragment();;  
 switch(item.getItemId()){  
 case(R.id.*page\_1*):  
 fragment = new ExerciseFragment();  
 break;  
 case(R.id.*page\_2*):  
 fragment = new HistoryFragment();  
 break;  
 case(R.id.*page\_3*):  
 fragment = new FriendsFragment();  
 break;  
 case (R.id.*page\_4*):  
 fragment = new SettingsFragment();  
 break;  
 }  
 getSupportFragmentManager().beginTransaction().replace(R.id.*fragmentContainer*,fragment).commit();  
 return true;  
 }  
 });  
  
 *//HANDLING USER CLASS* User.*setWeight*(55F);  
 Date date = new Date("12/02/04");  
 User.*setDateOfBirth*(date);  
 User.*setHeight*(166);  
  
 }  
  
  
  
}

ExerciseFragment.java

package com.example.exercisetracker;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContract;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.core.app.ActivityCompat;  
import androidx.fragment.app.Fragment;  
  
public class ExerciseFragment extends Fragment implements View.OnClickListener, ActivityCompat.OnRequestPermissionsResultCallback{  
 private String[] PERMISSIONS;  
  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.*fragment\_exercise*,container,false);  
 Button runBtn = (Button) view.findViewById(R.id.*runBtn*);  
 Button walkBtn = (Button) view.findViewById(R.id.*walkBtn*);  
 Button treadmillBtn = (Button) view.findViewById(R.id.*treadmillBtn*);  
 Button pushUpBtn = (Button) view.findViewById(R.id.*pushUpBtn*);  
 treadmillBtn.setOnClickListener(this);  
 pushUpBtn.setOnClickListener(this);  
 runBtn.setOnClickListener(this);  
 walkBtn.setOnClickListener(this);  
 return view;  
 }  
  
 @SuppressLint("NonConstantResourceId")  
 @Override  
 public void onClick(View v) {  
 switch(v.getId()){  
 case R.id.*runBtn*:  
 Intent intent1 = new Intent(getContext(), RunningActivity.class);  
 startActivity(intent1);  
 break;  
 case R.id.*walkBtn*:  
 Toast.*makeText*(getContext(), "Walking Tracking", Toast.*LENGTH\_SHORT*).show();  
 break;  
 case R.id.*treadmillBtn*:  
 Intent intent2 = new Intent(getContext(), TreadmillActivity.class);  
 startActivity(intent2);  
 break;  
 case R.id.*pushUpBtn*:  
 Intent intent3 = new Intent(getContext(), PushUpActivity.class);  
 startActivity(intent3);  
 break;  
 }  
 }  
  
  
}

FriendsFragment.java

package com.example.exercisetracker;  
  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
  
public class FriendsFragment extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.*fragment\_friends*,container,false);  
 }  
}

HistoryFragment.java

package com.example.exercisetracker;  
  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
  
public class HistoryFragment extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.*fragment\_history*,container,false);  
 }  
}

SettingsFragment.java

package com.example.exercisetracker;  
  
import android.annotation.SuppressLint;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.InputType;  
import android.view.LayoutInflater;  
import android.view.MotionEvent;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
  
import com.google.android.material.datepicker.CalendarConstraints;  
import com.google.android.material.datepicker.DateValidatorPointBackward;  
import com.google.android.material.datepicker.MaterialDatePicker;  
import com.google.android.material.datepicker.MaterialPickerOnPositiveButtonClickListener;  
import com.google.android.material.textfield.TextInputEditText;  
import com.google.android.material.textfield.TextInputLayout;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.time.Instant;  
import java.time.ZoneOffset;  
import java.time.ZonedDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Date;  
  
public class SettingsFragment extends Fragment implements View.OnClickListener{  
 private TextInputLayout weightField;  
 private TextInputLayout DOBField;  
 private TextInputLayout heightField;  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.*fragment\_settings*,container,false);  
 weightField = view.findViewById(R.id.*weightField*);  
 DOBField = view.findViewById(R.id.*DOBfield*);  
 heightField = view.findViewById(R.id.*heightField*);  
  
 *//loading user data presets* weightField.getEditText().setText(User.*getWeight*().toString());  
 heightField.getEditText().setText(User.*getHeight*().toString());  
  
 *//handling date of birth* Date dob = User.*getDateOfBirth*();  
 EditText dobText = DOBField.getEditText();  
 DateFormat df = new SimpleDateFormat("dd/MM/yyyy");  
 String strdob = df.format(dob);  
 dobText.setText(strdob);  
 dobText.setInputType(InputType.*TYPE\_NULL*);  
 dobText.setKeyListener(null);  
 CalendarConstraints.Builder constraints = new CalendarConstraints.Builder().setValidator(DateValidatorPointBackward.*before*(MaterialDatePicker.*todayInUtcMilliseconds*()));  
 MaterialDatePicker.Builder<Long> datepickerBuilder = MaterialDatePicker.Builder.*datePicker*();  
 datepickerBuilder.setCalendarConstraints(constraints.build());  
 MaterialDatePicker datepicker = datepickerBuilder.build();  
  
 dobText.setOnTouchListener(new View.OnTouchListener() {  
 @SuppressLint("ClickableViewAccessibility")  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 if (event.getAction() == MotionEvent.*ACTION\_UP*){  
 datepicker.show(getActivity().getSupportFragmentManager(), "Date Picker");  
 }  
 return false;  
 }  
 });  
  
 datepicker.addOnPositiveButtonClickListener(new MaterialPickerOnPositiveButtonClickListener() {  
 @Override  
 public void onPositiveButtonClick(Object selection) {  
 *//saving the entered date and formatting date* dobText.setText(df.format(datepicker.getSelection()));  
 Date date = new Date((Long) datepicker.getSelection());  
 User.*setDateOfBirth*(date);  
 datepicker.dismiss();  
 }  
 });  
 datepicker.addOnCancelListener(new DialogInterface.OnCancelListener() {  
 @Override  
 public void onCancel(DialogInterface dialog) {  
 datepicker.dismiss();  
 }  
 });  
  
  
  
