UNIVERSITY OF PORTSMOUTH

Workout Logging and Planning Mobile Application

By Michael Verdon - UP959856

Supervised by Dr. Matthew Poole

Final Year Engineering Project: M21287

May 2023

Abstract

Fitness is an ongoing journey for many people with many others intending on starting. Resistance training is considered one of the main pillars of fitness which unlike other forms of exercising, logging and planning is widely considered vital in helping to apply a key concept called “Progressive Overload”. Progressive overload can be defined as increasing training volume over time to allow you to progress and get stronger. This can involve lifting heavier, increasing repetitions and sets of an exercise or decreasing rest time. Many applications exist that help users log and plan workouts but very few help you to apply progressive overload.

This project document presents the production of a mobile application that will help users plan and log workouts as well as aiding the user in applying progressive overload through a high usability user interface.

Acknowledgments

The author wishes to thank Dr. Matthew Poole for agreeing to supervise this project and providing good constructive feedback which helped shape the app into what it is today.

Table of Contents

List of Figures

1. Introduction

This project aims to create a weightlifting planning and logging mobile application to make resistance training more accessible to anyone regardless of gym or equipment access and aid in applying progressive overload. This document will cover the motivation behind this application and the development process.

This chapter will introduce the problem. Firstly, in Section 1.1, the background of the issue will be discussed as the motivation for the application. Secondly, in Section 1.2, the aims of the project will be discussed along with the risks and constraints. Finally, in Section 1.3, the document structure will be presented.

1.1 Background

Fitness is an endeavour for all kinds of people whether it be cardio or resistance training and vice versa. Cardio can be as simple as going for daily morning runs. However, resistance training is divided into multiple categories. For example, there is classic bodybuilding which utilises lighter weights and greater volume which uses a greater number of repetitions a lift. Also, there is powerlifting which utilises heavy weights and high intensity for the core aim of building strength as opposed to muscle. Additionally, there is callisthenics which only uses body weight for resistance and many more. These are the sections of training this project aims to complement.

Many fitness applications exist but solely focus on either home workouts, the gym or are part of a gym's application but not everyone has access to a gym or equipment for several reasons whether it be financial or living location. Whatever resistance training category one chooses to approach, it is important to log progress and set goals for future workout sessions. This is because the key to growth in strength and muscle is 'progressive overload'. This involves increasing the

volume of training proportional to the time spent training to provide the body with new stimuli to adapt to and influence metabolic and hormonal responses (Peterson et al., 2010). This can be done by lifting heavier weights, decreasing rest time between sets and increasing the number of sets and repetitions week by week and month by month in your routine.

The internet is full of conflicting information on how one should work out which can inundate beginners making starting a struggle. A well-planned application could give beginners a plan to start as well as log their progress and set goals to implement progressive overload without the requirement of knowledge and research on this topic.

1.2 Aims

This project aims to create a workout planning and logging mobile application to make weightlifting and general working out beginner friendly and accessible for anyone regardless of financial situation, where they live and their level of fitness. Furthermore, this application will allow users to apply progressive overload without any knowledge of how to do it.

1.2.1 Criteria

For this project to be successful, the following tasks must be performed:

- Research the impact of mobile applications on fitness.
- Assess current implementations of workout apps.
- Examine software engineering methodologies to manage software development.
- Create specifications requirements and design documents for the application.

- Thorough testing of the software to ensure robustness and quality management.

1.2.2 Risks

To minimise the risks, the application will operate entirely locally on the user's device. This will ensure that credentials will not be compromised and security will not be much of an issue.

Because the application is a solo team project and the timescale is quite narrow, not all initially planned features may be implemented. As well as this, there is also required to learn new technologies such as Flutter and Dart for mobile application development. Moreover, there could be technical issues along the way with technology being used for development which could temporarily bottleneck progress.

There are many ways to approach creating a weightlifting app and it is vital to prevent scope creep from occurring in this solo project, so features must be ranked by priority to be able to reach a minimum viable product.

1.3 Structure of Document

This document consists of eight further chapters which are described here.

Chapter 2 is a review of relevant literature regarding the topic of mobile applications and fitness. Later on, current implementations will be evaluated.

Chapter 3 will discuss the software engineering model chosen and the management of the application development process.

Chapter 4 provides a software specification and user requirements based on existing implementations and new ideas.

Chapter 5 covers the original design of the application and how features and functionality can be represented in a high-usability interface.

Chapter 6 conveys the choice in technology and the incremental development process with challenges overcome and testing.

Chapter 7 evaluates the project and the artefact created.

Chapter 8 will conclude the project and explore future work that can be done on the application to improve it for mass usage.

2. Literature Review

This chapter is a review of the literature about gamifying fitness in mobile applications and current implementations of fitness applications. We begin in Section 2.1 which discusses the gamification of fitness in mobile applications. Then in Section 2.2, we review current workout logging and planning applications comparing the strengths and weaknesses and identifying the commonalities of them.

There are many ways in which mobile applications make the user experience rewarding and motivating. One of which is gamification which is what this literature review will analyse and how effective it is in increasing physical activity.

This literature review will begin by discussing the background of gamification in mobile apps and its history, the theoretical framework of gamification in systems, then how it is implemented in fitness applications, its impact on user engagement, and finally, the effects on the user experience.

2.1 Gamification of Fitness in Mobile Applications

Gamification is defined as adding game-like elements and mechanics to a non-game environment to increase user participation to boost productivity. This review will primarily focus on how it is implemented in fitness apps.

2.2.1 Background

The ideology of gamification has existed for a long time. A key element of it is a sense of feeling like you are playing. Playing can be defined as a free activity that is separated from ordinary life, however, it absorbs the participants in it (Caillois,

2001). The act of playing is a very human activity and has been used in many applications to increase engagement in an activity or a product on the market.

An early example of gamification is the Boy Scouts of America. The purpose of this movement is to teach children important life skills and one way that has been achieved is through earning badges. This is done by consulting their unit leader about a topic and the requirements for a badge in it, then undertaking the required steps to achieve it (Boy Scouts of America, n.d.). The more badges earned, the more badges they can pin onto their clothes to convey their achievements. Earning numerous badges individually instils a sense of progression reminiscent of completing levels in a video game or building a collection of trophies. This in turn can encourage boy scouts to carry on earning more badges and hence learning more life skills.

Generally, systems are function-focused which involves getting a job done with the objective of efficiency. This however does not account for the user's emotions and motivations. This is why Gamification is important in modern software applications as it can be considered a human-focused design to make it fun for the user and keep them entertained (Chou, 2015).

Fitness for most people may revolve around keeping motivated and having a reason to keep dedicating time and effort to maintaining a particular lifestyle. The use of gamification can be used to help fitness app users get more motivated to hit their fitness aspirations and keep consistent.

2.2.2 Theoretical Framework

To successfully implement gamification within any system, a framework is followed. Many gamification frameworks exist to guide developers in making more engaging applications. Each one of them is usually focused on a particular area such as business, healthcare and so forth.

The Octalysis Framework is a well-renowned general-purpose gamification framework that is divided into eight cores. It is modelled as an octagon that motivates human behaviour and was developed by Yukai Chou. An example of a section is ‘Ownership’ which is intended to motivate users by making them feel they own something. This can be in the form of set collections, virtual goods and a customizable avatar (Chou, 2015). A user creating an avatar from scratch and taking time into it can give a sense of ownership to the user. Another section is ‘Social Influence and Relatedness’, which revolves around social influence such as seeing a friend's statistics could make a user strive to be on the same level and encourage competition (Chou, 2015). A successful app using this framework does not have to achieve all eight cores in it but performs well on the ones it does apply.

Another framework made for video games, but can be applied to other implementations is the MDA framework. MDA formulates how games are consumed by breaking them down into their core components along with their design counterparts. It looks at the rules, system and fun in a game along with the mechanics, dynamics and aesthetics respectively (Hunicke et al., 2004). The mechanics refer to the rules of how the gamified system operates including what you can do in the game, what is in the game, what abilities you have and the constraints. Dynamics go into how the player interacts with the gamified system and what challenges are there. Finally, the aesthetics discuss what makes a game fun and what emotion it makes the player feel as well as the appearance of the system (Hunicke et al., 2004). Concepts from this can be used to gamify applications that aren’t inherently games.

2.2.3 Gamification of Fitness in Applications

There are many ways in which gamification can be applied to fitness apps to help keep users motivated on their goals. Many existing applications achieve this through several different methods with most applications having commonalities in their implementations.

The use of counters are commonly used in fitness applications, these can be step or calorie-burned counters. It helps a user visualise their physical activity with a number displayed. This feature can be taken further by implementing a leaderboard and challenges for users to undertake for social impact. Modern smartphones have built-in pedometers which are also used to estimate activity levels. An example of this is Apple's iPhones which use the 'Health' application built-in in the more recent models to track numerous categories including steps done and calories burned along with graphical views (Apple, nd). Another common example is smartwatches that usually have smartphone integration such as the Fitbit app. These devices also track steps and calories as well as other categories such as sleep (Fitbit Inc., 2021). A meta-analysis of studies on physical activity and pedometers has shown that step counters can have a moderate positive effect on a user's activity level. Groups with a 10,000-step-a-day goal have seen greater improvements in the amount of physical activity (Kaang et al., 2013). This would imply having a goal along with this feature makes it more effective. However, step counters can have varying degrees of accuracy which is why using a watch-based pedometer may be required for accurate readings. These watch-based pedometers are usually inexpensive anyways which makes getting started with one easier.

As well as using counters to increase steps done in a day, some applications that are not fitness apps gamify fitness such as games where physical activity is required to play. Pokemon Go is an application that was released in 2016 that garnered a large player base having 100+ million downloads and still ranks fairly high on the Play Store (Niantic, Inc., 2023). The objective of the game is to capture monsters scattered everywhere in the world by walking around to find them. The user is also presented with a menu of particular monsters of interest along with their distance encouraging users to walk to specific areas that can sometimes be far away. This game also allows you to battle other players and take control of gyms giving it a competitive element or team up to catch legendary monsters. It has been shown in a large-scale study that Pokemon Go had drastically increased activity levels by

25% or 1473 steps average in people across all age groups and especially in low-activity populations. Users who had the greatest increases also searched queries about the game frequently on how to achieve certain objectives (Althoff et al., 2016).

Many applications use a reward system in conjunction with activity counting. For example, unlockable trophies and badges for completing challenges. This technique of gamification can give the user a greater sense of achievement when completing their fitness routines and planning goals. As well as apps like Pokemon Go incorporating this, PlayFitt is a fitness application that allows users to use power-ups and earn coins upon task completion. Along with this, the app has a collection of daily goals for the user to complete like ticking off challenges (Guillemette et al. 2022). The use of a rewarding or trophy system is part of the Octalysis framework where emphasis is placed on challenging the user to overcome tasks to make progress towards their fitness goals (Chou, 2015). It has been shown that reward systems are a powerful tool for motivating people from an experiment analysing the performance of employees and job satisfaction from differing motivational methods. Reward systems accounted for 55% of motivation changes as well as leading to a considerable increase in worker performance (Kharami et al., 2013).

All of the features above are usually synergised with a leaderboard system. Most of the already mentioned applications have some form of leaderboards as well as social media sharing to entice competition and comparison among users adding social motivation to fitness. Most commonly, step count is used for this and step counts per week. Respondents in a survey mostly preferred a leaderboard system for fitness and wished to see comparisons with their friends or colleagues as opposed to strangers. This is because comparing with strangers could lead to an overly competitive environment as some users might be more serious about physical activity than others which could put off current users of the application.

When asked about their enjoyment of apps using leaderboard systems, most reported enjoying the apps more, even if they were at the bottom of the leaderboard and increased motivation for usage (Jia et al., 2017).

2.2.4 User Engagement and Challenges

Most gamification methods used in fitness appear effective in the short term for increasing user engagement and activity, however, these methods can have varying degrees of success. Personality can heavily influence what gamification methods are more effective for each individual as not everyone's the same. While more people preferred a leaderboard system than not, it was also noted that the majority of these people consider themself to be extraverted as opposed to the respondents who did not prefer it (Jia et al., 2017). Furthermore, as mentioned earlier, applications like Pokemon Go were effective with more users who would prefer to stay at home more often (Althoff et al., 2016). Considering this, it could be implied that a general solution does not always work for keeping users engaged but each application can be designed for a certain demographic such as Pokemon Go for fans of the Pokemon franchise.

A small-scale study looked into the sustainability of gamified fitness apps and if they keep users engaged and found that users were more consistent in using applications that had more gamification features than fewer. The same application was given to all participants with some receiving a lower-level version of the app to be updated gradually and others the full build. As the participants with the full build kept engaged would imply that the combination of gamification techniques is required for the application to be more successful in user retention.

2.2.5 Conclusions and Future Work

Overall, it can be concluded that gamifying fitness in mobile applications can be an effective way to increase the activity levels of users of all categories. However, the sustainability of this can vary among individuals and their personalities. Some

techniques in general were more effective than others. Most applications combine step counting with other game-like features to gamify fitness for its users in a variety of ways as step count is a simple method of tracking physical activity.

Further analysis of the demographics of applications can be looked at to see which gamification techniques are more effective for certain populations which could be clustered by machine learning. This could in turn allow developers to create applications much more focused on a target audience to increase user retention and enjoyment.

2.2 Existing Systems

There are a number of existing mobile applications that can be used as tools for working out and helping users get started on their fitness journey. Both examples discussed below are all available on the google play store for Android platforms.

2.2.1 Home Workout - No Equipment

The “Home Workout - No Equipment” application (Leap Fitness Group, 2023) is a workout planning and logging application with over 100+ million downloads and a 4.9-star rating on the google play store. It aims to allow users to workout without having to worry about equipment making it very accessible for anyone as long as they have a phone and a space to workout. User reviews remark on the app being simple and easy to use and giving them a real sense of progression. Due to the app's popularity and high reviews, it could allude to what is required from a workout planning and logging application.

The main features of this application are:

- Save user metrics.
- Providing an individual workout plan according to the experience level and goals if a questionnaire is done.

- Choosing preset workouts based on experience and what muscles the user would like to focus on.
- Guiding the user through selected workouts by providing animations alongside the user exercising and setting timers for each exercise.
- Allowing the user to set goals such as workout days in a week or month.
- Gives the user graphical reports on progress such as weight tracking.
- Tracking calories burned from working out.
- A ‘discover’ page that allows users to check out even more workouts designed for other goals if a premium price is paid.
- Challenges users can partake in and complete.
- Settings for preferences such as ‘rest time’.

One of the core strengths of this app is accessibility, allowing users of all experience levels to find a workout catered to their needs without needing equipment. Also, the guiding system acts as a coach to the user allowing them to not have to worry about timing their sets in their workouts. Finally, the application has a very easy-to-navigate interface making it easy to use

However, this application does not allow users to create their own workouts. In addition, the current workout routines provided cannot be altered besides altering the rest time in the settings in between not giving the user freedom to alter the workouts if they need to.

