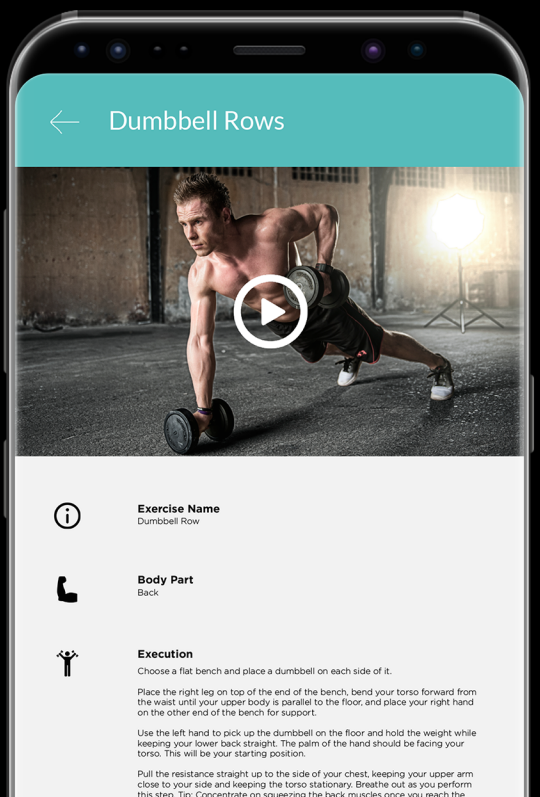
Workout Tracker

Cardiff University – School of Computer Science & Informatics

Khaledur Rahman
C1322141/CM3203

Supervisor: Andrew Jones
Moderator: Dave Marshall
Credits: 40 - One Semester Project



CARDIFF UNIVERSITY

GET IT ON Google Play

Abstract

The fitness industry has been continuing to grow year after year with more individuals becoming health conscious. Today's apps currently don't have a simple all in one application in helping users progress in the gym.

As a result, an Android application has been developing in this project using Java programming language, to help users their gym life. The user will be able to track workouts for a particular day with the ability to add exercises. Each exercise can have a corresponding detail of sets and information about how to do the corresponding exercise. The YouTube API was used to help with how to do exercises. The project utilises NFC tag to help users track exercises whilst in the gym. Additionally, users can share the workout with their friends and track the workout together. The main aim of the project was to develop a fully functioning app with Firebase.

The mobile application was uploaded to the Play Store and user feedback was given on how the app performed, future implementations and changes to UI to make.

Acknowledgements

Thank you to first and foremost God who has given me the ability, knowledge and hope to complete the project during the difficult times. Thank you to His Messenger Muhammed (PBUH) for bringing the beauty of the religion to us. Without my faith I would not have hope and it is hope that drove me through this project.

Thank you to my parents Khadija Begum & Mukid Miah for giving me support and love over the course of my studies. Their kindness, love, and encouragement have allowed me to believe in myself and give my best in everything that I do.
Thank you to my sister Sadia Begum and my brother Habib Rahman for being there and giving me unconditional love and help without the need to ever ask for it.

Thank you to Andrew Jones for supervising the project and providing the support, time and effort during troubled and confused times. The feedback was given and time taken to listen has contributed hugely to the success of the project.

Thank you to my friends Dheeraj Thankachan, Hafizur Rahman, Farhaan Alam Khan, Shahad Rahman, Fariah Rahman, Rehana Begum & Jaffrin Khan for support and belief in myself.

Thank you to the users who took the time out to download the app, review it and give feedback. Without them, I would not have been able to talk about future prospects and ideas. Their insight and feedback were very valuable.

Table of Contents

Abstract 1

Acknowledgements 2

Introduction 5

- Outline 5
- Summary of Aims 5
- Project Importance 7
- Audience and Potential Beneficiaries 7
- Project Scope 8
- Approach 10
- Assumptions 10

Background 11

- Overview 11
- Questionnaire Research 12
- NFC 13
- Constraints To The Approach 14
- Existing Solutions 15
- Methods and Tools for Solution 16

Specification 20

- App Specification 20
- Essential Requirements 20
- Desirable Requirements 21
- User Interface Specification 22

Design 24

- Overview 24
- Storyboard 25
- Mock Ups 26
- Application Use Case 35

Architecture and System Design 38

- System Architecture 38
- UML Diagram 39
- Data Model 40
- Data Modelling Issues 44

Back-Stack Model 44
Firebase 45
Data Flow 46
Considerations 47
Implementation 48
Tools Used 48
Structure 48
Walk Through 50
Testing 63
Test Cases 63
User Testing 69
Testing Review 70
User Feedback 71
Initial Thoughts 71
App Upload Stats 71
Post User Feedback 73
App Reviews 75
Post User Review 76
Post Feedback Pivot from Initial Plan 77
Evaluation 78
Acceptance Criteria 78
Project Achievements 79
Poor Decisions Made & If Project Was Done Again 80
Design Quality 81
Future Improvements 82
What Was Learnt 83
Future Work 85
Conclusion 86
Bibliography 87
List of Figures 89

Introduction

Outline

The fitness industry has grown hugely over the past few years. What used to be an Olympic sport has now grown into a lifestyle for individuals. Tracking workouts are therefore vital for individuals so they can monitor their progress and thus make the correct adjustments. There are currently a number of fitness apps on Android and iOS that cater for the need for individuals but nothing that dominates the market with good user experience and relevant functionality. Individuals are currently using either their memory or writing down their progress using pen and paper. Having a simple application that allows users to track exercises for a particular day with the additional features of logging body weight and uploading progress photos has not yet been explicitly done.

Going to the gym has now also become 'social' with friends now training together. There is currently no app that offers users to be able to track their workouts together seamlessly making it easier to compete and progress. Searching for exercises is quite difficult. Especially since there are 1000s of exercises and finding the correct one can through a search bar can be long. With the use of NFC and a single swipe, a user is able to find the relevant exercise for a machine without having to worry about searching through the database.

The project as whole aims to develop a mobile application where users can log and track their fitness lifestyle.

Summary of Aims

The purpose of the projects is split into two areas: personal aims and project aims. They are split into two so that it can be distinguishable between the two what has been achieved.

Personal Aims

Personal aims are results that are hoped to be achieved by the end of the project for personal growth and development. The personal aims are as follows:

1. Have a working mobile app
2. Learn how to plan, design and build the app
3. Increase self-confidence in programming
4. Manage time effectively whilst working part time

Justification of personal aims

Having played around with mobile development, the opportunity to develop a fully functioning application has not been possible. Having the opportunity to develop an app that was a problem faced by the author is essential. Planning, designing and building a total app in all areas has not been done before other than in Year 2 of University in a module known as Group Project. Having the opportunity to look at all aspects as an individual and completing this is key. This relates to the third personal aim. Self-confidence is important for anyone to have and as large project have not be done, completing a project with a completed solution was important to give self-confidence. Working part time whilst balancing University can be difficult at the time. It essential to managing time effectively whilst developing a system with little prior knowledge.

Project Aims

The overall aim of the project is to develop a mobile application for gym goers to help them with their fitness lifestyle primarily workout tracking, which can aid them to make the better decision on their fitness lifestyle. The use of NFC tags will be explored to provide an easier experience for the target audience whilst working out.

In order to satisfy the project aims, the system will be developed for Android that will be used by gym goers. Users will be able to add exercises to created workouts and a simple search functionality will be implemented to search for the specific exercise. However, NFC tags will be implemented to add exercises with a single swipe. The user will be able to share their workouts with a list of friends when working out with them or to show adding the social element to the app. Other features of the application include adding their body weight and uploading progress photos from their gallery.

There are currently few applications that allow users to track exercises with NFC and share workouts with their friends. Adding these functionalities will allow users to track more and train more consistently in the gym. There are a number of features that can be developed, however, the foundation of the app will be developed giving the potential to be built upon.

Justification of Project Aims

Being an active gym member for the past 3 years, I've faced a number of problems whilst working out. A lot of the aims of the project have been based off problems that were faced. The current application that is used by myself is an app for personal trainers that their clients could use to monitor progress. As it currently stands, no application was found with clean design and simple user experience. The workout tracker feature came from the fact that I could use any app on the Play Store other than the one my personal trainer provides.

The progress photo and body weight feature came about as an additional feature. Adding body weight and progress photo tracking came by having an understand of the fitness lifestyle. The research was conducted with other members of the gyms by asking them what they would like to see in an app in addition to workout tracking and they mentioned tracking their body weight extensively.

The NFC tag feature was obtained whilst working on in the gym. A QR code was visible on the exercise machine to track exercise but I was unsure of the app to use. This gave me the idea to track exercises with NFC but utilising it initially was the main concept.

Sharing workout with friends concept came from doing a doing a course online for Firebase. Firebase is known for the Realtime Database and integrating social elements into the app via Firebase was simpler to do compared to other databases.

Project Importance

The functionalities of the Android application isn't offering anything new that isn't already on the app store in its core functionality. The foundation of the application is to track workouts. The additional functionalities of adding exercises via NFC and adding friends to add their exercises, however, have not been introduced in any fitness app. NFC (Near Field Communication) is a simple way to add exercises with a simple swipe and although many phones don't incorporate it, it is a useful functionality to have for users when they are working out. The shared workout platform has not yet been introduced in any app. As mentioned before, working out has now become a social event and adding friends track workout with is key.

Audience and Potential Beneficiaries

The audience for my project assumes that they are active gym-goers who actively want to make a change to their fitness lifestyle by tracking workouts. An assumption is made that the users have no technical abilities and have not used a fitness application before thus ensuring clean and simple design.

The actual audience for the application is very wide as there are now more and more people going to the gym, as previously mentioned. Anyone who goes to the gym can essentially be a user who can use the application to track their workouts and in later stages see the progress for each exercise.

The target audience has no age restriction, however, to go to the most gym the minimum age is 16 usually. The application is primarily for users who are aged between 18 and 35 who are actively working out in the gym. The general design of the app is going to be clean and simple so it is easy to navigate around and once learned they can use without having to think too much.

Project Scope

The project will be released initially to be available for users in the UK via the Play Store. The primary object of the project is to allow users to track their fitness lifestyle. These include workouts, body weight, and photos. The following are the requirements to be completed by the end of the project:

1. User Sign Up & Log In
2. Track workouts
3. Add exercises to workouts and their corresponding sets
4. Add exercises using NFC
5. Display information about exercises
6. Share workouts with other users and allow them to track their exercises
7. Track their body weight
8. Upload photos to view their progress

To ensure that a fully functioning application is completed by the end of the project, the minimum of each feature (workout, body weight, and photo tracking) will be implemented with good design in mind and then built open modular to make it better.

How Requirements Were Selected

The project requirements were selected by facing the problems in the gym personally. There were no apps on the Play Store to help assist with workout tracking specifically to my needs so it felt best to develop a system to help alleviate this. Time is spent between sets resting which could be used to enter the weight and reps that were completed allowing users to track their progress for that specific exercise. Over time, all the information could be collected such as sets completed, volume, maximum weight, maximum reps and so forth. This is why tracking exercises and their sets were needed in the app. From being at the gym and looking at users printing out sheets of papers with their workouts or using notes in their phones showed how important it was to have a simple application which performs the functionality well with.

Research by Health Club Management [1] in 2016 showed that health clubs have more members than even before. No individual will go to the gym to look the same: people will either want to lose weight to add muscle primarily or just be healthier in general. This means a change in their body thus making progress photos an important requirement to have. Having one application that shows the progress of their fitness lifestyle was important and a key to having an all in one app for their fitness life.

The current app that is used personally doesn't have videos or description on how to perform exercises. This means that either a personal trainer has to be asked or a

Google search needs to be done on how to perform the exercises. Having gone through that made it a requirement on the app.

Having users upload their progress photos was important. Tracking body weight is just as important, to know how much a user weight on a particular day. Personally, I track my body weight every morning to see how much I weigh. The system that is currently used is excel spreadsheet which takes in previous 7 days weight and gives an average for that week. This is where the idea for tracking weights in-app came about.

The gym is becoming a social event, and although personally, I work out by myself, I do a workout with friends sometimes. Giving them access to the workout is a lot easier as they can know prior on what to do. This is where the idea of sharing workouts came from.

Adding exercises with near field communicated was selected initially for learning purposes and understanding of IoT and its potentially. Since researching online and seeing exercise machine implement QR code, having exercises added via NFC means it is a lot easier to swipe and add instantly.
More details about the project requirements and justification will be explained later.

App Desirability

The app has a lot of desirable features that could potentially help users in the fitness industry. For users who take gym seriously, training is very important to them making workouts a key element to their progression. By using a paper based system, everything will need to be done manually and there is a greater chance of errors being made. By having an app being able to link sets to the corresponding exercises: data such as maximum weight, volume, reps could be visualised and monitored. This is useful for users who could be entering competitions as the system could grow to learn and give advice on what to do.

No other application currently has NFC integration to add exercises. By enabling users to add exercise with a single swipe gives users more of a reason to quickly get their phones out and track the exercise.

Sharing workouts with friends has not also been implemented by other applications. Other apps can share the workouts but not add their friend to the workout to add their own exercises and track together. Having these unique features help with wanting to download the app and wanting to train together with their friends. This helps users train more and become fitter.

Approach

For the project development approach, the chosen development process that was initially decided was agile development with sprints for each functionality. This however changed to the feature-driven development (FDD [2]) after much deliberation. It was changed as it best suited the need of the project after reviewing the initial report.

In the initial plan, features were going to be developed in sprints and once completed, upload to the Play Store and get feedback from users. Whilst users were going to use the application, the sprints would be continued and updates were going to be added on accordingly. Once the base features were completed, feedback would have been taken from the users to steer the development according to their needs and requirements. The learning and development time took a lot longer than expected thus having to change the type of approach to the project.

The list of features was written in the initial plan which meant planning design and development was easier to do. A slight a refinement had to be done but the changes were minor. As the design was an important aspect of the app, FDD was the perfect model. Firebase, a database system by Google, was the chosen database system. The data was modelled based of the UI on the page. This correlates better with FDD because the design and development for each feature are done accordingly. The choice of why Firebase has been chosen will be explained later.

Changes from the Initial Plan

There were a number of changes in the objectives from the initial plan to achieve the aims. One of the biggest change that was undertaken was moving from MySQL database to Firebase (Firebase is explained further later). This meant instead of developing a solid architecture and API, learning Firebase and implementing it with Android was needed to be done.

Assumptions

There are a number of assumptions that have been made in developing the project:

- The end product will be a fully functional prototype with a visually appealing application
- The project scope will be limited to the United Kingdom
- There is an active internet connection, via either Wi-Fi or Data, on the user's Android smartphone. If not, the application will not be able to get all the data from the database
- The users Android SDK is 16 or above. This is based on the recommendation on Android Studio giving the app approximately targeting 95.2% of the devices on the Google Play Store

Background

Overview

Many mobile applications now utilise the convenience of carrying a mobile phone around everywhere for a variety of different reasons. Seeing an individual go to the gym is vital for companies to learn about an individual's lifestyle, habits, weight loss or gain progress and many other services that will enhance the quality for these companies. A user can benefit hugely from having their phone in the gym by listening to their own music, receiving messages and thus tracking their workouts.

Today's smartphones have the ability to be your own little personal trainer, nutrition coach, tracker, GPS and so much more! Newer devices and services are creating a workout that's more data-rich, smarter and more convenient than before. As more and more people are going to the gym to become fitter, healthier and 'cooler,' technology has now grown to help with these demands allowing companies to build Artificial Intelligence systems to learn a user's diet and make recommendations, wearables that track sleep, steps and heart rate and music that responds to the pace of walk or running speed. In a nutshell, technology is transforming the fitness industry.

Gyms are now incorporating applications to book classes and sessions through the app or online. Gyms are now so popular the unpopular 24-hour gyms are now becoming more and more popular. Tracking in its nature is very tedious to do. However, individuals have now adopted to tracking in the 21st century where users are now even doing video blogs everyday. MyFitnessPal is one of the leading apps in the fitness industry that allows users to search food and log them for each day. They can see the calories they've consumed, have left and even see the macronutrients for each of the individual food tracked. MyFitnessPal was bought out by Under Amour for $475 [3] million showing the user base for food tracking.

The best fitness application currently by 'CoachMag' is Strava. Strava has allowed users to track their cycling and in London alone 1.35 million [4] activities via cycling was recorded in 2015.

In the article for CoachMag, there currently isn't a single application that dominates the market in workout tracking. The above shows the importance of tracking in general and for a fitness industry 4.4 billion [5], you would think there would be.

Now there isn't an official reason as to why users aren't using a workout tracker. Assumed reasons could be culture, design, functionality and/or motivation.

A short survey, created by the author, was created to get details about the current situation in gyms and it's correlations with workout tracking. From the results, it showed that there was a 60% agreement to using a fitness app if it was available.

This project aims to develop a complete fitness application that includes tracking workouts, body weight and photos to fit the lifestyle of gym-goers.

Questionnaire Research

A questionnaire was created to find out better which the application was needed in the fitness industry as there are currently a number of applications already available on Android. The questionnaire was kept quite closed to find specific information about the industry and the culture that is in the gym.
(see appendix for detailed results)

Results

With a result of 193 responses who partook in the questionnaire the following are some of the results obtained.

Q: Do you currently use a fitness app to track workouts?
Yes: 72%
No: 28%

If they used an app the following was asked:

What app do you currently use? (Top 3)

1. *Trainerize: 6*
2. *Fitbit: 4*
3. *Strava: 3*

Is the app missing anything you'd like? (Top 3)

1. *Timer: 17*
2. *Tips: 18*
3. *Workout selected is limited: 12*

If they weren't using an app the following was asked:

How do you track your workouts? (Top 4)

1. *Mentally: 74 (53.2%)*
2. *I don't: 20 (14.4%)*
3. *Notes on my phone: 17 (12.2%)*
4. *Pen and paper: 7 (5%)*

Would you use an app to track your progress?

1. *Yes - 63.3%*
2. *No - 36.7%*

What would you like to see in an app that tracked workouts?

1. *Timers for between sets: 32 (23.9%)*
2. *Workout Plans: 48 (35.8%)*
3. *Body Measurements: 41 (30.6%)*
4. *Graphs that show progress: 41 (30.6%)*
5. *Clean & minimal design: 38 (28.4%)*
6. *All of the above: 43 (32.1%)*
7. *Other: 30 (22.4%)*

Conclusion

After analysing the data it can be shown that users, in Cardiff generally, are not familiar with a fitness application and tracking workouts specifically. A number of users are currently using an application to track but not their workouts. From analysing the data for users who don't use an app more than 60% said they would use an app. This shows that an application like this is needed but the correct designs and features have not been implemented accordingly.

NFC

Near field communication (NFC) is a method of wireless data being transferred that detects and then enables technology in close proximity to communicate without the need of an internet connection [6]. The number of mobile phones that are having NFC is only increasing. In 2015 there were 556 million phones with NFC-enabled. The estimated number for 2018 is currently 1.9 billion phones worldwide.

Justification of NFC

There are a number of reasons as to why NFC was used in the project. The first reason is for learning purposes. Developing a fully functioning app and interacting with the Internet of Things (IoT) and its potential is exciting to see and explore. As NFC implementation hasn't been taught before, it was a great opportunity to learn how it works and the benefits it could bring. After doing research on apps currently on the app store, there was none that utilised NFC. NFC has been around for a long period of time and is being used by a number of applications. Utilising the use of NFC into the app will give an opportunity to receive feedback from users on its benefits. In addition, with the statistics showed above, the number of mobile phones having NFC is increasing. Utilising NFC for small tasks makes it easy and convenient for users.