 *//handling update button* Button updateButton = view.findViewById(R.id.*UpdateButton*);  
 updateButton.setOnClickListener(this);  
  
  
 return view;  
 }  
  
  
 @SuppressLint("NonConstantResourceId")  
 @Override  
 public void onClick(View v) {  
 switch(v.getId()){  
 case R.id.*UpdateButton*:  
  
 try {  
 Float weight = Float.*parseFloat*(String.*valueOf*(weightField.getEditText().getText()));  
 User.*setWeight*(weight);  
 Integer height = Integer.*valueOf*(String.*valueOf*(heightField.getEditText().getText()));  
 User.*setHeight*(height);  
 weightField.getEditText().setCursorVisible(Boolean.*FALSE*);  
 heightField.getEditText().setCursorVisible(Boolean.*FALSE*);  
 Toast.*makeText*(getContext(), "Save successful", Toast.*LENGTH\_SHORT*).show();  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 Toast.*makeText*(getContext(), "Save unsuccesful", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
  
}

RunningActivity.java

package com.example.exercisetracker;  
  
import static com.example.exercisetracker.BaseApp.*CHANNEL\_1\_ID*;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.app.Notification;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Environment;  
import android.os.Handler;  
import android.text.Html;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.annotation.NonNull;  
import androidx.annotation.RequiresApi;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationChannelCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
import androidx.core.content.ContextCompat;  
  
import com.google.android.material.button.MaterialButton;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Locale;  
  
public class RunningActivity extends AppCompatActivity {  
 *//Sensors* private SensorManager sensorManager;  
 private SensorEventListener listener;  
 private LocationManager locationManager;  
 private LocationListener locationListener;  
 *//TextViews* private TextView timerText;  
 private TextView stepText;  
 private TextView calorieText;  
 private TextView distText;  
 private TextView paceText;  
 *//Buttons* private MaterialButton finishBtn;  
 private MaterialButton startStopBtn;  
 *//notification* private NotificationManagerCompat notificationManagerCompat;  
  
  
 *//Specialised running variables* private float MET;  
 private double distance;  
 private Filter filter;  
 private Detector detector;  
 private Boolean isRunning;  
 private Integer seconds;  
 private Integer steps;  
 private Integer calories;  
 private Float[] filtered\_data;  
 private Boolean hasProcessed;  
 private Route route;  
 private Date timeStarted;  
  
 *//Permissions* private String[] PERMISSIONS;  
 private ActivityResultLauncher<String> requestPermissionLauncher;  
  
 @SuppressLint("MissingPermission")  
 @RequiresApi(api = Build.VERSION\_CODES.*M*)  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 *//visuals* getSupportActionBar().hide();  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* getWindow().setStatusBarColor(ContextCompat.*getColor*(this,R.color.*main\_colour*));*// set status background white* setContentView(R.layout.*activity\_running*);  
  
 *//instantiating all private variables* isRunning = true;  
 seconds = 0;  
 steps = 0;  
 distance = 0f;  
 timerText = findViewById(R.id.*timerText*);  
 stepText = findViewById(R.id.*stepText*);  
 distText = findViewById(R.id.*distText*);  
 paceText = findViewById(R.id.*paceText*);  
 calorieText = findViewById(R.id.*calText*);  
 sensorManager = (SensorManager) getSystemService(*SENSOR\_SERVICE*);  
 MET = Float.*parseFloat*(getString(R.string.*met\_running*));  
  
 *//CUSTOM JAVA CLASSES* filter = new Filter(-10f, 10f);  
 detector = new Detector(1f, 2);  
 ArrayList<Double[]> currentRoute = new ArrayList<Double[]>();  
 route = new Route(currentRoute);  
 *//NOTIFICATION MANAGER* notificationManagerCompat = NotificationManagerCompat.*from*(this);  
  
 *//handling when start and stop button clicked* startStopBtn = findViewById(R.id.*startStopBtn*);  
 startStopBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isRunning) {  
 startStopBtn.setText("Resume");  
 isRunning = false;  
  
 } else {  
 startStopBtn.setText("Pause");  
 isRunning = true;  
 }  
 }  
 });  
 finishBtn = findViewById(R.id.*finishBtn*);  
 finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 isRunning = false;  
 sensorManager.unregisterListener(listener);  
 locationManager.removeUpdates(locationListener);  
 *//exiting the running activity and sending data back to main program* Activity activity = new Activity(timeStarted,seconds,"running");  
 if (activity.saveActivity(getFilesDir().toString()) == Boolean.*TRUE*){  
 Toast.*makeText*(RunningActivity.this, "Save successful", Toast.*LENGTH\_SHORT*).show();  
 }  
 else{  
 Toast.*makeText*(RunningActivity.this, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 finish();  
 }  
 });  
  
 *//HANDLING PERMISSIONS* requestPermissionLauncher =  
 registerForActivityResult(new ActivityResultContracts.RequestPermission(), isGranted -> {  
 if (isGranted) {  
 isRunning = Boolean.*TRUE*;  
 } else {  
 Toast.*makeText*(this, "Permission Denied", Toast.*LENGTH\_SHORT*).show();  
 isRunning = Boolean.*FALSE*;  
 this.finish();  
 }  
 });  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 PERMISSIONS = new String[]{  
 Manifest.permission.*ACCESS\_COARSE\_LOCATION*,  
 Manifest.permission.*ACCESS\_BACKGROUND\_LOCATION*,  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 };  
 }  
 else{  
 PERMISSIONS = new String[]{  
 Manifest.permission.*ACCESS\_COARSE\_LOCATION*,  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 };  
 }  
 if (checkPermissions(this,PERMISSIONS) == Boolean.*FALSE*){  
 for (String permission: PERMISSIONS){  
 requestPermissionLauncher.launch(permission);  
 }  
 }  
 else{  
 startRunning();  
 }  
 }  
  
 @SuppressLint("MissingPermission")  
 private void startRunning() {  
 timeStarted = Calendar.*getInstance*().getTime();  
 *//handling location changes* locationManager = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*);  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(@NonNull Location location) {  
 if (isRunning) {  
 *//location in form of latitude and longitude* double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 Double[] entry = {latitude, longitude};  
 route.addRoute(entry);  
 }  
 }  
  
 };  
 *//sensor managers* locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*,1000,2,locationListener);  
  