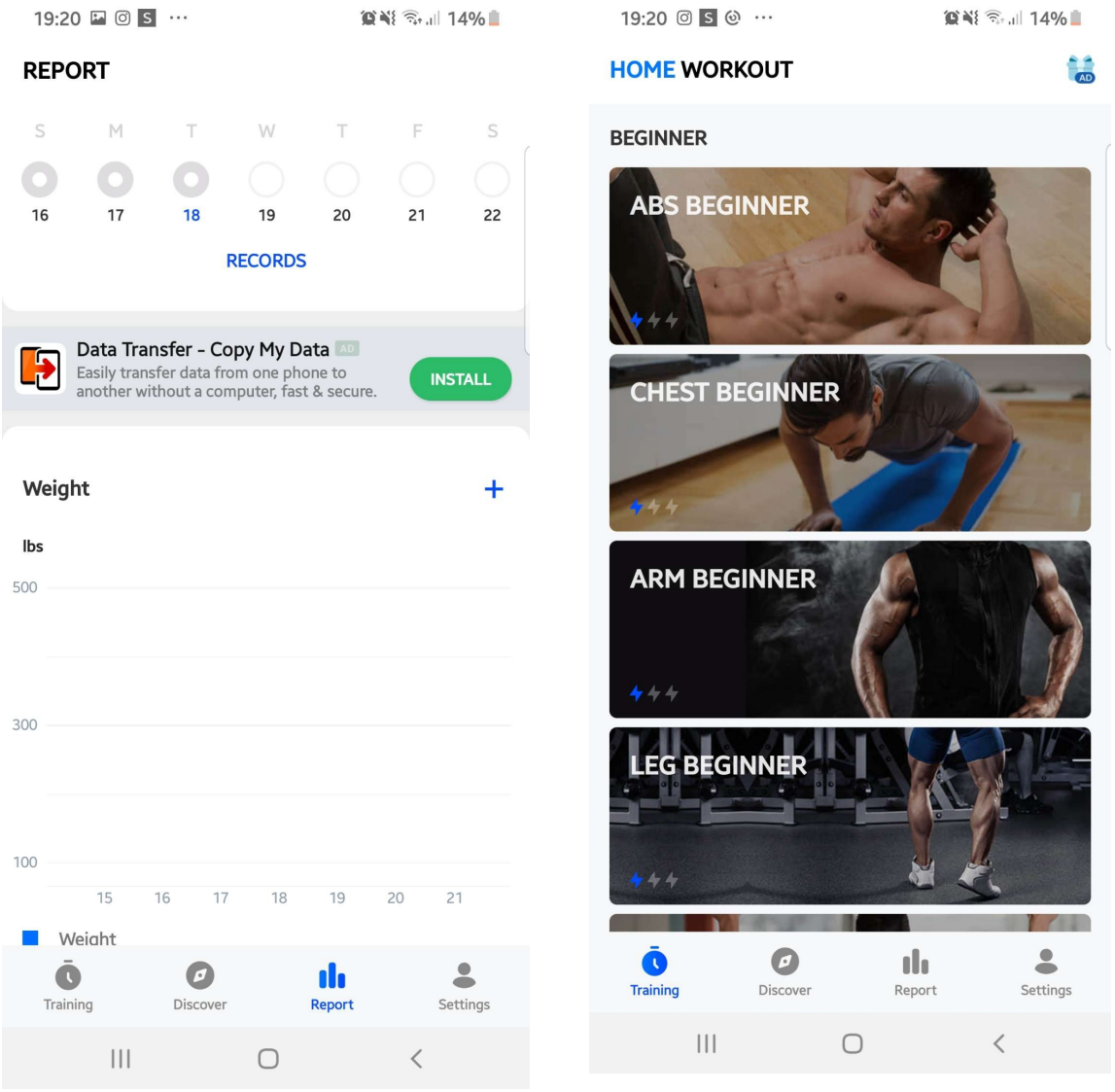

Figure 2.1 (Home Workout - No Equipment App)

2.2.2 Gym Workout Planner and Tracker

The ‘Gym Workout Planner and Tracker’ application (Fitness 22, 2023) is a workout planning and logging application focusing on gym workouts meaning a gym is required to fully utilise the app. With 1+ million downloads and an average of a 4.6-star rating, it could also be considered a good example of what a workout logging and planning app should be. Several users in the reviews stated they liked how the application could be used regardless of experience level and the tracking

features of the app. The goal of this application is to allow you to be your own personal trainer.

The main features of this application are:

- Store user metrics.
- Creating workouts.
- Set workout days with notification reminders on those days to work out.
- A comprehensive database of exercises to see explanations on them.
- Premium plans allow the user to have assistance from a personal trainer.
- Tips from the coaches section.
- Tailored workout plan from a questionnaire.
- A large number of programs are available to users based on certain goals (losing fat, gaining muscle e.t.c)

This application is very easy to navigate with a comprehensive workout creation system. Within the creation menu is a labelled diagram of muscles which the user can then select and pick exercises and view information about them. This allows the user to narrow down exactly what exercises they may want to incorporate into their own routine. A section is dedicated to advising newcomers to the gym and how they can get started.

On the other end, most features are premium meaning most of the application is not usable unless you pay the fee. Workouts can still be created otherwise. Some users complained that the IOS version of the application is much better than the Android one and some very common barbell exercises are not included in the app.

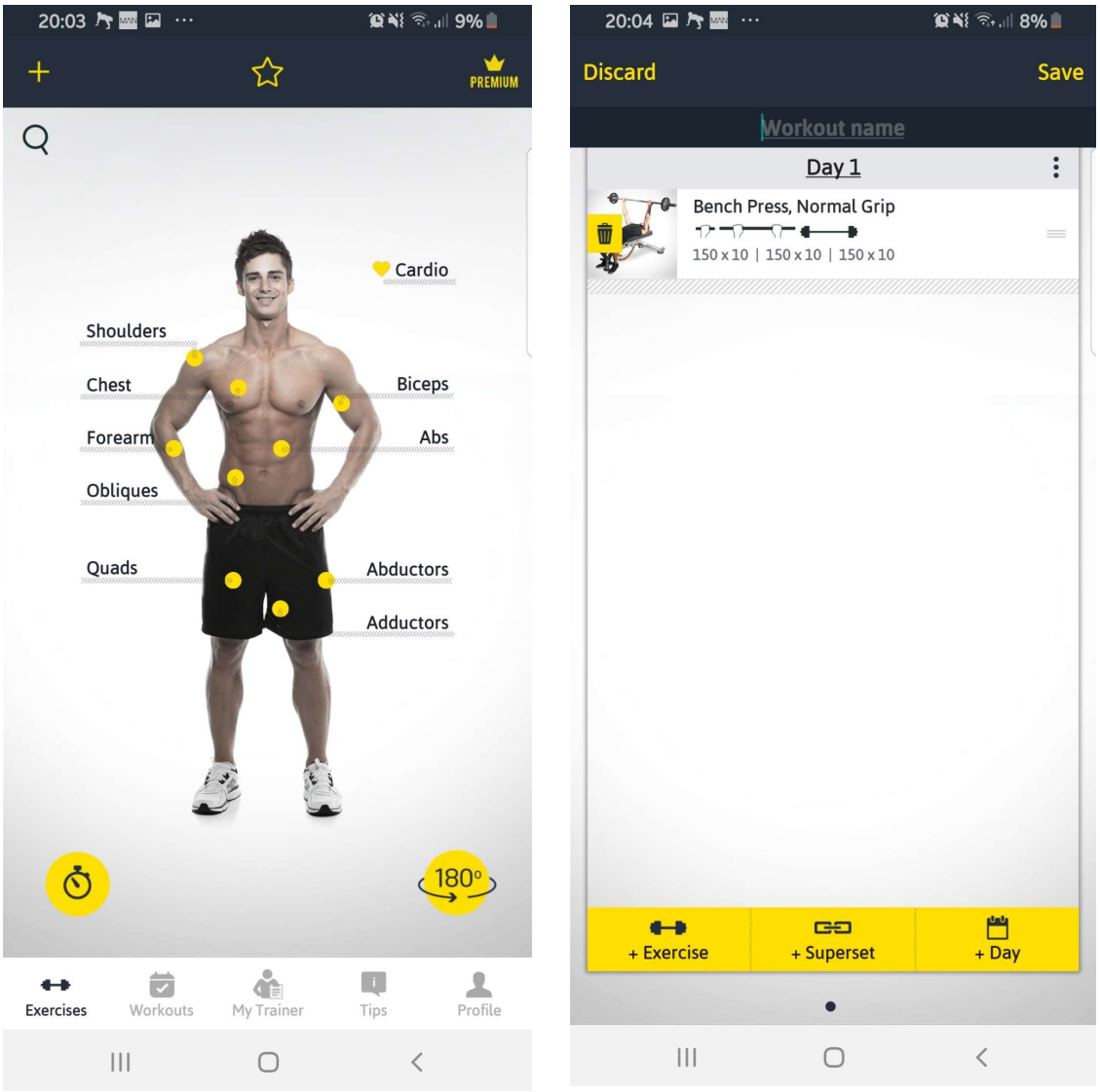

Figure 2.2 (Gym Workout Planner and Tracker)

2.2.3 Comparisons

Both applications have shared features but also differ. Observing those features can be used to derive requirements of a workout planning and logging application. The following table (Figure 2.3) below compares the features in both applications:

Feature	**Home Workout - No Equipment**	**Gym Workout Planner and Tracker**
Create workout	-	✔
View progress	✔	✔
Questionnaire	✔	✔
Store user metrics	✔	✔
Calorie tracking	✔	-
Do existing workouts	✔	✔
Instructional animations	✔	-
Set goals	✔	-
Talk to a personal trainer	-	✔
Challenges	✔	-
User preferences	✔	-
Database of exercises	-	✔

Figure 2.3 (Comparison of existing features in both applications)

From what can be observed, being able to store user metrics, and observe progress and workout are the core of a workout logging and planning application. Other features can be used to gamify the application. However, the application does not help you directly in implementing progressive overload which was already established as a key for growth. This means none of these applications meets the goals of this project but can be used as a guide on how to make the application and what a workout logging and planning app should look like.

Both applications use a navigation bar along with a clear and concise design for the interface conveying high usability. This implies that usability is also a key factor in developing a successful workout logging and planning application. Furthermore, the

use of buttons with pictures on them and icons are prevalent which are eye-catching and make it obvious for the user what clicking them will do.

3. Methodology and Management

Within this chapter, software engineering models will be discussed along with the model chosen. We begin in Section 3.1 which outlines Agile and the inspiration taken from it. Then in Section 3.2, Incremental development is discussed with pros and cons. Following this in Section 3.3, the chosen model is discussed with justification. Finally, Section 3.4 conveys the management of the project.

3.1 Software Development Models

To ensure the successful delivery of the artefact, a Software Development model must be chosen. Within this section, two software engineering models thought to suit the project are discussed with the justification of the approach.

3.1.1 Agile

The Agile Manifesto consists of many principles which could be applied to the development of this software. This development model is relevant and important in modern software development practices for its ability to adapt to changing requirements and work collaboratively with the investors to ensure a high-quality delivery (Principles Behind the Agile Manifesto, 2011).

Some of Agile's core principles include:

- Satisfy the customer through early consistent delivery of software.
- Working software created is the measure of success.
- Minimise the amount of work not done.
- At regular intervals, the team will consider how to become more effective together.
- The optimal way of conveying ideas is through face-to-face meetings.

However, Agile's requirement of regular meetings with the customer is an issue as there is no customer. As a result, Agile standalone cannot be used to its full effectiveness. On the other hand, inspiration can be taken to adapt it for this project.

3.1.2 Incremental Development

This software development model consists of breaking down the features of the software and working on them individually working your way up. Requirements are clearly defined from the beginning and a final result can be visualised. Incremental development is considered to be a capstone for Agile development and was intended to replace the restrictive Waterfall model (Larman & Basili, 2003).

An important factor for considering this development model is that it allows for risks in the design to be identified and handled sooner rather than later. In addition to this, the application can be criticised for flaws more frequently to help boost quality.

3.1.3 Approach

For this project, the development cycle will follow an incremental approach. Requirements will be categorised by priority to ensure a minimum viable product can be achieved and prevent scope creep. Inspiration from the Agile Manifesto will be taken to aim for a prototype as soon as possible and openness to altering requirements. After each prototype, more features can be appended and analysed for improvements whilst other features could be reworked to prevent complications later down the line.

3.2 Management

Efficient project management is important to ensure the delivery of the artefact. It is salient to have a time management plan to guide this project especially as it is an individual project with time constraints.

Figure 3.1 below illustrates the time organisation of the project in the form of a Gantt chart:

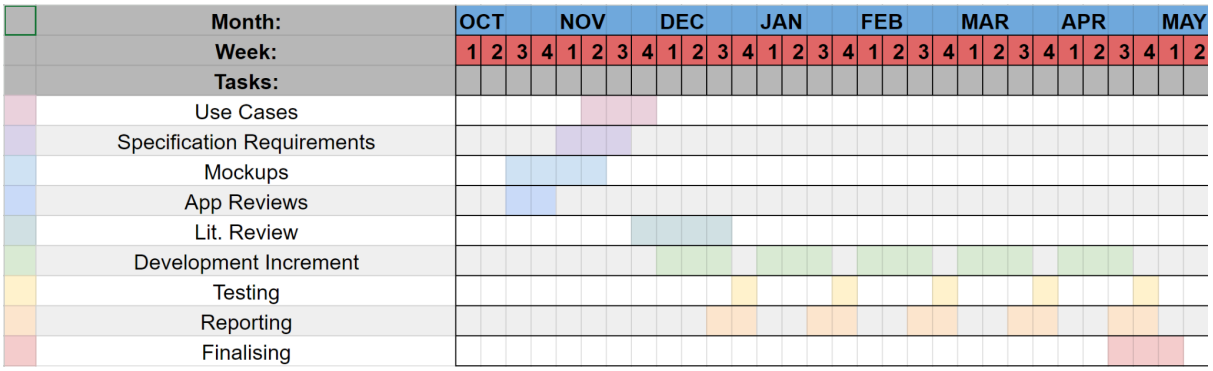

Figure 3.1 (Gantt Chart of Project Timeline)

Designs of the application will be handled sooner to obtain a clear plan before development begins. Over time, the app will be worked on in increments with reporting towards the end of each cycle to ensure both aspects of the project are constantly being worked on.

In the initial stages of development, all time will be allocated to designing the front end of the application as usability is a key focus. Producing this application like this will simplify the backend creation as all the inputs and outputs will be well-defined beforehand and can easily be adapted for altering requirements. Allowing adaptation for changing requirements sooner will help ensure development does not get halted due to restructuring numerous parts of the application as the codebase grows.

All requirements that are considered the highest priority will be completed first before any other ones that are not as important. This will make sure that issues such as scope creep will not prevent a minimal viable product from being achieved.

4. Software Requirements

In this chapter, the requirements of the software will be presented. Firstly in Section 4.1, requirements elicitation is covered as well as features in existing implementations. Then in Section 4.2, Functional and Non-Functional requirements will be outlined and ordered by priorities: “Must have”, “Should have” and “Could have”.

4.1 Requirements Gathering

Requirements for this application were gathered in two ways: Firstly from the Author's personal experience with working out and what he would want in this software. And secondly by taking a look at features from other applications that were analysed in Chapter 2.

4.1.1 Personal Experience

The author has a history of working out and understands the importance of tracking your workouts to progress. This made it clear what features must be included in the app to achieve its goal without needing to reference existing systems. In addition, the application will also aim to be a tool the author could use to complement his workouts which makes visualising requirements easier.

4.1.2 Features of Existing Implementations

The research done in Chapter 2 of existing software helped build an understanding of some of the core features of workout applications. By looking at these apps, It is known what new users would expect if they were to download this application from the Google or Apple marketplace.

The advantages and disadvantages of existing systems were analysed to also allow for improvements to be made on where some applications fall short and see where they achieve.

4.2 Requirements

Within this Section, the functional (F) and non-functional (NF) requirements will be listed, include a description and be organised based on their priority. Every requirement will fall into one of the following categories:

- Must have (MH) - These features are a must and have to be completed for the project to be considered a minimum viable product.
- Should have (SH) - Features within this category would compliment that app very nicely and boost current functionality but failure of completion would not jeopardise the development of the app.
- Could Have (CH) - For the features that could be too ambitious or unimportant to the overall scope of the app.

In Sections 4.2.1 and 4.2.2, the requirements will be put into tables and commence with a code. For example, MH-F-1 means “Must have functional requirement 1” and CH-NF-1 means “Could have non-functional requirement 1”.

4.2.1 Functional Requirements

Functional requirements define what the software must achieve and how it will respond to particular inputs. If functional requirements are not met, the software will fail to work (Uptech team n.d.).

Below is a list of the functional requirements with their respective rankings, feature, description and justification.

MH-F-1 Save User Metrics. User metrics such as age, height, weight and gender will be saved as they are important for determining metabolism. Furthermore, the

weight will be tracked over time upon updates as weight is one of the most common metrics for determining success regarding gains in muscle mass.

MH-F-2 Save Data. Data must be saved not only on the user metrics but on the user's workouts, progress and history locally on the device. This will allow progression to be tracked and goals to be set for the user for each workout according to their performance on their last workout.

MH-F-3 Workout Logging. The system must allow users to choose a workout, input their performance and save goals set from MH-F-4. This will allow users to track and plan their next workout and see where they are at with their exercise routine.