Benefit of NFC with Adding Exercises

There are a number of benefits to adding exercises with NFC tags. It is very convenient as it is instant. As soon as you have NFC enabled, all you need to do is a swipe and the job is done. It is easy to use and doesn't take much time to learn on how to do it. NFC is very versatile making it offer a number of services. One service that it could potentially offer with is having a counter built in. When users use the

NFC tag to add exercises the counter increases. This will show the gym how popular certain exercises are and thus make changes to the gym accordingly.

Constraints To The Approach

Mobile (Android) Development Knowledge

Java was the primary language that was taught in University and this is used by Android as the back end language. However, there are considerable differences including the GUI system being completely different than what is provided by the JDK. Android does not have a main function. They have onCreate, onResume, onPause, onDestroy functions that need to be overwritten. These differences will need to be learned to ensure best practices for Android development.

Sensitive Data

The application requires personal data to be taken from the user in order for it to be utilised properly. The user has the option to upload progress photo and their body weight which is personal to them and cannot be shared with other users. Security must be ensured to the highest point to ensure that only that user is able to see their own photo.

Database

There were a number of different database systems to use including MySQL, MongoDB, and SQLite. Firebase was a new database system that was acquired by Google in 2014 which held a number of benefits including the Firebase Realtime Database being one instance so building cross platform is easier and users constantly get updates with the most current data. Firebase doesn't require an API to be developed as it handles all the components that come along with creating a backend for applications. Firebase also comes with Authentication making it easy to integrate Facebook, Google and Email log in system. Firebase also provides Storage which gives secure document transfers and downloads for apps regardless of the network quality. With all these benefits Firebase was chosen, however, no prior knowledge was had in developing mobile apps with Firebase. A lot of courses had to be taken to learn Firebase and it’s best practices.

Exercises

A database consisting of exercises was required so users can add to their workouts. After doing thorough research into exercises online during my initial report, there was no database that was suitable to be used for my application. A bespoke collection of exercises would need to be created that caters to the needs of my app.

Existing Solutions

There are a number of third-party applications that allow users to track their workouts but immediately after reviewing them, it was clear that they were lacking a number of technologies that is available now.

The applications reviewed had a number of features but did face issues in either functionalities or design. The fitness apps reviewed had a range of downloads ranging from 10 million to 100,000. The reviews will be conducted from the most downloaded app to the least.

The first app reviewed was JEFIT Workout Tracker. This app was one of the best apps on the Play Store with a huge download range of 5 million to 10 million. They offered the same functionalities as the requirements that were set out and more however after reviewing the app: lacked in design. For an app with a large user base, the user experience was poor and very linear. The functionalities were great as you could track workouts, search exercises, built in timer, upload progress photos and see data visualisation. They also had a social element to the app where you could add friends comment, like and share statues and more. However, they did not allow users to add exercises with NFC and add friends to workout to track with.

The second app reviewed was Gym Workout Tracker and Trainer by Fitness22 with a range of 1 million to 5 million downloads. This app had the best design out of all the apps reviewed. Clean and simple navigation with clear typography and professional imagery. The app stands out from other apps because it has a number of pre-made workouts for users to follow instead of tracking personalised workouts, although that feature is still available. The app doesn't allow users to upload progress photos and doesn't utilise NFC for adding exercises or a social element apart from sharing workouts on social media.

The third app that was reviewed with a range of 100,000 to 500,000 downloads was 'WORKIT Gym Log Workout Tracker' which had a number of good functionalities which allows the user to log their workouts, corresponding exercises, and sets which also allowed users to upload progress photos. From a UX point of view, it did lack a number of design principles and felt all over the place. The functionalities available however were great and a lot of ideas were taken for the project development.

The fourth app that was reviewed was FitNotes - Gym Workout Log (the same downloads as the third app) which lacked UX and could be seen to mimic MyFitnessPal but in a poorer way. The app has the fundamental functionalities of tracking workouts and seeing progress via a graph. The design was the poorest out of all the applications reviewed.

The fifth and final app that was reviewed was the Simple Workout Log by Selah Soft which had again 100,000 - 500,000 downloads. The design of the app is again very poor and very similar to FitNotes. There were no extra functionalities like uploading weight or progress photos. It has the basic functionalities and did it well but was not designed well at all.

With all the apps reviewed, it was clear to see that near field communication was not implemented at all. In addition, sharing workouts with friends was also not present meaning the features of the app will have some unique features compared to its competitors.

Current Apps

There was a number of fitness apps that was online and published on the play store however they were not accessible to manipulate and amend. After doing further research online, there were a few GitHub Repositories of developers who made simple workout tracking application but they were too simple and not unique to the ideas that I had for the application. This lead me to develop the application from scratch.

Example GitHub Workout Trackers

1. https://github.com/sudar/Workout-Tracker
2. https://github.com/yeriomin/WorkoutLog
3. https://github.com/nolanlawson/AppTracker

Methods and Tools for Solution

Android

Before delving into the project, one of my first priorities was to learn Android Programming as well as find a good IDE to develop the project on. Android does provide an official IDE with has a number of extensive functionalities and support than any other Android IDE. The Android Studio IDE for Android development is a Java IDE like Eclipse [7] but provides a vast amount of extra functionality. It had the ability to automatically generate the necessary XML files, get access to inbuilt libraries to help with designing the User Interface, and maintenance for resources. Android lives on Gradle build system and this comes as a standard when using Android Studio. Researching online and using both Eclipse and Android Studio, the code completion was far better with Android Studio which was very useful, especially when programming for long hours.

A number of courses were taken on Udacity to equip me with the knowledge and familiarisation of Android Studio and development. Courses were taken specifically on Udacity because the teachers were Google Engineers for Android who knew

what they were talking about and could teach industry standard code and design principles.

Justification of Android

There are a number of reasons as to why the project was developed on Android. The first reason being the cost of upload. As I initially wanted the project to be user centric, the cost of uploading an app on iOS was $99 as opposed to Android fee of $25. In addition to this, updates with Android can take hours, however, with iOS could take weeks for approval. This would prevent any user feature implementation to be feasible. Additionally, the phone that is used by myself is a Samsung Galaxy Note 4. Although emulators could be used to test on iOS or Windows, having a physical phone meant, testing was easier and can see how the design flows in my hand. Furthermore, Windows uses .NET and iOS uses Swift as their back-end language. The preferred language for myself is Java and as my skill sets were more towards Java, it made more sense to develop on Java and increase my knowledge in that field.

Firebase

After doing research into a number of database systems that could be used for the workout tracker in my preliminary research, it was decided after discussing with my supervisor to use Firebase. As I had no previous knowledge with NoSQL, it was a great opportunity to learn how to utilise NoSQL database and see its potential.

Firebase is essentially a mobile and web application platform with tools and infrastructure design to help develop high-quality apps. It has a number of complementary features including for development: Realtime Database, Authentication, Cloud Messaging, Storage, Hosting, Test Lab, Crash Reporting. To grow the application they have features including Notifications, Remote Config, App Indexing and much more.

A number of courses were taken to learn Firebase but some of the issues that were faced were because Firebase was essentially a new database system: courses were limited and the community was small. Udemy is a global marketplace for learning and teaching course online. I was able to find courses on Udemy [8] that taught me the fundamentals of Firebase and adjustments were made accordingly to meet the needs of the project.

Exercises API

Finding an exercise database online was very difficult, near impossible. However, one developer had created an existing REST API [9]. The problem with the API was that the system that was developed was a full-scale application and didn't fit the needs of my application. It also used languages (jQuery and Django) which were languages I was unfamiliar with. Creating my own JSON exercises collection was the easiest option to move forward with. The data was stored within Firebase and displayed accordingly. The details of the setting table in the REST API was not

visible not giving me access to the information to aid with the structure of my exercise collection. To move forward, I looked online at what information was available online about exercise and created a JSON collection based of that. Looking at the REST API did give an idea of the objects to create but no information was taken in regards to data modelling as Firebase stores the data very differently. No online resources relatable to gym structure was found so articles had to be read on how to structure this data.

Library: Picasso [10]
When a user is uploading images Android does provide a View called ImageView. ImageView does, however, face a number of problems which Picasso fixes. Picasso is able to handle ImageView recycling and download cancellation as an adapter, compute complex image transformation with minimal memory use and has the ability to automatically memory and disk cache.

Library: EventBus [11]
EventBus is essentially a publish/subscribe event bus which is highly optimised for Android. This library was used as it simplifies the communication between components thus decoupling event senders and receivers, avoiding complex and error-prone dependencies and life cycle issues. It allowed the code to be simpler and is proven as it is used with over 100,000,000+ app installs.

Library: Butter Knife [12]
To get access to the Views in the Java classes, each of View IDs has to instantiate on the onCreate whilst casting the view. Butter Knife removes this by annotating fields with @BindView and a view ID for Butter Knife to find and automatically case the corresponding view in the layout. This helped with eliminating anonymous inner classes for listeners by annotating methods with @OnClick and others.

YouTube API
The YouTube Android API enabled me to incorporate video playback functionality into my Android application. The API allowed defining methods for loading and playing YouTube videos. The YouTube API is vast and has the ability to cue or load videos into a player view. This helped with giving details on how to perform each exercise. As many users don't like reading or don't have the ability to, the YouTube video will be the second option they can use.

Facebook SDK
Having multiple options to log in is becoming more popular. The Facebook SDK for Android enables users to sign into the app with their Facebook Login. When the user logs into the application with Facebook, the user grants permissions to the app so it can retrieve information or perform actions on Facebook in their half. Once the user has confirmed authorisation with Facebook they can automatically be redirected to the main menu of the app.

NFC Library from GitHub: android-nfc-lib

A library was found online which had pre-set methods for reading and writing to tags. This helped with adding exercises automatically on the swipe of an NFC tag.

Specification

App Specification

The completed project should be a fully functional Android application that provides the functionalities that have been set out initially. The application will essentially allow users to sign up and/or log in. After completing a successful sign-up or log in they have access to do the app to track workouts. The created workout will allow users to add exercises to it and their corresponding sets. The user is able to search through a list to find the relevant exercises. The app will also allow users to view a description of how to perform the exercise, add their body weight and upload progress photos from their gallery.

Essential Requirements

Below are listed the essential and desirable requirements for the project:

1. Log In/Sign Up using Email or Facebook
    1. Sign out the user
2. Create workouts and visualise the date, time and user created
    1. Delete workouts
    2. Change workout name
    3. Change list visualisation by either time, name or user
    4. Add users to make a shared workout so they can add their own exercises
3. Add exercises to the workout with the name of the added exercises by
    1. Delete exercises
    2. Add exercises using NFC
    3. Search for exercises using the search bar
4. Add sets to the corresponding exercises with the weight and reps
    1. Delete sets
    2. Change reps and/or weight for sets
5. Upload progress photos
    1. Delete progress photos
6. Display account details on the home page

Essential Requirements Justification

As the project is primarily to learn about Firebase and making a fully functioning application it is vital to implement the foundation. These include 1,2,3,4 and 5. These functionalities are key to the fitness users as they can essentially go to the gym and do the bare basic. The user will need to sign up and log in so all workouts, photos etc can be linked to that account.

The chosen sign up is currently Facebook and Email. This is because Facebook is currently used by large app companies such as Tinder and for future data can be

exported to Facebook quite easily. Email allows us for marketing to users and communicates with them easier.

The workouts will initially be added to the newest workout at the top, however, the user can change this to also either the name of the workout or user. It is essential for users to search through the exercises because many Android phones don't have NFC so the only way to find the right exercise is by searching.

The user will be able to add their body weight in KG and the date they added it. This will then be visible on a graph to see the trend of when their losing weight or gaining and make the relevant adjustments.

Uploading progress photos is vital for any fitness enthusiast as they can see the progress of their body. The user will be able to add photos from their gallery, review it and upload. This will make sure they have one standalone app that has all the photos which they can look back on in the future.

The more complex functionalities are essential to the system as they are unique to the application. Currently, no fitness app allows users to track exercises using NFC. This is an easier way to track exercises as they can just swipe the NFC when on the exercising as opposed to searching through it.

Another complex functionality to implement is the shared workout which is essential as no fitness application has this feature. This is also using Firebase's Real Time Database as users can add exercises together real time.

Desirable Requirements

If all the essential requirements are fulfilled in a good manner the following will be implemented:

1. Create training plans for workouts
2. Scheduling workouts
    1. Link to Calendar e.g. Google Calendar
3. Countdown timer system
4. Connect with third party apps e.g. MyFitnessPal & FitBit
5. Add body measurements
6. Calculate calorie intake with information such as body weight, height, activity level and thus provide Macronutrients

Desirable Requirements Justification

There are a number of desirable requirements that could be added but the above listed are what I feel to be most important. The first requirement is a huge benefit for regular gym goers who taking tracking a lot more seriously as they pre-plan their workouts and schedule it for a set period of time. This will allow the workout tracking to be a lot more in depth. The second functionality allows users to schedule workouts so they know what days they are training and a set time for it. This could also be linked to Google Calendar so they have their diary listed for a day or week. The third functionality is essential for gym-goers who track their weight. Instead of having a separate app to start the timer, a built-in system would save time and effort for the user.

It is more than likely that the user is using more than one application for the fitness lifestyle. Connecting to the likes of MyFitnessPal and FitBit can mean they have one app where they can review the progress. The app could essentially manipulate the data and thus give the user informative decisions to make. Some users measure their body and this could be a useful addition to track and see the trends.

The last desirable functional is something that is very useful for users. By taking in the details from the users and giving them macronutrients will allow them to log their food on MyFitnessPal more accurately and plan their workouts better.

User Interface Specification

There will be a strong emphasis on the design aspect of the application to ensure it is clean and simple to use. The users who will be using the application are most likely not going to be technology experts so simple navigation and words will be vital to use. As the project is being developed for Android, Material Design [13] will be learned and used.

UI Requirements

1. Navigation page to have access to the whole app
2. Follow Material Design
3. Consist colour scheme throughout the app
4. Simple text/commonly used icons for changes or navigation
5. Give feedback to users via ‘Toasts’ when changes have been made
6. Highlight relevant areas in ‘red’ for errors
7. Set title for each page with information relevant to the activity
8. Consistent typography throughout the app with important information set to bold

UI Justification

The ideal way to navigate in Android is to use a Drawer [14], however, this means using fragments instead of activities. With the limited time that is available and the better understanding of activities, it was decided to go ahead with a single page after sign in to navigate around the page. This is still an easy way to navigate around the app as there aren't many tiers to the app.

Material Design is a design language that was developed by Google in 2014. It allows for unified experience across platforms and device sizes. It is a comprehensive guide for visual, motion and interaction design across platforms and devices. The Material Design holds a number of benefits including the design being grounded in reality and is actually inspired by design with paper and ink. A number of aspects are covered including typography, grids, space, scale, colour and imagery. It also allows the design to be adaptive so it works for small screens as well as large screens including TVs.

Having a consistent colour scheme throughout the application means the user won't be confused and when they do see new colours e.g. red for errors, they will understand that something has happened. It is important for the 'common language' to be used or relevant icons that the user understands and this will be done with the relevant research to find the best representation of a functionality.

Toasts are using by all major apps including Facebook, Instagram, and Snapchat when a user is making a change or update. The app with make use of this when adding, updating or deleting. It is important for users they know what page they are on so relevant titles will be set to each of the activities to give a bit of information about that page.

Typography is a key element in design as it can represent the brand and have adverse effects on the eye. Ensuring a good typography with the right sizes and spacing is key so it works for all ages, in low light conditions in the gym and on smaller phones.

Design

Overview

To keep the design clean, simple & modern - consistency was key. The app follows a trend throughout so that the user doesn't get confused and understands how the application works. The elements of Material Design used are as follows:

CardView

A card is a sheet of material that serves as an entry point to more information. They may contain a photo, text or other valuable information. Throughout the application, CardView is used to help with navigation and displaying information to the user such as workouts and progress photos.

Floating Action Button

A floating action button represents the primary action in the app and is used for a promoted action. FABs are used throughout the application when a user needs to add or do an action. This makes it easier for the user to understand what to do when they go on to a specific page after doing it once. The colour and size of the FAB are kept consistent throughout the application.

Dialogs

Dialogs are used to inform users about specific tasks and may contain crucial information, require decisions, or involve multiple tasks. Dialogs have been used throughout the app the aid with adding, deleting and updating. This is a simple way of making changes without having to navigate away from the page.

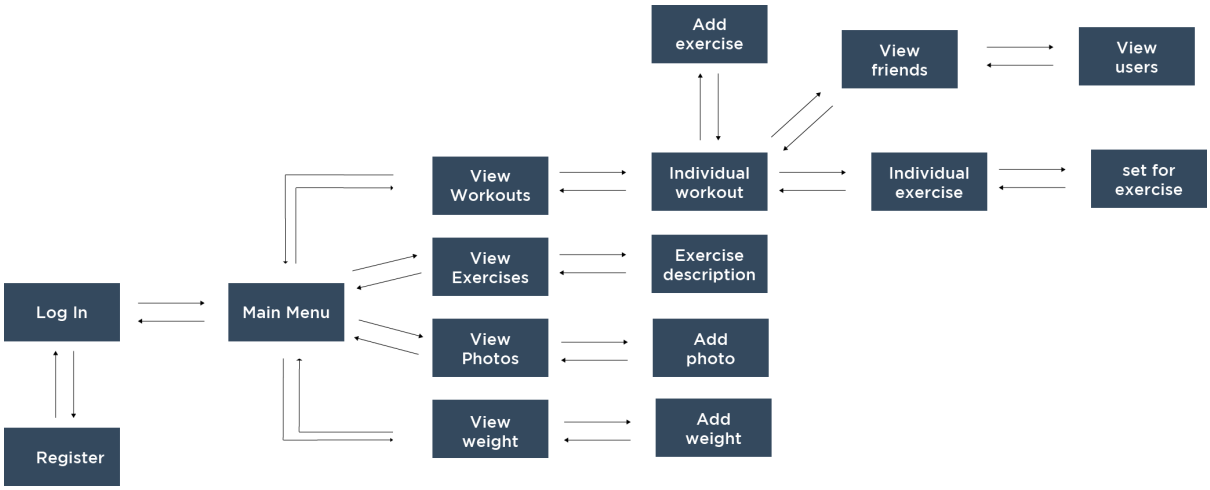
Toolbar

A toolbar is where actions can take place above a view. All the toolbars within the application have a set title to give the user information about that page. Some pages will have a back button to allow the user to navigate back to the original page. Other pages include icons to help with saving or sharing. Having toolbars aids with navigation and information.

