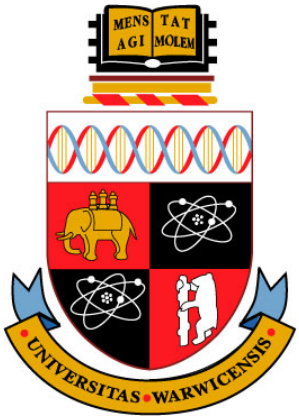


Application Support for Physical Development based on Performance Analytics and Recommendation

CS310 Computer Science

Project Report

William Thompson

Supervisor: Dr. Matthew Leeke

Department of Computer Science

University of Warwick

2016–17

Abstract

In modern day society, the prevalence of obesity is becoming a significant issue. With the advancements in technology reducing everyday effort, how are we going to remain active and increase fitness? This paper covers the production of an iOS application aimed at providing the user with a personalised fitness aid to support them in becoming healthier, using research-backed methods. The system seeks to increase the user's health by providing personalised nutritional recommendations, in addition to supplying a platform that allows the user to safely and efficiently use the gym to complete strength based workouts. To optimise the workouts for each user, workouts are inferred from user entered data and are then tailored to helping them achieve an individually chosen fitness goal. Based on the feedback gathered from testing, the system shows it has the ability to increase the user's fitness. How this was achieved will be covered throughout this paper, showing key areas of research, as well as the significant technical content relating to the construction of the application.

Acknowledgements

The success of this project could not have been achieved without the support and contributions of numerous people. Firstly I must especially thank my Supervisor, Dr. Matthew Leeke, for not only helping to conceive the project but the enormous amount of guidance and time he has selflessly given throughout the project, and in fact my whole time at the university. I have been exceptionally lucky to receive the level of support given, and it is greatly appreciated. My next thanks must go to Professor Graham Martin, for providing a high degree of engagement through the presentation in addition to some productive comments. I particularly appreciate the time he has given to reviewing this report.

I would like to thank further my family and friends who have not only provided a tremendous amount of support and encouragement throughout, but they have also given time to review the project. I have to thank Lucy separately, as she has gone above and beyond in supporting me, from the start of this project to the end, not to mention devoted countless hours to proofreading. I'm certain without the help of these people the project would not be the level it is today, and I'm sincerely thankful to all of you.

Finally, as with any user-focused project, there must be testers. I would like to thank everyone who has supplied feedback to the project, be that officially through structured testing or just with passing comments in the department. All of these observations have helped to produce a more well-rounded solution.

Contents

List of Figures

CHAPTER 1

Introduction

There is a health crisis affecting the UK that is being massively overlooked by its population. Currently over 66% of the UKs male and 56% of the UK's female population are clinically overweight or obese [36]. Solutions need to be found to solve this problem, or at a minimum, some measures must be put in place to prevent the number of overweight individuals from rising further; if nothing is done, by 2050 50% of the population will be clinically obese [33].

1.1 Background Knowledge

This section will be used to cover any prerequisites that are required to understand the areas covered in this chapter. If the areas being discussed are already known, the relevant areas can be skipped.

1.1.1 Obesity

As the fundamental motivation for this system concerns the classification of an individual's weight, a definition of obesity must first be established. The official definition by the NHS UK is as follows:
"There are many ways in which a person's health in relation to their weight can be classified, but the most widely used method is body mass index (BMI). BMI is a measure of whether you're a healthy weight for your height. You can use the BMI healthy weight calculator to work out your score.

For most adults, a BMI of:
18.5 to 24.9 means you're a healthy weight
25 to 29.9 means you're overweight
30 to 39.9 means you're obese
40 or above means you're severely obese

BMI isn't used to definitively diagnose obesity, because people who are very muscular sometimes have a high BMI without excess fat. But for most people, BMI is a useful indication of whether they're a healthy weight, overweight or obese. A better measure of excess fat is waist circumference, which can be used as an additional measure in people who are overweight (with a BMI of 25 to 29.9) or moderately obese (with a BMI of 30 to 34.9). Generally, men with a waist circumference of 94cm (37in) or more and women with a waist circumference of 80cm (about 31.5in) or more are more likely to develop obesity-related health problems." [68]

1.1.2 Personal Health and Fitness

If a person is classified as being in the overweight or obese category, they have a greater chance of developing certain diseases than someone who is in the healthy category. Some of the most common diseases are

type 2 diabetes, coronary heart disease, bowel cancer, breast cancer, strokes and also depression [68]. All of these conditions have a serious affect on an individual's life, with some being classified as potentially terminal.

This is a serious issue for individuals who are overweight or obese. However, one way these individuals can reduce their weight is through exercise, which in turn will reduce their risk of developing previously mentioned diseases. There are a vast array of different ways to exercise, the most common of which is running. Other common ways to exercise involve: sports, both team and individual, yoga as well as strength and conditioning training. Below are some common fitness definitions that have been taken from Zamora's "The Absolute Beginner's Guide to Exercise" [101].

- **Aerobic/cardiovascular activity.** These are exercises that are strenuous enough to temporarily speed up your breathing and heart rate. Running, cycling, walking, swimming, and dancing fall in this category.
- **Maximum Heart Rate is based on the person's age.** An estimate of a person's maximum age-related heart rate can be obtained by subtracting the person's age from 220.
- **Flexibility training or stretching.** This type of workout enhances the range of motion of joints. Age and inactivity tend to cause muscles, tendons, and ligaments to shorten over time. Contrary to popular belief, however, stretching and warming up are not synonymous. In fact, stretching cold muscles and joints can make them prone to injury.
- **Strength, weight, or resistance training.** This type of exercise is aimed at improving the strength and function of muscles. Specific exercises are done to strengthen each muscle group. Weight lifting and exercising with stretchy resistance bands are examples of resistance training activities, as are exercises like push-ups in which you work against the weight of your own body.
- **Set.** Usually used in discussing strength training exercises, this term refers to repeating the same exercise a certain number of times. For instance, a weight lifter may do 10 biceps curls, rest for a few moments, then perform another "set" of 10 more biceps curls.
- **Repetition or 'rep'.** This refers to the number of times you perform an exercise during a set. For example, the weightlifter mentioned above performed 10 reps of the bicep curl exercise in each set.
- **Warm up.** This is the act of preparing your body for the stress of exercise. The body can be warmed up with light intensity aerobic movements like walking slowly. These movements increase blood flow, which in turn heats up muscles and joints. "Think of it as a lube job for the body," Bryant explains. At the end of your warm-up, it's a good idea to do a little light stretching.
- **Cooldown.** This is the less-strenuous exercise you do to cool your body down after the most intense part of your workout. For example, after a walk on a treadmill, you might walk at a reduced speed and incline for several minutes until your breathing and heart rate slow down. Stretching is often part of a cooldown.

1.1.3 Personal Training and Gym Techniques

In addition to the definitions covered in the previous section, some more terms will be discussed which link more specifically to strength training, which are taken from Shakeshaft [83].

- **Circuit Training** This workout combines a series of strength and cardio moves to blast maximum calories and fat.
- **Compound/Functional Movements** Also known as multi-joint or complex exercises, these movements work multiple muscles as a functional unit, promoting stability and maximum calorie burn.
- **Drop Sets** Using this gruelling strength training technique, weight is reduced mid-set, and the exercise continued until exhaustion.
- **Failure** Sometimes failure can be a good thing at least when it comes to resistance training. When training to failure, an exercise is repeated until exhaustion, the point when the muscles pretty much go on strike. While this is one tool for building muscular strength, size, and endurance, proceed with caution, as using this method can potentially increase the risk of injury.

- **Forced Reps** These are extra repetitions at the end of a set that require the help of a spotter (think a bit beyond failure).
- **Interval Training** By alternating bursts of light and intense activity, this popular training method helps maximise fat-burning potential while boosting metabolism and cardiovascular fitness levels.
- **Isolation Exercises** Unlike compound/functional movements, these targeted exercises hit just one muscle at a time.
- **Negatives** Also called "eccentric contraction," this is the act of lowering the weight slowly under tension to the start position.
- **Plateau** Seeing results takes time and practice, and even then, its common for progress to eventually come to a halt. Since the body naturally adapts to the stresses of exercise (especially if performing the same routine daily), try varying the program and increasing the intensity to push past it.
- **Plyometrics** These movements (like broad jumps, vertical jumps, and even explosive skipping) are designed to increase speed and explosiveness while strengthening joints and muscles.
- **Supersets** An approach in which two exercises are performed back-to-back with no (or at least minimal) rest in between. Adding in a third exercise is referred to as a tri-set.
- **Split** A routine that involves dividing up the muscle groups into different training days.

1.2 Motivation

Society as a whole has changed and the vast majority of populations now lead sedentary lives, a change which has contributed to the rise in obesity. It has been suggested that this new sedentary lifestyle is linked to the advancement and the increased use of technology and technological solutions [77]. All technology is designed to aid certain aspects of day-to-day life, but how much help is too much?

With each technological advancement comes an improvement to everyday life that reduces an individuals physical effort. The range of these advancements spans from online shopping with home delivery, to robot vacuum cleaners. Most people would agree that these are great advancements, however, they reduce the amount physical activity it takes to complete a task, driving more and more of the population to complete even less exercise than before. As technology is never going to be designed to make life harder this trend is only going to get worse, reducing activity further and further. This is enforced further by the rise in computer focused jobs, reducing exercise at work as well as at home. One of the largest advancements to effect physical fitness is not a particularly modern one. Television is one of the largest driving factors to stay sedentary, with more and more users sitting and watching TV for hours at a time. This is worsened by the fact that people are not even aware of how much TV they watch, with this quote coming from the TV Licensing company in the UK "People routinely understate how much television they watch by around a third. When asked, adults estimated they watch less than 20 hours a week, but the actual figure is over 30 hours a week. And the amount of time spent watching is increasing." [56]. Streaming services are a modern aspect to TV which are further increasing this motivation to stay sedentary. Services such as Netflix [66] allow users to watch full series of shows back to back without the need for any input, by continuing to keep users sat down by automatically playing the next episode to reduce user effort. The TV services available are a much easier source of entertainment than exercise due to the simplicity that has been added, however, it is not helping the health of the population.

The most pressing reason why a solution of to obesity is necessary is for medical reasons. As previously stated the UK is currently under an overweight epidemic. Unfortunately, this is not limited to just the adults, children are equally as affected and childhood obesity is on the rise as shown in Figure 1.1.

The UK is not alone in its troubles with obesity; a large majority of the developed world is also struggling with this issue. Figure 1.2 shows a selection of countries and their relative percentage of overweight and obese males and females. It is clear to see that a vast majority of the countries have populations where over 50% are obese or overweight. Therefore a large majority of the global population are at risk of developing the associated health conditions discussed previously in Section 1.1.2, showing just how much of a threat obesity is to modern day society.

As afore-mentioned in Section 1.1.2, strength and conditioning training would be one way to improve an individuals fitness, helping to reduce their weight as well as preventing further increase. A major issue

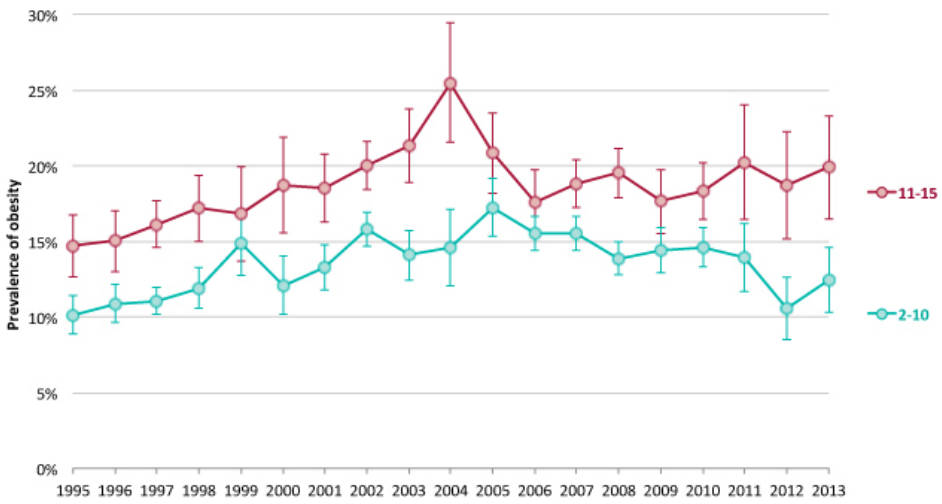


Figure 1.1: Trends in child (aged 2-15 years) obesity prevalence (1995-2013) [35]

that is commonly overlooked when starting strength training is that certain knowledge is required to correctly use the equipment and complete exercises, in addition to planning efficient and effective workouts. In order to correctly plan workouts, maximise results and achieve a fitness goal, an understanding of human biology is required, particularly the knowledge of muscle structure and operation. This is covered in basic exercise and fitness courses such as the one provided by AMAC [2]. This knowledge also helps prevent common injuries brought on by incorrect usage of a muscle or joint whilst in a gym environment. The next requirement is a good knowledge of the equipment that is available in a gym, as this will dictate which exercises or at least which variation of the exercise can be completed. This is important as the same exercise may have different movements and rotations based on the equipment used. In addition to this, a large knowledge of exercises is also required to partner with the available equipment. Even with all of this knowledge a user may still not be able to create effective workout plans to achieve their fitness goal as they will also need a clear understanding of how the number of repetitions and weight of an exercise effect the outcome of a workout. The issue with attempting to understand such information is that it is timely to learn and the supporting material available from a range of sources is very saturated, hence it can be hard to find the relevant information. The predominant issue that arises when trying to understand repetitions and weights required for exercises is that workout routines vary greatly depending on the individual's personal health goal and body structure. Additionally, the routine will have different results when different people complete them as a result of everyone's body composition and genetics being slightly different. Although the degree of improvement in fitness may vary, it is still possible to suggest routines that will benefit an individual's health. This was discussed by Skwarecki in "How Much Does Genetics Really Affect Your Fitness?", where she interviewed a professor of Kinesiology, Stephan Roth [87].

The clear solution to the problems previously discussed would be to get experience from a trained professional, such as a personal trainer. This would be a good fix to these problems as they have in-depth training on how to help someone become fit and achieve their personal goals. However, there is one issue with this solution which is that it is not financially accessible for everyone. Hiring a personal trainer is expensive, with the average price per session costing around £40-50 outside London and £50-60 inside London [79]. To aggregate this over the course of a month, with 3 sessions per week the average cost would total roughly £500, at the lowest stated rate. This is not an option for a large percentage of the population. Despite the price of personal trainer sessions, people can still afford to join a gym; "for the first time ever, member numbers have exceeded 9 million. 1 in every 7 people in the UK is a member of a gym, an all-time penetration rate high of 14.3%" [53]. As personal trainer sessions are not affordable for a large percentage of the populations, a large number of these gym members must be going to the gym without proper support, likely leading to sub-optimal workout sessions, and even potentially damaging their body due to lack of support and education in the area. The large uptake in gym members and the large financial cost of personal trainers means that this area could benefit greatly from a lower cost workout assistant that is easily accessible and provides support in terms of efficiency and safety.

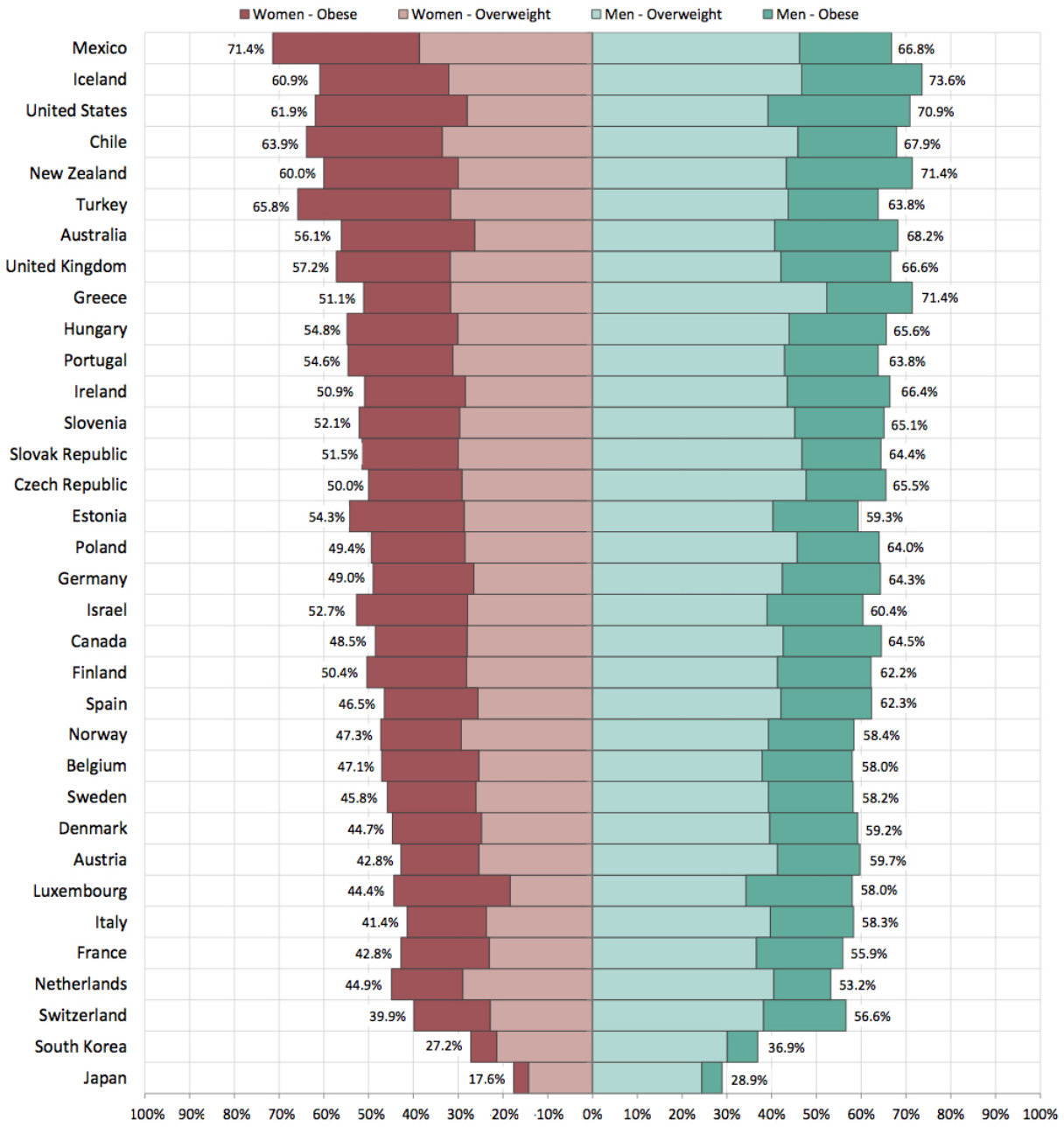


Figure 1.2: International overweight and obesity prevalence in adults by OECD country [34]

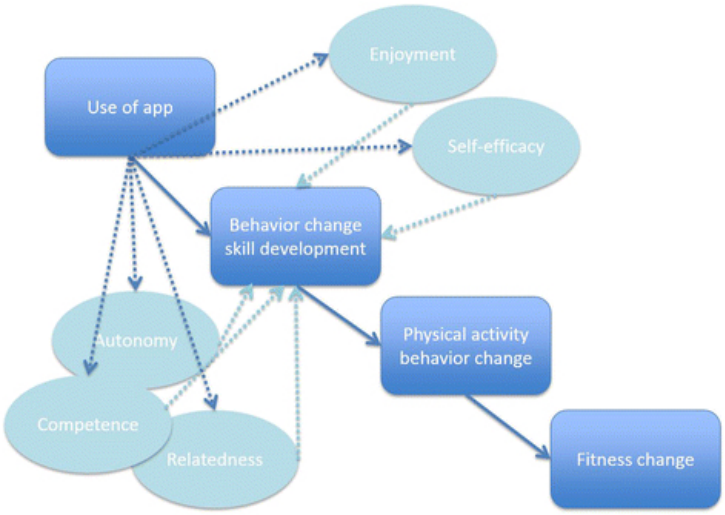


Figure 1.3: "Proposed conceptual model and pathway of how use of the app may influence possible mediators that change physical activity behaviour and consequently influence fitness " [15]

In the UK the number of smartphone users is growing, with approximately 66% of the UK's population owning a smartphone [70]. This makes make smartphones the perfect development target for a low-cost workout assistant as it would be accessible to a lot of people. The other benefit of creating a workout assistant in the form of a mobile application is the versatility it allows when used in a gym environment as it would not be as convenient to use a large device such as a laptop within a gym making the application more of a hindrance than a support. Therefore producing a fitness aid in the form of mobile application seems an appropriate method.

1.3 Project Aims

The overall aim of this project is to provide the user with a mobile application that can support them in using a gym effectively, efficiently and safely. The app should support every user regardless of ability and physical composition, as it is equally important to prevent currently healthy individuals from becoming overweight just as much as it is important to reduce the weight of individuals who are overweight or obese. Figure 1.3 shows how the use of an app may lead to health benefits. To achieve the project's aim a wide variety of tasks will have to be completed including research in the surrounding areas; software development and algorithm creation or utilisation, depending upon the chosen solution to project problems. This allows the project to have suitable depth thanks to the full system development as well as breadth thanks to the research and algorithmic sections.

After work had started on the project it was clear that an aspect of the project had been overlooked, specifically the projects academic focus. This should also be a major aim with the intent to increase the developer's skill level and knowledge. The overall project aim has also shifted slightly, moving the project to a proof of concept based approach as opposed to producing a feature complete solution by the deadline. This will allow key new features to be developed without the need for a breadth of options common to an application of this type.

To try and achieve this goal some areas will be of key importance, and their specific aims will be discussed further.

1.3.1 User Interface

The user interface will be a key area for development for multiple reasons. The first is that this will be the sole way a user will interact with the system, therefore data entry needs extensive thought to promote ease of use. Another reason is that it will provide key information to the user, therefore the way information is displayed also needs to be thought through to ensure it remains simple. Finally, an application is normally judged by the way it looks and how it feels to use, regardless of any unique

features it may have, therefore producing a good user interface could dictate whether or not the app is used.

1.3.2 Content

To enable the app to have a strong grounding as a fitness application the content needs to be strong. It should, therefore, contain a catalogue of exercises, each accompanied by images to show correct operation and a detailed explanation of exactly how to complete the exercise, including key details about positioning and form.

1.3.3 Routine suggestion

The main role of a personal trainer is to guide an individual through a full workout. This includes what exercises to complete in what order, as well as establish how much weight they should be lifting for how many reps. This is all based around the individual's specific fitness goals. Therefore the app should also provide this type of guidance that caters specifically to the user's fitness goal.

1.3.4 Workout Motivation

One of the main reasons for someone stopping exercise is a lack of motivation. This feature aims to provide one of the advantages of having a personal trainer that is sometimes overlooked by other fitness apps. Motivating and encouraging clients is a key role for a personal trainer as this is what will enable the client to keep working towards their fitness goal regardless of how a specific session turned out. The app should gather feedback from a user to establish their opinions on the workout and use this to provide support and encouragement like a personal trainer would do. The aim will be to try to motivate the user to continue to go to the gym. This will be an area of research to decide which methods of motivation are most effective.

1.3.5 Developers Education

This addition is focused around increasing the developer's knowledge of mobile app development and the surrounding technologies used in this area. It will also give the developer experience of a full-scale development and the issues that are associated with such work relating to delays and meeting deadlines, which will carry through to employment.

1.4 Project Stakeholders

For this project there will only be two main stakeholders, this will be the developer and the supervisor. As the end goal will eventually be to distribute the application for outside use, user feedback will also be valuable as they will inevitably be the ones to use the application.

1.5 Report Structure

This report will cover all aspects of the development of the final project. This will be composed of three main areas which are Research, Production and Review.

1.5.1 Research and Planning

Chapters 2 to 5 can be thought of as research-based as well as including any considerations and planning for the project. Chapter 2 will give insight into existing systems already available, as well as looking into the surrounding areas and any other areas specific to this project that fall outside the obvious surrounding areas. Chapter 3 will detail the ethical, social, legal and professional issues that surround this development. Chapter 4 will establish the requirements for the project based on the research previously conducted. Finally, the management of the project will be outlined in chapter 5 to discuss the techniques that will be followed throughout the development. In addition, the tools that will be used in the project will also be covered in this chapter.

1.5.2 Production

This will be the main feature of the report and will encompass chapters 6 to 8. Chapter 6 will discuss the design of the system and how decisions will be made, covering both front-end and back-end. Chapter 7 will then follow on from this and explain how the system will be turned from design to physical solution, including any areas of difficulty. Chapter 8 will cover how the solution will be tested as well as any findings resulting from the testing.

1.5.3 Review

The final part of the report will span chapters 9 and 10 and will reflect on how the project has gone. Chapter 9 will be used to directly compare the project against the original requirements set out to confirm the system is in-line with the original aims. The final chapter, 10 will be used to summarise the full project, as well as discuss further work that could be completed to enhance the project further.

CHAPTER 2

Research

For this project there is a heavy reliance on research, this is due to the scale of the project along with the magnitude of research areas which are needed to meet the previously discussed aims. These areas will be categorised within either Existing Systems or Related Work, each of which will be discussed in greater detail highlighting the specific areas.

2.1 Existing Systems

A large number of fitness-focused apps currently exist within the Apple app store [1]. Some of the most popular apps will be detailed below with information on the apps key features. All of the apps have the same focus which is to increase a user's overall health; this is done with varying approaches.

2.1.1 7 Minute Workout

Figure 2.1: 7 Minute workout app logo [44]

As of 11/10/2016, 7 Minute Workout is the third most downloaded paid app on the app store worldwide, and the most downloaded paid app within the health and fitness category [45]. It has over 1,200 reviews, with an average of 4.5 stars out of 5. The app focuses on those who do not have time to complete a full gym workout or have little spare time. Therefore the app is designed in a way that workouts can be completed in any environment, without the need for additional equipment. Instead of using equipment, the workouts are bodyweight based and have a low completion time. Hence they can easily be fit into most user's schedules without issue. The key features are scientifically proven workouts; full video, audio, image and text instructions for all exercises; tracking of activities; detailed records of past activities to monitor progress. The app store description is also included below to give further detail.

"The "7 Minute Workout" is a research-backed workout program that has become an international hit!

Published in the leading research journal ACSM, and then popularised by the NYT, the"7 Minute Workout" is simple but effective.

Researchers have selected 12 exercises that are performed for 30 seconds with 10 second rest intervals. This high-intensity training with little rest results in higher daily metabolism and is the equivalent of working out for over an hour - for only slightly longer than 7 minutes.

The best part? The exercises are simple to perform, do not require any equipment, and therefore, can be

done anywhere! NO MORE EXCUSES.

This app takes this research-proven workout and guides you through the process. Further, it tracks your results, and makes it fun by allowing you to unlock rewards as you continue working out." [45]

2.1.2 Full Fitness : Exercise Workout Trainer

Figure 2.2: Full Fitness: Exercise Workout Trainer logo [39]

After 7 Minute Workout, Full Fitness is the next most popular workout application on the app store, with a ranking of 37 currently. This application varies considerably from the previously covered app; however, both have the same overarching aim, but their approaches differ. This application is a gym aid with a multitude of material inbuilt, all of which focus around improving fitness. The largest area of this application is a diverse exercise catalogue, which provides a great detail of information, including photo and video demonstrations and detailed statistical analysis of previously completed sets of the exercise. The other main features are pre-set and custom-built routines; body tracking with additional tracking for exercise weight, reps and sets; an inbuilt calorie counter and BMI calculator. The description taken directly from the app store is given below.

" The top selling fitness app of all time has returned! Becoming and staying fit has never been easier with the help of Full Fitness! Hundreds of exercises are explained with clear pictures, videos and text instructions all within the palm of your hand!

We understand personal trainers can be costly, but changing your workout routine is essential for keeping your body from growing accustomed to the same old exercises. If anything, adding some variety keeps things interesting and enjoyable - making it more likely that you will stick with the program.

For under the cost of a cup of coffee, Full Fitness not only provides instructions for hundreds of exercises, but it sorts them by body region, muscle of target, and the equipment needed. Use our easy-to-use exercise builder to create your own custom exercise routine, and then track your progress as you perform each exercise. Full Fitness allows you to log your exercises, view progress graphs, and email or backup the results online.

Not sure which exercises to perform? Use one of our 30 pre-defined routines to reach a particular goal. All of our routines are developed by licensed fitness professionals and come with complete instructions.

If you have no equipment available, or are more interested in cardio or stretches, Full Fitness has you covered! Tracking of cardio exercises, over 40 stretches and exercises and routines that require no equipment are all included! Full Fitness does even more. Track your food intake and body weight, schedule workouts, setup profiles to track more than one user, and more. This is the intuitive and beautiful fitness app you have been looking for!