MH-F-4 Workout Goals. After each workout is performed, a goal is set with consideration to the user's performance. This will ensure too high of goals aren't set if the user struggles to meet the goal but will also push the user if appropriate. Furthermore, it will aid in applying the key principle of progressive overload for strength and muscle growth.

MH-F-5 Create Workouts. The user must be able to create workouts with their own defined parameters to allow personalisation in the application as many people have their preferences for how they lift weights. In addition, users will be able to create a workout routine regardless of gym or equipment access.

MH-F-6 Workout Split Generator. Through a short questionnaire, any user will be able to have a personal plan generated for them if they wish regardless of their equipment or gym access. This will also consider their goals and experience level. As a result, this will give the user a plan appropriate for them and allow beginners to start working out without the knowledge of working out.

MH-F-7 Edit Workouts. Some exercises or weights may be more difficult than the user may anticipate so the system must be able to adapt to it automatically or manually. This will ensure any wrong choice in weights for the user will not

compromise the application's functionality and allow the user to always to also delete workouts they wish to no longer do.

MH-F-8 Data on Exercises. Data on the most common exercises and muscles they work on will be readily available for use with workout generation and creation. This will include exercises that do not need equipment to further emphasise making working out more accessible.

SH-F-1 BMR Calculator. BMR will be displayed with the user's details after user metrics are input to give them an idea of what their baseline metabolism is. Having this value dynamically change upon metric updates and be visible at all times will be useful for tracking baseline calories burned and their dietary caloric needs.

SH-F-2 Activity Tracking. The system will track the steps, distance and calories burned from those activities along with calories burned from lifting weights. Calories burned from weight lifting will be calculated with respect to the user's body weight. This will give the user an idea of their activity level day to day.

SH-F-3 Gallery Page. Users will be provided with an interface to upload and organise pictures which are timestamped to allow comparisons in physical progress over time to be made. Progress pictures are a common way people determine their progress and Implementing this could make the application feel more personalised for the user.

SH-F-4 Progress Graphs. A system within the app will allow the generation of graphs based on time periods and the metrics selected to be plotted against each other which will help the user visualise their progress. Metrics used in this include the user's weight, steps, distance travelled and calories burned.

SH-F-5 Featured Workouts. Sample workouts will be accessible to the user and they can choose to add to their list of workouts. These could be fetched from an online database and sample ones built in to give users something to work with if they are unclear of their goals when filling in the workout generator questionnaire.

CH-F-1 Forum. A community-driven forum could build a community around the application and provide a supportive platform for others. This feature would be like another application inside which is separate from the main application and would require account creation and internet access.

CH-F-2 Rep-Visualiser. Taken as inspiration from some modern smart gyms, this feature would give the user visual guides regarding their breathing and pacing repetitions of exercises. This would in turn add some gamification to working out and ideally increase user retention.

CH-F-3 Altimeter. This will track how high the user has travelled. This will be integrated with the calorie tracking system to build on the granularity and accuracy of the results already provided.

CH-F-4 Sleep Tracking. If implemented, it will aid users in tracking the quality of their sleep and suggest improvements. This could be useful because good quality sleep is widely regarded as a key factor for good health and fitness as well as being a requirement for recovery.

4.2.2 Non-Functional Requirements

As opposed to functional requirements, if non-functional requirements are not fulfilled, the software will still work. They are not the foundation of the application. Non-functional requirements are oriented towards the user (Uptech team n.d.).

Below is a list of the non-functional requirements of this application done in the same format as the functional requirements.

MH-NF-1 Usability. The application's user interface must be easily navigable and intuitive to anyone using it. Achieving high usability will make the software appealing to users and make it easy to use.

MH-NF-2 Lightweight Storage. A Lot of information will be stored and it is important it is stored in a way that is lightweight and easy to call. This will also contribute to performance and speed of the application.

MH-NF-3 Portability. The application must be able to perform well on most Android system versions and possibly IOS. This will ensure what phone model and system version you have will not be a factor in whether you can use this app or not.

SH-NF-1 Availability. As the application will not require wifi for its main features, it will always be available but other features may require wifi. If such features are implemented, they must always be available to the user granted a wifi connection is established.

SH-NF-2 Performance. The application's features must respond fast to the user and provide a seamless experience. Moreover, the application should not be prone to crashing and if so, very infrequently to prevent loss of data and corruption.

CH-NF-1 Scalability. If the app becomes popular, ensuring it can be expanded on and further developed will allow for competition with existing applications on the market.

5. Design

This chapter will display the design of the artefact to be created. We begin in Section 5.1 where the software architecture will be mapped out. Moving onto Section 5.2, use case diagrams will be presented. Then in Section 5.3, mockups of the user interface are displayed including older and newer versions with the justification of choices and evaluating them. Finally, in Section 5.4 the use of colour will be discussed.

5.1 Software Architecture

To understand how elements of the application will run a conjunction, a Software Architecture plan was created shown in Figure 5.1:

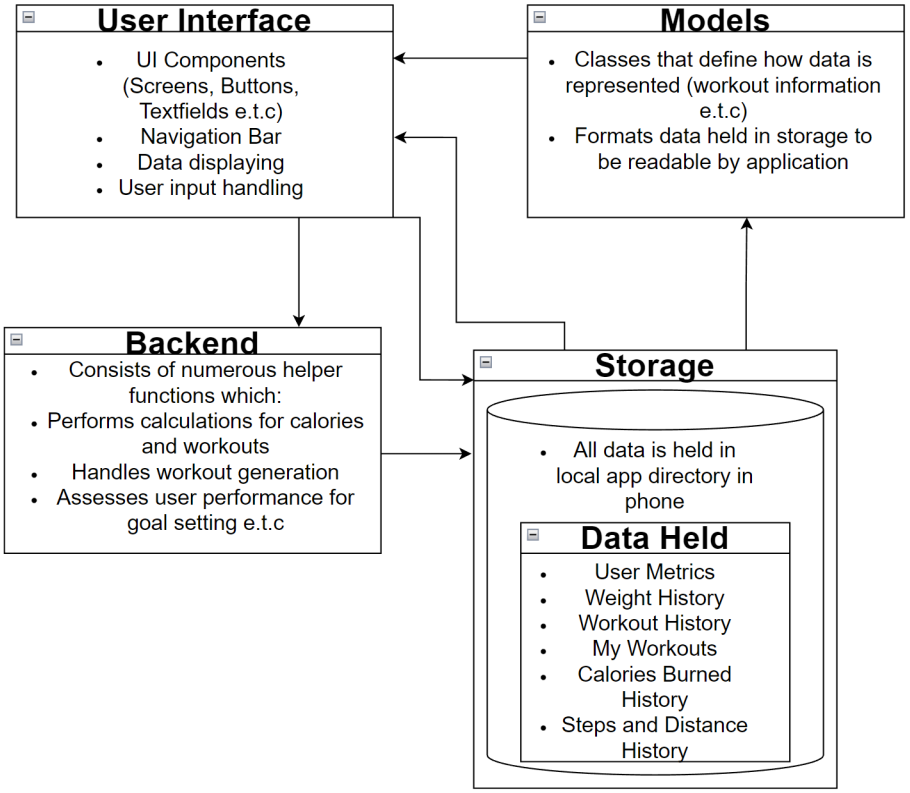

Figure 5.1 (Software Architecture)

Interactions in the application follow a cycle of operations from numerous parts. The UI will handle events and what the user wishes to do. The local storage will fetch data for the input directly or through models for easier manipulation. For example, the user may fetch information on a workout and what needs to be done. After the user finishes a certain activity, the information is processed by the backend and put back into storage.

The models are class objects which will define how data such as exercises and workouts are structured and allow easier access to changing the variables inside.

The proposed storage format will primarily be JSON due to its data structure allowing the keying out of particular variables by a string such as “date” or “calories” as well as its ability to cross over to other data structures in many programming languages. Moreover, it could be useful for saving histories of data as arrays can be created therefore an array of data marked by dates could prove to be a suitable format for the progress and history features.

As the application is going to be locally based, there is no need for communicating with the internet unless the step counting in SH-F-2 requires the use of APIs to achieve its functionality.

5.2 Use Cases

To further understand the flow of the application, use case diagrams were created showing how the user would interface with the system.

5.2.1 Me Page Use Case

Presented in Figure 5.2 is how the user would save their metrics and get their BMR calculated:

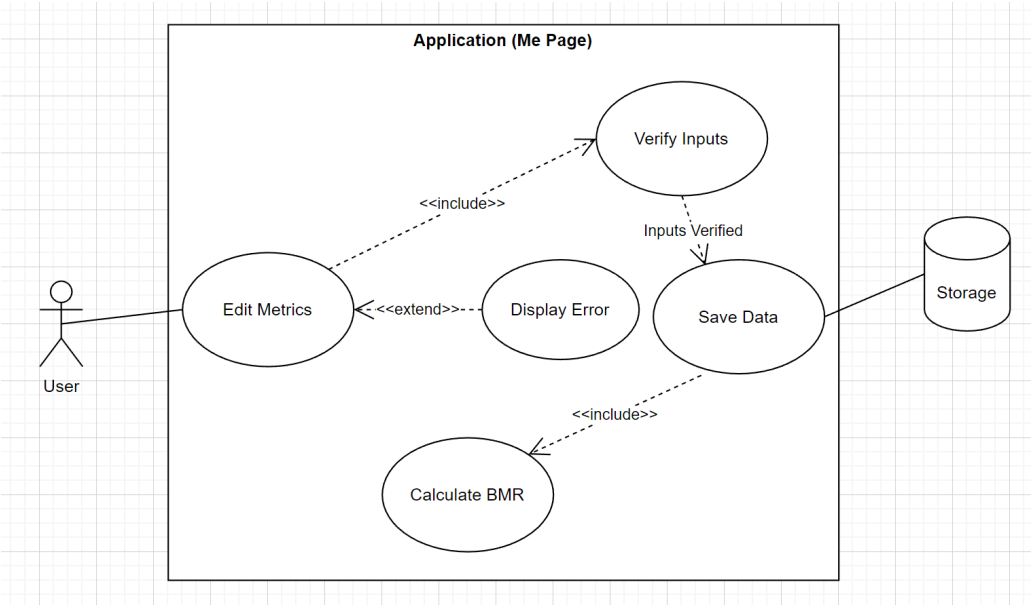

Figure 5.2 (User Metrics Use Case)

Within this interface, the user can edit their metrics. Once they hit save, it will verify the inputs to ensure they are valid and return an error otherwise. Once data is successfully saved, the BMR will be calculated and displayed.

5.2.2 Workouts Use Case

Figure 5.3 demonstrates how the user would interact with the working out system of the application:

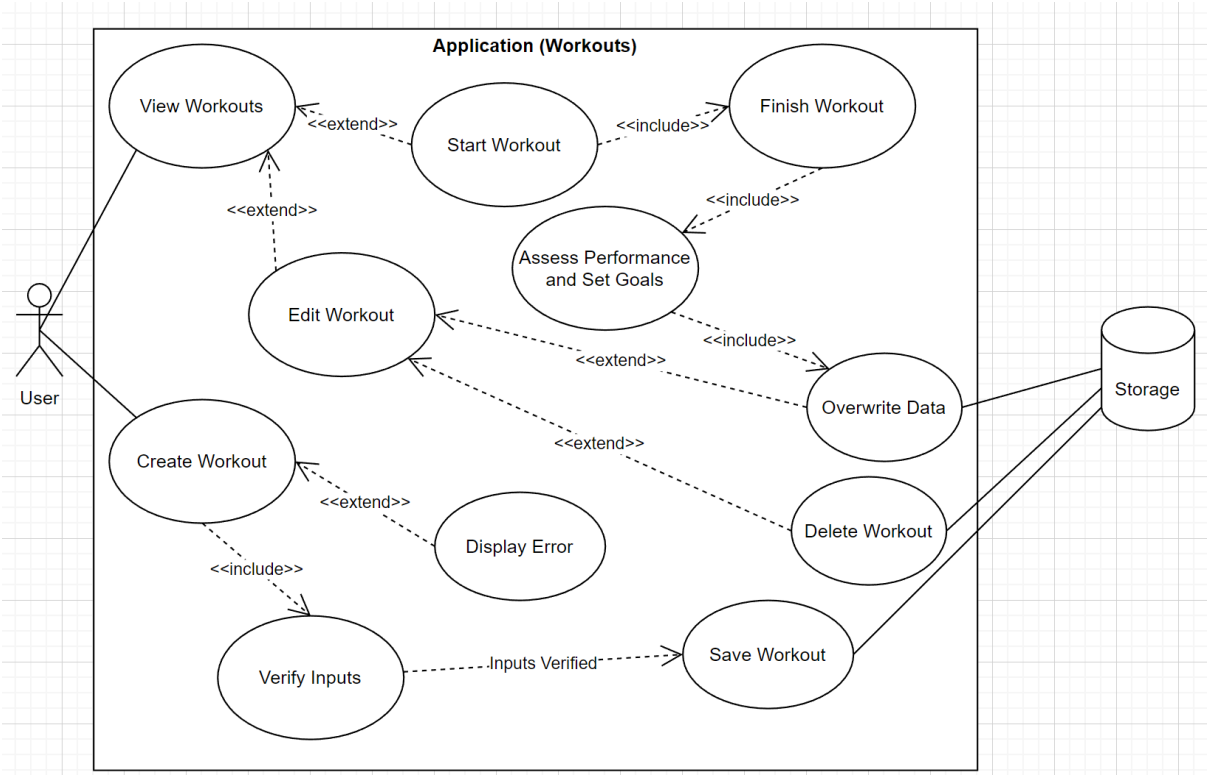

Figure 5.3 (Workouts Use Case)

The workout interface consists of performing workouts and creating/editing workouts. Each time a workout is selected, the user can view the details or edit it. If the user starts the workout, they perform and log the workout and then finish. After the workout is finished, the performance of the user is assessed with new goals set. This new information will then be stored.

When a user chooses to edit a workout, they can either change some details of it or delete the workout from the storage. If they choose to create a new workout, the inputs will be verified when they attempt to save it to storage and throw an error message if it is not done correctly.

5.2.3 Generate Workouts Use Case

Figure 5.4 conveys the process of a user having workouts generated for them:

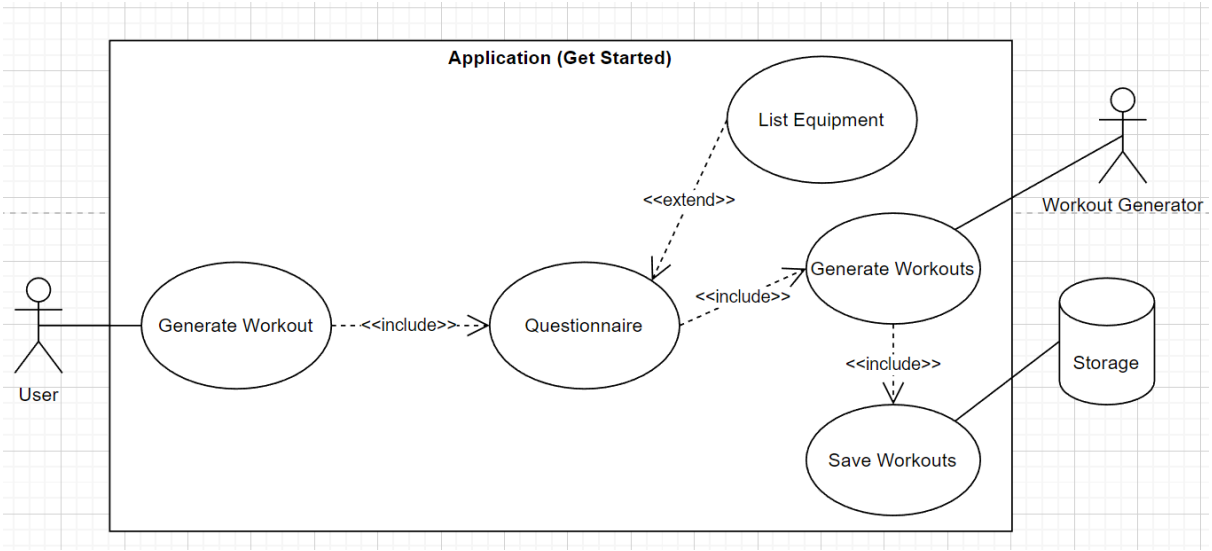

Figure 5.4 (Generate Workouts Use Case)

When the user selects the generate option in the form of a “Get Started” button, they will fill a questionnaire out with the option of listing their equipment if they wish to adapt it to that. This will then save the workouts after it is complete and save them to storage for the user to access.