 *//creating handler to run simultaneously to track duration in seconds* final Handler handler = new Handler();  
 handler.post(new Runnable() {  
 @SuppressLint("DefaultLocale")  
 @Override  
 public void run() {  
 handler.postDelayed(this, 1000);  
 if (isRunning) {  
 seconds++;  
 *//calculating distances between location updates and updating text views* if (route.getRouteSize()>=2) {  
 DecimalFormat df = new DecimalFormat("#.##");  
 route.calculateDistance();  
 distance = route.getDistance();  
 distText.setText(String.*format*("Distance:\n%sm", df.format(distance)));  
 *//changing pace text view* paceText.setText(Html.*fromHtml*("Pace:\n"+String.*valueOf*(df.format(distance/seconds.floatValue()))+"ms<sup>-1</sup"));  
  
 }  
 *//changing timer text view* int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 int secs = seconds % 60;  
 String time = String.*format*(Locale.*getDefault*(), "%d:%02d:%02d", hours, minutes, secs);  
 timerText.setText(time);  
  
 *//changing calorie text view* calories = Math.*round*(MET \* User.*getWeight*() \* (seconds.floatValue() / 3600));  
 calorieText.setText(String.*format*("Calories:\n%d", calories));  
  
 *//changing step text view* stepText.setText(String.*format*("Steps:\n%d", steps));  
 *//allowing preprocessing to happen at the instance of a 5 second interval* if ((seconds % 5) == 0) {  
 hasProcessed = Boolean.*FALSE*;  
 }  
  
 }  
  
 }  
 });  
  
  
 *//2d arrays to store a variable amount of samples, each sample consisting of the x y z values* ArrayList<Float[]> accel = new ArrayList<Float[]>();  
 ArrayList<Float[]> grav = new ArrayList<Float[]>();  
  
 listener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
  
 Sensor sensor = event.sensor;  
 *//formatting by rounding to 2 decimal places* DecimalFormat df = new DecimalFormat("#.##");  
 if (sensor.getType() == Sensor.*TYPE\_LINEAR\_ACCELERATION* & isRunning) {  
 *//handling the linear acceleration* Float x = Float.*parseFloat*(df.format(event.values[0]));  
 Float y = Float.*parseFloat*(df.format(event.values[1]));  
 Float z = Float.*parseFloat*(df.format(event.values[2]));  
 Float[] entry = new Float[3];  
 entry[0] = x;  
 entry[1] = y;  
 entry[2] = z;  
 System.*out*.println("acceleration:" + String.*format*("%f, %f, %f", entry[0], entry[1], entry[2]));  
 accel.add(entry);  
  
  
 }  
 if (sensor.getType() == Sensor.*TYPE\_GRAVITY* & isRunning) {  
 *//handling gravimeter* Float x = Float.*parseFloat*(df.format(event.values[0]));  
 Float y = Float.*parseFloat*(df.format(event.values[1]));  
 Float z = Float.*parseFloat*(df.format(event.values[2]));  
 Float[] entry = new Float[3];  
 entry[0] = x;  
 entry[1] = y;  
 entry[2] = z;  
 grav.add(entry);  
 System.*out*.println("gravity: " + String.*format*("%f, %f, %f", entry[0], entry[1], entry[2]));  
 }  
  
  
 *//PROCESSING DATA* if (((seconds % 5) == 0 && (grav.size() > 0)) && (accel.size() > 0) && (hasProcessed == Boolean.*FALSE*)) {  
 ArrayList<Float> results = new ArrayList<Float>();  
 *//PRE-PROCESSING DATA  
 //handling when grav array and accel array are unequal:* while (accel.size() != grav.size()) {  
 if (accel.size() > grav.size()) {  
 accel.remove(accel.size() - 1);  
  
 } else {  
 grav.remove(grav.size() - 1);  
 }  
 }  
  
 System.*out*.println("Seconds: " + seconds);  
 *//PERFORM DOT PRODUCT* System.*out*.println(String.*format*("gravsize: %d accelsize: %d", grav.size(), accel.size()));  
 for (int j = 0; j < grav.size(); j++) {  
 Float[] accelValues = accel.get(j);  
 Float[] gravValues = grav.get(j);  
 Float result = Float.*parseFloat*(df.format(gravValues[0] \* accelValues[0] + gravValues[1] \* accelValues[1] + gravValues[2] \* accelValues[2]));  
 if (result < 0) {  
 result = (float) Math.*sqrt*(0 - result);  
 result = 0 - result;  
 } else {  
 result = (float) Math.*sqrt*(result);  
 }  
 results.add(result);  
 System.*out*.println("result: " + j + " " + result.toString());  
 }  
 grav.clear();  
 accel.clear();  
 *//hasProcessed to true to prevent small chunks of data being processed* hasProcessed = Boolean.*TRUE*;  
  
 *//FILTERING DATA* filter.filter(results);  
 filtered\_data = filter.getFiltered\_data();  
 *//WRITING TO FILE FOR DEBUGGING  
// for (int i = 0; i < filtered\_data.length; i++) {  
// String entry = filtered\_data[i].toString() + "\n";  
// System.out.print(entry);  
// try {  
// File storage = Environment.getExternalStorageDirectory();  
// File dir = new File(storage.getAbsolutePath() + "/documents");  
// File file = new File(dir, "output.csv");  
// FileOutputStream f = new FileOutputStream(file, true);  
// try {  
// f.write(entry.getBytes());  
// f.flush();  
// f.close();  
// } catch (IOException e) {  
// e.printStackTrace();  
// }  
// } catch (FileNotFoundException e) {  
// e.printStackTrace();  
// }  
// }  
  
 //detecting steps* detector.detect(filtered\_data);  
 steps = detector.getStepCount();  
  
 }  
 }  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
 }  
 };  
 sensorManager.registerListener(listener, sensorManager.getDefaultSensor(Sensor.*TYPE\_LINEAR\_ACCELERATION*), SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 sensorManager.registerListener(listener, sensorManager.getDefaultSensor(Sensor.*TYPE\_GRAVITY*), SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
 *//PERMISSIONS* private boolean checkPermissions(Context context, String[] PERMISSIONS) {  
 *//CHECKING FOR EXISTING PERMISSIONS* if (context!=null && PERMISSIONS!=null){  
 for (String permission: PERMISSIONS){  
 if (ActivityCompat.*checkSelfPermission*(context,permission)!= PackageManager.*PERMISSION\_GRANTED*){  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
 *//handling live notification bar* public void sendOnChannel1(View v){  
 Notification notification = new NotificationCompat.Builder(this,*CHANNEL\_1\_ID*)  
 .setSmallIcon(R.id.*icon*)  
 .setContentTitle("Running Tracking")  
 .setContentText(String.*valueOf*(steps))  
 .setCategory(NotificationCompat.*CATEGORY\_WORKOUT*)  
 .build();  
 notificationManagerCompat.notify(1,notification);  
 }  
}