FEATURES

- *hundreds of unique exercises (more than any other app)*
- *clear images of people doing every exercise with full text instructions*
- *hundreds of video instructions for many of the more complicated exercises*
- *calorie tracker with over 90,000 food items*
- *cleverly designed logging feature to record and track each exercise*
- *ability to add your own exercise and track your progress*

- *exercises ordered by target (abs, arms, back, etc), muscle they target (deltoids, biceps, etc) or equipment they require (swissball, kettlebells, nothing, etc)*
- *30 routines to reach various goals (weight loss, strength, ab definition, golf program, etc)*
- *ability to email your workout logs to yourself, back them up online or view them on the device*
- *graph your workout results to give yourself the encouraging boost you need*
- *stop-watch timer to keep track of your rest times in between sets*
- *weight monitor/BMI calculator and measurements tracker to view your progress*
- *schedule your workouts ahead of time so you never forget your workout routine*
- *ability to track MULTIPLE users*
- *iCloud support and the ability to import content from Full Fitness*
- *workout sharing tool to share your workouts between other Full Fitness users" [39]*

2.1.3 MyFitnessPal

Figure 2.3: MyFitnessPal logo [65]

MyFitnessPal currently stands at the 109th position in the app store. This ranking may seem high, but it is the second most downloaded within the health and fitness free category; therefore it is still highly relevant. Unlike both previously covered applications, this application is focussed on improving fitness through diet. It does this by recommending calorie intake based on the user's weight, height, age and activity level. Throughout the day the user then inputs all foods consumed, of which the application has an extensive catalogue. These foods are converted into calories and macronutrients so the user can see how their diet adds up throughout the day, the aim of which is to prevent overeating in terms of both calories and food groups. The main advantage of this application is the extensive food catalogue, which can be easily searched using a barcode or text entry. Additionally, every user can extend the catalogue if an item is not included, giving a full coverage of food products. This application, therefore, allows the user to understand their eating, promoting a healthier lifestyle by awareness.

" Lose weight with MyFitnessPal, the fastest and easiest-to-use calorie counter for iOS. With the largest food database by far (over 5,000,000 foods) and amazingly fast and easy food and exercise entry, we'll help you take those extra pounds off! And it's FREE! There is no better diet app - period.

- *Consumer Reports #1 rated diet*
- *PC Magazine Editor's Choice Selection*
- *#1 Health and Fitness app for 4 years straight*

Also featured in the NY Times, Wall Street Journal, Wired, USA Today, Family Circle, Marie Claire, NBC, CNET, Shape, the Today Show and more.
USER REVIEWS

- *The first diet tool that has ever worked for me! I've lost 30 pounds!*
- *This is the best calorie counter, free or paid, and I've tried them all.*
- *The food database is HUGE! I've NEVER had a missing food.*
- *This app takes seconds to use, it's that simple.*

WHY WE'RE BETTER THAN OTHER APPS

- *FREE sign up with no strings attached*

- *Track your diet and exercise in less than 5 minutes a day! It's that fast and easy*
- *Largest food database of any iPhone calorie counter over 5,000,000 foods and growing daily*
- *Easiest and fastest food entry remembers your favorites, add multiple foods at once, save and add entire meals, and more. THERE IS NO FASTER OR EASIER APP THAN MYFITNESSPAL.*
- *Connect with over 50 devices and apps including Apple Health, Fitbit, Jawbone UP, Garmin, MapMyFitness, Runkeeper, Strava, Runtastic, Misfit, Withings, and more!*
- *Recipe importer - cook a lot? Our recipe importer lets you visit any recipe on the web and easily import and track it with just a tap! It feels like magic!*
- *Built-in step tracker - iPhone 5S/6/6+ users can track steps right from their phone, no separate tracker required*
- *MYFITNESSPAL WORKS our members have lost almost 200 MILLION POUNDS combined!*

EASIEST APP TO TRACK DIET & EXERCISE

- *5,000,000+ food database of global items and cuisines. Virtually every food you eat is in our database already*
- *Barcode scanner - track a food just by scanning it's barcode. Over 4 million barcodes recognized!*
- *Step tracker - iPhone 5S and iPhone 6/6+ users can track steps and overall calorie burn right from their phone. No separate tracker required!*
- *Track all major nutrients: calories, fat, protein, carbs, sugar, fiber, cholesterol, and more*

CONNECT APPS &DEVICES

- *Easily connects and seamless integrates with over 50 apps and devices including Apple Health, Fitbit, Jawbone UP, Garmin, MapMyFitness, Runkeeper, Strava, Runtastic, Misfit, Withings, and more! Works with virtually every fitness app and device.*

GET SUPPORT

- *Connect with friends and easily track and motivate each other, or meet new friends from the MyFitnessPal community!*

GOALS & REPORTS

- *Received personalized goals based on your individual diet profile, or enter your own goals if you've gotten specific recommendations from a doctor, nutritionist, etc.*
- *Gain insights into where your calories and nutrients are coming from and how to make healthier choices*
- *View charts of your progress over time for motivation" [7]*

2.1.4 Fitbit

Figure 2.4: Fitbit logo [38]

Fitbit's approach to improving fitness is different from all the other applications but takes points from each. For correct use of this application, an external device must be purchased and worn by the user at all times. This device allows tracking of the user's movements and heart rate. Both of these metrics are used to calculate when the user was exercising automatically. The intensity of the exercise can also be checked using the inbuilt heart rate tracking. This tracking allows Fitbit to provide users with simple automatic tracking and recording of exercise. Similar to MyFitnessPal, this promotes a healthier lifestyle by showing the user how active they are. Another benefit is its simplicity as it does not require any manual entry from the user, saving them time. However, there are limitations to this simplicity; Fitbit

cannot track strength training exercises; instead most of the exercise it can track revolves around improving cardiovascular health. The limited tracking is due to the complexity in analysing exercise movements, as well as the fact that entering a weight for an exercise is vital for strength training and cannot be avoided. Further details on the application's features are given below.

" Live a healthier, more active life with Fitbit, the worlds leading app for tracking all-day activity, workouts, sleep and more. Use the app on its own to track basic activity and runs on your phone, or connect with one of Fitbits many activity trackers and the Aria Wi-Fi Smart Scale to get a complete picture of your healthincluding steps, distance, calories burned, sleep, weight, and more.

TRACK ACTIVITY: Accurately record your steps and distance with MobileTrack when you carry your phone. For all-day tracking of stats like calories burned, active minutes, and sleep, pair the app with a Fitbit tracker.

RUN SMARTER: Enhance runs, walks and hikes by using MobileRun to track your pace, time and distance. You can also control your music, get voice cues and use your phones GPS to map your routes. (Continued use of GPS running in the background can dramatically decrease battery life.)

RECORD WORKOUTS: Use your Fitbit tracker to track your exercise, then check the app to see your stats, their impact on your day, and how your performance is improving.

MONITOR HEART RATE: Use a Fitbit tracker with PurePulse to analyze heart rate graphs in the app. Identify trends, manage stress and see the results of your workouts. Review resting heart rate trends to see when your fitness is improving.

LOG FOOD FASTER: Easily log calories with our barcode scanner, calorie estimator, and expanded food database of more than 350,000 foods. See your meal history at a glance, and get nutritional insights.

MEASURE HYDRATION: Quickly log your water intake to make sure youre properly hydrated during workouts and throughout the day.

SET & MANAGE GOALS: Create weight, nutrition and exercise goals, and start a food plan to stay on track. Then get a visual picture of your progress with colorful, easy-to-read charts and graphs.

SEE HOW YOU SLEEP: Set sleep goals in the app, and use a Fitbit tracker to monitor how much time you spent awake, restless or peacefully sleeping.

SHARE & COMPETE: Connect with friends and family by sharing stats, sending direct-messages, and competing on the leaderboard or in Fitbit Challenges.

STAY MOTIVATED: Get a nudge in the right direction with notifications that pop up when youre close to reaching a goal or have already met one.

SYNC WIRELESSLY: Fitbit trackers sync your stats to computers and 200+ leading devices so you can continuously track your progress without needing to plug in.

MANAGE WEIGHT: Connect wirelessly to the Aria Wi-Fi Smart Scale to seamlessly track your weight, BMI, lean mass and body fat percentages, and to see your weight trends over time. " [3]

2.1.5 Project Comparison

This section will outline the areas where this project will differ from the existing systems detailed previously, as well as the key areas of interest within each.

7 Minute Workout

This application has a large usage and has been very successful. A key factor behind the success lies in the simplicity of the solution, as it requires little of the user's time while still promoting fitness. Unfortunately, this is also an area that causes it issues as the simplicity limits what workouts can be done and the level of difficulty that can be reached. This simplicity leaves the user with a very repetitive schedule regarding exercises, which can become uninteresting and reduce the likelihood of user-completion. The time constraint also limits difficulty as weight cannot be increased, only the number of repetitions can be increased. However increasing this would also increase completion time for the workout, hence the difficulty is also capped.

Although the app has shortcomings, it would be foolish to ignore the obvious success of this application; therefore traits from this application should be carried through to the development. One area that should be focused on from this application is the exercise display, as the application has a very clear and informative way to portray this information to the user. This display helps support novice users and also prevents injury from incorrect exercise execution. The area that needs to be factored into the solution is the simplicity of the application; this application makes it easy for people to exercise and this should be at the core of the application produced and influence the design decisions made. Another design that aids the app's simplicity is the step by step guidance through a workout, informing the user when to start, stop and the rest duration. This guidance removes any uncertainty from a workout, as well as ensuring the workout is completed as expected as the number of repetitions and rest durations can significantly affect how successful a workout is. This kind of instruction is very fitting of the app. However, it could cause issues when transferred to weighted exercises where completion time varies, and a set schedule may be hard to adhere to. This instruction should be an aspect further considered in the design of the system.

Full Fitness: Exercise Workout trainer

Full fitness's largest selling point is the extensive catalogue of workouts. This catalogue is a feature that is unparalleled in this area, especially to include the degree of information included for each. The benefit of this feature is that users can train the same muscle in a significant number of variations; therefore workouts can vary greatly and continue to be interesting and not repetitive. This exercise catalogue, therefore, allows a broad range of inbuilt workouts to be included, with the option to create personal workouts extending the range even further. The application places more responsibility on the user compared to 7 Minute Fitness, as it requires them to take control of their workouts including weight, reps and timings. This responsibility could limit newcomers from correctly using the application, as they may be unaware of the best values for the fields given. The application also strays from the fully guided approach, taking a more hybrid set-up of providing workout routines but not workout specifics.

Fitbit

Fitbit focuses on tracking the user's activity with minimal input; this simplicity within the application and the additional wearable were key to Fitbit's success. The wearable application side of Fitbit could be incorporated into this application, by integrating with the Apple Watch. However, auto-recognition of exercises is not built into this platform; therefore an algorithm to calculate active periods based on all movement data collected would be required. This task would be sizeable and is therefore not within reach for the project at this stage. However using the device to help track workouts could be possible with appropriate user input. There are many additional tracking resources for workouts built into Apple Health that could be utilised, such as heart rate tracking.

MyFitnessPal

Diet is as crucial as exercise when completing any fitness regime, as well as being vital in achieving set health goals. Diet should be therefore be incorporated into the system. Due to the time constraints, it will be impossible to create a solution similar to MyFitnessPal within the application itself, as this is a full-scale development in itself. It is more achievable to take basic functionality, such as base calorie intake. The best solution would be to integrate with MyFitnessPal to give the desired diet tracking without the implementation overhead. This integration will rely on approval from MyfitneesPal to access their API; hence it may not be possible to carry this out, and a more simplified approach may be the only option.

Unique Features

The key unique feature for this application will be an ability to generate tailored workouts for the user, based on their previous workouts completed and specific fitness goal. Currently, there does not exist a fitness application which recommends strength based workouts specific to the user. As previously discussed there are general approaches available but no user specific applications. The application produced will include the number of repetitions to complete for an exercise, the weight to lift, the number of sets to complete and the exercises to complete. These fields are something no app currently features as it requires more intricate knowledge of the user, which is hard to gather and use appropriately. The closest feature to this is where an app automatically fills in the weight for an exercise, which is identical to the last weight lifted for this exercise. This automatic filling reduces the effort for the user but lacks any recommendation. Additionally, most of these apps do not tailor the rep count for exercises either; they are usually set for a workout and not tailored to the user's fitness goals. Most apps allow users to create their own personal workouts, but this requires the user to have additional knowledge as detailed in Section **??**. Hence, reducing the necessary knowledge required by the user from the outset should also be focused on; this should be done without limiting the functionality for more advanced users.

2.2 Related Work

Based on the research conducted into the existing systems and the motivation behind the project, research into the surrounding areas that apply to the project will be a focus. These will fall into a number of surrounding categories, all of which are listed below. These will be built upon as the project progresses and more research areas become necessary and apparent.

2.2.1 iOS Development

The most apparent area of research that needs to be conducted is regarding the technologies and skills that are incorporated in an iOS mobile development. Due to the nature of this research, it will be discussed alongside the implementation in Chapter 7.

2.2.2 Usability

The user interface will feature heavily in this development, as well as playing a vital role in the success of the application. The first focus of this will be considering Human-Computer Interaction (HCI). This is defined as "a cross-disciplinary area that deals with the theory, design, implementation, and evaluation of the ways that humans use and interact with computing devices" [49], as well as having "an associated design discipline, sometimes called Interaction Design or User-Centered Design, focused on how to design computer technology so that it is as easy and pleasant to use as possible. A key aspect of the design discipline is the notion of "usability", which is often defined in terms of efficiency, effectiveness and satisfaction" [29]. The secondary definition will be concentrated on for the majority of this research section.

One of the core principles discussed in HCI is interface metaphors which are involved in most modern systems. An interface metaphor is an icon or UI object that has a visual representation of the real world, and the function of this object in the real world is used to explain how this feature of the system operates. A common real-world example of this is using a magnifier icon containing a plus or minus to represent zooming in or out. This principle will be carried forward into the application, as using this technique reduces the time taken for a user to learn how the application functions. The learning time is reduced because the interface metaphors that are shared between applications have the same function and additionally, users are aware of the real-world operation and therefore can infer that this is also how the component will operate [85].

2.2.3 Nutrition

To begin research into nutrition, there are a number of fields that need to be explored as a baseline. To start research in this area, a baseline calorie intake formulae needs to be sourced. For this to be done successfully, some specific goals need to be established so that the user can select what their fitness aim is, and the calorie intake can be adjusted according to these goals. The goals will be weight loss, muscle

maintenance and mass gain. Weight loss as the name describes will be for users who would like to lose weight. Muscle maintenance will be aimed at users who would like to remain the same weight but become more physically active. Finally, mass gain will be aimed at users who would like to add muscle mass. This goal will also increase their overall weight, but this will be muscular weight not due to excess fat. From these defined goals, the base calorie intake should be modified for the specific goal as well as calculating the macronutrients percentages. Therefore these modifications and macronutrients percentages will also be researched.

Base Calorie Intake

As previously stated, a formula for base calorie intake of a user must be researched first. To be accurate, this needs to also take into account their lifestyle as well as age, gender and weight. The paper that most fit this description was "An Easy Approach to Calculating Estimated Energy Requirements" [84]. This provided all the necessary formulae to calculate a base calorie intake which is built around the desired characteristics. This base calorie intake is referred to as total energy expenditure (TEE) in the paper and to successfully calculate this value a number of steps need to be taken, all of which are shown below. Before the calculations are shown a definition of a MET will first be given. "A MET is a numerical value that represents a multiple of the resting metabolic rate for a particular activity. This value applies to the level of energy expenditure achieved during the performance of a specific activity at a designated intensity and provides a way of expressing the total caloric cost of the activity. One MET equates to a rate of O_2 consumption of 3.5 millilitres per kilogram of body weight per minute (ml/kg/min) for adults. MET values between 1.0 and 12.0 represent the typical range of PAL, from light to moderate to vigorous. The MET value applies to the level of energy expenditure achieved during the performance of an activity and provides a way of expressing the total caloric cost of the activity (Δ PAL). [84]

Basel Energy Expenditure for men:

$$BEE = 293 + 3.8 * age(years) + 456.4 * height(meters) + 10.12 * weight(kg) \tag{2.1}$$

Basel Energy Expenditure for women:

$$BEE = 247 + 2.67 * age(years) + 401.5 * height(meters) + 8.6 * weight(kg) \tag{2.2}$$

Impact of each reported physical activity (PA) on energy expenditure is given by:

$$\Delta PAL = \frac{(\frac{(METs - 1) * [\frac{1.15}{0.9} * Duration(minutes)]}{1440})}{\frac{BEE}{0.0175 * 1440 * weight(kg)}} \tag{2.3}$$

Calculating PA for men:
Sedentary: $PA = 1.0$, *when* $1.0 < PAL < 1.4$
Low active: $PA = 1.12$, *when* $1.4 < PAL < 1.6$
Active: $PA = 1.27$, *when* $1.6 < PAL < 1.9$
Very active: $PA = 1.54$, *when* $1.9 < PAL < 2.5$

Calculating PA for women:
Sedentary: $PA = 1.0$, *when* $1.0 < PAL < 1.4$
Low active: $PA = 1.14$, *when* $1.4 < PAL < 1.6$
Active: $PA = 1.27$, *when* $1.6 < PAL < 1.9$
Very active: $PA = 1.45$, *when* $1.9 < PAL < 2.5$

Total Energy Expenditure (TEE) for men:

$$TEE = 864 - 9.72 * age(years) + PA * [(14.2 * weight(kg) + 503 * height(meters)] \quad (2.4)$$

Total Energy Expenditure (TEE) for women:

$$TEE = 387 - 7.31 * age(years) + PA * [(10.9 * weight(kg) + 660.7 * height(meters)] \quad (2.5)$$

Weight Loss

For someone looking to lose fat but maintain muscle, it is recommended that "Caloric intake should be set at a level that results in bodyweight losses of approximately 0.5 to 1%/wk to maximise muscle retention. Within this caloric intake, most but not all body-builders will respond best to consuming 2.3–3.1 g/kg of lean body mass per day of protein, 15–30% of calories from fat, and the remainder of calories from carbohydrate. Eating three to six meals per day with a meal containing 0.4-0.5 g/kg bodyweight of protein prior and subsequent to resistance training" [37]. This is further backed by earlier research from The American Journal of Clinical Nutrition, which found that high protein weight loss diets are more beneficial than standard protein weight loss diets [93]. In addition to the previous research, where the participants initially had a higher muscle percentage than average, another study focusing on overweight participants found that the ratios of macronutrients played a less vital role. However, it concluded that "Reduced-calorie diets result in clinically meaningful weight loss" [80]. Therefore regardless of physical structure, the combination of both of these approaches are proven to be effective.

Muscle Maintenance

As a muscle maintenance plan does not need to lower a user's body weight, the calorie intake should stay the same. Therefore, the research already carried out regarding the calculation of a user's recommended calories can remain unchanged. Further research is only needed based around the macronutrients. There are a multitude of different recommendations available [32] [19] [73] [23] [28]. However, all of these tend to be contradictory. As no paper could be established as suitable for this scenario, it was decided that the macronutrients combination from the weight loss section previously discussed was equally relevant. This is because the aims are still the same, as the user will be engaging in more muscular focused workouts; therefore the protein will be required to help their body repair effectively. Additionally, the fat and carbohydrates are kept at levels that should limit additional fat gain. Due to these reasons, it seems appropriate to use this research. Once sufficient usage has been confirmed this hypothesises can be tested, and the formula can be altered to help more accurately calculate macronutrients for muscle maintenance if necessary.

Mass Gain

There are many very specific sources online for this field [99] [100] [31]. After further reading of the gathered papers, it was decided that *Macronutrient Considerations for the Sport of Bodybuilding* would be used [52]. This article had a strong grounding in the area, showing that it was built upon extensive research in the field and relevant areas, all of which were from established and highly regarded sources. "In summary, the composition of diets for body builders should be 55–60% carbohydrate, 25–30% protein and 15–20% of fat, for both the off-season and pre-contest phases. During the off-season the diet should be slightly hyperenergetic (15% increase in energy intake)" [52]. This summery from the paper provides all the figures needed to recommend an established nutrition plan for adding muscle mass. Based on the additional papers, however, a value of 30% for protein will be used as a majority of the papers favoured higher protein intake than fat and carbohydrates.

2.2.4 Workout Routines

For any workout, the scheduling and selecting of exercises are crucial in the overall effect of that workout. Therefore research will be conducted to find appropriate workouts for all of the fitness goals previous discussed within Section 2.2.3. These workouts should also focus on working all of the muscle groups. There exist a number sources providing workouts tailored to the specific goals, all created by industry

experts. These are published on a number of health and fitness websites, such as Bodybuilding.com [18], MensHealth [60] and MensFitness [59]. All of these provided an ample amount of workouts for the goals, along with rationale to their benefit for the specific fitness goals. The issue with all of these sources and other similar sources is that all content published is subject to copyright, and therefore none of the workouts found may be used within the application. This prompted a shift in the research in this area; instead, research was conducted on the desired traits of the workouts specialised to a specific fitness goal, which will be used to help design routines for the application.

Based on research into this area, the most highly regarded resources was Designing Resistance Training Programs [40]. A significant amount of background to the area is provided within this paper, which will be used to enhance the product further. Key points from this will be discussed further, along with additional focused material from other specialised journals.

Sets

Workouts that use multiple sets per exercise are shown to provide greater gain in strength [22], therefore for all of the fitness goals, multiple set exercises should be included to maximise the muscular gain from completing that exercise. A set range of 2–6 seems to be the most recommend range from the sources. Additionally shorter rest periods have been shown to reduce training volume significantly [63]; therefore rest time should also be kept optimal to reduce the loss in volume for a specific exercise. For weight loss exercises lowering the rest period will increase the intensity of the workout which will promote fat loss, whereas for mass gain maximum volume is necessary to build the optimal amount of mass, requiring longer rest periods.

Repetitions

Each of the different goals will benefit from a different repetition range. For instance "Maximal strength improved significantly more for the Low Rep group compared to the other training groups" [21], which is further enforced by"Low-repetition, high-resistance exercises produce power." [27]. These show that a lower rep range will provide the maximal benefit for mass gain workouts. However, "High-repetition, low-resistance exercises produce endurance" [27], hence a higher rep range will give better muscular endurance, benefiting the muscle maintenance plan. For weight loss, an even higher rep range is recommended as this again increases intensity which promotes fat loss [95], although this should remain within the 1–20 range which is proven to increase strength [40].

Weight

Currently, there does not exist any papers that give a set weight that a user should lift to help achieve any of the goals. Instead, the weight should be relevant to achieving the desired repetitions for the specified sets. This is all user specific, and in fact, even for a given user, this value will fluctuate over time as the user's fitness level increases and decreases. The recommendation of weight is still an aim of the project, hence how weight should be inferred will be discussed further.

2.2.5 Inference

As no current solutions exist that infer workouts for a user, there is no set way for this to be done. However, there are formulas and methods that exist to predict the maximum weight a person could lift for an exercise given. This is an area that could be used to help infer correct weight for a user; therefore this will be researched further.

One Rep Maximum

"A repetition maximum, or RM, is the maximal number of repetitions per set that can be performed in succession with proper lifting technique using a given resistance. Thus, a set at a certain RM implies that the set is performed to momentary voluntary fatigue usually in the concentric phase of a repetition. The heaviest resistance that can be used for one complete repetition of an exercise is called 1-RM. A lighter resistance that allows completion of 10, but not 11, repetitions with proper exercise technique is called 10-RM" [40]. A 1-RM can be used as an upper limit to calculate the desired weight for a set number of repetitions, as a percentage of the 1-RM.

It has been shown that using a 1-RM formula for estimating the weight to lift for a given exercise is beneficial. "Obtaining a one-rep max offers a safe, practical, and reasonably accurate means of qualifying the muscular strength of large numbers of people in an inexpensive, convenient, and time-efficient manner" [20]. Additionally, "Results indicate that the 1-RM prediction equations could be used to determine the maximum load at the bench press exercise in subjects with low strength training experience." [64]. These show that using a 1-RM calculation for weight is appropriate and can produce the desired result in a manner that is fitting with the design of the application.

There are two main ways to test a person's 1-RM for an exercise, the first measures 1-RM directly through maximal testing, where a person attempts to lift heavy weights until they reach a weight they cannot lift correctly. The second measures 1-RM indirectly, through sub-maximal estimation, this approach is preferred, as it is safer, quicker, and less unnerving for inexperienced exercisers [57]. However, it has been shown to underestimate the actual 1-RM [50]. Other methods also exist but are less commonly used due to the difficulty in calculating, one such example uses anthropometric characteristics to predict the 1-RM, "Anthropometric characteristics are traits that describe body dimensions, such as height, weight, girth, and body fat composition. The physical therapist uses tests and measures to quantify anthropometric traits and to compare an individual's current data with his or her previous data or with relevant predictive norms" [58]. Another example of this is to use an accelerometer to measure the speed at which a weight is lifted to calculate the maximum weight [74]. This second approach has been proven to provide accurate results, and therefore could also be a worthwhile approach if an accelerometer is available. Based on the target market for the application containing inexperienced users, a sub-maximal approach to 1-RM estimation seems most appropriate to prevent injury and limit excess hardware.

There currently exist a number of formulas to estimate a person's 1-RM using a sub-maximal approach. Some of the most common are listed below [86].

Brzycki:

$$weight * \frac{36}{37 - reps} \tag{2.6}$$

Epley:

$$weight * (1 + 0.0333 * reps) \tag{2.7}$$

Lander:

$$\frac{100 * weight}{101.3 - 2.67123 * reps} \tag{2.8}$$

Lombardi:

$$weight * reps * 0.1 \tag{2.9}$$

Mayhew et al.:

$$\frac{100 * weight}{52.2 + (41.9 * e^{-0.055 * reps})} \tag{2.10}$$

O'Conner et al.:

$$weight * (1 + 0.025 * reps) \tag{2.11}$$

Wathan:

$$\frac{100 * weight}{48.8 + (53.8 * e^{-0.075 * reps})} \tag{2.12}$$

The most popular of these formulas is Brzycki [76], as it is used in scientific journals as the go-to formula for calculating 1-RM. Hence, this will be the proposed method used for 1-RM calculations within the application.

2.2.6 The Cold Start Problem

The cold start problem is one that effects most recommendation systems. The problem occurs as the system has no information about a user before starting; therefore no recommendations can be made for this user. There exists a number of techniques to solve this in common recommender systems. However, due to the nature of the suggested usage of this application, no such solutions exist that can be applied to this specific system. Due to this research cannot be carried out to find a solution for this case; instead a proposed solution will be discussed later in Section 6.3.2 of Design, to try and cater for this problem.

2.2.7 Muscular Decay

Muscular decay is defined as "an exercise-induced reduction in muscular performance" [97]. This decay will have an effect on all exercises completed within a workout and therefore needs to be considered. It is easy to understand that as a muscle is repeatedly worked with little rest, its level of performance will be limited. Consequently, the position in a workout schedule that an exercise occurs will have an effect on the maximal power the user will have and thus this should also play a factor when inferring the weight a user should lift for an exercise. However, there is much debate as to a specific definition of fatigue and its exact effects as well as how it is measured. These contradictions cause issues when deciding on how these issues should be resolved by the system.

To be able to factor this into the inference, a way to calculate muscular fatigue is first required. Williams later goes on to discuss how muscular fatigue is measured: "Two different measurement models are generally used to quantify such impairment. The first consists of quantifying the reduction of power output during real exercises such as cycling or running (generally at a maximal intensity), where dynamic performance decline could be attributed to reduced force and/or velocity. The second model is based on the assessment of maximal isometric muscle fore-generating capacity before and immediately after real or simulated exercise, to describe the decline in static muscle performance." [97]. Of course, there are other sources which give other examples of how it can be measured, none of which seem conclusive and all of which give varying results. Because of these reasons, no conclusive measurement technique can be incorporated into the application; instead, a custom solution will be proposed within design Section 6.3.2.

2.2.8 Workout Motivation

For a person to maintain a healthy lifestyle, their fitness regime also needs to be maintained. This regime can be difficult to follow, especially when motivation to keep exercising runs out. A common example of this can be seen at the start of a new calendar year when a mass of people join gyms nationwide, only to lose interest weeks later. This loss of interest is an issue that needs to be considered by the application, but first, it needs to be considered how a lack of motivation would be detected.

If a person were to plateau while under the instruction of a personal trainer, this would be apparent to the personal trainer through either a discussion with the client, or through the client's body language. The application will not be able to detect body language. However, it could detect their state based on messages using sentiment analysis.

2.2.9 Sentiment Analysis

Sentiment analysis is a form of computationally distinguishing the opinions of a piece of text. If the user were to enter a message after each session, this would give the system a chance to analyse the user's current motivation level. This message could then be used to help motivate the user accordingly.

There are multiple ways sentiment analysis could be incorporated into a project, and these are: custom-built algorithms, use of existing algorithms and existing systems. Due to the time constraints placed on this project, it would be infeasible to create a custom algorithm. It is also likely to be unnecessary for the use case. Due to the simplicity of using an existing system, this option will be considered first, with the implementation of an existing algorithm remaining an option if no suitable option is found during research.