5.2.3 Progress Use Case

Figure 5.5 presents the use case for checking progress:

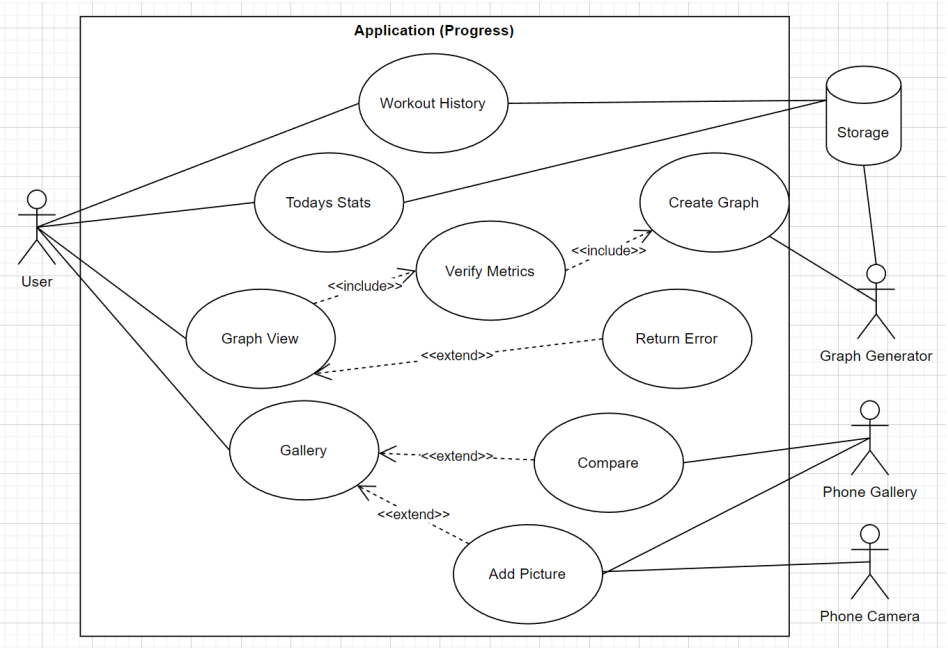

Figure 5.5 (Progress Use Case)

The progress interface consists of viewing daily statistics, viewing workout history, a graphical representation of their progress and a gallery. The “Workout History” and “Today’s Stats” directly fetch data from the storage to view. When a user chooses the “Graph View”, they are met with an interface that allows them to choose the metric within a time frame. An error is returned if there are invalid inputs in the time frames. Finally, when a user navigates to the “Gallery”, they can add pictures and compare between pictures at different time frames.

5.3 Mockups

To better visualise how the application’s user interface will work, mockups were created. There are two versions of mockups, the first was the original made to theoretically see what functionalities could fit onto what page, but was then later altered as the first set of mockups had a lot of room for improvement in a second version.

5.2.1 Version One

The following mockups were made with Adobe XD (Adobe Inc., 2021). Shown below are a few examples of the numerous original designs that were created for some of the main features of the application:

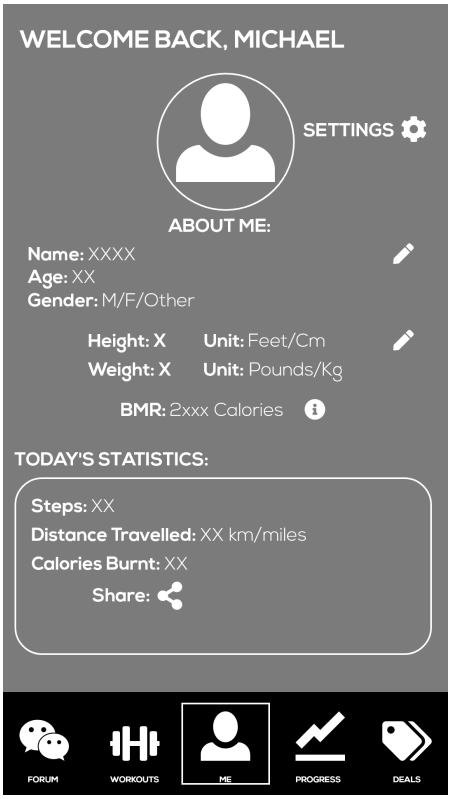

Figure 5.6 (Me/Home page)

Considered to be the core home page of the application, it should provide a clear interface for changing user metrics and updating weight as the user progresses in their workouts. This will be the page that opens first when the user runs the application with a navigation bar below inspired by analysing existing systems in Chapter 2. This will allow for easy navigation to the application's other features and make a clear distinction on where to tap to access the features.

Figure 5.7 (Progress page)

Presented here in Figure 5.7 was the first design of the progress page which showed how statistics and progress viewing features could be incorporated together. This design highlights some of the key weak points of the version one designs as a lot of features are packed tightly together and may not seem clear to the user.

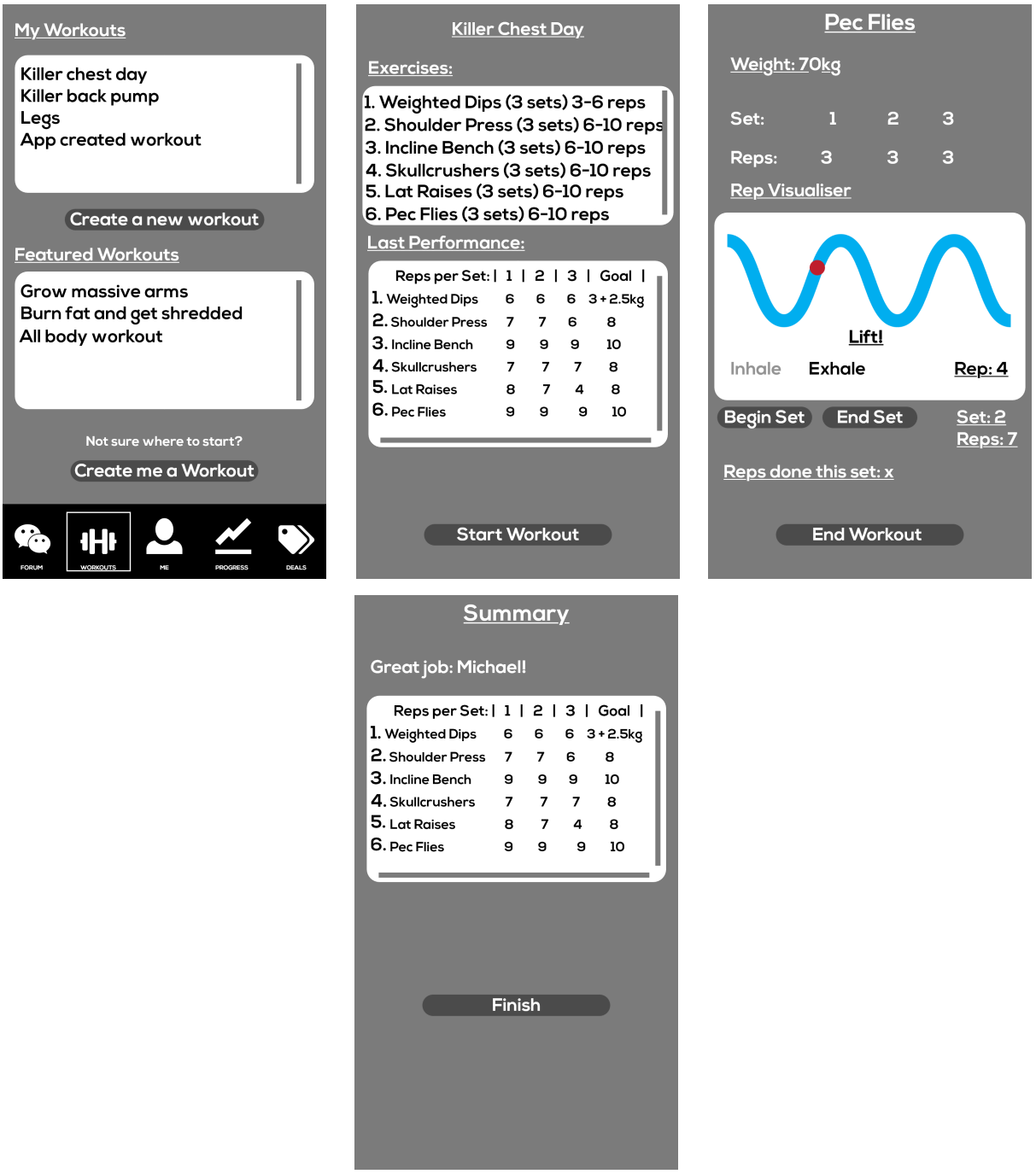

Figure 5.8 (Workout interface)

Shown in Figure 5.8 is the navigation flow of the workout interface. The flow was kept the same for version two as it was considered a strong point however once again, everything is packed together and data is difficult to read. The workout home page could have looked more visually appealing and the ‘Featured Workouts’ could

have been put on a separate page from ‘My Workouts’. The page shown when in the middle of a workout does not look like it can accommodate very well for a higher number of sets if the user chooses so.

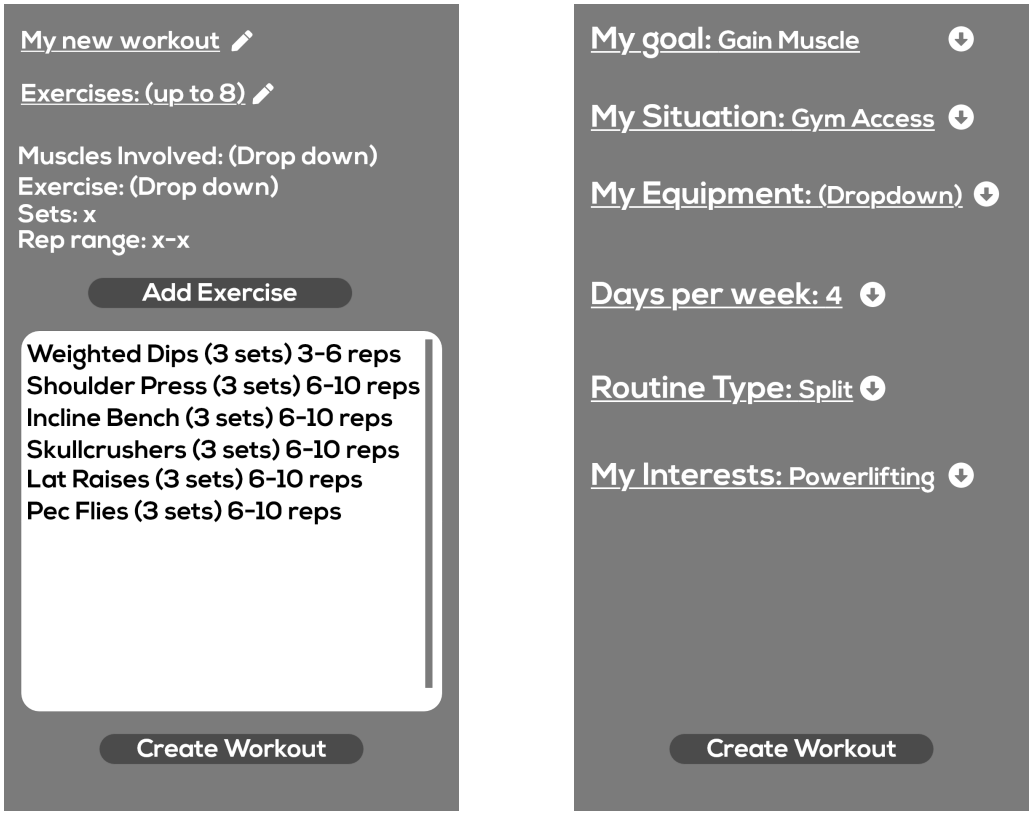

Figure 5.9 (Creating workouts manually and automatically)

In Figure 5.9, the forms of workout creation are shown. It was considered that the ‘Create Workout’ page could have a separate interface for looking up exercises to add to the workout and simplify the page further. The workout questionnaire could look much better with questions being presented at the top of the screen and big bold buttons for the answer to move on to the next question and make answering more clear.

The takeaway from these designs is that while the navigation flow is clear, a lot of the functionalities could be put on separate pages which could increase usability as it will in turn make the screens easier to read. Additionally, the presentation of

workout data also needs to be considered as there is a lot to display and keeping it as clean looking as possible will ensure it is not an eyesore for the user.

5.2.2 Version Two

The second attempt designs were made as wireframes with Adobe XD as well. These wireframes; made that way to save time are the layouts used for providing the improved user interface and layout.

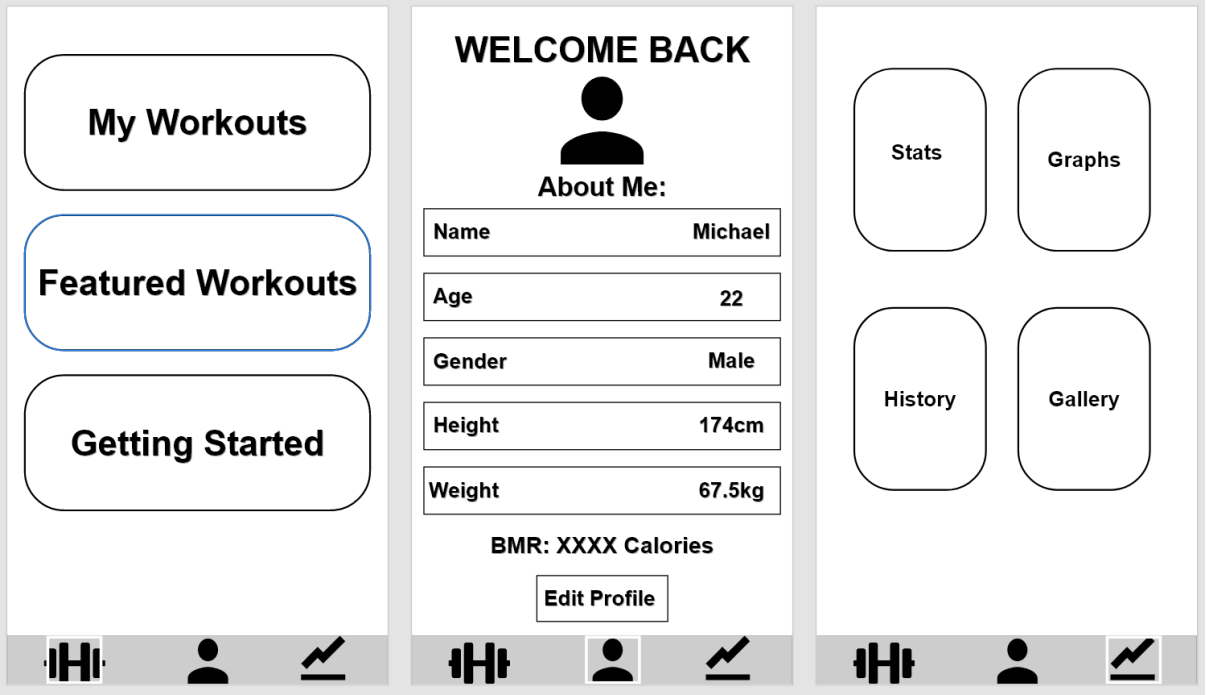

Figure 5.10 (Main pages from navigation bar)

In these new designs from Figure 5.10, it is far more clear on how to access each of the application's features with much clearer distinctions when dividing them into menus. This keeps the layout much more organised and user-friendly, whilst also being easier to look at. Data presentation on the 'Me Page' looks much neater and has a clear format which will make editing easier thus further increasing usability.

