COMPUTER SCIENCE AQA A-LEVEL NEA

EXERCISE DETECTION PROJECT

CONTENTS

[1 ANALYSIS 4](#_Toc96799248)

[1.1 Introduction 4](#_Toc96799249)

[1.2 Sensors Overview 5](#_Toc96799250)

[1.3 Exercises and Activities 11](#_Toc96799251)

[1.4 Current Market and Applications 12](#_Toc96799252)

[1.5 User Research 14](#_Toc96799253)

[1.6 Prototyping and Spiking 16](#_Toc96799254)

[1.7 Objectives and the Critical Path 20](#_Toc96799255)

[2 DESIGN 23](#_Toc96799256)

[2.1 High Level Overview 23](#_Toc96799257)

[2.2 Pedometer Algorithm 25](#_Toc96799258)

[2.3 GPS Tracking 31](#_Toc96799259)

[2.4 Computer Vision and ML Kit Library 32](#_Toc96799260)

[2.5 User Defined Classes 34](#_Toc96799261)

[2.6 Database Design 35](#_Toc96799262)

[2.7 Data Flow Diagram 42](#_Toc96799263)

[2.8 User Interface Design 43](#_Toc96799264)

[3 TECHNICAL SOLUTION 45](#_Toc96799265)

[3.1 Issues and Logs: 45](#_Toc96799266)

[3.2 Object Orientated Programming 50](#_Toc96799267)

[3.2.1 Activities and Fragments: 50](#_Toc96799268)

[3.2.2 User-Defined Classes: 50](#_Toc96799269)

[3.2.3 Class Associations: 51](#_Toc96799270)

[3.2.4 Methods and Attributes: 51](#_Toc96799271)

[3.3 Database Model 53](#_Toc96799272)

[3.4 Mathematical Operations 57](#_Toc96799273)

[3.5 APIs and Services 62](#_Toc96799274)

[3.6 User Interface and Design 65](#_Toc96799275)

[3.7 Program Code 66](#_Toc96799276)

[3.7.1 Activity XML Files 66](#_Toc96799277)

[3.7.2 Fragment XML Files: 93](#_Toc96799278)

[3.7.3 Fragments and Activities 104](#_Toc96799279)

[3.7.4 Custom Java Classes 198](#_Toc96799280)

[3.7.5 MSSQL Database & DBhelper.java 218](#_Toc96799281)

[3.7.6 Miscellaneous Files 233](#_Toc96799282)

[4 TESTING 244](#_Toc96799283)

[4.1 Key Objectives Tests: 244](#_Toc96799284)

[4.2 Boundary and Erroneous Tests: 246](#_Toc96799285)

[5 EVALUATION 249](#_Toc96799286)

[5.1 Critical Path 249](#_Toc96799287)

[5.2 Third-Party Feedback 249](#_Toc96799288)

[5.3 Changes and Improvements 250](#_Toc96799289)

[5.4 GitHub Repository 250](#_Toc96799290)

[5.5 Conclusion 251](#_Toc96799291)

[6 APPENDIX 252](#_Toc96799292)

[7 REFERENCES 260](#_Toc96799293)

# 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. |

## User Research

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. 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 APIs.

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. **MSSQL, JDBC and JDTS connector**

Used to set up a remote database and allow data to be transferred between devices and a main server hosting the database. JDBC 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 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.

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.

Pose Detection API:

Here is a list of all the landmarks provided by the pose detector:

Landmarks https://developers.google.com/ml-kit/vision/pose-detection

My main focus for tracking the reps of a push up exercise will be one/two landmarks from each region of the body. I.e. the nose, hips and knees. The detection algorithm will utilize the x,y and z values of each landmark given by ML kit.

These details how I plan on creating an algorithm to detect the number of reps:

* Firstly, check if the user is in a push up position. This is done by the Z value, which refers to how far into the image, a user’s body part is. (Gives 3-D perception from an image). If the lower body is further away from the upper body, which is also further away from the nose, we can assume that the user is in a push up position.
* Whilst the user is in this push up position, we must then record the first initial point where the user begins to push down.
* To detect if a user is pushing down, we use the y values from the landmarks to see if multiple landmarks move downwards by a large margin.
* Once the user has pushed down, we can now wait to detect if the user returns to the original starting position we initially recorded.
* Once the user reaches the original position, we can increment the rep count, and the cycle repeats.
* At any time within the cycle, if the user no longer is within a push up position, the algorithm would have to start again and any records within that rep are destroyed.

There are some limitations for these algorithms:

* Algorithm relies on the fact that the camera is positioned vertically, in that the values of y increase positively down the image
* Algorithm relies on the user’s device to have minimum processing power for ML kit to run
* Algorithm relies on the user’s body to be fully in frame for the duration of the exercise, in order for landmarks to be compared

Furthermore, rather than simply analyse the variation of these landmarks to count the number of repetitions of an exercise, I thought it would also be helpful for user’s to see the live tracking in terms of an overlay on their camera feed, to show how the app can track the exercise.

Official Google Documentation:

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

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

**Accessing Camera Sensor:**

The CameraX library by Google allows for an easy way to initiate and access the user’s camera, without having to deal with the complexities of differing camera sensors, pixel depth, resolution etc. By using this library, I can ensure that the app will work for all Android devices, no matter the type of camera sensor. Furthermore, because it is developed by Google, it is easily implemented to the pose detection API, as one of its use cases.

## 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.

Route – way of storing latitude and longitude coordinates in a single object

User – static class to allow access to user details and info

RepCounter – Handling rep counting for push up detection from landmarks

Detector – Detecting steps from filtered data

Filter – Used to filter data from unfiltered data

StepCounter – Handling step counting, composed of Detector and Filter

DBhelper – Handling requests to database

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 Users(

UserID int NOT NULL Identity(1,1),

username varchar(50) NOT NULL UNIQUE,

password varchar(50) NOT NULL,

salt varchar(50) 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(

User1ID int NOT NULL,

User2ID int NOT NULL,

PRIMARY KEY(User1ID, User2ID),

FOREIGN KEY (User1ID) REFERENCES Users(UserID) ON DELETE CASCADE,

FOREIGN KEY (User2ID) REFERENCES Users(UserID) ON DELETE CASCADE

);

CREATE TABLE Activity(

ActivityID int NOT NULL IDENTITY(1,1),

ExerciseID int NOT NULL,

UserID int NOT NULL,

Date DATE NOT NULL,

timeStarted time NOT NULL,

duration int NOT NULL,

calories int NOT NULL,

steps int,

distance float,

reps int,

PRIMARY KEY (ActivityID),

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE,

FOREIGN KEY (ExerciseID) REFERENCES Exercise(ExerciseID) ON DELETE SET NULL

);

CREATE TABLE Exercise(

ExerciseID int NOT NULL IDENTITY(1,1),

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 IDENTITY(1,1),

Name varchar(255) NOT NULL,

PRIMARY KEY (TypeID)

);

CREATE TABLE Route(

RouteID int NOT NULL IDENTITY(1,1),

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 Diagram

Here is a data flow diagram enlisting how the app should send and receive data. When a system is involved, this refers to the actual Android Activity class which will initiate the respective data connections and handles the processing of the data. Although this diagram does not show all the entities involved and all the possible points at which data flows in my real application, this diagram will help me design the basic structure of how and where the data flows.



## 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:**

Users 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:

In 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 Diagram

Description automatically generatedbreaking functionality.

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.



Initial Permissions Flow

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.

ML Kit:

In drawing a graphical tracking of the body of the user in a push up, scaling would need to be considered. I had many issues with this, for plotting the points on a surface on Android was much harder than anticipated, due to status and navigation bar heights varying etc. Although this would not affect the actual push up detection algorithm, it did affect the graphical overlay, as some devices there was a clear offset. Furthermore, detection relies on the travel in x and y direction of the user on a phone’s screen, which changes between device to device.

Using ratio of screen’s display size, we can scale down an image sent to ML Kit to the same dimensions for all device

freesqldatabase.com issues

With the aspect of having a mysql server easily accessible via freesqldatabase.com, and the easy set up, it comes with it’s cost. During development, there was a period of time where the host of my database completely shut down, all my tables and data was not accessible for this lengthy period of time.

As a result, I have had to find other cloud hosting sql database services. This led me to transfer entirely to Microsoft Azure services. Although much more complicated to set up than using freesqldatabase.com, this was much more secure in both reliability and security. The only withdrawal is that I had limited credit, whereas freesqldatabase was entirely free.

Unused classes

The following classes I coded but were not used for the final technical solution:

Service.java – Provide services to communicate between the app and the phone whilst the app was not running in the foreground. Due to time factor of project and complexity of services in Android, I was unable to implement this

BaseApp.java -wraps around the whole project, was intended for notifications and services, but was not implemented due to the same reason as mentioned for Service.java.

Storing passwords on database

After receiving initial feedback from my users, one of them who was a computer scientist student suggested to use a hash and salt algorithm to store passwords, because I mentioned how I could see their passwords. This was implemented at a late stage to the project, but researching I used <https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/> as a guide.

Things to implement:

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

## Object Orientated Programming

### Activities and Fragments:

As mentioned previously, my Android application uses Java, which is an object orientated language. As a result, everything is built on the OOP paradigm, including the “Activities” and “fragments”, which handle the logic.

An “Activity” in an Android application, extends (inherits) the AppCompatActivity class, which provides the window in which the app draws on the UI. Within an activity class, you can also provide custom functionality on how the app should respond to a user interaction. Similarly, a class which inherits the base class “Fragment”, represents a reusable portion of your app's UI. A fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events. However, a fragment cannot exist without the activity class hosting it. [17] My usage of fragments is to easily switch the UI, by means of a main navigation menu.

It is upon these activity and fragment classes, where my app’s functionality lies. The following java classes are my activity classes: MainActivity.java, LogInScreen.java, RegisterUserActivity.java, RunningActivity.java, TreadmillActivity.java, PushUpActivity.java, SquatsActivity.java and AddFriendsActivity.java.

The following java classes are my fragment classes: ExerciseFragment.java, HistoryFragment.java, FriendsFragment.java, SettingsFragment.java.

The code for the activities and fragments can be found at 3.6.3

### User-Defined Classes:

As well as the underlying activity and fragment classes, I have had to create various other custom defined classes to aide in the app’s functionality. The names of these classes are: Activity.java, Route.java, DBhelper.java, User.java, Detector.java, Filter.java, Stepcounter.java, RepCounter.java, Friend.java

Activity – Not to be confused with Android’s activity class. Used as a way to easily store information about an activity which has been saved on the database. Acts to easily manipulate information received from database in the form of a single object

Route – way of storing latitude and longitude coordinates in a single object, along with its manipulations and distance calculation

User – static class to allow access to user details and information

RepCounter – Handling rep counting for push up and squat detection from landmarks processed by ML Kit.

Detector – Detecting steps from filtered data

Filter – Used to filter data from unfiltered data

StepCounter – Handling step counting, composed of Detector and Filter

DBhelper – Handling requests to database

Friend – Simple object to hold information about users other than the current user

### Class Associations:

Examples of aggregation:

StepCounter associated with classes Detector and Filter. However, Detector and Filter class can exist without the StepCounter class. This can be seen at 3.7.4.4

Examples of composition:

The following classes cannot exist without the containing class, because of the specificality of those classes to the containing class.

GraphicOverlay in PushUpActivity.java and SquatsActivity.java, used to draw graphic overlay over the feed from the camera to highlight how the app is tracking the movements of the body. This can be seen at 3.7.1.6 and 3.7.1.7

RecyclerViewAdapter classes in HistoryFragment.java and AddFriendsActivity.java used to convert information from the database to multiple instances of cards held in a recycler view to the user. This can be seen at 3.7.3.4 and 3.7.3.13

Task classes found in History, Leaderboard and Settings fragments, classes which inherit from Asynctask to allow parallel processing to retrieve data from the Azure SQL database. This can be seen at 3.7.3.4, 3.7.3.3 and 3.7.3.2.

### Methods and Attributes:

Here is an explanation of the main methods and attributes in my code. I have not included a description of every single method used, as some do not need explaining and I have already included the comments within the code, which have a red font and contain //

**OnCreateView():**

All classes that inherit themselves from an Android class, such as fragments or activities, must override the principal method “OnCreateView()”. This method is the starting point of that activity/fragment. It acts almost like a constructor of that class, but rather is what occurs when that activity/fragment is first instantiated to a View. This is the point at which anything that needs to be done at the start of the program is instantiated and declared, such as different attributes and objects.

**init():**

For all exercise activities, I have further separated the instantiation of variables to the method init(). This serves only to instantiate values to variables and the creation of objects, such as the seconds, calories, steps, repcounter objects etc.

**createTimer()**

Moreover, the method createTimer() is found in all exercises, which serves to create a Handler Thread to keep track of time, incrementing the seconds every 1000 milliseconds. It is in this method also, that any changes/updates that need to occur are done, i.e., updating the text view, reading out quotes every 60 seconds.

**finishTracking(), finishRunning()**

To close an Android fragment/activity, the finish() function is used to close that instance of that class. This deals with safely closing all processes and views associated to the instance of that activity/fragment. However simply calling the finish() function when finishing an exercise is not enough. This is because I need to save the data of that exercise to the database, as well as close any sensor/service listeners safely, without causing a crash. Thus, the finishTracking()/finishRunnnig() methods are used to safely close the activity.

## Database Model

In order to host the database on the cloud, for multiple users to access, I originally used freesqldatabase.com to do this, which gives me an account and database saved on the cloud along with a free SQL server to handle requests. However, during the development of the project, freesqldatabase servers shut down and would not respond to any requests. As a result, I moved to using Azure services to hold and store a SQL database and server for the project. The limitations of this were my limited credit as a student, so I could only keep it running during the development and testing phase of the project.

To connect to this database, I have used the JDBC connector library and the JDTS library, which provides a library in Java to run and execute SQL commands to a database [17]. (As part of the app, this is installed as an external library).

SQL Statements:

Adding pre-set Exercise entities:

INSERT INTO `Exercise`(`Name`, `TypeID`, `metValue`)

VALUES ("running", (SELECT ExerciseType.TypeID FROM ExerciseType WHERE ExerciseType.Name = 'cardiovascular'),'7.0');

INSERT INTO `Exercise`(`Name`, `TypeID`, `metValue`)

VALUES ("walking", (SELECT ExerciseType.TypeID FROM ExerciseType WHERE ExerciseType.Name = 'cardiovascular'),'7.0');

INSERT INTO `Exercise`(`Name`, `TypeID`, `metValue`)

VALUES ("treadmill", (SELECT ExerciseType.TypeID FROM ExerciseType WHERE ExerciseType.Name = 'cardiovascular'),'3.5');

INSERT INTO `Exercise`(`Name`, `TypeID`, `metValue`)

VALUES ("pushup", (SELECT ExerciseType.TypeID FROM ExerciseType WHERE ExerciseType.Name = 'conditioning'),'8.0');

VALUES ("squats", (SELECT ExerciseType.TypeID FROM ExerciseType WHERE ExerciseType.Name = 'conditioning'),'3.5’);

Adding pre-set ExerciseType entities:

INSERT INTO `ExerciseType`(`Name`) VALUES (‘conditioning’);

INSERT INTO `ExerciseType`(`Name`) VALUES (‘cardiovasuclar);

INSERT INTO `ExerciseType`(`Name`) VALUES (‘sports);

Graphical user interface, text, website

Description automatically generateddbhelper Java Class

Entity Relationship Diagram for my database

To have a centralised place where all database interactions is vitally important, especially as it means less replication of code in lots of different places. Thus, I have added a DBhelper class which manages the connection to the database, with methods that provide the results of a certain query. The full code for the dbhelper.java class is found in 3.7.5

For storing passwords, I decided implemented a simple hash and salt algorithm, specifically using the MD5 algorithm. The extra salt was required due to the limitations of MD5 being able to brute force, so the salt adds a layer of security and more randomness to the hashed passwords, to prevent people who have the same passwords from having the same hash.

The following code in the dbhelper.java class (at 3.7.5) allowed me to do so, creating and retrieving salts when necessary, and checking hashed values with the hashed values in the database. I used this guide to help me implement a hash and salt algorithm: <https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/>

*/\*\*  
 \* Following methods dealing with hashing a secure password  
 \* https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/  
 \*/*private static String getSecurePassword(String passwordToHash, String salt) {  
 String generatedPassword = null;  
 try {  
 *// Create MessageDigest instance for MD5* MessageDigest md = MessageDigest.*getInstance*("MD5");  
 *// Add password bytes to digest* md.update(salt.getBytes());  
 *// Get the hash's bytes* byte[] bytes = md.digest(passwordToHash.getBytes());  
 *// This bytes[] has bytes in decimal format;  
 // Convert it to hexadecimal format* StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < bytes.length; i++) {  
 sb.append(Integer.*toString*((bytes[i] & 0xff) + 0x100, 16)  
 .substring(1));  
 }  
 *// Get complete hashed password in hex format* generatedPassword = sb.toString();  
 } catch (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
 return generatedPassword;  
}  
  
  
private static String createSalt() throws NoSuchAlgorithmException, NoSuchProviderException{  
 *//Creating a salt for a new user  
 // SecureRandom generator* SecureRandom sr = SecureRandom.*getInstance*("SHA1PRNG", "AndroidOpenSSL");  
 *// Create array for salt* byte[] salt = new byte[16];  
 *// Get a random salt* sr.nextBytes(salt);  
 *// return salt* return salt.toString();  
}  
  
private String getSalt(String username){  
 *//retrieving salt from db of existing user* Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 resultset = statement.executeQuery(  
 "SELECT salt FROM Users " +  
 String.*format*("WHERE username = '%s'", username)  
 );  
 resultset.next();  
 String salt = resultset.getString(1);  
 if (salt.isEmpty()){  
 return null;  
 }  
 return salt;  
 }  
 catch (Exception e) {  
 return null;  
 } finally {  
 closeConnection(conn);  
 }  
}

## Mathematical Operations

As mentioned in my design section, in order to create an effective tracking application, I would need to implement various algorithms to do so, especially with the multiple sets of data that I would need to manipulate.

Running Tracking:

Text

Description automatically generatedBy using the Android’s in built acceleration sensors and gravity sensors, I can use the dot product to get me the component of the user acceleration in the line of gravity, as a user moves up and down (a step).

Text, letter

Description automatically generated

The following code shows how I have done this and is implemented by the StepCounter.java class which encapsulates all the methods required to calculate steps. The full code can be found at 3.7.4.4

private ArrayList<Float> processData() {  
 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);  
 }  
 }  
  
 *//PERFORM DOT PRODUCT* 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]));  
 result = result / 9.81f;  
 results.add(result);  
 System.*out*.println("result: " + j + " " + result.toString());  
 }  
 return results;  
}

GPS Tracking:

Here is the implementation of the Haversine formula to calculate the distance between two points on the Earth in the format of longitude and latitude. The full code can be seen at 3.7.1 and 3.7.4 in the RunningActivity and Route classes.

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);  
 }  
}

Push-Up Detection:

My push-up detection utilizes many comparisons between the positioning of landmarks, causing for many different Boolean statements.

private void detectPushUpReps(Map<Integer, PointF3D> relevantLandmarks) {  
 *//detecting push up reps* if (duration == 0) {  
 *//first time entering pose* startPoint = relevantLandmarks;  
 }  
 *//calculation of reps is done in a process:  
 //when user first initially enters pose, the position is recorded  
 //user has to then push down by at least x amount and return to original position for a rep to be counted* if (countedRep) {  
 *//instance of the rep was already counted previously, thus all boolean variables must reset* reset();  
 } else {  
 if (!pushedDown) {  
 *//checking to see if user has pushed down by tracking movement of nose* pushedDown = Objects.*requireNonNull*(relevantLandmarks.get(PoseLandmark.*NOSE*)).getY() >= Objects.*requireNonNull*(startPoint.get(PoseLandmark.*NOSE*)).getY() + minDistance;  
 } else {  
 *//user has pushed down, thus checking to see if they have returned to position by tracking movement of nose* if (!returnedToPosition) {  
 returnedToPosition = Objects.*requireNonNull*(relevantLandmarks.get(PoseLandmark.*NOSE*)).getY() < Objects.*requireNonNull*(startPoint.get(PoseLandmark.*NOSE*)).getY();  
 }  
 }  
 if (pushedDown && returnedToPosition && !countedRep) {  
 *//if that instance of rep was not counted yet* Toast.*makeText*(this.context, "Rep Counted", Toast.*LENGTH\_SHORT*).show();  
 reps++;  
 countedRep = true;  
 } else {  
 duration++;  
 }  
 }  
}

Sqauts Detection:

My push-up detection utilizes many comparisons between the positioning of landmarks, causing for many different Boolean statements.

private void detectSquatReps(Map<Integer, PointF3D> relevantLandmarks) {  
 *//detecting squat reps  
 //calculation of reps is done in a process:  
 //when user first initially enters pose, the position is recorded  
 //user has to then move in the z direction by a min amount and return to original position for a rep to be counted* if (duration == 0) {  
 *//first time entering pose* startPoint = relevantLandmarks;  
 }  
 if (countedRep) {  
 reset();  
 } else {  
 if (!pushedDown) {  
 *//checking to see if user has squatted down by tracking movement of difference of z distance  
 //between the knees and the hips* pushedDown = Objects.*requireNonNull*(relevantLandmarks.get(PoseLandmark.*NOSE*)).getY() > Objects.*requireNonNull*(startPoint.get(PoseLandmark.*NOSE*)).getY() + minDistance;  
 } else {  
 *//user has pushed down, thus checking to see if they have returned to position by tracking movement of nose/shoulder* if (!returnedToPosition) {  
 returnedToPosition = Objects.*requireNonNull*(relevantLandmarks.get(PoseLandmark.*NOSE*)).getY() < Objects.*requireNonNull*(startPoint.get(PoseLandmark.*NOSE*)).getY() + minDistance/2f  
 || Objects.*requireNonNull*(relevantLandmarks.get(PoseLandmark.*LEFT\_SHOULDER*)).getY() < Objects.*requireNonNull*(startPoint.get(PoseLandmark.*LEFT\_SHOULDER*)).getY() + minDistance/2f  
 ;  
 }  
 }  
 if (pushedDown & returnedToPosition & !countedRep) {  
 *//if that instance of rep was not counted yet* Toast.*makeText*(this.context, "Rep Counted", Toast.*LENGTH\_SHORT*).show();  
 reps++;  
 countedRep = true;  
 }  
 else{  
 duration ++;  
 }  
  
 }  
}

## APIs and Services

Pose Detection API (Part of ML Kit by Google)

Here is the implementation of the Pose Detection API by google, which is used by my PushupActivity class in 3.7.3, to track the number of reps of a push up done by a user. The idea is the app sends an image to the Pose Detection API at regular intervals, and the API returns back data about the image in terms of the location of body parts (landmarks).

private void handleCamera() {  
 final ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.*getInstance*(this);  
 *//getting display size (dependent on device)* ImageAnalysis imageAnalysis =  
 new ImageAnalysis.Builder()  
 .setTargetResolution(displaySize)  
 .setBackpressureStrategy(ImageAnalysis.*STRATEGY\_KEEP\_ONLY\_LATEST*)  
 .build();  
 *//setting the configuration for the image analysis* imageAnalysis.setAnalyzer(ContextCompat.*getMainExecutor*(PushUpActivity.this), new ImageAnalysis.Analyzer() {  
 @Override  
 public void analyze(@NonNull ImageProxy imageProxy) {  
  
 int rotationDegrees = imageProxy.getImageInfo().getRotationDegrees();  
 @SuppressLint("UnsafeOptInUsageError") Image image = imageProxy.getImage();  
 if (image != null) {  
 *//receiving the input image from camera* InputImage inputimage = InputImage.*fromMediaImage*(image, rotationDegrees);  
 Task<Pose> result = poseDetector.process(inputimage).addOnSuccessListener(new OnSuccessListener<Pose>() {  
 @Override  
 public void onSuccess(@NonNull Pose pose) {  
 if (isTracking) {  
 *//when the pose detector successfully can attach to image  
 //Receiving and processing landmarks from Google's ML kit software* List<PoseLandmark> allPoseLandmarks = pose.getAllPoseLandmarks();  
 processLandmarks(allPoseLandmarks);  
 *//drawing on the landmarks onto the user's screen* graphic.drawGraphic(allPoseLandmarks);  
 } else {  
 graphic.clearGraphic();  
 }  
  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 *//when the pose detector cannot attach to image* System.*out*.println("Failed Pose Detection");  
 }  
 }).addOnCompleteListener(new OnCompleteListener<Pose>() {  
 @Override  
 public void onComplete(@NonNull Task<Pose> task) {  
 *//making sure to close the instance of the image to allow the next image to be processed* imageProxy.close();  
 }  
 });  
 }  
 }  
 });  
  
 *//attaching the image analysis object to the camera* cameraProviderFuture.addListener(() -> {  
 try {  
 *//configuring camera to preview to display on the user's screen* ProcessCameraProvider provider = cameraProviderFuture.get();  
 preview = new Preview.Builder().build();  
 preview.setSurfaceProvider(tv.getSurfaceProvider());  
 cameraSelector = new CameraSelector.Builder()  
 .requireLensFacing(CameraSelector.*LENS\_FACING\_FRONT*)  
 .build();  
 try {  
 *//binding the camera, preview and analyser together* provider.unbindAll();  
 provider.bindToLifecycle((LifecycleOwner) this, cameraSelector, preview, imageAnalysis);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 } catch (ExecutionException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }, ContextCompat.*getMainExecutor*(this));  
 *//creating graphic object to deal with drawing on landmarks onto the preview* graphic = new Graphic(displaySize);  
  
}

## User Interface 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.

These XML files detail the layout and design of all the Android activities and the Android fragments.

The full code can be found at 3.7.1 and 3.7.2

Here are some of the things I implemented to make the application user friendly:

* Created dialogs to confirm any user actions, i.e., deleting an activity, updating the user details, logging out
* Created a remember me option when logging in, to allow the user to remain logged in even when the app is closed, prevents user from having to input details each time
* Tried to keep all interaction towards the bottom of the screen where possible
* Added help buttons to push up and squat exercises, explaining how the exercise works, especially important as it uses the camera which may be sensitive to the user.
* Rounded buttons, conforming to Material Design give a more modern feel to the user, as well as choosing a warm pastel colour, which is considered neutral to users.
* Made use of Async Tasks (Android concurrency) to load data from database in a separate thread from the main thread. Whilst loading, a loading animation would appear on the screen till the data has been fully loaded
* Toasts (Android component) used whenever encountering issues/errors, as well as showing confirmation of saving data.
* Disclaimers shown to users as text on screen when no data was received from the database. Used when showing the leader board and activity history.