There exists a broad range of services offering professional solutions to sentiment analysis. The first researched was Google's cloud platform [41]; this allows sentiment analysis to be completed, however, the preliminary learning of the data needs to be completed first. This learning increases the overhead of the task and specialises on the data more than is necessary for this product, therefore this service will no longer be considered. The next service to be examined was Microsoft's Azure service which allows sentiment analysis [62]. This service has a simple to use platform, strong reviews and accurate results. However, the cost of this service is impractical for this type of project; therefore this solution is also infeasible. There were other services that also had this problem, such as PreCeive API [92]. IBM's Bluemix [43] does allow 10,000 characters of free requests making it a viable option for testing the concept within the application. However to be used at full scale the pricing again would not be appropriate for the app; therefore a different solution would have to be found, and the overhead of the full implementation of this solution would be required. Hence a more appropriate solution should be found through further research if possible. If no such solution is found, this service shall be used to show the concept. The

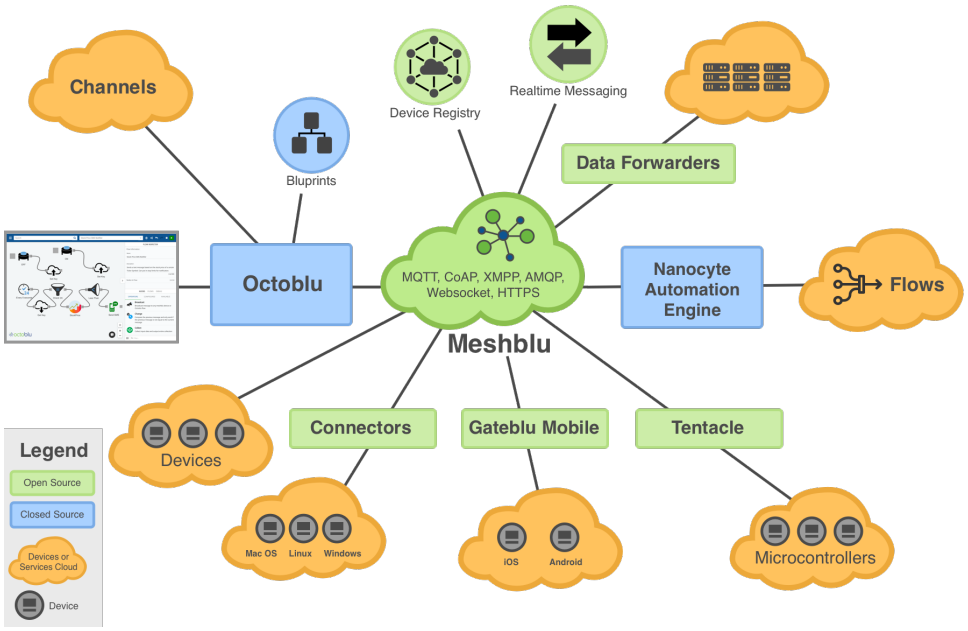


Figure 2.5: The top level system architecture of the Octoblu platform

final solution to be considered was Octoblu [69]; although the core function of this platform is to connect Internet of Things (IoT) devices, there exists multiple components within the platform that perform operations, one of these is a sentiment analysis component. This component allows text to be converted into a value between -5 and +5 depending on how positive, or negative the message is. Although this does not provide as sophisticated output as the other solutions, this platform is free to use and provides enough information to distinguish a user's mood which is the goal. Additionally, the developer has experience working with this platform which will allow a faster implementation. Further details about Octoblu will be discussed next to ensure it is the most appropriate system for this problem.

2.2.10 Octoblu

Octoblu is self-described as being "a full-stack Internet of Things messaging and automation platform that enables companies to create IoT services with secure real-time exchange of data.

Octoblu's IoT services are built on our open source Meshblu platform, an open communications and management platform that supports a variety of protocols for physical devices to communicate seamlessly with each other, people, and web services. Through public, private, or hybrid clouds users can connect, design, process, and analyse the flow of information. All services have been designed through a robust security and right management architecture." [69].

Octoblu is highly configurable and can be adapted to handle most tasks by modifying the systems architecture, which is shown in Figure 2.5. Adapting this will be avoided initially as the core functionality for sentiment analysis is provided already, however the ability to extend the platform further if required is useful, as this allows the feature to be extended without a significant re-implementation overhead. Additionally, the breadth of alterations possible make it perfect for this feature, as it can be extended with the future development plans past the project's initial deadlines.

The are other reasons why Octoblu fits the use case past simplicity and extensibility. Due to its external nature, it takes the processing overhead off the mobile device which benefits power consumption and processing time. The service is fully tested and established allowing integrity to the results produced. The additional components also make the delivery of the motivation to the user more attractive, as it provides many different forms, such as: email, SMS, Instant messaging and web requests. These are just the most common delivery options. The final decision on which to use will be discussed in design, but the breadth of the options further enhance this platforms ability to complete the task.

Based on the research gathered, the issues surrounding the project will be discussed.

CHAPTER 3

Ethical, Social, Legal and Professional Issues

With any system development undertaken it is important to understand all of the issues relating to that development. These issues should be considered carefully and be brought into the development.

3.0.1 Ethical Issues

There is one main ethical issue raised by this project which is linked to the storage of personal information. Due to the nature of the app, it will require lots of personal information about the user, details which they are likely unwilling to distribute or share. It is, therefore important that this information is stored in a way that the user is comfortable with and most importantly in a way that it is not shared with any other parties. There is one exception to this which will be workout reporting where some details will have to be shared to provide workout feedback. The details shared for this feature should be kept to a bare minimum, and not include any unnecessary personal details. Additionally, any services used should also be researched to ensure that they meet these same standards. Finally, the user should be allowed to select whether or not they would like to use any features that require information to be shared.

3.0.2 Social Issues

The use of technology to aid an individual's fitness is still in its early stages, although usage is rising, especially in the wearable devices area. Due to its recent uptake opinions on both effectiveness and social acceptability are mixed. There are numerous articles supporting the use of technology to increase fitness, including credible sources such as the NHS (National Health Service UK). It states that using technology can help improve the outcomes, therefore creating a healthier overall society [67] [26]. This even extends to how socialising physical activity through technology to incorporate social accountability and communities, further encourages a person to be healthier [54]. Of course not all opinions are in favour of the use of technology in increasing fitness, there are also arguments against it. One thesis that analysed a small sample set of users, showed that using technology as a fitness aid actually had a detrimental effect on results in comparison to those who did not use technological aids [48]. It is clear that opinions are split on whether or not technology should be used in the fitness industry, and the split should be taken into account during development. However, it is worth noting that usage of the application will be the consumers choice and they should therefore accept any social accountability associated. Although opinions are split on the outcome of using an application for physical support, there does not appear to be any overwhelming stigma against an individual using them.

3.0.3 Legal Issues

The main legal issue to focus on is the "Data Protection Act 1998" [42]. This is because the system will store and manipulate the user's personal information, therefore extra consideration needs to be taken to ensure this legislation is followed. Additionally, it is important to understand how any third party

services will use any personal information it is given. The user should also be aware of exactly how their information will be stored.

The next clear legal issue is regarding liability, especially as there is an opportunity for a user to injure themselves whilst using the application. To avoid the app being liable for any injuries incurred whilst using the app, a disclaimer needs to be added that removes any responsibility from the app and places it on the user. To help the user avoid injury recommendations could be added such as recommending asking a member of staff at the gym if there is any uncertainty in the completion of an exercise.

Copyright is a legal issue that should be considered regardless of project, as this can be easily infringed if careful practices aren't followed. If any information or tool is being used from outside sources, the rights usages need to be checked clearly to ensure that the material is legally usable for this project. If such data is used, consent for use of that data must be provided in a written form, or an online declaration must exist stating that it is open-source.

The final legal consideration is patenting. Any material covered by a patent cannot be used; if it is used, legal action and a significant fine could be incurred. Therefore the application should be checked throughout to ensure no patented materials are used.

3.0.4 Professional Issues

The system has a user focus and it is therefore important that the user's sense of security is maintained during use so that a user feels they can trust the app. Additionally, the developer should act in a manner set out by the BCS (British Computing Society) in its code of conduct [89].

Now a suitable level of research has been conducted, and the issues affecting the project have been raised; a set of requirements for the project to achieve will be covered.

CHAPTER 4

Requirements

For any project to be successful, it is important to have a set of well-defined requirements. These requirements will ensure that the system stays on track during the development and that it does not sidetrack from the initial plan. However, as the project still has uncertainties, the requirements will also be adapted to suit the needs of the system as it develops and more is learnt about the feasibility of features. This should not affect any of the overarching aims of the project. In this section, the initial requirements fro the system will be covered in a simplistic form that does not require technical knowledge. Then more in-depth technical requirements will be established. If any requirements are added or removed for the project, the reasoning behind these changes will be covered.

4.1 Initial Requirements

As previously stated this section will cover the initial plan for the system's functionality for April 2017. The requirements will be covered in language that is accessible to someone who does not have specific technical knowledge in this area.

4.1.1 Functional

F1: *The system should allow a user to enter details about a workout, such as exercises completed, number of sets, number of reps, weight lifted and workout review*
F2: *The system should store data entered by the user in a persistent form*
F3: *The system should allow user to select a preferred fitness objective, e.g. weight loss, muscle gain*
F4: *Add ability to retrieve user data from other apps, such as my fitness pal*
F5: *Integrate with Apple health to retrieve user information as well as writing workout to data*
F6: *An option to use fitness functions built into Apple watch should be given, to gain a larger amount of information and insight into the workout*
F7: *A flow should be created on Octoblu to handle sentiment analysis of message workouts*
F8: *The system should have the option to send information to the previously specified Octoblu flow*
F9: *Create a set of messages to send depending on result of sentiment analysis*
F10: *The system should employ some inference techniques to suggest workouts for a user*
F11: *The system should have some way to find a baseline of a users abilities for use by the inference mechanism*
F12: *The system should provide some information on a recommended diet based upon the fitness goal already entered. Equivalently this could be done by linking to another application which focuses on this*

4.1.2 Non-Functional

NF1: *The system should have a responsive UI, such that the user is never unaware of system processes*
NF2: *The system should have a simple UI that can be used by users with a range of technical ability*

NF3: *The system should be modular to allow for further or improved functionality*
NF4: *The system should have a low time solution to recording exercises, reps, sets and weight*

4.2 Updated Requirements

After starting development and further research, it was clear that the original requirements were not comprehensive enough. Also, they contained requirements that are out of reach for the time-frame of this project. Therefore this section will outline the reviewed requirements to encompass a more realistic project achievement, as well as adding the necessary details. These requirements will be described in a manner that maintains consistency with the original requirements already discussed. Therefore the numbering convention will remain where any removed requirements will be marked as redundant, and the new requirements will be added to the end of the list. Again these will be phrased in a less technical manner; additionally, the motivation behind these changes will be covered after the new requirements have been stated.

4.2.1 Functional

F1: *The system should allow a user to enter details about a workout, such as exercises completed, number of sets, number of reps, weight lifted and workout review*
F2: *The system should store data entered by the user in a persistent form*
F3: *The system should allow user to select a preferred fitness object, e.g. weight loss, muscle gain*
F4: *Redundant*
F5: *Integrate with apple health to retrieve user information as well as writing workout to data*
F6: *Redundant*
F7: *A flow should be created on Octoblu to handle sentiment analysis of a user's message review of workout*
F8: *The system should have an option to send information to the previously specified Octoblu flow*
F9: *Create a set of messages to send depending on result of sentiment analysis*
F10: *The system should employ some basic inference techniques to suggest workouts for a user*
F11: *Revised*
F12: *The system should recommend a user's daily calorie intake, corresponding to the user's lifestyle, which should be backed by scientific research*
F13: *The system should employ some inference techniques to suggest the specific parameters of a workout, such as number of reps, weight to be lifted and number of sets to complete for the exercise*
F14: *The system should feature a number of inbuilt workout routines, to allow the system to gather metrics about a user*
F15: *There should exist a number of inbuilt workouts tailored for each of the specific fitness objectives, which are backed by scientific research*
F16: *Employ some learning technique to stop the user excessively using a specific workout routine*
F17: *The base calorie intake set out in 'F12' should be modified to be specific to the user's fitness goal*
F18: *The system should adhere to the characteristics of a Modal View Controller (MVC) system development*
F19: *There should be a View Created for each function of the system, which should be linked to an appropriate model*
F20: *The calorie intake set out in 'F12' should be used to calculate the recommended percentage consumption of macronutrients*
F21: *Maintain a constant styling throughout the UI of the application*

4.2.2 Non-functional

NF1: *The system should have a responsive UI, such that the user is never unaware of system processes*
NF2: *The system should have a simple UI so it can be used by users with a range of technical abilities*
NF3: *The system should be modular to allow for further or improved functionality*
NF4: *The system should have a low time solution to recording exercises, reps, sets and weight*
NF5: *The developer should have a greater knowledge of iOS development*
NF6: *The application should limit hardware resources used, to a level that is in-line with other applications of this type*

4.2.3 Alterations

As can be seen above certain aspects of the project have been modified to reflect the current project progress, each alteration will be covered further below.

F4: *Add ability to retrieve user data from other apps, such as my fitness pal*
This requirement has been removed, as, despite an attempt to gain access to the MyFitneesPal API, no response has been received from the company. Without this, it is impossible for this feature to be integrated into the application, therefore, the requirement has been removed. If the request is granted in the future, a review of this feature will be possible, and at that point, a decision will be made as to whether or not the feature can be implemented; this will depend on the complexity of the integration, the current state of the project and the remaining time for the project.

F6: *An option to use fitness functions built into Apple watch should be given, to gain a larger amount of information and insight into the workout*
Once development was underway, it was evident there had been an oversight when predicting the time requirements for creating the system. Once further research had been conducted about watchOS it was clear that incorporating this would require a full-scale development for the watch application. The time-frame for this project makes it infeasible to take on such a development; therefore this requirement has been removed. However it is not completely ruled out, and if the project is finished ahead of schedule the incorporation of this could be considered, again this would be dependent on workload and time constraints.

F10: *The system should employ some basic inference techniques to suggest workouts for a user*
To implement this feature to allow the same recommendation as a personal trainer, a significant amount of user specific data would be required. Gathering this amount of information will be timely, as well as also being complex to implement precisely dependent on all the information gathered. Therefore a simplified approach is required whereby appropriate recommendations are given, but they are not as finely specified as a personal trainer may be able to give.

F11: *The system should have some way to find a baseline of a users abilities for use by the inference mechanism*
This requirement has been replaced by requirement 'F14'. These requirements have very similar outcomes; however, the motivation and usage are different therefore the requirement has been changed to portray this. 'F11' was focused around finding a baseline for whole workout recommendation, whereas 'F14' is based around providing pre-built workouts to find exercise specific information. This will allow for progression towards adapting the specified exercise for the user's goal.

F12: *The system should provide some information on a recommended diet based upon the fitness goal already entered. Equivalently this could be done by linking to another application which focuses on this*
This requirement was updated as it was decided that the application should focus on providing the nutritional information itself as opposed to using another app's functionality, as this method would be more reliable and reduced the dependency on other applications. Additionally, the integrity of the information provided can also be confirmed, and be based on published scientific research. Therefore the system will still provide information about daily calories as originally planned, however, the method of providing this will just be altered.

F17: *The base calorie intake set out in 'F12' must be modified to a user-specific fitness goal*
To enable the app to provide accurate nutritional recommendations after 'F12' was amended, additions were required to enable the application to personalise these recommendations to the user. Therefore this requirement was added to ensure that the nutritional information supplied is custom to the user's body composition, lifestyle and their personal fitness goal.

F16: *Employ some learning technique to stop the user excessively using a specific workout routine*
As 'F14' states that inbuilt workouts are now to be included in the application, a method to prevent overuse of one workout needs to added, to guarantee that the user does not train ineffectively, repeatedly completing the same workout.

F18: *The system should adhere to the characteristics of a Modal View Controller (MVC) system development*
Although not completely necessary, this requirement was added as a way to ensure correct programming practices are followed by the developer.

F19: *There should be a Model and View Created for each function of the system*
As with 'F18', this requirement was necessary to ensure correct programming and development practices are followed throughout the development.

F20: *The calorie intake set out in 'F12' should be used to calculate the recommended percentage consumption of macronutrients*
Further to the nutrition features set out in 'F12' & 'F17', research found that macronutrients are equally as important as calorie intake. Therefore the amount of macronutrients the user should eat should also be calculated. As with 'F17', this should be focused on the user's specific fitness goal, as this will affect how the percentage of each is broken down.

NF5: *The developer should have a greater knowledge of iOS development*
This non-functional requirement was added to line-up with the new overarching project aim of increasing the developer's knowledge and ability in mobile application development, discussed previously in Section 1.3.

NF6: *The application should limit hardware resources used, to a level that is in-line with other applications of this type*
This requirement was added to ensure that the application performs suitably on the device and that it uses the available resources appropriately. This includes the processor, memory, secondary storage and most importantly battery. This requirement has been listed as non-functional, as no specific levels are set for the application to achieve in each area. Instead, each level must be suitable for the task being completed, and the value must be within a reasonable level.

4.3 Technical Requirements

This section will cover the requirements stated before in Section 4.2.1. However, they will now be converted into more technical language, breaking down each requirement into the specific tasks that are required to implement the feature, therefore removing any slight ambiguities. The previous numbering convention will be maintained, but unnecessary requirements will be skipped as they are now redundant. Additionally, requirements may be broken down further into subsections to give the required technical detail in an easier to understand form.

4.3.1 Functional

F1.1: *Create an object to store metrics of a workout, which includes: exercises, sets, repetitions, weight lifted, workout review*
F1.2: *Create an exercise object that stores specifics about an exercise, such as: description and personal best*
F1.3: *Create a view in the UI that displays a workout, which allows user input of metrics, which can be passed to the workout model via a controller*
F2.1: *Create a database schema that has tables to workout an exercise*
F2.2: *Create a method to connect a model to the database*
F2.3: *Create a query to retrieve exercises based on muscle groups*
F2.4: *Create a query to retrieve workout information for a given day*
F2.5: *Create a query to store workout data at the end of a workout*
F2.6: *Create a query to store and retrieve user profile information*
F2.7: *Create a query to store and retrieve user settings*
F3.1: *A view should be created in the UI to allow a user to choose one of the following fitness options: Weight Loss, Muscle Maintenance, Mass Gain*
F3.2: *Create a class to store specifics about a user*
F3.3: *The chosen option from the view should be passed to a model which assigns this as the users*

specified goal in a user class
F5.1: *The user class should use API's provided to retrieve information about the user from apple's health kit app.*
F5.2: *An option needs to be added in a settings screen that allows the user to specify whether or not health kit should be used to gather information*
F7.1: *A flow should be created on Octoblu with the appropriate "things" to handle sentiment analysis and message passing*
F8.1: *An option needs to be added to the settings view to display whether or not Octoblu is initialised*
F8.2: *If Octoblu is initialised the workout review message should be sent to the pre-defined flow*
F9.1: *Create a minimum of 5 messages to cater for the range of different sentiment values*
F9.2: *Choose the correct form to return the message to the user; either text, email or sms*
F10.1: *An extensive knowledge base needs to be created to store rules about the exercises*
F10.2: *The knowledge based should be referenced at the creation of workouts to check none of the rules are violated*
F10.3: *At the end of a workout the rules should be referenced and possibly updated to reflect any rule established*
F12.1: *A view needs to be created to display a users nutrition*
F12.2: *A value for a users recommended calories needs to be calculated based on that specific users personal information and lifestyle*
F13.1: *Based on the knowledge base specified, numbers of weights, reps and sets should be provided for all exercises that have a sufficient data*
F13.2: *An alternate metric needs to be given for all exercises that cannot be inferred*
F14: *A varied workout needs to be created to gather information about the users current level, to be used for future inference*
F15.1: *At least 3 workouts need to be created for those with a lose weight fitness goal*
F15.2: *At least 3 workouts need to be created for those with a muscle maintenance fitness goal*
F15.3: *At least 3 workouts need to be created for those with a mass gain fitness goal*
F16.1: *The system should inform the user if they have completed a workout a set number of times, and suggest that they carry out a different workout*
F16.2: *An alternative workout should be suggested based upon the workout chosen and the users current goal, alongside other workouts previously undertaken*
F17.1: *The calories shown on the nutrition screen should be adapted to meet the requirements of the fitness goal of the user*
F18.1: *The system should adhere to the characteristics of a Modal View Controller (MVC) system development*
F19.1: *There should be a view created for each function of the system, which should be linked to an appropriate model*
F20.1: *The nutrition screen should have sections to display each of the following macronutrients: Protein, Carbohydrates, Fat*
F20.2: *The percentage of each of the macronutrients needs to be calculated based on the recommended calories, as well as the fitness goal of the user*
F21.1: *Throughout the app a constant colour scheme needs to be used*
F21.2: *Every button within the app should follow the same design pattern*
F21.3: *Elements with similar functions should share similar design characteristics*

4.3.2 Non-functional

Due to the nature of the non-functional requirements, these do not need to be converted into more technically detailed requirements for the developer. Therefore the non-functional requirements for the project will be the same as the ones previously stated in Section 4.2.2.

4.4 Project Constraints

Multiple constraints are imposed on this project, all of which fall within the following categories: hardware, software and management. Each of these categories will be discussed further covering all the constraints that fall within the category.

4.4.1 Hardware

The main hardware constraint placed on the project is the limited resources available in a mobile device. These constraints are further enforced by the effect of high processing tasks on the device's battery. Limiting the effects of this has already been covered as a requirement for the project, stated in 'NF6'. Due to these constraints, the produced code needs to be as efficient as possible; additionally the level of inference computable on the device will also be limited. A solution to work around this would be to offload processing to a separate server if necessary; this will be discussed later in the design and implementation chapters if it is required. There are also hardware constraints that affect the hardware that can be used; the core functionality should be limited to functioning with just the use of the mobile device. Therefore additional devices cannot be used, as said devices are unlikely to be accessible by all users.

4.4.2 Software

There are many software constraints on this project, the first one is due to the use of external libraries and packages. Both of these will be used to provide functionality with minimal implementation time. Certain aspects of the system will rely solely on the utilisation of an appropriate library or package, therefore if the library is found to be non-functional, this feature may have to be dropped; this could also be caused by a library or package that contains an excess number of bugs. Octoblu is another service that will be used, and again this is heavily relied upon to produce the desired sentiment analysis. Hence the results provided by this service are fixed and cannot be modified; additionally, the rate of processing of this service cannot be altered either or held to any defined performance measure. Consequently, any of these issues will have to be accepted or the feature may have to be dropped, this will also be the case if the service goes off-line or is taken out of service.

4.4.3 Management

The deadline imposed on this project constrains what is possible for the project to achieve, this due to the fact both research and implementation have to be completed, alongside a number of other tasks. Consequently, a balance between content and quality has to be made, as it would be impossible to complete true in-depth research in all areas of the project as the fields it covers are broad. Hence this also affects the amount content that can be incorporated into the application.

4.5 Foreseeable Technical Challenges

This project contains a full-scale development, as well as a broad spectrum of research areas. These will both take significant time to complete and will be completed in parallel for most of the development. This could cause an issue as the difficulty of a given task may not be evident until the time it is set to be completed. The biggest challenge will be the developer producing the application while also learning the development language and tools. Another area significantly affected by this will be workout inference, as this has not been completed before and there will be no supporting information for this. Therefore relevant information will have to be found by the developer, which will be difficult in an area that has a broad range of sources. Additionally applying this information will have its limitations, especially when working with a mobile device that has limited processing power. Hence the algorithm employed will have to be efficient. These are just apparent issues from the outset, but as the project progresses and greater knowledge is gained, it is likely more issues will become apparent; because of this the requirements will adapt to fit the realistic achievements of the project within the time-frame.

For the project to achieve the requirements set, a structured plan for the project will be required; this will be discussed next.

CHAPTER 5

Project Management

This project is based around building a new system; therefore time will be spent on a combination of both research and development. Both research and development will be ongoing throughout most of the development. This combination of both research and development is essential because new knowledge will be required for most features to be completed. It is important that the investigation in an area be completed before any implementation to avoid unnecessary work and mistakes. The project has a set finish date; thus it is important to have a well-defined timetable. Additionally, flexibility will be required to allow for unforeseen occurrences. The methodology to account for this will be discussed in Section 5.2. Due to the nature of the project aspects are likely to change, therefore any changes to the project management will also be discussed, alongside rational as to why these changes were deemed necessary and appropriate.

5.1 Design Approach

?? A bottom-up approach has been chosen for the project, as its characteristics most suit the project. This method has been selected because a working screen is required at the start of the development; from this base point more screens will be added to provide additional functionality, and the system will grow in a tree style structure spanning from the initial screen, with each component of the system being developed and integrated over time. This style of development perfectly matches a bottom-up approach, although there are common integration issues with this type of approach [47], steps should be considered when designing the system to avoid this.

5.2 Software Development Methodology

There are several aspects of this development that do not meet the key principles set out in the agile manifesto [16]. This is an individual project; hence there will be several issues with the principles that refer to team-based activities, such as daily scrum meetings to discuss progress, or paired programming common in the XP (extreme programming) approaches. There will be principles that are appropriate also such as using the completed working software as a measure of progress. Additionally, the flexibility available in an agile approach fits the suggested flexible research and development stages of this project.

Coleman describes a list of principles that are fitting of a plan-driven approach [25]; again the project is not completely suitable to this methodology, as there are aspects which cannot be met. The first issue arises, as the tasks which are shown in Figure 5.1 cannot be broken down further into more intricate tasks at the current stage of the project. The tasks cannot be broken down because the developers knowledge in the area is limited, and research still needs to be conducted; hence, the task is open-ended and cannot be accurately estimated. The lack of research in areas also affects the estimation of other metrics for tasks. The fact that this is an individual project affects its ability to meet team based principles for plan-driven methodologies also, such as those set out by Williams [98], whereby different team members

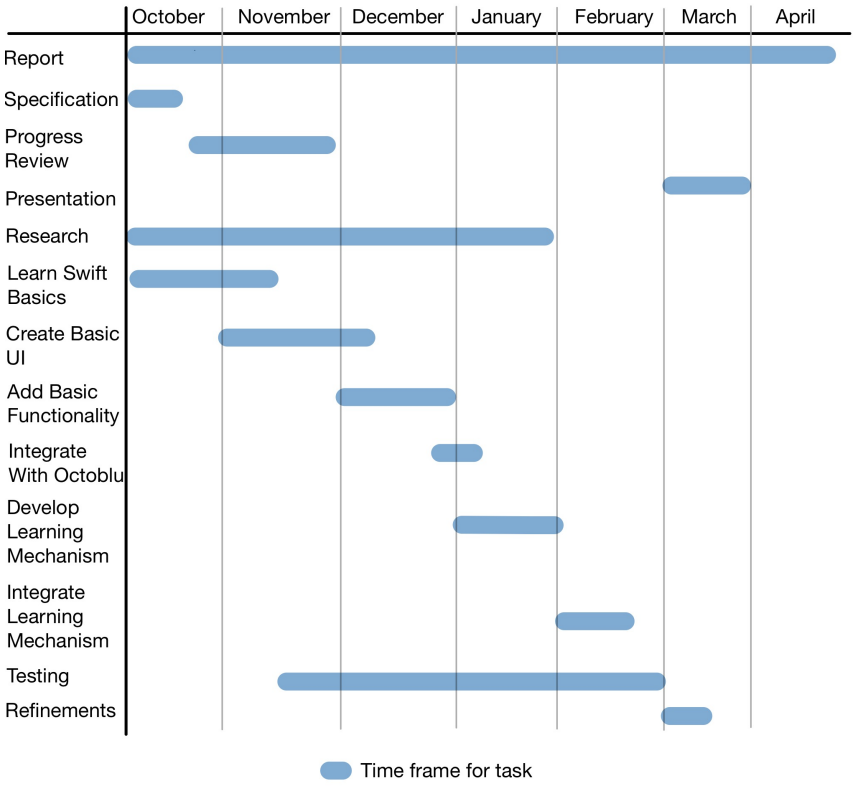


Figure 5.1: Gantt chart to shown projected task completion dates

have different roles and responsibilities for the development, instead all roles and responsibilities will be placed on the sole developer. Although not every aspect of a plan-driven approach can be applied there are aspects which can be, such as certain characteristics from the incremental model, which comply with the bottom-up development previously proposed for the project in Section **??**. Te incremental model also focuses on implementing a working model early on and then adding features in a prioritised fashion. These iterations will not be able to conform to the strict plan driven staging; however, they would fit a more agile approach with flexible completion goals.

There is no clear development methodology suited to this project, both agile and plan-driven approaches are not entirely appropriate. However, an adaptation between the two would fit taking the relevant characteristics from each to form a hybrid style approach.

5.3 Project Schedule

A timetable has been set out in the Gantt chart shown in Figure 5.1, the aim will be to follow this schedule as closely as possible. There exists some tasks for this project, which are not flexible and it will be required for these to be completed by a set date, these tasks and dates are shown in Table 5.1. The remainder of the tasks from the timetable, shown in Table 5.2 are more flexible and they do not have firm task completion dates, they do however have to abide by the project's finish date, which is the 27th April 2017.

To match the projects agile characteristics, tasks have been shown spanning across months, where each month will be considered as a sprint; thus any task that is set to end in the sprint should be completed by the date when the sprint finishes. Because of the agile approach, this timetable is likely to be adapted as more is learnt about the specific tasks.

5.3.1 Updated Schedule

As suspected, once the project had progressed and further research was conducted the original Gantt chart previously shown in Figure 5.1, had to be updated to a revised timeline for the project sections,

Table 5.1: Project Submissions

Task	**Submission date**
Specification	13th September 2016
Progress Review	28th November 2016
Presentation	6th-17th March 2017
Final	27th April 2017

Table 5.2: Project Tasks and Completion Dates

Task	Expected Start Date	Expected Finish Date
Research	October 2016	January 2016
Learn Swift Basics	October 2016	November 2016
Create Basic UI	November 2016	December 2016
Add Basic Functionality	December 2016	December 2016
Integrate With Octoblu	December 2016	January 2017
Develop Learning Mechanism	January 2017	January 2017
Integrate Learning Mechanism	February 2017	February 2017
Testing	November 2016	February 2017
Refinements	March 2017	March 2017

this is shown in Figure 5.2. Additionally, new tasks have been added, and some existing tasks have been refined to more accurately represent the current state of the project. As with the original chart, this is flexible and open to change as the project progresses further. Although the flexible aspects of the project have changed, the fixed deliverables remain in place, with Table 5.1 still representing these.

The updated Gantt chart includes week numbers for each month to give a more accurate deadline for each of the tasks. The week labelling is from 1 to 4. All months other than February are longer than four weeks, the extra days have been purposefully omitted from this diagram to allow unscheduled time at the end of each month to complete any tasks that overrun their allotted completion date. Additionally, the time assigned to learning the Swift language has been updated to correspond with the updated requirements of the project.

5.4 Tools

The project will require a range of different tools to be completed successfully; these will not just be physical tools, software tools will be equally as important. The tools set out to be used for the project will be discussed in their relevant categories below. Note that as the project develops it is likely new tools will be used that were not initially planned; any tools that will be added will also be covered alongside a description as to why it was a necessary addition.