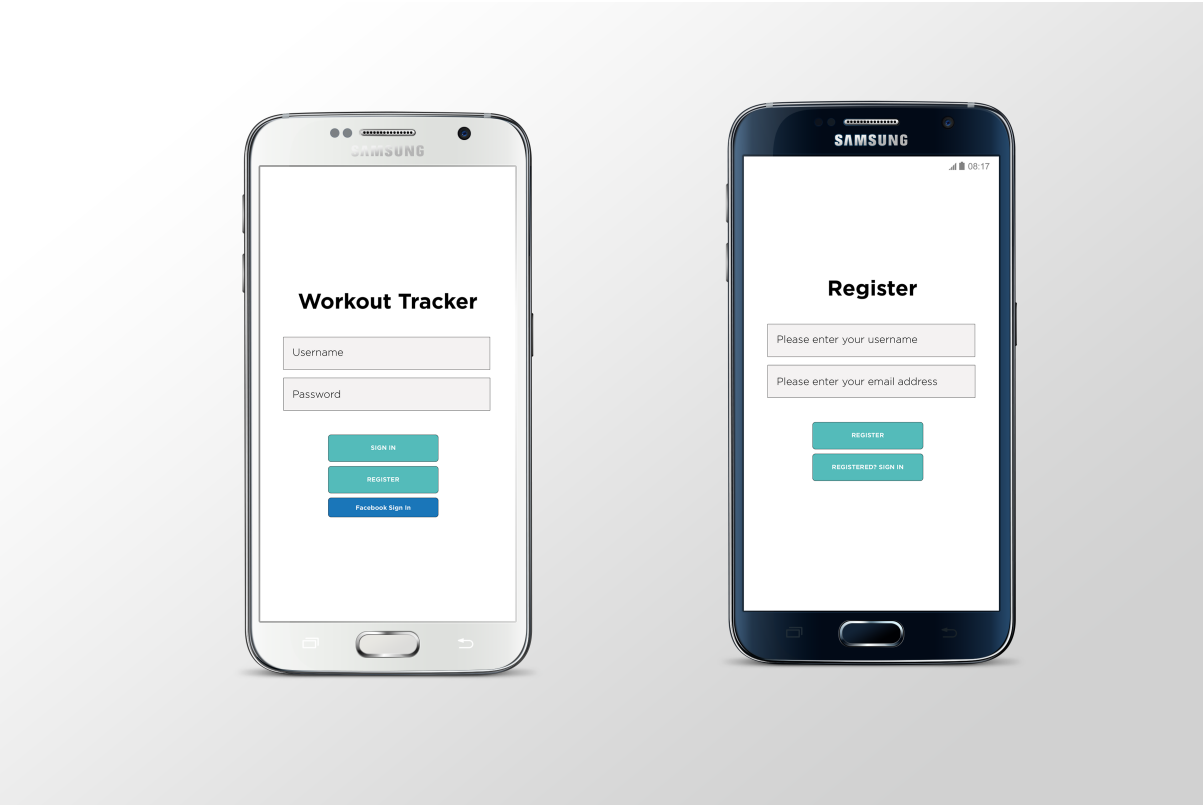
Storyboard

Storyboards have been designed to show the route that the user goes on in the app. The first storyboard created shows the individual pages that the users can navigate to and from. It gave an idea of how many activities were needed and which activity will display the data. Dialogs will be used to add information and to confirm details such as deletion but the main activities are viewed. A second storyboard has been created below to describe the flow after creating the mockups.

App Story Board



Mock Ups



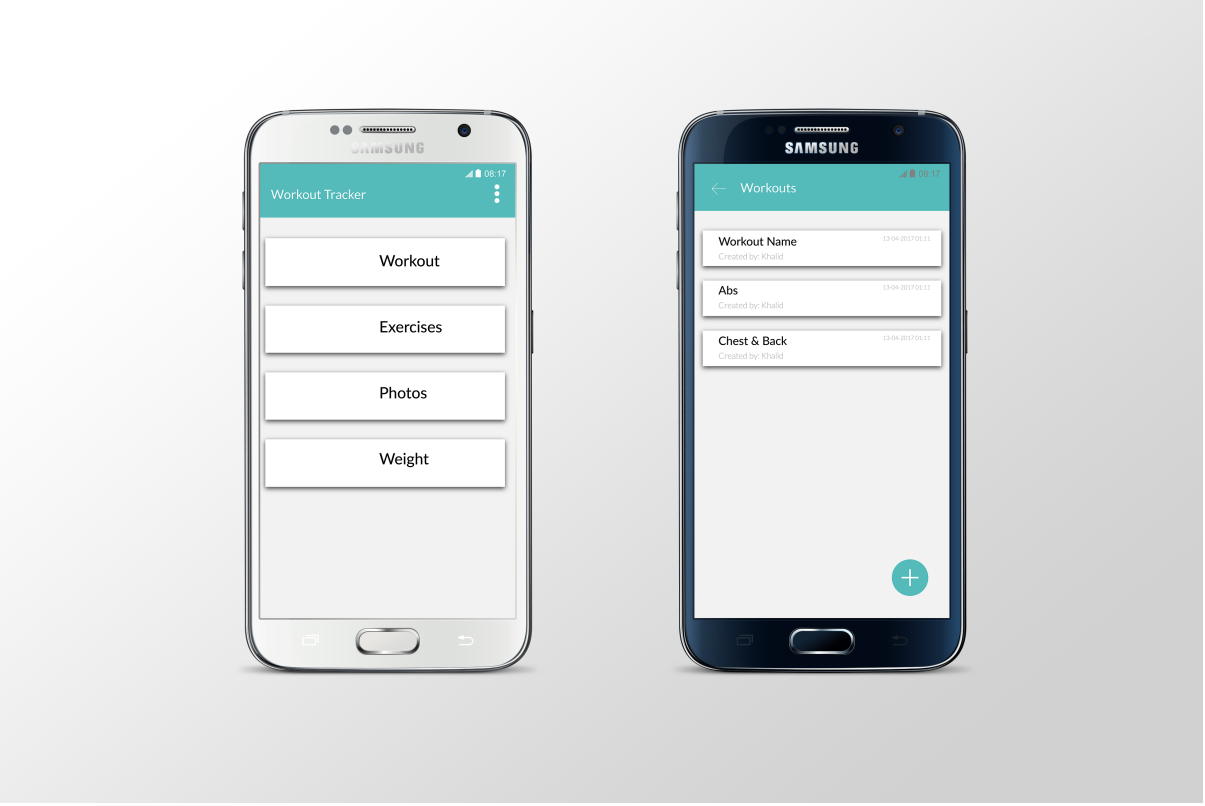
Log In/Sign Up

- Sign In (Left)

When a user downloads the app they will initially see the sign in page. The sign in page displays the name of the application with two text boxes where the user can enter their username and password. Below the two text boxes are three buttons. The first button is for sign in after entering the username and password. If they have no account they can register by clicking the register button. If they don't want to use their email and password then they can sign in using Facebook. They will be redirected to the Facebook app where they will need to verify sharing some details and authenticating.

- Register (Right)

The register is a very similar look to the sign in page. The only difference is having removed the Facebook login button. Within the text boxes, there are some hints, however, to guide them on what information is needed for each of the text boxes. They can register by clicking the register button or click the second button to navigate back to the sign in page where they can sign in.



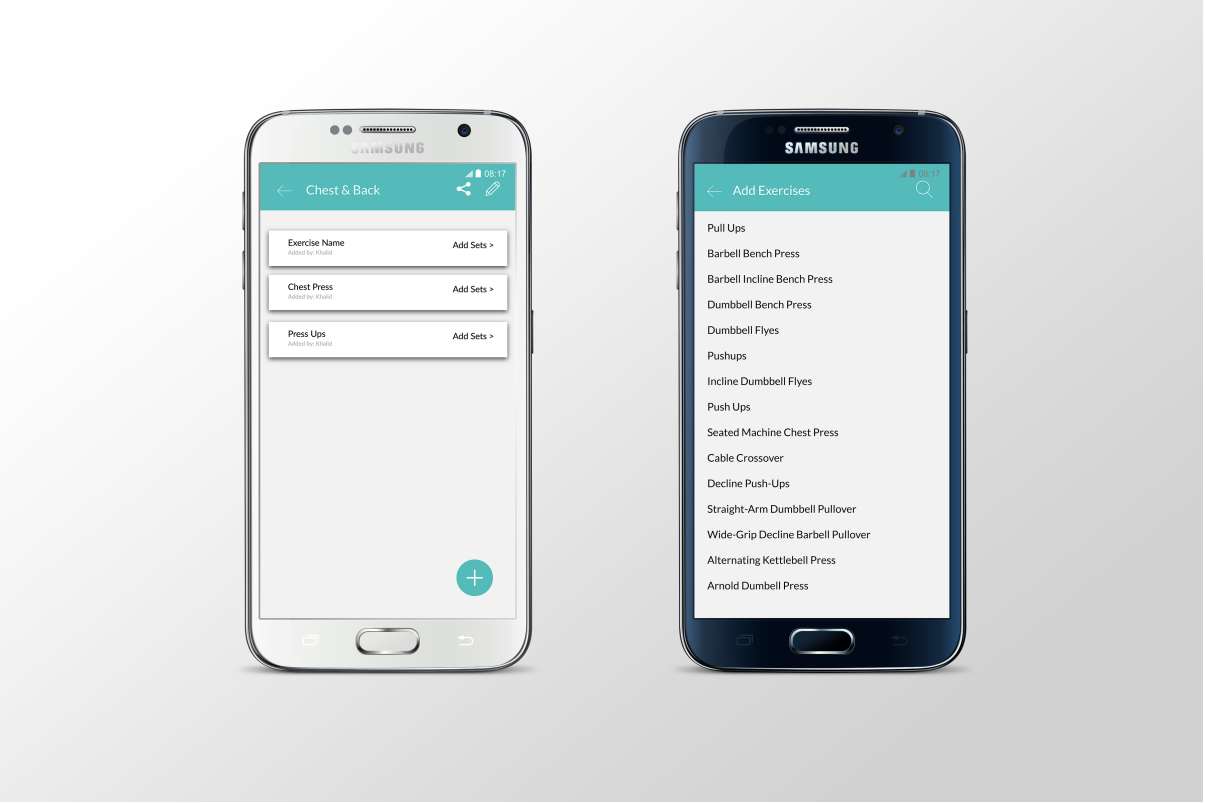
Navigation/Workout Tracker

- Main Menu (Left)

The design of the main menu is kept to a minimal so that the users are able to navigate around the pages as easily as possible. The options menu on the top right displays an overflow menu that which when clicked will display a dropdown so that the user can sign out. The four tabs are kept simple, with large enough text to give the user how to navigate. Icons will be displayed next to each of the text. The user will be able to click on either the text or box to navigate to the relevant page. What isn't displayed on the design is that the toolbar will also be a collapsing toolbar. This is where when the user pulls down the toolbar, it will cover approximately 1/3rd of the screen giving the user the details of their name and email. They can also scroll up to get to the original view which is displayed above. Having a clear and minimal design here was key so that user is able to navigate around effectively.

- View Workouts (Right)

The workout page is important to display the correct amount of information. Having the workout name in bold was essential as it is the first thing that the user will see when they click on that page. The most recent workout will be displayed at the top but in the toolbar, there will be an icon which the user can click to change the listing of the workouts in either time, workout name or email. The other information displayed are the date and time of the workout so users know when they added the workout. Below the workout name is the name of the user who created it. This is to help when sharing workout as the user will know who created the workout initially. A floating action button (FAB) is located on the bottom right, which follows Google Material Design, to create a new workout. Once clicked a dialog will show up where the user can enter the workout name in an EditText. If the user wants to delete the workout they can hold down on the workout and a dialog will pop up for confirmation. To navigate to add exercises the user can click on the corresponding workout box.



- Individual Workout (Left)

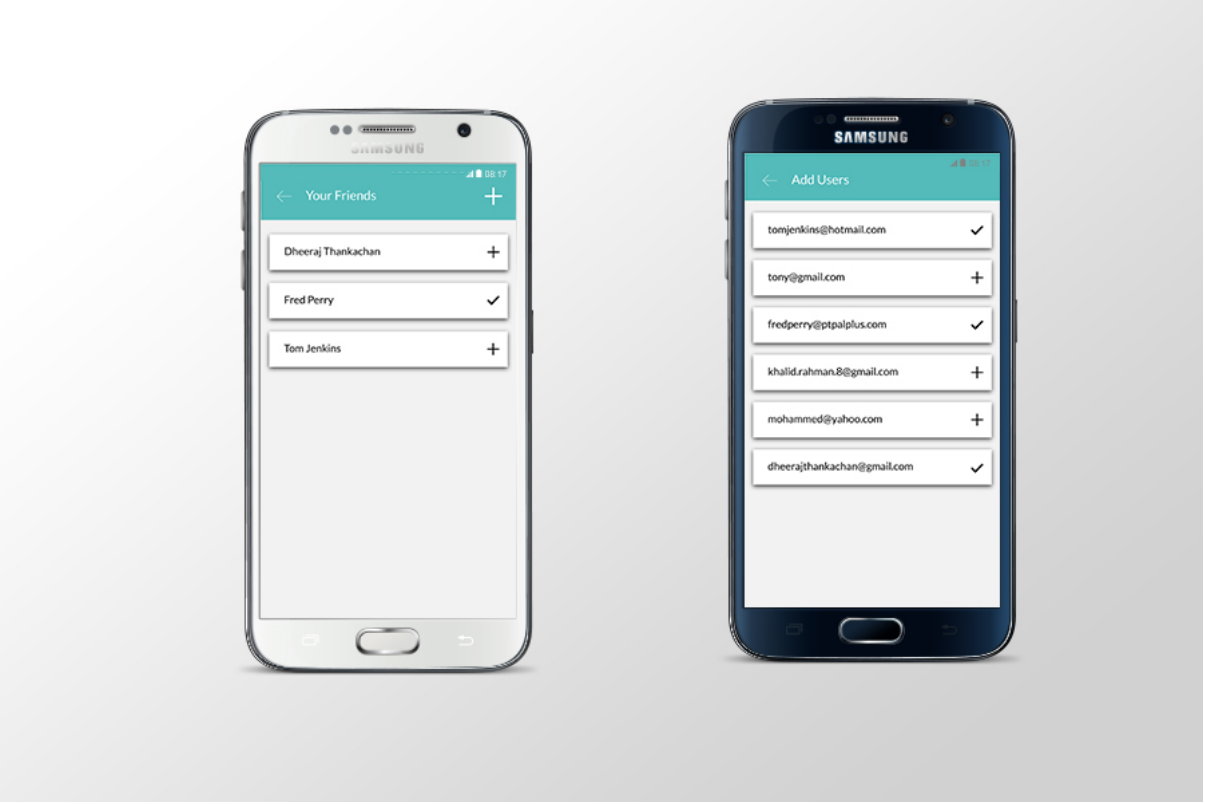
This page is where the user can add the corresponding exercises for that specific workout. The toolbar displays the name of the workout that they clicked on. It also contains two icons. The first icon is a pencil which is to edit the name of the workout. It will display a dialog in which will populate the current name to which they can change. The second icon is the share icon. The user will navigate to another page where they can see their friends and add users which will be explained below. CardViews are used to display each of the exercises. Only the important information is displayed to keep it clean and minimal. Following the same design principles, the name of the exercise is set to bold and the user who added it is located below. There is a button the right of each box to which the user can click on to add the corresponding sets. A FAB is located on the bottom right so the user can click on to add exercises. They will be navigated to the 'Add Exercises' page. To delete an exercise, a user is able to hold on the box and a dialog will pop up to confirm whether they want to delete the exercise.

- Add Exercise (Right)

This is a very simple page where the user can essentially click on the exercises that they would like to add. It displays all the exercises in list format to make it easier to read. The toolbar contains a search icon which changes the toolbar to a search bar once it has been clicked. This will allow the user to search for specific exercises once they click on enter or on submit. Once the user clicks on the exercise they want to add a Toast will be displayed to let the user know that an exercise has been added. This gives the user feedback once they have done a task. Padding is used around each of the text so users aren't trying to click on the specific text with difficulty when trying to select it.

- Add Set

When a user wants to add a set and they click on the exercise they will be redirected to the 'Add Set' page. To add a set a user will click on the FAB location on the bottom right which will open a dialog. Within the dialog, they can enter their rep and sets and save. This will populate the view. They can add as many sets as they like and if they make a mistake they can edit it by clicking on the pencil icon. If a user wants to delete a set they can hold on the box and another dialog will be displayed to confirm whether they would still like to delete the set.

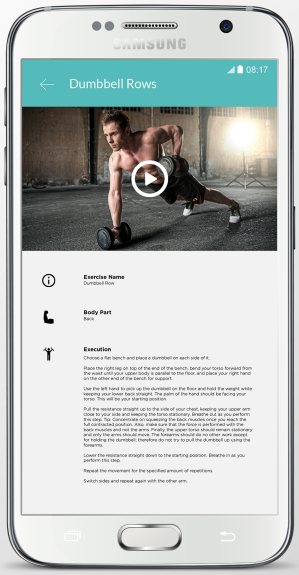


- Your Friends (Left)

When a user wants to share a workout with one of their friends and track the workout with them they will click on the share icon in the individual workout page. Once clicked they will be able to see a list of their friends. Their friend's name will be displayed with a plus or check icon next to it. The check icon means that the user has the ability to view that workout on their phone. It will dynamically change from the plus icon to the check when a user wants to add them to the workout. The toolbar contains a plus icon where the user can add further friends. If a user wants to remove a user from a workout, they can simply click on the check icon to which it will change dynamically back to plus icon.

- View Users (Right)

If a user wants to add a new friend to share a workout with they can click on the plus button on the 'Your Friends' page. The page displays a list of all the users who have signed up for the application. The design is the same as the 'Your Friends' page - they can click on the icon to either add or remove a friend. The page, however, does change the text from the name to the email address as there could be two users who have the same name and they won't be able to differentiate them.



Exercises

It was important to display a simple and clean page on how to do perform specific exercises. When a user clicks on an exercise, they will be able to see a YouTube video on how to perform the exercise. The page will initially have a thumbnail of the video and once a user has clicked on the thumbnail, a popup with the video will be played. The database has a collection of all the exercises and these details will be displayed back to the user. The information is broken down into three sections: name, body part, and execution. Each area has its own icon to represent the data and will be divided by a divider. The title of each section is set to bold with enough spacing between each section to make it easier to read. The view is set to scroll view so that if there is more information then the user is able to scroll down to continue reading.

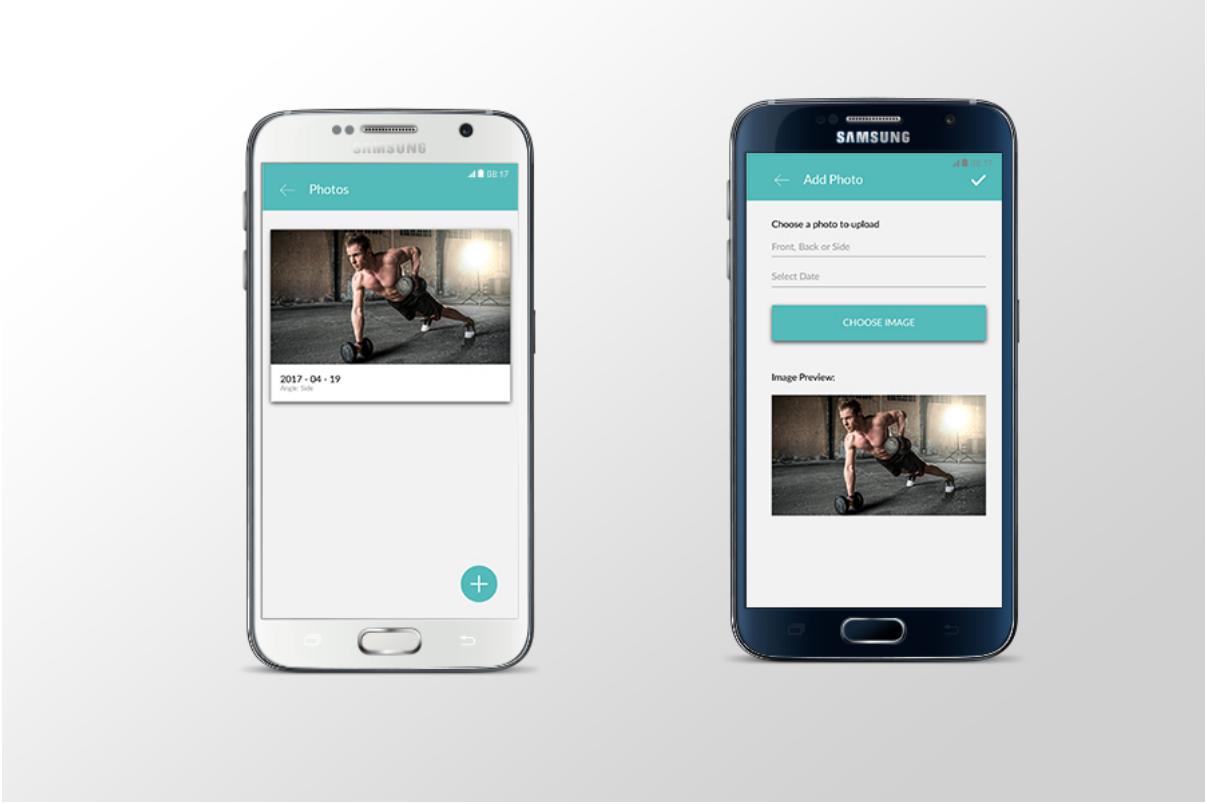
Body Weight

- Body Weight Progress (Left)

When a user wants to track their body weight, on the main menu they can click on 'Weight' and it will display the body weight page. This page displays their body weight in boxes with the date they tracked that specific weight. A line graph will be displayed to show their progress on whether they are losing or gaining weight depending on their goals. They can delete a body weight by holding on the box to open the confirmation dialog. A user can add a weight by clicking on the FAB.