<https://youtu.be/-qbyFzP9rsw>

## Program Code

### Activity XML Files

#### 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:id="@+id/parent"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 tools:context=".activities.MainActivity">  
  
  
 <FrameLayout  
 android:id="@+id/fragmentContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/bottom\_navigation">  
  
 <ImageView  
 android:id="@+id/refresh"  
 android:layout\_width="50dp"  
 android:layout\_height="50dp"  
 android:layout\_gravity="center"  
 android:src="@drawable/refresh"  
 android:visibility="invisible" />  
  
 </FrameLayout>  
  
 <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>

#### 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">  
  
 <ImageView  
 android:layout\_width="180dp"  
 android:layout\_height="180dp"  
 android:alpha="0.2"  
 android:src="@drawable/running"  
 app:layout\_constraintBottom\_toTopOf="@+id/startStopBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/row2" />  
  
 <TextView  
 android:id="@+id/textView6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="60dp"  
 android:fontFamily="@font/gothic"  
 android:text="Running Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="30sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="150dp"  
 android:fontFamily="@font/gothicbb"  
 android:gravity="center"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="80sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/stepText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Steps:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/calText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Calories:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/distText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Distance:\n0m"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/row2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/paceText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Pace:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 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="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:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/main\_colour"  
 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" />  
  
 <ImageButton  
 android:id="@+id/audioBtn"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_margin="20dp"  
 android:background="@color/transparent"  
 android:scaleType="fitCenter"  
 android:src="@drawable/audio"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### 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"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/main\_colour">  
  
 <ImageView  
 android:layout\_width="180dp"  
 android:layout\_height="180dp"  
 android:alpha="0.2"  
 android:src="@drawable/treadmill"  
 app:layout\_constraintBottom\_toTopOf="@+id/startStopBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/row2" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="60dp"  
 android:fontFamily="@font/gothic"  
 android:text="Treadmill Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="30sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="150dp"  
 android:fontFamily="@font/gothicbb"  
 android:gravity="center"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="80sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/stepText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Steps:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/calText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Calories:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/distText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Distance:\n0m"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/row2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/paceText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Pace:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 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="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:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/main\_colour"  
 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" />  
  
 <ImageButton  
 android:id="@+id/audioBtn"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:background="@color/transparent"  
 android:scaleType="fitCenter"  
 android:src="@drawable/audio"  
 android:layout\_margin="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintHorizontal\_bias="1"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### activity\_walking.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"  
 android:background="@color/main\_colour">  
  
 <ImageView  
 android:layout\_width="180dp"  
 android:layout\_height="180dp"  
 android:alpha="0.2"  
 android:src="@drawable/walking"  
 app:layout\_constraintBottom\_toTopOf="@+id/startStopBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/row2" />  
  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="60dp"  
 android:fontFamily="@font/gothic"  
 android:text="Walking Tracker"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="30sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="150dp"  
 android:fontFamily="@font/gothicbb"  
 android:gravity="center"  
 android:text="00:00:00"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="80sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/stepText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Steps:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/calText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Calories:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/distText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Distance:\n0m"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="20dp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/row2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/paceText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothic"  
 android:text="Pace:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 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="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:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/main\_colour"  
 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" />  
  
 <ImageButton  
 android:id="@+id/audioBtn"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:background="@color/transparent"  
 android:scaleType="fitCenter"  
 android:src="@drawable/audio"  
 android:layout\_margin="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintHorizontal\_bias="1"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### activity\_squats.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"  
 android:background="@color/main\_colour">  
  
 <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" />  
  
  
 <FrameLayout  
 android:id="@+id/framelayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.camera.view.PreviewView  
 android:id="@+id/tv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start" />  
  
 <View  
 android:id="@+id/nose"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_shoulder"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_shoulder"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/lelbow"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/relbow"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_hip"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_hip"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_knee"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_knee"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
 </FrameLayout>  
  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="40dp"  
 android:fontFamily="@font/gothicbb"  
 android:gravity="center"  
 android:text="00:00:00"  
 android:textColor="@color/white"  
 android:textSize="80sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:id="@+id/textView5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:padding="5dp"  
 android:text="Squats Tracker"  
 android:textColor="@color/white"  
 android:textSize="30sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:id="@+id/PoseIndicator"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:padding="5dp"  
 android:text="Squat Not Entered"  
 android:textAlignment="center"  
 android:textColor="@color/red"  
 android:textSize="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/finishBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/timerText"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:id="@+id/debugTV"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothicbb"  
 android:padding="5dp"  
 android:text="debug"  
 android:visibility="gone"  
 android:textColor="@color/white"  
 android:textSize="20sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/finishBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintVertical\_bias="0" />  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/repText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Reps:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/calText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Calories:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20sp" />  
 <TextView  
 android:id="@+id/paceText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Pace:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20sp" />  
 </LinearLayout>  
  
 <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="30sp"  
 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:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/main\_colour"  
 android:textSize="30sp"  
 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" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/helpBtn"  
 android:layout\_width="50dp"  
 android:layout\_height="50dp"  
 android:layout\_marginTop="7dp"  
 android:background = "@color/transparent"  
 android:text="?"  
 android:textColor="@color/white"  
 android:textSize="25sp"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintBottom\_toBottomOf="@+id/framelayout"  
 app:layout\_constraintEnd\_toStartOf="@id/finishBtn"  
 app:layout\_constraintStart\_toEndOf="@+id/startStopBtn"  
 app:layout\_constraintTop\_toBottomOf="@+id/PoseIndicator" />  
  
 <ImageButton  
 android:id="@+id/audioBtn"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:background="@color/transparent"  
 android:scaleType="fitCenter"  
 android:src="@drawable/audio"  
 android:layout\_marginTop="50dp"  
 android:layout\_marginRight="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintHorizontal\_bias="1"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:tint="@color/white" />  
  
  
  
</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"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/main\_colour">  
  
 <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" />  
  
  
 <FrameLayout  
 android:id="@+id/framelayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.camera.view.PreviewView  
 android:id="@+id/tv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start" />  
  
 <View  
 android:id="@+id/nose"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_shoulder"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_shoulder"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/lelbow"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/relbow"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_hip"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_hip"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/right\_knee"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
  
 <View  
 android:id="@+id/left\_knee"  
 android:layout\_width="5dp"  
 android:layout\_height="5dp"  
 android:background="@drawable/rectangle" />  
 </FrameLayout>  
  
  
 <TextView  
 android:id="@+id/timerText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="40dp"  
 android:fontFamily="@font/gothicbb"  
 android:gravity="center"  
 android:text="00:00:00"  
 android:textColor="@color/white"  
 android:textSize="80sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:padding="5dp"  
 android:text="Push-Up Tracker"  
 android:textColor="@color/white"  
 android:textSize="30sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/timerText"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:id="@+id/PoseIndicator"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:padding="5dp"  
 android:text="Push Up Not Entered"  
 android:textAlignment="center"  
 android:textColor="@color/red"  
 android:textSize="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/finishBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/timerText"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <TextView  
 android:id="@+id/debugTV"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothicbb"  
 android:padding="5dp"  
 android:text="debug"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 android:visibility="gone"  
 app:layout\_constraintBottom\_toTopOf="@+id/finishBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline3"  
 app:layout\_constraintVertical\_bias="0" />  
  
 <LinearLayout  
 android:id="@+id/row1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center\_horizontal"  
 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">  
  
 <TextView  
 android:id="@+id/repText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Reps:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20dp" />  
  
 <TextView  
 android:id="@+id/calText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Calories:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20dp" />  
 <TextView  
 android:id="@+id/paceText"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/gothic"  
 android:text="Pace:\n0"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:textSize="20dp" />  
 </LinearLayout>  
  
 <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:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/main\_colour"  
 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" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/helpBtn"  
 android:layout\_width="50dp"  
 android:layout\_height="50dp"  
 android:layout\_marginTop="7dp"  
 android:background = "@color/transparent"  
 android:text="?"  
 android:textColor="@color/white"  
 android:textSize="25sp"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintBottom\_toBottomOf="@+id/framelayout"  
 app:layout\_constraintEnd\_toStartOf="@id/finishBtn"  
 app:layout\_constraintStart\_toEndOf="@+id/startStopBtn"  
 app:layout\_constraintTop\_toBottomOf="@+id/PoseIndicator" />  
  
 <ImageButton  
 android:id="@+id/audioBtn"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:background="@color/transparent"  
 android:scaleType="fitCenter"  
 android:src="@drawable/audio"  
 android:layout\_marginTop="50dp"  
 android:layout\_marginRight="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintVertical\_bias="0"  
 app:layout\_constraintHorizontal\_bias="1"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:tint="@color/white" />  
  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### activity\_loginscreen.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"  
 android:background="@color/main\_colour">  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="30dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Welcome"  
 android:textSize="60dp"  
 app:layout\_constraintBottom\_toBottomOf="@id/guideline"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Caleb's Exercise Tracker App\nA-Level C.S. NEA"  
  
 android:textAlignment="center"  
 android:textSize="20dp"  
 app:layout\_constraintBottom\_toBottomOf="@id/guideline"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="30dp"  
 android:gravity="center\_horizontal"  
 android:orientation="vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/guideline"  
 app:layout\_constraintVertical\_bias="1">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/usernameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:hint="Username:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/passwordField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="20dp"  
 android:hint="Password:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
  
 <com.google.android.material.checkbox.MaterialCheckBox  
 android:id="@+id/rememberBox"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:buttonTint="@color/main\_colour\_on\_primary"  
 android:text="Remember Me" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/loginbtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Log In"  
 android:textColor="@color/main\_colour"  
 android:textSize="20sp"  
 app:cornerRadius="20dp" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/createaccount"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Create a new account"  
 android:textColor="@color/main\_colour"  
 android:textSize="20dp"  
 app:cornerRadius="20dp" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/githubRepo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:autoLink="web"  
 android:clickable="true"  
 android:fontFamily="@font/gothicbb"  
 android:linksClickable="true"  
 android:textAlignment="center"  
 android:textColorLink="@color/main\_colour\_on\_primary"  
 android:textSize="15sp"  
  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="1" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### activity\_registeruser.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"  
 android:background="@color/main\_colour">  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="30dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Create An Account"  
 android:textSize="35sp"  
 app:layout\_constraintBottom\_toBottomOf="@id/guideline"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <ScrollView  
 android:id="@+id/scrollview"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:layout\_marginBottom="20dp"  
 android:scrollbarThumbVertical="@color/grey"  
 app:layout\_constraintBottom\_toTopOf="@id/createBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/guideline"  
 app:layout\_constraintVertical\_bias="0">  
  
  
 <LinearLayout  
 android:id="@+id/linearLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:orientation="vertical">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/firstnameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Forename:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/lastnameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Surname:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/usernameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Username:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/passwordField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Password:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/DOBfield"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Date Of Birth (yyyy-mm-dd)"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="date"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/weightField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Weight:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/heightField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="10dp"  
 android:hint="Height:"  
 app:boxStrokeColor="@color/black"  
 app:hintTextColor="@color/black">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
  
 </LinearLayout>  
 </ScrollView>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/createBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="180dp"  
 android:layout\_marginBottom="20dp"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="REGISTER"  
 android:textColor="@color/main\_colour"  
 android:textSize="30sp"  
 app:cornerRadius="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/scrollview" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/newUser\_cancelBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/main\_colour\_on\_primary"  
 android:fontFamily="@font/gothicbb"  
 android:text="Cancel"  
 android:textColor="@color/main\_colour"  
 android:textSize="30sp"  
 app:cornerRadius="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/createBtn"  
 app:layout\_constraintTop\_toBottomOf="@id/scrollview" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

#### activity\_addfriends.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">  
  
  
 <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" />  
  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Friends List"  
 android:textSize="40sp"  
  
 app:layout\_constraintBottom\_toTopOf="@+id/guideline2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/friendsRV"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginBottom="20dp"  
 app:flow\_verticalBias="0"  
 app:layout\_constraintBottom\_toTopOf="@id/searchView"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline2" />  
  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/searchView"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="30dp"  
 android:hint="Find A User:"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@id/searchBtn"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline2"  
 app:layout\_constraintVertical\_bias="1">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/searchBtn"  
 android:layout\_width="60dp"  
 android:layout\_height="60dp"  
 android:layout\_marginEnd="20dp"  
 android:background="@drawable/round\_button"  
 android:foreground="@drawable/search"  
 app:layout\_constraintBottom\_toBottomOf="@+id/searchView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/searchView"  
 app:layout\_constraintTop\_toTopOf="@+id/searchView"  
 app:rippleColor="@color/black" />  
  
 <TextView  
 android:id="@+id/noFriendsTV"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Your Friends List is empty :(\nGo to \'Find a user' to add one!"  
 android:textAlignment="center"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="17sp"  
 android:visibility="invisible"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout

### 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="70dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Exercise"  
 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:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:scrollbarThumbVertical="@color/grey"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/guideline2">  
  
 <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"  
 app:cardCornerRadius="20dp"  
  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:cardUseCompatPadding="true"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="2dp">  
  
 <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="90dp"  
 android:src="@drawable/running" />  
  
 <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"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/runBtn"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false" />  
 </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:cardCornerRadius="20dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="2dp">  
  
 <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/walking" />  
  
 <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"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false" />  
 </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:cardCornerRadius="20dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="2dp">  
  
 <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"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false" />  
 </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:cardCornerRadius="20dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="2dp">  
  
 <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/pushup" />  
  
 <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"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false" />  
 </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:cardCornerRadius="20dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:strokeColor="@color/main\_colour"  
 app:strokeWidth="2dp">  
  
 <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/squat" />  
  
 <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="Squats"  
 android:textAlignment="center"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/squatBtn"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:text="@string/start\_tracking"  
 android:textAllCaps="false"/>  
 </LinearLayout>  
 </LinearLayout>  
 </com.google.android.material.card.MaterialCardView>  
  
 </LinearLayout>  
 </ScrollView>  
</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:id="@+id/textView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="70dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Activity History"  
 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"  
 app:layout\_constraintVertical\_weight="0" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="180dp" />  
  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/HistoryRV"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/guideline2"  
 app:layout\_constraintVertical\_bias="1.0"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
 <TextView  
 android:id="@+id/noExercises"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="No Activities have been recorded\nGo to \'Exercise' to start one!"  
 android:textAlignment="center"  
 android:textColor="@color/main\_colour\_on\_primary"  
 android:textSize="17sp"  
 android:visibility="invisible"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="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"  
 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="70dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Settings"  
 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" />  
  
  
 <ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginBottom="10dp"  
 app:layout\_constraintBottom\_toTopOf="@id/UpdateButton"  
 app:layout\_constraintTop\_toTopOf="@id/guideline2"  
 app:layout\_constraintVertical\_bias="0">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:gravity="center\_horizontal"  
 android:orientation="vertical">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/settings\_usernameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:hint="Username:"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/settings\_passwordField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Password:"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/forenameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Forename:"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/surnameField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Surname:"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/DOBfield"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Date Of Birth (yyyy-mm-dd)"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/dobText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="date"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/weightField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Weight/kg"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="numberDecimal"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/heightField"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="30dp"  
 android:hint="Height/cm"  
 app:boxStrokeColor="@color/main\_colour\_on\_primary"  
  
 app:hintTextColor="@color/main\_colour\_on\_primary">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/heightText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:textSize="20sp" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 </LinearLayout>  
 </ScrollView>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/UpdateButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="10dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Update Details"  
 android:textAllCaps="false"  
 android:textSize="18sp"  
 app:cornerRadius="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@id/logoutBtn"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline2"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/logoutBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="10dp"  
 android:fontFamily="@font/gothicbb"  
 android:text="Logout"  
 android:textAllCaps="false"  
 android:textSize="18sp"  
 app:cornerRadius="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@id/deleteBtn"  
 app:layout\_constraintStart\_toEndOf="@id/UpdateButton"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline2"  
 app:layout\_constraintVertical\_bias="1" />  
  
 <com.google.android.material.button.MaterialButton  
  
 android:id="@+id/deleteBtn"  
 android:layout\_width="35dp"  
 android:layout\_height="35dp"  
 android:layout\_marginBottom="10dp"  
 android:background="@drawable/delete"  
 app:backgroundTint="@color/black"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/logoutBtn"  
 app:layout\_constraintTop\_toTopOf="@id/UpdateButton" />  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

### 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)

#### ExerciseFragment.java

package com.example.exercisetracker.fragments;  
  
import android.annotation.SuppressLint;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationManagerCompat;  
import androidx.fragment.app.Fragment;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.activities.PushUpActivity;  
import com.example.exercisetracker.activities.RunningActivity;  
import com.example.exercisetracker.activities.SquatsActivity;  
import com.example.exercisetracker.activities.TreadmillActivity;  
import com.example.exercisetracker.activities.WalkingActivity;  
  
public class ExerciseFragment extends Fragment implements View.OnClickListener, ActivityCompat.OnRequestPermissionsResultCallback {  
 private String[] PERMISSIONS;  
 *//notification* private NotificationManagerCompat notificationManagerCompat;  
  
 @Override  
 public void onResume() {  
 super.onResume();  
 }  
  
 @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);  
 Button squatBtn = (Button) view.findViewById(R.id.squatBtn);  
 treadmillBtn.setOnClickListener(this);  
 pushUpBtn.setOnClickListener(this);  
 runBtn.setOnClickListener(this);  
 walkBtn.setOnClickListener(this);  
 squatBtn.setOnClickListener(this);  
 return view;  
 }  
  
 @SuppressLint("NonConstantResourceId")  
 @Override  
 public void onClick(View v) {  
 switch (v.getId()) {  
 case R.id.runBtn:  
 *//starting running activity* Intent intent1 = new Intent(getContext(), RunningActivity.class);  
 startActivity(intent1);  
  
 break;  
 case R.id.walkBtn:  
 *//starting walking activity* Intent intent2 = new Intent(getContext(), WalkingActivity.class);  
 startActivity(intent2);  
 break;  
 case R.id.treadmillBtn:  
 *//starting treadmill activity* Intent intent3 = new Intent(getContext(), TreadmillActivity.class);  
 startActivity(intent3);  
 break;  
 case R.id.pushUpBtn:  
 *//starting push up activity* Intent intent4 = new Intent(getContext(), PushUpActivity.class);  
 startActivity(intent4);  
 break;  
 case R.id.squatBtn:  
 *//starting squat activity* Intent intent5 = new Intent(getContext(), SquatsActivity.class);  
 startActivity(intent5);  
 }  
 }  
  
  
}

#### LeaderboardFragment.java

package com.example.exercisetracker.fragments;  
  
import android.app.ProgressDialog;  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.Typeface;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.RadioGroup;  
import android.widget.TableLayout;  
import android.widget.TableRow;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.core.content.res.ResourcesCompat;  
import androidx.fragment.app.Fragment;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.activities.AddFriendsActivity;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.LinkedHashMap;  
import java.util.List;  
import java.util.Map;  
  
public class LeaderboardFragment extends Fragment implements View.OnClickListener {  
 private TableLayout table;  
 private Integer timeframe;  
 private Map<String, Integer> userScores;  
 private Boolean isPublic;  
 private ProgressDialog loadingDialog;  
 private TextView noLeaderboard;  
 private android.app.Activity mcontext;  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
  
 if (context instanceof android.app.Activity) {  
 mcontext = (android.app.Activity) context;  
 }  
 }  
  
  
 @Override  
 public void onResume() {  
 super.onResume();  
 if (getActivity() != null) {  
 *//loading leaderboard when fragment resumed* new GetLeaderboardTask().execute(isPublic);  
 }  
 }  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.fragment\_leaderboard, container, false);  
 view.findViewById(R.id.navigateToFriendsActivity).setOnClickListener(this);  
 table = view.findViewById(R.id.table\_main);  
 noLeaderboard = view.findViewById(R.id.noLeaderboard);  
 *//by default, leaderboard set to public leaderboard at 24Hr* isPublic = true;  
 timeframe = 1;  
  
  
 RadioGroup publicPrivateRG = view.findViewById(R.id.leaderboard\_publicPrivateRG);  
 publicPrivateRG.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(RadioGroup group, int checkedId) {  
 if (checkedId == R.id.friendsBtn) {  
 isPublic = false;  
 *//when user selects friends radio button, private leaderboard is shown  
 //updating User class for friends list* DBhelper helper = new DBhelper(mcontext);  
 *//retrieving friends from database* if (helper.getFriends()) {  
 User.clearFriendsList();  
 for (String query : helper.getResult()) {  
 String[] arr = query.split(" ");  
 *//parsing user id to user class* System.out.println(arr[0]);  
 User.addFriendsList(Integer.parseInt(arr[0]));  
 }  
 helper.clearResults();  
 } else {  
 Toast.makeText(mcontext, "Could not retrieve your friends", Toast.LENGTH\_SHORT).show();  
 }  
  
 new GetLeaderboardTask().execute(isPublic);  
 } else if (checkedId == R.id.allUsersBtn) {  
 isPublic = true;  
 *//when user selects all users radio button, public leaderboard is shown  
// createTable(getPublicLeaderboard());* new GetLeaderboardTask().execute(isPublic);  
 }  
 }  
 });  
  
 RadioGroup periodRG = view.findViewById(R.id.period\_radiogroup);  
 periodRG.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(RadioGroup group, int checkedId) {  
 *//when user changes the time period of the leaderboard* switch (checkedId) {  
 case R.id.oneDay:  
 timeframe = 1;  
 new GetLeaderboardTask().execute(isPublic);  
 *//period of the last 24 hrs* break;  
 case R.id.oneMonth:  
 timeframe = 30;  
 new GetLeaderboardTask().execute(isPublic);  
 *//period of the last 30 days* break;  
 case R.id.allTime:  
 timeframe = 0;  
 new GetLeaderboardTask().execute(isPublic);  
 *//period of all time* break;  
 }  
 }  
 });  
 return view;  
 }  
  
  
 private Map<String, Integer> getPublicLeaderboard() {  
  
 *//getting all activities from database, and performing calculations  
 //for top most active users* DBhelper helper = new DBhelper(mcontext);  
 LinkedHashMap<String, Integer> userScoresHashMap = new LinkedHashMap<>();  
 if (helper.getAllActivities(timeframe)) {  
 for (String string : helper.getResult()) {  
 String[] row = string.split(" ");  
 if (userScoresHashMap.get(row[0]) != null) {  
 *//if the user exists on the hash map  
 //previous total added on top* userScoresHashMap.put(row[0], Integer.parseInt(row[1]) + userScoresHashMap.get(row[0]));  
  
 } else {  
 *//if user does not yet exist on hash map* userScoresHashMap.put(row[0], Integer.parseInt(row[1]));  
 }  
 }  
  
 *//converting hashmap to arraylist* ArrayList<String> listOfKeys = new ArrayList<>();  
 ArrayList<Integer> listOfValues = new ArrayList<>();  
 for (Map.Entry<String, Integer> entry : userScoresHashMap.entrySet()) {  
 String username = entry.getKey();  
 Integer calories = entry.getValue();  
 *//parsing to arraylists* listOfKeys.add(username);  
 listOfValues.add(calories);  
 }  
 *//returning the sorted hash map rather than converting to array then sorting* return sortHashMapByValues(userScoresHashMap);  
 } else {  
 return null;  
 }  
  
 }  
  
 private Map<String, Integer> getPrivateLeaderboard() {  
 *//if the user has at least one friend* if (!User.getFriendsList().isEmpty()) {  
 *//getting all activities from database, and performing calculations  
 //for top most active users* DBhelper helper = new DBhelper(mcontext);  
 if (helper.getFriends()) {  
 for (String friend : helper.getResult()) {  
 *//as query returns multiple columns, must slice string  
 //first column is always the id of the friend* String[] arr = friend.split(" ");  
 User.addFriendsList(Integer.parseInt(arr[0]));  
 }  
 helper.clearResults();  
 } else {  
 Toast.makeText(mcontext, "Could not retrieve your friends", Toast.LENGTH\_SHORT).show();  
 }  
 LinkedHashMap<String, Integer> userScoresHashMap = new LinkedHashMap<>();  
 *//getting activities of friends* if (helper.getFriendsActivities(timeframe, User.getFriendsList())) {  
 for (String string : helper.getResult()) {  
 String[] row = string.split(" ");  
 if (userScoresHashMap.get(row[0]) != null) {  
 *//if the user exists on the hash map  
 //previous total added on top* userScoresHashMap.put(row[0], Integer.parseInt(row[1]) + userScoresHashMap.get(row[0]));  
  
 } else {  
 *//if user does not yet exist on hash map* userScoresHashMap.put(row[0], Integer.parseInt(row[1]));  
 }  
 }  
  
 *//converting hashmap to arraylist* ArrayList<String> listOfKeys = new ArrayList<>();  
 ArrayList<Integer> listOfValues = new ArrayList<>();  
 for (Map.Entry<String, Integer> entry : userScoresHashMap.entrySet()) {  
 String username = entry.getKey();  
 Integer calories = entry.getValue();  
 *//parsing to arraylists* listOfKeys.add(username);  
 listOfValues.add(calories);  
 }  
 *//returning the sorted hash map rather than converting to array then sorting* return sortHashMapByValues(userScoresHashMap);  
 } else {  
 return null;  
 }  
 }  
 return null;  
 }  
  
 private Map<String, Integer> sortHashMapByValues(HashMap<String, Integer> passedMap) {  
 *//lists of values and keys in separate lists* List<String> mapKeys = new ArrayList<>(passedMap.keySet());  
 List<Integer> mapValues = new ArrayList<>(passedMap.values());  
 *//sort in descending order* Collections.sort(mapValues, Collections.reverseOrder());  
 Collections.sort(mapKeys, Collections.reverseOrder());  
  
 LinkedHashMap<String, Integer> sortedMap = new LinkedHashMap<>();  
  
 for (Integer val : mapValues) {  
 Iterator<String> keyIt = mapKeys.iterator();  
 while (keyIt.hasNext()) {  
 String key = keyIt.next();  
 Integer comp1 = passedMap.get(key);  
 Integer comp2 = val;  
 if (comp1.equals(comp2)) {  
 *//if two key pair values have the same value* keyIt.remove();  
 sortedMap.put(key, val);  
 break;  
 }  
 }  
 }  
 return sortedMap;  
 }  
  
 private void createTable(Map<String, Integer> hashMap) {  
 if (hashMap != null) {  
 noLeaderboard.setVisibility(View.INVISIBLE);  
 *//method to create TableLayout view graphic, to display leaderboard* int pos = 1;  
 *//creating table headers* String[] arr = {"Pos. ", "User ", "Score"};  
 TableRow row = new TableRow(mcontext);  
 for (String string : arr) {  
 TextView tv = new TextView(mcontext);  
 handleViews(tv, string, true, 25);  
 row.addView(tv);  
 }  
 table.addView(row);  
 for (Map.Entry<String, Integer> entry : hashMap.entrySet()) {  
 *//getting values from sorted hash map* Integer fontsize = 20;  
 String name = entry.getKey();  
 Integer score = entry.getValue();  
 *//creating a row* row = new TableRow(mcontext);  
 TableRow.LayoutParams lp = new TableRow.LayoutParams(TableRow.LayoutParams.WRAP\_CONTENT);  
 row.setLayoutParams(lp);  
 TextView posTV = new TextView(mcontext);  
 handleViews(posTV, Integer.toString(pos) + ". ", false, fontsize);  
 TextView nameTV = new TextView(mcontext);  
 handleViews(nameTV, name, false, fontsize);  
 TextView scoreTV = new TextView(mcontext);  
 handleViews(scoreTV, score.toString(), false, fontsize);  
 *//if the corresponding name corresponds to the current user's username  
 //change colour to green to highlight their place on the leaderboard* if (name.equals(User.getUsername())) {  
 posTV.setTextColor(mcontext.getResources().getColor(R.color.green));  
 scoreTV.setTextColor(mcontext.getResources().getColor(R.color.green));  
 nameTV.setTextColor(mcontext.getResources().getColor(R.color.green));  
 }  
 row.addView(posTV);  
 row.addView(nameTV);  
 row.addView(scoreTV);  
 table.addView(row);  
 pos++;  
 }  
 } else {  
 *//show message to user that no activities found during this period on leaderboard* noLeaderboard.setVisibility(View.VISIBLE);  
 }  
  