5.4.1 Development Tools

The core development tool will be Apple's integrated development environment (IDE) software Xcode [10]. This IDE will be used to produce an iOS application [5], a companion WatchOS application is also planned [9]. The proposed coding language is Swift [8]; this is because it is Apples own native language and has more optimisations than the alternatives. Additionally, it has inbuilt libraries for integration with the Apple Watch device as well as the Apple Health Kit software. All of these development files will be stored with iCloud [11]; this is to help mitigate against loss. Cocoapods [17] will be used as the package manager to easily incorporate external packages into the development, as well as maintaining up-to-date versions of these packages.

Octoblu [69], the third party IoT service discussed earlier in Section 2.2.10 will also be used during development. This service will handle all of the sentiment analysis as well as the delivery of the appropriate response to the message received from the user.

Additional tools that will be focused on designing the user interface of the system will also be required, as a base drawing application GoodNotes [55] will be used to allow simple screen sketches to be produced. For more accurate design mock-ups and icon creation, a combination of graphic design and image editing

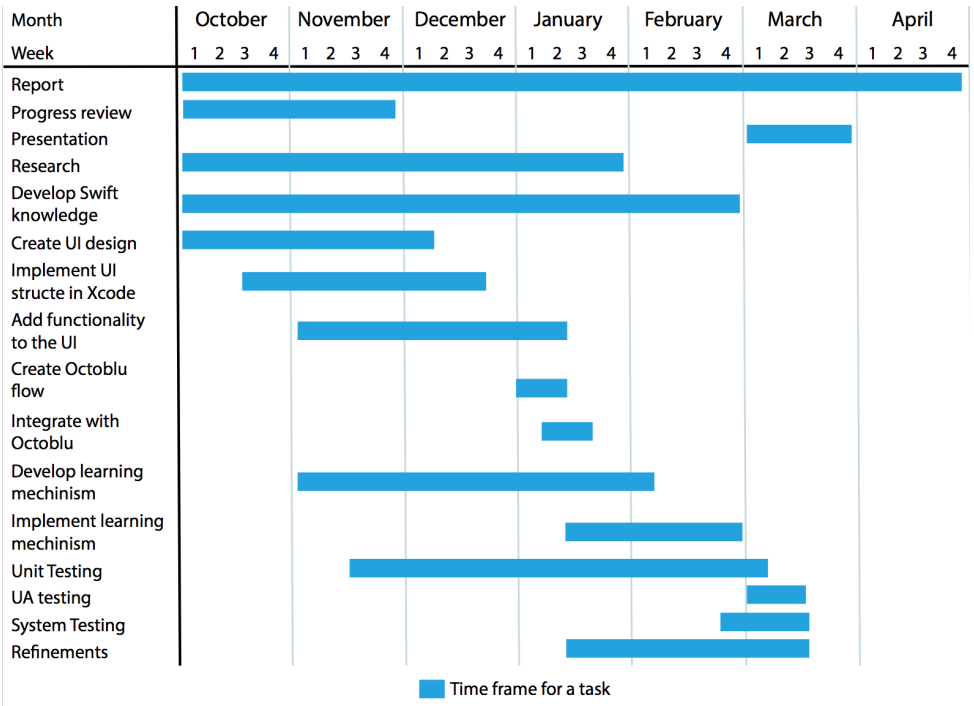


Figure 5.2: Updated Gantt chart to shown projected task completion dates

software will be used, the applications chosen are Affinity Designer [81] and Affinity Photo [82]. These applications provide all the necessary features and have excellent integration with one another.

5.4.2 Hardware

Three core hardware tools will be used throughout the project development; the main tool will be an Apple MacBook. It will be required to run Xcode development software previously mentioned in Section 5.4.1. A system running MacOS is necessary, as the Xcode software cannot be installed and run on any other operating systems such as Microsoft Windows [61] or Linux distributions. The other hardware devices which will be used are an Apple iPhone and an Apple Watch; the primary use for both of these devices will be in testing the system; ensuring that the produced application works on the intended device is an essential part of system testing.

5.4.3 Management

A variety of tools will be used to help manage the project, some of which will be focused on facilitating communication and coordination between the developer and the supervisor of the project. Communication is vital in ensuring the project progresses effectively and that it does not fall off-track. The most used tool for communication will be Instant Messaging, which will be provided by Apple's cross-platform Messages app [6]. If communication cannot be maintained through this platform, Email will be used to communicate. Video chat will also be utilised, as sections of the project's schedule will fall on periods where the developer and supervisor are not located within accessible distances; Skype [88] will be used to provide the video chat functionality. Finally, to ensure that both the developer and supervisor have access to all the project deliverables, a shared folder will be used to store all these files; this will also allow both parties to have anytime access to the latest versions of all documents. Dropbox has been chosen as the tool to provide this feature [30].

There will also be multiple tools that will be used to produce the documents that are required to support this project. The first will be a latex editor, for this TexShop [51] will be employed. This application lacks the ability to have tapped windows, therefore an additional text editor will also be used; for this Sublime Text 3 [90] was decided upon. Keynote [12], Apple's presentation software will be used to complete the slides for the project presentation. Video examples of the running application will

be required for the presentation, hence software to screen capture is also necessary; the only solution for this is a feature within QuickTime [13] so this will also be used.

An additional management tool was required after the implementation had begun; this was necessary as the number of tasks grew and it became exceedingly difficult for the developer to track at what stage each unit was at, alongside any bugs that had been discovered. Additionally, the units that needed testing also needed to be tracked. The tool chosen for this was Trello [94]. Trello was chosen as the developer had had previous experience using the tool and it provided an easy to use visual interface that facilitated tracking easily.

5.5 Management approach

Certain practices will be employed to ensure that the project progress in an effective manner. To achieve this, weekly meetings with the project supervisor will be scheduled. These meetings will be used to discuss the current status of the project, along with any issues encountered and the upcoming work that is planned. Additionally, comparisons of the iterations of the project will be shown to highlight any work that has been completed between iterations. To further ensure communication is maintained between the developer and supervisor, the tools stated previously in Section 5.4.3 will be used.

5.6 Risk Management

As already mentioned there are a number of hardware devices that are core to the development. The most essential of these is the MacBook used for writing the application, as previously stated a device running MacOS is critical, therefore ensuring this device is available throughout the development is critical to the project's success. As there is no way to prevent against hardware failure, a resolution needs to be planned for in case such a failure occurs. To prepare for this failure, insurance will be purchased for the laptop to ensure that if any issue arises a working replacement will be made available. An extra cover will also be taken out to protect against accidental damage to further reduce risk. If a failure occurs, it is probable that a delay in the project will also be caused, as a replacement is unlikely to be supplied instantly. The next two devices which will be used in development are the iPhone and Watch, if any of these devices become unavailable the simulators built into Xcode will be utilised as a replacement to ensure development and testing can continue. Although the app cannot be tested during expected use with the simulator, all actions can be simulated and this will prevent any possible loss of time that could occur from failure of a device.

Another aspect where risk needs to be minimised is regarding the loss or corruption of files. As all of the development will be software based it is vital the files are not lost or corrupted as this would significantly affect the project's completion. To minimise this risk a combination of technologies and practices will be used. The first will be to have the software stored in the cloud to prevent corruption and loss through hard drive malfunction; additional copies will also be stored on the hard drive periodically to ensure that the files are not lost from the cloud storage. It is also important to store multiple versions of the files to make sure that a file change will not cause significant issues; to prevent this from happening a git repository will be used and regular commits will be performed alongside appropriate branching. These practices will ensure that any changes are not irreversible. A final tool will be TimeMachine; this is Apple's inbuilt backup utility that also stores versions of files and allows files and folders to be reverted to a previous state. The use of all these tools and practices will provide sufficient avoidance of the loss of files.

The use of Octoblu as the core sentiment analysis service is also a high risk; if Octoblu ceases to operate the feature would have to be dropped to ensure that the remaining features of the system are not affected. This risk is deemed acceptable as this is a large scale distribution by an established company [24], and therefore it is unlikely that the service will be stopped; any downtime is also likely to be minimal.

Now a plan for the management of the project has been established, the design work for the project will be covered.

CHAPTER 6

Design

Before implementation of the system can begin an adequate design of the system is required, this will be separated into two categories, the user interface design, and the back-end design.

6.1 Architectural Design

Based on the requirements and goals for the system, it is clear there will exist some set functions of the application; each of these specific functions can be thought of as a component. The components will be menu, new workout, past workout, exercises, nutrition, user profile and settings. A suggested component diagram has been included as Figure 6.1, to show how the components will be structured, as well as how they will interact.

Each of the components from this diagram will be covered as a section in both the user interface and back-end components.

6.2 User Interface

This section will be used to establish a baseline user interface for the system based on the sections discussed above. They will include descriptions of design decisions made for the appropriate section, alongside some initial sketches of what the UI should look like, which will be used by the developer

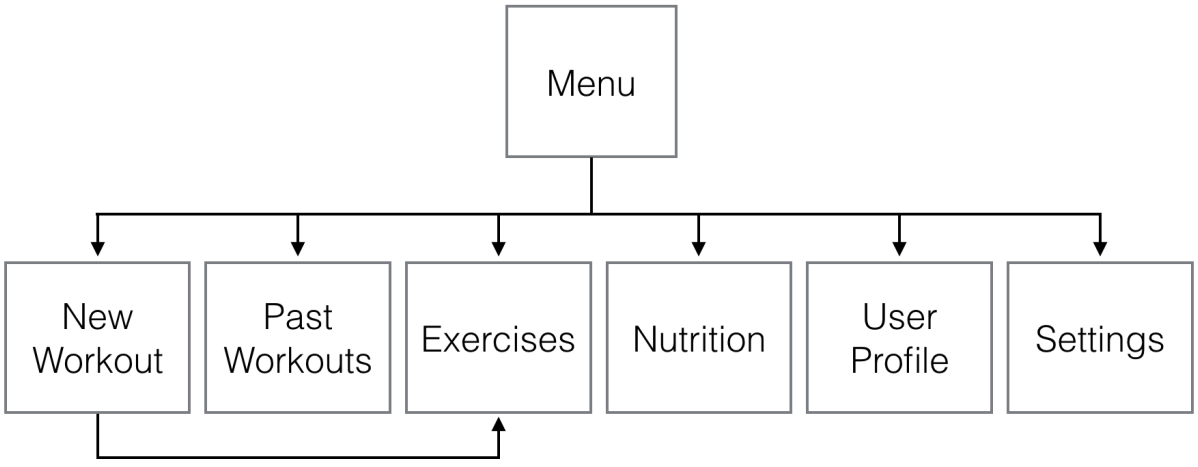


Figure 6.1: Component diagram for the system

when fabricating the system. These will be attached within the appendices; each screen will have a hand -written number which will be used to refer to the screens in this section. It is likely that as development progresses, new views will be necessary to implement a feature successfully. It is possible for such a screen to be designed as it is being constructed and therefore not require a full design mock-up, any examples of this that occur will be discussed in the implementation of the relevant section.

6.2.1 Menu

The menu screen will be the first screen of the application that is loaded and therefore must provide an easy way to navigate to each section of the application. These will be shown in a list form with dividers between the different options to create a box; all of the box should act as a button. Also, the size of these boxes should be equal to the screen height dived by the number of options; this will ensure that all options are visible without the need to scroll through options. The addition of interface metaphors for each option will also be used for each section, in an aim to reduce time selecting the desired option. A priority ordering will also be applied to attempt to rank the sections in expected usage, with the highest used at the top and the lowest at the bottom. If the user scans the list from top to bottom, they are likely to find the section they wish to choose quicker with this ordering. An example mock-up is supplied in Section 1 of Appendix A.

6.2.2 New Workout

The new workout section of the application will be broken down further into multiple screens to reduce the complexity for the user.

Workout Selection

Workout selection will be the first screen displayed after the user selects new workout from the menu screen; this will provide the user with many possible workouts, again in lists. There will be two sections to this screen, one half will show custom workouts the user has created, and the other section will show the inbuilt workouts; both of these sections will have scroll ability to enable the user to select their desired workout. Every workout in the lists will also act as a button that will navigate the user to the in-progress workout screen which will be discussed next. Two other buttons will also be present on this screen. The first will be used to generate a custom workout for the user, again this button will navigate to the in-progress workout screen. The other button will navigate users to the create workout screen, because of this the icon for this button has been represented as a '+'. An example mock-up is supplied in Section 2.1 of Appendix A.

In-progress Workout

The in-progress workout view is likely to be the most used screen of the application and one that is vital in providing the user with clear workout instructions. A vast majority of this screen will be assigned to the workout information; this will be supplied with exercises as headings and set details listed below, all of these will be scrollable. The fields within a set, such as weight and reps should be selectable, and on a press they should load up the keyboard to enter the new values. A narrow static header bar will also be used to give workout information, such as time and total weight lifted, a tap feature could also be included to change the statistics given on a tap. The end workout button will be placed in the menu bar to avoid it being pressed accidentally. A pop-up could then appear to confirm the user wants to terminate a workout, upon confirmation the user would be navigated to the workout review screen which will be covered next. An example mock-up is supplied in Section 2.1.2 of Appendix A.

Workout Review

The workout review screen will be used to gather the user's opinions of the completed workout. To do this a text box will be used to receive a text review; for further information, a rating system will also be used. To represent this rating system, five unfilled stars will be placed on the screen, on the selection of a star all stars up to that star will become highlighted; this highlighting should adapt to lowering or increasing star level further. Once both of these fields have been completed, a save button should appear.

The save button should navigate back to the main menu of the application. An example mock-up is supplied in Section 2.1.2.1 of Appendix A.

Create Workout

As the name describes, this screen will be based around allowing users to produce their own workouts. To enable workout creation, certain aspects will be required. This will include: a text box to name the workout; a section that lists the exercises currently within the workout; a button to save the workout and a way to add new exercises. The new exercises button will redirect to the Add exercise screen, which will be discussed next. An example mock-up is supplied in Section 2.1.1 of Appendix A.

Add Exercise

This screen will list all the exercises that are contained in the application; each exercise name will act as a button that allows selection and will change its appearance on selection and de-selection. An add exercises button will also be required to navigate back to the previous create workout screen. The addition of a search bar will be included to filter exercises by name. An example mock-up is supplied in Section 2.1.1.1 of Appendix A.

6.2.3 Past Workout

For past workouts only two screens are required, these will be a screen to show the past workouts and another to show the actual workout completed for a selected day.

Past Workouts

The main focus of this screen will be a calendar; this should show the current month. Each day within a month should act as a button; upon pressing a button the day should become highlighted to indicate that it has been selected. Additionally, a scrollable list should be located below the calendar for the date selected, all workouts completed on that day should be displayed in the list. Each workout name should also act as a button that navigates to the workout view screen, which will be discussed next. An example mock-up is supplied in Section 3.1 of Appendix A.

Workout

This screen will take inspiration design from the in-progress workout screen as it has a similar focus. For this screen less user interaction and detail is required therefore this screen will display a list of exercises and sets. As before the exercise will be the heading and each set will appear below the corresponding exercise showing the weight and reps. For this screen, the weight and reps value will remain static and will not be modifiable; this is because the view is focused around showing a completed past workout, therefore the alteration of values should not be necessary. A back button will be included to return to the past workout screen. An example mock-up is supplied in Section 3.2 of Appendix A.

6.2.4 Exercises

For this component three sections will be used. The first two will be used to filter the exercise catalogue down to specific muscle groups, and the final screen will show exercise details. A staggered approach has been chosen to ease use for the user and prevent an overload of information.

Exercise Groups

The exercise groups view is the first screen that will appear when exercises are selected from the main menu; its design will also be very similar to the main menu as their functions are similar to one another. This screen will show a list of top-level muscle groups; these lists will be restricted in size to enable it to fit on the screen without the need to scroll. Another feature that will be taken from the home screen is the addition of an image to help aid the selection of the correct group when using the app quickly; this should also remove the necessity to read every option reducing usage time. Each one of these sections will act as a button that navigates to the muscle group screen, which will be covered next. An example mock-up is supplied in Section 4.1 of Appendix A.

Muscle Group

The focus of this screen will be to show all the exercises that work the muscle group selected. For a given exercise the muscles worked can be either the primary or secondary focus of the exercise; two sections will be created to show the exercises where the muscle group is the primary focus of the exercise and a list where the muscle group is a secondary focus. Both of these halves will have scrollable exercise lists, and each exercise within the list will be a button that navigates to the exercise screen. There is a possibility this screen could be extended further to include an addition button to allow the user to add their own exercises to the current exercise catalogue; this button would be placed in the top right corner and share the same icon as the create workout button to maintain design convention if implemented. An example mock-up is supplied in Section 4.1.1 of Appendix A.

Exercise

The main focus of the exercise screen is to provide information to the user about the exercise, depending on the user's level of knowledge the information required from this section will vary. For a novice, information regarding how to complete the exercise and the benefits it has are most important, whereas a more experienced user is more likely to be focused on their metrics for that exercise, such as 1-RM and max weight lifted. Therefore the exercise screen needs to incorporate all of these. A visual aid section will be placed at the top of the screen, with a scrollable read-only text section below; these should take-up approximately 66% of the screen, and they will be focused on new users and giving them a varied selection of information about how to complete the exercise. The remaining screen will be used to display metrics about the exercise for the more experienced user; this will contain 1-RM, most weight lifted and most reps completed as a baseline. An example mock-up is supplied in Section 4.1.1.1 of Appendix A.

6.2.5 Nutrition

This component will require only one screen for the user interface; the focus of this screen is to provide the user with their daily calorie and macronutrients intake. For most users the calorie intake will be their main focus, therefore this will be given the biggest priority on the screen appearing at the top with the largest font, leading the user to read this first. The macronutrients will then follow underneath the calories in a list format, with a smaller font than calories to ensure they are do not draw user's attention first. Finally, a calculate button will be included at the bottom of the view, which will be used to update the calorie intake displayed; a visual aid should be added to this to make the user aware when the button has been pressed. This update is especially necessary considering the values displayed may not change even if new calculations have been completed; this would occur if the user's personal information has not changed between calculations. A possible solution is to modify the button text from calculate to re-calculate once pressed; this would show that a calculation has already been performed. An example mock-up is supplied in Section 5.1 of Appendix A.

6.2.6 User Profile

The user profile section of the application will contain the most screens of any component, as it requires at least one additional screen for each of the user's characteristics. Currently, height, weight and fitness goal is covered under the user profile; this list can be expanded as necessary.

6.2.7 Profile Summary

The profile summary view will be the focus of the user profile component of the application, as it will display all the information currently stored about the user. It is import this is clear; this is required not only for usability but also to make the user aware of what personal details are currently stored about them within the app. The preliminary design of this screen is to have a list of the fields stored about the user, with the field name pushed left and the value relative to this field to be pushed to the right. Each of the elements in the list will navigate to the appropriate screen corresponding to that option. Each of these screens will be covered next. An example mock-up is supplied in Section 6.1 of Appendix A.

Weight

The Weight screen will have two operations; the first will be focused on displaying a history of the user's weight in a chart. As motivation for creating the application was to aid weight loss, this feature is imperative to help the user visualise the progress they have made. The next function is allowing the user to update the weight value stored by the application; the addition of this value should also refresh the weight chart. The chart should be the main focus of the screen with the update field below, an option to select the measurement units may also be useful, as conventions vary between user age ranges and location. An example mock-up is supplied in Section 6.2 of Appendix A.

Height

The height screen will have a much more simplified interface; it is highly likely that this screen will never be accessed as a large majority of users height values will not change during usage. This option is still necessary to allow for any mistakes made during data entry. The design of this screen will be a duplicate of the weight screen, with the chart swapped for a larger text representation of the user's current height. An example mock-up is supplied in Section 6.3 of Appendix A.

Goal

This screen will be used to allow the user to select or update which fitness goal they would like to aim towards. During the Research chapter in Section 2.2, three goals were established for the system; these are weight loss, muscle maintenance and mass gain. As there are three goals the system will be broken down into three sections, the goal name will be the main focus of the section. Once a goal is selected it will become highlighted to represent that it has been chosen. To help provide users with more information, an info button will also be included which will link to a screen showing a detailed description of that goal; this icon should be shown as an 'i' with a circle surrounding it to follow Apple design conventions. An example mock-up is supplied in Section 6.4 of Appendix A.

Goal Description

For new users, the titles given for the fitness goals may not be descriptive enough for an informed decision to be made, therefore, it is important to supply the user with more information about the focus of each goal and its aims. This description has been designed as a separate screen, however, it could be possible to have this description as a pop-over on the goal display. The design will be the simplest of any implement displaying only text; this will be in a scrollable format to ensure if the description exceeds the screen it can still be read. Additionally, only one screen will be created for all goals, and the text will be dynamically loaded depending on which goal the user requires information about. An example mock-up is supplied in Section 6.4.1 of Appendix A.

6.2.8 Settings

At this stage the contents of the settings screen are still undecided; hence, some options within settings have been left unassigned to enable them to be allocated as required throughout the development, if they are necessary at all. Similar to the menu screen the settings screen will just be a list of options, the main difference will be that this screen can be made scrollable if required to account for the necessary number of settings. However, a tiered approach to settings should be maintained to follow conventions used and prevent the settings list from growing too large that it is arduous to find the required setting. Because there are limited settings to consider, a preliminary example of possible settings for the system has been shown in Section 7.1 of Appendix A. Within the settings, unit selection has been demonstrated as an example of how a tiered approach could be used to link relevant settings within a single screen; an example mock-up of this is supplied in Section 7.2 of Appendix A.

6.3 Back-end Components

Now the interface design of the system have been completed, the background functional aspects of these components can be discussed. As the developer's current knowledge is limited for iOS development, not all screens from design will be covered, especially those which are based on data retrieval and entry or

navigation. To cover this section, the topics will be broken down as they were above into the components of the system. There will be two core areas for design which are known to be complex; these are nutrition and workout generation. Both of these sections will be discussed within their relevant components.

6.3.1 Menu

The function of the menu screen is minimal; it's only required function will be to retrieve the user's information from the database. The user's information will then need to be passed to the relevant component depending on user selection. No other functionality can be discussed at this point; if this screen requires more detailed functionality, this will be covered in the appropriate section of the implementation chapter.

6.3.2 New Workout

The bulk of the complexity for this component that requires design is regarding the generation of workouts; therefore this will be covered in depth.

Workout Generation

For the app to behave as an actual personal trainer would, there are a number of key areas that need to be concentrate on when generating workouts for the user. These are as follows:

1. **Weight to lift for a given exercise**
2. **Number of repetitions to complete, for a given exercise**
3. **Number of sets to complete, of a given exercise**

Additional to the points mentioned above, a decay rate needs to be applied to the exercises. The decay value will be used to reduce the recommended weight to be lifted further, as the number and type of exercises increases. Based on the research conducted in this area previously discussed in Section 2.2.7, there is much debate to a correct value, and instead, a user specific approach is the best possible solution to this, therefore it would be desired to develop a learning mechanism to more accurately calculate a specific user's cooling off period. Such a solution will not be possible within the initial time frame, therefore, a more simplified approach will be discussed later. Before any calculations can be made for a user a baseline working knowledge is required. However when a user first uses the application, no recommendation can be provided as no information on that user currently exists. This problem is referred to as the Cold Start Problem, and it will be discussed further next, along with each of the points listed above.

Cold Start Problem

Based on a formula for 1-RM previous discussed in Section 2.2.5 it could be possible to predict the recommended weight to lift for a person, given only one complete set of the given exercise with a suitable weight. This would greatly simplify the cold-start problem for this application, as it would require only one set of each exercise to be completed to gather sufficient information about the user to give recommendations. There are two prominent ways to collect this exercise information; the first is to use a pre-built workout which has multiple common exercises and requires that the user only completes one set of each exercise with a light weight to avoid injury. The use of a light weight should not affect the results of stronger trainers as they should be physically able to complete more repetitions of the exercise leading to a higher recommended weight to lift when the number of repetitions are adjusted to meet a user's fitness goal. The second would be to ask the user to complete a baseline set whenever a new exercise is set to be undertaken. It is likely that a hybrid approach will be used, as not all exercises can be completed by a user in one workout; this would take an inappropriate amount of time to gather information about possible exercises as there are thousands currently documented. Additionally, it is highly unlikely that the user will have all possible fitness equipment available to them. Additionally, a significant majority of those exercises will never be undertaken again, and the gathered results will have been unnecessary. Hence a hybrid approach seems most fitting. It could test for the most common exercises on a user's first visit and from then on only asks for a baseline to be carried out when a new exercise is set to be undertaken. Further to this, an option for experienced users could also be incorporated, such that they

can enter either their usual lifting metrics or their 1RM on that specific exercises screen, which would provide the system with the information it requires and skip the need for a baseline to be gathered. The secondary approach to also collect information on exercises without a record documented would also be implemented alongside this.

Weight Calculation

The research previously conducted in Section 2.2.5 was utilised to provide the formula required to calculate the weight to lift for a user given the number of reps. As earlier mentioned Brzycki's formula for 1-RM is the most highly regarded, hence, this is the formula that is proposed for the implementation. This original formula is shown below; however, this is not in the exact form required. Therefore the desired formula was established to focus on the weight value instead of 1-RM.

Brzycki:

$$1RM = weight * \frac{36}{37 - reps} \tag{6.1}$$

Rearranged:

$$weight = 1RM * \frac{37 - reps}{36} \tag{6.2}$$

To successfully calculate weight, a value for reps is also necessary; how to estimate a value for this will follow on next.

Rep Calculation

Again the previous research conducted in this area that will be used for the calculation of a repetition 2.2.4 found that lower rep ranges benefited mass gain, medium ranges promoted maintenance, and high repetitions promote weight loss. Based on this the following suggestions for repetitions has been decided upon.

Weight Loss : 15 Reps
Muscle Maintenance : 12 Reps
Mass Gain : 8 Reps

Set Calculation

A suggested range for the optimal number of sets is between 2 and 6, based on the research earlier conducted in Section 2.2.4. Therefore a set value of 4 shall be used for all fitness goals. This value will be kept modular in order to ensure that it can be updated if it is found to be inappropriate.

Weight Decay Function

As there is no current research to suggest a general use value, an approximation based on the developers tests will be used. The chosen value of decay is 5% per exercise completed; this value will only be used for exercises which work the same primary muscle. This component should be made very modular to allow this method to be changed for a more advanced model when necessary.

6.3.3 Exercises

The specific screens for exercise will have a very apparent operation; the functionality of these components should be provided by the core elements. There are aspects of exercises that will need to be designed, the first of which is storage. From early on in the development, it has been stressed how a detailed catalogue of exercises will be required. Therefore the design of how these exercises are accessed and the information it will contain are crucial. The design of the storage will be covered later in Section **??**. There are a number of aspects that must be available within the exercise object and these are exercise images and a

workout description. To produce a catalogue from scratch would take significant effort from the developer in addition to consuming a large segment of the allotted development time. Hence the information for the catalogue must be externally sourced to make this feature viable within the time-frame given. The exact sources used will be discussed in the implementation of this feature, as the content chosen will be dependent on the implementation of the feature.

6.3.4 Past Workout

The functionality behind all the past workout screens will revolve around creating efficient queries to retrieve the information required from the database. This information will then be used to update the UI. The specific queries will be produced when needed in implementation, as only a suggested schema has been produced at this stage.

6.3.5 Nutrition

The supporting implementation for the display of this screen will require little implementation. Instead, the focus for this feature will be the calculation of the nutritional values. This component of the application will aim to implement the formulas previously researched in Section 2.2.3. As discussed a base calorie intake must first be established, therefore the equations given in Section 2.2.3 will be used. There is an issue with implementing all of these values, and that is the complexity of using MET's; this would require the user to enter all exercise performed over the course of a typical 24 hour period in their lives, providing each specific exercise with a MET rating. Providing such a value would be exceedingly difficult for most users, hence a simplified version will be used where the user simply selects their level of day-to-day activity according to the four PAL category names directly. The corresponding PA value will then be utilised for the full calculation, therefore the remaining calculations will remain unaltered. This, therefore, requires the following formulas to be implemented.

Basel Energy Expenditure for men:

$$BEE = 293 + 3.8 * age(years) + 456.4 * height(meters) + 10.12 * weight(kg) \tag{6.3}$$

Basel Energy Expenditure for women:

$$BEE = 247 + 2.67 * age(years) + 401.5 * height(meters) + 8.6 * weight(kg) \tag{6.4}$$

Calculating PA for men:
Sedentary: $PA = 1.0$
Low active: $PA = 1.12$
Active: $PA = 1.27$
Very active: $PA = 1.54$

Calculating PA for women:
Sedentary: $PA = 1.0$
Low active: $PA = 1.14$
Active: $PA = 1.27$
Very active: $PA = 1.45$

Total Energy Expenditure (TEE) for men:

$$TEE = 864 - 9.72 * age(years) + PA * [(14.2 * weight(kg) + 503 * height(meters)] \tag{6.5}$$

Total Energy Expenditure (TEE) for women:

$$TEE = 387 - 7.31 * age(years) + PA * [(10.9 * weight(kg) + 660.7 * height(meters)] \tag{6.6}$$

Using the TEE value generated the daily calorie recommendation will be adjusted further to match the user's fitness goal. A user's macronutrients will also be recommended. This will use the new goal adjusted calorie intake as a baseline for calculations. Additionally, to recommend macronutrients effectively they should be shown as a value of weight as this is the common method of measuring macronutrients in food. However, the formulas previously researched will supply this in calories, so a formula for conversion from calories to grams is also required. The conversions are listed below.

Fat: 9 *calories* = 1 *gram*
Protein: 4 *calories* = 1 *gram*
Carbohydrates: 4 *calories* = 1 *gram*

Now each fitness goal will be covered individually showing what adjustments need to be made to the calorie intake based on the research earlier conducted.