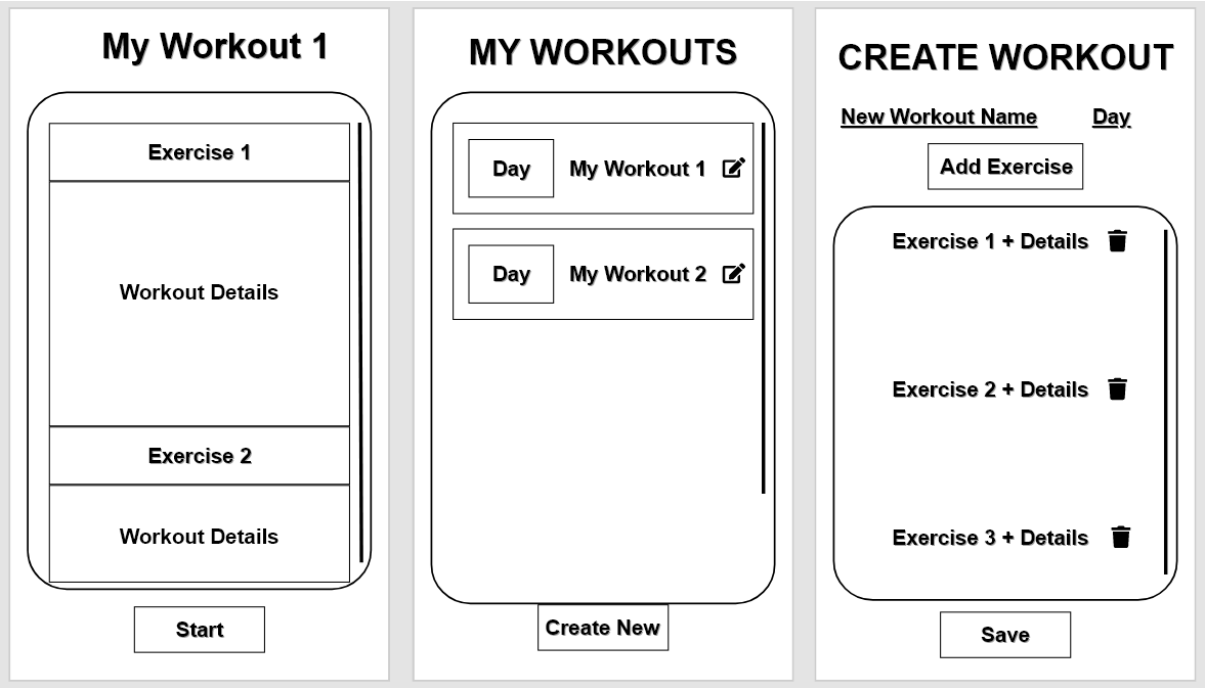

Figure 5.11 (Workouts viewing and creating)

Upon selecting ‘My Workouts’, it will bring you to a page displaying your workouts with an option to create new workouts. Once again, it is far more presentable with more pages allowing the simplification of each page as functionalities are divided up. Adding an exercise on the workout creation interface will be a separate page allowing for precise filtering and selecting other parameters of the exercise.

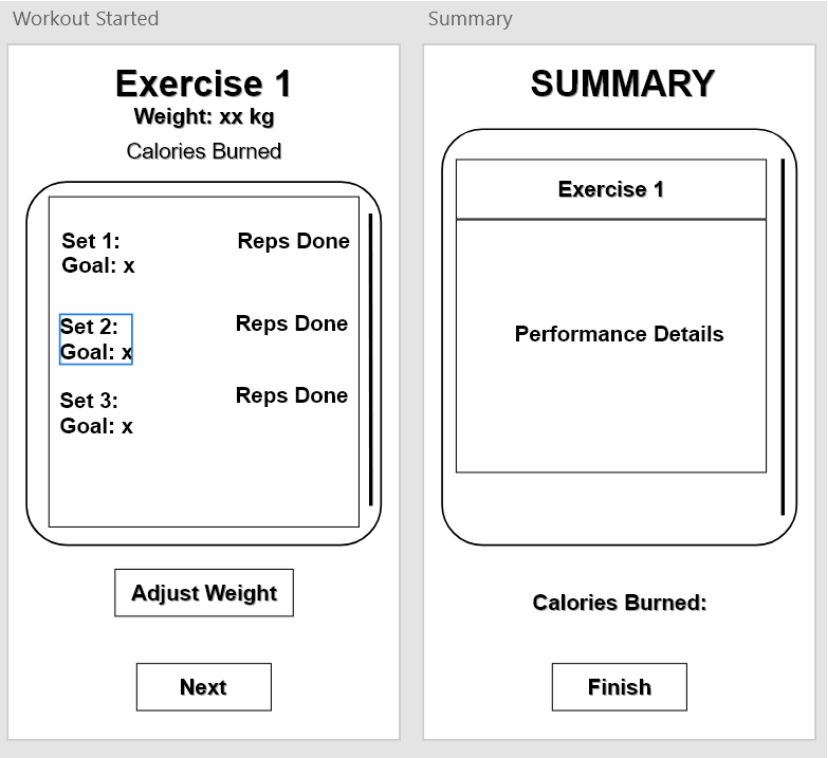

Figure 5.12 (Performing workouts and finishing)

Upon starting a workout, you will be presented with the following interface in Figure 5.12. Compared to version one, the way workout details and user inputs are displayed is more scalable and easier to adapt without sacrificing readability in a scrollable vertical list.

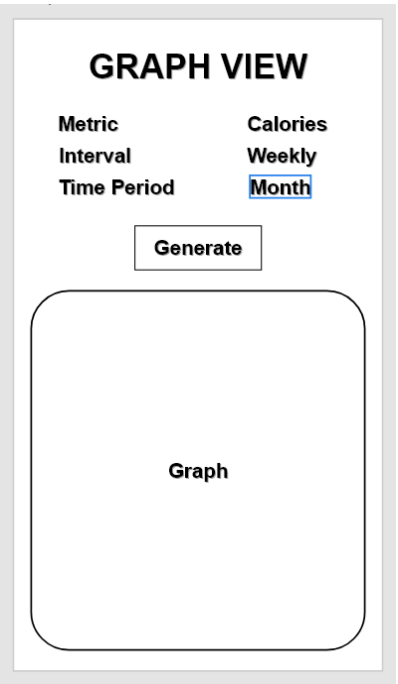

Figure 5.13 (Graph View)

Leaving the ‘Graph View’ page separate further benefits the interface as there is now more room for the graph allowing for greater visibility, this could be combined with a scroll view for closer analysis in periods such as by year.

Overall, separating the functionalities into more pages appears to make the interface much more navigable and cleaner looking. Pages aren’t as congested and it is more clear on what to do on each one. Careful consideration should be taken into how the workout data should be represented because there are numerous variables to display.

5.4 Colour

Colours in the application will be distributed using the 6:3:1 rule to make it stand out more. This means 60% will be for the background, 30% is used for texturing and 10% is the call to action. The call to action refers to areas users will click to do something such as buttons.

6. Implementation

In this chapter, the development of the application will be covered in chronological order where each increment denotes the feature that was worked on as it was created using an incremental approach. We start in Section 6.1 where the choice of technology and frameworks are discussed to create the application. Then in Sections 6.2 to 6.5, each increment of the development process will be discussed along with challenges, solutions and testing where appropriate.

6.1 Technology

This section covers the choice of technology for the implementation of this application. It overviews the original choice and why it was changed as well as the IDE selected as it was significant for

6.1.1 Android Studio

Android Studio (Jetbrains, 2013) is an IDE specifically made for the Android platform. It uses Java or Kotlin code to create Android applications and provides many useful tools for development. Originally the project started in Android Studio but was swiftly abandoned due to the author constantly experiencing bugs with the IDE which was considered a huge risk.

6.1.2 Flutter Framework

The author chose to migrate to the Flutter Framework created by (Google, 2017) which is a software development kit for making cross-platform mobile applications. It is supported in numerous IDEs and uses the Dart programming language which was quick for the author to learn. In addition, the author found the framework to be easy to learn.

6.1.3 VS Code

To utilise Flutter, the author chose to use VS Code from (Microsoft Corporation, 2015) which provides compatibility with Dart and Android emulators to test the application.

6.2 Navigation and Home Pages

This section covers the development of the navigation bar and home pages which marks the base of the application.

6.2.1 Development

Referencing the applications analysed in Chapter 2, it was decided the navigation bar would be the very first feature to be created as it would establish a solid foundation for the application where the home pages and routes could then be appended. Figure 6.1 below shows the navigation bar design:

Figure 6.1 (Navigation Bar)

Icons were made by the author through Adobe Illustrator (Adobe Inc., 2021). When each icon is clicked, the text will be displayed below for the page selected with icon highlighting to make it clear what page the user has selected.

With the foundation established, it was time to develop the home pages according to the wireframe diagrams in Chapter 5. The designs for the home pages are presented here in Figure 6.2:

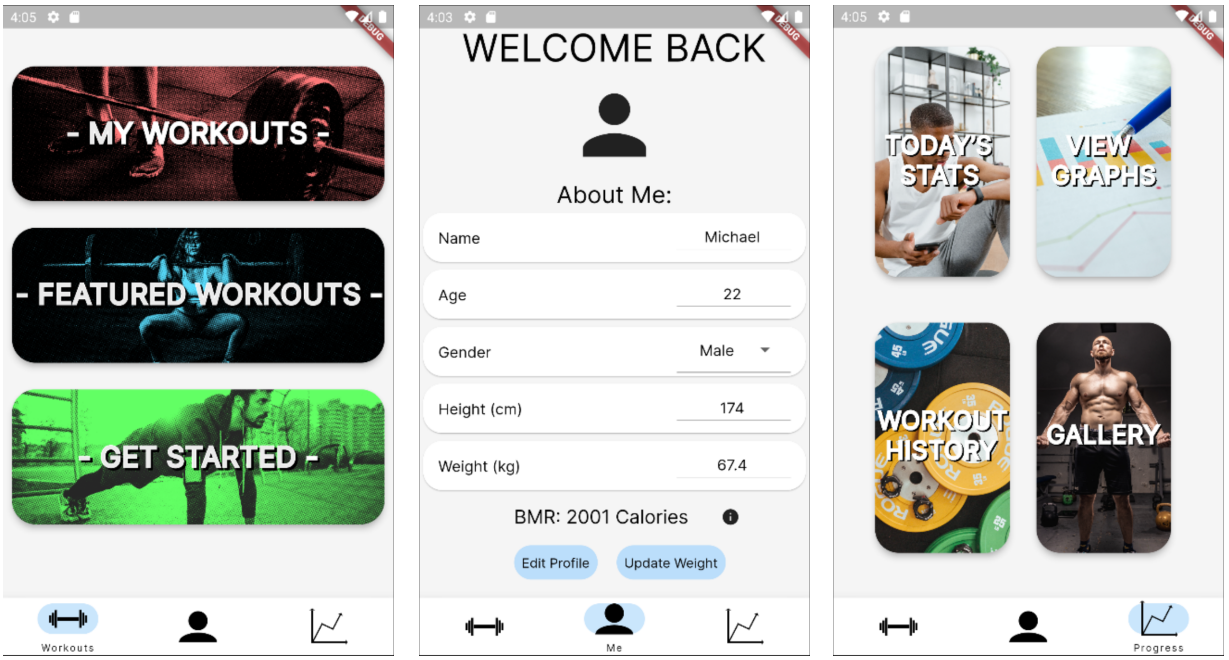

Figure 6.2 (Home Pages)

The ‘Workouts’ and ‘Progress’ pages have big bold picture buttons also taken as inspiration from analysed implementations to add more colour to the application and give it a human touch. All picture button designs were made from scratch by the author using Adobe After Effects (Adobe Inc., 2022) with royalty-free stock images provided by (Pexels 2019).

As the user metrics are well-defined and not subject to change, the functionalities in the ‘Me Page’ could be finished off right away allowing testing for the proposed JSON format for saving data. This would also allow the BMR calculator to be made to commence the backend development.

6.2.2 Challenges

The creation of these foundational features was not problematic as the Flutter Framework offers many parameters within its widget creation that allow fitting images neatly within the ‘Card’ widgets used and preventing pixel overflow errors.

However, challenges arose in file saving as a mechanism would need to be implemented for when the file does not exist to prevent errors which were solved with exception handling where file writing would occur when an exception occurs. In addition to this, a system would need to check the date of the system for use with the progress and history features later on, the solution is presented in Figure 6.3:

```
DateTime now = DateTime.now();
    DateTime date = DateTime(now.year, now.month);
    String formattedDate = date.toString().substring(0,10);

    var weightMap={
      "weight":weight,
      "date":formattedDate
    };

    bool dateInSave = false;

    if(decodedContents.isEmpty){
      decodedContents.add(weightMap);
    }
    else{
      for(var element in decodedContents){
        if(element["date"] == formattedDate){
          dateInSave = true;
          break;
        }
      }
      if(!dateInSave){
        decodedContents.add(weightMap);
      }
    }
```

Figure 6.3 (Adding and formatting date fields)

In Dart, JSONs can be converted to and from maps, so a default map will need to be created to structure the data each time it is saved on a different day. This is to ensure data saves on days the user updates their weight on the application to preserve storage as opposed to automatically creating entries each day. A loop will iterate through each entry in the storage to see if the date exists on the storage and

if not, the default “weightMap” will be appended. The use of this mechanism would then be used for any file saving that includes date tracking.

Another challenge arose regarding data loading as when the ‘Me Page’ loads, the data does not load instantly and causes a very brief flickering in the text fields which was resolved by adding a very short loading screen to allow the data to load.

To make the application more robust, a system to check if user input is correct needed implementation to prevent errors in files which was executed by verifying user input through regular expressions before attempting a file save. Presented in Figure 6.4 is a snippet of the solution for this:

```
if(!RegExp(r'^[A-Za-z]+$').hasMatch(_nameTextController.text)){
      final snackBar = SnackBar(content: Text('Enter a valid name'),
            duration: Duration(seconds: 2));

ScaffoldMessenger.of(context).showSnackBar(snackBar);
      return false;
    }
```

Figure 6.4 (Regular Expressions for Input Verification)

When the save changes button is clicked, it will check if all the inputs are correct before allowing data saving to occur, this reduces the need for exception handling in the file saving function and increases the robustness of the application. This will also provide the user visual feedback on what the error is using the ‘Snack Bar’ widget which appears like a notification on the bottom of the screen.

Finally, to make the application more gender inclusive, an ‘Other’ option was included which needed to be adapted for the BMR calculation, this was resolved by taking an average of the male and female BMR results.

6.2.3 Testing

Presented below is a table with results from generally testing the features implemented so far in Figure 6.5:

Feature	Description of Test	Result
Navigation Bar	The navigation bar takes the user to the correct pages.	Success
Save User Metrics	The user can enter and save user metrics with a BMR calculation provided below.	Success
Update Weight	The user can update their weight with a new entry being stored if it is a different day.	Success
Input Handling	The program will not permit the user from submitting invalid inputs.	Success

Figure 6.5 (Testing features for first and second increment)

Unit testing was also done on the BMR calculator to ensure the correct outputs were consistently produced with a variety of inputs:

```
Run | Debug
void main(){
  Run | Debug
  test("Male output", (){
    BMRCalculator calc = BMRCalculator(
      22, "Male", 174, 67.4);
    int result = calc.getBMR();
    expect(result, 2001);
  });
  Run | Debug
  test("Female output", (){
    BMRCalculator calc = BMRCalculator(
      29, "Female", 163, 60);
    int result = calc.getBMR();
    expect(result, 1682);
  });
  Run | Debug
  test("Other output", (){
    BMRCalculator calc = BMRCalculator(
      24, "Other", 190, 70);
    int result = calc.getBMR();
    expect(result, 1991);
  });
}
```

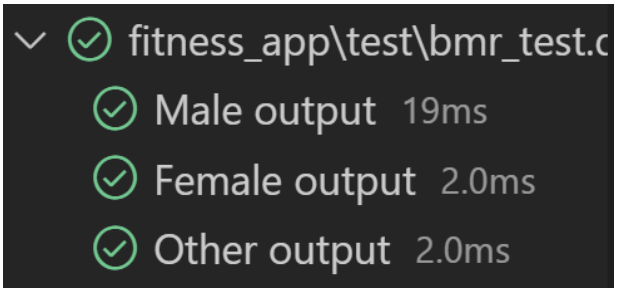

Figure 6.6 (Unit testing BMR calculator)

6.3 My Workouts and Goal Setting

In this increment, the main workout features were developed allowing users to view their workouts, create workouts, complete workouts with goals set, edit settings related to these workouts and have their performance assessed.