 }  
  
 private void handleViews(TextView view, String text, Boolean bold, Integer fontsize) {  
 if (bold) {  
 Typeface face = ResourcesCompat.getFont(mcontext, R.font.gothicbb);  
 view.setTypeface(face);  
 view.setText(text + " ");  
 view.setTextSize(fontsize);  
 } else {  
 Typeface face = ResourcesCompat.getFont(mcontext, R.font.gothic);  
 view.setTypeface(face);  
 view.setText(text + " ");  
 view.setTextSize(fontsize);  
 }  
  
 }  
  
 @Override  
 public void onClick(View v) {  
 if (v.getId() == R.id.navigateToFriendsActivity) {  
 Intent intent1 = new Intent(mcontext, AddFriendsActivity.class);  
 startActivity(intent1);  
 }  
 }  
  
 *//using async task to retrieve data from database* private class GetLeaderboardTask extends AsyncTask<Boolean, Integer, Map<String, Integer>> {  
 protected Map<String, Integer> doInBackground(Boolean... isPublic) {  
 mcontext.runOnUiThread(new Runnable() {  
 *//clearing table and disclaimer for loading animation* @Override  
 public void run() {  
 noLeaderboard.setVisibility(View.INVISIBLE);  
 table.removeAllViews();  
 if (mcontext.findViewById(R.id.progressBar) != null) {  
 mcontext.findViewById(R.id.progressBar).setVisibility(View.VISIBLE);  
  
 }  
 }  
 });  
 *//bool is if public, if true get public leaderboard  
 //if false then get private leaderboard* Boolean bool = isPublic[0];  
 if (bool) {  
 return getPublicLeaderboard();  
 } else {  
 return getPrivateLeaderboard();  
 }  
 }  
  
 protected void onPostExecute(Map<String, Integer> result) {  
 mcontext.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 if (result != null) {  
 if (!result.isEmpty()) {  
 *//creating table* createTable(result);  
 *//hiding progress bar* } else {  
 *//table was empty, disclaimer shown to user* noLeaderboard.setVisibility(View.VISIBLE);  
 }  
 } else {  
 *//table was empty, disclaimer shown to user* noLeaderboard.setVisibility(View.VISIBLE);  
 }  
 if (mcontext.findViewById(R.id.progressBar) != null) {  
 mcontext.findViewById(R.id.progressBar).setVisibility(View.GONE);  
 }  
 }  
 });  
 }  
 }  
}

#### SettingsFragment.java

package com.example.exercisetracker.fragments;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.AsyncTask;  
import android.os.Bundle;  
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.ProgressBar;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AlertDialog;  
import androidx.fragment.app.Fragment;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.login.LogInScreen;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
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.dialog.MaterialAlertDialogBuilder;  
import com.google.android.material.textfield.TextInputLayout;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Objects;  
  
public class SettingsFragment extends Fragment implements View.OnClickListener {  
 private TextInputLayout usernameField;  
 private TextInputLayout passwordField;  
 private TextInputLayout weightField;  
 private TextInputLayout DOBField;  
 private TextInputLayout forenameField;  
 private TextInputLayout surnameField;  
 private TextInputLayout heightField;  
 private SharedPreferences sp;  
 private ProgressBar progressBar;  
 private android.app.Activity mcontext;  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
 *//saving the attached activity to preserve lifecycle of fragment  
 //ensures that UI thread runs on an instance of an activity* if (context instanceof android.app.Activity) {  
 mcontext = (android.app.Activity) context;  
 }  
 }  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.*fragment\_settings*, container, false);  
 sp = mcontext.getSharedPreferences("userprefs", Context.*MODE\_PRIVATE*);  
 usernameField = view.findViewById(R.id.*settings\_usernameField*);  
 passwordField = view.findViewById(R.id.*settings\_passwordField*);  
 weightField = view.findViewById(R.id.*weightField*);  
 forenameField = view.findViewById(R.id.*forenameField*);  
 surnameField = view.findViewById(R.id.*surnameField*);  
 DOBField = view.findViewById(R.id.*DOBfield*);  
 heightField = view.findViewById(R.id.*heightField*);  
 progressBar = view.findViewById(R.id.*progressBar*);  
 *//handling update and logout buttons* Button updateButton = view.findViewById(R.id.*UpdateButton*);  
 updateButton.setOnClickListener(this);  
 Button logoutBtn = view.findViewById(R.id.*logoutBtn*);  
 logoutBtn.setOnClickListener(this);  
 Button deleteBtn = view.findViewById(R.id.*deleteBtn*);  
 deleteBtn.setOnClickListener(this);  
 new GetSettings().execute(true);  
 return view;  
 }  
  
 @SuppressLint("NonConstantResourceId")  
 @Override  
 public void onClick(View v) {  
 switch (v.getId()) {  
 case R.id.*UpdateButton*:  
 *//saving user details to User class and updating database  
 //show dialogue to user to confirm if they want to delete account* MaterialAlertDialogBuilder builder3 = createDialogBuilder("Update My Details?", "Are you sure you want to update your account?");  
 builder3.setPositiveButton(R.string.*confirm*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 try {  
 *// use of .replaceAll to sanatise inputs given by user, to remove any whitespaces* String username = String.*valueOf*(Objects.*requireNonNull*(usernameField.getEditText()).getText());  
 String password = String.*valueOf*(Objects.*requireNonNull*(passwordField.getEditText()).getText());  
 Float weight = Float.*parseFloat*(String.*valueOf*(Objects.*requireNonNull*(weightField.getEditText()).getText()));  
 Integer height = Integer.*valueOf*(String.*valueOf*(Objects.*requireNonNull*(heightField.getEditText()).getText()));  
 String forename = String.*valueOf*(Objects.*requireNonNull*(forenameField.getEditText()).getText());  
 String surname = String.*valueOf*(Objects.*requireNonNull*(surnameField.getEditText()).getText());  
 String DOB = Objects.*requireNonNull*(DOBField.getEditText()).getText().toString();  
  
 *//username and password must be at least 8 characters  
 //max length of username is 16 characters* boolean requirements = username.length() >= 8 && password.length() >= 8 && username.length() <= 16;  
 *//if one field is empty, cannot update account* boolean isEmpty = username.isEmpty() || password.isEmpty() || forename.isEmpty() || surname.isEmpty() || DOB.isEmpty();  
 boolean containsSpace = username.contains(" ") || password.contains(" ") || forename.contains(" ") || surname.contains(" ");  
 if (requirements) {  
 if (!isEmpty) {  
 if (!containsSpace) {  
 DBhelper helper = new DBhelper(mcontext);  
 if (helper.updateUser(username, password, forename, surname, DOB, weight.toString(), height.toString())) {  
 *//if update on database was successful* Toast.*makeText*(mcontext, "Save successful", Toast.*LENGTH\_SHORT*).show();  
  
 *//saving to SharedPreferences if user had checked remember me* SharedPreferences prefs = mcontext.getSharedPreferences("sharedPrefs", Context.*MODE\_PRIVATE*);  
 Boolean checkbox = prefs.getBoolean("remember", false);  
 if (checkbox) {  
 *// results received in format ID, forename, surname, DOB, weight, height* SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("username", username);  
 editor.putString("password", password);  
 editor.putString("forename", forename);  
 editor.putString("surname", surname);  
 editor.putString("DOB", DOBField.getEditText().getText().toString());  
 editor.putString("weight", weight.toString());  
 editor.putString("height", height.toString());  
 editor.apply();  
 *//saving to user class* User.*saveUser*(username, password, forename, surname, DOB, weight, height);  
 }  
 }  
 } else {  
 Toast.*makeText*(mcontext, "No fields can contain spaces", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 else{  
 Toast.*makeText*(mcontext, "No fields can be left empty", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(mcontext, "You have not met the requirements", Toast.*LENGTH\_SHORT*).show();  
 Toast.*makeText*(mcontext, "Username and password must be between 8-16 characters", Toast.*LENGTH\_SHORT*).show();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 Toast.*makeText*(mcontext, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 dialog.cancel();  
 }  
 });  
 AlertDialog dialog3 = builder3.create();  
 dialog3.show();  
  
 break;  
 case R.id.*logoutBtn*:  
 *//When the user wants to logout, clearing User details  
 //Clearing shared preferences  
 //show dialogue to user to confirm if they want to delete account* MaterialAlertDialogBuilder builder = createDialogBuilder("Logout?", "Are you sure you want to logout?");  
 builder.setPositiveButton(R.string.*confirm*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 User.*logout*(mcontext);  
 Intent intent1 = new Intent(mcontext, LogInScreen.class);  
 startActivity(intent1);  
 dialog.cancel();  
 getActivity().finish();  
 }  
 });  
 AlertDialog dialog = builder.create();  
 dialog.show();  
 break;  
 case R.id.*deleteBtn*:  
 *//handling deleting an account  
 //show dialogue to user to confirm if they want to delete account* MaterialAlertDialogBuilder builder2 = createDialogBuilder("Delete My Account?", "Are you sure you want to delete your account?");  
 builder2.setPositiveButton(R.string.*confirm*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 DBhelper helper = new DBhelper(mcontext);  
 if (helper.deleteAccount(User.*getUserID*())) {  
 Toast.*makeText*(mcontext, "Account Deleted", Toast.*LENGTH\_SHORT*).show();  
 User.*logout*(mcontext);  
 Intent intent1 = new Intent(mcontext, LogInScreen.class);  
 startActivity(intent1);  
 getActivity().finish();  
 } else {  
 Toast.*makeText*(mcontext, "Error", Toast.*LENGTH\_SHORT*).show();  
 }  
 dialog.cancel();  
 }  
 });  
 AlertDialog dialog2 = builder2.create();  
 dialog2.show();  
 break;  
 }  
 }  
  
 private MaterialAlertDialogBuilder createDialogBuilder(String title, String message) {  
 MaterialAlertDialogBuilder builder = new MaterialAlertDialogBuilder(mcontext);  
 builder.setNegativeButton(R.string.*cancel*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 })  
 ;  
 builder.setMessage(message)  
 .setTitle(title);  
 return builder;  
 }  
  
 *//using async task to load user details* private class GetSettings extends AsyncTask<Boolean, Integer, ArrayList<String>> {  
 protected ArrayList<String> doInBackground(Boolean... isPublic) {  
 ArrayList<String> arr = new ArrayList<>();  
 *//loading user data presets* arr.add(User.*getUsername*());  
 arr.add(User.*getPassword*());  
 arr.add(User.*getWeight*().toString());  
 arr.add(User.*getHeight*().toString());  
 arr.add(User.*getForename*());  
 arr.add(User.*getSurname*());  
 Date dob = User.*getDateOfBirth*();  
 *//date of birth, date picker* DateFormat df = new SimpleDateFormat("yyyy-MM-dd");  
 String strdob = df.format(dob);  
 arr.add(strdob);  
 return arr;  
 }  
  
 protected void onProgressUpdate(Integer... progress) {  
 }  
  
 protected void onPostExecute(ArrayList<String> queryResults) {  
  
 mcontext.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 *//loading user data presets* Objects.*requireNonNull*(usernameField.getEditText()).setText(queryResults.get(0));  
 Objects.*requireNonNull*(passwordField.getEditText()).setText(queryResults.get(1));  
 Objects.*requireNonNull*(weightField.getEditText()).setText(queryResults.get(2));  
 Objects.*requireNonNull*(heightField.getEditText()).setText(queryResults.get(3));  
 Objects.*requireNonNull*(forenameField.getEditText()).setText(queryResults.get(4));  
 Objects.*requireNonNull*(surnameField.getEditText()).setText(queryResults.get(5));  
 String strdob = queryResults.get(6);  
 *//hiding progress bar* progressBar.setVisibility(View.*GONE*);  
  
 *//handling date of birth* EditText dobText = DOBField.getEditText();  
 dobText.setText(strdob);  
 dobText.setInputType(InputType.*TYPE\_NULL*);  
 dobText.setKeyListener(null);  
  
 *//date of birth picker constraints, must be at least 10 yrs old to register account* CalendarConstraints.Builder constraints = new CalendarConstraints.Builder().setValidator(DateValidatorPointBackward.*before*(MaterialDatePicker.*todayInUtcMilliseconds*() - 315569260000L));  
 MaterialDatePicker.Builder<Long> datepickerBuilder = MaterialDatePicker.Builder.*datePicker*();  
 *//by default starts picker on min age* datepickerBuilder.setCalendarConstraints(constraints.build()).setSelection(MaterialDatePicker.*todayInUtcMilliseconds*() - 315569260000L);  
 MaterialDatePicker datepicker = datepickerBuilder.build();  
  
 *//when date of birth is touched* dobText.setOnTouchListener(new View.OnTouchListener() {  
 @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* DateFormat df = new SimpleDateFormat("yyyy-MM-dd");  
 dobText.setText(df.format(datepicker.getSelection()));  
 java.sql.Date date = new java.sql.Date((Long) datepicker.getSelection());  
 User.*setDateOfBirth*(date);  
 datepicker.dismiss();  
 }  
 });  
 datepicker.addOnCancelListener(new DialogInterface.OnCancelListener() {  
 @Override  
 public void onCancel(DialogInterface dialog) {  
 datepicker.dismiss();  
 }  
 });  
  
 }  
 });  
 }  
 }  
  
}

#### HistoryFragment.java

package com.example.exercisetracker.fragments;  
  
import android.content.Context;  
import android.content.DialogInterface;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AlertDialog;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.activities.Activity;  
import com.example.exercisetracker.other.DBhelper;  
import com.google.android.material.dialog.MaterialAlertDialogBuilder;  
  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Locale;  
  
public class HistoryFragment extends Fragment {  
 private ActivityAdapter courseAdapter;  
 private ArrayList<Activity> activityArr;  
 private RecyclerView historyRV;  
 private LinearLayoutManager linearLayoutManager;  
 private TextView noHistory;  
 private ProgressBar progressBar;  
 private android.app.Activity mcontext;  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
 *//saving the attached activity to preserve lifecycle of fragment  
 //ensures that UI thread runs on an instance of an activity* if (context instanceof android.app.Activity) {  
 mcontext = (android.app.Activity) context;  
 }  
 }  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.fragment\_history, container, false);  
 noHistory = view.findViewById(R.id.noExercises);  
 historyRV = view.findViewById(R.id.HistoryRV);  
 activityArr = new ArrayList<>();  
 courseAdapter = new ActivityAdapter(mcontext, activityArr);  
 linearLayoutManager = new LinearLayoutManager(mcontext, LinearLayoutManager.VERTICAL, false);  
 progressBar = view.findViewById(R.id.progressBar);  
  
  
 *// setting layout manager and adapter to our recycler view.* historyRV.setLayoutManager(linearLayoutManager);  
 historyRV.setAdapter(courseAdapter);  
  
 *//EXECUTING ASYNC TASK to retrieve data history from database* new GetHistory().execute(true);  
 return view;  
 }  
  
 private Activity handleQuery(String query) {  
 *//method to handle query* String[] arr = query.split(" ");  
 int id = Integer.parseInt(arr[0]);  
 String name = arr[1];  
 *//description consisting of date and time* String[] desc = Arrays.copyOfRange(arr, 2, arr.length);  
 String description = "";  
 for (String string : desc) {  
 description = description + string + " ";  
 }  
 int img = -1;  
 *//adding images to corresponding exercise* switch (name) {  
 case "running":  
 img = R.drawable.running;  
 name = "Running";  
 break;  
 case "treadmill":  
 img = R.drawable.treadmill;  
 name = "Treadmill";  
 break;  
 case "walking":  
 img = R.drawable.walking;  
 name = "Walking";  
 break;  
 case "pushup":  
 img = R.drawable.pushup;  
 name = "Push Up";  
 break;  
 case "squats":  
 img = R.drawable.squat;  
 name = "Squats";  
 break;  
 }  
 if (img != -1) {  
 return new Activity(name, description, img, id);  
 } else {  
 return null;  
 }  
  
 }  
  
 *//handling dynamic card production using recycler views* public static class ActivityAdapter extends RecyclerView.Adapter<ActivityAdapter.Viewholder> {  
  
 private final Context context;  
 private final ArrayList<Activity> ActivityArr;  
  
 public ActivityAdapter(Context context, ArrayList<Activity> courseModelArrayList) {  
 this.context = context;  
 this.ActivityArr = courseModelArrayList;  
 }  
  
 @NonNull  
 @Override  
 public ActivityAdapter.Viewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 *//inflate the layout for each item of recycler view.* View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.history\_card\_layout, parent, false);  
 return new Viewholder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull Viewholder holder, int position) {  
 *//set data to textview and imageview of each card layout  
 //get details from activity array holding items of activity class  
 //providing details for the holder views* Activity activity = ActivityArr.get(position);  
 holder.exerciseNameTV.setText(activity.getName());  
 holder.exerciseDescTV.setText(activity.getDate().toString() + " " + activity.getTimeStarted());  
 holder.exerciseIV.setImageResource(activity.getImg());  
 holder.deleteBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//handling when delete button is pressed (deleting activity record on database)  
 //first show dialog to user to confirm if they really want to delete the activity from history* String title = "Delete Activity?";  
 String message = "Are you sure you want to delete this activity?";  
 MaterialAlertDialogBuilder builder = new MaterialAlertDialogBuilder(context);  
 builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 DBhelper helper = new DBhelper(context.getApplicationContext());  
 if (helper.deleteActivity(activity.getId())) {  
 Toast.makeText(context.getApplicationContext(), "Activity Deleted", Toast.LENGTH\_SHORT).show();  
 ActivityArr.remove(activity);  
 notifyItemRemoved(holder.getAdapterPosition());  
 notifyItemRangeChanged(holder.getAdapterPosition(), getItemCount());  
 } else {  
 Toast.makeText(context.getApplicationContext(), "Delete Failed", Toast.LENGTH\_SHORT).show();  
 }  
 dialog.cancel();  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int i) {  
 dialog.cancel();  
 }  
 });  
 builder.setTitle(title)  
 .setMessage(message);  
 AlertDialog dialog = builder.create();  
 dialog.show();  
  
 }  
 });  
 holder.moreDetailsBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//handling when more details button is pressed  
 //displays as a dialogue the full stats of a workout* createDialogBuilder(holder.exerciseNameTV.getText().toString(), activity);  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return ActivityArr.size();  
 }  
  
 private void createDialogBuilder(String title, Activity activity) {  
 *//creating the alert dialog to show stats to user from a previous exercise* int hours = activity.getDuration() / 3600;  
 int minutes = (activity.getDuration() % 3600) / 60;  
 int secs = activity.getDuration() % 60;  
 Float pace = Float.valueOf(activity.getDistance()) / Float.valueOf(activity.getDuration());  
 pace = pace \* 3.6f; *//converting from m/s to km/hr* DecimalFormat df = new DecimalFormat("#.##");  
 String time = String.format(Locale.getDefault(), "%dh:%02dm:%02ds", hours, minutes, secs);  
 String message = "";  
 if (activity.getName().equals("Running") || activity.getName().equals("Walking") || activity.getName().equals("Treadmill")) {  
 message =  
 String.format(Locale.getDefault(), "Duration: %s\nCalories: %d\nSteps: %d\nDistance: %dm\nAvg. Pace: %skm/hr",  
 time, activity.getCalories(), activity.getSteps(), activity.getDistance(), df.format(pace));  
 } else if (activity.getName().equals("Push Up") || activity.getName().equals("Squats")) {  
 message =  
 String.format(Locale.getDefault(), "Duration: %s\nCalories Burnt: %d\nReps: %d\nAvg. Pace: %sreps/min",  
 time, activity.getCalories(), activity.getReps(),df.format(((float) activity.getReps())\*60f/(float)activity.getDuration()));  
 }  
  
 MaterialAlertDialogBuilder builder = new MaterialAlertDialogBuilder(this.context);  
 builder.setNegativeButton("dismiss", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 })  
 ;  
 builder.setTitle(title)  
 .setMessage(message)  
 ;  
 AlertDialog dialog = builder.create();  
 dialog.show();  
 }  
  
 *// View holder class for initializing of views such as TextView and Imageview.* public static class Viewholder extends RecyclerView.ViewHolder {  
 private final ImageView exerciseIV;  
 private final TextView exerciseNameTV;  
 private final TextView exerciseDescTV;  
 private final Button deleteBtn;  
 private final Button moreDetailsBtn;  
  
 public Viewholder(@NonNull View itemView) {  
 super(itemView);  
 *//binding views to instance of Viewholder* exerciseIV = itemView.findViewById(R.id.ExeciseImg);  
 exerciseNameTV = itemView.findViewById(R.id.ExerciseName);  
 exerciseDescTV = itemView.findViewById(R.id.ExerciseDesc);  
 deleteBtn = itemView.findViewById(R.id.ExerciseDeleteBtn);  
 moreDetailsBtn = itemView.findViewById(R.id.ExerciseDetailsBtn);  
 }  
 }  
  
 }  
  
 *//using async task to retrieve data from database* private class GetHistory extends AsyncTask<Boolean, Integer, ArrayList<String>> {  
 protected ArrayList<String> doInBackground(Boolean... isPublic) {  
 DBhelper helper = new DBhelper(mcontext);  
 if (helper.readActivities()) {  
 if (isCancelled()) return null;  
 *//if activities was read successfully from database* ArrayList<String> queryResults = helper.getResult();  
 if (queryResults.isEmpty()) {  
 *//activity was not read successfully, recycler view not created  
 //show disclaimer text view on screen* mcontext.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 noHistory.setVisibility(View.VISIBLE);  
 }  
 });  
 }  
 return queryResults;  
 } else {  
 *//activity was not read successfully, recycler view not created  
 //show disclaimer text view on screen* mcontext.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 noHistory.setVisibility(View.VISIBLE);  
 }  
 });  
 }  
 return null;  
 }  
  
 protected void onProgressUpdate(Integer... progress) {  
 }  
  
 protected void onPostExecute(ArrayList<String> queryResults) {  
 if (queryResults != null) {  
 mcontext.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 *//RecyclerView allows us to dynamically produce card views as a list  
 // Arraylist for storing data* activityArr = new ArrayList<>();  
 for (String query : queryResults) {  
 activityArr.add(handleQuery(query));  
 }  
 *// we are initializing our adapter class and passing our arraylist to it.* courseAdapter = new ActivityAdapter(mcontext, activityArr);  
 *//setting a layout manager for our recycler view.  
 // creating vertical list* linearLayoutManager = new LinearLayoutManager(mcontext, LinearLayoutManager.VERTICAL, false);  
 *// setting layout manager and adapter to our recycler view.* historyRV.setLayoutManager(linearLayoutManager);  
 historyRV.setAdapter(courseAdapter);  
 }  
 });  
 }  
 *//hiding progress bar* progressBar.setVisibility(View.GONE);  
 }  
 }  
}

#### MainActivity.java

package com.example.exercisetracker.activities;  
  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.view.MenuItem;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.fragments.ExerciseFragment;  
import com.example.exercisetracker.fragments.HistoryFragment;  
import com.example.exercisetracker.fragments.LeaderboardFragment;  
import com.example.exercisetracker.fragments.SettingsFragment;  
import com.google.android.material.bottomnavigation.BottomNavigationView;  
import com.google.android.material.navigation.NavigationBarView;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* setContentView(R.layout.*activity\_main*);  
 getSupportActionBar().hide();  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
  
 *//getting bottom menu view* BottomNavigationView bottomMenu = findViewById(R.id.*bottom\_navigation*);  
 bottomMenu.setSelectedItemId(R.id.*page\_1*);  
 *//by default fragment is on exercisefragment* 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 Fragment();  
 switch (item.getItemId()) {  
 case (R.id.*page\_1*):  
 *//when exercise is navigated to, switch to exercise fragment* fragment = new ExerciseFragment();  
 break;  
 case (R.id.*page\_2*):  
 *//when history is navigated to, switch to history fragment* fragment = new HistoryFragment();  
 break;  
 case (R.id.*page\_3*):  
 *//when the leaderboard is navigated to, switch to leaderboard fragment* fragment = new LeaderboardFragment();  
 break;  
 case (R.id.*page\_4*):  
 *//when settings is navigated to, switch to settings fragment* fragment = new SettingsFragment();  
 break;  
 }  
 *//replacing current fragment with new fragment* getSupportFragmentManager().beginTransaction().replace(R.id.*fragmentContainer*, fragment).commit();  
 return true;  
 }  
 });  
  
 }  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 }  
}

#### RunningActivity.java

package com.example.exercisetracker.activities;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.content.res.Resources;  
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.speech.tts.TextToSpeech;  
import android.text.Html;  
import android.view.View;  
import android.widget.Button;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.Route;  
import com.example.exercisetracker.other.User;  
import com.example.exercisetracker.stepcounting.StepCounter;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.sql.Date;  
import java.sql.Timestamp;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Locale;  
import java.util.Objects;  
import java.util.Random;  
  
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;  
 private Button startStopBtn;  
 *//Specialised running variables* private float MET;  
 private double distance;  
 private Boolean isRunning;  
 private Integer seconds;  
 private Integer steps;  
 private Integer calories;  
 private Route route;  
 private String timeStarted;  
 private Date date;  
 private double pace;  
 private StepCounter stepCounter;  
  
 *//audio* private TextToSpeech tts;  
 private String[] quotes;  
 private Boolean isAudio;  
 private DecimalFormat df;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* Objects.*requireNonNull*(getSupportActionBar()).hide();  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* }  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 getWindow().setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 setContentView(R.layout.*activity\_running*);  
 init();  
 handlePermissions();  
 }  
  
  
 private void init() {  
 *//instantiating all private variables* long millis = System.*currentTimeMillis*();  
 Timestamp timestamp = new Timestamp(millis);  
 timeStarted = timestamp.toString().substring(11, 16);  
 date = new Date(millis);  
  
 *//running variables* isRunning = true;  
 seconds = 0;  
 steps = 0;  
 distance = 0f;  
 calories = 0;  
 isAudio = true;  
 *//text views* 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*));  
  
 *//buttons* startStopBtn = findViewById(R.id.*startStopBtn*);  
 *//Buttons* Button finishBtn = findViewById(R.id.*finishBtn*);  
 ImageButton audioBtn = findViewById(R.id.*audioBtn*);  
 audioBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isAudio) {  
 *//user requesting audio is switched off* audioBtn.setImageResource(R.drawable.*noaudio*);  
 isAudio = false;  
 } else {  
 *//user requesting audio switched on* audioBtn.setImageResource(R.drawable.*audio*);  
 isAudio = true;  
 }  
 }  
 });  
  
 *//click listeners* finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finishRunning();  
 }  
 });  
 startStopBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//when start/stop button is clicked* if (isRunning) {  
 *//if was not paused, pause the tracker  
 //set startstop button to "resume"* startStopBtn.setText("Resume");  
 isRunning = false;  
  
 } else {  
 *//if was paused, restart the tracker  
 //set start/stop button to "Pause"* startStopBtn.setText("Pause");  
 isRunning = true;  
 }  
 }  
 });  
  
  
 df = new DecimalFormat("#.##");  
  
 *//quotes from resources* Resources res = getResources();  
 quotes = res.getStringArray(R.array.*quotes*);  
  