Weight Loss

To calculate the adaptations for weight loss, multiple calculations are required; the first was to create a formula that would convert from percentage of mass to the number of calories to reduce by. To produce this value, a baseline was needed relating calorie reduction to weight loss per week. This decrease is estimated at 0.1 kg loss per 100 calorie reduction to a daily diet; it is worth stating that this figure is not final and will vary based on the individual. However, it is used as an approximation in a significant number of calorie calculators, therefore it is most suitable for this calculation. The reduction in body weight is suggested to be between 0.5 – 1% per week; therefore a value of 0.75 will be used as an average between the two. This can be changed if it is proving ineffective. From these calculations, the equation for goal specific calorie intake can be formed; this is shown in its most simplified form below.

$$CalorieIntake = TEE - (weight(kg) * 7.5)$$

To calculate each of the macronutrients for weight loss a decision on the percentage of fat consumes needed to be decided; as the range is from 15–30% a value of 22.5% will be used, and again this value can be modified at a later date to make the formula more effective. Additionally, the protein percentage needs to be calculated based on grams of protein per kg, where a value of 2.7g/kg will be used. The final complete set of formulas is shown below.

$$CalorieIntake = TEE - (weight(kg) * 7.5)$$

$$ProteinPercentage = \frac{weight(kg) * 2.7 * 4}{CalorieIntake}$$

$$Fat(g) = \frac{CalorieIntake * 0.225)}{9}$$

$$Protein(g) = weight(kg) * 2.7$$

$$Carbohydrates(g) = \frac{\frac{CalorieIntake}{100} * (100 - proteinpercentage - 22.5)}{4}$$

Muscle Maintenance

For this fitness goal, the user's weight is meant to be maintained therefore the calorie recommendation already calculated is correct for this aim. From the research conducted, a macronutrient breakdown the same as weight loss was deemed as appropriate; therefore the same formulas will be used for this.

$$CalorieIntake = TEE$$

$$ProteinPercentage = \frac{weight(kg) * 2.7 * 4}{CalorieIntake}$$

$$Fat(g) = \frac{CalorieIntake * 0.225)}{9}$$

$$Protein(g) = weight(kg) * 2.7$$

$$Carbohydrates(g) = \frac{\frac{CalorieIntake}{100} * (100 - proteinpercentage - 22.5)}{4}$$

Mass Gain

In comparison to the previous goals, the formulas for mass gain are straight forward. Based on the research conducted, calorie intake should be increased by 15% and the breakdown of macronutrient percentages are 55% carbohydrate, 30% protein and 15% Fat. The formulas for this gaol are shown below based on these values.

$$CalorieIntake = TEE * 1.15$$

$$Fat(g) = \frac{CalorieIntake * 0.15}{9}$$

$$Protein(g) = \frac{CalorieIntake * 0.3}{4}$$

$$Carbohydrates(g) = \frac{CalorieIntake * 0.55}{4}$$

6.3.6 User Profile

The functionality for the user profile screens will all be very similar; each screen will be focused on the user's information, by either displaying the information or receiving it. This data will be the same throughout the screens, hence, a centralised user data model should be used to ease the complexity of the implementation. For the main user profile and its child scenes the information should be retrieved and provided using the native Apple functions, therefore, no design is required. The only screen that will require further planning is the weights screen as this has a more complex functionality compared to all other user screens.

Weight

For the weight display, a way to access the weight values was most important as this would be required to generate the chart proposed for the user interface. It was decided that storing a list of tuples would be the best approach, with the tuple containing a weight and the date the weight was recorded. This structure needs to be stored permanently to allow the chart to be produced between app uses and data entries. To add new information to the app, only the weight value will be required. The tuple will be formed by using the system clock to get the current date and then this tuple should be appended to the list. For the generation of the chart, a library should be used to simplify the implementation; the choice of the library will be covered later in Section 7.4.1.

6.3.7 Settings

One feature that will require implementation in settings is the ability to restore the application to default, for this the entire user entry in the database will be erased. As the information and function provided by the settings screen are open to change, the function of the remainder of these screens will be designed when required to avoid unnecessary work at this stage.

6.4 Data Storage

Data storage will be a main component of the application, and there currently exists many options. The first decision to be made is whether to use an SQL based database or the newer NoSQL databases. Based on the way data would be accessed, such as through user objects, tuple lists and the way it will flow through the system, a NoSQL database seems fitting of the project. The two most popular iOS NoSQL databases are Core Data [14] and Realm [78]. Core Data is promoted by lots of sources, and there are good reasons for this; Core Data is Apples own database solution, meaning it is very stable

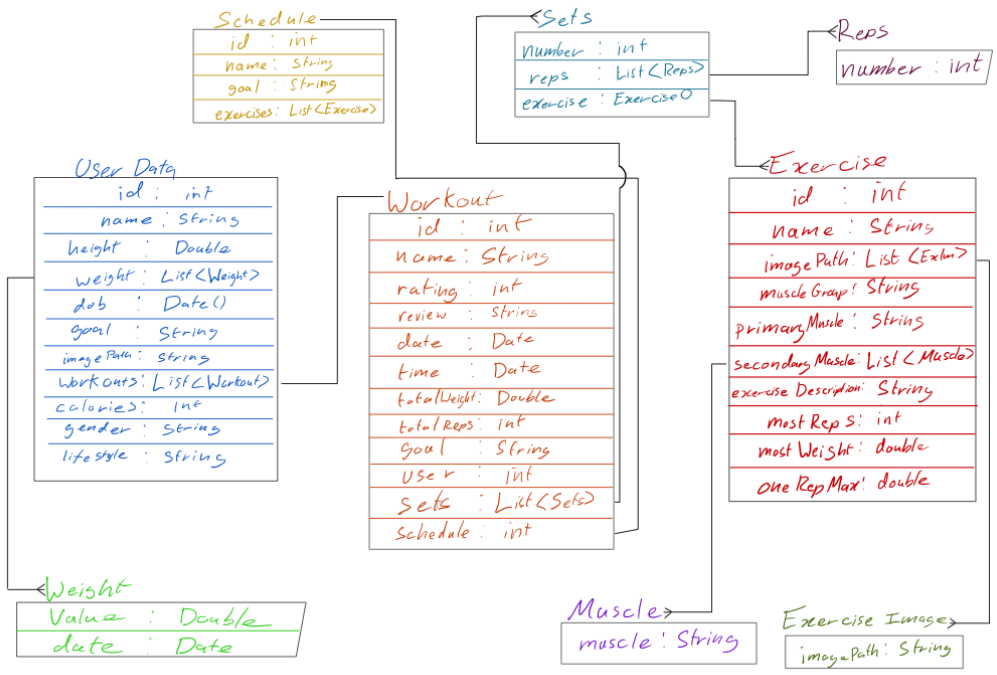


Figure 6.2: Proposed Realm object database schema

and well integrated into iOS. The usage of this is complex, and it is expected that this may take extra time from other areas of the development. Realm's Mobile Database in comparison has a much more simplified usage. This is not the only benefit to using Realm, it has faster query times than Core Data also. There would be an overhead in the usage of this as only certain objects can be stored within a Realm database requiring a thorough design of the database schema. Based on a discussion with the Head of App Development at The Hut Group [91], as well as the points previously made, Realm was finalised as the database of choice. As previously stated Realm requires that objects be stored in a specific format; this is limited to Realm specific types as well as primitives. An example schema is shown in Figure **??**. This very different from a standard SQL schema; the biggest difference is the use of lists to store objects which contain only a single primitive type. This is necessary as Realm only allows lists to store Realm objects, therefore, a special object must be made to be able to store primitives in a list. It is likely that this schema will be updated throughout development to allow for the changing requirements and agile approach. This is another significant benefit of a NoSQL database; any alterations required can be easily added to the object.

6.5 Workout Motivation

At the core of the workout motivation feature is sentiment analysis, this will be carried out by Octoblu. This feature will not have an interface other than for user information retrieval which has already been covered within Section 6.2.2. Therefore the remainder of this section will discuss how data is expected to flow for this feature. The data flow diagram for this unit is shown in Figure 6.3.

To maximise the use of this flow it should be designed in a way that no manual data entry is required. Instead, all of the required values should come from the request itself; implementing the flow in this way would allow it to be used by all of the users of the application. This implementation would also provide security, as this way even if the flow is accidentally triggered it will not complete as the required parameters are not supplied, hence, making it redundant and preventing misuse of the flow. This secondary benefit is achieved by adding aspects to the flow which only proceed when certain keys are given for values; therefore this technique should also be employed in the final flow.

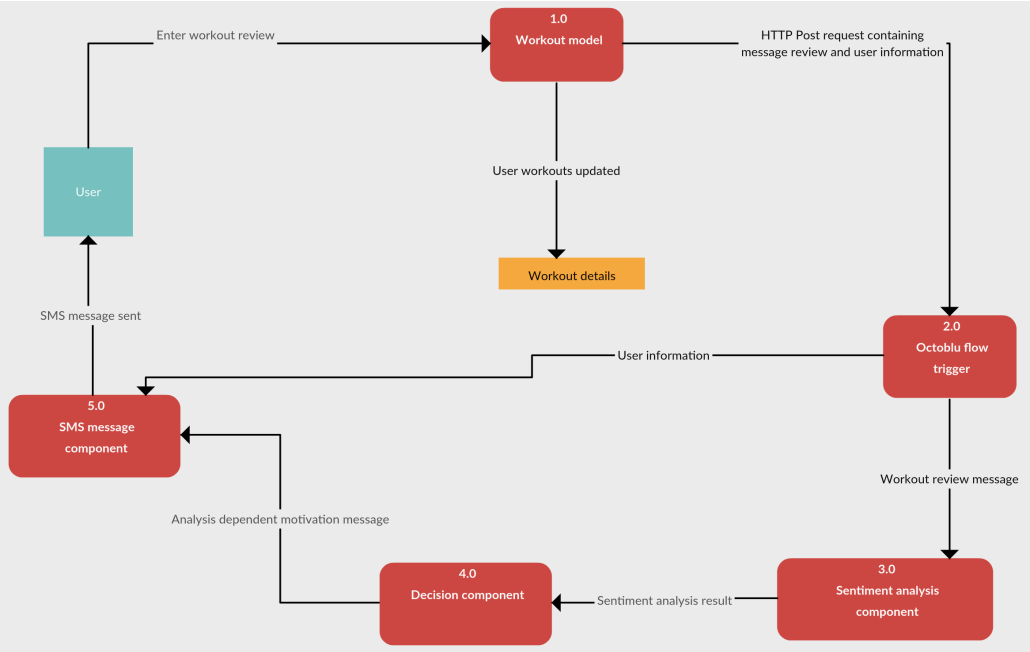


Figure 6.3: Data flow diagram for the sentiment analysis using Octoblu

Now a suggested design of the system has been outlined, implementation of the proposed system can begin.

CHAPTER 7

Implementation

This chapter will discuss the full implementation of the project, covering both the user interface and the back-end functionality. Any issues encountered will be discussed, alongside the solution implemented to resolve the issue.

7.1 System Architecture

The implementation of the system closely follows the component diagram set out in Figure 6.1 of Design. This is shown by the full system diagram illustrated in figure 7.1. The sections of this document will be separated into each of the original components set out in design.

7.1.1 Early Development

To provide the developer with the knowledge required to start developing, Apple's online tutorial "Start Developing iOS Apps (Swift)" [4] was undertaken. There were many issues in the early stages of development due to the developer's lack of understanding of the Swift development language. This lack of knowledge compounded to the further issues encountered, as the developer was uncertain about not only the error messages provided but the language itself.

The issues faced were first apparent during the tutorial, despite the fact all of the code base for the tutorial being supplied, this code would still not compile correctly, and error messages were produced. The reason for errors within the provided code was due to aspects from the latest version of Swift not being backwards compatible with the previous version; this caused multiple errors due to compatibility issues with using the latest version of the language. Some methods had been deprecated from the current version and replaced with new methods that had a similar function, whereas other methods had changed operation significantly. This was an unexpected setback, and due to the nature of the issues, support from blogs on how to resolve these problems was lacking as the update had just taken place. These errors caused multiple setbacks not only in the project's timeline but also the developers learning, as the tutorial was hard to follow due to the errors. It was highly unexpected that the tutorial supplied code that would not work especially considering it was created by the developers of the language, and the driving factor behind taking the given course was its expected quality entirely due to the fact it was supplied by Apple themselves.

The Apple tutorial had proved to be insufficient in providing the developer with a greater knowledge of the language. Therefore it was decided that this tutorial should no longer be used as it was not fulfilling its purpose and it was unlikely that further progressing with this tutorial would provide the required knowledge. Instead, alternate methods were researched and "CS139P iPhone Application Development" [96], a module produced by Stanford University and made publicly available to view was undertaken. Other courses were also sourced online to give the developer further options if required; one further option considered was "The Complete iOS 10 Developer Course" [72]. Only CS139P was scheduled to

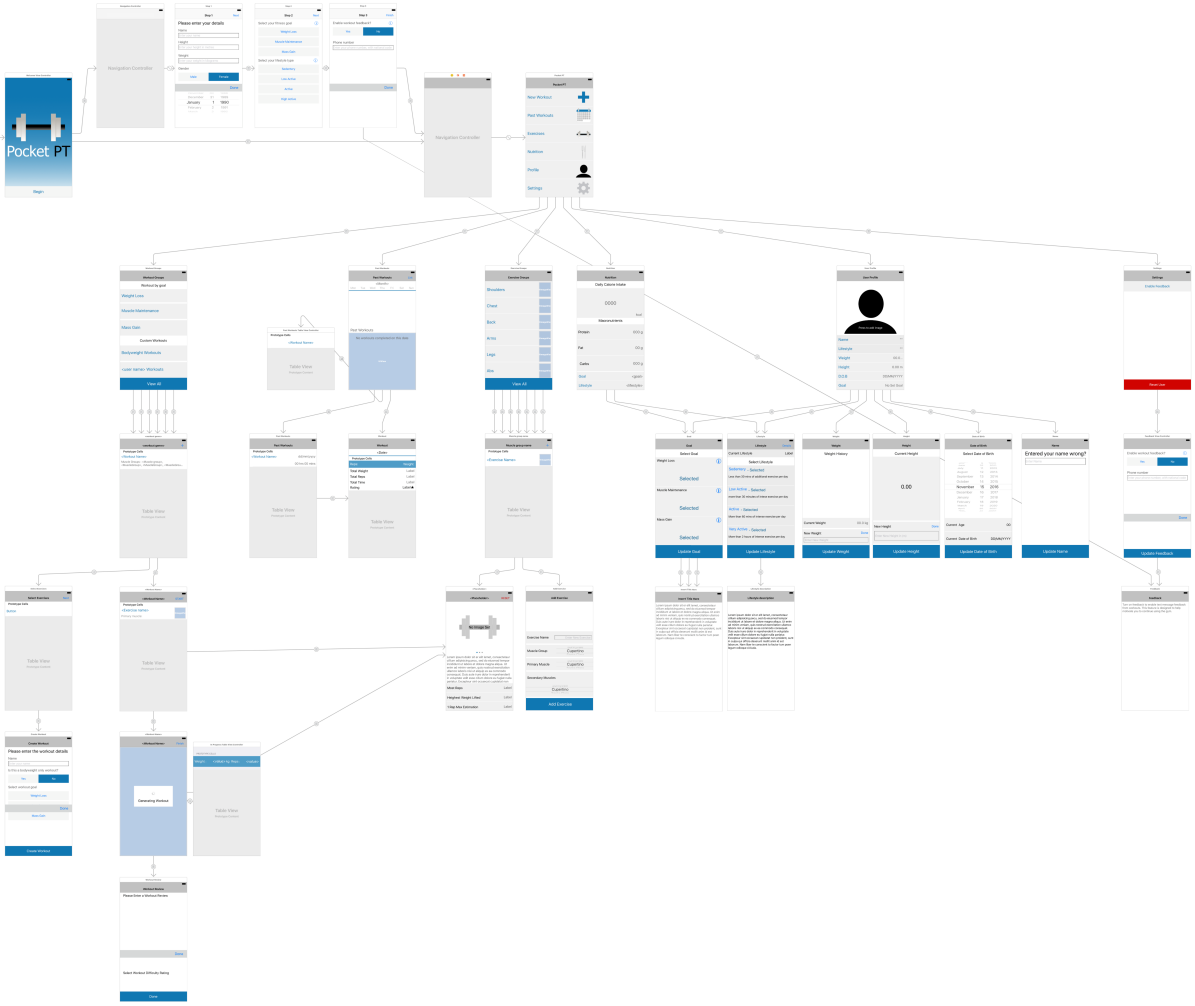


Figure 7.1: Overview of the system

be undertaken due to time constraints imposed on the project. However, the remaining course could be carried out if it was deemed as necessary.

The new CS139P tutorial chosen was designed focused on a more recent version of the language. Additionally, the number of resources available for this course were far superior to the original Apple course. The resources were not the only benefit; the quality of teaching provided in these classes was of a high standard, and it was soon evident that this course would supply the developer with a high level of education. The course was of great help to the developer, which was shown by the rate at which their degree of understanding grew in comparison to the old course. The supplied recorded lectures were the most useful resource, as they provided the developer with first-hand experience of how to use Xcode, as well as covering general principles of the Swift language. These principles taught could be applied to a variety of apps which contrasted the previous tutorial significantly, as the previous tutorial was highly focused on making a particular style of irrelevant application. Soon after undertaking the new course implementation of the app began.

The effect of this caused a re-evaluation of the project schedule, to re-establish what was possible for the project now a greater knowledge of iOS development had been gained. A decision was made to focus on the key features of the application, such as workout inference and nutrition. To enable these sections to be completed the aim of the app became proof of concept based as opposed to developing a marketable product. Additionally, certain aspects were dropped from the project such as integrating with the Watch; it was realised that this would require a full-scale development in itself which was not feasible for the project's schedule. Although modifications were decided upon for the project, the overarching aims were still kept.

7.2 Menu Screen

The first focus of development was the menu screen; this was chosen as it had minimal complexity and produced a good root for the applications bottom-up developmental as most features are likely to span from this screen. It was decided that the interface for this screen would be created first and the background functionality would be applied afterwards.

7.2.1 User Interface

To ease the initial implementation of the interface the implementation would be purely design focused, however, correct elements were used to enable the correct functionality to be implemented when necessary. To produce this screen two options were available; the first was to use the inbuilt features of Xcode which allows elements to be dragged and dropped onto an empty canvas. These elements can then also be modified programmatically if necessary. The second option was to purely create the elements of the screen in a programmatic nature from within that views controller. It was decided that Xcode's inbuilt functionality would be used wherever possible, this is because all of the inbuilt elements have swift compliment code supporting them, including touch and gesture detection.

Xcode's inbuilt functionality was used to create a representation of the home screen first shown in the design. This proved troublesome when sourcing the icons for each section; a complete image set could not be sourced, which had a consistent styling and usage that was not restricted by copyright. Therefore each icon was created by the developer from scratch, using design software and inserted into the screen within the storyboard views for each of the components. This ensured that the button used still overlapped this element to allow the user to press anywhere within a section to navigate to that component.

Although the screen appeared to be fine within the storyboard view in Xcode, when the application was run on the simulator the screen appeared completely different with elements overlapping and out of place. This was a result of how the items were stacked within the display, as there was no precedence of items, the controller could not establish which element should be drawn in what position.

The solution for this was to use stack views and constraints within the development environment. A stack view allows multiple different elements to be combined within one single view and these are aligned in either a horizontal or vertical stack; this would resolve some of the issues caused. Further work was still necessary to provide the controller with locations for elements; this was completed using constraints. Constraints were added to the elements of the screen where they seemed necessary; this still did not resolve the issue. The use of incorrect constraints was at fault now; constraints can be given as set distances from other items in the frame, or they can be given relative to the view that encapsulates them. Fixed constraints were assigned for each element. Instead, relative constraints should

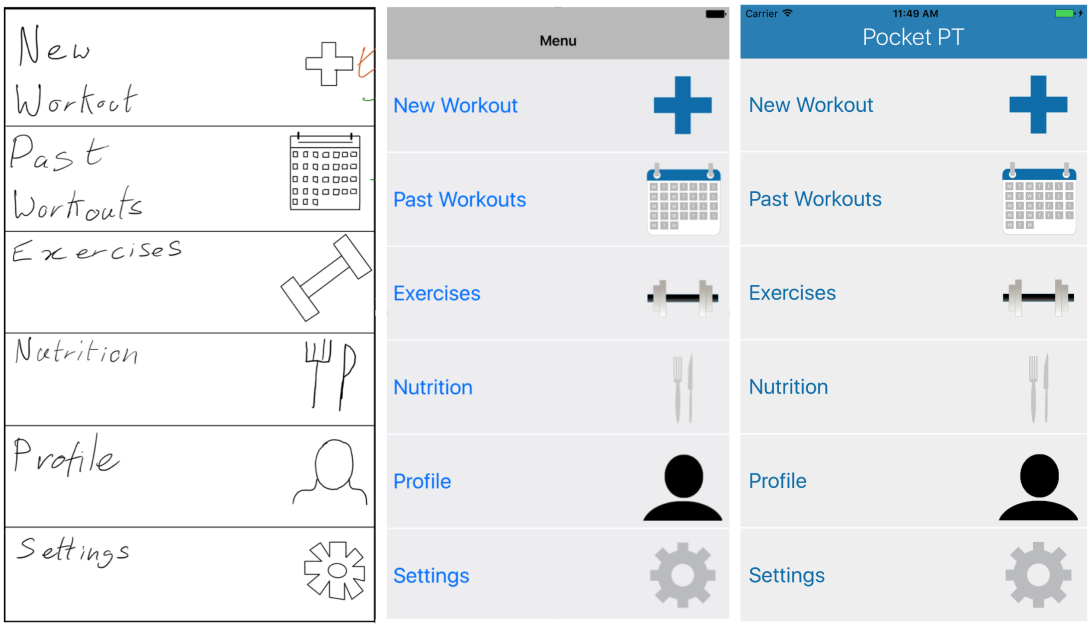


Figure 7.2: Comparison of initial menu interface mock-up and the implementation stages

have been applied; once this mistake was corrected, the screen showed as expected within the simulator. A screenshot comparison of the stages of the development of the menu screen is shown in Figure 7.2.

It can be seen how the initial design has been followed closely in terms of visuals, including the use of imagery to suggest operation as discussed in the research section 2.2.2. Both real world references and design standards have been employed here to further simplify the usage of the app. A navigation bar has been added, which includes a title for the page to provide the user further reference as to where the user is within the app. Although the navigation bar cannot be used on this page, it has been included to ensure consistency throughout the application, as all other pages will have this bar and relevant page title.

The far right screen of comparison in Figure 7.2, shows the revisions made towards the end of the project in the refinements stage of development. The focus of this was to improve the aesthetics of the screen which included redesigning the icons with a higher resolution, changing the colour scheme to tie in with comments received during testing and adjusting the text to give a cleaner appearance.

7.2.2 Functionality

The implementation of the basic functionality of the menu screen was minimal when compared to the user interface. Due to the correct usage of components within the interface, this allowed buttons to be used as navigation triggers; the code for this was also automatically generated as correct techniques were followed when connecting buttons with their corresponding screen which will be the navigation destination. There were still aspects which required coding from the developer and these regarded user information retrieval and user data transfer to the new controllers. The retrieval of the user information was simplified thanks to the use of Realm, which will be discussed later in Section 7.11. This resulted in easy access to the userData object which was then stored ready for the user. Additionally, Realm has the ability to update the object in memory if the object is updated within the database; this object will also be updated to ensure both objects are not contradictory. This feature means one user data object could be passed throughout the system, without the need for updates from the database. Within any view a method is called before a segue is performed; this method was utilised to pass the user data stored to the next screen selected by the user. Each separate screen requires a method for sending this data; therefore, each segue had to be given an ID to check which segue was being completed, allowing correct transmission of the user data. The code for this is shown in Figure 7.3; once this was completed the menu screen was completed.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

    // Pass the user object to the new view controller.
    switch segue.identifier! {
    case "newWorkoutSegue":
        let newView = segue.destination as! NewWorkoutViewController
        newView.user = user
        break;
    case "pastWorkoutSegue" :
        let newView = segue.destination as! PastViewController
        newView.user = user
        break;
    case "exerciseSegue" :
        let newView = segue.destination as! ExerciseViewController
        newView.user = user
        break;
    case "nutritionSegue" :
        let newView = segue.destination as! NutritionViewController
        newView.user = user
        break;
    case "profileSegue" :
        let newView = segue.destination as! UserProfileViewController
        newView.user = user
        break;
    case "settingsSegue" :
        let newView = segue.destination as! SettingsViewController
        newView.user = user
        break;
    default :
        break;
    }
}
```

Figure 7.3: Code snippet showing the navigation code

7.3 Nutrition

The next step in the implementation was to add a further screen to the application following the bottom up design. Based on the design mock-ups produced, the nutrition screen was most appropriate to implement next; this is because it is a single page component with a simple UI design that did not require further knowledge to implement.

7.3.1 User Interface

The majority user interface for nutrition was composed of views and labels. Additionally, a button was added to the bottom of the screen to follow the design mock-up drawn. The rest of the element of the display were also followed to produce a solution the was almost identical to the mock-up. This included a calculate calories button which would update the button text to re-calculate after pressing; this was implemented to give the user visual aids to show an action has been completed, even if the values have not changed. The screen and original design can be seen in Figure 7.4.

The final iteration of this screen was completed during the refinements period; the alterations were made based on the feedback from user testing. The most noticeable element change was the removal of the calculate calories button. Instead, calories are calculated when the user enters this screen; the technical functionality to provide this will be covered next. Calculating the calories on load ensures that the user always has the correct information provided to them. Additionally, including both lifestyle and goal within this screen helps show the user what their calories are tailored towards. Hence the user can now easily update these as they also work as buttons to navigate to the update pages for each parameter. To make it clear these options can be selected, they have been coloured in the same scheme as the rest of the buttons within the app to follow the design conventions.

7.3.2 Functionality

Th core functionality of the nutrition screen was in the calculating of the macronutrients and calories. The formulas created in the design section would need to be implemented; for this, a new nutrition calculator object was created to enable the code to be modular. The first method that was required was the base calculation for calorie intake; this was done using the formula from Section 6.3.5 and the code produced is shown in Figure 7.5. Due to design decisions, the PA value was calculated using the simplified method; this part of the calculation has been separated into a separate module to allow the calculation method to be changed without affecting the rest of the calculations if the design decision made is not proving effective.

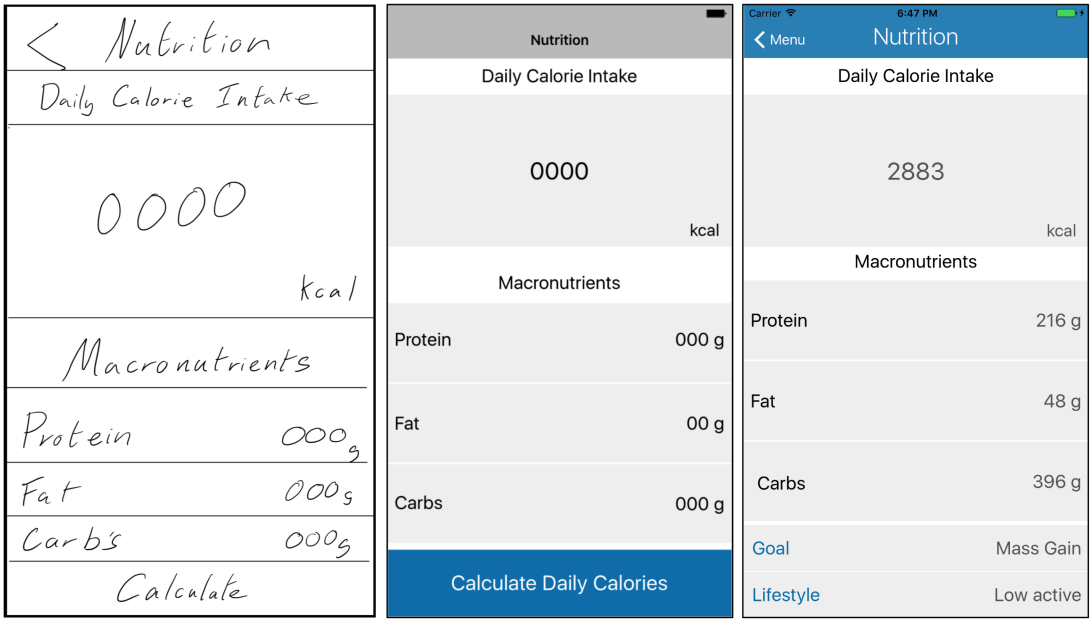


Figure 7.4: Evolution of the nutrition screen from design to final product

```swift
private func calculateTotalEnergyExpenditure() -> Double {
    var result = 0.0
    if (gender == "Male"){
        result = 864 - (9.72 * Double(age))
        result += calculatePA() * ((14.2 * weight) + (503 * height))
        return result
    }else{
        result = 387 - (7.31 * Double(age))
        result += calculatePA() * (10.9 * weight + 660.7 * height)
        return result
    }
}

private func calculatePA() -> Double {
    if (gender == "Male"){
        switch lifestyle {
            case "Sedentary":
                return 1.0
            case "Low active":
                return 1.12
            case "Active":
                return 1.27
            case "High active":
                return 1.54
            default:
                return 0.0
        }
    }else{
        switch lifestyle {
            case "Sedentary":
                return 1.0
            case "Low active":
                return 1.14
            case "Active":
                return 1.27
            case "High active":
                return 1.45
            default:
                return 0.0
        }
    }
}
```

Figure 7.5: The implementation of formulas for base calorie calculation