TreadmillActivity.java

package com.example.exercisetracker;  
  
import static com.example.exercisetracker.BaseApp.*CHANNEL\_1\_ID*;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.app.Notification;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Environment;  
import android.os.Handler;  
import android.text.Html;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.annotation.NonNull;  
import androidx.annotation.RequiresApi;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationChannelCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
import androidx.core.content.ContextCompat;  
  
import com.google.android.material.button.MaterialButton;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class TreadmillActivity extends AppCompatActivity {  
 *//Sensors* private SensorManager sensorManager;  
 private SensorEventListener listener;  
 *//TextViews* private TextView timerText;  
 private TextView stepText;  
 private TextView calorieText;  
 private TextView distText;  
 private TextView paceText;  
 *//Buttons* private MaterialButton finishBtn;  
 private MaterialButton startStopBtn;  
 *//notification* private NotificationManagerCompat notificationManagerCompat;  
  
  
 *//Specialised running variables* private float MET;  
 private double distance;  
 private Filter filter;  
 private Detector detector;  
 private Boolean isRunning;  
 private Integer seconds;  
 private Integer steps;  
 private Integer calories;  
 private Float[] filtered\_data;  
 private Boolean hasProcessed;  
 private Integer height;  
  
 *//Permissions* private String[] PERMISSIONS;  
 private ActivityResultLauncher<String> requestPermissionLauncher;  
  
  
 @SuppressLint("MissingPermission")  
 @RequiresApi(api = Build.VERSION\_CODES.*M*)  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getSupportActionBar().hide();  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* getWindow().setStatusBarColor(ContextCompat.*getColor*(this,R.color.*main\_colour*));*// set status background white* setContentView(R.layout.*activity\_treadmill*);  
 *//instantiating all private variables* isRunning = true;  
 seconds = 0;  
 steps = 0;  
 distance = 0f;  
 height = User.*getHeight*();  
 MET = Float.*parseFloat*(getString(R.string.*met\_treadmill*));  
  
 timerText = findViewById(R.id.*timerText*);  
 stepText = findViewById(R.id.*stepText*);  
 distText = findViewById(R.id.*distText*);  
 paceText = findViewById(R.id.*paceText*);  
 calorieText = findViewById(R.id.*calText*);  
 sensorManager = (SensorManager) getSystemService(*SENSOR\_SERVICE*);  
  
 *//CUSTOM JAVA CLASSES* filter = new Filter(-10f, 10f);  
 detector = new Detector(1f, 2);  
 *//NOTIFICATION MANAGER* notificationManagerCompat = NotificationManagerCompat.*from*(this);  
  
 *//handling when start and stop button clicked* startStopBtn = findViewById(R.id.*startStopBtn*);  
 startStopBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isRunning) {  
 startStopBtn.setText("Resume");  
 isRunning = false;  
  
 } else {  
 startStopBtn.setText("Pause");  
 isRunning = true;  
 }  
 }  
 });  
 finishBtn = findViewById(R.id.*finishBtn*);  
 finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 isRunning = false;  
 sensorManager.unregisterListener(listener);  
 *//exiting the running activity and sending data back to main program* finish();  
 }  
 });  
  
 *//HANDLING PERMISSIONS* requestPermissionLauncher =  
 registerForActivityResult(new ActivityResultContracts.RequestPermission(), isGranted -> {  
 if (isGranted) {  
 isRunning = Boolean.*TRUE*;  
 } else {  
 Toast.*makeText*(this, "Permission Denied", Toast.*LENGTH\_SHORT*).show();  
 isRunning = Boolean.*FALSE*;  
 this.finish();  
 }  
 });  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 PERMISSIONS = new String[]{  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 };  
 }  
 else{  
 PERMISSIONS = new String[]{  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 };  
 }  
 if (checkPermissions(this,PERMISSIONS) == Boolean.*FALSE*){  
 for (String permission: PERMISSIONS){  
 requestPermissionLauncher.launch(permission);  
 }  
 }  
 else{  
 startRunning();  
 }  
 }  
  
 @SuppressLint("MissingPermission")  
 private void startRunning() {  
 *//creating handler to run simultaneously to track duration in seconds* final Handler handler = new Handler();  
 handler.post(new Runnable() {  
 @SuppressLint("DefaultLocale")  
 @Override  
 public void run() {  
 handler.postDelayed(this, 1000);  
 if (isRunning) {  
 seconds++;  
 *//changing timer text view* int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 int secs = seconds % 60;  
 String time = String.*format*(Locale.*getDefault*(), "%d:%02d:%02d", hours, minutes, secs);  
 timerText.setText(time);  
  
 *//changing calorie text view* calories = Math.*round*(MET \* User.*getWeight*() \* (seconds.floatValue() / 3600));  
 calorieText.setText(String.*format*("Calories:\n%d", calories));  
  
 *//changing step text view* stepText.setText(String.*format*("Steps:\n%d", steps));  
  
 *//rather than using geolocation and routes, treadmill is in one location  
 //distance is calculated using the average stride based off their height\*0.4 to a good approximation  
 //calculating distance and changing distance text view* DecimalFormat df = new DecimalFormat("#.##");  
 distance = height.floatValue() \* ((float) Math.*floor*(steps/2)) \* 0.004f; *//0.004 as user height stored as cm* distText.setText(String.*format*("Distance:\n%sm",df.format(distance)));  
  
 *//calculating pace and changing pace text view* paceText.setText(Html.*fromHtml*("Pace:\n"+String.*valueOf*(df.format(distance/seconds.floatValue()))+"ms<sup>-1</sup"));  
 *//allowing preprocessing to happen at the instance of a 5 second interval* if ((seconds % 5) == 0) {  
 hasProcessed = Boolean.*FALSE*;  
 }  
  
 }  
  
 }  
 });  
  