 *//text to speech instantiation* tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 *// if No error is found then only it will run* if (i != TextToSpeech.*ERROR*) {  
 *// To Choose language of speech* tts.setLanguage(Locale.*UK*);  
 }  
 }  
 });  
  
 *//CUSTOM JAVA CLASSES* stepCounter = new StepCounter(this, 2, 0.1f, -10f, 10f, new DecimalFormat("#.##"));  
 ArrayList<Double[]> currentRoute = new ArrayList<>();  
 route = new Route(currentRoute);  
  
 }  
  
 @SuppressLint("MissingPermission")  
 private void startRunning() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 *//requesting background permission for android q+  
 //Android forces you to request this separately* isRunning = false;  
 requestPermissions(new String[]{Manifest.permission.*ACCESS\_BACKGROUND\_LOCATION*}, 1);  
 }  
  
  
 *//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* pace = location.getSpeed();  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 Double[] entry = {latitude, longitude};  
 System.*out*.println(Arrays.*toString*(entry));  
 *//adding entry to route* route.addRoute(entry);  
 }  
 }  
  
 };  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 1000, 3, locationListener);  
 createTimer();  
  
 *//2d arrays to store a variable amount of samples, each sample consisting of the x y z values* listener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 if (isRunning) {  
 Sensor sensor = event.sensor;  
 if (sensor.getType() == Sensor.*TYPE\_LINEAR\_ACCELERATION*) {  
 *//getting values from accelerometer* stepCounter.addEntry(0, event.values[0], event.values[1], event.values[2]);  
 } else if (sensor.getType() == Sensor.*TYPE\_GRAVITY* & isRunning) {  
 *//getting values from gravimeter* stepCounter.addEntry(1, event.values[0], event.values[1], event.values[2]);  
 }  
 *//PROCESSING DATA* if ((seconds % 5) == 0 && (!stepCounter.isEmpty())) {  
 stepCounter.countSteps();  
 steps = stepCounter.getSteps();  
 }  
 }  
 }  
  
 @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*);  
 }  
  
 *//saving to csv file for debugging* private void saveToStorage(Float[] filtered\_data) {  
 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();  
 }  
 }  
 }  
  
 private void handleQuotes() {  
 if (isAudio) {  
 *//starting with random quote for text to speech* Random r = new Random();  
 int currquote = r.nextInt(quotes.length);  
 if (seconds % 60 == 0) {  
 *//every 60 seconds a quote is spoken to help motivate the user* if (currquote > quotes.length) {  
 *//moving to front of quote array* currquote = 0;  
 }  
 *//speaking quote, and moving to next quote* tts.speak(quotes[currquote], TextToSpeech.*QUEUE\_FLUSH*, null);  
 currquote++;  
 }  
 }  
 }  
  
 private void createTimer() {  
 *//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* DecimalFormat df = new DecimalFormat("#.##");  
 if (route.getRouteSize() >= 2) {  
 route.calculateDistance();  
 distance = route.getDistance();  
 }  
 updateViews();  
 *//allowing preprocessing to happen at the instance of a 5 second interval* if ((seconds % 5) == 0) {  
 stepCounter.setHasProcessed(Boolean.*FALSE*);  
 }  
 *//updating audio* handleQuotes();  
  
  
 }  
  
 }  
 });  
 }  
  
 private void updateViews() {  
 *//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*(Locale.*getDefault*(), "Calories:\n%d", calories));  
 *//changing step text view* stepText.setText(String.*format*(Locale.*getDefault*(), "Steps:\n%d", steps));  
 distText.setText(String.*format*("Distance:\n%sm", df.format(distance)));  
 *//changing pace text view  
// paceText.setText(Html.fromHtml("Pace:\n" + df.format(distance / seconds.floatValue()) + "ms<sup>-1</sup"));* paceText.setText(Html.*fromHtml*("Pace:\n" + df.format(pace) + "ms<sup>-1</sup"));  
 }  
  
 private void handlePermissions() {  
 *//HANDLING PERMISSIONS  
 //Permissions* String[] PERMISSIONS = new String[]{  
 Manifest.permission.*ACCESS\_COARSE\_LOCATION*,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
  
 if (checkPermissions(this, PERMISSIONS) == Boolean.*FALSE*) {  
 *//dealt with overriding onRequestPermissionsResult method* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 requestPermissions(PERMISSIONS, 0);  
 }  
 } else {  
 startRunning();  
 }  
 }  
  
 *//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;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
  
 switch (requestCode) {  
 case 0:  
 *//case 0 applies to all android devices* isRunning = false;  
 if (grantResults.length > 0) {  
 *//checking if all permissions are granted on UI dialog* boolean granted = true;  
 for (int result : grantResults) {  
 if (result == PackageManager.*PERMISSION\_DENIED*) {  
 granted = false;  
 }  
 }  
 if (granted) {  
 isRunning = true;  
 startRunning();  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 finishRunning();  
 }  
 break;  
 }  
  
 case 1:  
 *//case 1 applies to when android is Q+ (as background location has to be seperately asked)* isRunning = false;  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 isRunning = true;  
 return;  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 finishRunning();  
 }  
 break;  
 }  
  
 }  
  
 private void finishRunning() {  
 isRunning = false;  
 sensorManager.unregisterListener(listener);  
  
 if (locationManager != null && timeStarted != null) {  
 locationManager.removeUpdates(locationListener);  
 *//exiting the running activity and saving data to database  
 //will only save activities which last longer than 60s* if (seconds >= 60 && steps>=10) {  
 DBhelper helper = new DBhelper(RunningActivity.this);  
 if (isAudio) {  
 *//audio text to speech to congratulate user* tts.speak(String.*format*(Locale.*getDefault*(), "Congratulations, you burnt %d calories and ran %d steps, a total distance of %f. See you next time!", calories, steps, distance),  
 TextToSpeech.*QUEUE\_FLUSH*, null);  
 }  
 if (helper.saveActivity("running", date.toString(), timeStarted, seconds.toString(), calories.toString(), steps.toString(), String.*valueOf*(Math.*round*(distance)), null)) {  
 Toast.*makeText*(RunningActivity.this, "Save successful", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(RunningActivity.this, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//saves space and resources on database* Toast.*makeText*(RunningActivity.this, "Activity too short, save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }  
 finish();  
  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 }  
  
 @Override  
 protected void onPostResume() {  
 super.onPostResume();  
 }  
}

#### WalkingActivity.java

package com.example.exercisetracker.activities;  
  
import static com.example.exercisetracker.other.BaseApp.*CHANNEL\_1\_ID*;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.app.Notification;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.content.res.Resources;  
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.Handler;  
import android.speech.tts.TextToSpeech;  
import android.text.Html;  
import android.view.View;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.Route;  
import com.example.exercisetracker.other.User;  
import com.example.exercisetracker.stepcounting.StepCounter;  
import com.google.android.material.button.MaterialButton;  
  
import java.sql.Date;  
import java.sql.Timestamp;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Locale;  
import java.util.Random;  
  
public class WalkingActivity 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 walking variables* private float MET;  
 private double distance;  
 private Boolean isWalking, isAudio;  
 private Integer seconds;  
 private Integer steps;  
 private Integer calories;  
 private Route route;  
 private String timeStarted;  
 private Date date;  
 private StepCounter stepCounter;  
 private Float pace;  
  
 *//audio* private TextToSpeech tts;  
 private String[] quotes;  
  
 *//Permissions* private String[] PERMISSIONS;  
 private ActivityResultLauncher<String> requestPermissionLauncher;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* getSupportActionBar().hide();  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* }  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 getWindow().setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 setContentView(R.layout.*activity\_walking*);  
  
 init();  
  
 if (checkPermissions(this, PERMISSIONS) == Boolean.*FALSE*) {  
 *//dealt with overriding onRequestPermissionsResult method* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 requestPermissions(PERMISSIONS, 0);  
 }  
 } else {  
 startWalking();  
 }  
 }  
  
 private void init() {  
 *//instantiating all private variables* long millis = System.*currentTimeMillis*();  
 Timestamp timestamp = new Timestamp(millis);  
 timeStarted = timestamp.toString().substring(11, 16);  
  
 date = new Date(millis);  
 isWalking = true;  
 isAudio = true;  
 seconds = 0;  
 steps = 0;  
 pace = 0f;  
 distance = 0f;  
 calories = 0;  
  
 *//text views* 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\_walking*));  
  
 *//quotes from resources* Resources res = getResources();  
 quotes = res.getStringArray(R.array.*quotes*);  
  
 ImageButton audioBtn = findViewById(R.id.*audioBtn*);  
 audioBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isAudio) {  
 *//user requesting audio is switched off* audioBtn.setImageResource(R.drawable.*noaudio*);  
 isAudio = false;  
 } else {  
 *//user requesting audio switched on* audioBtn.setImageResource(R.drawable.*audio*);  
 isAudio = true;  
 }  
 }  
 });  
  
 *//text to speech instantiation* tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 *// if No error is found then only it will run* if (i != TextToSpeech.*ERROR*) {  
 *// To Choose language of speech* tts.setLanguage(Locale.*UK*);  
 }  
 }  
 });  
  
 *//CUSTOM JAVA CLASSES* stepCounter = new StepCounter(this, 2, 0.1f, -10f, 10f, new DecimalFormat("#.##"));  
 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*);  
 finishBtn = findViewById(R.id.*finishBtn*);  
  
 *//HANDLING PERMISSIONS* PERMISSIONS = new String[]{  
 Manifest.permission.*ACCESS\_COARSE\_LOCATION*,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
 }  
  
 @SuppressLint("MissingPermission")  
 private void startWalking() {  
 *//click listeners* finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finishWalking();  
 }  
 });  
 startStopBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isWalking) {  
 startStopBtn.setText("Resume");  
 isWalking = false;  
  
 } else {  
 startStopBtn.setText("Pause");  
 isWalking = true;  
 }  
 }  
 });  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 *//requesting background permission for android q+  
 //Android forces you to request this separately* isWalking = false;  
 requestPermissions(new String[]{Manifest.permission.*ACCESS\_BACKGROUND\_LOCATION*}, 1);  
 }  
  
 *//handling location changes* locationManager = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*);  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(@NonNull Location location) {  
 if (isWalking) {  
 *//location in form of latitude and longitude* double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 Double[] entry = {latitude, longitude};  
 System.*out*.println(entry);  
 route.addRoute(entry);  
 pace = location.getSpeed();  
 }  
 }  
  
 };  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 1000, 3, locationListener);  
  
 createTimer();  
 *//2d arrays to store a variable amount of samples, each sample consisting of the x y z values* listener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 if (isWalking) {  
 Sensor sensor = event.sensor;  
 if (sensor.getType() == Sensor.*TYPE\_LINEAR\_ACCELERATION*) {  
 *//getting values from accelerometer* stepCounter.addEntry(0, event.values[0], event.values[1], event.values[2]);  
 } else if (sensor.getType() == Sensor.*TYPE\_GRAVITY* & isWalking) {  
 *//getting values from gravimeter* stepCounter.addEntry(1, event.values[0], event.values[1], event.values[2]);  
 }  
 *//PROCESSING DATA* if ((seconds % 5) == 0 && (!stepCounter.isEmpty())) {  
 stepCounter.countSteps();  
 steps = stepCounter.getSteps();  
 }  
 }  
 }  
  
 @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;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 switch (requestCode) {  
 case 0:  
 *//case 0 applies to all android devices* if (grantResults.length > 0) {  
 *//checking if all permissions are granted on UI dialog* boolean granted = true;  
 for (int result : grantResults) {  
 if (result == PackageManager.*PERMISSION\_DENIED*) {  
 granted = false;  
 }  
 }  
 if (granted) {  
 startWalking();  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 finishWalking();  
 }  
 return;  
 }  
  
 case 1:  
 *//case 1 applies to when android is Q+ (as background location has to be seperately asked)* if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 isWalking = true;  
 return;  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 finishWalking();  
 }  
 }  
  
 }  
  
 private void finishWalking() {  
 isWalking = false;  
 sensorManager.unregisterListener(listener);  
 if (locationManager != null && timeStarted != null) {  
 locationManager.removeUpdates(locationListener);  
 *//exiting the walking activity and saving data to database  
 //will only save activities which last longer than 60s* if (seconds >= 60 && steps>=10) {  
 if (isAudio) {  
 *//audio text to speech to congratulate user* tts.speak(String.*format*(Locale.*getDefault*(), "Congratulations, you burnt %d calories and walked %d steps. See you next time!", calories, steps), TextToSpeech.*QUEUE\_FLUSH*, null);  
 }  
 DBhelper helper = new DBhelper(WalkingActivity.this);  
 if (helper.saveActivity("walking", date.toString(), timeStarted, seconds.toString(), calories.toString(), steps.toString(), String.*valueOf*(Math.*round*(distance)), null)) {  
 Toast.*makeText*(WalkingActivity.this, "Save successful", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(WalkingActivity.this, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//saves space and resources on database* Toast.*makeText*(WalkingActivity.this, "Activity too short, save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }  
 this.finish();  
  
 }  
  
 private void createTimer() {  
 *//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 (isWalking) {  
 seconds++;  
 *//calculating distances between location updates and updating text views* DecimalFormat df = new DecimalFormat("#.##");  
 if (route.getRouteSize() >= 2) {  
 route.calculateDistance();  
 distance = route.getDistance();  
 }  
 *//allowing preprocessing to happen at the instance of a 5 second interval* if ((seconds % 5) == 0) {  
 stepCounter.setHasProcessed(Boolean.*FALSE*);  
 }  
 updateViews(df);  
 *//updating notification every second* handleQuotes();  
 }  
  
 }  
 });  
 }  
  
 private void handleQuotes() {  
 if (isAudio) {  
 *//starting with random quote for text to speech* Random r = new Random();  
 int currquote = r.nextInt(quotes.length);  
 if (seconds % 60 == 0) {  
 *//every 60 seconds a quote is spoken to help motivate the user* if (currquote > quotes.length) {  
 *//moving to front of quote array* currquote = 0;  
 }  
 *//speaking quote, and moving to next quote* tts.speak(quotes[currquote], TextToSpeech.*QUEUE\_FLUSH*, null);  
 currquote++;  
 }  
 }  
 }  
  
 private void updateViews(DecimalFormat df) {  
 *//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*(Locale.*getDefault*(), "Calories:\n%d", calories));  
 *//changing step text view* stepText.setText(String.*format*(Locale.*getDefault*(), "Steps:\n%d", steps));  
 distText.setText(String.*format*("Distance:\n%sm", df.format(distance)));  
 *//changing pace text view* paceText.setText(Html.*fromHtml*("Pace:\n" + df.format(pace) + "ms<sup>-1</sup"));  
 }  
  
 *//handling live notification bar* public void sendOnChannel1(View v) {  
 Notification notification = new NotificationCompat.Builder(this, *CHANNEL\_1\_ID*)  
 .setSmallIcon(R.id.*icon*)  
 .setContentTitle("Walking Tracking")  
 .setContentText(String.*valueOf*(steps))  
 .setCategory(NotificationCompat.*CATEGORY\_WORKOUT*)  
 .build();  
 notificationManagerCompat.notify(1, notification);  
 }  
}

#### TreadmillActivity.java

package com.example.exercisetracker.activities;  
  
import static com.example.exercisetracker.other.BaseApp.*CHANNEL\_1\_ID*;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.app.Notification;  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.content.res.Resources;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Handler;  
import android.speech.tts.TextToSpeech;  
import android.text.Html;  
import android.view.View;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.annotation.NonNull;  
import androidx.annotation.RequiresApi;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
import com.example.exercisetracker.stepcounting.StepCounter;  
import com.google.android.material.button.MaterialButton;  
  
import java.sql.Date;  
import java.sql.Timestamp;  
import java.text.DecimalFormat;  
import java.util.Locale;  
import java.util.Objects;  
import java.util.Random;  
  
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 Boolean isRunning, isAudio;  
 private Integer seconds;  
 private Integer steps;  
 private Integer calories;  
 private Integer height;  
 private String timeStarted;  
 private Date date;  
 private StepCounter stepCounter;  
  
 *//audio* private TextToSpeech tts;  
 private String[] quotes;  
  
  
 private ActivityResultLauncher<String> requestPermissionLauncher;  
  
  
 @SuppressLint("MissingPermission")  
 @RequiresApi(api = Build.VERSION\_CODES.*M*)  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 Objects.*requireNonNull*(getSupportActionBar()).hide();  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 getWindow().getDecorView().setSystemUiVisibility(View.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);*// set status text dark* }  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 getWindow().setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 setContentView(R.layout.*activity\_treadmill*);  
  
 init();  
 handlePermissions();  
 }  
  
 private void init() {  
 *//instantiating all private variables* seconds = 0;  
 steps = 0;  
 distance = 0f;  
 height = User.*getHeight*();  
 MET = Float.*parseFloat*(getString(R.string.*met\_treadmill*));  
 isAudio = true;  
  
 long millis = System.*currentTimeMillis*();  
 Timestamp timestamp = new Timestamp(millis);  
 timeStarted = timestamp.toString().substring(11, 16);  
 date = new java.sql.Date(millis);  
  
 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*);  
  
 *//quotes from resources* Resources res = getResources();  
 quotes = res.getStringArray(R.array.*quotes*);  
  
  
 *//CUSTOM JAVA CLASSES* stepCounter = new StepCounter(this, 2, 0.1f, -10f, 10f, new DecimalFormat("#.##"));  
 *//NOTIFICATION MANAGER* notificationManagerCompat = NotificationManagerCompat.*from*(this);  
  
 *//text to speech instantiation* tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 *// if No error is found then only it will run* if (i != TextToSpeech.*ERROR*) {  
 *// To Choose language of speech* tts.setLanguage(Locale.*UK*);  
 }  
 }  
 });  
  
 *//handling when start and stop button clicked* startStopBtn = findViewById(R.id.*startStopBtn*);  
 finishBtn = findViewById(R.id.*finishBtn*);  
 ImageButton audioBtn = findViewById(R.id.*audioBtn*);  
 audioBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isAudio) {  
 *//user requesting audio is switched off* audioBtn.setImageResource(R.drawable.*noaudio*);  
 isAudio = false;  
 } else {  
 *//user requesting audio switched on* audioBtn.setImageResource(R.drawable.*audio*);  
 isAudio = true;  
 }  
 }  
 });  
  
 }  
  
 @SuppressLint("MissingPermission")  
 private void startRunning() {  
 isRunning = true;  
 finishBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//exiting the running activity and sending data back to main program* finishRunning();  
 }  
 });  
 startStopBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isRunning) {  
 startStopBtn.setText("Resume");  
 isRunning = false;  
  
 } else {  
 startStopBtn.setText("Pause");  
 isRunning = true;  
 }  
 }  
 });  
 createTimer();  
  
 listener = new SensorEventListener() {  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 if (isRunning) {  
 Sensor sensor = event.sensor;  
 if (sensor.getType() == Sensor.*TYPE\_LINEAR\_ACCELERATION*) {  
 *//getting values from accelerometer* stepCounter.addEntry(0, event.values[0], event.values[1], event.values[2]);  
 } else if (sensor.getType() == Sensor.*TYPE\_GRAVITY* & isRunning) {  
 *//getting values from gravimeter* stepCounter.addEntry(1, event.values[0], event.values[1], event.values[2]);  
 }  
 *//PROCESSING DATA* if ((seconds % 5) == 0 && (!stepCounter.isEmpty())) {  
 stepCounter.countSteps();  
 steps = stepCounter.getSteps();  
 }  
 }  
 }  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
 }  
 };  
 *//registering sensor listeners* 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*);  
 }  
  
 private void handleQuotes() {  
 if (isAudio) {  
 *//starting with random quote for text to speech* Random r = new Random();  
 int currquote = r.nextInt(quotes.length);  
 if (seconds % 60 == 0) {  
 *//every 60 seconds a quote is spoken to help motivate the user* if (currquote > quotes.length) {  
 *//moving to front of quote array* currquote = 0;  
 }  
 *//speaking quote, and moving to next quote* tts.speak(quotes[currquote], TextToSpeech.*QUEUE\_FLUSH*, null);  
 currquote++;  
 }  
 }  
 }  
  
  
 private void createTimer() {  
 *//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) {  
 *//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* seconds++;  
 distance = height.floatValue() \* ((float) Math.*floor*(steps / 2f)) \* 0.004f; *//0.004 as user height stored as cm* DecimalFormat df = new DecimalFormat("#.##");  
 updateViews(df);  
 *//allowing preprocessing to happen at the instance of a 5 second interval* if ((seconds % 5) == 0) {  
 stepCounter.setHasProcessed(Boolean.*FALSE*);  
 }  
 handleQuotes();  
  
 }  
  
 }  
 });  
 }  
  
 private void updateViews(DecimalFormat df) {  
 *//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*(Locale.*getDefault*(), "Calories:\n%d", calories));  
 *//changing step text view* stepText.setText(String.*format*(Locale.*getDefault*(), "Steps:\n%d", steps));  
 distText.setText(String.*format*("Distance:\n%sm", df.format(distance)));  
 *//changing pace text view* paceText.setText(Html.*fromHtml*("Pace:\n" + df.format(distance / seconds.floatValue()) + "ms<sup>-1</sup"));  
 }  
  
 private void handlePermissions() {  
 *//HANDLING PERMISSIONS  
 //Permissions* String[] PERMISSIONS = new String[]{  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 };  
  
 if (checkPermissions(this, PERMISSIONS) == Boolean.*FALSE*) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 requestPermissions(PERMISSIONS, 0);  
 }  
 } else {  
 startRunning();  
 }  
 }  
  
 *//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;  
 }  
  
 private void finishRunning() {  
 isRunning = false;  
 sensorManager.unregisterListener(listener);  
  
 *//exiting the running activity and saving data to database* if (seconds >= 60 && steps>=10) {  
 if (isAudio) {  
 *//audio text to speech to congratulate user* tts.speak(String.*format*(Locale.*getDefault*(), "Congratulations, you burnt %d calories. See you next time!", calories), TextToSpeech.*QUEUE\_FLUSH*, null);  
 }  
 DBhelper helper = new DBhelper(TreadmillActivity.this);  
 if (helper.saveActivity("treadmill", date.toString(), timeStarted, seconds.toString(), calories.toString(), steps.toString(), String.*valueOf*(Math.*round*(distance)), null)) {  
 Toast.*makeText*(TreadmillActivity.this, "Save successful", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(TreadmillActivity.this, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//saves space and resources on database* Toast.*makeText*(TreadmillActivity.this, "Activity too short (less than 1 min), save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 this.finish();  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == 0) {  
 if (grantResults.length > 0) {  
 *//checking if all permissions are granted on UI dialog* boolean granted = true;  
 for (int result : grantResults) {  
 if (result == PackageManager.*PERMISSION\_DENIED*) {  
 granted = false;  
 break;  
 }  
 }  
 if (granted) {  
 startRunning();  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 finishRunning();  
 }  
 }  
 }  
 }  
  
  
 *//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.activities;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.pm.PackageManager;  
import android.content.res.Resources;  
import android.graphics.Point;  
import android.graphics.Rect;  
import android.media.Image;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Handler;  
import android.speech.tts.TextToSpeech;  
import android.util.Size;  
import android.view.Display;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.CameraSelector;  
import androidx.camera.core.ImageAnalysis;  
import androidx.camera.core.ImageProxy;  
import androidx.camera.core.Preview;  
import androidx.camera.lifecycle.ProcessCameraProvider;  
import androidx.camera.view.PreviewView;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationManagerCompat;  
import androidx.core.content.ContextCompat;  
import androidx.lifecycle.LifecycleOwner;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
import com.example.exercisetracker.repDetection.RepCounter;  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.OnFailureListener;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.android.material.dialog.MaterialAlertDialogBuilder;  
import com.google.common.util.concurrent.ListenableFuture;  
import com.google.mlkit.vision.common.InputImage;  
import com.google.mlkit.vision.pose.Pose;  
import com.google.mlkit.vision.pose.PoseDetection;  
import com.google.mlkit.vision.pose.PoseDetector;  
import com.google.mlkit.vision.pose.PoseLandmark;  
import com.google.mlkit.vision.pose.defaults.PoseDetectorOptions;  
  
import java.sql.Timestamp;  
import java.text.DecimalFormat;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Locale;  
import java.util.Map;  
import java.util.Objects;  
import java.util.Random;  
import java.util.concurrent.ExecutionException;  
  
public class PushUpActivity extends AppCompatActivity {  
 *//Text Views* private TextView timerText,repText,calText,paceText;  
 *//buttons* private Button startBtn, finishBtn, helpBtn;  
 *//pushup custom variables* private Boolean isTracking, isAudio;  
 private java.sql.Date date;  
 private Integer seconds;  
 private String timeStarted;  
 private Float MET;  
 private Integer reps;  
 private Integer calories;  
 private PoseDetector poseDetector;  
 private Preview preview;  
 private PreviewView tv;  
 private CameraSelector cameraSelector;  
 private RepCounter repcounter;  
  
 *//audio* private TextToSpeech tts;  
 private String[] quotes;  
  
 *//notification* private NotificationManagerCompat notificationManagerCompat;  
  
 private Graphic graphic;  
 private Size displaySize;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* Objects.*requireNonNull*(getSupportActionBar()).hide();  
 Window w = getWindow();  
 w.setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 w.setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 w.setFlags(WindowManager.LayoutParams.*FLAG\_LAYOUT\_NO\_LIMITS*, WindowManager.LayoutParams.*FLAG\_LAYOUT\_NO\_LIMITS*);  
 Rect rectangle = new Rect();  
 getWindow().getDecorView().getWindowVisibleDisplayFrame(rectangle);  
 int statusBarHeight = rectangle.top;  
 setContentView(R.layout.*activity\_pushup*);  
  
 init();  
  
 *//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* finishTracking();  
 }  
 });  
 helpBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//when help button is clicked, show dialog to user explaining how to user  
 //how app can track squats and what user should do* String title = "How does this work?";  
 String message = "This app uses Google's machine learning kit to track your movements.\n" +  
 "Place your device on a flat stable surface (such as the floor) where your full body is in front view of the camera.\n" +  
 "As you push down and up, the app will detect your reps and calories!";  
 new MaterialAlertDialogBuilder(PushUpActivity.this)  
 .setTitle(title)  
 .setMessage(message).setPositiveButton("Dismiss", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 }).show();  
 }  
 });  
 *//when audio button is clicked* ImageButton audioBtn = findViewById(R.id.*audioBtn*);  
 audioBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isAudio) {  
 *//user requesting audio is switched off* audioBtn.setImageResource(R.drawable.*noaudio*);  
 isAudio = false;  
 } else {  
 *//user requesting audio switched on* audioBtn.setImageResource(R.drawable.*audio*);  
 isAudio = true;  
 }  
 }  
 });  
 *//permissions* String[] PERMISSIONS = new String[]{  
 Manifest.permission.*INTERNET*,  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE* };  
  
 if (checkPermissions(this, PERMISSIONS) == Boolean.*FALSE*) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 *//dealt with overriding onRequestPermissionsResult method* requestPermissions(PERMISSIONS, 0);  
 }  
 } else {  
 startTracking();  
 }  
 }  
  
 private void init() {  
 *//instantiating all variables  
 //getting display size for graphic* Display display = getWindowManager().getDefaultDisplay();  
 Point temp = new Point();  
 display.getSize(temp);  
 displaySize = new Size(temp.x, temp.y);  
  
 isAudio = true;  
 isTracking = true;  
 seconds = 0;  
 calories = 0;  
 reps = 0;  
  
 helpBtn = findViewById(R.id.*helpBtn*);  
 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*);  
 paceText = findViewById(R.id.*paceText*);  
 TextView poseIndicatorTV = findViewById(R.id.*PoseIndicator*);  
 *//getting met from string values* MET = Float.*parseFloat*(getString(R.string.*met\_pushup*));  
 tv = findViewById(R.id.*tv*);  
 TextView debug = findViewById(R.id.*debugTV*);  
 *//min distance is by a fifth of the screen height  
 //uncertainty is 1/10 of the screen height* repcounter = new RepCounter(this, 0, poseIndicatorTV, debug, displaySize.getHeight() / 6f);  
  