```
func calculateWieghtLoss(){
    dailyCalories = dailyCalories - ((weight * 0.0075) * 1000)
    protein = Int(weight * 2.7)
    fat = Int( (dailyCalories * 0.225 ) / 9 )
    carbs = Int(dailyCalories) - ((fat * 9) + (protein * 4))
    carbs = Int(carbs / 4)

}

func calculateMuscleMaintenance(){
    protein = Int(weight * 2.7)
    fat = Int( (dailyCalories * 0.225 ) / 9 )
    carbs = Int(dailyCalories) - ((fat * 9) + (protein * 4))
    carbs = Int(carbs / 4)
}

func calculateMuscleMass(){
    dailyCalories = dailyCalories * 1.15
    protein = Int(dailyCalories * 0.3 / 4)
    carbs = Int(dailyCalories * 0.55 / 4)
    fat = Int(dailyCalories * 0.15 / 9)
}
```

Figure 7.6: The implementation of formulas for modification of calories and calculation of macronutrients

The next step was to add in the calculations for each of the macronutrient categories as well as the calorie modification based on the goal. The formulas provided in Sections 6.3.5, 6.3.5 and 6.3.5 of design have been implemented and the code is shown in Figure 7.6. One adaptation was required to correctly implement this feature compared to design, and that was to reduce the number of calculations completed in one as this resulted in a too high complexity for the system. Therefore, some of the calculations were divided over multiple lines. Each of the specific goals have been separated into individual modules which are disconnected from the base calorie calculation and other goal methods; this allows any of the goals to be modified without needing to carry out any other modifications to the other methods. This is a major factor in the implementation of these methods as there exists no exact method for calculating the values, hence it is highly likely that further modifications to the code will be necessary for further development.

All of these methods in the nutrition calculator are called from the nutrition screen view controller on loading of the screen. The resultant figures are used to programmatically update the labels within the interface.

Once the core functionality of the nutrition component had been implemented, navigation between screens could be implemented between the menu screen and this screen. Navigation of the application would follow a stack model; each new page navigated to will be pushed onto the top of the stack, and each time the back button is pressed, a page will be popped from the stack. When using this type of model, there are issues that need to be considered. Firstly, as the view is loaded when it is navigated to, the companion model is initialised in the set-up code of the controller as well. The model is created here, and its properties are stored in memory for as long as this view remains in the stack. Therefore a cycle can be formed; this is where views build up on the stack and hold resources until a full memory error is triggered. To recover memory, the app is terminated. There are multiple ways a cycle can be created, the most obvious is by allowing a page to navigate to an ancestor of itself, by use of a button that is not the back button. The more complex scenario involves data dependencies; this was an issue that affected development. This issue occurs when a strong dependency is held between a parent and child variable from different views; if a strong dependency holds between views when the back button is pressed in the containing view of the child variable, the view's information does not get removed from memory. Instead, it is kept as it deems the information as required due to the dependency held. This is a hard to detect error, especially to someone who is unfamiliar with the language; even with experience, they can still be hard to detect and often, they are overlooked until a crash is caused. The solution to this problem is to force the child variable to make its dependency weak; this allows the controller to break the dependency if necessary, so that it can be removed from memory when the view is navigated away from.

When the nutrition screen was updated certain aspects of the UI changed requiring extra functionality; the first was that calculation was no longer user triggered and all of the previous button functionality could be removed. Therefore the method that was called by the previous button code was relocated into the method which is called just before a view is loaded. This method is slightly less efficient than the previous solution as nutrition is re-calculated every time the screen is loaded; the processing required to complete this is minimal, and the benefit of never having out of date information supplied is worth this slight efficiency decline. Additionally, the code for transitioning to the user profile screens was required

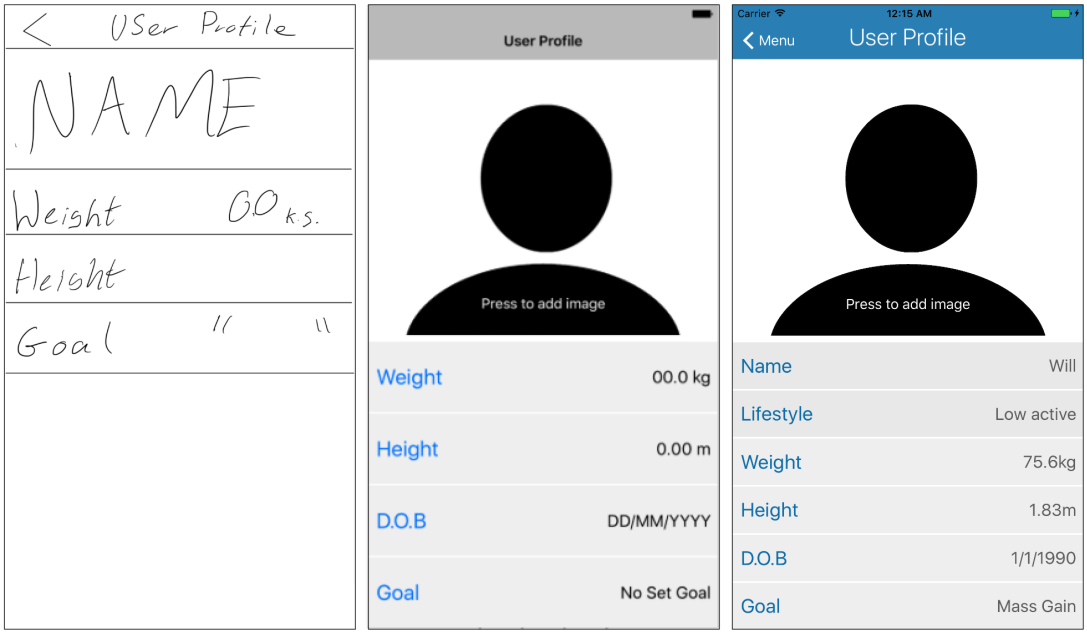


Figure 7.7: The progression of the user profile screen from initial design to final implementation

within the navigation method.

7.4 User Profile

A large majority of the units produced for this component are likely never to be used; this does not mean that they are not necessary. It is essential that a user can update any information they entered, whether it is because they entered it wrong initially or due to a change in their lifestyle goal or body characteristics. There are some aspects that are expected to be used more than others; these are the weight and goal fields. A user's weight fluctuates daily, and as a focus of the project is to help a user lose weight, being able to track their weight is vital in showing the user their progress. Additionally, at the other end of the user group, it is common for bodybuilders to go through different stages of training commonly referred to as bulking and cutting. For bulking the user adds muscle mass, and for cutting the user reduces their body fat percentage; therefore, the goal selection screen is likely to be used commonly by this target market.

7.4.1 User Interface

As this is the primary source for users to enter their details the main user profile screen went through many iterations; this was due to the fact more details were required for a user. Therefore, a method to update these values was also necessary. At the start, the first implementation of the user interface resembled the decision closely. One main change was made which was to add a user profile photo. The motivation behind adding a photo was similar to the motivation behind tracking weight; it is a method of displaying progress. Using a progress image in the user profile gives the user an ability to visually track how their body has changed, whether that be losing weight or gaining muscle, the picture will still produce motivation. When implementing the nutrition screen, it was evident an oversight was made, as there did not exist a way to update or view their currently chosen lifestyle options, hence this was added to the profile. Additionally, the name field from design was also included as well as the profile image that replaced it in the initial implementation. The interface progression of this screen can be seen in Figure 7.7.

In addition to the home screen previously discussed each field requires at least one separate view to handle data entry; each of the interfaces for these will be covered next.

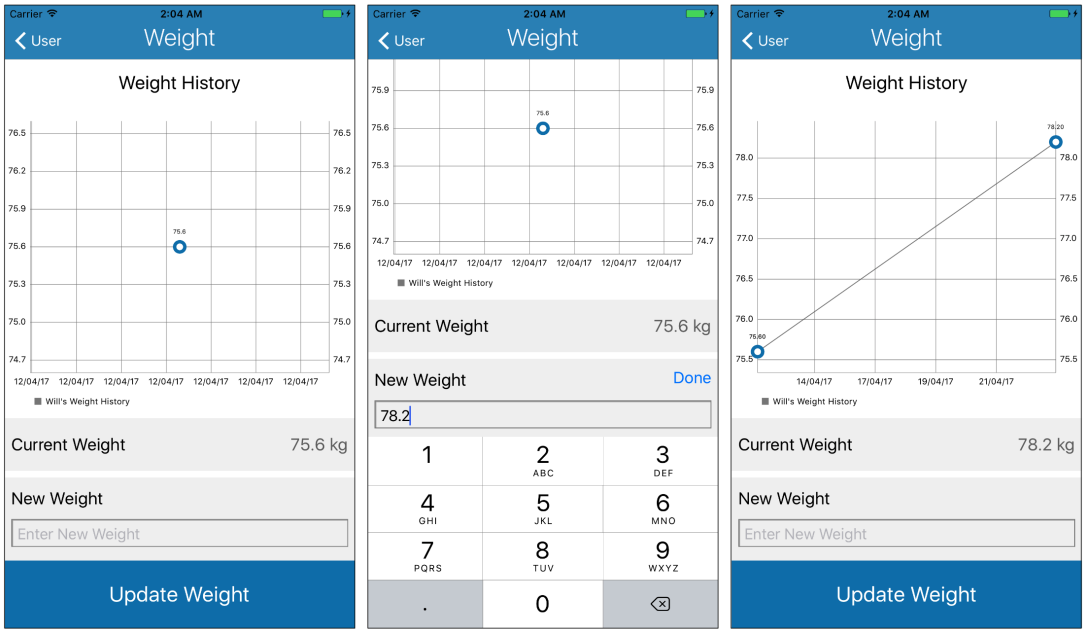


Figure 7.8: Final implementation screens of each of the user profile unit views, excluding weight

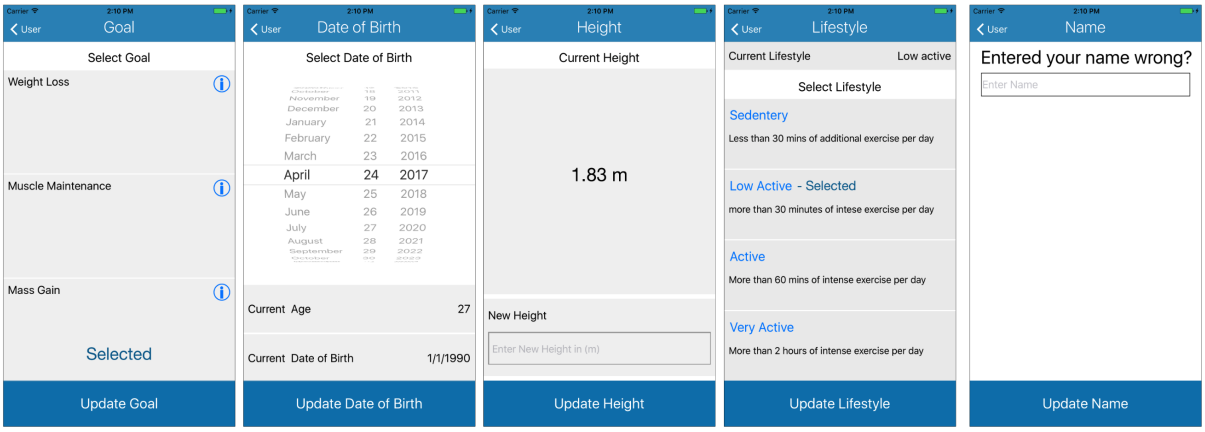


Figure 7.9: Final implementation screens of each of the user profile unit views, excluding weight

Weight

The focus of the weight screen within the profile is on allowing the user to track their progress regarding how their weight has changed. The best way to show this is through using a chart. Therefore a library was sourced that had sufficient material to support the development, regarding tutorials and examples [75]. The supporting tutorials were followed to produce the solution shown in Figure 7.8.

User Profile Unit Views

Each of the remaining screens for the user profile component all have a similar functionality to the views previously discussed, including programmatically updating values based on user selection, showing and hiding elements based on user interaction and allowing data entry. To avoid replication of implementation aspects, only the final implementation of the each screen has been shown in Figure 7.9.

7.4.2 Functionality

For the implementation of this component, two units will be focused on; these are the main profile overview screen and the weight screen. These two screens have been focused on due to their significance

in the project and the issues encountered during their implementation.

Profile Overview

The main feature that has not been implemented before for this screen is the profile image. The inbuilt photo selection handled linking the app to the photos stored on the user's device, as well as displaying this photo within the app. The issue occurred when the screen was navigated away from; the functionality provided did not store the photo permanently instead it was only in main memory, so would only remain while the view existed on the navigation stack. As the photo was in memory the first attempt was to recover the actual image from the photo element to store in the database. Realm limits what data can be stored in the database and photos are not allowed to be stored. A different solution was required, and the next attempt was dependent on being able to find the file path of the image and store this file path instead of the image itself. No such file path was accessible from the element, but manually storing files in the system was reached and this was possible. The solution to this problem was, therefore, to store the image file in a location which was known to the system, store this file path within the user's data and then recover it upon loading the user profile screen. The next screen where the background functionality was implemented was the weights view.

Weight

At the start of implementation for this screen no chart was included, instead only a Text Field existed, designed for weight entry. This caused an unforeseen issue regarding text entry and specifically the keyboard. The correct elements were used for the interface having a Text Field to allow user input; the inbuilt functionality for this automatically produces a keyboard overlapping the screen when this field was pressed. The first issue was that this was a weight field and the keyboard loaded was a standard QWERTY keyboard allowing users to enter text. If this were to happen the app would crash due to incompatible types attempting to be stored. The other issue was the most confusing to the developer. Once the keyboard had been brought up, there was no way to remove it from the screen; taps outside the keyboard or pressing the return button on the keyboard itself did not work. The screen could be navigated away from using the back button, but as the screen was removed from memory so were all the values for the Text Field; therefore this was of no benefit. The cause of this issue was due to incorrect use of delegates within the controller; the correct implementation was found within the lecture material and once these were implemented the keyboard could be removed from the screen with the use of the return button. The next issue to solve was the issue regarding the wrong keyboard allowing characters to be entered in a number field. Within the Xcode inspector section for the selected Text Field, a styling option is included for the keyboard. The styling was updated to produce only a decimal keypad for this TextField from now on. Upon the next test of this screen, the new keyboard was displayed. When the Text Field was selected, this keypad did not contain any return key which caused the original issue to occur again, requiring a different solution to the problem. As the method to remove the keyboard was now known a custom button could be created to perform the same action upon being pressed. A button was added to the screen, and it functioned successfully allowing Text to be entered and the keyboard to be moved away from. The final issue regarding text entry came after the chat view had been added to the screen; adding this view required the weight Text Field to be moved down to the bottom of the screen, to keep user focus on the chart and not a text box. Now when the text box was selected, the keypad appeared on top of the Text Field and the previously created done button. The text box and button could not be moved permanently as this would intrude on the space required for the chart. Instead the Text Field and done button needed to be moved dynamically when selected and deselected. The method required to do this was sourced online and adapted to fit this implementation; now on selection of the Text Field, the view containing the Text Filed and done button is moved up the screen above the keyboard. Additionally, as the done button is only required when entering the value, the same method is used to show and hide this button to remove ambiguity and confusion from the screen when the button is not required. The code produced to solve the delegate and movement issues is shown in Figure 7.10

Once the components of the charts library had been implemented within the UI, correct background functionality was required. The tutorials supplied within the library were used along with the corresponding demonstrations to produce the core methods required. The most difficult part of this implementation was converting from the storage object implemented and the necessary data type for the application. The X and Y values for the different points were required in separate list objects which would then be converted into datapoint objects required by the library, which in turn would be used to create the chart

```swift
// MARK: UITextFieldDelegate
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    return true
}

func textFieldDidEndEditing(_ textField: UITextField) {
    //currentWeightLabel.text = textField.text
    self.animateTextField(textField: textField, up:false)
}

@IBAction func textEntryComplete(_ sender: Any) {
    newWeightTextField.resignFirstResponder()
    doneButton.isHidden = true
}

func textFieldDidBeginEditing(_ textField: UITextField) {
    doneButton.isHidden = false
    self.animateTextField(textField: textField, up:true)
}

//method to move text field above keyboard
//modifed from : http://stackoverflow.com/questions/1126726/how-to-make-a-uitextfield-move-up-when-keyboard-is-present
func animateTextField(textField: UITextField, up: Bool)
{
    let movementDistance:CGFloat = -140
    let movementDuration: Double = 0.3

    var movement:CGFloat = 0
    if up
    {
        movement = movementDistance
    }
    else
    {
        movement = -movementDistance
    }
    UIView.beginAnimations("animateTextField", context: nil)
    UIView.setAnimationBeginsFromCurrentState(true)
    UIView.setAnimationDuration(movementDuration)
    self.view.frame = self.view.frame.offsetBy(dx: 0, dy: movement)
    UIView.commitAnimations()
}
```

Figure 7.10: Code example of the methods used to solve issues with Text Field entry within the weight view

dataset unique to the chart implemented. The use of dates as a value for the X-axis also complicated the implementation and required an additional method to convert from seconds since 1970 to a date style object to be displayed. Once all of this implementation had been completed, the chart functioned correctly.

Each of the remaining individual unit screens will not be covered, as their function is limited to data retrieval and entry. Additionally, the issues encountered within this section enabled the same problems to be prevented when creating the supporting background functionality for each of the user profile screens previously shown in Figure 7.9.

7.5 Exercises

This component will have a high usage and will be critical in providing newcomers with sufficient knowledge to be able to carry out an exercise in a safe and controlled manner. This is important to experienced users also as it should also provide some personalised statistics about each of the exercises.

7.5.1 User Interface

Based on the designs produced, three screens were created for this component; each of these will be broken down further detailing how the interface for each was produced.

Exercise Groups

The design for this screen follows the design mock-up almost exactly. One alteration was made which was to add an all exercises option within the sections. One aspect of the design that was not produced was the addition of the muscle group images. The element to hold the image along with all the necessary constraints are present in each of the group options; no images were included was due to the lack of images available. No consistent image set could be found that meets the necessary legal requirements. Therefore, an image set would have to be created for these options. At this stage of the project time is not available to create the image set as other features have a higher priority; additionally, the lack of images does not affect the function of this screen. The construction for this screen was taken mostly from the home-screen as they share a similar design and containing elements. The final product is shown in the left-hand image of Figure 7.11.

Exercises

Compared to the original design, the exercise list screen changed a great deal for the implementation. If the designed screen were to be implemented, it would have produced a cluttered interface that would be difficult to use. This was mostly due to there being two lists; one for primary and one for secondary muscle focuses. Instead of showing exercises where the muscle was worked as a secondary muscle, only the exercises that work the muscle as their primary muscle will be shown. The contents of this screen needs to be adaptable as this screen will show the list of exercise for all muscle groups. Because of this, the list needs to be adaptable; the table view element within Xcode is most appropriate for this as it provides the exact functionality required. To use a table view within the storyboard a new controller is required that is unique to a table view. Initially, the table view is configured only to contain static elements that are hard-coded in, and then the table view provides the scroll functionality expected of such a list. This was not the desired operation for this screen as the exercises needed to be loaded dynamically; for this to be done a prototype cell was required to be produced. This cell was created as if it were its own view, thus requiring the same supporting files and programming. This cell was then implemented using the Xcode elements like all the other views; the cell contained a button which would act as the exercise label and a small image of the exercise. On selection of the exercise, the screen would navigate to the exercise details screen. The implemented table view and the cell can be seen in the central image of Figure 7.11.

Exercise Details

The implementation for this screen followed the design exactly showing a workout image as the main focus of the page; this included small page indicators below the image as it is likely multiple images will be included for each exercise. Below the image was a scrollable text section; for this, a new Xcode

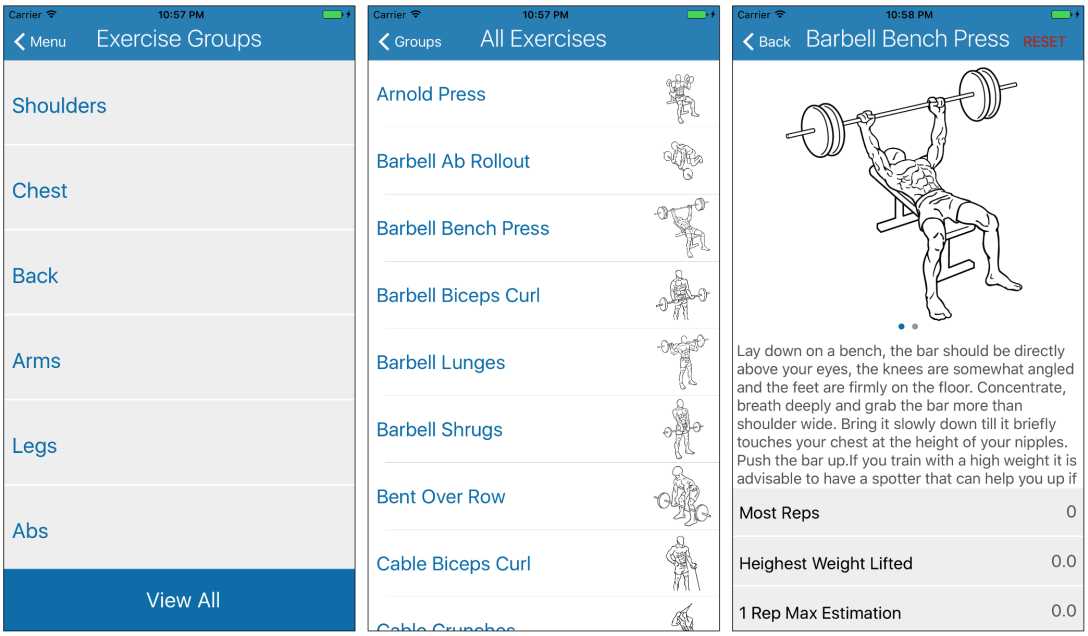


Figure 7.11: Screens for the exercise component of the system

element was used as it was important the preferences were right for this text view. It was possible to allow user text entry within the text view; this was not desired therefore it was set to read only in the preferences. At the bottom of the screen were views containing labels to show the user specific workout statistics suggested in design. The produced view can be seen as the right-hand image of Figure 7.11.

7.5.2 Functionality

The functionality required for the group's screen does not differ from the main menu screen; therefore the exact implementation of this will not be covered. Both the exercises list screen and exercise details screen had new elements; hence these have been included as sections to discuss the new aspects of implementation further.

Exercises

The exercises list screen was the first screen that required implementation of a table view and the attempted implementation resulted in numerous issues. To start implementation the original view and controller created needed to be deleted as specific table views and controllers were required. It was soon evident why when the controller file was opened; this was provided with a plethora of functions ready to be implemented. The size of this list made it difficult to understand what methods were required and which methods were not. Further tutorials were sought out to try and implement this feature. Once sufficient knowledge had been gained the implementation of this screen could begin.

The first step of implementation was to gather the exercises for the correct category chosen; upon navigation, the previous view set the muscle group which was selected, or an all value. For the all value, all the exercises stored are retrieved and sorted in alphabetical order within a list. For the muscle groups, the same query is carried out with the addition of a filter on muscle group by the name provided from the previous view. From this, a list of exercise objects now exists.

The first method that requires to be set is the number of sections in the table, as for this example no sections are used this is manually set to be 1. Next, the number of rows for the section are required. As there is only one section, this value is set to the size of the list produced from the query. The next step is to generate the cells and populate them with information. To retrieve a usable cell, it must first be dequeued and force-casted to the type of the prototype cell created in the storyboard. This cell will now have all the parameters set in the storyboard. Each of these parameters is set using the information about the exercise in the list, and then the cell is returned. Following all of these steps produced the table view, and the code to generate this table view is displayed in Figure 7.12.

```
override func numberOfSections(in tableView: UITableView) -> Int {
    // #warning Incomplete implementation, return the number of sections
    return 1
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // #warning Incomplete implementation, return the number of rows
    return exercises.count
}


override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "ExerciseCell", for: indexPath) as!
        ExerciseTableViewCell
    let exercise = exercises[indexPath.row]
    cell.exerciseButton.setTitle(exercise.name, for: .normal)
    cell.exerciseImage.image = UIImage(named: (exercise.imagePaths[0].path))