6.3.1 Development

From the ‘Workouts’ page within the ‘My Workouts’ section, a basic scrollable list interface was made to present the user with their workouts as shown in Figure 6.7:

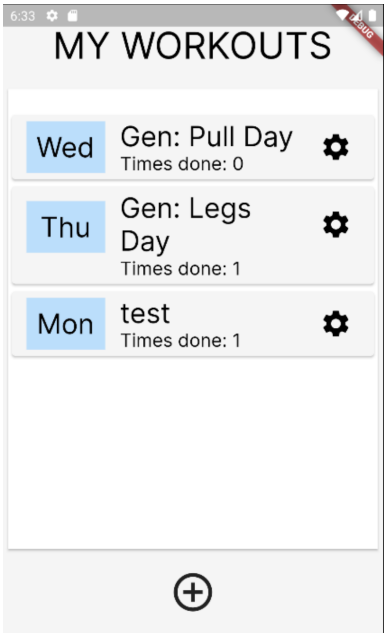

Figure 6.7 (My Workouts)

It was important to keep this page simple with a create workout button at the bottom of the page and a settings button for each workout to edit them. Clicking on the workout otherwise takes the user to view and begin a workout. Displayed in Figure 6.8 below is the interface for viewing and performing workouts:

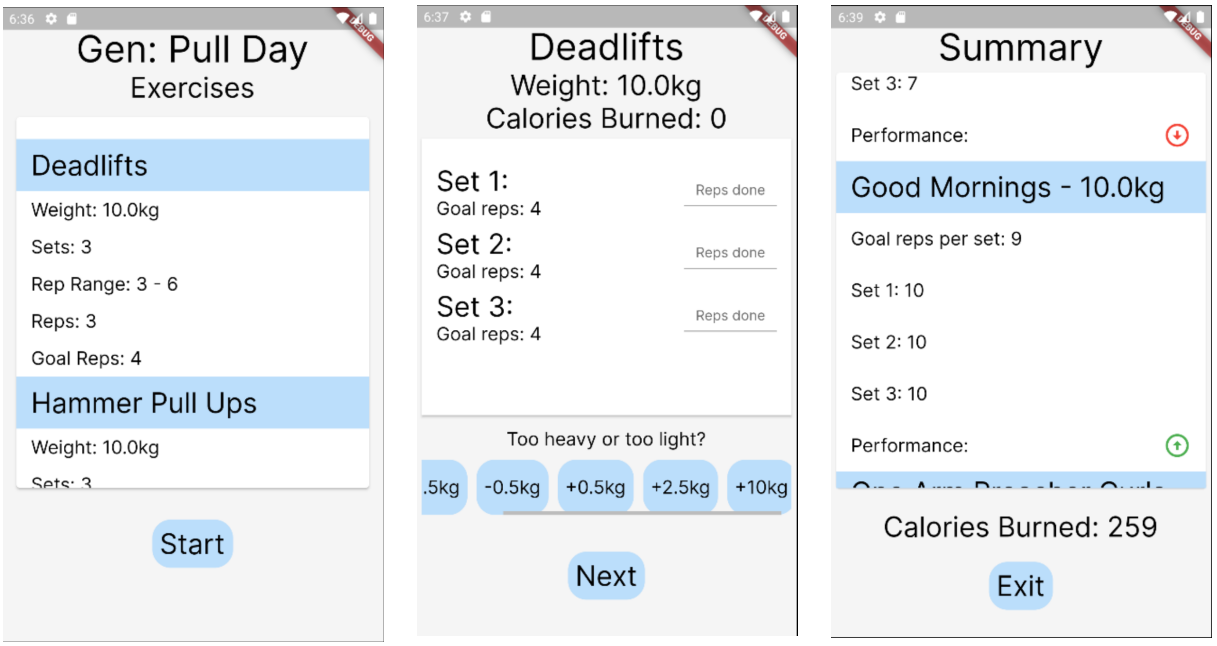

Figure 6.8 (Workout interface)

On selecting a workout, the user is shown the details along with the goal repetitions for each set. The goal adds one repetition to the last performed number unless the user had previously reached the upper bound of the rep range or higher, or way lower than the lower bound which would then adjust the weight accordingly.

The user can adaptively alter the weight as they go on depending on if it is too difficult or easy. Along the way, calories are counted using the METs formula for calorie tracking. The Mets formula attributes a number for activities based on how many calories they burn in a given time period which can then be multiplied by the user's weight and duration of the exercise to obtain a total for calories burned. Many different resistance exercises have a range of MET values however none of them were far off each other so an average was taken to give a general solution.

Upon finishing the workout, the user is given an overview of their performance with arrows indicating how they did against the goals assessed by the backend.

At this stage of the development process, the ‘models’ were made for exercises and workouts with exercises being contained within workouts. Shown in Figures 6.9 and 6.10 are the models for the exercises and workouts:

```
class Exercise{

  String name;
  double weight;
  int reps;
  int sets;
  List<int> repRange;

  Exercise(this.name, this.weight,
    this.reps, this.sets, this.repRange);

  incrementReps(){
    reps = reps + 1;
  }

  checkWeightIncrement(){
    if(reps >= repRange[1]){
      weight += 2.5;
      reps = repRange[0];
    }
    else if(reps <= repRange[0] - 1){
      weight -= 2.5;
      reps = repRange[1] - 1;
    }
  }

  String toParse() => "&$name/$weight/$reps/$sets/$repRange";

}
```

Figure 6.9 (Model for exercises)

The structure of an exercise contains the name, weight, repetitions, sets and the rep range. The method “checkWeightIncrement()” will help to apply progressive overload by increasing the weight if the repetitions reach the upper bound of the rep range meaning the user is ready to move up.

```
import 'exercise.dart';

class Workout{

  String name;
  List<Exercise> exercises;
  int timesDone;
  String day;

  Workout(this.name, this.exercises, this.timesDone, this.day);

  //exercise stuff
  addExercise(Exercise exercise){
    exercises.add(exercise);
  }

  removeExercise(Exercise exercise){
    exercises.remove(exercise);
  }

  workoutDone(){
    timesDone++;
  }

}
```

Figure 6.10 (Model for workouts)

Within a workout, the name, list of exercises, times completed and the day is stored. Functions are added to add and remove exercises for workout creation.

A useful feature in Dart is not requiring getter and setter methods for variables in classes as they can easily be accessed and edited directly.

With these features complete, a method for making and editing workouts without having to edit them after starting a workout was developed. Figure 6.11 illustrates the designs of these pages:

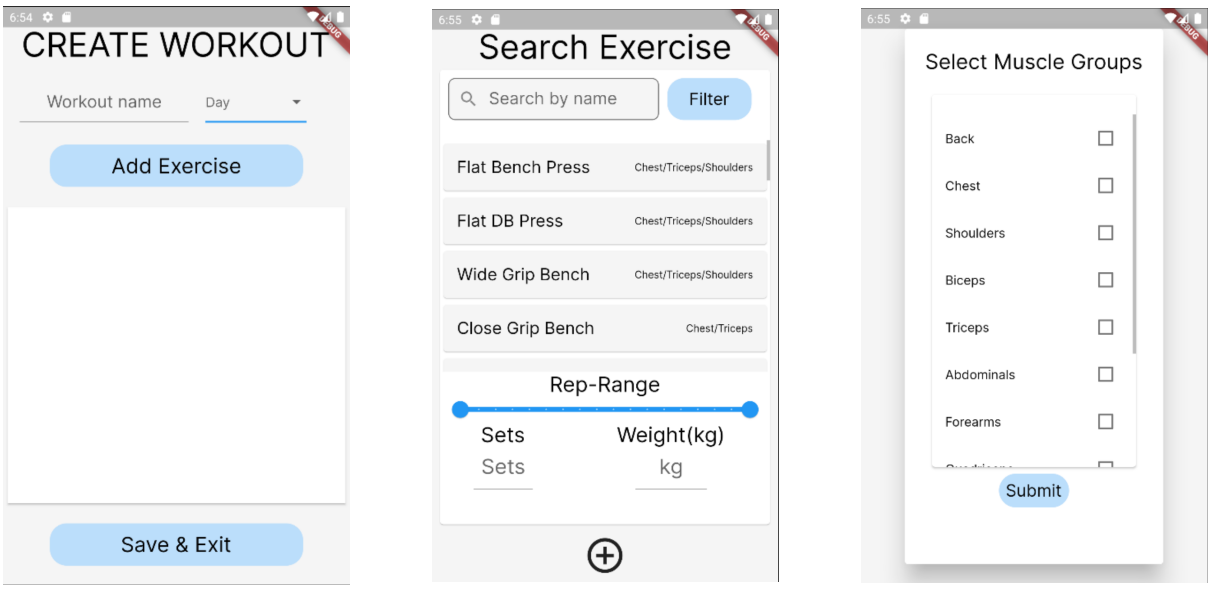

Figure 6.11 (Creating a Workout)

To make workout creation intuitive, it was kept simplified with an add exercise button taking the user to a page where they can filter and search the exercises they want to add. Exercises are also able to be removed from the added exercise list if need be. The same mechanism for verifying inputs in the 'Me Page' was used in the workout creation interface as well to prevent invalid inputs from going through.

Finally, a small edit workout page was created to allow renaming, date changes and other adjustments to be made in Figure 6.12:

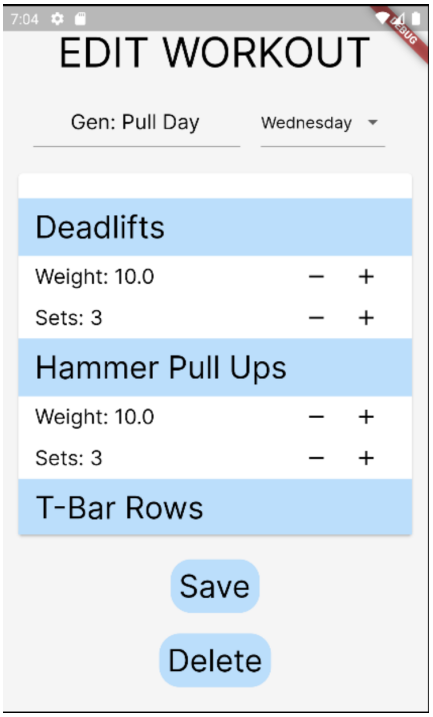

Figure 6.12 (Edit workout page)

6.3.2 Challenges

Challenges were presented with the creation of the workout interface. Most of it derived from frontend functionalities as the nesting of adaptive listviews was problematic in areas such as workout selection where data is observed. This was required as it allowed the titles of the exercises to be highlighted with data listed below to give a clear distinction on what is being read. Tweaking around certain parameters of the ListView objects in Flutter allowed for such adaptations.

Counting the calories from working out proved an issue as the duration had to be estimated based on the sets and repetitions done of the exercise but a general solution was used. The approach is presented here in Figure 6.13:

```dart
import 'package:collection/collection.dart';

class ExerciseCalorieCounter{

  List<int> reps;
  int sets;
  double bodyWeight;


  static const double METvalue = 6.0; // Mets formula
  static const double repTime = 5/3600; // 2 seconds per rep in  hours
  static const double restTime = 1/30; // 2 mins rest time in hours

  ExerciseCalorieCounter(this.reps, this.sets, this.bodyWeight);

  int exerciseFinished(){

    double duration = getExerciseDuration();
    double burned = METvalue * bodyWeight * duration;

    return burned.round(); // For display when working out

  }

  getExerciseDuration() => (reps.sum * repTime) + (sets * restTime);

}
```

Figure 6.13 (Calorie Tracking from Exercises)

After each exercise is completed, it will be passed through a calorie counting class using generally well-known rest times and rep times as default values to give this general solution. The results output look very reasonable but are not the most accurate as they do not take into account the weight being lifted as lifting a heavier weight requires more energy.

A format for saving the workouts had to be decided as implementing it as a JSON would be too time-consuming so CSV was decided. Implementing an SQLite database would be even better however CSV was very quick to work with and especially considering more variables would be added later for the workout

generator. Considering the time constraints of this project, having something that works and is quick was considered more beneficial for the project.

Within the Edit Workout feature, the most prominent challenge was ensuring workouts of the same name don't get deleted together when one needs to get deleted. This had to be considered as the way workout generation would work is it would give general names for workouts which could match others resulting in numerous workouts with the same name being deleted if removing was based on the name. This problem was rectified by making comparisons not only in existing workout names but in the exercise details inside to ensure the right workout would be deleted.

Finally, saving the workouts directly to JSON was a topic of concern. The workout model object contains an array of exercise objects as a parameter and many attempts at converting this to JSON failed as it would include a JSON array within each JSON array entry. The solution settled on consists of converting the exercise list into a particularly formatted string rather than making it a JSON array. The solution executed was a fast one but not necessarily the best. However, this bug was a big bottleneck to the development of the application. The code for formatting exercises into this string format can be seen in the “toParse()” method in Figure 6.9

The “&” characters separate exercises from one another whilst “/” characters distinguish between the fields stored in them.

6.3.3 Testing

Show in Figure 6.14 is a general assessment of the features implemented in this increment:

Feature	Description of Test	Result
View Workouts	All workouts stored will be loaded and can be individually selected to view details.	Success
Workout Summary	Upon workout completion, performance is assessed and goals are adjusted correctly.	Success
Edit Workout	Users can edit workout details and goal setting adjusts to that.	Success
Delete Workout	Correct workout is deleted without deleting others of the same name.	Success
Create Workout	The user can create workouts with exercises they choose and other parameters.	Success
Search Exercises	Exercises data can be searched and filtered based on muscle groups they exercise.	Success
Input Handling	The program will not permit the user from submitting invalid inputs.	Success

Figure 6.14 (Testing features for third increment)

To further support that some of the features tested above were robust, unit tests and widgets were performed. Figure 6.15 shows unit testing for calorie tracking in exercises:

```
test("Exercise 1", (){
  ExerciseCalorieCounter counter = ExerciseCalorieCounter([8,7,6],
  3, 67.5);
  int toAdd = counter.exerciseFinished();
  expect(toAdd, 52);
});
Run | Debug
test("Exercise 2", (){
  ExerciseCalorieCounter counter = ExerciseCalorieCounter([12,12,12],
```

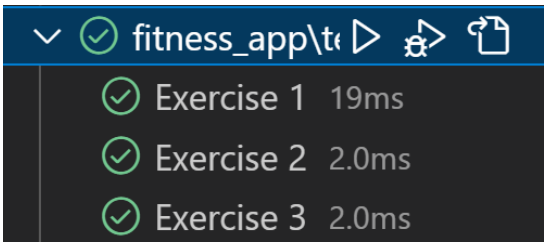

Figure 6.15 (Unit Testing for exercise calorie counting)

Figure 6.16 shows widget testing for performance assessment:

```
void main() {
  Run | Debug
  testWidgets("Goal Met", (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: assessPerformance([8,8,8], 8)));
    expect(find.byWidgetPredicate(
      (widget) => widget is Icon &&
                  widget.icon == Icons.arrow_circle_up_outlined &&
                  widget.color == Colors.green &&
                  widget.size == 30.0
    ), findsOneWidget);
  });

  Run | Debug
```

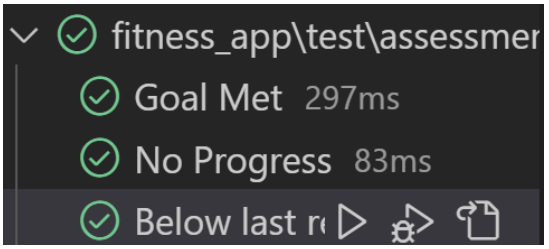

Figure 6.16 (Widget testing performance assessment)

6.4 Workout Generation and Featured Workouts

This section discusses the implementation of the workout split generator for the application along with a system for implementing featured workouts.