- Add Body Weight (Right)

Once a user have tracked their weight physically, they can add it to the app quickly simply by entering their weight which will automatically be set to KG and by selecting the date they've taken it. When a user clicks on the date a DateDialogPicker will be shown so it is easier to enter a date. They can save the body weight by clicking on the check icon in the toolbar. They will be then redirected to the 'Body Weight' page where they can see their newly added weight and how it's changed their progression. The weight is configured so that it changes the input keyboard to set numbers only. This prevents the user from entering text or unique characters.



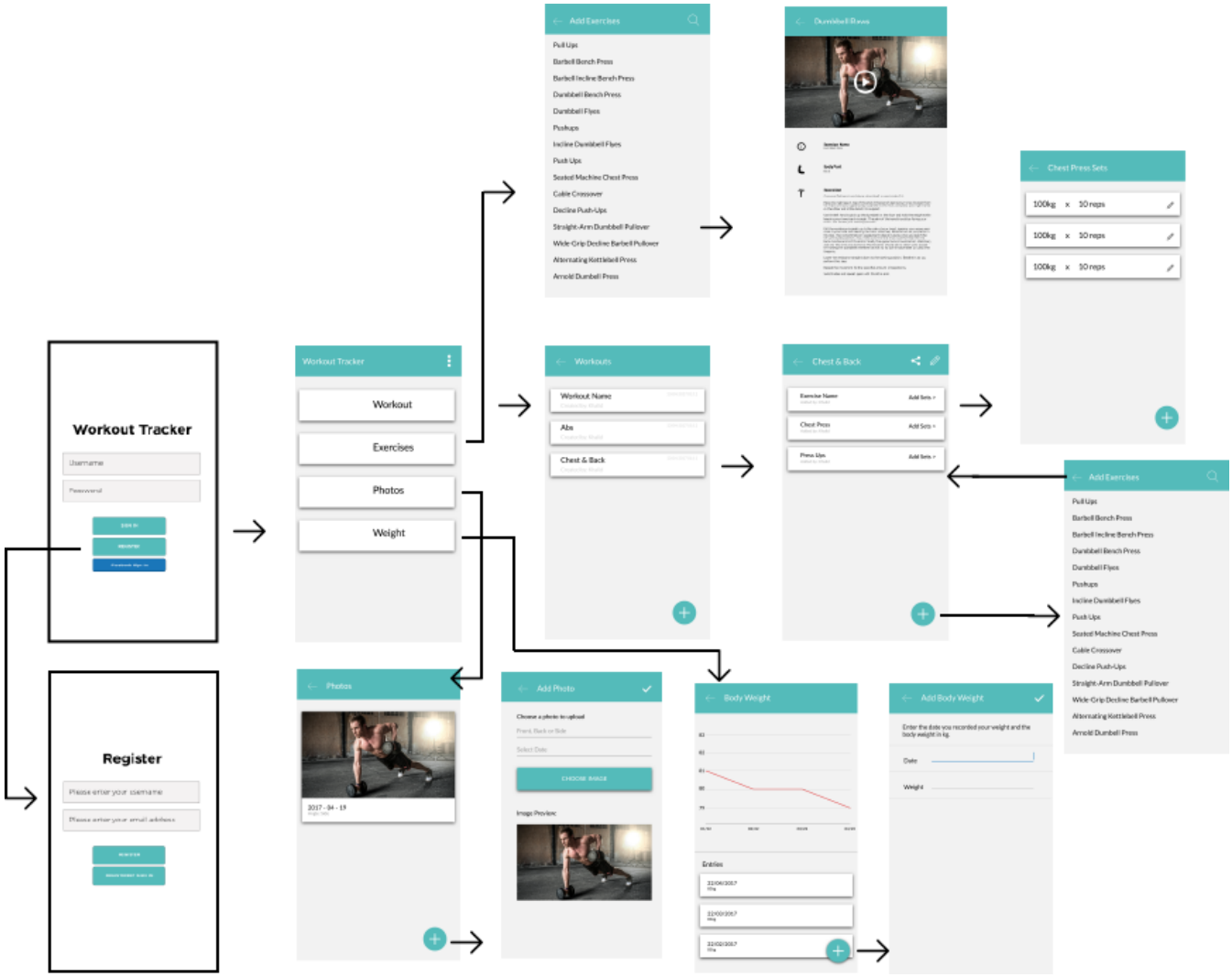
Progress Photos

- Progress Photo (Left)
If a user wants to upload photos, they can navigate to the photos page through the main menu. The page displays the uploaded image in a rectangle box with the date they uploaded it and the side of the photo. The date is important to see how they looked on a particular date thus highlighted boldly. Assigning a side to the photo means they can filter through the front, back or side. A FAB is located on the bottom right so they can add a new photo.

- Upload Progress Photos (Right)
The add photo page has two EditTexts displayed after a TextView on what to do. The first EditText is where the user inserts the side of the photo it is. They will then select a date from the EditText which will open a DatePickerDialog. A button is used to choose a photo which will open their gallery. Once they've selected an image, it will be previewed in the previewer section below the button. If the user is happy with the photo, they can confirm the photo by clicking the check icon in the toolbar. The user will be redirected to the photos page where they will see the newly added photo.

App Design Story Board

The updated storyboard shows the mock-up created and how the navigation between the activities will happen. Each of the views on the activities has a purpose and help with navigation between the activities. Have a visual storyboard help with distinguishing what was important and how the app will function.



Application Use Case

Use cases were designed to help with understanding the components of the application. More specifically understanding how the flow of the application will be for users and what details are important. Meeting the requirements set out initially is important and creating the use cases help with identifying necessary requirements to meet them.

Use Case: 1	**Register (Email & Password)**
Description	How the user registers for the Android app
Actors	Gym goers, fitness enthusiastic, app users
Triggers	User downloads and selects the app
Pre-conditions	The device has the app installed The device has an active internet connection
Basic Flow	- User selects the app on their device - Log in page opens - User clicks on register - User enters username & email and submits details - Confirmation email sent to users email to add password - User logs in with email and set password
Exception Flow	If the app does not have an internet connection, it will not be able to send the details to the database to register. No email will then be sent to the users email address.
Post-conditions	User has access to the app and main menu is visible

Use Case: 2	**Tracking a new workout**
Description	When the user is working out, they will create a new workout and track the exercises they completed and its corresponding sets
Actors	Gym goers, fitness enthusiastic, app users
Triggers	User clicks on workout button on main menu
Pre-conditions	User has signed in
Basic Flow	- User clicks on ‘add new workout’ FAB - User enters workout name and confirms - Click on the newly created workout name - User clicks on ‘add exercise’ FAB and selects exercises - User clicks on individual exercise to go to sets page - User enters weights and reps completed for each set
Exception Flow	If there is no active internet connection, the exercises won’t be visible
Post-conditions	Completed workout with exercises and their corresponding sets

Use Case: 3	Add friend to friends list and share workout
Description	Once a user create a workout they can share the workout with their friends to either view or track their own exercises
Actors	Gym goers, fitness enthusiastic and their friends
Triggers	User clicks on the created workout they want to share
Pre-conditions	User has signed in and created a workout
Basic Flow	- User navigates to the corresponding workout they want to share - User clicks on the share button - User can view existing friends - User clicks on add icon: navigate to view all users - User adds their relevant friend and navigates back - User clicks add icon to share workout with corresponding friend
Exception Flow	Users friend must have downloaded the app and signed up
Post-conditions	Workout is visible on friends list and has access to add view and add exercises

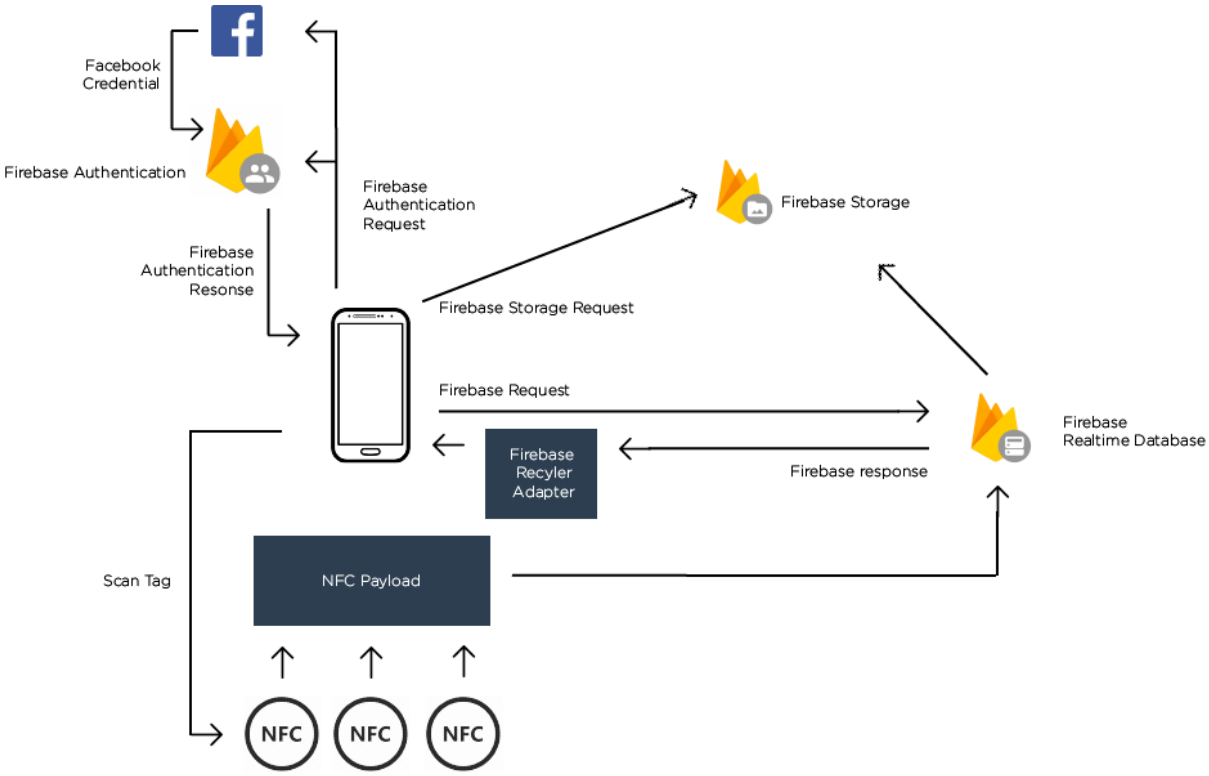
Use Case: 4	Add body weight
Description	User weighs themselves and tracks that weight in app
Actors	Gym goers, fitness enthusiastic, app users
Triggers	User clicks on ‘weight’ button from the main menu
Pre-conditions	User has signed in
Basic Flow	- User clicks on FAB to open add body weight dialog - Enters the body weight in kg and selects date - Clicks on ‘check’ icon in the toolbar to save
Exception Flow	User knows their body weight and have an active internet connection
Post-conditions	Body weight is viewable in the weight section

Use Case: 5	Add progress photo
Description	A user can select a photo from their gallery to upload
Actors	Gym goers, fitness enthusiastic, app users
Triggers	User clicks on photos from the main menu
Pre-conditions	User has signed in
Basic Flow	- User clicks on FAB to open add photos dialog - User enters the side of photo taken, the date and clicks on select photo button - User selects which app to select photo from - User selects the photo and preview is shown in app - User clicks on check in toolbar to save
Exception Flow	User has a the photo they want in their chosen app or gallery
Post-conditions	Photo is viewable in the photo activity with date uploaded and side of photo

Architecture and System Design

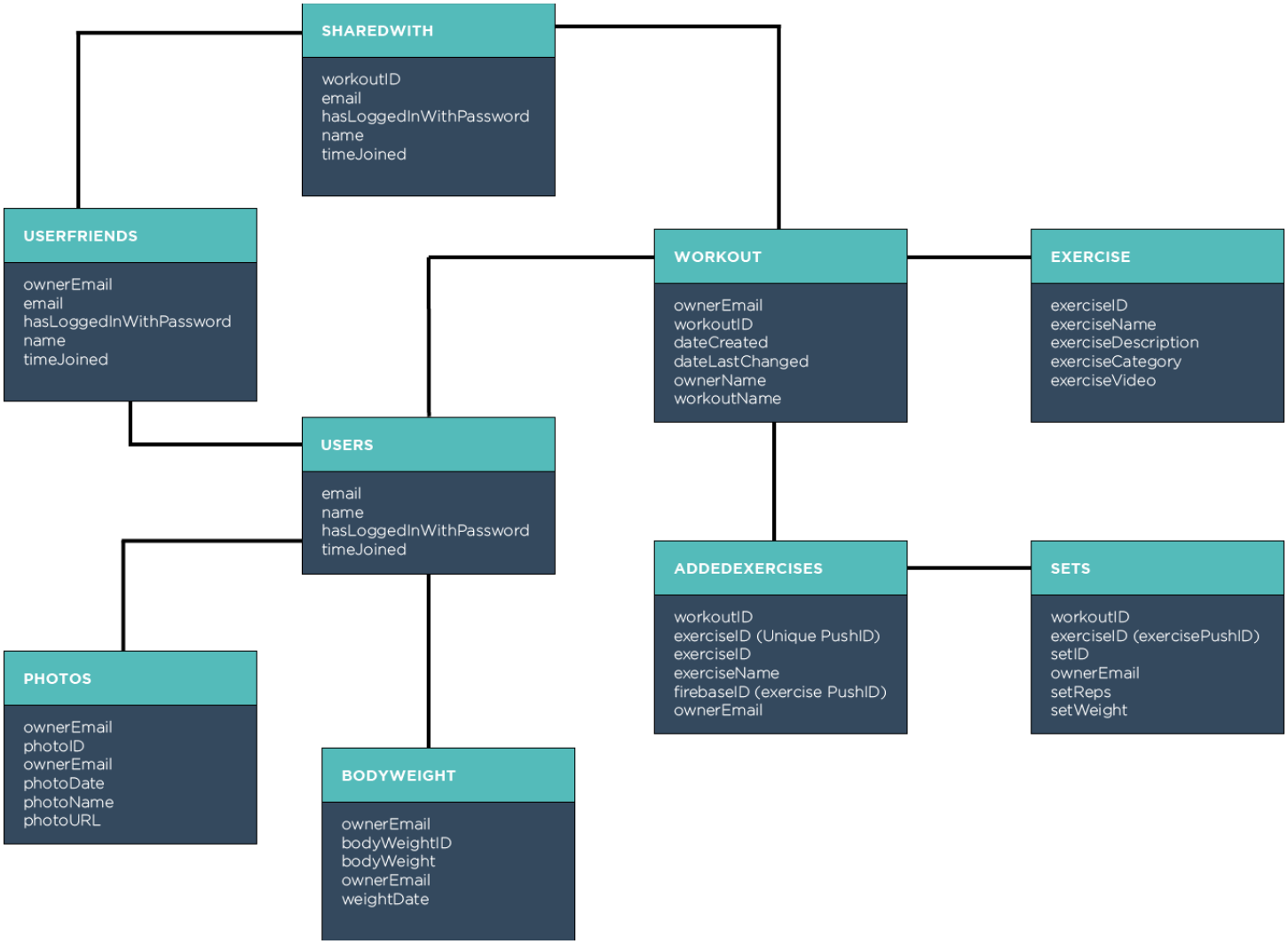
System Architecture

The system makes use of Firebase with Authentication, Storage, and the Realtime Database. The NFC has the exerciseID stored which corresponds to the ID in the database. More details about Firebase and its services are detailed below.



UML Diagram

The class diagram visualises how the different parts of the application work together. As displayed, the application is purely based on the user who registers. The UML diagram helped with the implementation of the app to a high quality ensuring that each feature corresponded to the correct aspect to the UML diagram.



Data Model

User

Once a user signs up in the application the User POJO will assign the necessary details. The email of the user is taken so that all details can be linked back to that user. The password is not required to be stored as Firebase Authorisation handles all the security once they've signed up. The name is taken from the username section in the register page. Once a user signs up it will time stamp the date they joined so it is known how long they've been using the app. The hasLoggedInWithPassword is to give details on whether the user signed in with Facebook or not. The other details such as height and gender will be taken later once they've signed up to give the user a unique profile.

User
- String email - String name - HashMap<String, Object> timeJoined - boolean hasLoggedInWithPassword - char gender - int height - int weight - int goalWeight
+ User() + User(String email, String name, HashMap<String, Object> dateJoined, boolean hasLoggedInWithPassword) + String getEmail() + String getName() + HashMap<String, Object> getTimeJoined() + boolean isHasLoggedInWithPassword()

Workout

The workout object stores a unique ID to represent that workout once the user creates it. The name of the workout is also taken and stored so it can be displayed back to the user. The ownerEmail is used to link the workout with a specific user. The ownerName is taken to help when sharing workouts. Having the name of the original owner means it will be displayed when sharing the workout with friends. It is important to know when the workout was created so they can log it as well as when it was last changed. This gives the user an idea of when they last updated it.

Workout
- String id - String workoutName - String ownerEmail - String ownerName - HashMap<String, Object> dateCreated - HashMap<String, Object> dateLastChanged
+ Workout() + Workout(String id, String workoutName, String ownerEmail, String ownerName, HashMap<String, Object> dateCreated, HashMap<String, Object> dateCreated) + String getID() + String getWorkoutName() + String getOwnerEmail() + HashMap<String, Object> getDateCreated() + HashMap<String, Object> getDateLastChanged()

Exercise

Once a user has created a workout, they will need to add exercises to that workout. The exercises will be identified by its ID. Information such as name, description, and category will also be stored to help with filtering in the future. The exerciseVideo is where the end URL of its corresponding YouTube video is located to link with the YouTube API. When a user adds an exercise to a workout, a unique ID is generated and the second exercise constructor details are stored. The firebaseID stores the ID of that unique identifier. This was used to assist with duplicate exercises being added. By using the exerciseID, two of the same exercises won't be added. The first constructor is used for when a user wants to know how to do the exercise.