  
 *//2d arrays to store a variable amount of samples, each sample consisting of the x y z values* ArrayList<Float[]> accel = new ArrayList<Float[]>();  
 ArrayList<Float[]> grav = new ArrayList<Float[]>();  
  
 listener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
  
 Sensor sensor = event.sensor;  
 *//formatting by rounding to 2 decimal places* DecimalFormat df = new DecimalFormat("#.##");  
 if (sensor.getType() == Sensor.*TYPE\_LINEAR\_ACCELERATION* & isRunning) {  
 *//handling the linear acceleration* Float x = Float.*parseFloat*(df.format(event.values[0]));  
 Float y = Float.*parseFloat*(df.format(event.values[1]));  
 Float z = Float.*parseFloat*(df.format(event.values[2]));  
 Float[] entry = new Float[3];  
 entry[0] = x;  
 entry[1] = y;  
 entry[2] = z;  
 System.*out*.println("acceleration:" + String.*format*("%f, %f, %f", entry[0], entry[1], entry[2]));  
 accel.add(entry);  
  
  
 }  
 if (sensor.getType() == Sensor.*TYPE\_GRAVITY* & isRunning) {  
 *//handling gravimeter* Float x = Float.*parseFloat*(df.format(event.values[0]));  
 Float y = Float.*parseFloat*(df.format(event.values[1]));  
 Float z = Float.*parseFloat*(df.format(event.values[2]));  
 Float[] entry = new Float[3];  
 entry[0] = x;  
 entry[1] = y;  
 entry[2] = z;  
 grav.add(entry);  
 System.*out*.println("gravity: " + String.*format*("%f, %f, %f", entry[0], entry[1], entry[2]));  
 }  
  
  
 *//PROCESSING DATA* if (((seconds % 5) == 0 && (grav.size() > 0)) && (accel.size() > 0) && (hasProcessed == Boolean.*FALSE*)) {  
 ArrayList<Float> results = new ArrayList<Float>();  
 *//PRE-PROCESSING DATA  
 //handling when grav array and accel array are unequal:* while (accel.size() != grav.size()) {  
 if (accel.size() > grav.size()) {  
 accel.remove(accel.size() - 1);  
  
 } else {  
 grav.remove(grav.size() - 1);  
 }  
 }  
  
 System.*out*.println("Seconds: " + seconds);  
 *//PERFORM DOT PRODUCT* System.*out*.println(String.*format*("gravsize: %d accelsize: %d", grav.size(), accel.size()));  
 for (int j = 0; j < grav.size(); j++) {  
 Float[] accelValues = accel.get(j);  
 Float[] gravValues = grav.get(j);  
 Float result = Float.*parseFloat*(df.format(gravValues[0] \* accelValues[0] + gravValues[1] \* accelValues[1] + gravValues[2] \* accelValues[2]));  
 if (result < 0) {  
 result = (float) Math.*sqrt*(0 - result);  
 result = 0 - result;  
 } else {  
 result = (float) Math.*sqrt*(result);  
 }  
 results.add(result);  
 System.*out*.println("result: " + j + " " + result.toString());  
 }  
 grav.clear();  
 accel.clear();  
 *//hasProcessed to true to prevent small chunks of data being processed* hasProcessed = Boolean.*TRUE*;  
  
 *//FILTERING DATA* filter.filter(results);  
 filtered\_data = filter.getFiltered\_data();  
 detector.detect(filtered\_data);  
 steps = detector.getStepCount();  
  
 }  
 }  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
 }  
 };  
 sensorManager.registerListener(listener, sensorManager.getDefaultSensor(Sensor.*TYPE\_LINEAR\_ACCELERATION*), SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 sensorManager.registerListener(listener, sensorManager.getDefaultSensor(Sensor.*TYPE\_GRAVITY*), SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
 *//PERMISSIONS* private boolean checkPermissions(Context context, String[] PERMISSIONS) {  
 *//CHECKING FOR EXISTING PERMISSIONS* if (context!=null && PERMISSIONS!=null){  
 for (String permission: PERMISSIONS){  
 if (ActivityCompat.*checkSelfPermission*(context,permission)!= PackageManager.*PERMISSION\_GRANTED*){  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
 *//handling live notification bar* public void sendOnChannel1(View v){  
 Notification notification = new NotificationCompat.Builder(this,*CHANNEL\_1\_ID*)  
 .setSmallIcon(R.id.*icon*)  
 .setContentTitle("Treadmill Tracking")  
 .setContentText(String.*valueOf*(steps))  
 .setCategory(NotificationCompat.*CATEGORY\_WORKOUT*)  
 .build();  
 notificationManagerCompat.notify(1,notification);  
 }  
}

PushUpActivity.java

package com.example.exercisetracker;  
  
import android.Manifest;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.CameraSelector;  
import androidx.camera.core.Preview;  
import androidx.camera.lifecycle.ProcessCameraProvider;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
  
import com.google.common.util.concurrent.ListenableFuture;  
import com.google.mlkit.vision.pose.PoseDetection;  
import com.google.mlkit.vision.pose.PoseDetector;  
import com.google.mlkit.vision.pose.defaults.PoseDetectorOptions;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Locale;  
  
public class PushUpActivity extends AppCompatActivity {  
 *//Text Views* private TextView timerText,repText,calText;  
 *//buttons* private Button startBtn, finishBtn;  
  
 *//pushup custom variables* private Boolean isTracking;  
 private Integer seconds;  
 private Date timeStarted;  
 private Float MET;  
  
 *//MLK variables* private PoseDetectorOptions options;  
 private PoseDetector poseDetector;  
 private ListenableFuture<ProcessCameraProvider> cameraProviderFuture;  
 private Preview preview;  
 private CameraSelector cameraSelector;  
  
  
 *//permissions* private String[] PERMISSIONS;  
 private ActivityResultLauncher<String> requestPermissionLauncher;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 *//visuals* getSupportActionBar().hide();  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* getWindow().setStatusBarColor(ContextCompat.*getColor*(this,R.color.*main\_colour*));*// set status background white* setContentView(R.layout.*activity\_pushup*);  
 *//instantiating all variables* isTracking = Boolean.*TRUE*;  
 startBtn = findViewById(R.id.*startStopBtn*);  
 finishBtn = findViewById(R.id.*finishBtn*);  
 timerText = findViewById(R.id.*timerText*);  
 repText = findViewById(R.id.*repText*);  
 calText = findViewById(R.id.*calText*);  
 MET = Float.*parseFloat*(getString(R.string.*met\_pushup*));  
 *//handling button clicks* startBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//when start stop button is clicked* if (isTracking) {  
 startBtn.setText("Resume");  
 isTracking = false;  
  
 } else {  
 startBtn.setText("Pause");  
 isTracking = true;  
 }  
 }  
 });  
 finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//when finish button is clicked* isTracking = false;  
 finish();  
 }  
 });  
  