    return cell
}
```

Figure 7.12: Code listing to show how cells are implemented in a table view

Navigation between screens also proved to be an issue, as although navigation had been configured between the button within the cell and the exercise details page, the exercise details needed to be passed. This time a segue identifier could not be used as the selection mechanism for the transition like it has in the previously discussed segues. This was because all of the cells would have the same identifier; therefore a new solution was required. The method for this would be to use the button text to retrieve the exercise details from the database after a filter by name had been applied. This solution was implemented, and correct navigation was now functional.

Exercise details

In comparison to the previous screen, the implementation behind this screen was much simpler; the only new functionality required was gesture detection. For this feature to be implemented, first it was required that each of the elements from the interface be connected to the controller. Once all the elements could be accessed the functionality was added. Built into Xcode is a gesture recognizer object; this is transparent to the user, but it can be used to trigger methods. This gesture recognizer covered the full view of the image to allow all interactions on the image to be detected. Upon a gesture, a created image swipe method would be called; within this method, the direction of the swipe was determined. Depending on the swipe direction the image may or may not need to be updated as swiping right on the first image would not show an image to the left as the first image is already shown. If a swipe was detected and an image was available in that direction the image would be updated. For each exercise, there exists a list of the inbuilt system images which relate to that exercise stored with the exercise data object. Using this name, exercises can be loaded, therefore when a swipe occurred, the new image name was used to load the updated image. In this method, the page icon at the bottom of the image also needed to be updated. This occurred within the same swipe methods called earlier. For the remaining elements of the screen, implementation was repeated from other similar sections, with the exception of the reset button. This would start a method that would reset the data stored about the user's details for that exercise; this was done by writing 0 values to each of the user specific fields of the exercise. Once this had been implemented the screen was complete.

7.6 New Workout

This component will be the most used in the application as this is the feature used to carry out a workout. To enable the user to do this a large range of features and views were required; each of these will be broken down further regarding their interface construction and back-end functionality.

7.6.1 User Interface

Between the design and implementation of this component, multiple aspects of the interface design changed. There were multiple factors driving these changes, and each will be discussed alongside the implementation of the specific feature of the new workout.

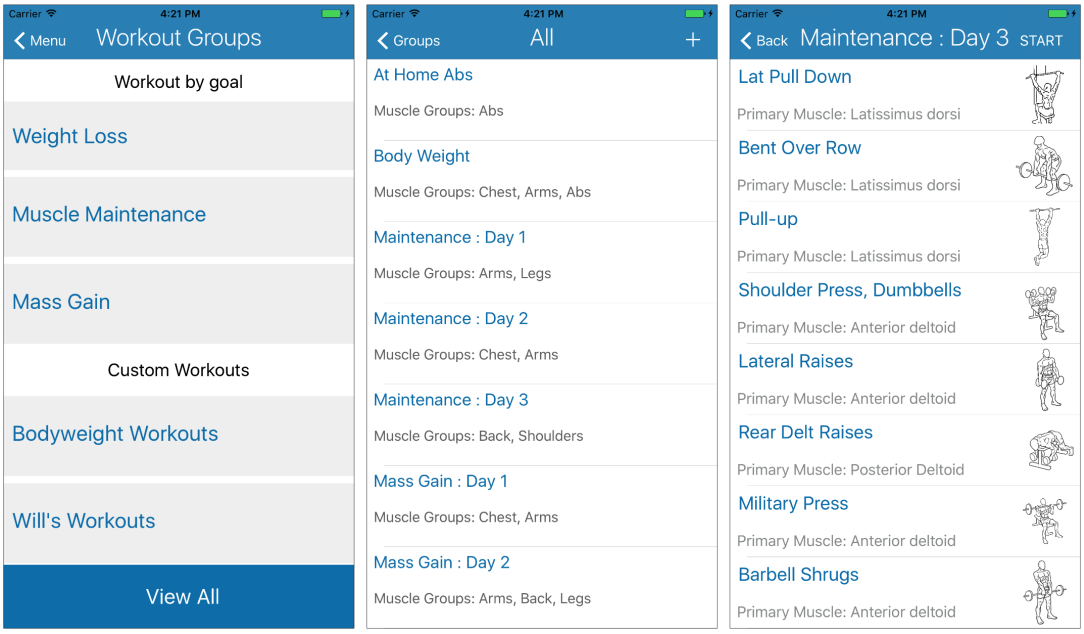


Figure 7.13: Screens showing the the new workout menu, an example workout group and an example workout routine

Main Screen

It was decided to reduce the complexity of the originally designed home screen as it was overly complex and filled with too many elements. Instead, an earlier screen would be used to refine the options before displaying all possible workouts. The design of this screen would be focused on the workout goals, bodyweight workouts and the users created workouts. The final screen for this is shown in the left image of Figure 7.15.

Workout Group

This screen displays all the available workouts which are specifically aimed at the goal or option chosen. Each workout is listed in alphabetical order. The name of each workout acts as a button to the workout schedule screen. Additionally, an add button signified by a '+' is placed in the menu bar; this navigates to the create workout screen. This approach of spreading features over multiple views is preferred to the ones originally designed; this is because it eases use by providing simpler screens that are less cluttered, thus allowing the user to select the desired option faster. The implementation of this is shown in the central image of Figure 7.15.

Routine Screen

The design of this screen is very minimal, but it provides the perfect way for a user to view a workout. The workout schedule is shown in a list, ordered by their execution. To further aid the user, demonstration images have been included next to the exercise name. This view allows users to quickly determine whether or not they would like to complete this workout. If they would, the start button is located in the top right corner which begins the workout and navigates to the in-progress workout screen. The implementation of this is shown in the right image of Figure 7.15.

In-progress Workout

The style of the in-progress workout screen has aimed to provide the user with all of the necessary information concisely within a single screen. Additionally, modifications have been added to further help the user. The first example of this is the ability to track which set you are completing; after selecting the set, the row will become highlighted in grey. When a new row is selected, the highlighting is removed from the previously selected row. The other aid is the inbuilt transitions to the exercise details upon

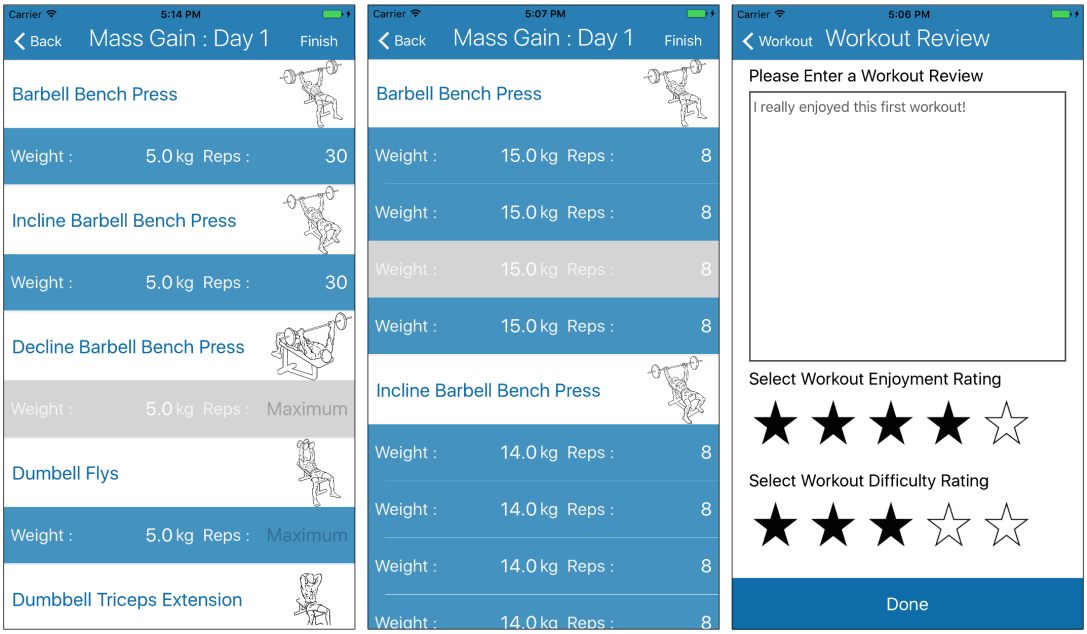


Figure 7.14: In-progress screen for new and existing exercise completion, alongside workout review screen

pressing the heading where the exercise is shown; this allows users to quickly gain insight into how the exercise should be completed without having to leave the workout screen and return. The appearance of an exercise's set will change dependent on whether or not an exercise has been completed before. If an exercise is new no repetition value will be supplied; instead, the maximum label is displayed to inform the user they should complete as many repetitions as possible and then enter the information. Regardless of the values generated, all the weight and rep values within the application can be user modified to show the actual workout completed by the user. This information will also be used in creating better recommendations for the user for future workouts; the exact details of this will be discussed later in the functionality. The final implementation of this view is shown in the left and central images of Figure 7.14.

Workout Review

The workout review screen consists of three components each of which are designed to be as simple to use as possible. The main element is the text view to allow the user to enter a description of their experience of the workout. Producing a solution that is as simple to use as possible will help the uptake in users who complete this section, which in turn will allow the system to function optimally by providing feedback. Two additional star ratings are provided to gain more information about the user's workout. Currently, these ratings are not utilised, but going forward it will be good to have a collection of results to be used to produce better recommendations. This will benefit the user still in its current state, as they can see how difficult a past workout was; this can then be used to track improvement. The final implementation of this view can be viewed in the right-hand image of Figure 7.14.

Create Workout

The create workout design differed almost completely from the original design. Instead of having a single screen that acted as the workout display for the workout created, the first screen was based on choosing the exercises for the workout; therefore a list of all the exercises was required. This feature is targeted at the more experienced user; therefore exercise images are not included in this list. To add an exercise to the workout, the list cell is selected, and the cell colour is changed to show it is selected. The option for the user to specify the weight and reps for an exercise was also removed, as this does not follow the purpose of the application. Thus, even user created schedules will have programmatically generated weight and repetition values. The final design of this screen can be seen in the left image of Figure 7.15. The next section from this is based on allowing the user to provide the details for the workout, such as

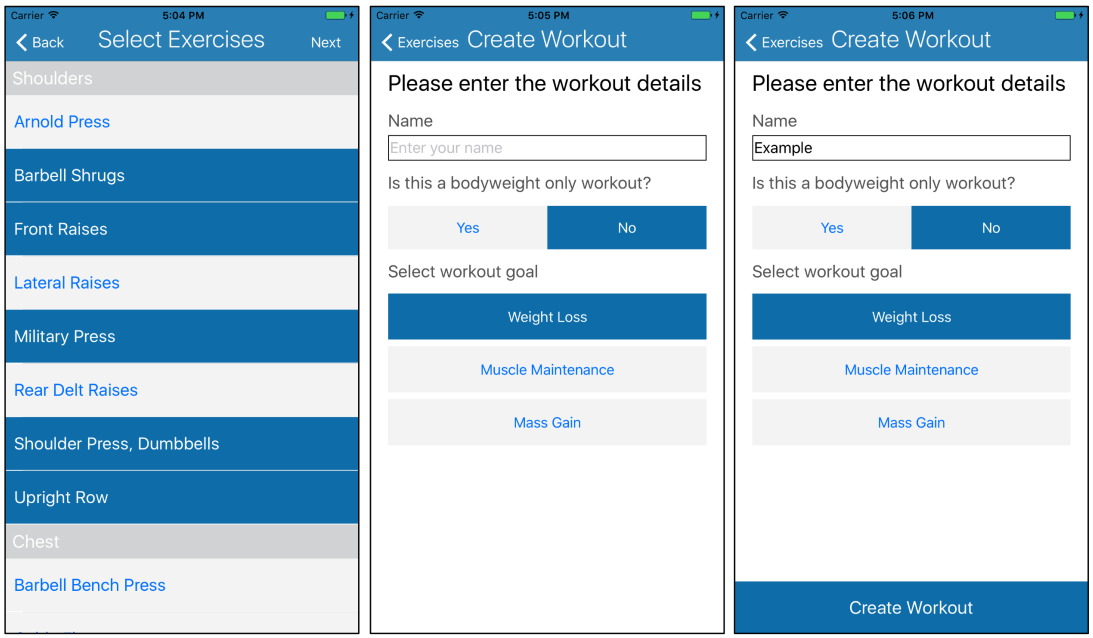


Figure 7.15: Create workout screens, showing adaptive display

name and their goal in focus for this workout. Once all sections have been completed, and a name has been given to the workout, a save button is shown to indicate that this section has been finalised. The example of the dynamic button display and final screen implementation is demonstrated by the central and right images of Figure 7.15.

7.6.2 Functionality

This was the largest of any component implemented, which can be seen from the number of views discussed above. The first three screens of this component all have similar functions to those already implemented; therefore none of these will be discussed in detail. The remaining create workout, in-progress workout and workout review screens will be discussed. The additional implementations for workout generation and the solution to the cold-start problem will also be covered.

Cold Start

Before functionality of the cold start workout generation feature could begin one aspect from design had been overlooked; this was the weight that should be lifted for the cold start problem solution proposed. A 5kg weight was chosen as the weight that would be used for this aspect of the generation. The use of a 5kg weight to gain the baseline for users strength is a choice that is motivated foremost by safety; this weight should not be ale to cause the user harm through incorrect completion of an exercise. The risk of using such a low weight is that the user may not reach their full potential concerning reps due to boredom. Although this may happen the next recommendation will be closer to their actual strength, and they should complete more repetitions of this exercise resulting in a further update of weight to lift until an appropriate weight is found. This can lead to the system requiring multiple sub-optimal workouts regarding the rep range; however, this is a worthwhile sacrifice to ensure the user's safety while using the application. This lower approximation should only be an issue when the user starts to carry out a new exercise as over time the system will become more accurate at predicting the recommended weight to lift. To know when an exercise had not been completed before, the most reps field within the workout was checked; if the reps value was zero, the exercise had never been completed before and the first time calculations were required. This particular weight calculation has been separated from the weight calculation method within the workout generator; this was done to maintain the modularity of code and allow the method for calculating the initial weight to be modified in the future separate to the core weight calculation function and vice versa. For the repetition and set calculations, the considerations for a first-time exercise are considered within the method, as it is unlikely they will change. The sets method

returns one for any new exercise as multiple sets are not necessary, the focus is on retrieving information. Similarly, the reps value calculated is '999'; this is a code that acknowledges the user interface that no text should be displayed in the field, leaving only the maximum placeholder text showing. The code implemented for this can be seen in Figures C.2 and C.3 of Appendix C. Once this element of workout generation had been finalised, the rest of the feature could also be implemented. The exact details will be discussed next.

Workout Generation

The main objective of the workout generation object is to produce a workout, based on a supplied routine. The workout should contain the number of sets to complete for each exercise, with the values for weight and repetitions supplied for each of these sets. To produce these values the equations built during the design stage of the development, discussed in Sections 6.3.2, 6.3.2, 6.3.2 and 6.3.2 were implemented. Where possible methods were used to separate the specific functions, this was to allow the code to be updated easier in the future. This is of particular importance for this object as the methods contained are likely to be adapted to provide additional calculation methods that may prove to be more accurate. The full code implemented for this can be seen in Figures C.1, C.2 and C.3 of Appendix C.

In-progress Workout

This screen is the main screen that handles the generation and display of the workout. A workout generator object is defined to allow the previously discussed methods to be utilised. Additionally, when the screen is about to load, the workout is set to be generated; at the same time, a loading spinner is set to appear on the screen. Once the workout generation has finished, the loading spinner is set to be hidden. Due to the size of the workouts, it is unlikely the user will ever see the spinner as the generation is produced quickly. Despite this it is import the user is aware the application is loading data if the generation is slower for an exercise; this follows the HCI practices defined earlier. Once the generation is complete, the workout object is passed to an embedded table view which is used to show all of the workout information. To follow the table view styling the prototype cell was produced to represent a set utilising two text boxes for weight and reps fields. There is a fundamental difference between this table view and the ones previously implemented. This table view will require multiple sections for each exercise. Additionally, there needs to exist a heading for the user to see which exercise they are completing. To create headings in table views, the heading view needs to be created programmatically on table generation. To speed up the design, a separate file was used called a 'xib'; this file allows general use views to be created. Therefore a view to match the heading design was produced. This xib was then used instead of manually creating the design using code programmatically during table creation, reducing the complexity of the code and also increasing efficiency. The exact code to produce this table is shown in Figure 7.16.

Create Workout

Most of the functionality for the create workout section uses aspects from other elements previously explained; therefore these elements will not be covered in further detail. There are aspects of producing this screen that did require new functionality. The first of these was regarding the selection of exercises. With any workout schedule, the ordering of the exercises plays a huge role in the effectiveness of a workout. All of the exercises shown in the view are displayed alphabetically by muscle group. Therefore the exercises chosen could not be taken in this order, as it is likely this scheduling will result in a sub-optimal routine. Therefore, the exercises would be stored in the order selected to ensure the routine selected by the user matches their selection. The first issue with this was that the selection occurred within cell object whereas the schedule would be stored within the table view containing the cell. To allow the cell to access this a reference to the table view was passed to the cell on creation; the permissions for the schedule were set accordingly as well to allow the cell to access the schedule. Within the schedule object, the list of exercises is already stored as an ordered list; this allowed the exercise to be stored in the order selected, giving the schedule in the order selected as required. The first issue arose as the ability to remove objects from the list was not implemented. Initially, the method only allowed adding of exercises to the schedule and not removal. A variable was used to check whether or not the exercise had already been added to the list; depending on this, the exercise was either added or removed from the list. This variable would also be updated on each operation. To remove the exercise for the list, the index

```
// MARK: UITextField Methods

func textFieldDidEndEditing(_ textField: UITextField) {
    let cell = textField.superview?.superview as! InProgressTableViewCell

    let index = tableView.indexPath(for: cell)!

    if (textField.tag == 1 && textField.text != ""){ //weight textfield
        try! realm.write{
            parentView.workout.sets[index.section].weights[index.row].weight = Double(textField.text ??
                "1.0")!
        }
    }else if (textField.tag == 2 && textField.text != ""){ //height textfield
        try! realm.write{
            parentView.workout.sets[index.section].reps[index.row].count = Int(textField.text ?? "1")!
        }

    }

}

// MARK: - Table view data source

override func numberOfSections(in tableView: UITableView) -> Int {
    return parentView.workout.sets.count
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return parentView.workout.sets[section].reps.count
}

override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "InProgressCell", for: indexPath) as!
        InProgressTableViewCell

    cell.weightTextField.text = String(describing: parentView.workout.sets[indexPath.section].weights
        [indexPath.row].weight)
    if (parentView.workout.sets[indexPath.section].reps[indexPath.row].count == 999){
        cell.repsTextField.placeholder = "Maximum"
        cell.repsTextField.text = ""
    }else{
        cell.repsTextField.placeholder = "Maximum"
        cell.repsTextField.text =  String(describing: parentView.workout.sets[indexPath.section].reps
            [indexPath.row].count)
    }

    cell.weightTextField.delegate = self
    cell.repsTextField.delegate = self

    return cell
}

override func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
    let headerView = UINib(nibName: "setHeader", bundle: nil).instantiate(withOwner: nil, options: nil)[0]
        as! InProgressHeader

    let pred = NSPredicate(format: "id == %d", parentView.workout.sets[section].exerciseId)
    let exercise = realm.objects(Exercise.self).filter(pred).first!

    headerView.nameButton.setTitle(exercise.name, for: .normal)
    headerView.image.image = UIImage(named: exercise.imagePaths[0].path)
    headerView.tintColor = UIColor.white
    headerView.parentView = parentView

    return headerView
}
```

Figure 7.16: The code used to produce the table view for the in-progress workout screen

of the exercise needed to be found, therefore the list was queried for the object's location and once the location was known the exercise was removed from the schedule. To avoid issues with navigation within this feature, the transition from this screen when a workout had been created needed to be considered. Up to this point either the back navigation button has been used or a transition has been completed to a new screen. For this transition, however, the next screen was the workout group screen; this was an ancestor of the current screen but not the direct parent. Therefore the back button could not be used, and if navigation were implemented in the same manner as others to this point, then a cycle would be created which could lead to a memory overflow issue as previously discussed. To solve this on the press of the create workout button both this view and its direct parent need to be popped from the stack in one go.

Workout Review

The final view of the new workout component is the workout review screen. Its primary function is to allow the user to provide feedback about the workout completed. Feedback is gained in two ways; the first is using the text view, the second is through a star rating. To retrieve the text from the text view requires one line. However, the star rating in comparison has a larger implementation. The reason for this is that there does not exist an element for this in Xcode. Instead, it was created by the developer. To increase its usability, this was built within its own view, allowing it to be used in any of the other views throughout the application. Once feedback is supplied by the user, the review information is also sent to Octoblu via an HTTP Post if this feature us enabled. The exact implementation of this feature will be discussed below in Section 7.12. The other primary function that happens within this unit is the aggregation of workout statistics; every set of the workout is aggregated to give total weight and reps lifted. Additionally, the start and end times of the workout are used to provide a duration. The most important step that occurs is the new calculation for 1-RM; for each exercise, every set completed is used to make a new 1-RM estimation for that exercise. If any of these produce a 1-RM higher than the currently stored 1-RM, this becomes the new value. This checking at the end of every workout allows the system to provide a more accurate recommendation for the future.

7.7 Past Workout

The core focus for the past workouts screen and functionality was based around providing a calendar interface similar in style and operation to the calendar supplied within Apple's calendar application. At the time of design, it was expected that calendar functionality would be a feature built into Xcode, hence this feature would not be excessively difficult to implement. This was an oversight, as no such feature existed, requiring a full implementation of the calendar by the developer. Preliminary research showed that this would not be a simple task and that approximately four controllers and four corresponding .xib files would be required. Therefore it was decided that library should be used to provide the calendar view aspect of this screen; the motivation for this would be that it would be easier and less time consuming to learn how to use the library and slight modifications could be made if necessary. To find a library, Cocoapods was used and "JTAppleCalendar" [46], the highest rated package was chosen. The supporting tutorial for this calendar was undertaken resulting in many hours of design work to give the look and feel wanted. Despite following all instructions on the tutorial and an in-depth search through the provided material, desired operation could not be achieved. It was required that on a swipe the month would be updated, this would include the dates and the header also. Swipe events were detected, and the dates were successfully updated. However, the calendar header would not update to anything other than the placeholder text. Multiple different attempts were made to resolve this issue without success. It was decided that to reduce further loss of time a different solution was required which will be discussed next within the user interface.

7.7.1 User Interface

The remaining time to complete implementation on this screen was limited. Therefore a different user interface was decided upon to simplify implementation and prevent the issue from affecting the implementation of other components. Instead of using a calendar view to select dates then workouts from a list below, just the list element would be produced. Therefore a table of every workout completed would be included ordered from most recent workout at the top to the oldest workout at the bottom of the list.

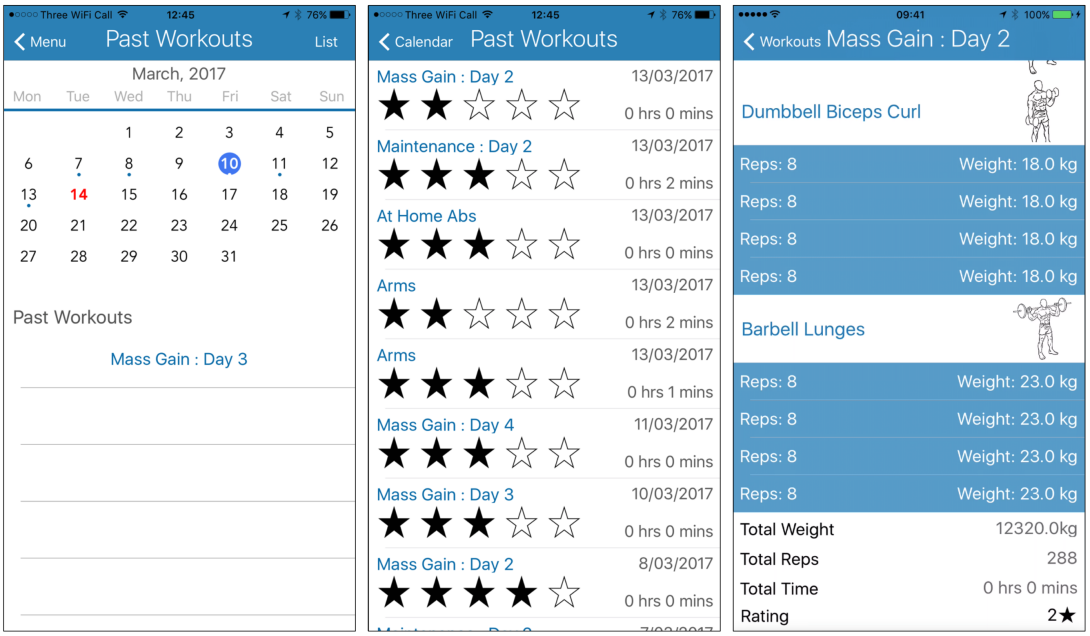


Figure 7.17: Past workouts screens showing calender view, list of completed workouts and previously completed workout with workout statistics

As table views had been created already the implementation of this approach would be simplified. A new cell for the table would be required to represent information about the workout; the chosen information to include in the cell were: workout name, workout date, workout enjoyment rating. In addition to this screen another screen was also required to display the workout; for this display, the design mock-up was strictly followed, and as suggested in design aspects for this screen were taken from the in-progress workout screen. A few minor alterations were required for this screen which was not included in the design; these were regarding the extra information gathered regarding the workout. To show the values for workout time rating and total weight, new cells were appended to the bottom of the table view to enable these workout details to be shown. The created screens are shown as the central and right images of Figure 7.17.

Although the implemented interface provided all the necessary functionality, the developer was not content with the solution to the problem and decided that other libraries should be considered as options. Again Cocoapods was the source of the libraries to review and "CVCalendar" [71]; this calendar had reduced functionality to the previous library used, but for this implementation, this was desirable as it made implementation and integration into the project much simpler than the previous calendar library. Once the new library had been selected, implementation of the new interface containing this calendar was completed in 2 days, as the old .xib design files were utilised. The new past workout screen with the calendar included is shown in the left image of Figure 7.18. To ensure the previous implementation was not wasted the option to change to the previously created screen is available in the top right of the new screen and labelled as 'List'. As set out in design when a day is selected all the workouts completed for that day are shown in the list below; if no workouts were completed for that day a message is displayed to make the user aware that no workouts were recorded for this date. Other interface modifications were added to further help the user, as a quick identifier that a date contains completed workouts, a small dot is placed below that date. Additionally, to provide the user with a visual aid of which date is selected, it will be circled in blue, and the text colour will invert to ensure the date is still easily readable. Another visual cue used is to highlight the current date; this is done by using a red font for this date only.

7.7.2 Functionality

For the workout list screen of past workouts, aspects of the previously written code for table views was utilised resulting in only a query to recover the required information from the database; as all past workouts were required for this list, all past workouts were selected from the database. This selection was ordered by decreasing date to give the desired ordering also. Once this information had been gathered,

the table could be populated with the correct information.

Similar to the workout list, all functionality for the viewing of the workout was taken from the in progress workout screen already discussed; this includes the utilisation of the same programmatically included header and sectioning for exercises. The only required additional implementation was to gather the additional static field from the past workout object and populate these fields.

The implementation of the main calendar screen required two screens to be implemented as the table view beneath the calendar was implemented as its own view which was then contained within the screen. This was done to increase the modularity of this section allowing either the calendar or list to be updated without affecting the other. The first step was, therefore, to link the two separate views together. This is completed when the page loads, as Xcode treats a container view as an automatic segue, allowing information to be passed to the contained table view. Methods and variables could now be accessed from the container view; this was required as the table needed to be updated upon selection of a new date. Additionally, the results retrieved from the database about the past workout also needed to be supplied to table view. The remaining functionality of the table view was created using the table views already created as a baseline.

The highest complexity for this screen was regarding the calendar itself; this had a large number of components requiring implementation. The calendar was produced through the use of the supplied tutorial; using this, each of the required views was connected as intended to provide the end functionality shown in the left image of Figure 7.18. Small adaptations were made to the supplied calendar methods to give the desired operation and allow the calendar header produced to be updated upon swiping, to correctly represent the month shown. Additionally, advanced queries were made to retrieve workouts completed on selected days. This was because a workout could have occurred any time within the chosen date. As the date objects also contain a time component which cannot be removed during a query, the query needed to filter the workouts by two dates. This would be the chosen date at 00:00 and 23:59, allowing a workout to be retrieved for any time that day. Once this query had been finalised the calendar had full operation.

7.8 Settings

There were a number of delays that occurred within the project which has been discussed previously. To account for these delays and the adjusting requirements of the system, a focus was placed on producing an application to show proof of concept as opposed to a market-ready solution. To enable enough time to be spent on the main areas of the project, aspects of the settings screen from design were not included. The first option removed was the ability to change the units used for the measures used within the application, the removal of this feature was seen as appropriate as it gave little functionality and time was required to complete other aspects of the project. Additionally, an app info and legal section were not included to also give time to other areas of the applications development. Should this application be brought to market, these features will be implemented.

7.8.1 User Interface

Only two settings options that remained were deemed necessary for the current level of implementation. The first was the ability to enable or disable the workout motivational feedback; the screens for this were utilised from the user creation screens, therefore the implementation of this required little overhead. The other setting was the option to clear the database. The settings screens are shown below to include the feedback screen as well as the info screen for feedback.

7.8.2 Functionality

The additional background functionality of these screens is very limited with the Xcode components and the previously implemented motivation sections performing most of the required actions. The only main requirement left was to connect the reset button to the back-end, to enable touch events to be recognised and trigger additional methods. A method would then be called which would erase all information in the database; this was done to ensure that no user data remained. The code for erasing the database is shown in Figure **??**

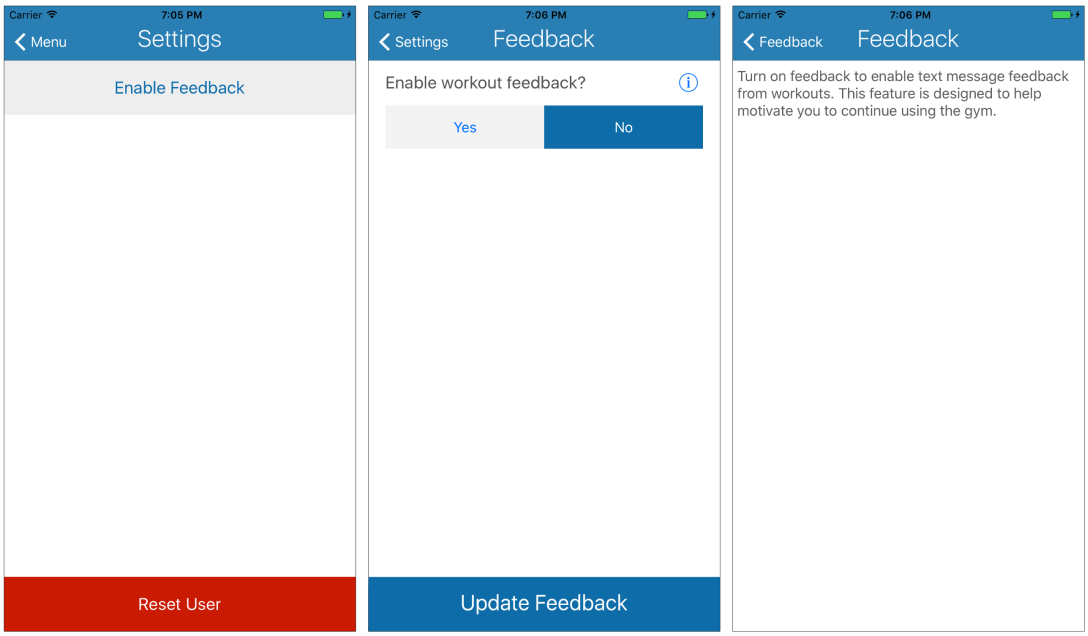


Figure 7.18: Settings screen, including workout motivation option and info screen