Exercise
- String exerciseID - String exerciseName - String exerciseDescription - String exerciseCategory - String exerciseVideo - String ownerEmail - String firebaseID
+ Exercise() + Exercise(String exerciseID, String exerciseName, String exerciseDescription, String exerciseCategory, String exerciseVideo, String ownerEmail, String firebaseID) + Exercise(String exerciseID, String exerciseName, String ownerEmail, String firebaseID) + String getExerciseID() + String getExerciseName() + String getExerciseDescription() + String getExerciseCategory() + String getExerciseVideo() + String getOwnerEmail() + String getFirebaseID()

ExerciseSet

Once a user adds an exercise they will need to add sets. When a user adds a new set a new ID will be generated. The set reps and weight are used to store the rep and weight of the set. The owner email is stored to link it back to that user. The ID is a reference to the workoutID.

ExerciseSet
- String setID - String setWeight - int setReps - String ownerEmail - String id
+ ExerciseSet() + ExerciseSet(String setD, Float setWeight, int setReps, String ownerEmail, String workoutID) + String getSetID() + String getSetWeight() + int getSetReps() + String getOwnerEmail() + String getId()

Users

The user's object is an object to store all the users. The user can select their friends to add as their friend and share their workouts with.

Users
- HashMap<String, User> userFriends
+ Users() + Users(HashMap<String, User> userFriends) + HashMap<String, User> getUserFriends()

UsersSharedWith

The usersSharedWith object will store all the users that the user shared their workouts with.

UsersSharedWith
- HashMap<String, User> sharedWith
+ UsersSharedWith() + UsersSharedWith(HashMap<String, User> sharedWith) + HashMap<String, User> getSharedWith()

Body Weight

A user is able to add their body weight. Once a new body weight is added a unique ID will be generated. The date they want to track for will be stored with the actual body weight. The ownerEmail is used to link the body weight back to that user.

BodyWeight
- String weightDate - String bodyWeight - String ownerEmail - String bodyWeightID
+ BodyWeight() + BodyWeight(String weightDate, Float bodyWeight, String ownerEmail, String bodyWeightID) + String getWeightDate() + Float getBodyWeight() + String getOwnerEmail() + String getBodyWeightID()

Photo

When a user created a new photo a unique ID will be generated. The photoName is to store the side of the photo that they took of. The date is also chosen by the user and displayed back to them for when they took the photo. The actual photo is stored in Firebase Storage and a link to that storage is stored in photoURL. The owner email is used to check which user added the photo.

Photo
- String photoID - String photoName - String photoURL - String photoDate - String ownerEmail
+ Photo() + Photo(String photoID, String photoName, String photoURL, String photoDate, String ownerEmail) + String getPhotoID() + String getPhotoURL() + String getPhotoDate() + String getOwnerEmail() + String getPhotoName()

Data Modelling Issues

One major difficulty that was faced was how to model the data in Firebase. Firebase provides very little guidance on how to structure the unstructured JSON data. The data model was pretty much up to me. It is known that most JSON data structures are completely denormalised. This means that references aren't typically used with JSON. With Firebase it is very easy to follow this through as you can have up to 32 child nodes.

This, however, is different with Firebase as it is best to keep the data structure shallow [15] which means having to normalise the data to achieve this. After researching online on how to structure the data, it was found that having different collections with a reference to link them was better as it is more efficient with looping and not pulling down excess data.

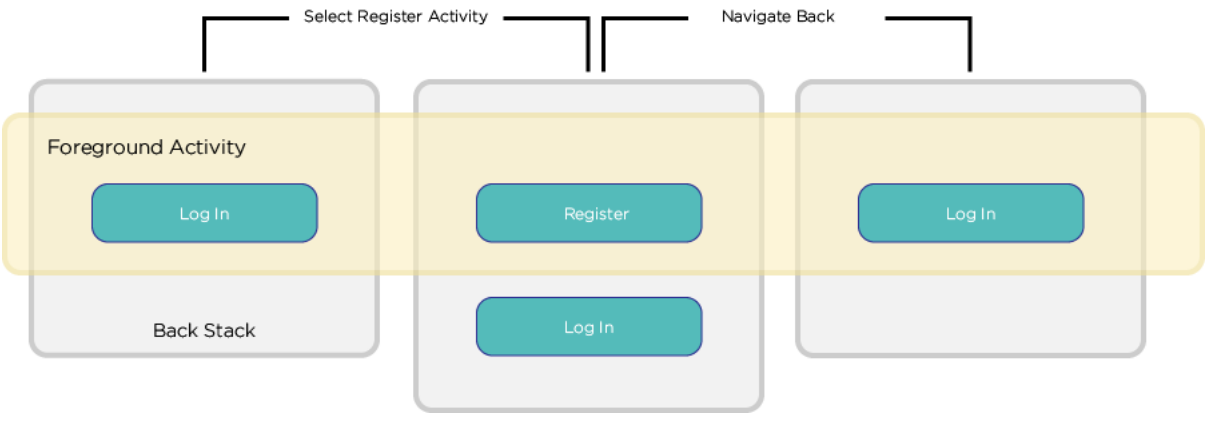
If the data is too deeply nest Firebase will face difficulty is querying the data as Firebase can only query on one child node at a time, and it cannot be a "grandchild" node: it must be a direct child of the list's top level.

Back-Stack Model

The diagram displays how the each of the activities interacts when a user is navigating through the activities. When a user goes back the original page, the foreground activity is destroyed and the previous activity that was called continues.

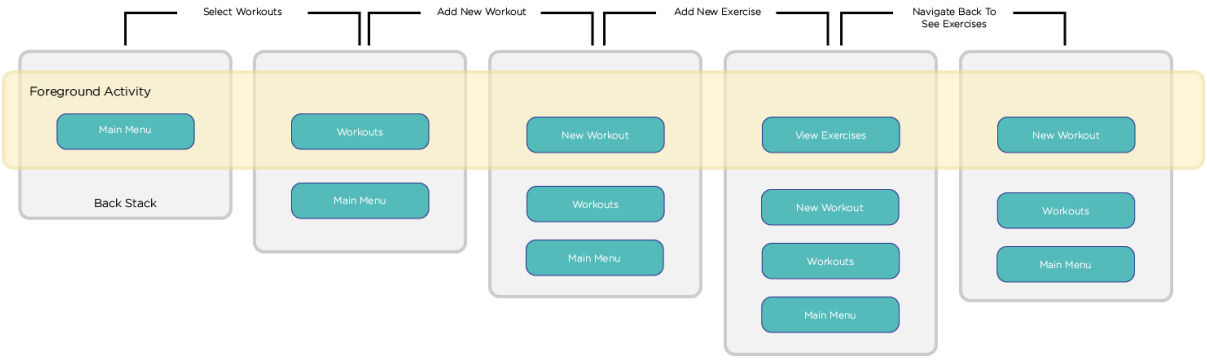
Log In

When a user opens the app for the first time the login page is the first activity that is visible to them. They will need to register and thus leaving navigating to the register activity. The log in activity is still on the back stack and once the user goes back to the login page, the register page is destroyed and is reverted to the previous activity.



Workout

The workout back stack model shown shows part of how it works when a user adds a new workout. When a user adds a new exercise, it is destroyed from the back stack but the other activities are continued to be placed in the back stack.



Firebase

Firebase is a mobile and web application development platform. It is made up of complementary features that can be mixed and matched to fit the requirements. It is a two tier architecture with an API which allows it to store and sync data across multiple clients.

Firebase Services

Authentication

The application required the identity of the user to be known. Knowing the user helps with securely saving the users data in the cloud and provide the same personalised experience across all user devices. Firebase Authentication was used to register and login users. The current sign in methods is via Email or Facebook. Firebase Authentication provided a backend service, easy-to-use SDKs, and ready-made UI library to authenticate the users.

Realtime Database

The Firebase Realtime Database is a cloud-hosted database. The data is stored in JSON and is synchronised in real-time to every connected client. If the application was to also be built on iOS or The Web, all of the clients will share one RealTime database instance and automatically receive updates with the newest data. It is a NoSQL database and as such has different optimisations and functionality compared to relation database.

Cloud Storage

The progress photos required users to upload photos from their gallery. Cloud storage by Firebase was used to store the user's photos. The Firebase SDKs for Cloud Storage add Google security for file uploads and downloads for the application regardless of network quality.

Near Field Communication (NFC)

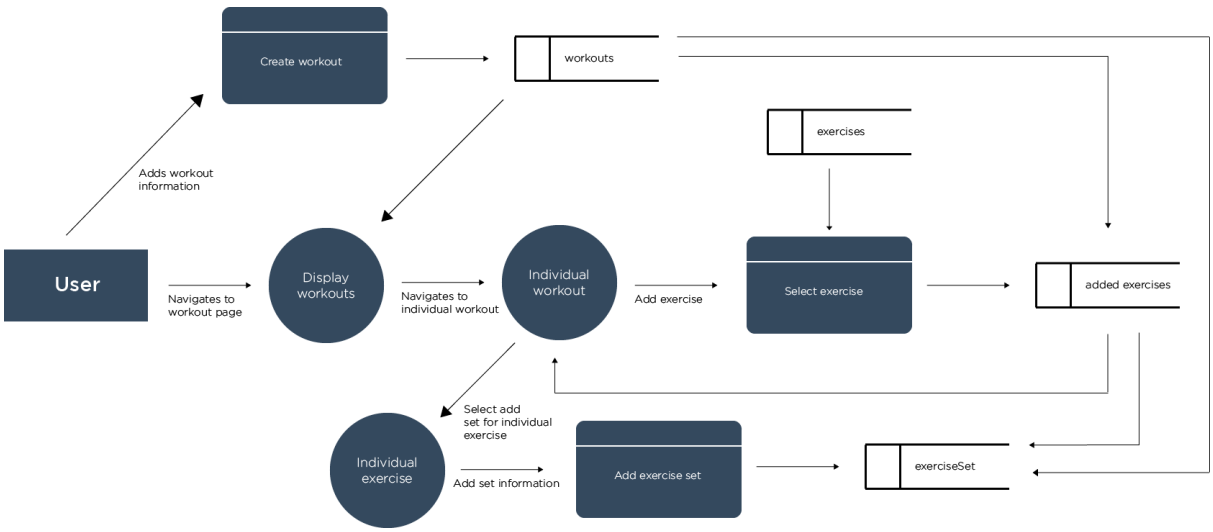
The NFC tag contains an ID from the Firebase Database for a specific exercise. Once the NFC tag has been scanned by the Android phone, it will check to see which exercise it is and add it to the workout. The ID was added to the NFC tag through an Android phone.

Data Flow

A data flow diagram was created for the major functionalities of the application to understand how they would interact with the users, databases, and activities. By creating a visual representation, it aided with describing the boundaries of the application. It also provided a detailed representation of the system and its components.

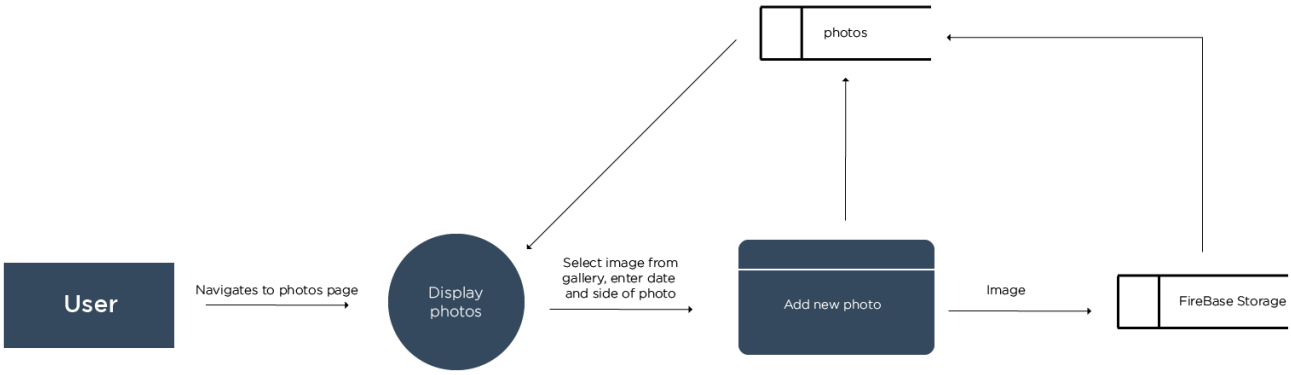
Workout Data Flow

The workout data flow diagram shows what would happen if the user was to create a workout and track exercises and sets for that workout. There are a number of collections that are interlinked with each other to ensure that the workout is linked together but the details for each collection is easily accessible and can make it easy to filter through.



Progress Photo Data Flow

The progress photo is displayed from the photos database and can be viewable when the user navigates to the photos page. When the user wants to add a new photo, they will need to select an image from the gallery, the date of when the photo was taken and the side of the photo. The actual photo will be stored in the Firebase Storage with the URL of the storage location stored in the photos collection.



Considerations

Exercise Data Acquisition

To collect the data for exercises there was a number of possible solutions. The first solution was to connect to a REST API by WGER online. However, the exercise information provided by the company was not in sync with how I wanted to represent the data. Another method to getting the data was to use a program to scrape data online in JSON. Scraping data from certain web pages, however, does violate their terms and conditions which meant this was unfeasible. The last solution was to create the exercise collection myself by finding information online. This meant that the data collected would be approved by myself, the exact information needed would be represented and it would be within Firebase and no extra coding would be required to connect to an API.

Implementation

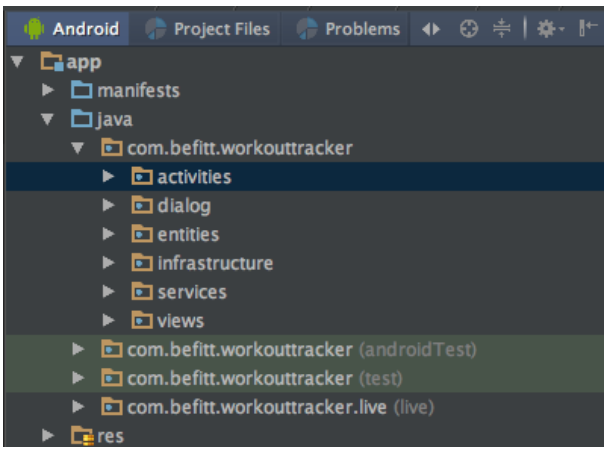
Tools Used

Product Flavour [16]
Developing clean and usable code was important when developing this application. A lot of courses were undertaken to get to the best solution for a fully functioning application. Product flavours is a powerful feature of the Gradle plugin in Android Studio that allows developers to manage different 'flavours' of an application. This is useful for when the same application is needed to be split into the full version and free version.

As the application was going to be free it meant that I could use product flavour to separate the UX code to the server side code. Breaking down the code into these two components was very beneficial as when there were any errors, it could be checked in its corresponding area. Understanding the code was easier and it made the code to be modular.

Structure

It was important to place all the Java classes into their corresponding packages to understand the code better and develop with ease. The IDE used was Android Studio which is powered by IntelliJ. This made it easy to place all the classes into their own relevant packages. Using product flavour meant that there were two different packages.



Activities

An android activity is Java demo that supports a screen or a user interface. A BaseActivity was created which is extended by all other activities made to prevent duplication of code. The BaseActivity contains code for when the user is logged in already into the application then they won't need to log in again. A check is done to see if they haven't logged in. If they haven't then they will be directed to the login page. If they are already authenticated, then a splash screen will be displayed and they will be navigated to the navigation activity. The user name and user email are called in the BaseActivity making it available throughout the application. EventBus needs to be registered in each activity but having it already registered in the BaseActivity means it doesn't need to be registered in each activity created.

Dialog

A dialog is a small window that prompts the user to make a decision or enter additional information. The dialog which is prompted does not fill the entire screen and is normally used for model events. A BaseDialog was created like BaseActivity which all corresponding dialogs extend from. Similar code to the BaseActivity is placed in the BaseDialog such as registering the bus, username, and email of the current user and closing the bus once the dialog has been closed.

Entities

The entities package contains all the relevant Java objects.

Infrastructure

The infrastructure package contains classes which are relevant to the infrastructure of the app. The first Java class is the ServiceResponse which contains code for error handling. It contains methods to set and get errors when communicating with the Firebase Database.

The Java file Utils contains all the variables which cannot be changed. These include references to the Firebase Database. It also contains methods to encode and decode the email. The Firebase Database cannot store full stops so the encoding method changes the full stop to commas. The decode email does the opposite. Another method within the Utils is logging out the user and navigating back to the log in screen.

Services

The package 'Services' contains all the request and response code to the database. For each of the UI code that needs to interact with the database, the request code will be called with event bus posting it. In the live package, the eventBus library using @Subscribe will connect to the UI code and handle all the server request and responses.

Views
The app makes use of RecyclerViews which is a container for rendering large data set of views that can be recycled and scrolled very efficiently. An adapter is needed to connect the UI to the object data. A holder or an adapter is used to sit in the middle and communicate between the object and the UI. The 'Views' package contains all the holders for the objects to communicate with the activities.

Walk Through

Register/Log In
The first functionality that was implemented was the register and log in. As the application required each user to have their own profile, a user account was required.

Two types of logins were implemented: Email & Facebook. Firebase provides authentication so other providers such as Google and Twitter can be implemented in the future.

Considerations
Google login was probably the best log in type to use the app as the code was available from doing the courses on Udacity (details in evaluation). However, as Google Log In had already been taught and implemented, it felt best to learn how to create two unique authentications that had not been done before.

Register Email - UX Code
In the base activity, the onCreate() (initialisation of the activity) initialised the Firebase Authentication. The onStart() method was overwritten to check if the user had already been signed in. If not the sign in page was displayed and the user can do the following.

When the user first downloads the app, the login page will be the first page to be open. A register button is located under the sign in page which will redirect them to the register page.

Firebase has an extensive guide on how to implement password authentication. It starts off by adding the Firebase Authentication to the Grade file. This allows the Android app to be connected to Firebase Authentication. A shared instance of the FirebaseAuth object is created in the BaseActivity to allow access to all the activities. When initialising the activity, a check to see if the current user is signed is done.

The user enters a username and their email address. If the username or user email is empty a method was created to show errors and that they need to enter both details. If the details are entered correctly the bus posts the server side code which

is explained below. If the user accidentally goes onto the register page, there is a button to navigate back to the sign in page.

Register Email - Server Side Code
A new account is created by using the createUserWithEmailAndPassword method provided by Firebase which passes in the user's email. A password also needs to be passed in and this is randomly generated in the app. The user will enter the password in the confirmation email sent to them.

Once the request has been submitted, it will check to see if the correct details have been inputted. If it is not unsuccessful, a message via a toast will be displayed. If it is successful the user details will be verified and added to the Firebase database in the 'user' collection with their email as the ID. The details stored will be their username and email they inputted, the date they joined and if they logged in with the password set to false (as they have not logged in with a password yet).

A message will be sent displayed via a toast to check their email. This will allow the user to confirm the sign-up and to add a password. Once the user has set the password, they will be able to open the application and sign in. The hasLoggedInWithPassword for their account will also change to true once logged in.

Facebook Log In
To implement Facebook login the SDK had to be downloaded and added as a dependency to the Gradle file. An 'app profile' had to be created on Facebook in the developers account section - workout tracker was registered as an Android app. Two details were needed to connect to Facebook: App ID and App Secret. Facebook login had to be enabled in Firebase Authentication with the corresponding App Secret. The Firebase Guide to integrating Facebook login was followed.