 *//getting current date and time* long millis = System.*currentTimeMillis*();  
 Timestamp timestamp = new Timestamp(millis);  
 timeStarted = timestamp.toString().substring(11, 16);  
 date = new java.sql.Date(millis);  
  
 *//quotes from resources* Resources res = getResources();  
 quotes = res.getStringArray(R.array.*quotes*);  
  
 *//text to speech instantiation* tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 *// if No error is found then only it will run* if (i != TextToSpeech.*ERROR*) {  
 *// To Choose language of speech* tts.setLanguage(Locale.*UK*);  
 }  
 }  
 });  
  
 }  
  
 private void startTracking() {  
 *//NOTIFICATION MANAGER* notificationManagerCompat = NotificationManagerCompat.*from*(this);  
  
 createTimer();  
 handleCamera();  
 *//MLK tracking, instantiating pose detector* PoseDetectorOptions options = new PoseDetectorOptions.Builder()  
 .setDetectorMode(PoseDetectorOptions.*STREAM\_MODE*)  
 .build();  
 poseDetector = PoseDetection.*getClient*(options);  
 }  
  
 private void createTimer() {  
 final Handler handler = new Handler();  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 handler.postDelayed(this, 1000);  
 *//updating live notification every second  
 //sendOnChannel1();* 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);  
 calories = Math.*round*(MET \* User.*getWeight*() \* (seconds.floatValue() / 3600));  
 reps = repcounter.getReps();  
 *//updating text views* timerText.setText(time);  
 calText.setText("Calories:\n" + calories.toString());  
 repText.setText("Reps:\n" + reps.toString());  
 *//calculating average pace of reps* DecimalFormat df = new DecimalFormat("#.##");  
 paceText.setText("Pace:\n" + df.format((float)reps\*60f/(float)seconds)+ "\nreps/min");  
 handleQuotes();  
 }  
  
  
 }  
 });  
 }  
  
 private void handleQuotes() {  
 if (isAudio) {  
 *//starting with random quote for text to speech* Random r = new Random();  
 int currquote = r.nextInt(quotes.length);  
 if (seconds % 60 == 0) {  
 *//every 60 seconds a quote is spoken to help motivate the user* if (currquote > quotes.length) {  
 *//moving to front of quote array* currquote = 0;  
 }  
 *//speaking quote, and moving to next quote* tts.speak(quotes[currquote], TextToSpeech.*QUEUE\_FLUSH*, null);  
 currquote++;  
 }  
 }  
 }  
  
 private void handleCamera() {  
 final ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.*getInstance*(this);  
 *//getting display size (dependent on device)* ImageAnalysis imageAnalysis =  
 new ImageAnalysis.Builder()  
 .setTargetResolution(displaySize)  
 .setBackpressureStrategy(ImageAnalysis.*STRATEGY\_KEEP\_ONLY\_LATEST*)  
 .build();  
 *//setting the configuration for the image analysis* imageAnalysis.setAnalyzer(ContextCompat.*getMainExecutor*(PushUpActivity.this), new ImageAnalysis.Analyzer() {  
 @Override  
 public void analyze(@NonNull ImageProxy imageProxy) {  
  
 int rotationDegrees = imageProxy.getImageInfo().getRotationDegrees();  
 @SuppressLint("UnsafeOptInUsageError") Image image = imageProxy.getImage();  
 if (image != null) {  
 *//receiving the input image from camera* InputImage inputimage = InputImage.*fromMediaImage*(image, rotationDegrees);  
 Task<Pose> result = poseDetector.process(inputimage).addOnSuccessListener(new OnSuccessListener<Pose>() {  
 @Override  
 public void onSuccess(@NonNull Pose pose) {  
 if (isTracking) {  
 *//when the pose detector successfully can attach to image  
 //Receiving and processing landmarks from Google's ML kit software* List<PoseLandmark> allPoseLandmarks = pose.getAllPoseLandmarks();  
 processLandmarks(allPoseLandmarks);  
 *//drawing on the landmarks onto the user's screen* graphic.drawGraphic(allPoseLandmarks);  
 } else {  
 graphic.clearGraphic();  
 }  
  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 *//when the pose detector cannot attach to image* System.*out*.println("Failed Pose Detection");  
 }  
 }).addOnCompleteListener(new OnCompleteListener<Pose>() {  
 @Override  
 public void onComplete(@NonNull Task<Pose> task) {  
 *//making sure to close the instance of the image to allow the next image to be processed* imageProxy.close();  
 }  
 });  
 }  
 }  
 });  
  
 *//attaching the image analysis object to the camera* cameraProviderFuture.addListener(() -> {  
 try {  
 *//configuring camera to preview to display on the user's screen* ProcessCameraProvider provider = cameraProviderFuture.get();  
 preview = new Preview.Builder().build();  
 preview.setSurfaceProvider(tv.getSurfaceProvider());  
 cameraSelector = new CameraSelector.Builder()  
 .requireLensFacing(CameraSelector.*LENS\_FACING\_FRONT*)  
 .build();  
 try {  
 *//binding the camera, preview and analyser together* provider.unbindAll();  
 provider.bindToLifecycle((LifecycleOwner) this, cameraSelector, preview, imageAnalysis);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 } catch (ExecutionException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }, ContextCompat.*getMainExecutor*(this));  
 *//creating graphic object to deal with drawing on landmarks onto the preview* graphic = new Graphic(displaySize);  
  
 }  
  
 private void processLandmarks(List<PoseLandmark> allLandmarks) {  
 *//method to deal with analyzing the landmarks in a particular instance, provided by ML Kit* if (!allLandmarks.isEmpty() && allLandmarks.get(PoseLandmark.*NOSE*).getInFrameLikelihood() > 0.5f  
 && allLandmarks.get(PoseLandmark.*LEFT\_HIP*).getInFrameLikelihood() > 0.5f &&  
 allLandmarks.get(PoseLandmark.*LEFT\_KNEE*).getInFrameLikelihood() > 0.5f) {  
 repcounter.addEntry(allLandmarks);  
 }  
 }  
  
 *// --------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;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 isTracking = false;  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == 0) {*//checking if all permissions are granted on UI dialog* boolean granted = true;  
 for (int result : grantResults) {  
 if (result == PackageManager.*PERMISSION\_DENIED*) {  
 granted = false;  
 }  
 }  
 if (granted) {  
 isTracking = true;  
 startTracking();  
 } else {  
 Toast.*makeText*(this, "Permissions Denied\nPlease allow permissions in settings", Toast.*LENGTH\_SHORT*).show();  
 this.finish();  
 }  
 }  
 }  
  
 private void finishTracking() {  
 *//handles the safe closing of the activity, and presenting any information to the user* isTracking = false;  
  
 if (seconds >=60 && reps>=5) {  
 if (isAudio) {  
 *//audio text to speech to congratulate user* tts.speak(String.*format*(Locale.*getDefault*(), "Congratulations, you burnt %d calories and did %d reps. See you next time!", calories, reps),  
 TextToSpeech.*QUEUE\_FLUSH*, null);  
 }  
 *//saving activity results to database, as long as activity lasted for more than a minute* DBhelper helper = new DBhelper(PushUpActivity.this);  
 if (helper.saveActivity("pushup", date.toString(), timeStarted, seconds.toString(), calories.toString(), null, null, reps.toString())) {  
 Toast.*makeText*(PushUpActivity.this, "Save successful", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(PushUpActivity.this, "Save unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//saves space and resources on database* Toast.*makeText*(PushUpActivity.this, "Activity Not Saved (Too Short)", Toast.*LENGTH\_SHORT*).show();  
 }  
 *//destroying notification* notificationManagerCompat.cancel(1);  
 this.finish();  
 }  
  
 *//<---------Graphics---------->* private class Graphic {  
 *//composition class (cannot draw graphic without the push up activity  
  
 //hash map, associating landmark name to graphic View* private final Map<String, View> graphicViewsMap;  
 private final Size displaySize;  
 private float scalex;  
 private float scaley;  
 private int yoffset;  
 private int xoffset;  
 *//n pixels to offset in order to fit scaled up image on view* private Graphic(Size displaySize) {  
 *//HASH MAP to store all views corresponding to landmarks* graphicViewsMap = new HashMap<String, View>();  
 graphicViewsMap.put("nose", (View) findViewById(R.id.*nose*));  
 graphicViewsMap.put("left\_shoulder", (View) findViewById(R.id.*left\_shoulder*));  
 graphicViewsMap.put("right\_shoulder", (View) findViewById(R.id.*right\_shoulder*));  
 graphicViewsMap.put("left\_elbow", (View) findViewById(R.id.*lelbow*));  
 graphicViewsMap.put("right\_elbow", (View) findViewById(R.id.*relbow*));  
 graphicViewsMap.put("left\_hip", (View) findViewById(R.id.*left\_hip*));  
 graphicViewsMap.put("right\_hip", (View) findViewById(R.id.*right\_hip*));  
 graphicViewsMap.put("left\_knee", (View) findViewById(R.id.*left\_knee*));  
 graphicViewsMap.put("right\_knee", (View) findViewById(R.id.*right\_knee*));  
 this.displaySize = displaySize;  
 }  
  
*// private int gcd(int p, int q){  
// //euclid's algorithm to find smallest possible ratio  
// if (q == 0) return p;  
// else return gcd(q, p % q);  
// }  
// private int ratio(int a, int b) {  
// final int gcd = gcd(a,b);  
// return (b/gcd);  
// }  
//  
// private void setScaleFactor(Point displaySize, Size imageSize){  
// Toast.makeText(PushUpActivity.this, String.format("Ratio of y: %d",ratio(displaySize.x,displaySize.y)), Toast.LENGTH\_SHORT).show();  
// Toast.makeText(PushUpActivity.this, String.format("Display size: %d, %d",displaySize.x,displaySize.y), Toast.LENGTH\_SHORT).show();  
// scalex = 2.25f;  
// scaley = 2.25f;  
//// int[] arr = new int[2];  
//// tv.getLocationInWindow(arr);  
//// yoffset = arr[1];  
// }* private void updateLandmarkGraphic(String name, Float x, Float y) {  
 View view = graphicViewsMap.get(name);  
 view.setVisibility(View.*VISIBLE*);  
 *//inverting x coordinate, as camera is in mirroring position* view.setX(displaySize.getWidth() - x);  
 view.setY(y);  
 }  
  
 private void clearGraphic() {  
 for (View view : graphicViewsMap.values()) {  
 view.setVisibility(View.*GONE*);  
 }  
 }  
  
 private void drawGraphic(List<PoseLandmark> allLandmarks) {  
 if (allLandmarks.isEmpty()) {  
 *//if no landmarks are detected, remove points from graphic* clearGraphic();  
  
 }  
 *//drawing points on the preview, corresponding to where ML Kit has detected the positions  
 //of the body parts* for (PoseLandmark landmark : allLandmarks) {  
 switch (landmark.getLandmarkType()) {  
 case PoseLandmark.*LEFT\_SHOULDER*:  
 updateLandmarkGraphic("left\_shoulder", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*RIGHT\_SHOULDER*:  
 updateLandmarkGraphic("right\_shoulder", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*NOSE*:  
 updateLandmarkGraphic("nose", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*LEFT\_ELBOW*:  
 updateLandmarkGraphic("left\_elbow", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*RIGHT\_ELBOW*:  
 updateLandmarkGraphic("right\_elbow", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*LEFT\_HIP*:  
 updateLandmarkGraphic("left\_hip", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*RIGHT\_HIP*:  
 updateLandmarkGraphic("right\_hip", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*LEFT\_KNEE*:  
 updateLandmarkGraphic("left\_knee", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.*RIGHT\_KNEE*:  
 updateLandmarkGraphic("right\_knee", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 }  
  
 }  
 }  
  
 }  
}

#### SquatsActivity.java

package com.example.exercisetracker.activities;  
  
import android.Manifest;  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.pm.PackageManager;  
import android.content.res.Resources;  
import android.graphics.Point;  
import android.graphics.Rect;  
import android.media.Image;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Handler;  
import android.speech.tts.TextToSpeech;  
import android.util.Size;  
import android.view.Display;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.CameraSelector;  
import androidx.camera.core.ImageAnalysis;  
import androidx.camera.core.ImageProxy;  
import androidx.camera.core.Preview;  
import androidx.camera.lifecycle.ProcessCameraProvider;  
import androidx.camera.view.PreviewView;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationManagerCompat;  
import androidx.core.content.ContextCompat;  
import androidx.lifecycle.LifecycleOwner;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
import com.example.exercisetracker.repDetection.RepCounter;  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.OnFailureListener;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.android.material.dialog.MaterialAlertDialogBuilder;  
import com.google.common.util.concurrent.ListenableFuture;  
import com.google.mlkit.vision.common.InputImage;  
import com.google.mlkit.vision.pose.Pose;  
import com.google.mlkit.vision.pose.PoseDetection;  
import com.google.mlkit.vision.pose.PoseDetector;  
import com.google.mlkit.vision.pose.PoseLandmark;  
import com.google.mlkit.vision.pose.defaults.PoseDetectorOptions;  
  
import java.sql.Timestamp;  
import java.text.DecimalFormat;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Locale;  
import java.util.Map;  
import java.util.Objects;  
import java.util.Random;  
import java.util.concurrent.ExecutionException;  
  
public class SquatsActivity extends AppCompatActivity {  
 *//Text Views* private TextView timerText,calText,repText,paceText;  
 *//buttons* private Button startBtn, finishBtn, helpBtn;  
 *//pushup custom variables* private Boolean isTracking, isAudio;  
 private java.sql.Date date;  
 private Integer seconds;  
 private String timeStarted;  
 private Float MET;  
 private Integer reps;  
 private Integer calories;  
 private PoseDetector poseDetector;  
 private Preview preview;  
 private PreviewView tv;  
 private CameraSelector cameraSelector;  
 private RepCounter repcounter;  
  
 *//audio* private TextToSpeech tts;  
 private String[] quotes;  
  
 *//notification* private NotificationManagerCompat notificationManagerCompat;  
  
 private Graphic graphic;  
 private Size displaySize;  
  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 *//visuals* Objects.requireNonNull(getSupportActionBar()).hide();  
 Window w = getWindow();  
 w.setNavigationBarColor(getResources().getColor(R.color.main\_colour));  
 w.setStatusBarColor(getResources().getColor(R.color.main\_colour));  
 w.setFlags(WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS, WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS);  
 Rect rectangle = new Rect();  
 getWindow().getDecorView().getWindowVisibleDisplayFrame(rectangle);  
 setContentView(R.layout.activity\_squats);  
 init();  
  
 *//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* finishTracking();  
 }  
 });  
 helpBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//when help button is clicked, show dialog to user explaining how to user  
 //how app can track squats and what user should do* String title = "How does this work?";  
 String message = "This app uses Google's machine learning kit to track your movements.\n" +  
 "Place your device on a flat stable surface where your body is in full frame of the camera.\n" +  
 "Avoid places where the camera is at a steep incline.\n" +  
 "As you squat, the app will detect your reps and calories!";  
 new MaterialAlertDialogBuilder(SquatsActivity.this)  
 .setTitle(title)  
 .setMessage(message).setPositiveButton("Dismiss", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 }).show();  
  
 }  
 });  
 *//when audio button is clicked* ImageButton audioBtn = findViewById(R.id.audioBtn);  
 audioBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (isAudio) {  
 *//user requesting audio is switched off* audioBtn.setImageResource(R.drawable.noaudio);  
 isAudio = false;  
 } else {  
 *//user requesting audio switched on* audioBtn.setImageResource(R.drawable.audio);  
 isAudio = true;  
 }  
 }  
 });  
 *//permissions* String[] PERMISSIONS = new String[]{  
 Manifest.permission.INTERNET,  
 Manifest.permission.CAMERA,  
 Manifest.permission.WRITE\_EXTERNAL\_STORAGE  
 };  
  
 if (checkPermissions(this, PERMISSIONS) == Boolean.FALSE) {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 *//dealt with overriding onRequestPermissionsResult method* requestPermissions(PERMISSIONS, 0);  
 }  
 } else {  
 startTracking();  
 }  
 super.onCreate(savedInstanceState);  
 }  
 private void init(){  
 *//instantiating all variables  
 //getting display size for graphic* Display display = getWindowManager().getDefaultDisplay();  
 Point temp = new Point();  
 display.getSize(temp);  
 displaySize = new Size(temp.x, temp.y);  
  
 isAudio = true;  
 isTracking = true;  
 seconds = 0;  
 calories = 0;  
 reps = 0;  
  
 startBtn = findViewById(R.id.startStopBtn);  
 finishBtn = findViewById(R.id.finishBtn);  
 helpBtn = findViewById(R.id.helpBtn);  
 timerText = findViewById(R.id.timerText);  
 repText = findViewById(R.id.repText);  
 calText = findViewById(R.id.calText);  
 paceText = findViewById(R.id.paceText);  
 TextView poseIndicatorTV = findViewById(R.id.PoseIndicator);  
  
 *//getting met from string values* MET = Float.parseFloat(getString(R.string.met\_squats));  
 tv = findViewById(R.id.tv);  
 TextView debug = findViewById(R.id.debugTV);  
 *//min distance is by a fifth of the screen height  
 //uncertainty is 1/10 of the screen height* repcounter = new RepCounter(this,1, poseIndicatorTV, debug, displaySize.getHeight() / 9f);  
  
 *//getting current date and time* long millis = System.currentTimeMillis();  
 Timestamp timestamp = new Timestamp(millis);  
 timeStarted = timestamp.toString().substring(11, 16);  
 date = new java.sql.Date(millis);  
  
 *//quotes from resources* Resources res = getResources();  
 quotes = res.getStringArray(R.array.quotes);  
  
 *//text to speech instantiation* tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {  
 @Override  
 public void onInit(int i) {  
 *// if No error is found then only it will run* if (i != TextToSpeech.ERROR) {  
 *// To Choose language of speech* tts.setLanguage(Locale.UK);  
 }  
 }  
 });  
 }  
  
 private void startTracking(){  
 *//NOTIFICATION MANAGER* notificationManagerCompat = NotificationManagerCompat.from(this);  
  
 createTimer();  
 handleCamera();  
 *//MLK tracking, instantiating pose detector* PoseDetectorOptions options = new PoseDetectorOptions.Builder()  
 .setDetectorMode(PoseDetectorOptions.STREAM\_MODE)  
 .build();  
 poseDetector = PoseDetection.getClient(options);  
 }  
  
 private void createTimer() {  
 final Handler handler = new Handler();  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 handler.postDelayed(this, 1000);  
 *//updating live notification every second  
 //sendOnChannel1();* 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);  
 calories = Math.round(MET \* User.getWeight() \* (seconds.floatValue() / 3600));  
 reps = repcounter.getReps();  
 *//updating text views* timerText.setText(time);  
 calText.setText("Calories:\n" + calories.toString());  
 repText.setText("Reps:\n" + reps.toString());  
  
 *//calculating average pace of reps* DecimalFormat df = new DecimalFormat("#.##");  
 paceText.setText("Pace:\n" + df.format((float)reps\*60f/(float)seconds) + "\nreps/min");  
 handleQuotes();  
 }  
  
 }  
 });  
 }  
  
 private void handleQuotes() {  
 if (isAudio) {  
 *//starting with random quote for text to speech* Random r = new Random();  
 int currquote = r.nextInt(quotes.length);  
 if (seconds % 60 == 0) {  
 *//every 60 seconds a quote is spoken to help motivate the user* if (currquote > quotes.length) {  
 *//moving to front of quote array* currquote = 0;  
 }  
 *//speaking quote, and moving to next quote* tts.speak(quotes[currquote], TextToSpeech.QUEUE\_FLUSH, null);  
 currquote++;  
 }  
 }  
 }  
  
 private void handleCamera() {  
 final ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.getInstance(this);  
 *//getting display size (dependent on device)* ImageAnalysis imageAnalysis =  
 new ImageAnalysis.Builder()  
 *//instantiating ImageAnalysis, with user's phone display dimensions  
// .setTargetResolution()* .setTargetResolution(displaySize)  
 .setBackpressureStrategy(ImageAnalysis.STRATEGY\_KEEP\_ONLY\_LATEST)  
 .build();  
 *//setting the configuration for the image analysis* imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(SquatsActivity.this), new ImageAnalysis.Analyzer() {  
 @Override  
 public void analyze(@NonNull ImageProxy imageProxy) {  
  
 int rotationDegrees = imageProxy.getImageInfo().getRotationDegrees();  
 @SuppressLint("UnsafeOptInUsageError") Image image = imageProxy.getImage();  
 if (image != null) {  
 *//receiving the input image from camera* InputImage inputimage = InputImage.fromMediaImage(image, rotationDegrees);  
 Task<Pose> result = poseDetector.process(inputimage).addOnSuccessListener(new OnSuccessListener<Pose>() {  
 @Override  
 public void onSuccess(@NonNull Pose pose) {  
 if (isTracking) {  
 *//when the pose detector successfully can attach to image  
 //Receiving and processing landmarks from Google's ML kit software* List<PoseLandmark> allPoseLandmarks = pose.getAllPoseLandmarks();  
 processLandmarks(allPoseLandmarks);  
 *//drawing on the landmarks onto the user's screen* graphic.drawGraphic(allPoseLandmarks);  
 } else {  
 graphic.clearGraphic();  
 }  
  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 *//when the pose detector cannot attach to image* System.out.println("Failed Pose Detection");  
 }  
 }).addOnCompleteListener(new OnCompleteListener<Pose>() {  
 @Override  
 public void onComplete(@NonNull Task<Pose> task) {  
 *//making sure to close the instance of the image to allow the next image to be processed* imageProxy.close();  
 }  
 });  
 }  
 }  
 });  
  
 *//attaching the image analysis object to the camera* cameraProviderFuture.addListener(() -> {  
 try {  
 *//configuring camera to preview.* ProcessCameraProvider provider = cameraProviderFuture.get();  
 preview = new Preview.Builder().build();  
 preview.setSurfaceProvider(tv.getSurfaceProvider());  
 cameraSelector = new CameraSelector.Builder()  
 .requireLensFacing(CameraSelector.LENS\_FACING\_FRONT)  
 .build();  
 try {  
 *//binding the camera, preview and analyser together* provider.unbindAll();  
 provider.bindToLifecycle((LifecycleOwner) this, cameraSelector, preview, imageAnalysis);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 } catch (ExecutionException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }, ContextCompat.getMainExecutor(this));  
  
 graphic = new Graphic(displaySize);  
  
 }  
  
 private void processLandmarks(List<PoseLandmark> allLandmarks) {  
 *//method to deal with analyzing the landmarks in a particular instance, provided by ML Kit* if (!allLandmarks.isEmpty() && allLandmarks.get(PoseLandmark.LEFT\_HIP).getInFrameLikelihood() > 0.5f &&  
 allLandmarks.get(PoseLandmark.LEFT\_KNEE).getInFrameLikelihood() > 0.5f) {  
 repcounter.addEntry(allLandmarks);  
 }  
 }  
  
 *// --------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;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 isTracking = Boolean.FALSE;  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == 0) {*//checking if all permissions are granted on UI dialog* boolean granted = true;  
 for (int result : grantResults) {  
 if (result == PackageManager.PERMISSION\_DENIED) {  
 granted = false;  
 }  
 }  
 if (granted) {  
 isTracking = true;  
 startTracking();  
 } else {  
 Toast.makeText(this, "Permissions Denied\nPlease allow permissions in settings", Toast.LENGTH\_SHORT).show();  
 this.finish();  
 }  
 }  
 }  
  
 private void finishTracking() {  
 *//handles the safe closing of the activity, and presenting any information to the user* isTracking = false;  
 if (seconds > 60 && reps>=5) {  
 if (isAudio){  
 *//audio text to speech to congratulate user* tts.speak(String.format(Locale.getDefault(), "Congratulations, you burnt %d calories and did %d reps. See you next time!", calories, reps),  
 TextToSpeech.QUEUE\_FLUSH, null);  
 }  
 *//saving activity results to database, as long as activity lasted for more than a minute* DBhelper helper = new DBhelper(SquatsActivity.this);  
 if (helper.saveActivity("squats", date.toString(), timeStarted, seconds.toString(), calories.toString(), null, null, reps.toString())) {  
 Toast.makeText(SquatsActivity.this, "Save successful", Toast.LENGTH\_SHORT).show();  
 } else {  
 Toast.makeText(SquatsActivity.this, "Save unsuccessful", Toast.LENGTH\_SHORT).show();  
 }  
 } else {  
 *//saves space and resources on database* Toast.makeText(SquatsActivity.this, "Activity Not Saved (Too Short)", Toast.LENGTH\_SHORT).show();  
 }  
 this.finish();  
 }  
  
 *//composition class (cannot draw graphic without the push up activity* private class Graphic {  
 *//hash map, associating landmark name to graphic View* private final Map<String, View> graphicViewsMap;  
 private final Size displaySize;  
 private float scalex;  
 private float scaley;  
 private int yoffset;  
 private int xoffset;  
 *//n pixels to offset in order to fit scaled up image on view* private Graphic(Size displaySize) {  
 *//HASH MAP to store all views corresponding to landmarks* graphicViewsMap = new HashMap<String, View>();  
 graphicViewsMap.put("nose", (View) findViewById(R.id.nose));  
 graphicViewsMap.put("left\_shoulder", (View) findViewById(R.id.left\_shoulder));  
 graphicViewsMap.put("right\_shoulder", (View) findViewById(R.id.right\_shoulder));  
 graphicViewsMap.put("left\_elbow", (View) findViewById(R.id.lelbow));  
 graphicViewsMap.put("right\_elbow", (View) findViewById(R.id.relbow));  
 graphicViewsMap.put("left\_hip", (View) findViewById(R.id.left\_hip));  
 graphicViewsMap.put("right\_hip", (View) findViewById(R.id.right\_hip));  
 graphicViewsMap.put("left\_knee", (View) findViewById(R.id.left\_knee));  
 graphicViewsMap.put("right\_knee", (View) findViewById(R.id.right\_knee));  
 this.displaySize = displaySize;  
 }  
  