 *//handling camera and write permissions* requestPermissionLauncher =  
 registerForActivityResult(new ActivityResultContracts.RequestPermission(), isGranted -> {  
 if (isGranted) {  
 isTracking = Boolean.*TRUE*;  
 } else {  
 Toast.*makeText*(this, "Permission Denied", Toast.*LENGTH\_SHORT*).show();  
 isTracking = Boolean.*FALSE*;  
 this.finish();  
 }  
 });  
 PERMISSIONS = new String[]{  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE* };  
  
 if (checkPermissions(this,PERMISSIONS) == Boolean.*FALSE*){  
 for (String permission: PERMISSIONS){  
 requestPermissionLauncher.launch(permission);  
 }  
 }  
 else{  
 startTracking();  
 }  
  
 }  
 private void startTracking(){  
 cameraProviderFuture = ProcessCameraProvider.*getInstance*(PushUpActivity.this);  
  
 seconds = 0;  
 timeStarted = Calendar.*getInstance*().getTime();  
 final Handler handler = new Handler();  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 handler.postDelayed(this,1000);  
 if (isTracking == Boolean.*TRUE*){  
 seconds++;  
 *//updating timer view* int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 int secs = seconds % 60;  
 String time = String.*format*(Locale.*getDefault*(), "%d:%02d:%02d", hours, minutes, secs);  
 timerText.setText(time);  
  
  
 }  
 }  
 });  
  
 *//MLK tracking* options = new PoseDetectorOptions.Builder()  
 .setDetectorMode(PoseDetectorOptions.*STREAM\_MODE*)  
 .build();  
 poseDetector = PoseDetection.*getClient*(options);  
  
 }  
  
 *//PERMISSIONS* private boolean checkPermissions(Context context, String[] PERMISSIONS) {  
 *//CHECKING FOR EXISTING PERMISSIONS* if (context!=null && PERMISSIONS!=null){  
 for (String permission: PERMISSIONS){  
 if (ActivityCompat.*checkSelfPermission*(context,permission)!= PackageManager.*PERMISSION\_GRANTED*){  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
}

## Custom Java Classes

To help the application handle data better, I have created custom java classes. Some are designed to be instantiated whereas others are purely static classes. These classes are: Route, Activity and User classes.

User Class

package com.example.exercisetracker;  
  
import java.util.Date;  
  
public class User {  
 private static Float *weight*;  
 private static Date *dateOfBirth*;  
 private static Integer *height*;  
  
 public static Integer getHeight() {  
 return *height*;  
 }  
  
 public static void setHeight(Integer height) {  
 User.*height* = height;  
 }  
  
 public static Float getWeight() {  
 return *weight*;  
 }  
  
 public static void setWeight(Float weight) {  
 User.*weight* = weight;  
 }  
  
 public static Date getDateOfBirth() {  
 return *dateOfBirth*;  
 }  
  
 public static void setDateOfBirth(Date dateOfBirth) {  
 User.*dateOfBirth* = dateOfBirth;  
 }  
}

Activity Class

package com.example.exercisetracker;  
  
import static android.content.Context.*MODE\_PRIVATE*;  
  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.util.Date;  
  
public class Activity extends AppCompatActivity {  
 private Date timeStarted;  
 private Integer duration;  
 private String type;  
 private Integer calories;  
 private Route route;  
 private Float distance;  
  
 public Activity(Date timeStarted, Integer duration, String type) {  
 *//An activity must at it's minimum have the time started, duration and exercise type name* this.timeStarted = timeStarted;  
 this.duration = duration;  
 this.type = type;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public Date getTimeStarted() {  
 return timeStarted;  
 }  
  
 public Integer getDuration() {  
 return duration;  
 }  
  
 public Integer getCalories() {  
 return calories;  
 }  
  
 public void setCalories(Integer calories) {  
 this.calories = calories;  
 }  
  
 public Route getRoute() {  
 return route;  
 }  
  
 public void setRoute(Route route) {  
 this.route = route;  
 }  
  
 public Float getDistance() {  
 return distance;  
 }  
  
 public void setDistance(Float distance) {  
 this.distance = distance;  
 }  
  
 public Boolean saveActivity(String fileDir){  
 *//saving activities under a .bin file holding the activity objects* ObjectOutputStream oos = null;  
 try {  
 File file = new File(fileDir, "activities.bin");  
 file.createNewFile();  
 FileOutputStream fos = this.openFileOutput("activities.bin", *MODE\_PRIVATE*);  
 oos = new ObjectOutputStream(fos);  
 oos.writeObject(this);  
 oos.close();  
 fos.close();  
 return true;  
 } catch (FileNotFoundException err) {  
 return false;  
 } catch (Exception abcd) {  
 return false;  
 }  
 finally {*//makes sure to close the ObjectOutputStream* if (oos != null) {  
 try {  
 oos.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 public static Activity readActivities(String filesDir){  
 ObjectInputStream ois = null;  
 try {  
 ois = new ObjectInputStream(new FileInputStream(new File(filesDir+"/activities.bin")));  
 Activity temp = (Activity)ois.readObject();  
 ois.close();  
 return temp;  
 } catch (IOException e) {  
 e.printStackTrace();  
 return null;  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 return null;  
 } finally {  
 if (ois != null) {  
 try {  
 ois.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

Route Class

package com.example.exercisetracker;  
  
import java.util.ArrayList;  
  
public class Route {  
 private ArrayList<Double[]> route;  
 private Double distance;  
 private Integer lastRouteIndex;  
  
 public Route(ArrayList<Double[]> route) {  
 this.route = route;  
 this.distance = 0d;  
 this.lastRouteIndex = 0;  
 }  
  
 public void calculateDistance(){  
 for (int i=lastRouteIndex;i<route.size()-1;i++){  
 *//implementing haversine formula to get distance* Double [] entry1 = route.get(i);  
 Double [] entry2 = route.get(i+1);  
 double latDistance = Math.*toRadians*(entry2[0] - entry1[0]);  
 double lonDistance = Math.*toRadians*(entry2[1] - entry1[1]);  
 double a = Math.*sin*(latDistance / 2) \* Math.*sin*(latDistance / 2)  
 + Math.*cos*(Math.*toRadians*(entry1[0])) \* Math.*cos*(Math.*toRadians*(entry2[0]))  
 \* Math.*sin*(lonDistance / 2) \* Math.*sin*(lonDistance / 2);  
 double c = 2 \* Math.*atan2*(Math.*sqrt*(a), Math.*sqrt*(1 - a));  
 lastRouteIndex++;  
 *//converting to metres, and multiplying by the radius of the Earth* distance += (6370 \* c \* 1000);  
 }  
 }  
  
 public Double getDistance() {  
 return distance;  
 }  
  
 public Integer getLastRouteIndex() {  
 return lastRouteIndex;  
 }  
 public ArrayList<Double[]> getRoute() {  
 return route;  
 }  
  
 public void addRoute(Double[] entry){  
 this.route.add(entry);  
 }  
  
 public int getRouteSize(){  
 return route.size();  
 }  
  
}

## Miscellaneous Files

These files detail the fundamental values and resources that the application will use.