The Facebook Login button was initialised and configured to request the public profile and email of their account. After a user successfully signs into the application, in the LoginButton's onSuccess callback method, an access token for the signed-in user is exchanged for a Firebase credential, and thus authenticated using the Firebase credential.

The eventBus will send a request to the Firebase Real Time Database adding the user into the database with the same details as the Email login. The only difference would be in Firebase Authentication, the user will be registered as signing in with Facebook.

The following will explain the code more specifically with each unique functionality implemented.

Adding - (Example: Adding a Workout, Adding an Exercise)

To add a workout, the user will click on the floating action button (FAB) to open a new dialog. The will open a pop up with an EditText where the user can enter the name of the workout and click submit. The dialog has a built in onClick() method which has the Firebase request post by eventBus. The 'WorkoutService' has an 'AddWorkoutRequest' method which takes in the workout name, owner email and owner name. This is called within the onClick() method.

This is connected to the live package via the @Subscribe with EventBus to the LiveWorkoutService to add the details to the Firebase Database. If the workout name is empty then an error message will be displayed. If the response did succeed the following occurs.

The location of where the workouts are stored is made by creating a Firebase reference. The reference to specifically the workout being stored is workout/user email/pushID/workout details.
Firebase has a method known as Push() which creates a unique ID every time a new insertion is made to the database.
A new HasMap is created for the time the workout was created. The date and time are taken from the Firebase server.

A new object is initialised with the requests being passed into the relevant arguments from the UX. To add the new workout into the database, a child node is added to the reference using getters to get the corresponding information from the object.

Once the workout has been created successfully a toast message will be displayed saying 'Workout Created.'

It is stored in the database as follows:

Pseudo-Code

Algorithm: Add Workout to Firebase

```
        Set param: AddWorkout Request

        Start:
            initialise AddWorkout Response

            IF firebase request for workout name is empty THEN
                display toast

            IF response succeeded THEN
                initialise Firebase reference

                HashMap timeStampCreated
                put in timeStampCreated ServerValue.TIMESTAMP

                initialise workout object

                add a child to reference and set object value with getter
                repeat for each variable in object

                display successful toast

            eventBus POST the response
        END
```

Adding Exercises

When adding exercise, it needs to be linked to the corresponding workout. When a user selects an exercise they want to add, the exercises are added to a new collection known as 'addedExercises.'

When a user adds an exercise, the workout ID is used a reference, then a push ID is used instead of the exercise ID to prevent a collision. The details of the exercise are added as a child to the push ID node.

The exercises are linked to the corresponding workout in the database from the figure above (as shown with arrows):

A user can also add sets for each exercise and the same method was used to complete this. Each set also has a reference to the workout to make it compatible for deletion which is explained further below.

Displaying (Example: Displaying a Workout)

A RecyclerView [17] is a flexible and efficient way to display datasets. It is a container for rendering larger data sets of views that can be recycled and scrolled through efficiently. It helps with memory and storage preventing the app from running slow and crashing.

RecyclerViews were used throughout the application to display the data from the Database. The RecyclerView will display all the data from a dataset but the individual design for each of the dataset had to be created. For the workout, this meant how each of the new workout added to the database will look.

A holder is needed to connect the UI views to the object. WorkoutViewHolder is the class that calls the individual items in the workout view: workout name, owner name, and date created. The method populate() has the POJO as the argument and sets the corresponding object detail to the corresponding view by using the getter methods. For example, the workout name was set to the text view via the ViewHolder.

For the workout, the date and time it was created were stored as a Firebase server value. A method was created to convert the time into a time and date that was readable to the user.

The view holder extends to RecyclerView.ViewHolder so that it describes an item view and metadata about its place within the RecyclerView.

A recycler view adapter provides binding from a data set to views that are displayed within the recycler view. Firebase provides a FirebaseRecyclerAdapter to set to the RecyclerView. The adapter requires four arguments: the object, layout of

the individual workout, the view holder and the query of how the data will be viewed. The workout is queried differently which will be explained later.

The FirebaseRecyclerAdapter has a pre-defined method populateViewHolder which populates the view holder. The holder is called with the method populate() and the workout object is passed through. The object variables had to match exactly how the Firebase data is stored as otherwise the data won't be displayed and a ‘bounce to type’ error will be displayed.

A Linear Layout Manager is created to display the data in a linear format. Firebase displays the data how it is stored in the database. To have the latest workout displayed at the top, the linear layout had to start from the end and in reverse. The FirebaseRecyclerAdapter is set to the recyclerView which will then display all the data from the database.

The recycler view is currently empty and needs to connect with the database to display all the data. To retrieve the data, a new class was made in the WorkoutService with a request to the Firebase and a response that contained the Workout POJO and ValueEventListener. ValueEventListener was used to read data at a given path and listen for any changes.

The code for the LiveWorkoutService first instantiates the response. The onDateChange method was used to read a static snapshot of the contents in the path passed in, as they existed at the time of the event. This method is triggered once when the listener is attached and again every time the data, including the children, changes. The event callback is passed with a snapshot containing all the data at that location, including child data. If there is no data, the snapshot returned is null. The object variable names had to match exactly with how the data is saved in the Firebase Database.

The ValueEventListener also defines the onCancelled() method that is called if the read is cancelled. This can happen if the user doesn’t have permission to read from a Firebase database location.

The workouts will now be displayed in the recycler view that the user has added and every time they update it, the recycler view will be updated.

Pseudo-code

```
Algorithm: Get Current Workout

    Set param: DisplayWorkout Request

    START:
        initialise DisplayWorkout Response

        initialise valueEventListener from Response with Firebase reference
        from Request

        valueEventListener method onDataChange
            set param: DataSnapshot

            set workout response to dataSnapshot and get value

            if workout response IS NOT null THEN
                eventBus POST response

        valueEventListener method onCancelled
            display toast with Firebase Error
    END
```

Updating (Example: Changing Workout Name)

It was important for users to be able to change certain elements of the app in case they made a mistake e.g. when they add a workout name for a created workout. If they made a mistake, they will be able to change it and thus update the details in the Firebase Database.

When a user click on a 'pencil' icon when they're on an individual workout, a dialog will popup with the current name already displayed.

The workout name is passed from the individual workout by adding it to an ArrayList and the newInstance of the dialog fragment will pass it through. The EditText in the dialog will then be populated with the passed workout name.

The user will be able to enter a new workout name and on submit of the dialog it will send a request to the database by EventBus post method. The request passes in the same details as creating the workout except the new workout name is passed.

The server side code for updating the workout name use of Firebase's updateChildren() method. The method allows writing to a specific children node without overwriting other child nodes. A reference to the Firebase Workout

collection was initially made. The timeLastChanged node in the workout is updated to the current time of the change made. the new date is stored within a HashMap and the children to the reference is updated with the new workout information.

Pseudo-code

```
Algorithm: Update Workout Name in Firebase

        Set param: ChangeWorkoutName Request

        START:
                initialise ChangeWorkoutName Response

                IF request for new workout name is empty THEN
                        set property errors from response

                IF response did succeed THEN
                        initialise Firebase reference

                        initialise HashMap timeLastChanged
                        put ServerValue timestamp in HashMap

                        initialise HashMap newWorkoutData
                        put new workout name from Request
                        put timeLastChanged

                        update children in reference with the newWorkoutData

                EventBus POST response
        END
```

Deleting (Example: Deleting a workout)

Deleting a workout was important for users to do. The example explained is for exercises which had a number of locations that deletions had to occur at as it contains exercises and sets for each corresponding exercise. Users were able to delete a workout by using the onLongClickListener() which opens a new dialog and passes in the workoutID and user email. The deletion fragment works the same as changing a workout by using EventBus post method to send a request to the Database.

To delete a workout server side, a reference to each of the Firebase Database collections were made using the workoutID. The simplest way to delete data was to call removeValue() method provided by Firebase. This was placed on a reference to the location of the data.

Workout Sort Query

A user is able to sort the workout to their preference: name, email or by time. SharedPreferences was used to sort the workout based on the user preference. SharedPreferences is from the Android SDK which is used to store and retrieve application preferences. They are sets of data values that are stored persistently meaning that even if the application stops or turns off, it will still exist.

A new class was made named 'SettingActivity' which extended the 'PreferenceActivity' which included a set of core properties which allowed specifying things such as the title for the setting and the default value. The preference used to change workout order was 'ListPreference' which opens a dialog with a list of radio buttons. The saved value was stored as String. The ListPreference was configured so that an array of String were placed: time, name and email, and a separate array with the values for the database to render: orderByPushKey, workoutName, ownerEmail.

The method onPreferenceChange was called when a preference had been changed by the user. The method is called before the state of the Preference is about to be updated and before the state is persisted.

The setPreferenceSummary() method was used to check if the preference is an instance of list preference. If it was it was initially cast and then the index to where the string value is located inside the ListPreference was found which thus finally binds to the preference.

Back in the workout main activity, a variable was created which gets the user input from the 'LIST_ORDER_PREFERENCE.' If the user input equaled to 'orderByPushKey' the query was set to order by the pushID otherwise it was ordered by child based on the user selection.

Adding Exercise with Near Field Communication (NFC)

Configuring the NFC Tags

A library was found on GitHub to configure NFC on tags to be able to read and write from it. To get NFC working for users a uses-permission for NFC had to be stated in the Manifest. An activity was made which extended NfcActivity making it accessible to pre-set methods for reading and writing.

To write to the NFC tag the activity had to implement AsyncUiCallback. The method performWrite was used to write String to the NFC tag. The tag contained the exerciseID which corresponded to the one in the Firebase Collection. The testing Android phone has NFC which allowed the exerciseID to be added on by simply tapping on the tag.

To add the exerciseID to the tags as efficiently as possible, an EditText was created. Once running the app on the phone, a different ID was entered for a unique tag.

Reading the NFC Tag
When a user swipes a tag with NFC-enabled the corresponding exercises will be displayed in their corresponding workout.

The NFC was implemented within the individual workout activity. The Foreground Dispatch System had to be implemented. The foreground dispatch system allows an activity to intercept and claim priority over other activities that handle the same intent. Using this system involves constructing some data structures for the Android system to be able to send the appropriate intents to the application.

The PendingIntent object is created so that it can be populated with the details of the tag when it is scanned.

Intent filters are declared to handle the intents that needed to be intercepted. The foreground dispatch system checks the specified intent filters with the intent that is received when the device scans a tag. If the details are matched, then the application will handle the intent. However, if it does not match, the foreground dispatch system falls back to the intent dispatch system.

mTechLists is an array of tag technologies that the app wants to handle. The Ndefformatable.class.getName() method is used to obtain the class of the technology that needs to be supported.

The onPause(), onResume() and onNewIntent() methods are overwritten to add logic to enable and disable foreground dispatch when the activity loses and regains focus. The method enableForegroundDispatchSystem() had to be called from the main thread and only when the activity is in the foreground - calling in OnResume() guarantees this. The onNewIntent callback is required to process the data from the scanned NFC tag.

Once the NFC tag has been scanned an EventBus post request to the database is made where the workoutID, the exerciseID as a message and user email is passed. The only data that was stored in the tag was the exerciseID. This meant that when adding an exercise, swapping the tag wasn't enough to just add as exerciseName and its corresponding pushID also had to be stored.

The Firebase Database has a unique URL to the exercises collection. A reference to the Url was made and the exerciseID in the tag completed the URL giving access to all the details of that exercise. To read data, as previously mentioned, the Firebase addValueEventListened() method was used to the database reference. The POJO for the exercise was instantiated and linked to the Firebase collection with dataSnapshot.getValue() method. The getters were able to get the details of the of

the exercise and add it to the database reference ‘addedExercises.’ This was slightly different to the adding exercise mentioned above as a snapshot of the data wasn’t required.

Pseudo-code

```
Algorithm: Add Exercise with NFC to Firebase

        Set param: AddExerciseNFCRequest request

        initialise AddExerciseNFCResponse response

        IF response succeeded THEN
                set exerciseReference to request.firebaseURL
                set Firebase URL addExerciseReference
                set Firebase URL workoutReference
                set Firebase URL individualExercise + exerciseReference

                addValueEventListener for individualExercise

                        valueEventListener Method onDataChange
                                Set param: DataSnapshot

                                        set Exercise object to get value from
                                        DataSnapshot

                                        add a child to addExerciseReference and set
                                        object value with getter
                                        repeat for each variable in object

                        valueEventListener Method onCancelled
                                set param: Firebase Error

                                        display toast with Firebase Error

                HashMap timeLastChanged
                put timestamp of server value in HashMap

                HashMap newWorkoutData
                put timeLastChanged in newWorkoutData
                update children in workoutReference with newWorkoutData

        eventBus POST response
```

Sharing a Workout

Display Users and Add Friend
This activity displays all the users who have signed up and have logged in into the app so that the user can find their friend and add them to their friend's list. The activity displays the users using a Firebase Recycler Adapter, as previously mentioned. To add a user as a friend, an IF statement was used where if the plus icon was clicked on the user was added to their friend's list or if the check icon was clicked then the friend is removed from their list. This updates the Firebase Database real time so users can add and remove friends without any problems.

Share workout with Friends
The first step was displaying the added friends in the activity using the Firebase Recycler View again. Each user has an icon to either add or remove from sharing the individual workout. isSharedWith() method is used to get all the users its shared with. If it's shared with a user then the icon will display a check otherwise the icon will display an add/plus icon.

An onClickListener() method is used for when a user clicks on the icon. If the user has already shared the workout with the friend and they click on the icon then remove that user from the database and update the icon to a plus. If they haven't shared the workout with them, then add them to the workout by adding the corresponding workout to the friend's list and adding them to the sharedWith collection.

A method was created to update all the workouts and their references. If the usersSharedWith isn't empty then for each of the users in the list it will add the workoutID. The friend will also not be allowed to delete the workout: only the owner will be able to remove them. This then will update the Firebase collection so the friend has access to the workout to add their own exercises.

Progress Photos

A user is able to upload photos from their gallery to track their progress. This feature makes use of Firebase Storage. Firebase Storage in instantiated in Gradle to access to it and the reference is created in the AddPhotoActivity. The reference is under a folder 'Photos' followed by the user email ending with the last path of the file path uploaded. Once the photo is uploaded on Firebase Storage, it will have a unique URL referencing to it with a unique token only accessible to the owner. Once the image is uploaded to Firebase Storage, the URL is also stored in the Firebase Database with the ID, name, date and email. A progress bar is shown whilst its uploading and once completed, the image is visible in the PhotosActivity.

The PhotosActivity uses an Android RecyclerView instead of FirebaseRecyclerView to display the data and displays data the same way as the data as explained

previously. The holder which displays the image, however, is making us of Picasso by getting the PhotoURL and displaying it.

Watching YouTube Video

Users are able to watch YouTube videos on how to perform exercises. This makes use of the YouTube SDK [18]. It was initially set up by downloading the SDK and adding it as a dependency in the Gradle file. To play YouTube videos and have access to the methods, it had to extend the YouTube Activity. A pop-up was created using DisplayMetrics to a specific size based on the users' phone to make it easier to watch. The YouTube is initialised first and if it is successful it retrieves the URL from the individual exercise and plays the video.

Testing

Two different forms of testing were undergone to ensure that the app performs the way it was developed. The first type of testing undergone was test cases. Once completed, user-based testing was conducting by adding the app to the play store. If any errors were to occur, changes were implemented and retested to ensure they meet the criteria set out.

Purpose of Testing

Testing was conducted to ensure that all the aims of the project were met and made functional. Ensuring each aspect of the project was working the way it was intended to was important so that there is an understanding of the application and how it should operate. If there were any problems, then it would be easy to understand what went wrong and make the adjustments. The test cases went through each of the functional requirements of the application, step by step, and had an expected result. This ensures that each feature performed as intended to. The user-based testing was slightly different as there was no guidance on how to use the application. The users were able to use the application and a crash report was analysed.

Test Cases

The first form of testing conducted was test cases. Test cases were made with pre-defined steps to ensure the application performed all the steps and provided the expected results. The test cases were run on a physical phone (Samsung Galaxy Note 4 will API 23) and an emulator on a MacBook Pro (Nexus P) to ensure it worked. *(see appendix for test case evidence)*

Log In

Test Case: 1	**Register (Email & Password)**
Description	Register with email and password and be authenticated with Firebase
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User has downloaded the app from the play store Has not signed up with Facebook
Basic Flow	1. Open the app 2. Navigate to register page 3. Enter username and user email and click submit 4. Check email for confirmation and add password 5. User enters email and password 6. Navigate to main menu
Results	Passed - User authenticated with Firebase and was able to log in and view the main menu

Test Case: 2	Facebook Log In
Description	Authorise with Facebook and log in
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Has not signed up with email
Basic Flow	1. Open the app 2. Click on Facebook Log In 3. Navigate to Facebook Authorisation 4. Confirm Authorisation 5. Navigate to main menu
Results	Passed - User authenticated with Firebase and displays Facebook log in used

Workout

Test Case: 3	Add Workout
Description	Create a workout and display on workout page
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User has successfully logged in
Basic Flow	1. Navigate to workout page 2. Click on floating action button to open dialog 3. Enter workout name 4. Click submit 5. Visualise newly created workout
Results	Passed - Workout was added to the workout and displayed in the workout section

Test Case: 4	Add Exercises to the Workout
Description	Select exercises and add them to the created workout
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User has created a workout and on that corresponding workout page
Basic Flow	1. Click on floating action button to add exercises 2. Click on exercise(s) to add to the workout (display toast to say exercise was added) 3. Go back to individual workout 4. Added exercises will be displayed
Results	Passed - Exercises added to the created workout

Test Case: 5	Add Sets to Exercise
Description	Add weights and reps completed for each exercise and display it for the corresponding exercise
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User has added an exercise to a workout User has clicked on an exercise and navigated to set activity
Basic Flow	1. Click on floating action button to add set details 2. Enter exercise reps and weight 3. Click 'save' 4. Set is displayed in the corresponding exercise
Results	Passed - Exercises set is visible in the corresponding exercise

Test Case: 6	Rename a workout
Description	User changes the name of the existing workout to a new name
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Workout has been created
Basic Flow	1. User navigates to corresponding workout 2. Click on 'pencil' icon to edit name 3. Enter new workout name 4. Click submit 5. Updated name is visible
Results	Passed - New workout name is visible

Test Case: 7	Change set details
Description	User is able to change the weight or reps for a set already created
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Exercise set has been created
Basic Flow	1. Click on 'pencil' icon on created set 2. Add new weight and reps 3. Click submit 4. New exercise set details are visible
Results	Passed - Exercise set details have been updated

Test Case: 8	Delete set
Description	A user is able to hold on a card view for a set and delete it
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Exercise set has been created
Basic Flow	1. Hold on the exercise set until dialog pops up 2. Confirm deletion of set 3. Set is removed from the view
Results	Passed - Exercise set is not visible

Test Case: 9	Delete Exercise
Description	Deleting a exercise from a workout
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Exercise has been added to a workout
Basic Flow	1. Hold on the exercise until dialog pops up 2. Confirm deletion of the exercise 3. Exercise is removed from the view
Results	Passed - Exercise is not visible

Test Case: 10	Delete Workout
Description	Deleting a created workout by the user
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Workout has been created
Basic Flow	1. Hold on the workout until dialog pops up 2. Confirm deletion of the workout 3. Workout is removed from the view
Results	Passed - Workout is not visible

NFC

Test Case: 11	Add Exercise with NFC
Description	Scan a NFC tag and add a exercise to a workout
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Workout has been created
Basic Flow	1. NFC is enabled on the device 2. Scan NFC tag 3. Exercise is automatically added to the workout 4. Exercise is visible
Results	Passed - Exercise is visible