*// private int gcd(int p, int q){  
// //euclid's algorithm to find smallest possible ratio  
// if (q == 0) return p;  
// else return gcd(q, p % q);  
// }  
// private int ratio(int a, int b) {  
// final int gcd = gcd(a,b);  
// return (b/gcd);  
// }  
//  
// private void setScaleFactor(Point displaySize, Size imageSize){  
// Toast.makeText(PushUpActivity.this, String.format("Ratio of y: %d",ratio(displaySize.x,displaySize.y)), Toast.LENGTH\_SHORT).show();  
// Toast.makeText(PushUpActivity.this, String.format("Display size: %d, %d",displaySize.x,displaySize.y), Toast.LENGTH\_SHORT).show();  
// scalex = 2.25f;  
// scaley = 2.25f;  
//// int[] arr = new int[2];  
//// tv.getLocationInWindow(arr);  
//// yoffset = arr[1];  
// }* private void updateLandmarkGraphic(String name, Float x, Float y) {  
 View view = graphicViewsMap.get(name);  
 if (view != null) {  
 view.setVisibility(View.VISIBLE);  
 *//inverting x coordinate, as camera is in mirroring position* view.setX(displaySize.getWidth() - x);  
 view.setY(y);  
 }  
  
 }  
  
 private void clearGraphic() {  
 for (View view : graphicViewsMap.values()) {  
 view.setVisibility(View.GONE);  
 }  
 }  
  
 private void drawGraphic(List<PoseLandmark> allLandmarks) {  
 if (allLandmarks.isEmpty()) {  
 *//if no landmarks are detected, remove points from graphic* clearGraphic();  
  
 }  
 *//drawing points on the preview, corresponding to where ML Kit has detected the positions  
 //of the body parts* for (PoseLandmark landmark : allLandmarks) {  
 switch (landmark.getLandmarkType()) {  
 case PoseLandmark.LEFT\_SHOULDER:  
 updateLandmarkGraphic("left\_shoulder", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.RIGHT\_SHOULDER:  
 updateLandmarkGraphic("right\_shoulder", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.NOSE:  
 updateLandmarkGraphic("nose", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.LEFT\_ELBOW:  
 updateLandmarkGraphic("left\_elbow", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.RIGHT\_ELBOW:  
 updateLandmarkGraphic("right\_elbow", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.LEFT\_HIP:  
 updateLandmarkGraphic("left\_hip", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.RIGHT\_HIP:  
 updateLandmarkGraphic("right\_hip", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.LEFT\_KNEE:  
 updateLandmarkGraphic("left\_knee", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 case PoseLandmark.RIGHT\_KNEE:  
 updateLandmarkGraphic("right\_knee", landmark.getPosition().x, landmark.getPosition().y);  
 break;  
 }  
  
 }  
 }  
  
 }  
}

#### LogInScreen.java

package com.example.exercisetracker.login;  
  
import android.content.Context;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.text.Html;  
import android.view.View;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.CompoundButton;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.activities.MainActivity;  
import com.example.exercisetracker.fragments.HistoryFragment;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.User;  
import com.google.android.material.textfield.TextInputLayout;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
*/\*\*  
 \* handling log in screen  
 \* Extends from AppCompatActivity class  
 \*/*public class LogInScreen extends AppCompatActivity {  
 *//shared preferences strings* private final static String *remember\_me* = "remember";  
 private final static String *shared\_prefs* = "sharedPrefs";  
 private TextInputLayout usernameField;  
 private TextInputLayout passwordField;  
 private CheckBox remember;  
  
 public static String getRemember\_me() {  
 return *remember\_me*;  
 }  
  
 public static String getShared\_prefs() {  
 return *shared\_prefs*;  
 }  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 *// if user has previously logged in, and selected "remember me"  
 // user automatically logged into the app* if (getUserSP()) {  
 Intent intent1 = new Intent(getApplicationContext(), MainActivity.class);  
 startActivity(intent1);  
 finish();  
 }  
 super.onCreate(savedInstanceState);  
 *//visuals* getSupportActionBar().hide();  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 getWindow().setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 setContentView(R.layout.*activity\_loginscreen*);  
  
 *//referencing all variables* ProgressBar progressBar = findViewById(R.id.*progressBar*);  
 usernameField = findViewById(R.id.*usernameField*);  
 passwordField = findViewById(R.id.*passwordField*);  
 Button createbtn = findViewById(R.id.*createaccount*);  
 Button loginbtn = findViewById(R.id.*loginbtn*);  
 remember = findViewById(R.id.*rememberBox*);  
 TextView repo = findViewById(R.id.*githubRepo*);  
 repo.setText(Html.*fromHtml*("<a href='https://github.com/calebchan1/ALevelNEA'> github.com/calebchan1/ALevelNEA </a>"));  
 loginbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String username = Objects.*requireNonNull*(usernameField.getEditText()).getText().toString();  
 String password = Objects.*requireNonNull*(passwordField.getEditText()).getText().toString();  
 *//user validation here* DBhelper helper = new DBhelper(LogInScreen.this);  
 if (helper.login(username, password)) {  
 String[] results = helper.getResult().get(0).split(" ");  
 saveToUserClass(results, username, password);  
 if (remember.isChecked()) {  
 saveToSharedPreferences(results, username, password);  
 }  
 *//user details were valid, navigating into app* Intent intent1 = new Intent(getApplicationContext(), MainActivity.class);  
 startActivity(intent1);  
 finish();  
 }  
 }  
 });  
 createbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//if user wants to create an account, navigate to RegisterUserActivity* Intent intent1 = new Intent(getApplicationContext(), RegisterUserActivity.class);  
 startActivity(intent1);  
 }  
 });  
  
 remember.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
 *//using Android's SharedPreferences to store whether or not the user has requested  
 //to stay logged in, even after closing the app* if (buttonView.isChecked()) {  
 SharedPreferences prefs = getSharedPreferences(*shared\_prefs*, *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putBoolean(*remember\_me*, true);  
 editor.apply();  
 } else if (!buttonView.isChecked()) {  
 SharedPreferences prefs = getSharedPreferences(*shared\_prefs*, *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putBoolean(*remember\_me*, false);  
 editor.apply();  
 }  
 }  
 });  
 }  
  
 private void saveToUserClass(String[] results, String username, String password) {  
 *// results received in format ID, forename, surname, DOB, weight, height  
 //saving to static User class* User.*setUsername*(username);  
 User.*setPassword*(password);  
 User.*setUserID*(Integer.*valueOf*(results[0]));  
 User.*setForename*(results[1]);  
 User.*setSurname*(results[2]);  
 java.sql.Date date = java.sql.Date.*valueOf*(results[3]);  
 User.*setDateOfBirth*(date);  
 User.*setWeight*(Float.*valueOf*(results[4]));  
 User.*setHeight*(Integer.*valueOf*(results[5]));  
 }  
  
 private void saveToSharedPreferences(String[] results, String username, String password) {  
 *//saving to SharedPreferences  
 // results received in format ID, forename, surname, DOB, weight, height* SharedPreferences prefs = getSharedPreferences(*shared\_prefs*, *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("username", username);  
 editor.putString("password", password);  
 editor.putString("id", results[0]);  
 editor.putString("forename", results[1]);  
 editor.putString("surname", results[2]);  
 editor.putString("DOB", results[3]);  
 editor.putString("weight", results[4]);  
 editor.putString("height", results[5]);  
 editor.apply();  
 }  
  
 public boolean getUserSP() {  
 *// getting saved user details from shared preferences  
 // handles saving user details to User class from shared preferences* SharedPreferences prefs = getSharedPreferences(*shared\_prefs*, *MODE\_PRIVATE*);  
 Boolean checkbox = prefs.getBoolean(*remember\_me*, false);  
  
 if (checkbox.equals(true)) {  
 try {  
 prefs = getSharedPreferences(*shared\_prefs*, *MODE\_PRIVATE*);  
 String[] results = {  
 prefs.getString("id", ""),  
 prefs.getString("forename", ""),  
 prefs.getString("surname", ""),  
 prefs.getString("DOB", ""),  
 prefs.getString("weight", ""),  
 prefs.getString("height", "")  
 };  
 saveToUserClass(results, prefs.getString("username", ""), prefs.getString("password", ""));  
 return true;  
 } catch (Exception e) {  
 return false;  
 }  
  
 }  
 return false;  
 }  
  
*// //Async task handling logging in  
// //using async task to retrieve data from database  
// private class LogInTask extends AsyncTask<String, Integer, Boolean> {  
// protected Boolean doInBackground(String... args) {  
// String username = args[0];  
// String password = args[1];  
// //CHECKING IF NOTHING HAS BEEN ENTERED  
// if (username.isEmpty() | password.isEmpty()){  
// return false;  
// }  
// runOnUiThread(new Runnable() {  
// @Override  
// public void run() {  
// progressBar.setVisibility(View.VISIBLE);  
// }  
// });  
//  
// DBhelper helper = new DBhelper(LogInScreen.this);  
// if (helper.login(username, password)) {  
// String[] results = helper.getResult().get(0).split(" ");  
// saveToUserClass(results, username, password);  
// if (remember.isChecked()) {  
// saveToSharedPreferences(results, username, password);  
// }  
// return true;  
// }  
// return false;  
// }  
//  
//  
// protected void onPostExecute(Boolean queryResults) {  
// runOnUiThread(new Runnable() {  
// @Override  
// public void run() {  
// progressBar.setVisibility(View.GONE);  
// }  
// });  
// if (queryResults) {  
// Intent intent1 = new Intent(getApplicationContext(), MainActivity.class);  
// startActivity(intent1);  
// finish();  
// }  
// else{  
// Toast.makeText(LogInScreen.this, "Login Unsuccessful", Toast.LENGTH\_SHORT).show();  
// }  
// }  
//  
// }*}

#### RegisterUserActivity.java

package com.example.exercisetracker.login;  
  
import android.annotation.SuppressLint;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.text.InputType;  
import android.view.MotionEvent;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
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.TextInputLayout;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Objects;  
  
public class RegisterUserActivity extends AppCompatActivity {  
  
 private TextInputLayout usernameField;  
 private TextInputLayout passwordField;  
 private TextInputLayout forenameField;  
 private TextInputLayout surnameField;  
 private TextInputLayout DOBField;  
 private TextInputLayout weightField;  
 private TextInputLayout heightField;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* Objects.*requireNonNull*(getSupportActionBar()).hide();  
 getWindow().setNavigationBarColor(getResources().getColor(R.color.*main\_colour*));  
 getWindow().setStatusBarColor(getResources().getColor(R.color.*main\_colour*));  
 setContentView(R.layout.*activity\_registeruser*);  
  
 *//all input fields instantiated* usernameField = findViewById(R.id.*usernameField*);  
 passwordField = findViewById(R.id.*passwordField*);  
 forenameField = findViewById(R.id.*firstnameField*);  
 surnameField = findViewById(R.id.*lastnameField*);  
 DOBField = findViewById(R.id.*DOBfield*);  
 weightField = findViewById(R.id.*weightField*);  
 heightField = findViewById(R.id.*heightField*);  
  
 *//buttons* Button cancelbtn = findViewById(R.id.*newUser\_cancelBtn*);  
 cancelbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//returns back to log in screen* RegisterUserActivity.this.finish();  
 }  
 });  
  
 *//material date picker* DateFormat df = new SimpleDateFormat("yyyy-MM-dd");  
 EditText dobText = DOBField.getEditText();  
 Objects.*requireNonNull*(dobText).setInputType(InputType.*TYPE\_NULL*);  
  
 dobText.setKeyListener(null);  
 *//date of birth picker constraints, must be at least 10 yrs old to register account* CalendarConstraints.Builder constraints = new CalendarConstraints.Builder().setValidator(DateValidatorPointBackward.*before*(MaterialDatePicker.*todayInUtcMilliseconds*() - 315569260000L));  
 MaterialDatePicker.Builder<Long> datepickerBuilder = MaterialDatePicker.Builder.*datePicker*();  
 *//by default starts picker on min age* datepickerBuilder.setCalendarConstraints(constraints.build()).setSelection(MaterialDatePicker.*todayInUtcMilliseconds*() - 315569260000L);  
 MaterialDatePicker datepicker = datepickerBuilder.build();  
  
 *//when date of birth is touched* dobText.setOnTouchListener(new View.OnTouchListener() {  
 @SuppressLint("ClickableViewAccessibility")  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 if (event.getAction() == MotionEvent.*ACTION\_UP*) {  
 datepicker.show(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()));  
 datepicker.dismiss();  
 }  
 });  
 datepicker.addOnCancelListener(new DialogInterface.OnCancelListener() {  
 @Override  
 public void onCancel(DialogInterface dialog) {  
 datepicker.dismiss();  
 }  
 });  
  
 *//create button* Button createbtn = findViewById(R.id.*createBtn*);  
 createbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//all fields are required in order for user to create an account  
 //passing details to database* String username = Objects.*requireNonNull*(usernameField.getEditText()).getText().toString();  
 String password = Objects.*requireNonNull*(passwordField.getEditText()).getText().toString();  
 String forename = Objects.*requireNonNull*(forenameField.getEditText()).getText().toString();  
 String surname = Objects.*requireNonNull*(surnameField.getEditText()).getText().toString();  
 String DOB = Objects.*requireNonNull*(DOBField.getEditText()).getText().toString();  
 String weight = Objects.*requireNonNull*(weightField.getEditText()).getText().toString();  
 String height = Objects.*requireNonNull*(heightField.getEditText()).getText().toString();  
 *//username and password must be bigger than 8 characters conditions  
 //username must not exceed 16 characters* boolean requirements = username.length() >= 8 && password.length() >= 8 && username.length() <= 16;  
 boolean containsSpace = username.contains(" ") || password.contains(" ") || forename.contains(" ") || surname.contains(" ") || weight.contains(" ") || height.contains(" ");  
 *//if one field is empty, cannot create account* boolean isEmpty = username.isEmpty() || password.isEmpty() || forename.isEmpty() || surname.isEmpty() || DOB.isEmpty() || weight.isEmpty() || height.isEmpty();  
  
 if (!isEmpty) {  
 if (requirements) {  
 if (!containsSpace) {  
 DBhelper helper = new DBhelper(RegisterUserActivity.this);  
 if (helper.registerUser(username, password, forename, surname, DOB, weight, height)) {  
 *//if user was successfully added to database, enter the app* finish();  
 } else {  
 *//user was not successfully added to database, error shown to user* Toast.*makeText*(RegisterUserActivity.this, "Could not create account", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//User entered a space in the username or password* Toast.*makeText*(RegisterUserActivity.this, "All fields must not contain any spaces", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 *//User did not meet required number of chars for username and password* Toast.*makeText*(RegisterUserActivity.this, "Username must be between 8-16 characters", Toast.*LENGTH\_SHORT*).show();  
 Toast.*makeText*(RegisterUserActivity.this, "Password have at least 8 characters", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(RegisterUserActivity.this, "You have not entered all the required fields", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
}

#### AddFriendsActivity.java

package com.example.exercisetracker.activities;  
  
import android.app.ProgressDialog;  
import android.content.Context;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.exercisetracker.R;  
import com.example.exercisetracker.other.DBhelper;  
import com.example.exercisetracker.other.Friend;  
import com.example.exercisetracker.other.User;  
import com.google.android.material.textfield.TextInputLayout;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
public class AddFriendsActivity extends AppCompatActivity implements View.OnClickListener {  
 private EditText searchEditText;  
  
 *//recycler view* private AddFriendsActivity.FriendAdapter courseAdapter;  
 private ArrayList<Friend> friendArr;  
 private TextView noFriends;  
  
 *//loading dialog* private ProgressDialog loadingDialog;  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 *//handling when back button is pressed* if (item.getItemId() == android.R.id.home) {  
 this.finish();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *//visuals* setContentView(R.layout.activity\_addfriends);  
 *// showing the back button in action bar* Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setTitle("");  
 getWindow().setStatusBarColor(getResources().getColor(R.color.main\_colour));  
  
 *//setting on click listener to class* findViewById(R.id.searchBtn).setOnClickListener(this);  
 TextInputLayout input = findViewById(R.id.searchView);  
 searchEditText = input.getEditText();  
 noFriends = findViewById(R.id.noFriendsTV);  
  
 *//recycler views* RecyclerView recyclerView = findViewById(R.id.friendsRV);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getApplicationContext(), LinearLayoutManager.VERTICAL, false);  
 friendArr = new ArrayList<>();  
 *// we are initializing our adapter class and passing our arraylist to it.* courseAdapter = new AddFriendsActivity.FriendAdapter(getApplicationContext(), friendArr);  
 recyclerView.setAdapter(courseAdapter);  
 recyclerView.setLayoutManager(linearLayoutManager);  
  
 loadingDialog = new ProgressDialog(this);  
 loadingDialog.setMessage("Loading..");  
 loadingDialog.setTitle("Retrieving Your Friends List");  
 loadingDialog.setIndeterminate(true);  
 loadingDialog.setCancelable(false);  
 loadingDialog.show();  
 *//by default, the user's friends list shows on the screen  
 //they can decide to remove any of their friends* new LoadFriendsListTask().execute(true);  
 }  
  
 @Override  
 public void onClick(View v) {  
 if (v.getId() == R.id.searchBtn) {*//refreshing the static user class friend list from database* DBhelper helper = new DBhelper(this);  
 *//when user wants to search for other users* if (!searchEditText.getText().toString().equals("")) {  
 helper.clearResults();  
 if (helper.getUsers(searchEditText.getText().toString())) {  
 if (!helper.getResult().isEmpty()) {  
 noFriends.setVisibility(View.INVISIBLE);  
 *//list of users passed to recycler view  
 //resetting friendArr for new query* friendArr.clear();  
 courseAdapter.notifyDataSetChanged();  
 for (String row : helper.getResult()) {  
 Friend friendObj = handleQuery(row);  
 friendArr.add(friendObj);  
 courseAdapter.notifyItemInserted(courseAdapter.getItemCount());  
 *// setting layout manager and adapter to our recycler view.* }  
 } else {  
 *//no users found from result* Toast.makeText(getApplicationContext(), "No Users Found", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 } else {  
 *//user has searched empty query* Toast.makeText(getApplicationContext(), "You have not entered anything", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
  
 private Friend handleQuery(String query) {  
 String[] arr = query.split(" ");  
 int id = Integer.parseInt(arr[0]);  
 String firstname = arr[1];  
 String surname = arr[2];  
 String username = arr[3];  
 return new Friend(id, firstname, surname, username);  
 }  
  
 *//adapter class to exchange information between card views created and friend details* public static class FriendAdapter extends RecyclerView.Adapter<AddFriendsActivity.FriendAdapter.Viewholder> {  
 private final Context context;  
 private final ArrayList<Friend> friendsArr;  
  
 public FriendAdapter(Context context, ArrayList<Friend> friendsArr) {  
 this.context = context;  
 this.friendsArr = friendsArr;  
  
 }  
  
 @NonNull  
 @Override  
 public AddFriendsActivity.FriendAdapter.Viewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 *//inflate the layout for each item of recycler view.* View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.friends\_card\_layout, parent, false);  
 return new AddFriendsActivity.FriendAdapter.Viewholder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull AddFriendsActivity.FriendAdapter.Viewholder holder, int position) {  
 *//setting the instance holder data from the friend object at that position in the array* Friend friend = friendsArr.get(position);  
  
 if (User.getFriendsList().contains(friend.getId())) {  
 *//if friend is in the user's friend list change button to remove friend* holder.addFriendBtn.setText("Remove Friend");  
 } else {  
 holder.addFriendBtn.setText("Add Friend");  
 }  
  
  
 holder.realNameTV.setText(friend.getFirstname() + " " + friend.getSurname());  
 holder.usernameIDTV.setText("username: " + friend.getUsername());  
 holder.addFriendBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (holder.addFriendBtn.getText().equals("Add Friend")) {  
 *//handling when user clicks on add friend on a specific user* DBhelper helper = new DBhelper(context.getApplicationContext());  
 if (helper.addFriend(User.getUserID(), friend.getId())) {  
 Toast.makeText(context.getApplicationContext(), "Friend Added", Toast.LENGTH\_SHORT).show();  
 holder.addFriendBtn.setText("Remove Friend");  
 } else {  
 Toast.makeText(context.getApplicationContext(), "Could not add friend", Toast.LENGTH\_SHORT).show();  
 }  
 } else if (holder.addFriendBtn.getText() == "Remove Friend") {  
 *//when friend has been already added and wants to remove friendship* DBhelper helper = new DBhelper(context);  
 if (helper.removeFriend(User.getUserID(), friend.getId())) {  
 Toast.makeText(context.getApplicationContext(), "Friend Removed", Toast.LENGTH\_SHORT).show();  
 holder.addFriendBtn.setText("Add Friend");  
 } else {  
 Toast.makeText(context.getApplicationContext(), "Could not remove friend", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return friendsArr.size();  
 }  
  
 *// View holder class for initializing of views such as TextView* public static class Viewholder extends RecyclerView.ViewHolder {  
 private final TextView realNameTV;  
 private final TextView usernameIDTV;  
 private final Button addFriendBtn;  
  
 public Viewholder(@NonNull View itemView) {  
 super(itemView);  
 *//binding views to instance of Viewholder* realNameTV = itemView.findViewById(R.id.userRealName);  
 usernameIDTV = itemView.findViewById(R.id.usernameID);  
 addFriendBtn = itemView.findViewById(R.id.addFriendBtn);  
 }  
 }  
 }  
  
 private class LoadFriendsListTask extends AsyncTask<Boolean, Integer, ArrayList<String>> {  
 *//task to load friends list* protected ArrayList<String> doInBackground(Boolean... isPublic) {  
 *//retrieve data from database* DBhelper helper = new DBhelper(getApplicationContext());  
 if (helper.getFriends()) {  
 return helper.getResult();  
 }  
 return null;  
 }  
  
 protected void onPostExecute(ArrayList<String> result) {  
 loadingDialog.dismiss();  
 if (result == null) {  
 *//no friends found due to connection issue* noFriends.setVisibility(View.VISIBLE);  
 } else if (!result.isEmpty()) {  
 noFriends.setVisibility(View.INVISIBLE);  
 User.clearFriendsList();  
 friendArr.clear();  
 courseAdapter.notifyDataSetChanged();  
 for (String query : result) {  
 Friend friendObj = handleQuery(query);  
 *//adding to user's list of friends* User.addFriendsList(friendObj.getId());  
 friendArr.add(friendObj);  
 courseAdapter.notifyItemInserted(courseAdapter.getItemCount());  
 *//adding to recycler view (by default when user loads this section  
 //their friends will appear* }  
 } else {  
 *//no friends found, disclaimer shown to user* noFriends.setVisibility(View.VISIBLE);  
 }  
 }  
 }  
}

### Custom Java Classes

#### User.java Class

package com.example.exercisetracker.other;  
  
import android.content.Context;  
import android.content.SharedPreferences;  
  
import com.example.exercisetracker.login.LogInScreen;  
  
import java.sql.Date;  
import java.util.HashSet;  
import java.util.Set;  
  
public class User {  
 private static Integer UserID;  
 private static String forename;  
 private static String surname;  
 private static Float weight;  
 private static java.sql.Date dateOfBirth;  
 private static Integer height;  
 private static String username;  
 private static String password;  
 private static Set<Integer> friendsList = new HashSet<Integer>();  
  
 public static Set<Integer> getFriendsList() {  
 return friendsList;  
 }  
  
 public static void addFriendsList(Integer friend) {  
 *//adds to set of friends  
 //as set, no duplicate ids will exist* User.friendsList.add(friend);  
 }  
  
 public static void removeFriendsList(Integer friend) {  
 *//adds to set of friends  
 //as set, no duplicate ids will exist* User.friendsList.remove(friend);  
 }  
  
 public static void saveUser(String username, String password,String forename, String surname,String DOB, Float weight, Integer height ){  
 *//saving sanitised details to user class* setUsername(username);  
 setPassword(password);  
 setWeight(weight);  
 setHeight(height);  
 setForename(forename);  
 setSurname(surname);  
 setDateOfBirth(Date.valueOf(DOB));  
 }  
  
 public static void clearFriendsList() {  
 friendsList.clear();  
 }  
  
  
 public static String getUsername() {  
 return username;  
 }  
  
 public static void setUsername(String username) {  
 User.username = username;  
 }  
  
 public static String getForename() {  
 return forename;  
 }  
  
 public static void setForename(String forename) {  
 User.forename = forename;  
 }  
  
 public static String getSurname() {  
 return surname;  
 }  
  
 public static void setSurname(String surname) {  
 User.surname = surname;  
 }  
  
 public static String getPassword() {  
 return password;  
 }  
  
 public static void setPassword(String password) {  
 User.password = password;  
 }   
  
 public static Integer getUserID() {  
 return UserID;  
 }  
  
 public static void setUserID(Integer userID) {  
 UserID = userID;  
 }  
  
 public static String getName() {  
 return getForename() + " " + getSurname();  
 }  
  
 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 java.sql.Date getDateOfBirth() {  
 return dateOfBirth;  
 }  
  
 public static void setDateOfBirth(java.sql.Date dateOfBirth) {  
 User.dateOfBirth = dateOfBirth;  
 }  
  
 public static void logout(Context context) {  
 *//clearing User class data* setUsername(null);  
 setUserID(null);  
 setHeight(null);  
 setWeight(null);  
 setDateOfBirth(null);  
 setPassword(null);  
 setForename(null);  
 setSurname(null);  
 *//clearing shared preferences* SharedPreferences prefs = context.getSharedPreferences(LogInScreen.getShared\_prefs(), Context.MODE\_PRIVATE);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.clear();  
 editor.apply();  
 }  
  
}

#### Detector.java Class

package com.example.exercisetracker.stepcounting;  
  
*/\*\*  
 \* Detector class used to detect number of steps by considering how long  
 \* the user acceleration exceeds a certain amount  
 \*/*public class Detector {  
  
 private float Threshold;  
 private Integer stepduration;  
  
 public Detector(float Threshold, Integer stepduration) {  
 this.Threshold = Threshold;  
 this.stepduration = stepduration;  
 }  
  
 public Integer detect(Float[] filtered\_data) {  
 int i = 0;  
 int stepCount = 0;  
 while (i < filtered\_data.length - 1) {  
 Float data = filtered\_data[i];  
 if (data >= Threshold) {  
 int duration = 0;  
 while ((filtered\_data[i] > Threshold) && (i != filtered\_data.length - 1)) {  
 i++;  
 duration = duration + 1;  
 }  
 if (duration > stepduration) {  
 stepCount++;  
 }  
 } else {  
 i++;  
 }  
 }  
 return stepCount;  
 }  
  
 public float getThreshold() {  
 return Threshold;  
 }  
  
 public void setThreshold(float threshold) {  
 Threshold = threshold;  
 }  
}

#### Filter.java Class

package com.example.exercisetracker.stepcounting;  
  
import java.util.ArrayList;  
  
public class Filter {  
 private Float[] filtered\_data;  
 private Float minThreshold;  
 private Float maxThreshold;  
  
 public Filter(float minThreshold, float maxThreshold) {  
 *//setting max and min threshold on create* this.minThreshold = minThreshold;  
 this.maxThreshold = maxThreshold;  
 }  
  
 public void filter(ArrayList<Float> data) {  
 *//filter the data to remove bumpiness and outliers using thresholds given* filtered\_data = new Float[data.size()];  
 for (int i = 0; i < data.size(); i++) {  
 if (data.get(i) == 0.0) {  
 filtered\_data[i] = 0f;  
 } else if (data.get(i) > maxThreshold) {  
 filtered\_data[i] = maxThreshold;  
 } else if (data.get(i) < minThreshold) {  
 filtered\_data[i] = minThreshold;  
 } else {  
 filtered\_data[i] = data.get(i);  
 }  
 }  
 setFiltered\_data(filtered\_data);  
 }  
  
 public float getMaxThreshold() {  
 return maxThreshold;  
 }  
  
 public void setMaxThreshold(float maxThreshold) {  
 this.maxThreshold = maxThreshold;  
 }  
  
 public Float[] getFiltered\_data() {  
 return filtered\_data;  
 }  
  
 public void setFiltered\_data(Float[] filtered\_data) {  
 this.filtered\_data = filtered\_data;  
 }  
  
 public float getMinThreshold() {  
 return minThreshold;  
 }  
  
 public void setMinThreshold(float minThreshold) {  
 this.minThreshold = minThreshold;  
 }  
}