Android Manifest

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.exercisetracker">  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"  
 android:maxSdkVersion = "28"/>  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION"/>  
 <uses-permission android:name="android.permission.INTERNET"/>  
 <uses-permission android:name="android.permission.FOREGROUND\_SERVICE"/>  
 <uses-permission android:name="android.permission.CAMERA"/>  
 <application  
 android:name=".BaseApp"  
 android:allowBackup="true"  
 android:icon="@mipmap/appicon\_foreground"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/appicon\_round"  
 android:supportsRtl="true"  
 android:requestLegacyExternalStorage="true"  
  
 android:theme="@style/Theme.ExerciseTracker">  
 <activity  
 android:name=".MainActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:screenOrientation="portrait"  
 android:name=".RunningActivity">  
 </activity>  
 <activity  
 android:name=".TreadmillActivity"  
 android:screenOrientation="portrait">  
 </activity>  
 <activity  
 android:name=".PushUpActivity"  
 android:screenOrientation="portrait">  
 </activity>  
 </application>  
  
</manifest>

colors.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<resources>  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="black">#FF000000</color>  
 <color name="grey">#BCBCBC</color>  
 <color name="white">#FFFFFFFF</color>  
 <color name="lightBlue">#617bff</color>  
 <color name="mainBlue">#4563ff</color>  
 <color name="red">#F44336</color>  
 <color name="yellow">#FFEB3B</color>  
 <color name="main\_colour">#FFDCA8</color>  
 <color name="main\_colour\_variant">#B3FFEED5</color>  
 <color name="main\_colour\_on\_primary">#6C4100</color>  
</resources>

strings.xml

<resources>  
 <string name="app\_name">Exercise Tracker</string>  
 <string name="start\_tracking">Start Tracking</string>  
 *<!-- MET VALUES -->* <string name="met\_running">7.0</string>  
 <string name="met\_treadmill">8.0</string>  
 <string name="met\_pushup">3.8</string>  
 <string name="met\_walking">3.5</string>  
</resources>

themes.xml

<resources xmlns:tools="http://schemas.android.com/tools">  
 *<!-- Base application theme. -->* <style name="Theme.ExerciseTracker" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 *<!-- Primary brand color. -->* <item name="colorPrimary">@color/main\_colour</item>  
 <item name="colorPrimaryVariant">#33FFDCA8</item>  
 <item name="colorOnPrimary">@color/main\_colour\_on\_primary</item>  
 *<!-- Secondary brand color. -->* <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_700</item>  
 <item name="colorOnSecondary">@color/black</item>  
 *<!-- Status bar color. -->* <item name="android:statusBarColor" tools:targetApi="l">@color/white</item>  
 <item name="android:windowLightStatusBar" tools:targetApi="m">true</item>  
 *<!-- Customize your theme here. -->* <item name="android:fontFamily">@font/gothic</item>  
 <item name="android:textColor">@color/black</item>  
 </style>  
</resources>

build.gradle

plugins **{** id 'com.android.application'  
**}**android **{** compileSdk 30  
  
 defaultConfig **{** applicationId "com.example.exercisetracker"  
 minSdk 21  
 targetSdk 30  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** release **{** minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 **}  
 }** compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_1\_8* targetCompatibility JavaVersion.*VERSION\_1\_8* **}  
}**dependencies **{** implementation 'androidx.camera:camera-camera2:1.0.2'  
 implementation 'androidx.camera:camera-lifecycle:1.0.2'  
 implementation "androidx.camera:camera-view:1.0.0-alpha27"  
 implementation 'androidx.fragment:fragment:1.3.6'  
 implementation 'androidx.appcompat:appcompat:1.3.1'  
 implementation 'com.google.android.material:material:1.4.0'  
 implementation 'com.google.mlkit:pose-detection:17.0.1-beta6'  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.1'  
 testImplementation 'junit:junit:4.+'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
**}**

## SQL Server

In order to host the database on the cloud, for multiple users to access, I have used freesqldatabase.com to do this, which gives me an account and database saved on the cloud. Also, phpMyAdmin gives a UI to view the database, whilst still giving my functionality to use SQL commands.

Will have to use PHP, XAAMP,

# TESTING

# EVALUATION

# APPENDIX

#### Joey’s Workout routine

Monday Db shoulder press Military press Lateral raise Reverse peck deck Leg press Pin squats

Tuesday Pull ups Lat pull down Seated row Deadlift Leg curl Lat pull overs

Wednesday Bench press Peck deck Squat Leg extension Adductors machine Ez bar bicep curls

Thursday Deadlift Military press Incline bench press Dumbell chest press Lateral raise Tricep extensions

Friday Bent over rows Lat pull down Lat pull overs Hip thrusts Abductors machine Preacher bench 21s

Saturday Rest day

Sunday Tricep extensions Overhead tricep extensions Seated bicep curls Skull crushers Preacher bench 21s Dumbell bicep curls

#### Scoped Storage

Scoped Storage (Android API 30+, Android 11+)

Every app has its internal storage, which only the app itself has access to and is privatised to the scope of the app only. All apps have access to external storage, however, which is where images, documents etc. can be stored. From Android 11 onwards, it was made mandatory that scoped storage should be used, where storage in internal and external storage are all linked together so that when uninstalling an app, all the data is deleted rather than just the data in the internal storage.

#### Results of different sampling rates when taking 20 steps

#### API prototyping

An API, stands for application programming interface, to allow a developer access to data without the need for a developer to necessarily know all how this data is formed, etc. API gives way to the abstraction of needing to program everything from scratch.