Exercises

Test Case: 12	Display Exercise Details & Watch YouTube Video
Description	Be able to click on an exercise and see the details on how to do it and watch a Youtube video on how to do it in app
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User is logged in Navigate to the exercise page
Basic Flow	1. Click on exercise 2. Navigate to exercise details page 3. Exercise details is displayed 4. Click on Youtube video thumbnail 5. Youtube video is loaded
Results	Passed - Exercise details visible and Youtube video is playing

Body Weight

Test Case: 13	Add body weight
Description	Add body weight for a particular date
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	User is logged in Navigate to the body weight page
Basic Flow	1. Click on floating action button to navigate to add body weight activity 2. Enter body weight and date it was taken 3. Navigate to body weight page automatically 4. View newly added body weight
Results	Passed - New body weight is visible

Test Case: 14	Delete Body Weight
Description	Delete added body weight from the body weight section
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Added a body weight
Basic Flow	1. Hold on the body weight card view 2. Click ‘confirm’ on the delete dialog pop up 3. Body weight activity updates and removes the body weight
Results	Passed - Body weight is deleted

Test Case: 15	Update Body Weight
Description	Update body weight from the body weight section
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Added a body weight
Basic Flow	1. Click on the body weight list item 2. Enter the new body weight 3. Click 'confirm' 4. Body weight is updated
Results	Passed - Body weight is updated

Progress Photo

Test Case: 16	Add Progress Photo
Description	Upload progress photo from gallery and add it to the photos section
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Progress photo already taken and is in the gallery
Basic Flow	1. Click on floating action button to navigate to add photo activity 2. Enter side of photo e.g. front, side or back 3. Enter date of photo taken 4. Click on 'select photo' button 5. Select application to retrieve image from 6. Select the image 7. Preview the image 8. Save image - click on check icon in toolbar 9. Dialog with progress bar to view until completion 10. New photo is visible in photos activity
Results	Passed - Photo is visible

Sharing Workout

Test Case: 17	Add friends to share workout
Description	Add friend to friends list to select to share workouts with
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Workout has been created
Basic Flow	1. Click on individual workout to share 2. Click on 'share' icon in toolbar - navigated to 'view friends' activity 3. Click on 'add' icon in toolbar - navigate to 'all users' activity 4. Click on 'add' icon next to email of user to add to friends list 5. Added friends is visible in friends list
Results	Passed - User is visible in 'Friends List'

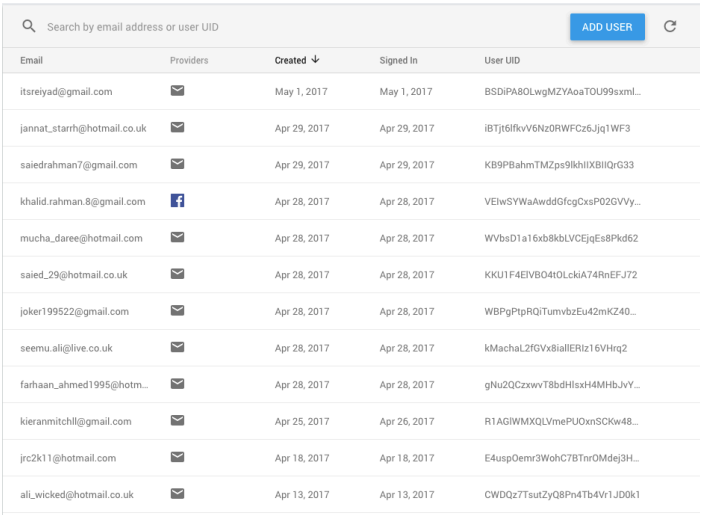
Test Case: 18	Select workout with friend(s)
Description	Select the friend(s) from the friends list to share the workout with
Actors	Gym goers, fitness enthusiastic, app users
Pre-conditions	Workout has been created User is added to friends list
Basic Flow	1. Click on ‘add’ icon next to users name 2. Workout visible in friends workout list
Results	Passed - Workout is visible in friends workout list

User Testing

There is no better testing that users being able to use the application and getting reports/notifications of any crashes. The application was used by over 15 users on a wide variety of devices.

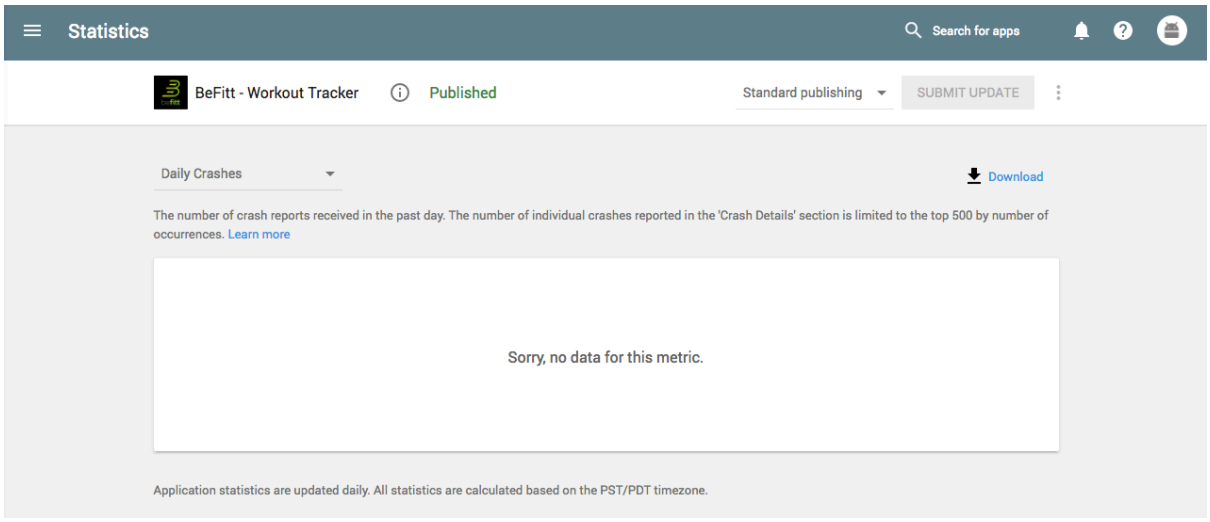
Register/Log In

Authentication was successful for all users who downloaded the application and signed up. Users primarily signed up via Email. If the user signs into the app, they are added to the ‘user’ collection and the child ‘hasLoggedInWithEmail’ will equal true.



Crashes

To see if the app was working with users or if there was a problem, a crash statement would be present in the Play Store Console. So far no crashes have been made in the app meaning that all features are working accordingly.



Testing Review

From both testing completed it was shown that the app performs the way as intended to. The test cases were all passed and met the requirements of the application. The user testing showed no crashes up until the last day of the project. This showed that the application hasn't crashed and that there are no errors in how it is meant to perform.

User Feedback

It was important to get feedback from the users who used the application to make amendments and push updates that help them use the application better. The application was uploaded onto the Play Store and promoted for users to download. Before pushing it live, some ideas of what I felt the application was lacking could be improved on were stated. This is to see if the user feedback given matches the ideas that were initially written.

Initial Thoughts

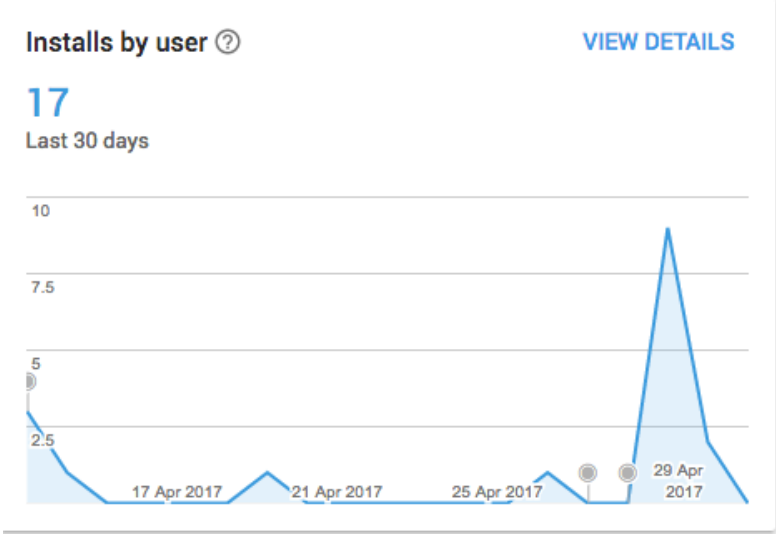
There are a number of areas that the application could be improved on. As the app currently stands, it is a foundation to be developed on. Initial thoughts on the app are as follows:

1. Too many pages to add workout: could be taken from four pages to two pages for easier navigation
2. Improve search functionality to be able to find exercises easier
3. Click on exercise to get information on 'how to do' instead of having its own section
4. Not knowing how to share a workout
5. Not knowing Material Design: e.g. deleting on long hold
6. Exercise description to be broken up so easier to read
7. Display text for empty workout, photo, exercises etc

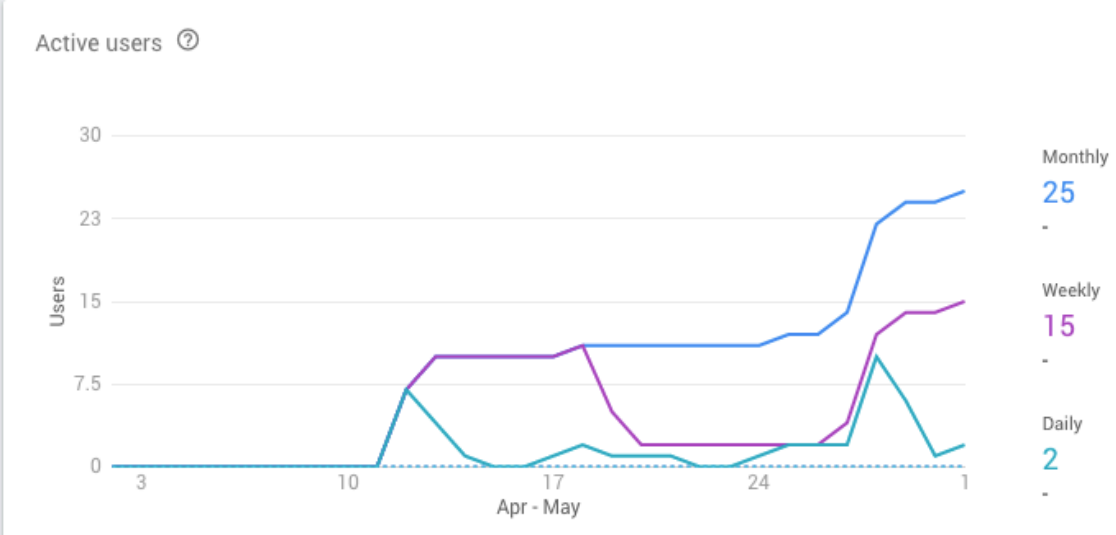
App Upload Stats

The app was uploaded to the Play Store with the initial thought of taking 2 weeks to be reviewed but it actually took a day. Two updates were made to the app during this time before promoting download.

The following stats are as follows for the app.



Installs by users taken from Google Play Store
Active Users from Firebase Analytics



How feedback was gathered
The general idea was to get as many users as possible by promoting the app on my personal social media: Facebook & Instagram. Posts were designed to get users to download the app, write reviews and message back feedback. Messages were sent to friends to promote the app and get users to download the application and in total 25 users downloaded the application.

This didn't feel like a good way to get information and feedback for improvements. So 5 fitness individuals were found (friends) who are active gym members. Honest feedback was vital so they were asked to use the application and give honest feedback based on what they liked, disliked, possible improvements and much more. Details about the user feedback are available below.

Changes for future feedback collection
If the project needed to gather information again from users, a promotion would be done through social media but permission would also be asked in gyms to speak to active gym members who are not friends. By getting members of a gym to use the application, better feedback would have been received and a larger data set of information would be available.

In addition, speaking to local influences, personal trainers, coaches and fitness teachers would be approached to get a different perspective to see if the application gave the correct information and if not, what could be improved.

Post User Feedback

A survey was given to users to get feedback on their thoughts of the app. The survey was split into two sections: features and design. This was to get an idea on if the app features were helpful to the users and if not, how they could be improved. The design aspect was important to get information on to see if it matched HCI principles and if not, why it didn't or how it could be improved.
(see appendix for detailed results)

Considerations made for Survey Design
There were a number of factors that influenced the design of the survey. Firstly, as the project was to build a fully functioning app with good UX the survey had to split into two as mentioned above.

In regards to the features, it was important to find out if they were actually helpful to them. There would be no point in adding a feature if it didn't bring any value to the user. Finding out the reasons why or why not was critical to the analysis of the reasonings. Although research had been done initially on what was important in a fitness app, it felt like a good idea to find out if they had any other ideas that could be implemented after using an app that had some features already.

It was vital to know if the user was able to navigate around the app easily and efficiently. Good UX for any application is key as it is a small screen and limited information can only be shown. As screens are very close to the eye and can be bright, colours are important. The colours used shouldn't affect the eye, even after long period of times. It was important to find this out from the users.

The questions and answers are as follows:

Features

Q: Are the current features helpful to you? (e.g. workout, photo, and weight tracking)

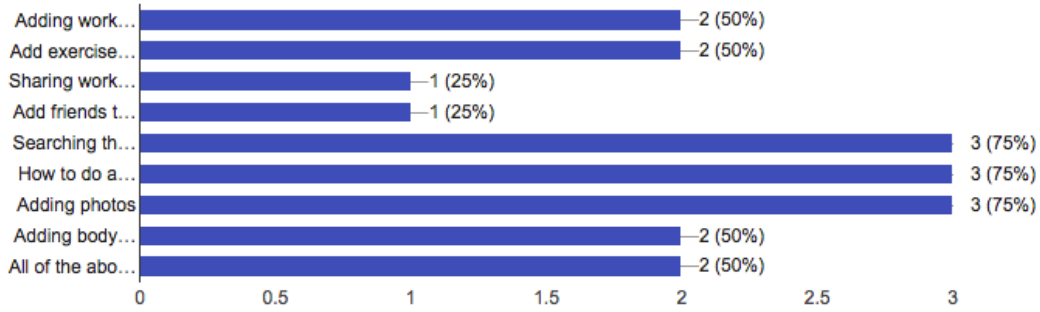
A: Yes: 100% | No: 0%

simple way to track progress, easy to understand, useful to keep all in one app.
They are helpful as I can keep tabs of my progress which is useful for looking good
Helps keep track of progress, and I like that i can add photos to show visible progress.
It is an efficient way to record your work out without the need of a classic cross out chart that would usually be taken in. It has a wide range of pre-listed work outs which allows them to be selected with ease. It has a weight tracker and a photo file to continually check progress without the need of any other means of storing this information. The app itself was very easy to use.

Q: If so, how are they helpful? If they aren't, why aren't they?
Q: What new features would you like to see?

very short tutorials rather than lengthy descriptions.
I would like to nutritional advice such as macros and calories
Better search, ability to add new exercises, and food tracking
Some of these work outs I havent seen before and would like to try them, i could simply google this but it would be useful and quick if there was a tab to show me how to carry out the exercise. Also im not sure if there was a way to name your own exercise and add it to the workout

Q: What features of the app did you see? (If you saw a feature but didn't use it please check)



Q: Would you use the app if the feedback you've provided is implemented?

A: Yes: 100% | No: 0%

Design

Q: Was it easy to navigate around the app?

A: Yes: 75% | No: 25%

Q: Why did you find it difficult to navigate?

A:
login was not very clear, an option of forgotten password not apparent, workouts should be categorised into legs/chest etc before they are listed. rather than clicking a workout and finding out it targets an area you're not wanting to focus on. My knowledge on the various workout is limited, would be simpler for the target areas to be clearer before knowing the types of exercises.

Q: Did the colour scheme affect your eye?

A: Yes: 0% | No: 100%

Q: Was there too many pages to navigate through?

A: Yes: 0% | No: 100%

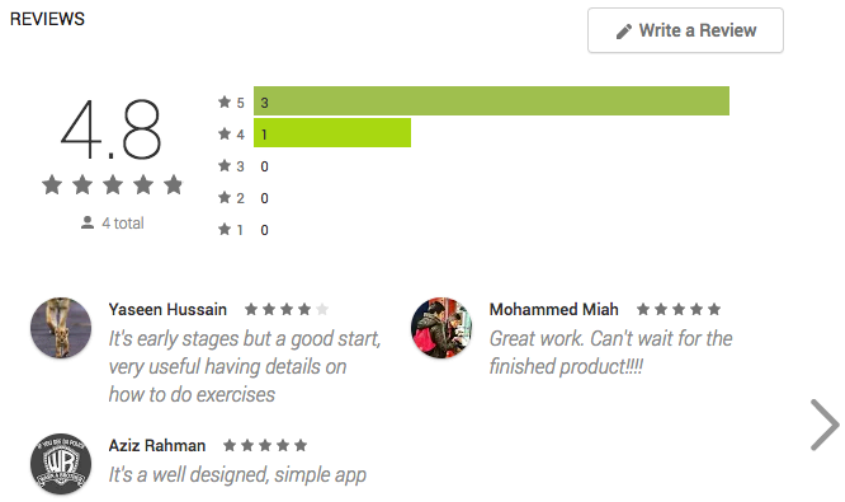
Q: Could the app look simpler or designed better? If so, how?

think I've answered this in the navigating section. Additionally, would be nice to be able to personalise the app, make a personal list by saving favourite exercises listed on the app, change a little bit of the colour scheme and profile- to make it feel a bit more fun and personal. Possible add a structured action log where you can note down number of reps/sets done for certain weights.
No very well designed
-
Simple and elegant is always the best combination, in this case it was

Q: How was your journey through the app and your experience?

pretty good, very good colour scheme and features to track workout progress. the photo idea is a good idea. an app I look forward to using.
Very easy and everything is where it should be
Positive, but needs some improvement in the search options
Ive only used it a couple times so far and I recommended it to my brothers also as it is an efficient way to keep all workouts in one place without the need to jump from other secondary apps which dont integrate all everything

App Reviews



Post User Review

After receiving the feedback from the users who used the app the following is a review of the user feedback matched against the initial pre-launch review.

Too Many Pages
Based on the feedback received from the users, it does seem like they felt there weren't too many pages to navigate through. Even after the feedback received, it will be a good idea to reduce the number of pages to navigate through as over time, it could get tedious and be time-consuming.

Search Functionality
The search was very basic and didn't work accurately and this was visible from the feedback given. The users found it difficult to find the exercises and as they weren't categorised, a user didn't actually know the workouts. This means taking into consideration the level of fitness enthusiastic is important.

Exercises
It wasn't specifically mentioned in the feedback given on having the how-to in the workout section however one of the users did mention that they did not know which exercises were what. It would be beneficial to add this to the workout section so that users are able to click on an exercise and see what to do but also to add it.

'Seen' Features
It was mentioned that users won't know how to share, delete a workout. From the survey results, it does seem like not all the users knew how to exactly find all the features meaning it might be a good idea to implement an 'introduction' to the app on where features are or making them a bit more clear.

Feedback Conclusion
No other mention of the issues that was initially written were faced by the users meaning that the application performed slightly better than expected. There are other improvements that could be made to the application with design and other features that could be added to help the user improve their fitness lifestyle.

Many features that users wanted was related to diet. This could be a possible future implementation so that they have access to information regarding gym and diet. Another feature requested was to add their own exercises which are possible to do but this can add duplicate exercises into the database with no information on how to perform it.