#### StepCounter.java Class

package com.example.exercisetracker.stepcounting;  
  
import android.content.Context;  
  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
  
*/\*\*  
 \* StepCounter class encapsulates all methods and classes involving step detection.  
 \* Is used in activities requiring step detection from accelerometer and gravimeter  
 \* Currently, these activities include: RunningActivity, TreadmillActivity and WalkingActivity  
 \* <p>  
 \* Is composed of Detector and Filter classes  
 \*/*public class StepCounter {  
 private final DecimalFormat df;  
 private final Context context;  
 private final Detector detector;  
 private final Filter filter;  
 private Integer steps;  
 *//2d arrays to store a variable amount of samples, each sample consisting of the x y z values* private final ArrayList<Float[]> grav;  
 private final ArrayList<Float[]> accel;  
 private Boolean hasProcessed;  
  
  
 public StepCounter(Context context, Integer stepDuration, Float DetectThresh, Float MinFilterThresh, Float MaxFilterThresh, DecimalFormat df) {  
 *//constructor* this.filter = new Filter(MinFilterThresh, MaxFilterThresh);  
 this.detector = new Detector(DetectThresh, stepDuration);  
 this.context = context;  
 this.df = df;  
 this.hasProcessed = Boolean.*FALSE*;  
 this.accel = new ArrayList<>();  
 this.grav = new ArrayList<>();  
 this.steps = 0;  
 }  
  
 public Boolean getHasProcessed() {  
 return hasProcessed;  
 }  
  
 public void setHasProcessed(Boolean hasProcessed) {  
 this.hasProcessed = hasProcessed;  
 }  
  
 public Integer getSteps() {  
 return steps;  
 }  
  
 public void addEntry(Integer SensorType, Float x, Float y, Float z) {  
 switch (SensorType) {  
 case 0:  
 *// when sensorType is 0, from accelerometer* Float[] entry = convertToEntry(x, y, z);  
 System.*out*.println("acceleration:" + String.*format*("%f, %f, %f", entry[0], entry[1], entry[2]));  
 accel.add(entry);  
 this.accel.add(entry);  
 case 1:  
 *// when sensorType is 1, from gravimeter* Float[] entry1 = convertToEntry(x, y, z);  
 System.*out*.println("gravity:" + String.*format*("%f, %f, %f", entry1[0], entry1[1], entry1[2]));  
 grav.add(entry1);  
 this.grav.add(entry1);  
 }  
 }  
  
 public boolean isEmpty() {  
 *//if grav and accel arraylists are empty* if (grav.isEmpty() && accel.isEmpty()) {  
 return true;  
 }  
 return false;  
 }  
  
 public void countSteps() {  
 ArrayList<Float> results = processData();  
 grav.clear();  
 accel.clear();  
 *//hasProcessed to true to prevent small chunks of data being processed* hasProcessed = Boolean.*TRUE*;  
  
 *//FILTERING DATA* filter.filter(results);  
  
 *//detecting steps* this.steps += detector.detect(filter.getFiltered\_data());  
 }  
  
 public double calculatePace(Float x, Float y, Float z) {  
 *//since change in second for acceleration in one second  
 //working out magnitude of velocity* double pace = Math.*sqrt*(Math.*pow*(x, 2) + Math.*pow*(y, 2) + Math.*pow*(z, 2));  
 return pace;  
 }  
  
 private Float[] convertToEntry(Float raw\_x, Float raw\_y, Float raw\_z) {  
 *//converting entries into single float array* float x = Float.*parseFloat*(df.format(raw\_x));  
 float y = Float.*parseFloat*(df.format(raw\_y));  
 float z = Float.*parseFloat*(df.format(raw\_z));  
 Float[] entry = new Float[3];  
 entry[0] = x;  
 entry[1] = y;  
 entry[2] = z;  
 return entry;  
 }  
  
 private ArrayList<Float> processData() {  
 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);  
 }  
 }  
  
 *//PERFORM DOT PRODUCT* 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]));  
 result = result / 9.81f;  
 results.add(result);  
 System.*out*.println("result: " + j + " " + result.toString());  
 }  
 return results;  
 }  
}

#### RepCounter.java Class

package com.example.exercisetracker.repDetection;  
  
import android.content.Context;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.core.content.ContextCompat;  
  
import com.example.exercisetracker.R;  
import com.google.mlkit.vision.common.PointF3D;  
import com.google.mlkit.vision.pose.PoseLandmark;  
  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
import java.util.Objects;  
  
*/\*\*  
 \* RepCounter class used to detect number of reps from results of mlk tracking  
 \* Landmarks added as entries to class  
 \*/*public class RepCounter {  
 private final Float minDistance;  
 private boolean enteredPose;  
 private final TextView indicator;  
 private final TextView debugTV;  
 private int reps;  
 private final Context context;  
 *//calculations* private int duration; *//duration of n samples lapsed since entered pose* private boolean countedRep;  
 private boolean pushedDown;  
 private boolean returnedToPosition;  
 private Map<Integer, PointF3D> startPoint;  
 private final int type; *//type refers to type of exercise  
 //0 for push up and 1 for squats* public RepCounter(Context context, Integer type, TextView indicator, TextView debug, Float minDistance) {  
 this.context = context;  
 this.minDistance = minDistance;  
 this.debugTV = debug;  
 this.indicator = indicator;  
 this.type = type;  
 reps = 0;  
 enteredPose = false;  
 duration = 0;  
 pushedDown = false;  
 returnedToPosition = false;  
 countedRep = false;  
  
 }  
  
 public int getReps() {  
 return reps;  
 }  
  
 private void reset() {  
 countedRep = false;  
 pushedDown = false;  
 returnedToPosition = false;  
 }  
  
 public void addEntry(List<PoseLandmark> landmarks) {  
 if (type == 0) {  
 *//rep counting for push up* if (!landmarks.isEmpty()) {  
 Map<Integer, PointF3D> relevantLandmarks = getRelevantLandmarks(landmarks);  
 *//Z value decreases as you move close to the phone  
 //Using Z value from ML KIT, determining if body is laying down horizontally  
 //i.e. Lower body has a LARGER z value than upper body* enteredPose =  
 Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_HIP)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getZ()  
 && Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_KNEE)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_HIP)).getZ()  
 && Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.RIGHT\_HIP)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getZ()  
 && Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.RIGHT\_KNEE)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.RIGHT\_HIP)).getZ();  
 if (enteredPose) {  
 detectPushUpReps(relevantLandmarks);  
 *//debugTV.setText(relevantLandmarks.get(PoseLandmark.NOSE).getY()+"\n Start pos:"+startPoint.get(PoseLandmark.NOSE).getY() + "\nDuration: "+duration);* } else {  
 reset();  
 duration = 0;  
 *//debugTV.setText("0\n0\n0");* }  
 updateIndicator();  
 }  
 } else if (type == 1) {  
 *//rep counting for squats  
 //rep counting for push up* if (!landmarks.isEmpty()) {  
 Map<Integer, PointF3D> relevantLandmarks = getRelevantLandmarks(landmarks);  
 *//Z value decreases as you move close to the phone  
 //Using Z value from ML KIT, determining if body in a squat position  
 //i.e. hips LARGER z value than knees and upper body by a minimum z distance away* enteredPose =  
 Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_HIP)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_KNEE)).getZ()  
 && Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.RIGHT\_HIP)).getZ() > Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.RIGHT\_KNEE)).getZ();  
 if (enteredPose) {  
 detectSquatReps(relevantLandmarks);  
 debugTV.setText(relevantLandmarks.get(PoseLandmark.NOSE).getY() + "\n Start pos:" + startPoint.get(PoseLandmark.NOSE).getY() + "\nDuration: " + duration);  
 } else {  
 reset();  
 duration = 0;  
 debugTV.setText("0\n0\n0");  
 }  
 updateIndicator();  
 }  
 }  
 }  
  
 private void detectPushUpReps(Map<Integer, PointF3D> relevantLandmarks) {  
 *//detecting push up reps* if (duration == 0) {  
 *//first time entering pose* startPoint = relevantLandmarks;  
 }  
 *//calculation of reps is done in a process:  
 //when user first initially enters pose, the position is recorded  
 //user has to then push down by at least x amount and return to original position for a rep to be counted* if (countedRep) {  
 *//instance of the rep was already counted previously, thus all boolean variables must reset* reset();  
 } else {  
 if (!pushedDown) {  
 *//checking to see if user has pushed down by tracking movement of nose* pushedDown = Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getY() >= Objects.requireNonNull(startPoint.get(PoseLandmark.NOSE)).getY() + minDistance;  
 } else {  
 *//user has pushed down, thus checking to see if they have returned to position by tracking movement of nose* if (!returnedToPosition) {  
 returnedToPosition = Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getY() < Objects.requireNonNull(startPoint.get(PoseLandmark.NOSE)).getY();  
 }  
 }  
 if (pushedDown && returnedToPosition && !countedRep) {  
 *//if that instance of rep was not counted yet* Toast.makeText(this.context, "Rep Counted", Toast.LENGTH\_SHORT).show();  
 reps++;  
 countedRep = true;  
 } else {  
 duration++;  
 }  
 }  
 }  
  
 private void detectSquatReps(Map<Integer, PointF3D> relevantLandmarks) {  
 *//detecting squat reps  
 //calculation of reps is done in a process:  
 //when user first initially enters pose, the position is recorded  
 //user has to then move in the z direction by a min amount and return to original position for a rep to be counted* if (duration == 0) {  
 *//first time entering pose* startPoint = relevantLandmarks;  
 }  
 if (countedRep) {  
 reset();  
 } else {  
 if (!pushedDown) {  
 *//checking to see if user has squatted down by tracking movement of difference of z distance  
 //between the knees and the hips* pushedDown = Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getY() > Objects.requireNonNull(startPoint.get(PoseLandmark.NOSE)).getY() + minDistance;  
 } else {  
 *//user has pushed down, thus checking to see if they have returned to position by tracking movement of nose/shoulder* if (!returnedToPosition) {  
 returnedToPosition = Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.NOSE)).getY() < Objects.requireNonNull(startPoint.get(PoseLandmark.NOSE)).getY() + minDistance/2f  
 || Objects.requireNonNull(relevantLandmarks.get(PoseLandmark.LEFT\_SHOULDER)).getY() < Objects.requireNonNull(startPoint.get(PoseLandmark.LEFT\_SHOULDER)).getY() + minDistance/2f  
 ;  
 }  
 }  
 if (pushedDown & returnedToPosition & !countedRep) {  
 *//if that instance of rep was not counted yet* Toast.makeText(this.context, "Rep Counted", Toast.LENGTH\_SHORT).show();  
 reps++;  
 countedRep = true;  
 }  
 else{  
 duration ++;  
 }  
  
 }  
 }  
  
  
 private void updateIndicator() {  
 *//method to update text view indicator on screen to show if user has entered a push up or not* if (type == 0) {  
 if (enteredPose && pushedDown && returnedToPosition) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Push Up Entered, Pushed Down, Returned");  
 } else if (enteredPose && pushedDown) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Push Up Entered, Pushed Down");  
 } else if (enteredPose) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Push Up Entered");  
 } else {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.red));  
 indicator.setText("Push Up Not Entered");  
 }  
 } else if (type == 1) {  
 if (enteredPose && pushedDown && returnedToPosition) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Squats Entered, Squatted Down, Returned");  
 } else if (enteredPose && pushedDown) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Squats Entered, Squatted Down");  
 } else if (enteredPose) {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.green));  
 indicator.setText("Squats Entered");  
 } else {  
 indicator.setTextColor(ContextCompat.getColor(context, R.color.red));  
 indicator.setText("Squats Not Entered");  
 }  
 }  
 }  
  
  
 private Map<Integer, PointF3D> getRelevantLandmarks(List<PoseLandmark> landmarks) {  
 *//filtering through all landmarks given by ML kit to only relevant ones for push up and squats  
 //these are the nose, hip and knee landmarks representing the face, lower and upper body* Map<Integer, PointF3D> relevantLandmarks = new HashMap<>();  
 for (PoseLandmark landmark : landmarks) {  
 switch (landmark.getLandmarkType()) {  
 case PoseLandmark.NOSE:  
 relevantLandmarks.put(PoseLandmark.NOSE, landmark.getPosition3D());  
 break;  
 case PoseLandmark.LEFT\_SHOULDER:  
 relevantLandmarks.put(PoseLandmark.LEFT\_SHOULDER,landmark.getPosition3D());  
 break;  
 case PoseLandmark.RIGHT\_SHOULDER:  
 relevantLandmarks.put(PoseLandmark.RIGHT\_SHOULDER,landmark.getPosition3D());  
 break;  
 case PoseLandmark.LEFT\_HIP:  
 relevantLandmarks.put(PoseLandmark.LEFT\_HIP, landmark.getPosition3D());  
 break;  
 case PoseLandmark.RIGHT\_HIP:  
 relevantLandmarks.put(PoseLandmark.RIGHT\_HIP, landmark.getPosition3D());  
 break;  
 case PoseLandmark.LEFT\_KNEE:  
 relevantLandmarks.put(PoseLandmark.LEFT\_KNEE, landmark.getPosition3D());  
 break;  
 case PoseLandmark.RIGHT\_KNEE:  
 relevantLandmarks.put(PoseLandmark.RIGHT\_KNEE, landmark.getPosition3D());  
 break;  
 }  
 }  
 return relevantLandmarks;  
 }  
}

#### Route.java Class

package com.example.exercisetracker.other;  
  
import java.util.ArrayList;  
  
public class Route {  
 private int ActivityID;  
 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 void addRoute(Double[] entry) {  
 this.route.add(entry);  
 }  
  
 public int getRouteSize() {  
 return route.size();  
 }  
  
 public Integer getLastRouteIndex() {  
 return lastRouteIndex;  
 }  
  
 public ArrayList<Double[]> getRoute() {  
 return route;  
 }  
  
}

#### Activity.java Class

Custom User defined class named “Activity”, which is not to be confused with Android’s class Activity. This class is purely for the ease of holding information received from a database as an object to allow easy manipulation and access

package com.example.exercisetracker.activities;  
  
import java.sql.Date;  
  
  
*/\*\*  
 \* Class used to store data about Activity  
 \*/*public class Activity {  
  
  
 private int img;  
 private String name;  
  
 private int id;  
 private String desc;  
 private String timeStarted;  
 private Date date;  
 private int duration;  
 private int calories;  
 private int steps;  
 private int distance;  
 private int reps;  
  
 public Activity(String name, String desc, int img, int id) {  
 this.name = name;  
 this.desc = desc;  
 this.img = img;  
 this.id = id;  
  
 String[] temp = desc.split(" ");  
  
 *// temp formatted as (date, time, duration, calories, steps, distance,reps)* if (name.equals("Running") || name.equals("Treadmill") || name.equals("Walking")) {  
 *//null values for number of reps* this.date = Date.valueOf(temp[0]);  
 this.timeStarted = temp[1].substring(0,5);*//slicing string to just fit hr:min* this.duration = Integer.parseInt(temp[2]);  
 this.calories = Integer.parseInt(temp[3]);  
 this.steps = Integer.parseInt(temp[4]);  
 *//removing decimal place from distance to convert to integer* this.distance = Integer.parseInt(temp[5].split("\\.")[0]);  
 } else if (name.equals("Push Up") || name.equals("Squats")) {  
 *//null values for steps and distance* this.date = Date.valueOf(temp[0]);  
 this.timeStarted = temp[1].substring(0,5);*//slicing string to just fit hr:min* this.duration = Integer.parseInt(temp[2]);  
 this.calories = Integer.parseInt(temp[3]);  
 this.reps = Integer.parseInt(temp[6]);  
 }  
  
 }  
  
 public String getTimeStarted() {  
 return timeStarted;  
 }  
  
 public Date getDate() {  
 return date;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
  
 public int getDuration() {  
 return duration;  
 }  
  
  
 public int getCalories() {  
 return calories;  
 }  
  
 public void setCalories(int calories) {  
 this.calories = calories;  
 }  
  
 public int getSteps() {  
 return steps;  
 }  
  
 public void setSteps(int steps) {  
 this.steps = steps;  
 }  
  
 public int getDistance() {  
 return distance;  
 }  
  
 public void setDistance(int distance) {  
 this.distance = distance;  
 }  
  
 public int getReps() {  
 return reps;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getImg() {  
 return img;  
 }  
}

#### Friend.java Class

Class allowing me to store information specific to another user which is not the current user using the app such as their username, surname, forename and calories.

package com.example.exercisetracker.other;  
  
public class Friend {  
 private int id;  
 private String firstname;  
 private String surname;  
 private String username;  
  
 public Friend(int id, String firstname, String surname, String username) {  
 this.id = id;  
 this.firstname = firstname;  
 this.surname = surname;  
 this.username = username;  
 }  
  
  
 public int getId() {  
 return id;  
 }  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
}

### MSSQL Database & DBhelper.java

The following code is for the DBhelper.java class, which specifically handles the SQL queries sent to the Azure database server via JDBC library. SQL language is in Microsoft’s SQL Server syntax.

package com.example.exercisetracker.other;  
  
import android.content.Context;  
import android.os.StrictMode;  
import android.widget.Toast;  
  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import java.security.NoSuchProviderException;  
import java.security.SecureRandom;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Locale;  
import java.util.Set;  
  
public class DBhelper {  
 *//freesqldatabase.com  
// private static final String url = "jdbc:mysql://sql4.freesqldatabase.com:3306/sql4456768";  
// private static final String dbuser = "sql4456768";  
// private static final String dbpassword = "gyFr8LHqQA";  
 //azure database* private static final String *dbuser* = "calebchanwy";  
 private static final String *dbpassword* = "gyFr8LHqQA";  
 private static final String *url* = String.*format*(Locale.*getDefault*(),  
 "jdbc:jtds:sqlserver://trackerplus2.database.windows.net:1433;databasename=trackerplus;user=%s@trackerplus2;password=%s;"  
 ,*dbuser*,*dbpassword*);  
 private Context context;  
 private int flag;  
 private ArrayList<String> result = new ArrayList<String>();  
  
 public DBhelper(Context context) {  
 this.context = context;  
 }  
  
 */\*\*  
 \* FOLLOWING METHODS DEAL WITH THE ACTUAL USER OF THE APP  
 \*/* public boolean registerUser(String username, String password, String forename, String surname, String DOB, String weight, String height) {  
 Connection conn = null;  
 if (checkSqlInjection(username,password,forename,surname,DOB,weight,height)) {  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 *//creating a new salt and generating hash* String salt = *createSalt*();  
 String hashedPw = *getSecurePassword*(password,salt);  
 *//executing SQL statement* int resultset = statement.executeUpdate(  
 "INSERT INTO Users(username,password,firstname,surname,dateOfBirth,weight,height,salt) " +  
 String.*format*("VALUES ('%s','%s','%s','%s','%s','%s','%s','%s')",  
 username, hashedPw, forename, surname, DOB, weight, height,salt)  
 );  
 if (resultset == 0) {  
 Toast.*makeText*(this.context, "Could not create an account", Toast.*LENGTH\_SHORT*).show();  
 return false;  
 }  
 Toast.*makeText*(this.context, "Account created", Toast.*LENGTH\_SHORT*).show();  
  
 return true;  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException | NoSuchAlgorithmException|NoSuchProviderException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this.context, "Could not connect to database", Toast.*LENGTH\_SHORT*).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
 else{  
 return false;  
 }  
 }  
  
 public Boolean login(String username, String password) {  
 *//handles login validation process* Connection conn = null;  
 if (checkSqlInjection(username,password)) {  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 *//getting salt from database* String salt = getSalt(username);  
 if (salt !=null) {  
 ResultSet resultset = null;  
 *//executing SQL statement  
 //getting hashed password* String hashedPw = *getSecurePassword*(password,salt);  
 resultset = statement.executeQuery(  
 "SELECT UserID, firstname, surname, dateOfBirth, weight, height " +  
 "FROM Users " +  
 String.*format*("WHERE username = '%s' AND password = '%s'", username, hashedPw)  
 );  
  
 try {  
 addResult(resultset, 6);  
 if (this.getResult().isEmpty()) {  
 Toast.*makeText*(this.context, "Login Unsuccessful", Toast.*LENGTH\_SHORT*).show();  
 return false;  
 }  
 Toast.*makeText*(this.context, "Login Successful", Toast.*LENGTH\_SHORT*).show();  
 return true;  
 } catch (Exception e) {  
 Toast.*makeText*(this.context, "Username or Password incorrect", Toast.*LENGTH\_SHORT*).show();  
 return false;  
 }  
 }  
 Toast.*makeText*(this.context,"User does not exist",Toast.*LENGTH\_SHORT*).show();  
 return false;  
  
  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 *//if connection throws exception, login failed and false is returned* Toast.*makeText*(this.context, "Could not connect to database", Toast.*LENGTH\_SHORT*).show();  
 e.printStackTrace();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
 else{  
 return false;  
 }  
 }  
  
 public boolean updateUser(String username, String password, String forename, String surname, String DOB, String weight, String height) {  
 Connection conn = null;  
 if (checkSqlInjection(username,password,forename,surname,DOB,weight,height)) {  
 try {  
 conn = createNewConnection();  
 *//getting stored salt* String salt = getSalt(username);  
 String hashedPw = *getSecurePassword*(password,salt);  
 Statement statement = conn.createStatement();  
 *//executing SQL statement* int resultset = statement.executeUpdate(  
 "UPDATE Users " +  
 String.format("SET username = '%s',password = '%s',firstname = '%s',surname = '%s',dateOfBirth = '%s',weight = '%s',height = '%s' ",  
 username, hashedPw, forename, surname,  
 DOB, weight, height  
 ) +  
 String.format("WHERE Users.UserID = '%s'", User.getUserID().toString())  
 );  
 if (resultset == 0) {  
 *//could not save activity* return false;  
 }  
 *//activity was saved successfully* return true;  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 e.printStackTrace();  
 Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
 else{  
 return false;  
 }  
 }  
  
  
 public boolean deleteAccount(int userID) {  
 Connection conn = null;  
  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 *//executing SQL statement* int resultset = statement.executeUpdate(  
 String.format(Locale.getDefault(), "DELETE FROM Users WHERE UserID = '%d'", userID)  
 );  
 if (resultset == 0) {  
 *//could not delete account* return false;  
 }  
 *//account was deleted successfully* return true;  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 e.printStackTrace();  
 Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 */\*\*  
 \* FOLLOWING METHODS DEAL WITH HANDLING ACTIVITIES  
 \*/* public boolean saveActivity(String exercise, String currDate, String timestarted, String duration, String calories, String steps, String distance, String reps) {  
 *//AS SAVING ACTIVITY does not receive direct user input  
 //not requried to check for SQL injection* Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 int resultset = 0;  
 *//executing SQL statement* if (reps == null) {  
 *//activity is either walking, running or treadmill* resultset = statement.executeUpdate(  
 "INSERT INTO Activity (ExerciseID, UserID,Date,timeStarted,duration,calories,steps,distance) " +  
 String.format("VALUES (%s,%s,'%s','%s','%s','%s','%s','%s');",  
 ("(SELECT Exercise.ExerciseID FROM Exercise WHERE Exercise.Name = '" + exercise + "')"),  
 ("(SELECT Users.UserID FROM Users WHERE Users.username = '" + User.getUsername() + "')"),  
 currDate, timestarted, duration, calories, steps, distance)  
 );  
  
 } else {  
 *//activity is either walking, running or treadmill* resultset = statement.executeUpdate(  
 "INSERT INTO Activity (ExerciseID, UserID,Date,timeStarted,duration,calories,reps) " +  
 String.format("VALUES (%s,%s,'%s','%s','%s','%s','%s');",  
 ("(SELECT Exercise.ExerciseID FROM Exercise WHERE Exercise.Name = '" + exercise + "')"),  
 ("(SELECT Users.UserID FROM Users WHERE Users.username = '" + User.getUsername() + "')"),  
 currDate, timestarted, duration, calories, reps)  
 );  
 }  
 if (resultset == 0) {  
 *//could not save activity* return false;  
 }  
  
 *//activity was saved successfully* return true;  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 e.printStackTrace();  
 Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean readActivities() {  
  
 Connection conn = null;  
 *//get all activities associated with the user's ID* try {  
 *//connecting to database* conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 *//executing SQL statement* ResultSet resultset = statement.executeQuery(  
 "SELECT Activity.ActivityID, Exercise.Name, Activity.Date, Activity.timeStarted, Activity.duration, Activity.calories, Activity.steps, Activity.distance, Activity.reps " +  
 "FROM Exercise, Activity " +  
 String.format("WHERE Activity.UserID = (SELECT Users.UserID FROM Users WHERE Users.username = '%s') ", User.getUsername()) +  
 "AND Exercise.ExerciseID = Activity.ExerciseID " +  
 "ORDER BY Activity.Date DESC, Activity.timeStarted ASC;"  
 );  
  
 try {  
 *//activities read in form:  
 //"exercise name", "date", "time", "duration", "calories","steps","distance","reps"* addResult(resultset, 9);  
 return true;  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 return false;  
 }  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 *//if connection throws exception, login failed and false is returned* e.printStackTrace();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean deleteActivity(int ActivityID) {  
 Connection conn = null;  
 try {  
 *//connecting to database* conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 *//executing SQL statement* int resultset = statement.executeUpdate(  
 String.format(Locale.getDefault(), "DELETE FROM Activity WHERE ActivityID = '%d'", ActivityID)  
 );  
 if (resultset == 0) {  
 *//could not delete activity* return false;  
 }  
 *//activity was deleted successfully* return true;  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean getFriendsActivities(Integer duration, Set<Integer> friendsList) {  
  
 *//by using WHERE IN VALUES(), we can limit which activities are requested  
 // in format ('id1','id2',....)* StringBuilder sqlCondition = new StringBuilder("(");  
 for (Integer id : friendsList) {  
 sqlCondition.append(String.format(Locale.getDefault(), "'%d',", id));  
 }  
 *//including user's own UserID to show where their relative position is on leaderboard* sqlCondition.append(String.format(Locale.getDefault(), "'%d',", User.getUserID()));  
 sqlCondition.deleteCharAt(sqlCondition.length() - 1);  
 sqlCondition.append(") ");  
  