6.4.1 Development

The most complicated feature of the application was the workout generator. This would be activated upon clicking “Get Started”. To understand how this would be made, the questionnaire page was created first to settle on what inputs would be required. To allow for versatility, if a user does not have gym access and have equipment, they have the option to list their equipment for workout generation. Figure 6.17 shows some example screens for the workout generation questionnaire:

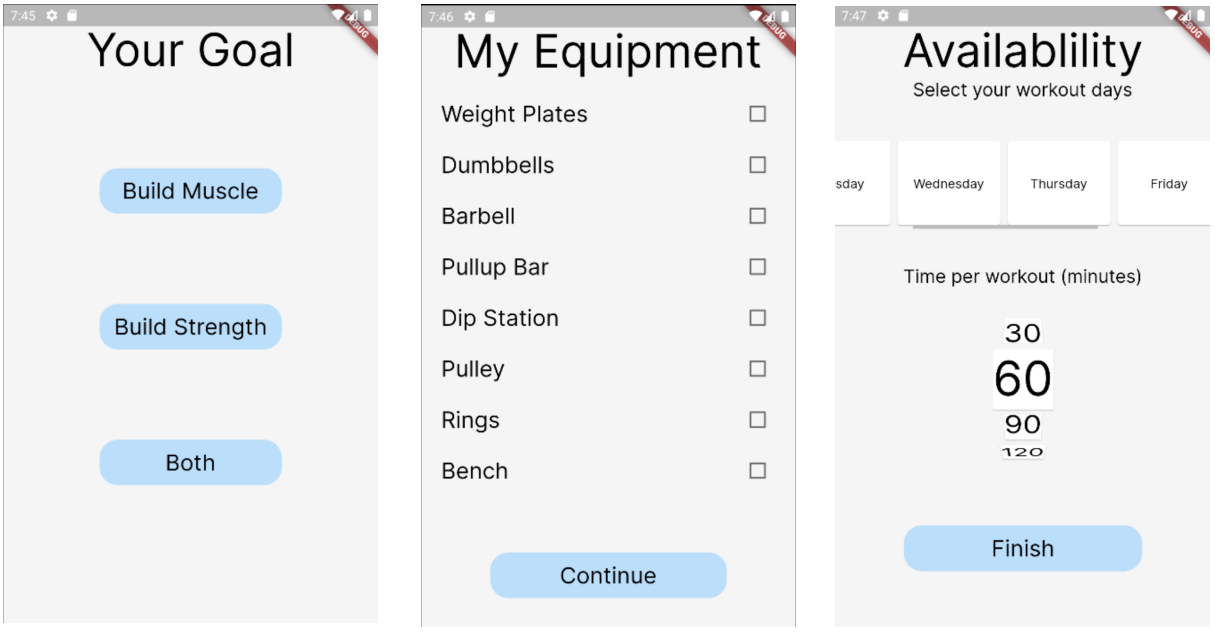

Figure 6.17 (Example questions from questionnaire)

When the user clicks ‘Finish’, numerous workouts are generated with ‘workouts days’ selected. The Workout Generator takes the following inputs to generate a workout split for the user:

- Experience Level (Beginner, intermediate or expert)

- Goal (Muscle, strength or both)
- Access (Gym, owned equipment or no equipment)
- Equipment List (If “Home/Equipment” is selected)
- Available Days (Maximum of 5 selected)
- Time for each workout (30 minutes up to 120 minutes)

Inputs such as experience level, equipment list and access were used to filter out what exercises are suited for the user and what they can do. The goal determines the repetition ranges as it is a widely considered metric for what is best for certain goals. However many available days will tell the generator how many workouts to create with their corresponding dates. And finally, the time for each workout will decide how many exercises go into each workout.

For a workout that would be generated, the first two exercises would always be high intensity with lower intensity exercises being added for every two that get added. This was to replicate a commonly recommended format of workout where the most energetically demanding exercises are done in descending order. An element of randomness was added to keep things more interesting, this was executed by randomly selecting a workout split and randomly selecting exercises using the rule above.

Shown in Figure 6.18 is the main function of the workout generator which outlines how it works:

```
List<Workout> generateWorkouts(){

    //Get all parameters to generate workouts.
    List<List<dynamic>> filteredExercises = filterExercisesList();
//Exercises available to user
    int numberOfWorkouts = getWorkoutAmount(); //1-5
    List<List<int>> repRanges = getRepRanges(); //Rep Ranges to use in
split
    int exercisesPerWorkout = getExercisesPerWorkout(); //Exercises for
each workout
    List<String> workoutSplit = decideSplit(numberOfWorkouts); //Workout
split to be used

    List<Workout> generatedWorkouts = [];

    for(var i = 0; i < workoutSplit.length; i++){
      generatedWorkouts.add(
        generateWorkout(
          filteredExercises, repRanges, exercisesPerWorkout,
workoutSplit[i], i));
    }

    return generatedWorkouts;
  }
```

Figure 6.18 (Workout generator main function)

Another feature that was quickly implemented was the ‘Featured Workouts’ section. Ideally, it would work by querying an active database however to save time, the author put their own workouts inside as this feature was not the most important.

Displayed in Figure 6.19 is the ‘Featured Workouts’ page:

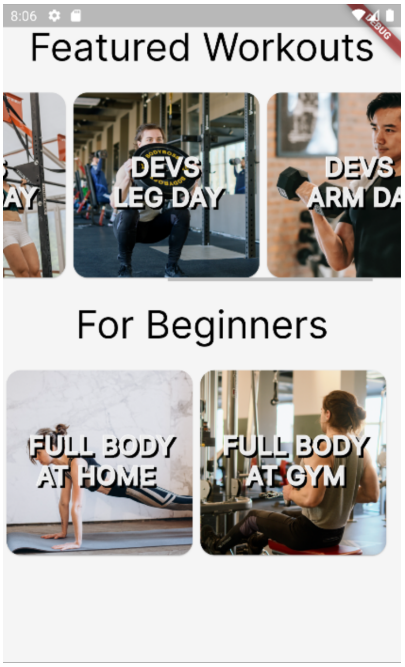

Figure 6.19 (Featured workouts page)

Users can select these workouts, view them, and then add them to their list of own workouts.

6.4.2 Challenges

Most issues arose in how the workout generator would work. Setting up the current CSV file to accommodate this was vital. Data was formatted in the following way:

```
1   exercise name,
2   muscle groups,
3   difficulty [1-3 beg, int, adv],
4   access level [1-3 home,equip,gym only],
5   equipmentlist,
6   intensity [1-3] 3 is most intense
```

```
Flat Bench Press,Chest/Triceps/Shoulders,Intermediate,2,Dumbbells+Bench,3
Flat DB Press,Chest/Triceps/Shoulders,Beginner,2,Barbell+Weight Plates+Bench,3
Wide Grip Bench,Chest/Triceps/Shoulders,Intermediate,2,Barbell+Weight Plates+Bench,3
Close Grip Bench,Chest/Triceps,Intermediate,2,Barbell+Weight Plates+Bench,3
Incline Bench Press,Chest/Triceps/Shoulders,Intermediate,2,Barbell+Weight Plates+Bench,2
DB Incline Press,Chest/Triceps/Shoulders,Beginner,2,Dumbbells+Bench,2
Decline Bench Press,Chest/Triceps/Shoulders,Intermediate,2,Barbell+Weight Plates+Bench,3
DB Decline Press,Chest/Triceps/Shoulders,Beginner,2,Dumbbells+Bench,3
Machine Chest Press,Chest/Triceps/Shoulders,Beginner,2,Gym,3
```

Figure 6.20 (CSV structure)

As discussed earlier, CSV was used to save time whereas the author thought the use of SQLite would have been better as it is more scalable. A list of widely performed exercises was placed in it with corresponding values set to them for the generator to filter out the right exercises.

The CSV file contains 116 entries however the generator had to accommodate for areas where there aren't many exercises. The most notable examples were back exercises using no equipment or a lack of calf exercises. To solve this, the workout generator would cut off adding exercises if none were eligible to be added or use exercises from a different intensity level to fill it out. Not much more data could have been added as there is a limited number of exercises.

6.4.3 Testing

Figure 6.21 presents the results for testing the features implemented:

Feature	Description of Test	Result
Generate Workouts	The user can create workout to accommodate for them	Success
Featured Workouts	The user can view and add featured workouts.	Success
Input Handling	The program will not permit the user from submitting invalid inputs.	Success

Figure 6.21 (Testing for fourth increment)

A unit test was performed on the filtering mechanisms of the workout generator shown in Figure 6.22:

```dart
void main() {

  List<List<dynamic>> _listData = TestData().returnList();
  Run | Debug
  test('Can filter by experience level', () async {

    WorkoutGenerator workoutGenerator = WorkoutGenerator(_listData, 'Beginner', 'Both', 2, [], ['Mon', 'Wed', 'Fri'], 60);
    List<List<dynamic>> filteredExercises = workoutGenerator.filterByExperience(_listData);
    expect(filteredExercises, [
      ["Calf Press","Calves","Beginner",3,"Gym",1],
    ]);
  });
  Run | Debug
  test('Can filter by access level', () async {
    WorkoutGenerator workoutGenerator = WorkoutGenerator(_listData, 'Intermediate', 'Both', 2, ["Barbell", "Weight Plates"
    List<List<dynamic>> filteredExercises = workoutGenerator.filterByAccess(_listData);
```

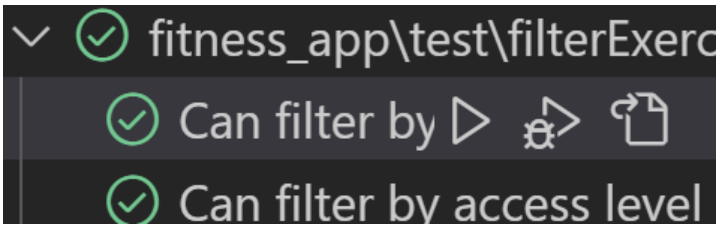

Figure 6.22 (Unit testing the workout generator)

6.5 Tracking Features

Within this section is the remainder of the features that were implemented. This includes counting statistics and workout history.

6.5.1 Development

With calorie counting implemented, it was time to add other tracking features. A saving system for calories was created and a pedometer was added. Shown in Figure 6.23 is the statistics page:

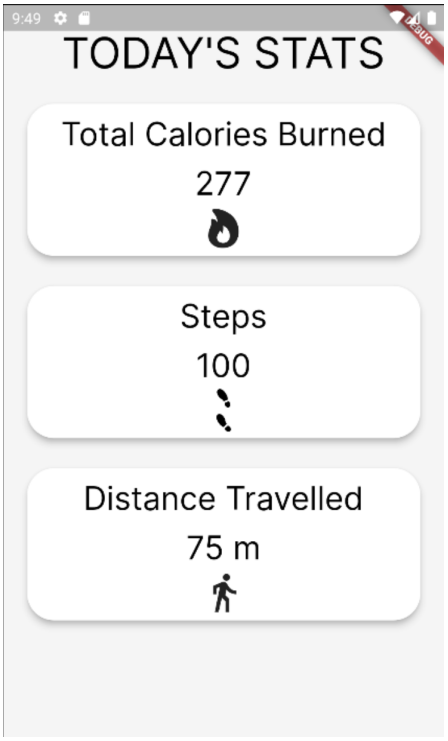

Figure 6.23 (Stats page)

While steps are counted, the distance is tracked along by multiplying each step by 0.74m which is considered an average stride length. The calories burned come from counting the calories burned from working out and the steps taken. An extra calorie is added to the total for every 45 steps taken as a general estimate.

Moving on, a page was made to show workout history. This displays the workout done and on what day. The page made is presented in Figure 6.24:

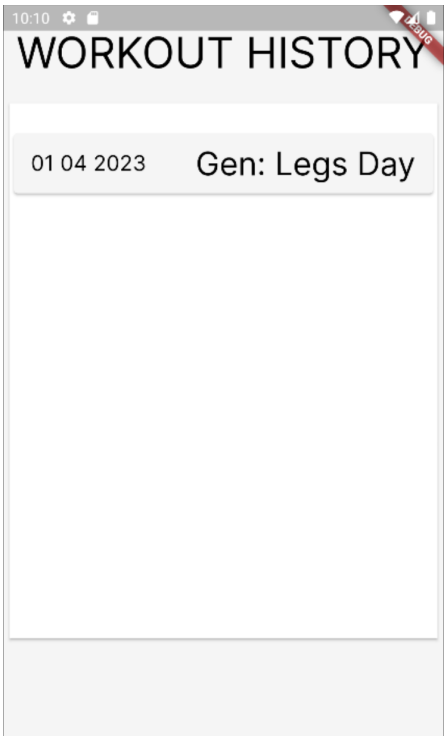

Figure 6.24 (Workout history page)

A date of when the workout was performed along with the workout done is presented in a list view.

6.5.2 Challenges

A notable problem was making the pedometer. Many libraries exist for implementing a pedometer automatically however none of these worked due to the Android version being too old. Keeping the version old means that the application would work on most phones so a pedometer was made from scratch instead. It was as simple as using the ‘sensors’ package and accessing the accelerometer of the phone. Figure 6.25 displays the pedometer code:

```
void initStepCounter() async {

    userAccelerometerEvents.listen((UserAccelerometerEvent event) async
{

      var x = event.x;
      var z = event.z;

      accel = sqrt(pow(x, 2) + pow(z, 2));

      if(!debounce && accel > 2){
        // accel = 2
        debounce = true;

        setState(() {
        stepCount++;
        tempStepCount++;
        distanceCount += 0.74;
        });

        await Future.delayed(Duration(milliseconds: 450));
        debounce = false;

      }

    },);
  }
```

Figure 6.25 (Pedometer implementation)

The accelerometer gives a reading recursively every half of a second of the current vectors in all directions. Only the x and z vectors were taken to allow the pedometer to be consistent for all walking styles. The vectors are squared, summed and square rooted to get the resultant vector and if it reaches above the constant 2 metres per second, a step is added. A ‘debounce’ flag was added to prevent the pedometer from incrementing numerous steps at a time and keep it one at a time. The goal was to make the pedometer accurate enough for general use and that was achieved through constantly tweaking the variables of it.

As mentioned, the step counter event is constantly executed which causes complications for file saving as we don't want to file save every less than half a second. A new function was added to save the data every 10 seconds instead of every time this step counter function executes to allow the application to remain as well performing as it can. Shown in Figure 6.26 is the function that handles the repeated saving:

```
void handleSaving() async {

    while(true){
      await Future.delayed(Duration(seconds: 10));
      if(tempStepCount >= 45){
         updateCalorieInfo();
      }
      saveStepsInfo();
      saveDistanceInfo();
    }

  }
```

Figure 6.26 (Statistics saving)

This mechanism is run asynchronously to prevent interruptions in the program control flow.

6.5.3 Testing

Figure 6.27 displays results for testing the features implemented in this increment:

Feature	Description of Test	Result
Statistics	User's statistics such as steps, distance and calories are tracked and presented.	Success
Workout History	The user can view their workout history with a timestamp and the workout done presented.	Fail - Date and time of system does not update for an unknown reason

Figure 6.27 (Testing for fifth increment)

The step counter was also tested to review the accuracy shown in Figure 6.28:

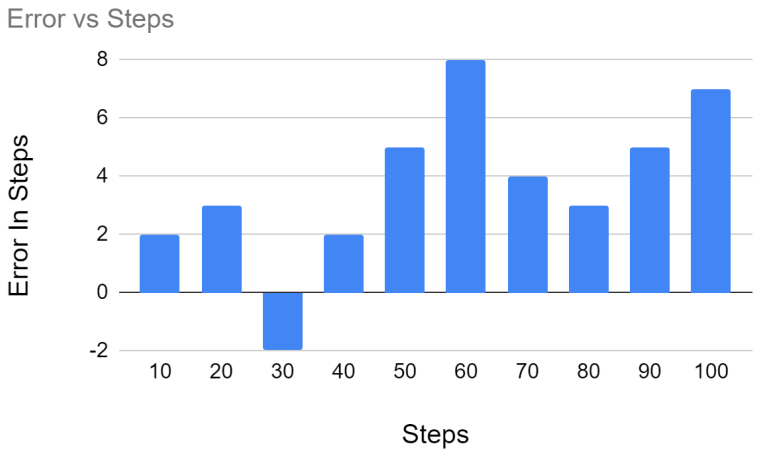

Figure 6.28 (Pedometer test results)

The author walked 100 steps and noted the steps recorded every 10 actual steps to assess the accuracy of the pedometer. More often than not, the pedometer overcounted by no more than 8 steps displaying a reasonable amount of accuracy but adequate for counting calories burned.