During my analysis of current fitness applications, Google Fit stood out to me, as they provided developers with their dedicated fitness API, specifically for app developers. As I have not dealt with APIs before, I wanted to learn more about APIs in general, so that I could test out their Fitness API so I could have a rough idea of how to approach developing a fitness app. Furthermore, the method of how Google detected steps and activities intrigued me, because of how easy it seemed for a developer to extract such information from the API.

<https://www.youtube.com/watch?v=GZvSYJDk-us&ab_channel=freeCodeCamp.org>

Graphical user interface, text

Description automatically generatedFirst use of API through the command line, sending a text message from a phone to another phone via SMS, using Twilio.

Graphical user interface, text, application, chat or text message

Description automatically generated

By using the curl code provided by Twilio, I was able to send a message from the Twilio phone to my phone, via Twilio’s SMS API, as a POST request.

Issues with regarding data structures and types when passing accelerometer data to Filter class. ArrayList class allows me to add elements dynamically, however, may be time inefficient, whereas using arrays of fixed size to store accelerometer data in 5 second increments will be easier to access. However, if no data is changed in a 5 second interval, creates unnecessary array. After some testing and comparing between using ArrayList and an array, I decided to use an ArrayList, since the quantities of values in a 5 second interval would only be around 10-20 values, so can still be accessed in a relatively quick amount of time.

//handling accelerometer  
ArrayList<Float> temp = new ArrayList<>();  
accelerometerEventListener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 DecimalFormat df = new DecimalFormat("#.####");  
 float x = event.values[0];  
 float y = event.values[1];  
 float z = event.values[2];  
 float mag = (float) Math.*sqrt*(x\*x + y\*y + z\*z);  
 //for every 5 seconds, filter the data and pass through detector  
 if ((seconds % 5) == 0) {  
 filter.filter(temp);  
 filtered\_data = filter.getFiltered\_data();  
 for (int i=0;i<filtered\_data.length;i++){  
 System.*out*.println(filtered\_data[i].toString());  
 }  
 //insert code here to handle detection of steps  
 //clearing temp for next sequences of values.  
 temp.clear();  
 } else {  
 temp.add(mag);  
 }  
 }

Filter Class

public void filter(ArrayList<Float> data){  
 filtered\_data = new Float[data.size()];  
 for (int i=0 ;i<data.size();i++){  
 if (data.get(i)<minThreshold){  
 filtered\_data[i] = (float) 0;  
 }  
 if (data.get(i)>maxThreshold){  
 filtered\_data[i] = maxThreshold;  
 }  
 if ((data.get(i)>=minThreshold) & (data.get(i)<=maxThreshold)){  
 filtered\_data[i] = data.get(i);  
 }  
 }  
 setFiltered\_data(filtered\_data);  
}

# REFERENCES

|  |  |
| --- | --- |
| [1] | “Association between physical exercise and mental health in 1·2 million individuals in the USA between 2011 and 2015: a cross-sectional study,” [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/30099000/. [Accessed 12 August 2021]. |
| [2] | Office for National Statistics, “Leading causes of death, UK: 2001 to 2018,” [Online]. Available: https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/causesofdeath/articles/leadingcausesofdeathuk/2001to2018. [Accessed 30 July 2021]. |
| [3] | GOV.UK, “Physical Activity,” [Online]. Available: https://www.ethnicity-facts-figures.service.gov.uk/health/diet-and-exercise/physical-activity/latest. [Accessed 30 July 2021]. |
| [4] | “staista,” [Online]. Available: https://www.statista.com/statistics/956297/ownership-of-smartphones-uk/. [Accessed 30 July 2021]. |
| [5] | Phys Org, “Samsung Introduces World's First '3-dimensional Movement Recognition Phone',” [Online]. Available: https://phys.org/news/2005-01-samsung-world-dimensional-movement-recognition.html#:~:text=0-,Samsung%20Introduces%20World's%20First%20'3%2Ddimensional%20Movement%20Recognition'%20Phone,%22%20mobile%20phone%20SCH%2DS310.. [Accessed 23 August 2021]. |
| [6] | cnet, “Motion Sensing Comes To Mobile Phones,” [Online]. Available: https://www.cnet.com/tech/mobile/motion-sensing-comes-to-mobile-phones/. [Accessed 23 August 2021]. |
| [7] | Android, “Sensor Types,” [Online]. Available: https://source.android.com/devices/sensors/sensor-types. [Accessed 24 August 2021]. |
| [8] | ifixit, “iPhone 4 Gyroscope Teardown,” [Online]. Available: https://www.ifixit.com/Teardown/iPhone+4+Gyroscope+Teardown/3156. [Accessed 23 August 2021]. |
| [9] | Gizmodo, “All the sensors in your phone and how they work,” [Online]. Available: https://gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002. [Accessed 24 August 2021]. |
| [10] | Britanica, “Magnetometer,” [Online]. Available: https://www.britannica.com/technology/magnetometer. [Accessed 24 August 2021]. |
| [11] | Techspot, “Know Your Smartphone: A Guide To Camera Hardware,” [Online]. Available: https://www.techspot.com/guides/850-smartphone-camera-hardware/. [Accessed 24 August 2021]. |
| [12] | Techopedia, “Six Degrees of Freedom,” [Online]. Available: https://www.techopedia.com/definition/12702/six-degrees-of-freedom-6dof. [Accessed 25 August 2021]. |
| [13] | aosabork.org, “A Pedometer In the Real World,” [Online]. Available: https://www.aosabook.org/en/500L/a-pedometer-in-the-real-world.html. [Accessed 14 09 2021]. |
| [14] | Electronics Tutorials, “Passive Low Pass Filter,” [Online]. Available: https://www.electronics-tutorials.ws/filter/filter\_2.html. [Accessed 29 09 2021]. |
| [15] | SCIENTIFIC AMERICAN, a Division of Springer Nature America, Inc., “How Do Food Manufacturers Calculate the Calorie Count of Packaged Foods?,” Scientific American, [Online]. Available: https://www.scientificamerican.com/article/how-do-food-manufacturers/. [Accessed 12 October 2021]. |
| [16] | CDC, “NCHS Data Brief, Number 359, December 2020,” [Online]. Available: https://www.cdc.gov/nchs/data/databriefs/db395-H.pdf. [Accessed 12 August 2021]. |

GitHub repository:

<https://github.com/calebchan1/ALevelNEA>