 System.out.println(sqlCondition);  
 Connection conn = null;  
 *//get request to database for all activities done for private leaderboard  
 //includes userID, and first name corresponding to each activity* try {  
 *//connecting to database server* conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 if (duration == 0) {  
 *//executing SQL statement  
 //requesting leaderboard of all time* resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID AND Activity.UserID IN " + sqlCondition +  
 "ORDER BY Activity.Date DESC;"  
 );  
 } else if (duration == 1) {  
 *//executing SQL statement  
 //requesting leaderboard from past day* long millis = System.currentTimeMillis();  
 Date date = new java.sql.Date(millis);  
 resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID AND Activity.UserID IN " + sqlCondition + "AND " +  
 String.format("Activity.Date = '%s' ", date.toString()) +  
 "ORDER BY Activity.Date DESC;"  
 );  
 } else if (duration == 30) {  
 *//requesting leaderboard from past 30 days* long millis = System.currentTimeMillis();  
 millis = millis - 2592000000L;  
 Date date = new java.sql.Date(millis);  
 resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID AND Activity.UserID IN " + sqlCondition + "AND " +  
 String.format("Activity.Date >= '%s' ", date.toString()) +  
 "ORDER BY Activity.Date DESC;"  
 );  
 }  
 try {  
 *//dealing with multiple rows* addResult(resultset, 2);  
 return true;  
 }  
 catch (Exception e){  
 return false;  
 }  
  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 *//if connection throws exception, login failed and false is returned* Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 e.printStackTrace();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
  
 public boolean getAllActivities(Integer duration) {  
 Connection conn = null;  
 *//get request to database for all activities done for public leaderboard  
 //includes userID, and first name corresponding to each activity* try {  
 *//connecting to database server* conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 if (duration == 0) {  
 *//executing SQL statement  
 //requesting leaderboard of all time* resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID " +  
 "ORDER BY Activity.Date DESC;"  
 );  
 } else if (duration == 1) {  
 *//executing SQL statement  
 //requesting leaderboard from past day* long millis = System.currentTimeMillis();  
 Date date = new java.sql.Date(millis);  
 resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID AND " +  
 String.format("Activity.Date = '%s' ", date.toString()) +  
 "ORDER BY Activity.Date DESC;"  
 );  
 } else if (duration == 30) {  
 *//requesting leaderboard from past 30 days* long millis = System.currentTimeMillis();  
 millis = millis - 2592000000L;  
 Date date = new java.sql.Date(millis);  
 resultset = statement.executeQuery(  
 "SELECT Users.username, Activity.calories " +  
 "FROM Activity, Users " +  
 "WHERE Activity.UserID = Users.UserID AND " +  
 String.format("Activity.Date >= '%s' ", date.toString()) +  
 "ORDER BY Activity.Date DESC;"  
 );  
 }  
 try {  
 *//dealing with multiple rows* addResult(resultset, 2);  
 return true;  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 return false;  
 }  
  
 } catch (SQLException | IllegalAccessException | InstantiationException | ClassNotFoundException e) {  
 *//if connection throws exception, login failed and false is returned* e.printStackTrace();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
  
 */\*\*  
 \* FOLLOWING METHODS DEAL WITH FRIENDS SYSTEM IN THE APP  
 \*/* public boolean getUsers(String name) {  
 Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 *//getting all users on the database, based on search  
 //removing user which has the same userID as the user currently in the app  
 //(you cannot be a friend of yourself)* resultset = statement.executeQuery(  
 "SELECT UserID, firstname, surname, username " +  
 "FROM Users WHERE (firstname LIKE " +  
 "'%" + name + "%' " +  
 "OR surname LIKE " + "'%" + name + "%' " +  
 "OR username LIKE " + "'%" + name + "%') " +  
 String.format(Locale.getDefault(), "AND UserID <> '%d' ", User.getUserID())  
  
 );  
 try {  
 addResult(resultset, 4);  
 return true;  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 return false;  
 }  
 } catch (Exception e) {  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean getFriends() {  
 Connection conn = null;  
 try {  
 *//checking both columns for any friendships between users* conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 resultset = statement.executeQuery(  
 "SELECT Friendship.User2ID, Users.firstname, Users.surname, Users.username " +  
 "FROM Friendship, Users " +  
 String.format(Locale.getDefault(), " WHERE Friendship.USER1ID = '%d' ", User.getUserID()) +  
 "AND Users.UserID = Friendship.USER2ID"  
 );  
 try {  
 addResult(resultset, 4);  
 User.getFriendsList().clear();  
 return true;  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 return false;  
 }  
  
 } catch (Exception e) {  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean addFriend(int user1, int user2) {  
 *//adding friend relationship to database  
 //where user1 is user who initiated friendship* Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 int resultset = 0;  
 resultset = statement.executeUpdate(  
 "INSERT INTO Friendship " +  
 String.format(Locale.getDefault(), "VALUES ('%d','%d');", user1, user2)  
 );  
 if (resultset == 0) {  
 return false;  
 }  
 *//adding to User class set of friends* User.addFriendsList(user2);  
 return true;  
 } catch (Exception e) {  
 Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
 public boolean removeFriend(int user1, int user2) {  
 *//adding friend relationship to database  
 //where user1 is user who initiated friendship* Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 int resultset = 0;  
 resultset = statement.executeUpdate(  
 "DELETE FROM Friendship " +  
 String.format(Locale.getDefault(), "WHERE User1ID = '%d' AND User2ID = '%d'", user1, user2)  
 );  
 if (resultset == 0) {  
 return false;  
 }  
 *//removing from User set of friends* User.removeFriendsList(user2);  
 return true;  
 } catch (Exception e) {  
 Toast.makeText(this.context, "Could not connect to database", Toast.LENGTH\_SHORT).show();  
 return false;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
  
  
 public ArrayList<String> getResult() {  
 return result;  
 }  
  
 public void addResult(ResultSet resultset, int columns) throws SQLException {  
 *//dealing with multiple rows* while (resultset.next()) {  
 String row = "";  
 for (int i = 1; i <= columns; i++) {  
 *//adding result to dbhelper* row = row + resultset.getString(i) + " ";  
 }  
 *//moving to next row (if there is any)* this.result.add(row);  
 }  
  
 }  
  
 private Connection createNewConnection() throws SQLException, ClassNotFoundException, InstantiationException, IllegalAccessException {  
 Connection conn = null;  
  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.setThreadPolicy(policy);  
 Class.forName("net.sourceforge.jtds.jdbc.Driver").newInstance();  
 *//connecting to database server* conn = DriverManager.getConnection(url);  
 return conn;  
 }  
  
 private void closeConnection(Connection conn) {  
 try {  
 if (conn != null) {  
 conn.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private Boolean checkSqlInjection(String...args){  
 *//check for any sql commands from any parameters given in args* for (String string : args){  
 if (string.contains("SELECT") || string.contains("DROP") || string.contains("DELETE") || string.contains("INSERT") || string.contains("UPDATE") ||  
 string.contains("select") || string.contains("drop") || string.contains("delete") || string.contains("insert") || string.contains("update")){  
 Toast.makeText(context, "You must not enter any SQL commands!", Toast.LENGTH\_SHORT).show();  
 *//sql command detected, returns false* return false;  
 }  
 }  
 *//no commands return true* return true;  
 }  
  
 public void clearResults() {  
 this.result.clear();  
 }  
  
  
 */\*\*  
 \* Following methods dealing with hashing a secure password  
 \* https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/  
 \*/* private static String getSecurePassword(String passwordToHash, String salt) {  
 String generatedPassword = null;  
 try {  
 *// Create MessageDigest instance for MD5* MessageDigest md = MessageDigest.getInstance("MD5");  
 *// Add password bytes to digest* md.update(salt.getBytes());  
 *// Get the hash's bytes* byte[] bytes = md.digest(passwordToHash.getBytes());  
 *// This bytes[] has bytes in decimal format;  
 // Convert it to hexadecimal format* StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < bytes.length; i++) {  
 sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16)  
 .substring(1));  
 }  
 *// Get complete hashed password in hex format* generatedPassword = sb.toString();  
 } catch (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
 return generatedPassword;  
 }  
  
  
 private static String createSalt() throws NoSuchAlgorithmException, NoSuchProviderException{  
 *//Creating a salt for a new user  
 // SecureRandom generator* SecureRandom sr = SecureRandom.getInstance("SHA1PRNG", "AndroidOpenSSL");  
 *// Create array for salt* byte[] salt = new byte[16];  
 *// Get a random salt* sr.nextBytes(salt);  
 *// return salt* return salt.toString();  
 }  
  
 private String getSalt(String username){  
 *//retrieving salt from db of existing user* Connection conn = null;  
 try {  
 conn = createNewConnection();  
 Statement statement = conn.createStatement();  
 ResultSet resultset = null;  
 resultset = statement.executeQuery(  
 "SELECT salt FROM Users " +  
 String.*format*("WHERE username = '%s'", username)  
 );  
 resultset.next();  
 String salt = resultset.getString(1);  
 if (salt.isEmpty()){  
 return null;  
 }  
 return salt;  
 }  
 catch (Exception e) {  
 return null;  
 } finally {  
 closeConnection(conn);  
 }  
 }  
}

### 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"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.example.exercisetracker">  
  
 <uses-permission  
 android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"  
 tools:ignore="ScopedStorage" />  
 <uses-permission  
 android:name="android.permission.READ\_EXTERNAL\_STORAGE"  
 tools:remove="android:maxSdkVersion" />  
 <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" />  
  
 <uses-feature android:name="android.hardware.camera.any" />  
 <application  
 android:name=".other.BaseApp"  
 android:allowBackup="false"  
 android:icon="@mipmap/appicon\_foreground"  
 android:label="@string/app\_name"  
 android:requestLegacyExternalStorage="true"  
 android:roundIcon="@mipmap/appicon\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.ExerciseTracker">  
 <activity  
 android:name=".login.LogInScreen"  
 android:exported="true"  
 android:screenOrientation="portrait">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:name=".activities.MainActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"></activity>  
 <activity  
 android:name=".activities.RunningActivity"  
 android:screenOrientation="portrait"></activity>  
 <activity  
 android:name=".activities.TreadmillActivity"  
 android:screenOrientation="portrait"></activity>  
 <activity  
 android:name=".activities.PushUpActivity"  
 android:fitsSystemWindows="true"  
 android:screenOrientation="portrait"></activity>  
 <activity  
 android:name=".activities.SquatsActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"></activity>  
 <activity  
 android:name=".login.RegisterUserActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"></activity>  
 <activity  
 android:name=".activities.WalkingActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"></activity>  
 <activity  
 android:name=".activities.AddFriendsActivity"  
 android:screenOrientation="portrait"  
 android:windowSoftInputMode="adjustPan"></activity>  
  
 <service android:name=".other.ExerciseService" />  
 </application>  
  
</manifest>

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>

history\_card\_layout

Defines the attributes used for cards containing the activity details. in the history fragment. Each activity in the history is instantiated with this card layout.

*<?*xml version="1.0" encoding="utf-8"*?>*<com.google.android.material.card.MaterialCardView 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="wrap\_content"  
 android:layout\_margin="2dp"  
 app:cardBackgroundColor="@color/white"  
 app:cardCornerRadius="15dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:cardUseCompatPadding="true"  
 app:strokeWidth="3dp">  
  
 <RelativeLayout  
  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 *<!--ImageVIew to display exercise Image-->* <ImageView  
 android:id="@+id/ExeciseImg"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="10dp"  
 android:contentDescription="@string/app\_name"  
 android:padding="5dp"  
 android:src="@drawable/running" />  
  
 *<!--Text View to display Exercise Name-->* <TextView  
 android:id="@+id/ExerciseName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="10dp"  
 android:layout\_toEndOf="@id/ExeciseImg"  
 android:text="Exercise Name"  
 android:textColor="@color/black"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 *<!--Text VIew to display Exercise Details-->  
 <!--Image used in present in drawable folder-->* <TextView  
 android:id="@+id/ExerciseDesc"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/ExerciseName"  
 android:layout\_marginStart="10dp"  
 android:layout\_toEndOf="@id/ExeciseImg"  
 android:drawablePadding="2dp"  
 android:fontFamily="@font/gothic"  
 android:text="Description" />  
  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/ExerciseDeleteBtn"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:layout\_alignBottom="@id/ExeciseImg"  
 android:layout\_marginStart="10dp"  
 android:layout\_toEndOf="@id/ExeciseImg"  
 android:fontFamily="@font/gothicbb"  
 android:text="Delete" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/ExerciseDetailsBtn"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:layout\_alignBottom="@id/ExeciseImg"  
 android:layout\_alignParentEnd="true"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_toEndOf="@id/ExerciseDeleteBtn"  
 android:fontFamily="@font/gothicbb"  
 android:text="DETAILS" />  
 </RelativeLayout>  
  
</com.google.android.material.card.MaterialCardView>

friends\_card\_layout

Defines the attributes used for cards containing the friends’ details. in the AddFriendsActivity.java class. Each friend is instantiated with this card layout.

*<?*xml version="1.0" encoding="utf-8"*?>*<com.google.android.material.card.MaterialCardView 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="wrap\_content"  
 android:layout\_margin="2dp"  
 app:cardBackgroundColor="@color/white"  
 app:cardCornerRadius="15dp"  
 app:cardElevation="8dp"  
 app:cardMaxElevation="10dp"  
 app:cardPreventCornerOverlap="true"  
 app:cardUseCompatPadding="true"  
 app:strokeWidth="3dp">  
  
 <RelativeLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 *<!--Text View to display User's Name-->* <TextView  
 android:id="@+id/usernameID"  
  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="10dp"  
 android:drawablePadding="2dp"  
 android:fontFamily="@font/gothici"  
 android:text="username id" />  
  
 <TextView  
 android:id="@+id/userRealName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/usernameID"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginBottom="10dp"  
 android:layout\_toStartOf="@id/addFriendBtn"  
 android:text="User's Name"  
 android:textColor="@color/black"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/addFriendBtn"  
 app:cornerRadius="15dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentEnd="true"  
 android:layout\_centerVertical="true"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="10dp"  
 android:text="Add Friend" />  
  
  
 </RelativeLayout>  
  
</com.google.android.material.card.MaterialCardView>

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">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>  
 <string name="met\_squats">3.5</string>  
  
 <string-array name="quotes">  
 *<!-- Motivational Quotes -->  
 <!--https://www.goodhousekeeping.com/health/fitness/g35310050/workout-quotes/?slide=7-->  
 <!--https://www.quoteambition.com/workout-quotes/-->* <item>If you don’t find the time, if you don’t do the work, you don’t get the results.</item>  
 <item >You’re only one workout away from a good mood.</item>  
 <item >If something stands between you and your success, move it. Never be denied.</item>  
 <item>It’s going to be a journey. It’s not a sprint to get in shape.</item>  
 <item>Today I will do what others won’t, so tomorrow I can accomplish what others can’t.</item>  
 <item>The hard days are the best because that’s when champions are made, so if you push through, you can push through anything.</item>  
 <item>The pain you feel today is the strength you feel tomorrow.</item>  
 <item>Motivation is what gets you started. Habit is what keeps you going.</item>  
 <item>The difference between the impossible and the possible lies in a person’s determination</item>  
 <item>We can push ourselves further. We always have more to give.</item>  
 <item>Your body can stand almost anything. It’s your mind that you have to convince</item>  
 <item>Success isn’t always about greatness. It’s about consistency. Consistent hard work gains success. Greatness will come</item>  
 <item>Train insane or remain the same.</item>  
 <item>Success starts with self-discipline</item>  
 <item>The body achieves what the mind believes.</item>  
 </string-array>  
  
 <string name="hide">Hide</string>  
 <string name="cancel">Cancel</string>  
 <string name="confirm">Confirm</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/gothicbb</item>  
 <item name="android:textColor">@color/black</item>  
 <item name="android:textCursorDrawable">@color/black</item>  
 <item name="android:navigationBarColor">@color/main\_colour</item>  
 <item name="buttonTint">@color/main\_colour\_on\_primary</item>  
 <item name="materialAlertDialogTheme">@style/AlertDialogTheme</item>  
 <item name ="android:indeterminateTint">@color/main\_colour\_on\_primary</item>  
 <item name="android:windowSplashScreenBackground" tools:targetApi="s">@color/main\_colour</item>  
  
  
 </style>  
  
 *<!-- Alert Dialog theme. -->* <style name="AlertDialogTheme" parent="ThemeOverlay.MaterialComponents.Dialog.Alert">  
 <item name="buttonBarNegativeButtonStyle">@style/NegativeButtonStyle</item>  
 <item name="buttonBarPositiveButtonStyle">@style/PositiveButtonStyle</item>  
 </style>  
  
 <style name="NegativeButtonStyle" parent="Widget.MaterialComponents.Button.TextButton.Dialog">  
 <item name="android:textColor">@color/main\_colour\_on\_primary</item>  
 </style>  
  
 <style name="PositiveButtonStyle" parent="Widget.MaterialComponents.Button.TextButton.Dialog">  
 <item name="android:textColor">@color/main\_colour\_on\_primary</item>  
 </style>  
</resources>

build.gradle

plugins {  
 id 'com.android.application'  
}  
  
android {  
 compileSdk 31  
 useLibrary 'org.apache.http.legacy'  
 defaultConfig {  
 applicationId "com.example.exercisetracker"  
 minSdk 21  
 targetSdk 31  
 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:1.4.0'  
 implementation 'androidx.appcompat:appcompat:1.4.0'  
 implementation 'com.google.android.material:material:1.4.0'  
 implementation 'com.google.mlkit:pose-detection:17.0.1-beta6'  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.2'  
 implementation files('libs\\mysql-connector-java-5.1.4-bin.jar')  
 implementation files('libs/mssql-jdbc-10.2.0.jre8.jar')  
 implementation files('libs/jtds-1.3.1.jar')  
 testImplementation 'junit:junit:4.+'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
 implementation 'com.google.guava:guava:27.0.1-android'  
}

#### Graphical user interface Description automatically generated with low confidenceAndroid Project Structure:

A picture containing text

Description automatically generated

# TESTING

The following YouTube video contains all video recordings of my tests, including any other videos that were recorded during the development of my application:

**YOUTUBE LINK HERE**

The following table details the full list of tests that I have conducted to test whether or not the exercise detection app met the requirements I made during the analysis and design stages of the project. They include both quantitative and qualitative tests, helping me to gauge the technicality and completeness of my final technical solution.

## Key Objectives Tests:

|  |  |  |  |
| --- | --- | --- | --- |
| Test No | Description | Outcome | Timestamp |
| 1.1  Login | With an existing account, can the user log in with access to their data? | Passed |  |
| 1.2  Create Account | Can the user create an entirely new account, and the given details are saved properly in the database? | Passed |  |
| 1.3  Stay Logged In | Does the app keep the user logged with the saved details, when the checkbox “Remember Me” is selected when logging in? | Passed |  |
| 1.4  Logout | Can the user log out of the app safely, and remove all information for the next user to log in? | Passed |  |
| 2.1  Load User Account Details | Can the app properly load the account details from the database? | Passed, however if the details were edited elsewhere outside of the app the details would not be updated unless relogged in. Should re update the user’s details every time they navigate to the settings menu to keep it up to date preventing differing details. |  |
| 2.2  Edit User Details | Does the app safely edit the user’s account details and save them to the database? | Passed, although could not change the username due to it being the primary key. |  |
| 2.3  Delete User Account | When requested, will the user’s details and all their following activities and data delete from the database, without affecting other users? | Passed. |  |
| 3.1  Load Leader boards | Does the leader board load up with the correct information? When switching to a public/friends-only leader board? When switching the time frame of the leader board? | Passed. |  |
| 3.2  Load, remove and add friends | Can the user add/remove friends from their friends list? Will the leader board update when a new friend is added/removed? | The simple friends’ system worked, however is not scalable, as anyone can add anyone as a friend. A verification system should be implemented a better method of storing friend requests is required. |  |
| 4.1  Load Exercise History | Can the app correctly load the user’s previous exercise history, in date and time order? | Passed. |  |
| 4.2  View statistics of exercise history | Can the user be able to view the statistics with the corresponding activity? | Passed. Running/walking could be improved to show the actual route the user took, however would need to find a way to store objects and bytes onto the cloud outside of the database. |  |
| 4.3  Delete Exercise History | Can the user safely delete a given exercise from their history? | Passed. |  |
| 5.1  Track a running exercise | Testing for accuracy against other market running apps |  |  |
| 5.2  Track a walking exercise | Testing for accuracy against other market running apps |  |  |
| 5.3  Track a treadmill exercise | Testing for accuracy against other market running apps |  |  |
| 5.4  Track a push-up exercise | Testing for accuracy against the actual number of reps | When using on my own android device, was quite successful producing around 80-90% accuracy, however, some other users and devices experienced lower accuracy |  |
| 5.5  Track squats exercise | Testing for accuracy against the actual number of reps | The squats exercise was the more accurate tracking compared to push ups and proved to be very effective. |  |
| 5.6  Save activity to database | Can the app, once the activity has been completed, save the activity details to the database. | Passed. Exercises were successfully saved to database, as long as user had an internet connection. |  |
| 5.7  Audio feedback during exercises | If audio feedback works for exercises and when the user wants to mute the audio. (Reads aloud motivational quotes and congratulates user when exercise finishes) | Passed. Could have included motivational music as well, or a countdown before starting the exercise. |  |

## Boundary and Erroneous Tests:

|  |  |  |  |
| --- | --- | --- | --- |
| Test No | Description | Outcome | Timestamp |
| 1.1  Login with fake credentials | When using account details which do not match any within the database, how does the app respond? | Passed. Showed to users the details were incorrect and stopped them from logging in. |  |
| 1.2  Create an Account with an existing username | When trying to create an account with the same username, how will the app respond? | Passed. Disclaimer shown to user that it is not possible to create that account. |  |
| 1.3  Create Account with SQL Injection | When trying to enter an SQL command in any user-entered fields, how will the app respond? Does the request go through to the server or will the app prevent this? | Passed. Any form of SQL statement that was injected was detected and prevented from executing. |  |
| 1.4  Creating account but not meeting requirements | Requirements:  All fields contain no spaces, username and password must be at least 8 characters long. The username is a maximum of 16 characters. | Passed. No user could not create an account not meeting those requirements. |  |
| 2.1  Updating Accounts with SQL injection | When trying to enter an SQL command in any user-entered fields, how will the app respond? Does the request go through to the server or will the app prevent this? | Passed. Any form of SQL statement that was injected was detected and prevented from executing. |  |
| 2.2  Updating account but not meeting requirements | Requirements:  All fields contain no spaces, username and password must be at least 8 characters long. The username is a maximum of 16 characters. | Passed. No user could not create an account not meeting those requirements |  |
| 3.1  Loading leader board when no activities are stored | What if the database returns an empty result to the app? How will the app respond? | Disclaimer shown to the user that no data was available. |  |
| 3.2  Loading friends’ leader board when friends list is empty | What if the database returns an empty result to the app? How will the app respond? | Disclaimer shown to the user that no exercises were stored on the database. |  |
| 4.1  Not fulfilling requirements for saving an activity | Must be longer than 1 minute for all activities. For push and squats, must at least do 5 reps. For running, treadmill and walking must have at least walked 10 steps.  Prevents users from using SQL server resources for negligible exercises. |  |  |
| 4.2  Permissions denied when starting activity | Push up and Squats exercises require access to the camera, due to the camera vision processing required.  Running and walking exercises require a location to track distance. |  |  |
| 4.3  A user travelling at unusual speeds | When the user is in a vehicle, going faster than speed is humanly possible. | Failed, no mechanism to stop exercise from tracking |  |
| 4.4  The user is in a dark room for rep detection | Can the app still track a user if the landmarks are not clearly defined due to lack of light? |  |  |
| 4.5  Reps done at a very fast pace | If the user attempts to push up/squat at a fast pace, can the app still track the number of reps? |  |  |
| 5.1  The user does not have an internet connection | Can the app still function, even if the user does not have an internet connection? | Should create an option when opening the app to exercise without an internet connection/an account |  |
| 5.2  Database shuts down | Will the app function if the database no longer exists/has no access to it? |  |  |

# EVALUATION

## Critical Path

From my testing, I can safely say that I was able to complete the critical functionality that I set out in my analysis for my application. However, not all my objectives were met. These objectives were related to security and general user interface and design.

## Third-Party Feedback

To collect third-party feedback from the app, I asked my friends and relatives who fit my criteria of being young and fit. The age ranges of my sample group were from 19-32 years old with varying different fitness lifestyles all who used the Android OS. I allowed them to fiddle with the app for a week or two and asked them to fill in this questionnaire: <https://forms.gle/uMnMHQ6udgTjrzDK9>

## Changes and Improvements

From the following feedback and from the results of my own testing here a list of improvements and changes that I could implement for the future.

1. Allow the running/treadmill/walking activities to run in the background, even when the user switches off their device or navigates away from the app. (This would require the use of Android Services and a communicator between the service and the app. Moreover, an interchange mechanism would be required to handle when an activity is resumed/closed etc.)
2. Add a notification service to keep users updated when working out (Again would require Android services)
3. Variable calorie calculations, not just based on the duration of exercise, but also on how much effort the user is putting in (derived from number of steps/reps).
4. Allow the user to have a calorie calculated, even when they have decided not to enter in their weight/height, by using an average weight/height for someone their age.
5. Secure Azure database further, by using hashing for all sensitive information of the user, such as their date of birth, their weight and height
6. Allow user to change overall theming colour of the app, as well as including dark mode.
7. Have better instructions towards users, adding a dialog when the user first opens the app to show them tips/steps to navigation, and when first tracking an exercise, informing them on how to use the app to track it, especially important for the camera driven exercises.
8. Have a bigger variety of exercises and the ability for a random exercise option where the app would cycle through random list of exercises for the user to do and be able to switch between the tracking options.
9. Include a better algorithm in detecting push ups and squats which can judge the number of reps even in different perspectives of how the body is positioned and where the camera is positioned. (This is much more complex and requires far deeper level of machine learning)

## GitHub Repository

My project GitHub repository for this entire project can be found at:

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

(Only for permitted for personal use only)

The following graph shows the commits I made over the course of starting and completing the project (includes commits for documentation). This consisted of around 200 commits over the span of around 5-6 months starting from the July/August of 2021 to the end of February 2022.

A screenshot of a video game

Description automatically generated with low confidence

## Conclusion

Overall, to conclude this project, I have felt that it has been an intense project, especially in documenting all of the project and the time spent on this project, when conflicting with time spent for other studies and revision. However, this has been a truly rewarding experience. This is especially the case, as I was able to achieve most of my key objectives, producing a well-rounded exercise tracker app that could track exercises.

Moreover, the unique aspect of using a device’s camera to detect squats/push ups using Google’s machine learning kit was very exciting to see it working, after months of progression. This is something which I held unique to this exercise tracking app compared to all the other tracking apps on the market.

Furthermore, the flexibility and scalability of this app in the future can be something I could work on, because of how easy it is to publish this app on the Google Play store as an official app, which could be very exciting.

Lastly, the wide range of areas of computing that I touched on in one project was very eye-opening, from parallel processing to SQL, to machine learning and User Interface. This has helped widen and strengthen my knowledge of Computer Science, covering the wide spectra of Computer Science.

# APPENDIX

1. 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

1. 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.

1. Results of different sampling rates when taking 20 steps
2. 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);  
}

1. Scatter chart

   Description automatically generated with medium confidenceA picture containing graphical user interface

   Description automatically generatedA picture containing shape

   Description automatically generatedXML Design Previews midway through developing technical solution

Text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generatedA picture containing graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

1. Database Design

Graphical user interface

Description automatically generated

Current Database Model (As of 16/12/21)

# 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]. |