A bug presented itself that remains unfixed is the date retrieving always retrieves the same date. This does not allow the history to be tracked over a longer period and makes the date remain fixed.

7 Evaluation

In this section, we evaluate the project starting in Section 7.1, which is against the project's aims and objectives. Then in Section 7.2, the evaluation against requirements is listed.

7.1 Evaluation against Objectives

Referring back to Chapter 1 Section 1.2, the project aimed to create a mobile application that lets users log and plan workouts whilst being able to apply progressive overload. Furthermore, this application will allow users to do this regardless of having equipment or not and their experience level. This aim has been met as the user can create workouts, perform workouts and have then logged and assessed for goals to push the user which in turn implements progressive overload. For users that are complete beginners to working out, they can have a workout split generated customised for them if they wish upon pressing ‘Get Started’ which produces relatively reasonable and balanced workouts most of the time.

Referring to the literature reviewed in Chapter 2, the application tracks user activity and sets goals which implement gamification on a low level to hopefully increase user retention. Compared to the applications assessed in that same chapter, most features that exist in current systems exist in the application developed which allows it to contend as a workout logging and planning application.

7.2 Evaluation against Requirements

Provided below is a list of all the requirements and whether they are satisfied or not. A requirement is considered satisfied if it fulfils the description of its requirement from Chapter 4.

7.2.1 Functional Requirements

Listed below is the evaluation of the functional requirements:

MH-F-1 Save User Metrics. (Satisfied) - User metrics can be edited accordingly and used for other features such as calorie counting. The weight is saved with the according date.

MH-F-2 Save Data. (Satisfied) - All data on the requirement is successfully saved and loaded which allows for the other features to work properly.

MH-F-3 Workout Logging. (Satisfied) - The system tracks the user's progress on the workout they decide to do and uses it for goal setting in MH-F-4. Users can input their performance on each exercise.

MH-F-4 Workout Goals. (Satisfied) - An appropriate goal is set for the user dependent on their last performance of the workout selected to help apply progressive overload.

MH-F-5 Create Workouts. (Satisfied) - Users can create workouts with exercises that can be done with or without equipment and input parameters of these workouts such as reps, rep range, sets and weight.

MH-F-6 Workout Split Generator. (Satisfied) - The application can generate users reasonable workout split routines based on their experience, access, owned equipment and time available.

MH-F-7 Edit Workouts. (Satisfied) - The user can adjust the weight used for each exercise in a menu or while working out to make it adaptive to them, also the user can adjust the number of sets in the edit menu, change days or delete a workout.

MH-F-8 Data on Exercises. (Satisfied) - A large amount of data on exercises is available to the user to allow for a good level of customisation in their workouts regardless of equipment access.

SH-F-1 BMR Calculator. (Satisfied) - Users upon saving their metrics in MH-F-1 will automatically get their BMR calculated for them on the ‘Me Page’ to give them an idea of their base metabolism.

SH-F-2 Activity Tracking. (Satisfied) - Steps, distance moved and calories burned are all tracked in the application and viewable on the ‘Today’s Stats’ page. Calories are tracked from steps and weight lifting with respect to the user’s weight.

SH-F-3 Gallery Page. (Not Satisfied) - Not developed.

SH-F-4 Progress Graphs. (Not Satisfied) - Not developed.

SH-F-5 Featured Workouts. (Partially Satisfied) - Workouts are recommended to the user from within the app but there is no linkage to an online database that can change giving the user a fresh set of workouts on a timely basis.

CH-F-1 Forum. (Not Satisfied) - Not developed.

CH-F-2 Rep-Visualiser. (Not Satisfied) - Not developed.

CH-F-3 Altimeter. (Not Satisfied) - Not developed.

CH-F-4 Sleep Tracking. (Not Satisfied) - Not developed.

In summary, all of the must-have features and most of the “Should have” features were satisfied showing a close-to-release level of completeness.

7.2.2 Non-Functional Requirements

Listed below is the evaluation of the non-functional requirements:

MH-NF-1 Usability. (Satisfied?) - The author and his friends believe that the interface is very simple to navigate and use. To see if the application is truly highly usable, the application should be distributed for use case tests.

MH-NF-2 Lightweight Storage. (Satisfied) - The format of file saving is very lightweight meaning not much storage space is taken up on the user's device which also allows for greater performance in SH-NF-2.

MH-NF-3 Portability. (Satisfied) - The application can be used on all of the newest Android models and most of the older ones meaning most people using the Android platform can use it. Support for IOS could increase portability.

SH-NF-1 Availability. (Satisfied) - The application can be fully utilised without the need for wifi meaning it is available just as long as the user has an Android phone that is not out of charge.

SH-NF-2 Performance. (Satisfied) - The application responds in a very quick manner to every operation performed including workout creation. The application only takes a little bit of time to initially launch.

CH-NF-1 Scalability. (Partially Satisfied) - There is room in the application for future developments based on the way the main menus are set out however the use of the CSV format for exercises is not as scalable as an SQL or SQLite database which would perform better for larger data sets.

In conclusion, all of the non-functional requirements were satisfied to some extent with room for improvement on CH-NF-1 and the lack of use case testing for MH-NF-1.

8 Conclusions and Future Work

8.1 Objectives

As referenced in Chapter 7 section 7.1, the application had met its goal of producing a mobile application to log and plan workouts whilst allowing the user to apply progressive overload. With a workout split generator and a large set of exercises provided, the user has a lot to play around with to create a workout suited for them or have one produced relative to their experience and goals. The artefact produced performs very fast, is lightweight and is free from defects besides the 'Workout History' as mentioned in Chapter 6, Section 6.5.3. The author acknowledges it as a tool that can be used by him now to aid in logging and planning workouts rather than frustratingly doing it by hand.

While the application functionality-wise is very close to a release build, the UI could have some enhancements. For example, a better colour scheme could be used using the 6:3:1 rule mentioned in Chapter 5 Section 5.4 rather than the dull colours currently used in this distribution. Many colour schemes were experimented on, however, the author felt they could not execute them well. To make this application release worthy, only the last "should have" requirements should be completed along with the UI changes just mentioned at the very minimum.

8.2 Methodology

The use of an agile-inspired incremental approach allowed the application to be built from the ground up implementing features in an order that made further developments easier. Each functionality was complete and bug-free before the next increment added another feature to the application. A minimum viable product was achieved at the third out of five increments planned.

The testing performed early on at the end of each increment allowed bugs to be detected sooner rather than later in the debugging process allowing them to be patched sooner.

Some informal discussions with peers helped shape some of the UI choices of the application such as green, black or red coloured arrows on a workout summary to give a quick evaluation of performance. As mentioned earlier in Section 7.2.2, use case testing should have been performed to assess the usability of the application.

8.3 Future Works

When the few of the remaining “Should have” requirements are finished and the application is released, future work could be done. The author firmly believes the implementation of a forum section CH-F-1 could be beneficial due to community influence on working out. A lot of workout tips are anecdotal and allow sharing and questioning like on classic forum pages; users can have a greater idea of how to work out based on their experience level and goals. Modes of training such as bodybuilding are considered to have a skill ceiling and the formation of a helpful community on the platform could enhance the user’s experience on it and further supplement their workout planning. In addition, the application could have a Fitbit integration system to yield more accurate results with higher-level gamification implementations such as earning badges or leaderboards as mentioned in the literature review in Chapter 2, on the influence of leaderboards and rewards for motivating users.

References

1. Peterson, M. D., Pistilli, E., Haff, G. G., Hoffman, E. P., & Gordon, P. M. (2010). Progression of volume load and muscular adaptation during resistance exercise. European Journal of Applied Physiology, 111(6), 1063–1071. https://doi.org/10.1007/s00421-010-1735-9

2. Caillois, R. (2001). Man, play and games. University Of Illinois Press ; Wantage.

3. Merit Badges. (n.d.). Boy Scouts of America. https://www.scouting.org/skills/merit-badges/

4. Chou, Y. (2015, March). Octalysis: Complete Gamification Framework - Yu-kai Chou. Yu-Kai Chou: Gamification & Behavioral Design. https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

5. Hunicke, R., Leblanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. https://users.cs.northwestern.edu/~hunicke/MDA.pdf

6. Apple, Use the Health app on your iPhone or iPod touch. (n.d.). Apple Support. https://support.apple.com/en-gb/HT203037

7. FitBit Inc. (2021). Fitbit (Version 3.2.1) [Mobile application software], Apple Store, https://apps.apple.com/us/app/fitbit-health-fitness/id462638897

8. Kaang, M., Marshall, S. J., Barreira, T. V., & Lee, J. (2013, January 23). Effect of pedometer-based physical activity interventions. Taylor & Francis. https://www.tandfonline.com/doi/abs/10.1080/02701367.2009.10599604?journalCode=urqe20

9. Niantic, Inc. (2023). Pokemon Go (Version 0.267. 1) [Mobile application software], Google Play Store, https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&gl=GB

10. Guillemette J, Morris J, (2022). Playfitt app [Mobile application software] https://www.playfitt.ca

11. Karami, A., Dolatabadi, H. R., & Rajaeepour, S. (2013, September). Analyzing the Effectiveness of Reward Management System on Employee Performance through the Mediating Role of Employee Motivation Case Study: Isfahan Regional Electric Company. CiteSeerX. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8e6b6e7462c51d958d18f37fc6eaa57e44e18a22

12. Jia, Y., Liu, Y., Yu, X., & Voida, S. (2017). Designing Leaderboards for Gamification. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3025453.3025826

13. Leap Fitness Group. (2023). Home Workout - No Equipment [Mobile application software], Google Play store, https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment&hl=en_GB&gl=US

14. Fitness22. (2023). Gym Workout Planner & Tracker [Mobile application software], Google Play store, https://play.google.com/store/apps/details?id=com.fitness22.workout&hl=en_GB&gl=US

15. Principles behind the Agile Manifesto. (2011). Agilemanifesto.org. https://agilemanifesto.org/iso/en/principles.html

16. Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. A brief history. Computer, 36(6), 47–56. https://doi.org/10.1109/mc.2003.1204375

17. Functional vs. Non-Functional Requirements: Why Are Both Important? (n.d.). Www.uptech.team. https://www.uptech.team/blog/functional-vs-non-functional-requirements

18. JetBrains. (2013). Android Studio (Version 4.2) [Computer software]. JetBrains. https://developer.android.com/studio

19. Flutter Team. (2017). Flutter (Version 2.8) [Computer software framework]. Google. https://flutter.dev/

20. Microsoft Corporation. (2015). Visual Studio Code (Version 1.63.2) [Computer software]. Microsoft. https://code.visualstudio.com/

21. Adobe Inc. (2021). Adobe XD (Version 43.0) [Software]. Available at: https://www.adobe.com/products/xd.html

22. Adobe Inc. (2023). Adobe Illustrator (Version 2023). Retrieved from https://www.adobe.com/products/illustrator.html

23. Adobe inc. (2022) Adobe After Effects (Version 2022). Retrieved from https://www.adobe.com/products/aftereffects.html

24. Pexels. (2019). Free stock photos · Pexels. Pexels.com; Pexels. https://www.pexels.com

Appendix A: Project Initiation Document

This contains the initial document produced that outlines the project completed.

School of Computing
Final Year Engineering Project

Project Initiation Document

Michael Verdon

Fitness Helper

1. Basic details

Student name:	Michael Verdon
Draft project title:	Fitness Helper
Course and year:	Bsc Computer Science
Project supervisor:	Matthew Poole
Client organisation:	
Client contact name:	

2. Degree suitability

This application will require confidence in programming to produce an artefact that will be user-friendly, robust and run efficiently on android devices. Furthermore, the use of algorithms will be required to make the application a more personalised experience. A back-end and front-end will need to be developed and work seamlessly together as well as working with apis (google maps and web-related).

3. Outline of the project environment and problem to be solved

Fitness is an ongoing endeavour for many people in a world where most people are very busy and can take a lot of time tracking, planning and researching for their goals. It can especially be daunting to know where to start and what to implement in your workout plan in order to achieve your personal goals. Many fitness applications exist but very few have good reviews and this application will aim to achieve what others fall short on.

An application like this will be worth taking on as fitness is especially important nowadays for many young people and old considering most professions require you to be physically sedentary and atrophy muscles and cardio-vascular health to not be optimal.

4. Project aim and objectives

The aim of this project is to help motivate people to take care of their fitness by giving them a tool which can streamline the process whilst forming a community in the process and save them time as time is a limited resource.

In order for this to be a reality, the app will require the use of algorithms to achieve a personalised experience, the use of apis to aid in tracking calories burned and possibly a working database to track progress without taking up space on the hardware.

5. Project constraints

- Testing the applications progression features would require a longer time frame as it is meant to track progress over time as fitness is a long-term endeavour.
- If a forum feature is successfully implemented, to properly test it would require a lot of subjects.
- The application at its fullest would implement a lot of features and there may not be time to develop them due to having to study for other exams.

6. Facilities and resources

The entire project can be completed using my own android device and desktop computer. No special facilities would be required in order to complete this.

7. Log of risks

Description	Impact	Mitigation/Avoidance
Numerous features that are planned for the application.	May not be able to finish them all.	Ensure I group planned features in ranking of priority so I have something operational. Follow an incremental development model.
Android studio is still relatively new to me and could take time to learn some new things.	Delays to the project could result in some aspects not being finished.	Allocate a lot of my time to researching to ensure I can learn what I need to learn.
Electricity bills being more expensive could mean I have to spend less time on my desktop.	Would not be able to work on the project as much as intended.	Create a GitHub repo for my project so I can work on it from anywhere such as the University Library.
The application containing diagrams would be optimal but I cannot make them myself.	Legal issues if I do not get permission unless there are royalty free ones.	Scrap the diagrams if I cannot get permission to use any and guide the user to sources on how to perform exercises.

8. Project deliverables

The application will contain many features and to ensure I am able to create something usable, I will rank these features by priority.

Level 1 - Must have features:
- Save user metrics.
- Workout generator based on interests and goals.
- Manually create a workout routine.
- Calories Burned and step counting with google api.
- Monitor workouts and progress.
- Rep visualiser.

Level 2 - Great to have features:
- Altimeter for calories burned.
- Forum
- BMR Calculator
- Body fat % Calculator
- 1rm Calculator
- Grab deals on fitness products.

Level 3 - Could have but not important features:
- Sleep tracking
- Social media integration/sharing

Documents that will be included in the project:
- Testing.
- Javadoc.
- Requirements specification.
- Use cases.
- Software engineering process.
- Mockups.

9. Project approach

In order to execute this project, A requirements specification along with use cases, mock ups and reviews and assessments of current applications in the market like this and what I can do to make my application stand out. When it comes to developing the application, incremental development seems ideal. There is no client so agile cannot be used. This will be effective as a prototype can be deployed as soon as possible to which it will then be built on and improved.

10. Project plan

In order for the project to be successful, a first prototype of the application should be released as soon as possible to be built on and made the best it could be. Beforehand, a literature review of applications and fitness with app reviews, requirements specification and mockups should be done.

As the application is being developed, the report will be continuously built on and testing documents will be made. Each increment will be documented.

11. Supervisor Meetings

Weekly or fortnightly meetings.

12. Legal, ethical, professional, social issues (mandatory)

The only legal issue would be acquiring diagrams for exercises when a plan is made. This would require permission from the creators and abide by copyright laws however diagrams are not vital.

The application will be locally run without needing to use an email to login and utilising local storage for user metrics and progress so data will not be shared non-consensually and be private to the user.

Appendix B: Ethics Certificate

This contains the ethics certificate issued for this project.

Certificate of Ethics Review

Project title: Fitness Helper

Name:	Michael Verdon	**User ID**:	959856	**Application date**:	20/10/2022 08:31:26	**ER Number**:	TETHIC-2022-103930

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Haythem Nakkas, Dalin Zhou**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Undergraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Matthew Poole**
Is the study likely to involve human subjects (observation) or participants?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: No
Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments: **##supervisorComments##**

Supervisor's Digital Signature: **##supervisorSig##** Date: **##supDate##**