In conclusion, the feedback given was positive with some adjustments to be made. It seemed the users has an understanding this was a V1 application.

Post Feedback Pivot from Initial Plan

Although the app was available on the Play Store and feedback was given from users, further feedback and downloads were hoped for.

The initial plan was to get a focus group of 10 users and split them into two whereby one group had a tutorial on how to use the app and the second where they use the app without any prior tutorial. The results would have been analysed and adjustment would have been made to the app accordingly.

Evaluation

Acceptance Criteria

In the initial plan, a number of requirements were set to achieve. Below is the list of requirements and how it was met.

There are a number of achievements made after completing this project. They are as follows.

Track Workouts - *Completed Successfully*
The application tracks workouts created by the user and displays it within the workout page. The workout can also be deleted or renamed.

Track Workout in Calendar - *Not implemented*
This was not implemented as the date is automatically set when the user adds a workout. The consideration taken was that users won't be using the app unless they're training so adding to a calendar was unneeded.

If more time was available, the UX of the design could change to cater to adding to a calendar so users can then schedule workouts.

Track Exercise for the Corresponding Workout - *Implemented Successfully*
A user is able to add exercises to a workout.

Data Visualisation of Each Exercise - *Not implemented*
This was not implemented due to time constraints and learning how to link Firebase data to a Graph View in Android. The time taken would have prevented the completion of other important features of the application.

Add Exercises with NFC - *Implemented Successfully*
The user is able to add exercises with NFC.

Display Information about Exercises - *Implemented Successfully*
The user is able to view details on how to perform exercises with details being broken down. The application goes one step further and has connected to YouTube and videos are displayed on how to perform the exercises.

Log In System with Google - *Implemented Successfully with changes*
The initial plan was to implement Google but as the knowledge and code for Google login were available, this changed for learning purposes. The log in changed to Facebook and Email login. Google Log In can be implemented later very simply.

Upload to Play Store/Focus Group to Download - *Completed Successfully with changes*
The app was successfully uploaded to the Play Store with a number of updates made to ensure the app performed as intended to. As there was limited time, a focus group was unable to be gathered so the app was pushed to friends and family through social media.

Log Weight & Show Trend - *Implemented Successfully with changes*
The user is able to log in their body weight however as mentioned above, connecting Firebase data with Android Graph View was not learned so trends were unable to be implemented. If more time was available, this would be implemented to display a line graph.

Upload Progress Photos - *Implemented Successfully*
The user is able to upload photos from their gallery and view in the app to see their progress.

Project Achievements

Mobile Development

When the project initially started: no previous Android app had been developed. Simple apps with simple features were played around with but the knowledge of Android development was minimal. To go out and learn Android in the short time given and apply the knowledge to the project was hard but rewarding. A number of courses were taken to learn how to develop on Android with clean and concise code. By learning how to program on Udacity by Google developers meant that the best coding practices adhered too and principles that industry standard application have were taught.

Learning Android and developing a fully functioning application was a big achievement because it allowed confidence in mobile development to increase and be able to show family, friends, and companies an application that was developed from scratch.

Firebase

There was no experience at all with Firebase, only knowing what it's potential was: building apps quickly. It was quickly noticed that the community for Firebase was very small, as it was a new technology. Firebase had only launched in 2011 and was acquired by Google in 2014. After Google acquired it, they changed the system, depreciating the 'legacy' system on May 18, 2016. This meant the information available was limited and reading the documentation and tutorials were key.

A number of courses were taken to learn Firebase and best practices. This took a lot of time to do, however, was rewarding because once completed was very straightforward to implement. Understanding of how Firebase worked was key, especially as previously the only knowledge was in SQL and as Firebase was NoSQL it meant changing and learning the new practices.

The end project makes use of Firebase extensively. The Realtime Database was used primarily for data storage, Firebase Storage was used to store and retrieve images from and Firebase Authentication was used to authenticate users. Analytics by Firebase was used to see the data on user usage of the app.

By completing courses and reading the documentation updating, adding, deleting and displaying data is known how to do in Android. To do this in less than 3 months is a personal achievement.

App Upload

One of the goals that were set out was to upload the app on the Play Store [19] and get used to downloading the app. Being able to have the app published on the Play Store is a big achievement as it is the first app developed to be live with over 15 people who have downloaded and used it within a 1 week of uploading. Getting feedback from users on what they liked, didn't like, feedback on improvements and features was exciting. Pushing updates and getting users to download the updates to fix bugs was interesting to see and learn from.

Poor Decisions Made & If Project Was Done Again

There were a number of poor decisions that were made by myself during the project but a number of lessons to take and learn from. To learn Android & Firebase separately made it very difficult in how they interlinked. Some courses that were taken for Android was teaching Android principles but would teach SQLite as the database. This was not helpful at all as Firebase worked completely different. This was recognised later and thus changes were made to find courses that were relevant to Firebase and Android. A payment had to be made to learn, but it was worth it as the knowledge gained was greater than the small payment.

In addition, the initial plan that was made included hours to be worked through the weekend. The hours were less than the weekdays as I was working but some allocated time was given to the weekend. In reality, no time was spent during weekends on working on the project as tiredness got the better of myself. Not thinking ahead and how work would have an impact on the work rate meant delays were being made in the plan and catch up had to be made throughout the week. If I could start again, a better understand of oneself would be taking into consideration and what actually could be done within the allocated time available. Allocating more time than less is better as learned through the process.

Furthermore, as front-end development was something that I enjoyed outside of University, the thought process initially had was that the front end would be completed a lot quicker than expected. Countless time was spent on attempting to get difficult navigation such as navigation drawer to be implemented which then changed to a simple box navigation. In the future, the foundation will be built and then built modularly on top of each other. This will give more time to think about the process, what is important and how the UX should flow.

Design Quality

Design Quality

The app currently follows Material design principles to ensure it is clean, simple and familiar to the user. This makes it understandable to the user as apps like Google Now and Google Calendar follow Material design. The documentation for Material Design was followed to ensure it meets the requirements. Elements such as typography, colour, spacing, icons and imagery were read and adhered to.

Shneiderman's eight golden rules of Interface Design was consistent throughout the application. Some of the principles it follows are as follows:

Strive for Consistency

The app follows a consistent design throughout with CardViews displaying data and dialogs being used to add, delete or make changes. Floating action buttons are used throughout the app to add items making it consistent.

Offer Informative Feedback

It's important for feedback to be given back to the user once there is a system change. The app makes use of toasts to notify the user of any changes. If the system makes a change and it is viewable change e.g. adding a workout, the user stays is redirected to the relevant page with the new update.

Offer Simple Error Handling

If a user makes a mistake on the app then error messages are displayed clearly and they are unable to progress. Reasons as to why the error is shown are also displayed.

Future Improvements

There are a number of improvements that the app could do with some time and effort being placed on it in the future. The improvements are split into two areas: design and features.

Design Improvements
The app could do with a makeover with its design to make it more user-friendly and simpler. Firstly, the navigation as it currently stands is very linear. Once a user goes into a section, the only way to navigate to another section is by going back. This could be improved by implementing either a navigation drawer or a bottom navigation. A navigation drawer is where an icon in the left top corner of the toolbar opens a 'drawer.' This enables a menu to be displayed off all the areas in an app making it easier to navigate through. Apps such as MyFitnessPal utilise the navigation drawer. The second type of navigation that could be implemented is the bottom navigation. An example of this is Instagram. This is simple to navigate around because it is located near the thumb making it easier on larger devices.

The workout page currently has 4 activities to complete a task from adding a workout to adding a set. This can be quite tedious to do after a while: going back and forth for simple tasks. It can be improved by having it all within 2 activities. Instead of creating a 'workout' users will instead be able to add workouts to a set date in the calendar. There will be a button to add exercises and the sets of the exercise will be displayed under the exercise instead of a separate activity. This makes it an easier to add, delete and visualise.

The progress photos activity currently displays users the photos in landscape. After doing research into types of photos taken, it was found that progress photos are primarily portrait. The layout could be improved so that it displays images accordingly to how it was uploaded.

There are no indicators of how to delete or make updates unless the user actually interacts with the CardView. This could be improved by utilising icons which perform the action. This will improve interaction and user interaction with the app.

Features Improvements
Due to lack of time the feature for users to visualise their body weight progress in a graph was not implemented. Having the ability to visualise the progress in a line graph within a preferred time frame gives the user a lot of feedback and the ability to make the relevant changes.

The photos activity currently only displays the image that they uploaded within the space given. A future improvement could be to click on the image and it opens it enlarged. This will make it easier to visualise the photo. Furthermore, another

feature that could be added is allowing the user to swipe through the images once enlarged making it easier to scroll and see the before and after. Another feature that could be added the photo activity is to filter the images based on the side they took so they only see that side giving them the opportunity to see before and afters easier.

The search functionality currently is very basic as Firebase does not provide client-side filtering. This could be improved in the future by implementing Elastic Search. There is a GitHub library (Flashlight) which is a pluggable integration with Elastic Search to provide advance content searches in Firebase. The script is able to monitor multiple Firebase paths and index the data in real time.

The exercise collection currently has 81 exercise which is practically nothing. The exercises, however, were collected online from reliable sources. A feature could be where users send in a request to add exercises by filling out a form with the details of the exercise. This makes it easier to add popular and relevant exercises based on user interest.

What Was Learnt

Spending the majority of a semester on developing an Android app provided technical knowledge and skills that could be worked on for the future. The two majors areas of what was learned are technical and soft skills.

Technical Skills

Mobile Development

Android uses Java are the primary back-end language for development. This meant having to learn Java but adapting it to the needs of Android. XML is the language that is used by Android for the front-end. This meant that XML also had to be learned to be able to develop a clean and simple user interface. Learning how to develop apps on Udacity gave the knowledge in both Java and XML. Being able to use different views and interact with them in Java with adapters and Firebase was complex but over time with practice became easy to understand.

Courses Taken

https://www.udacity.com/course/android-development-for-beginners--ud837
https://www.udacity.com/course/android-basics-multiscreen-apps--ud839

Firebase

It was important to fully understand how to work with Firebase and the services that it provides. The first part of Firebase that had to be taught was Authentication. Google Sign In was taught from a course so it felt best to

implement email and Facebook knowledge for further knowledge. Firebase Realtime Database was learned next. After doing research, it was apparent that knowing how to model the data in Firebase was important. Reading over the documentation: adding, displaying, updating and deleting was learned. Firebase Storage was needed to store progress photos whilst linking it to the Realtime Database. This was researched upon and the application is now able to store photos on Firebase Storage.

Courses Taken

https://www.udacity.com/course/firebase-in-a-weekend-by-google-android--ud0352
https://www.udemy.com/become-a-beast-android-developer-firebase-necessities/learn/
https://www.udemy.com/build-an-advance-android-app/

NFC

One of the biggest worries that were faced was utilising NFC into the application as there was no up to date tutorials online on how to implement it and reading the NFC guide on Android didn't really make any sense. After finding the library and reading through the Android guide, NFC was finally working with the app. As NFC is currently quite an old technology, it was difficult to find the correct information but learning about NFC, it's capabilities and integrating it into the application meant the app can connect with the physical realm which was one of the goals set out.

Soft Skills

Communication

Working on a project as an individual does come with some issues as you have no one to turn to when it comes to helping on programming, architecture or design. Having a supervisor always available at your need and meeting every week was important to give updates, advice, and security on the thoughts and ideas had. This meant communication with the supervisor effectively. Learning how important it is to ensure emails are sent and booking the meetings was key in progressing. Whilst in those meeting, being honest about progress and help needed was important otherwise information wouldn't be fed back. Learning this meant opening up to the supervisor about delays in the plan.

Time Management

Although 400 hours were allocating for the project, managing the time effectively was key in the completion of the project. Prior to the project, smaller projects were undergone, but working on a single project individually was not done. Having a plan already set out meant the time was allocated better. There were changes that occurred throughout the project, such as switching to Firebase meant time had to be spent on learning it. Making those adjustments and ensuring that there was enough time to complete the given requirements was important.

Future Work

The project has a lot of potentials to be developed into something great to help fitness enthusiastic. There are two areas that have been looked into: implementations the author would make and potential community project.

Future implementation by Author

Currently, the name and email of the user that signs in are displayed on the homepage but there is no profile page for the user. A feature that could be implemented in the future is allowing the user to have a profile page so the app is personalised to them. Taking it personal information from the user means recommendations can be made for the user to make changes and improvements.

The current application allows the user to track workouts but another popular aspect of working out is having a set training plan. Having a feature where users can create training plans and set them to be added on a specific date will give them the ability to use the app for a longer period and track their progress better. A notification could also be set up so that it sends a message to the user to remind them to workout.

There are currently a number of apps that users are using for specific areas such as MyFitnessPal for tracking food and calories and FitBit to track sleep, steps and heart rate. All this information could be taken into the application and provide information back to the user. For example, when a workout is working out, the FitBit could track the heart rate of the user through the workout giving an analysis of the user's heart rate through the workout.

When a person is training they will have a rest period between sets which is a set time e.g. 60 seconds or 120 seconds. Having a timer that they could set up in app means they don't need to navigate to a third party app making it easier and more user-friendly.

Community Project

The project does have the potential to be a community project as there are many gym goers who would love to track their progress and have one single app that 'does it all.' Making it open-source means that the community can check the code giving it fewer bugs and implement features that others may not have thought about. If the project was to be open-source, it is key to give information on how each of the functionality is to perform. Giving the community information on the research of how user interface, specific features and how they operate will give them a better idea on what and how to implement.

Conclusion

In conclusion, it can be reasonable to say that it is feasible to develop a fully functioning mobile application in Android with Firebase. With little to no experience in mobile development or Firebase and with some patience, anything is possible.

The project was set to out to look to help users in their fitness life by developing a system with simple and easy user experience for tracking workouts, photos, and body weight. The completed project is able to perform all the functions that were set out for a database to do including insertion, updating, deletion and displaying.

It was important for the project to have some type of feedback from users and although what was initially set out wasn't done, the feedback taken was positive and beneficial. The feedback given provided a number of possible future implementation which is understandable giving the app is only its first version.

Within the span of a week and small social media marketing, over 15 people had downloaded the application with 5 specific users for feedback. This shows that there is a need for this application as only gym goers can only download this app for it to be of any benefit.

Despite the fact that the app didn't perform the features as well as they could have, and the user experience being linear, the feedback taken shows that there is room for future changes and implementation with time and effort. Furthermore, the feedback taken from the users showed that navigation wasn't difficult contradicting the initial thoughts that I had. This is why feedback was essential to see how users felt about the app. If more time was available, changes would have been implemented and more feedback would have been gotten.

The sole purpose of the project was to help gym-goers and to learn how to develop a mobile application with Firebase in minimal time. This was completed within the time frame had and a fully functioning app is on the Play Store with updates having been made and user download. This was an unexpected thought half way through the project, however, with patience and time the project was completed.

Bibliography

[1] Anon 2017. UK gym memberships reach record high [Online]. Available at: http://www.healthclubmanagement.co.uk/health-club-management-news/latest-news/324058 [Accessed: 5 May 2017].

[2] Anon 2017. Feature Driven Development (FDD) and Agile Modeling [Online]. Available at: http://agilemodeling.com/essays/fdd.htm [Accessed: 5 May 2017].

[4] Anon 2017. The Best Health And Fitness Apps [Online]. Available at: http://www.coachmag.co.uk/fitness-technology/4226/the-best-health-and-fitness-apps [Accessed: 5 May 2017].

[8] Anon 2017. Online Courses - Anytime, Anywhere | Udemy [Online]. Available at: https://www.udemy.com/courses/ [Accessed: 5 May 2017].

[10] Anon 2017. Picasso [Online]. Available at: http://square.github.io/picasso/ [Accessed: 5 May 2017].

[11] Anon 2017. greenrobot/EventBus [Online]. Available at: https://github.com/greenrobot/EventBus [Accessed: 5 May 2017].

[13] Anon 2017. Introduction - Material design - Material design guidelines [Online]. Available at: https://material.io/guidelines/ [Accessed: 5 May 2017].

[17] Anon 2017. A Guide to Android RecyclerView and CardView | Java [Online]. Available at: https://www.binpress.com/tutorial/android-l-recyclerview-and-cardview-tutorial/156 [Accessed: 5 May 2017].

[18] Anon 2017. YouTube Android Player API | YouTube Android Player API | Google Developers [Online]. Available at: https://developers.google.com/youtube/android/player/ [Accessed: 5 May 2017].

[19] Anon 2017. Upload an app - Play Console Help [Online]. Available at: https://support.google.com/googleplay/android-developer/answer/113469?hl=en-GB [Accessed: 5 May 2017].

[15] Anon 2017. Firebase Data Modeling – How To Firebase [Online]. Available at: https://howtofirebase.com/firebase-data-modeling-939585ade7f4 [Accessed: 5 May 2017].

[12] Anon 2017. Butter Knife [Online]. Available at: http://jakewharton.github.io/butterknife/ [Accessed: 5 May 2017].

[14] Drawer, C. 2017. Creating a Navigation Drawer | Android Developers [Online]. Available at: https://developer.android.com/training/implementing-navigation/nav-drawer.html [Accessed: 5 May 2017].

[7] Eclipse Foundation, I. 2017. Eclipse - The Eclipse Foundation open source community website. [Online]. Available at: http://www.eclipse.org/ [Accessed: 5 May 2017].

[6] Faulkner, C. 2017. What is NFC? Everything you need to know [Online]. Available at: http://www.techradar.com/news/phone-and-communications/what-is-nfc-and-why-is-it-in-your-phone-948410 [Accessed: 5 May 2017].

[9] Geider, R. 2017. REST API [Online]. Available at: https://wger.de/en/software/api [Accessed: 5 May 2017].

[3] Leswing, K. 2017. Gigaom | Under Armour buys MyFitnessPal for $475M [Online]. Available at: https://gigaom.com/2015/02/04/under-armour-buys-myfitnesspal-for-475-million/ [Accessed: 5 May 2017].

[5] Quesnel, C. 2017. Press Release: 2016 State of the UK Fitness Industry Report [Online]. Available at: http://www.leisuredb.com/blog/2016/5/11/press-release-2016-state-of-the-uk-fitness-industry-report [Accessed: 5 May 2017].

[16] Variants, C. 2017. Configure Build Variants | Android Studio [Online]. Available at: https://developer.android.com/studio/build/build-variants.html [Accessed: 5 May 2017].

List of Figures

[1] App Story Board
[2] Mockup 1 - Log In/Register
[3] Mockup 2 - Navigation/View Workouts
[4] Mockup 3 - Individual Workout/Add Exercise
[5] Mockup 4 - Set for the corresponding exercise
[6] Mockup 5 - Your Friends/Add User to Your Friends
[7] Mockup 6 - Exercise Description
[8] Mockup 7 - View Photos/Add Photo
[9] App Story Board - Mock Ups
[10] System Architecture
[11] UML Database Diagram
[12] Back Stack Model - Registration
[13] Back Stack Model - Workout
[14] Workout Data Flow Diagram
[15] Progress Photo Data Flow Diagram
[16] Structure of Android Studio Java Classes
[17] Workouts Collection in Firebase
[18] Added Exercises Collection in Firebase
[19] Users in Firebase Authentication (downloaded and signed up)
[20] User Collection in Firebase Linked to Firebase Authentication
[21] Crash Report from Developer Play Store Account
[22] Installs by User Graph from Developer Play Store Account
[23] Active Users from Firebase Analytics
[24] App Reviews from Play Store