```
//MARK: Actions
@IBAction func clearDatabase(_ sender: Any) {
    //clear db of the old data
    try! realm.write {
        realm.deleteAll()
    }
}
```

Figure 7.19: Code snippet showing method to erase database information

7.9 Additional Implementations

An aspect of the implementation that was overlooked initially were the supporting designs and files for an application, these were the load screen and app icon. They are key to the application but were overlooked due to inexperience. As both of these require no functionality, only the design will be discussed in each.

7.9.1 App Icon

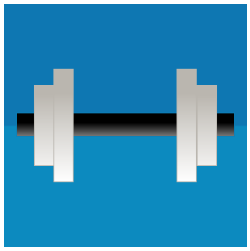


Figure 7.20: Pocket PT App Icon

Most applications follow the flat design style hand have a minimalistic look while linking back to either the apps name or function. As this section was started after the menu screen had already been implemented, the icons created here could be used. Therefore the dumbbell icon for the exercises was chosen as the main feature of the figure as this is frequently used as a visual metaphor for fitness and workouts. This alone would appear out of place. Therefore a blue background was added to tie in with the colour scheme of the application. The end design is shown in Figure **??**. Despite the fact that all iOS icons have rounded edges, this does not need to be accounted for; instead, this is handled by the development environment.

One issue that was not foreseen when designing was the varying image sizes required for the icon. Throughout iOS app icons are frequently used in different contexts; each of these requires the icon to be a set size. Therefore the icon had to be converted in multiple different set sizes for the application to compile successfully. Some warnings were produced, as the sizes for iPad devices were not catered for.

7.9.2 Load Screen

The initial styling of this screen was to use a stock image of a gym. However, this did not have the desired effect and seemed mismatched with the rest of the application. Therefore for the styling of this screen for other leading mobile applications were researched. This showed that the most common screens for the loading of an application were either an extension of the app icon or the company logo. The former of these two would be used for this application, whereby the dumbbell icon would be centred in the middle of the screen, and the application's name would be shown below. Again a blue background would be applied to maintain the constant styling throughout; however, a white gradient was applied to increase the overall appearance. The final welcome screen is shown next to the original stock image in Figure 7.21

7.10 User Creation

Another section for the application that was overlooked in design was the user creation screens, which gather the initial information from the user.

7.10.1 User Interface

At first the user interface for user creation was going to be based on incorporating the already created views from the user profile section into a new ordering. However, this was decided against as it would be cumbersome and long for the user to go through all of those screens; therefore a cleaner more condensed approach was decided upon. To further aid the user on sign up the title for the pages were numbered to quickly show the steps required for sign up. As different user information types would now be condensed, to keep the flow similar attribute gathering was combined for the different pages. User facts would be gathered first, followed by goal and lifestyle and finally, a feature screen would be added to enable motivational feedback, where if selected the user can enter their phone number otherwise this field is not shown. Constant styling remained throughout, with universal design principles employed, such as a scroll wheel for the date of birth, with the time removed. Additionally, selectable boxes which highlight to show selection are also included to provide a responsive interface for the user. The completed screens can be seen in Figures 7.22 and 7.23.

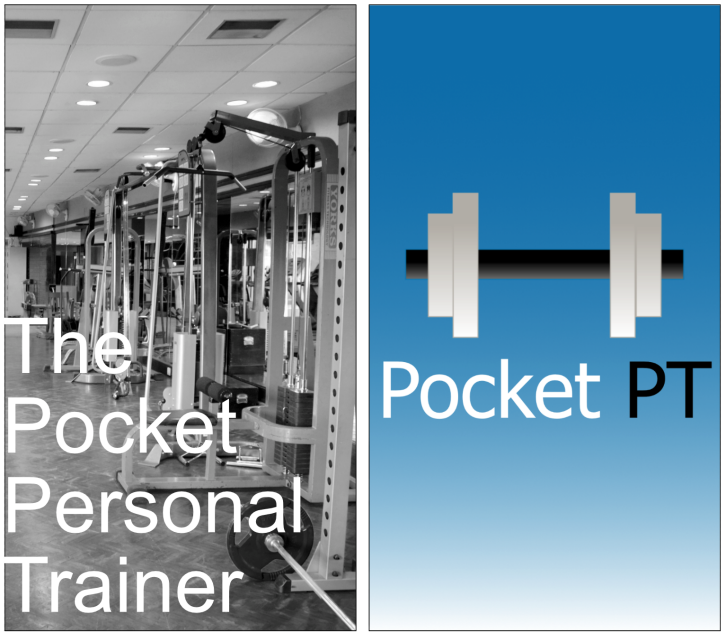


Figure 7.21: Comparison of welcome screen designs

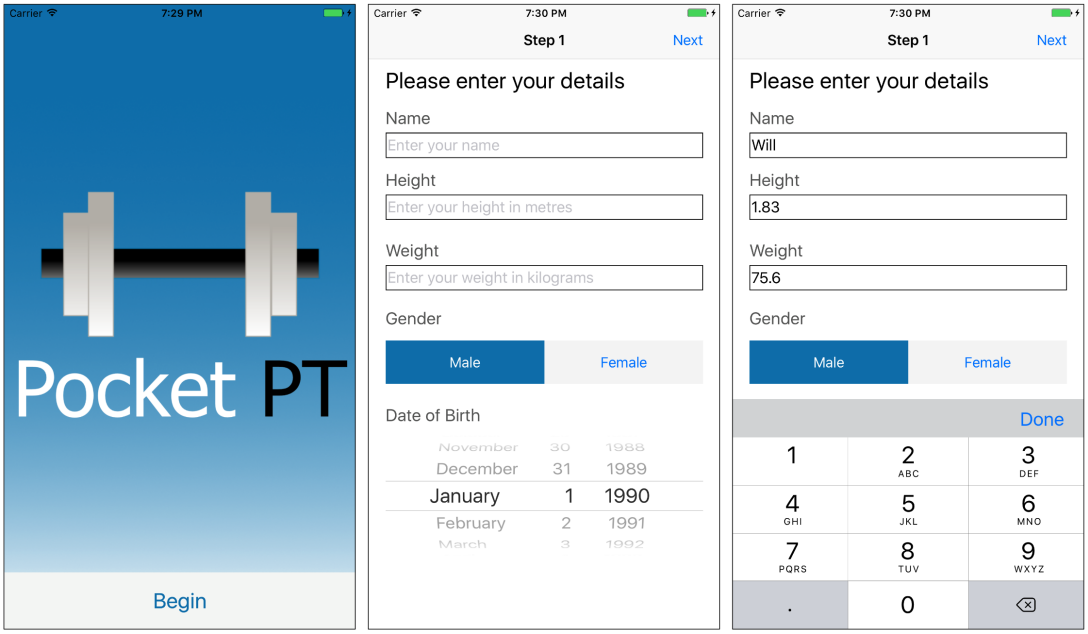


Figure 7.22: Start screen and step 1 of user registration, showing Text Field entry

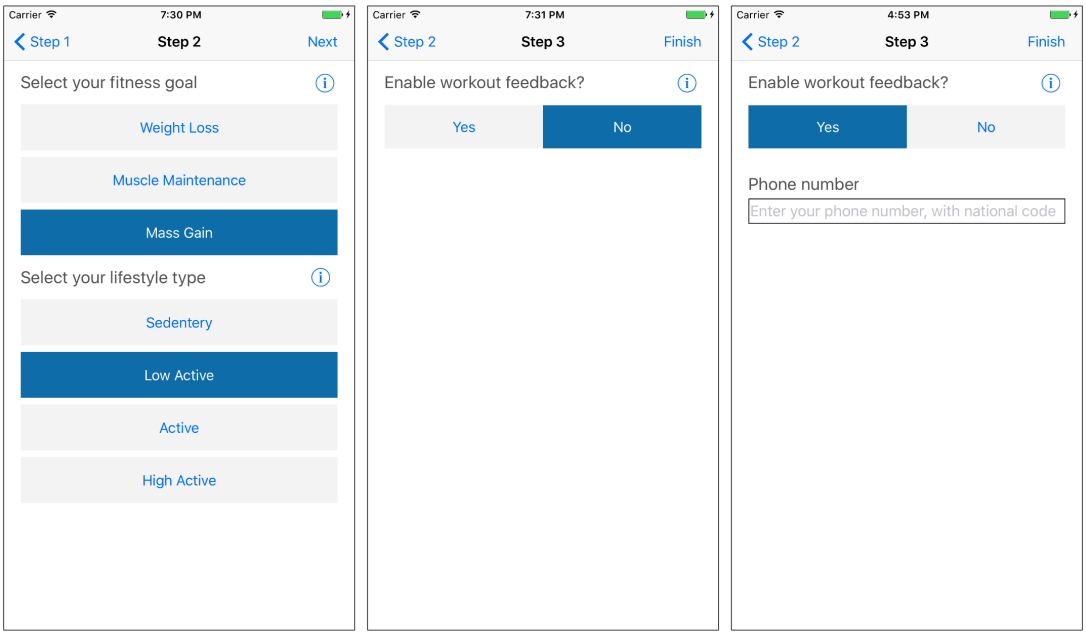


Figure 7.23: Step 2 and 3 of user registration, showing the dynamic screen for step 3

7.10.2 Functionality

The implementation of the functionality for this section was straightforward, with the only issue arising from the addition of the new screens as these were shown every time the app was loaded. The first solution was to add an existing user button, but this seemed to further complicate the system with no gain. Therefore a solution was employed that would check for the number of users stored in the database currently; if this were 0 the new user screens would be shown if this was not then a user existed, and the system would navigate programmatically to the menu page. Additionally, this was completed with a method that is run before a view loads. This then caused another issue where the user could navigate back to the beginning screen, which would force them to sign up again to be able to get back to the home screen or require them to close and re-open the app. To solve this, separate navigation controllers were used for the sign-up and main application. This stopped the sign-up views being navigated to outside of sign up as once the new navigation controller is navigated to all previous navigation controller information is forgotten by the application.

The rest of the implementation for sign up revolved either around updating the UI, or storing the information in the database. For the first step of sign up a new blank user was created, code fragments were taken from the previous user sections to handle text entry and write this to the user. Additionally, the done bar was also gathered from the user screen to allow the Text Field delegate to resign correctly when the user has completed that Text Field. There were also slight modifications to track when a new Text Field was selected; this was to allow the delegate to be changed accordingly. This code was added to the method called on the selection of a text box, and a global variable to signify which Text Field has the delegate currently. The user would then be written to the database upon the user pressing the next button, which would also navigate to the next screen.

The second step functionality heavily focused on UI updating as on the selection of a field, that field needed to change colour to represent it had been pressed. Additionally, all the other fields needed to be set to a unselected colour.

The final step required more programming to allow the phone number screen to shown and hidden as required. This was done by connecting the buttons to the background ground controller to detect a touch event; additionally, each element of the phone number section was stored as a variable. This allowed these elements to be altered programmatically, using a hide method to make the elements transparent to the user. This method was called when either of the buttons were pressed using the notification from the touch event triggered by a button; the yes button would show the element whereas the no button would hide it. If feedback was enabled and yes was pressed a flag would be set to true in the user data object stored. This would be set to false if no was selected and would also take place on the reception of

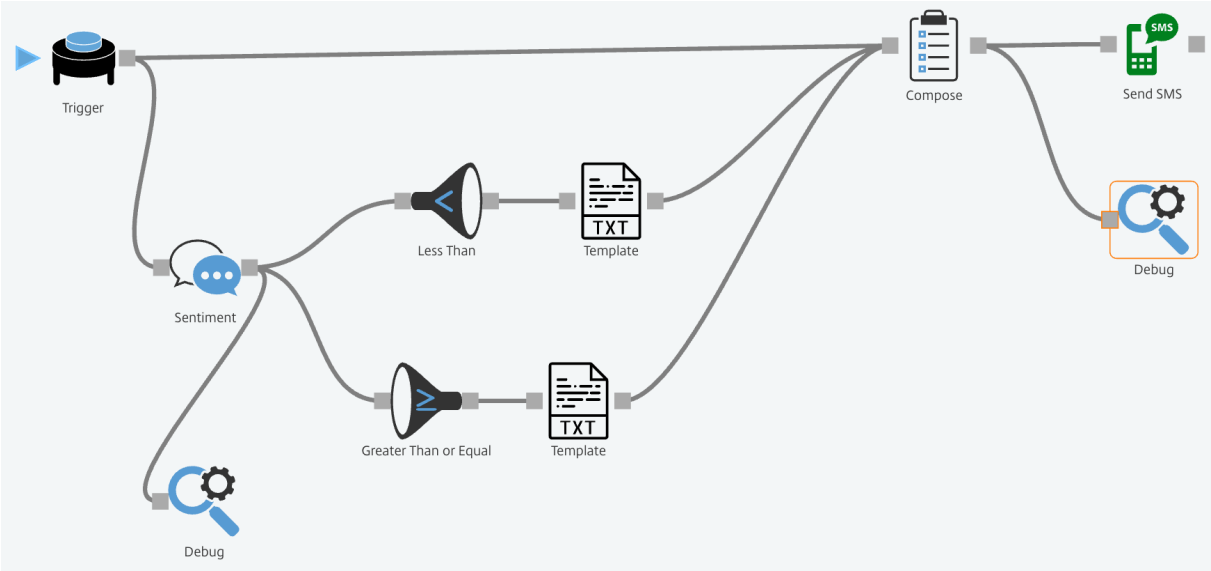


Figure 7.24: Octoblu Sentiment analysis flow

a touch event trigger.

There is a final aspect of the functionality that occurs directly after user creation. All of the information for workout schedules and exercises, do not exist within the application upon loading. Therefore the database needs to be populated with all of the workout schedules in addition to all of the exercise information, including the linking of the image stored to the exercise. This is done by two methods within the 'DataLoader' class. As stated these methods are run upon the user pressing the finish button. Once this has been completed the data remains persistent, hence it is only necessary on initial set-up up of the app, hence at the end of user creation is the most appropriate position for this to occur to prevent repeated loading of information.

7.11 Data Storage

An initial Realm database was created to match the schema set out in design. However, it soon became apparent when trying to access objects that there were critical flaws in the design of this. Most of these flaws spanned from developer inexperience with NoSQL databases and the ability to store objects; because this was possible whenever an object was required the object itself was stored. This lead to cyclic dependencies within the database, resulting in the same exercise object being stored unnecessarily within workouts. To solve this issue, particular objects were replaced with ID's to avoid the unnecessary storage of duplicate objects and prevent cycles. The Realm database schema was then adapted further on multiple occasions; this occurred when development reached a point where unforeseen information was required. Programmatically the overhead of this was simple to handle because of Realms object styling. Therefore, additional parameters could be added without significant overhead to the Realm data model as required. The final Realm storage objects code examples are included as Appendix B

7.12 Sentiment Analysis

To complete this feature two key components were required. The first was a networking component within the core application, specifically after a workout was completed. The second was an Octoblu flow which was discussed earlier in Section 6.5.

The Octoblu component was completed first, as the developer had previous experience with this technology. First, the essential components for the flow were sourced and joined appropriately to match the data flow diagram shown in Figure 6.3. This flow can be seen in Figure 7.24.

The trigger required no additional programming to function as it acts as a trigger at the start. The address for the trigger was recorded to be used later within the application as the HTTP Post destination.

```
func sendToOctoblu(){
    let parameters: Parameters = [
        "reviewText" : reviewText.text!,
        "difficultyRating" : difficultyRatingControl.rating,
        "enjoymentRating" : ratingControl.rating,
        "phoneNumber" : user.phoneNumber,
        "name" : user.name
    ]

    let address = "https://triggers.octoblu.com/v2/flows/3a4fe00f-0e73-400e-8a63-b05e9a0ceec4/triggers/fa645120-6703-11e6-ba6e-59ae2bcc34e1"

    //Not intrested in returned result, just fire and forget
    _ = Alamofire.request(address, method: .post, parameters: parameters, encoding: JSONEncoding.default)
}
```

Figure 7.25: Code snippet showing Alamofire web request

This was then connected to both the sentiment analysis and compose components. This allowed the items sent within the post request to the appropriate components. The review message was sent to the sentiment component, while all other items were sent to the compose component. Once the Sentiment component has received the message, the analysis is applied, and a value is established which correlates to how positive or negative this is. Hence a positive message would receive a value of 0 or greater and a negative message would receive a value of less than 0. For simplicity of these two extremes were the only values catered for. Therefore a neutral review would be treated as positive. Depending on this value, two possible text components could be triggered next within the flow. These each contain a constructed motivational message relevant to the sentiment rating. These both trigger the compose component which originally received the user's information other than the workout review message. The benefit of this component is that it does not trigger the next connected components until all of the stated parameters are received. Hence it can receive the user's information initially and then wait until sentiment analysis is completed to forward this information to the SMS component together. Once the SMS component receives the information from the compose component, the relevant sent fields are used to send an SMS message to the number supplied with the motivational message as the content of the message. This flow is running continuously waiting to be triggered by an HTTP Post request.

This implementation was completed using the Octoblu web platform which combines drag and drop of component and JSON style information passing. The system sticks to the key objectives set out in design, as well as following the requirements and surrounding issues correctly.

To complete the web request handling a package was installed called Alamofire [**?**]. This provided an easy way to implement HTTP Post requests; the implementation of this is shown in Figure 7.25.

To ensure the implemented system works as expected testing is required, the exact method and results of testing will be covered in the next chapter.

CHAPTER 8

Testing

Testing will be carried out using a number of techniques to ensure thorough coverage of the product. These methods will include Unit, Integration, System, Hardware and User tests. All of these types of testing are covered in detail in the following subsections. Each of the different test strategies will be designed to test the product against the requirement discussed in Section 4.

8.1 Unit

Unit tests will be carried out to verify that separate sections of the project function as expected. These will cover all of the functional requirements listed at the time of testing, as well as any other additions that are made to the code outside of the stated requirements. Additionally, tests will be as rigorous as possible to include forced failure and try edge cases to ensure the unit being tested has no issues. A white box testing approach will be used for the unit tests, as these tests require intricate knowledge of the system to try and identify possible edge cases in the code.

8.1.1 Strategy

As units are likely to only change by a small amount throughout the course of the project, manual unit testing will be carried out by the developer. This was decided upon as the overhead of producing the automated unit tests would be larger than the time taken to run manual testing on the feature.

To complete manual testing, the iPhone simulator built into Xcode will be used to simulate an iPhone environment. This method is preferred at first as it allows the software to be tested in a safe environment, without the risk of harming the physical device. Additionally, these tests can be run quicker, as the time to load the update is reduced when compared to loading onto a physical device. The real advantage of using the simulator is that it allows output to the terminal within Xcode; this can be used to confirm correct operation of background tasks that would otherwise be impossible to test. Another benefit of using the simulator is the ability to check the database manually using the realm tool earlier discussed in Section 5.4. This is an essential testing tool, especially when working with such a secure system like iOS, the ability to visually check a database for items to manually compare queries against returned results is invaluable in ensuring correct database operation.

For testing the functionality of the Octoblu unit of the system, additional inbuilt debug components will be used to check the produced information matches the expected output of a component. This allows each component of the unit to be tested as the information is otherwise unseen. The User interface of Octoblu also provides an enhancing effect to the component when it receives data. Therefore the flow of data through the system can be checked, and it also signifies when the flow is triggered, which again will be used to verify correct functionality.

8.1.2 Results

With the test-first approach, testing was underway as soon as a unit had been implemented; this led to a vast majority of the errors being resolved during the implementation of the feature itself. Examples of this have already been covered within Chapter 7. Due to the scale of the project displaying all of the unit tests completed would be infeasible. Therefore a sample of the unit tests completed for one screen will be shown as a representative example of how unit tests were completed on the whole system. These will be provided for the weight unit from the user profile; the details of each test will be shown in Table 8.1.

#	Test	Expected Outcome	Outcome	Result
1	Confirm all UI elements are correctly shown upon navigation to the screen.	The screen shown should resemble the one designed in the Xcode storyboard with and programmatic modifications also being enforced.	The screen produced as expected with constraints properly held and programmatically hidden elements not appearing.	Pass
2	Does the chart respond to touch events.	The chart view pans and zooms with user interaction.	Upon swiping the chart the view is panned, additionally, with a pinch gesture, the chart zooms in and out.	Pass
3	Are the programmatically updated labels and elements updated upon loading.	Weight chart is populated with values, and the current weight placeholder is replaced by a produced weight.	All values are generated, and the fields are populated with values.	Pass
4	Is the information about the user correctly retrieved.	Labels and chart show accurate data when compared to the physical data stored in the database.	All elements are correct and the values match the database.	Pass
5	On selection of new weight text field is the keyboard shown.	The keyboard is produced with an animation.	Keyboard is shown with animated entry.	Pass
6	When the keyboard is loaded does the text box moves to accommodate the keyboard.	When the text field is pressed, the screen moves up in accordance with the keyboard to allow the text box still to be seen while entering values.	The screen moves up when the keyboard is displayed, but the text box is still partially covered by the keyboard. Required work: adjust a parameter of moving function to allow the text field to be moved higher up the display.	Fail
6.1	As above.	As above.	The screen moves up when the keyboard is displayed, and the full text box is visible above the keyboard.	Pass
7	Does the done button appear when the text field is selected.	Done button appears within the new weight view adjacent to the heading.	Done button appeared in the correct location.	Pass
8	Does the done button successfully remove the keyboard from the screen.	The keyboard should disappear in an animated fashion, and the view should move back to its original placement on the screen.	The done button functions as expected and removes the keyboard from the screen, and the view is adjusted to match the original positioning.	Pass

9	Once selected does the done button become transparent to the user again.	Once pressed the done button should no longer be visible until the text field is shown again.	Done button remains visible once the keyboard has been removed. Required Work: A hide method needs to be added to the done buttons operation method.	Fail
9.1	As above.	As above.	Done button disappears once it has been pressed.	Pass
10	When the update weight button is pressed, does the chart and current weight label update.	The chart and weight label show the new weight entered.	Both the chart and label show the newly entered weight value.	Pass

Table 8.1: Table showing example of the unit tests conducted, showing the exact results of test completed on the user profile weight screen

8.2 Integration

The integration tests will be completed in a manner similar to the unit tests, as the system is being built in a bottom up approach the integration of two units can be tested when the new unit is added to the existing system. The tests will be completed using the simulator and will be completed in parallel to the development, to conform with the test-first strategy employed.

8.2.1 Strategy

For iOS development integration testing will mostly focus around confirming that the navigation between screens is functional, as well as that the necessary information being transferred is sent and received correctly. Checks on the resources in memory will also have to be confirmed to ensure that a dependency cycle does not exist between the two units. Finally, cyclic navigation between the unit added and any ancestors of the unit it was integrated with will be checked; again this is to confirm that there is not a build-up of repeated views in memory.

8.2.2 Results

Integration tests were carried out during the implementation as soon as a component was started and throughout the development of that component, therefore not all tests were documented. Additionally, a full breakdown of every integration test completed would be unnecessary, therefore, a representative example will be shown which tests the integration between the user profile screen and the weight screen previously covered for unit tests. A subset of the specific tests and results are presented in table 8.2

#	**Test**	**Expected Outcome**	**Outcome**	**Result**
1	Does navigation happen when the weight button is pressed within the user profile screen.	Swipe animation show the weight screen.	Screen correctly shown upon button press, with animated display.	Pass
2	Are all passed values correctly set on transitioning to new screen.	All local variables that are passed should be successfully assigned.	All values present on transition.	Pass
3	On selecting the back button is the user profile page navigated back to from the weight screen.	The user profile screen is displayed and the weight screen is animated away from with a reversed swipe animation.	The user profile screen is displayed when back button is pressed and the animation occurs..	Pass

4	Once the weight screen has been navigated away from are the variables from this screen removed from memory.	All variables that were in memory should no longer exist.	All values are successfully removed from memory.	Pass
5	Are their any navigation options in the child view that can connect to any of its ancestor views.	No navigations should be present in the code that navigate to an ancestor of that view.	No navigation options exist to an ancestor of the view.	Pass
6	If a value is updated within the weight section is this propagated back to the user profile.	User profile displays the new weight if one has been entered.	Old weight is shown on the user profile screen. Required work: A refresh method should be added to the view will load method. The refresh method should update the value of all labels to the current values in the user data object.	Fail
6.1	As above.	As above.	The new weight is successfully shown.	Pass

Table 8.2: A table to show a succinct example of the integration tests conducted, showing the results of tests completed on the user profile screen and the weight screen

8.3 System

System tests will be based on conducting general feature tests, which are less specific tests than the unit and integration test. This is because such tests would be time-consuming to complete and the depth of testing should have already been covered by the unit and integration tests.

8.3.1 Strategy

To test the core functionality of the app correctly, an extended test will be required where the app is used on an actual device and in the way intended. As the developer has a strong background in strength training this test will be carried out by the developer; this background knowledge will help prevent any bad recommendation by the application causing injury to the user. The system tests will be conducted over the course of a week where multiple workouts will be completed to include multiple completions of the same exercise and muscle group to verify all calculations made by the system are representative of actual lifting ability.

8.3.2 Results

Based on developers usage the app produces an application which is easy to use in the gym environment providing an uncluttered interface and simple exercise tracking. Additionally, the weights recommended where realistic of the developer's ability; at first the values may have been slightly low as expected. As more attempts of the given exercises were completed and updated entries of lifting capacity were entered, the recommendations made by the system became more accurate. There were occasions where the app produced a weight that caused the developer to struggle, however, this was not consistent, therefore this struggle is likely due to other factors affecting the developer's performance, such as energy levels, as opposed to being the cause of bad suggestions. Although these results are not conclusive, it did provide the standard of testing set out and proved to be informative, and suggests that the application is on-track to meet the project aims.

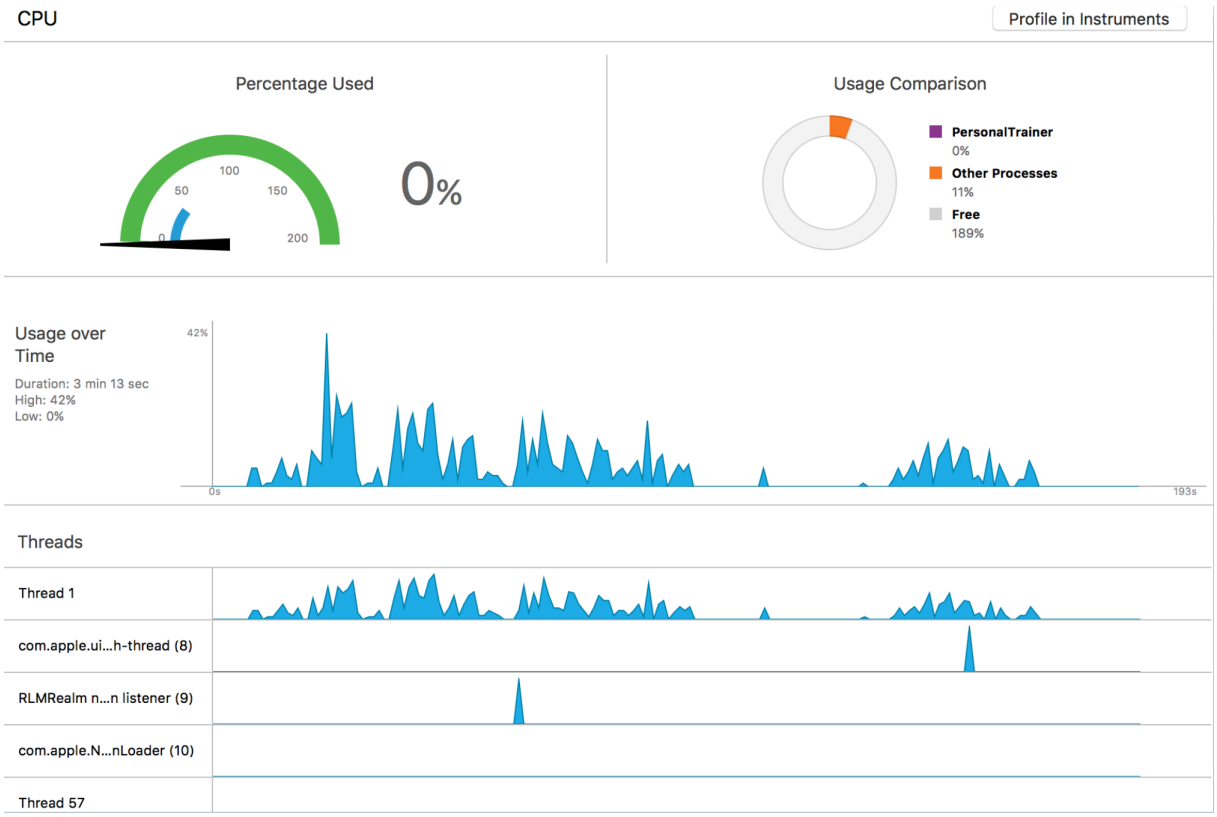


Figure 8.1: Graphical representation of CPU usage

8.4 Hardware Performance

Although hardware testing is uncommon in most system developments, from the early stages of this project it has been identified as an important area of focus, given that it is a mobile development.

8.4.1 Strategy

Xcode provides a performance utility when running the application via the simulator or by a connect physical device. To enable this to work correctly Xcode has to launch the application pragmatically. Once it has been launched the simulator or physical device can be used as normal. Throughout this usage, key performance statistics are recorded such as CPU usage, Memory usage and battery usage. These statistics are displayed using visual aids and graphs to represent the information. At multiple stages, through the development, these measures were checked to ensure that no currently implemented feature has an unexpected effect on any of the performance measures. Additionally, a final performance test will be carried out once the system is at a complete stage. This test will involve selecting and completing a workout, looking at the past workouts completed, changing the user's lifestyle from within the nutrition screen, before finally looking at a variety of the exercises from the catalogue. To speed up the test the workout will not be completed in real-time. Instead, some values will be edited before selecting to finish the workout; this should still give a fair representation of real use, as all complex operations are still necessary.

8.4.2 Results

The results of the final performance test is shown below in Figures 8.1, 8.2 and 8.3.

The metrics produced from this test were evidence to the efficiency of the system created, highlighting the minimal use of all primary resources of the mobile device. The resultant values were significantly lower than expected and far surpass similar apps. This is shown in the memory screen where despite the application running in the foreground, it was 5% of the total system memory being used indicating the requirements from the background applications were much higher in comparison to the produced system.

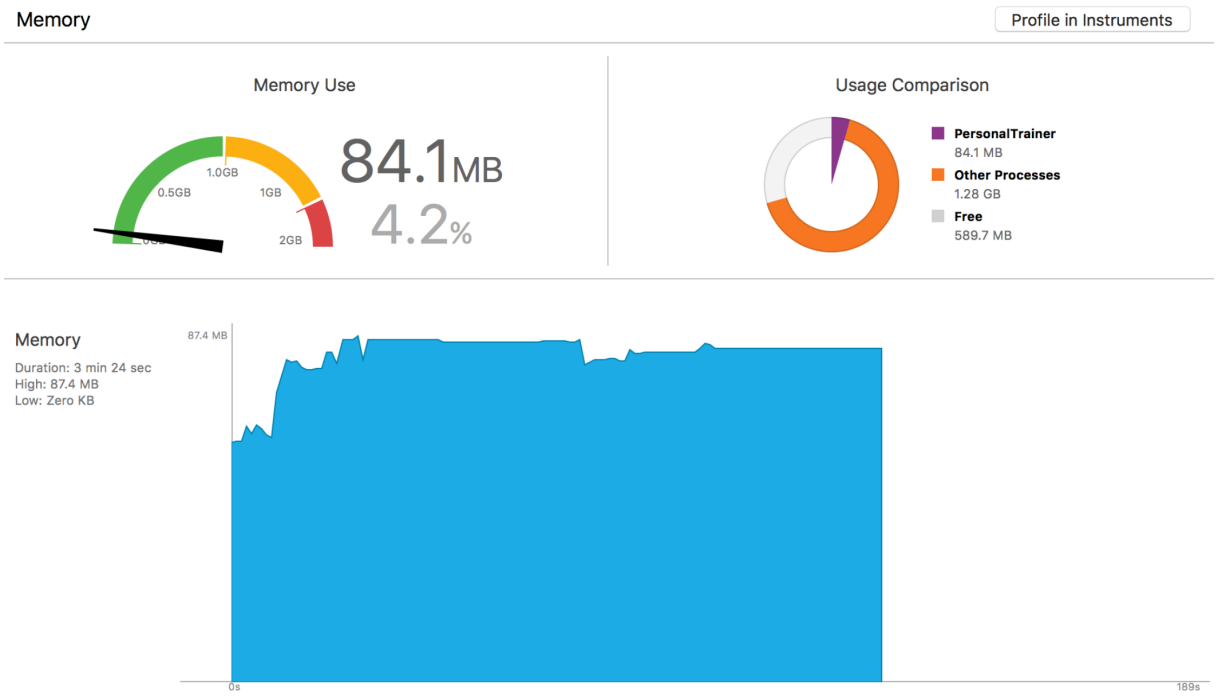


Figure 8.2: Graphical representation of memory usage

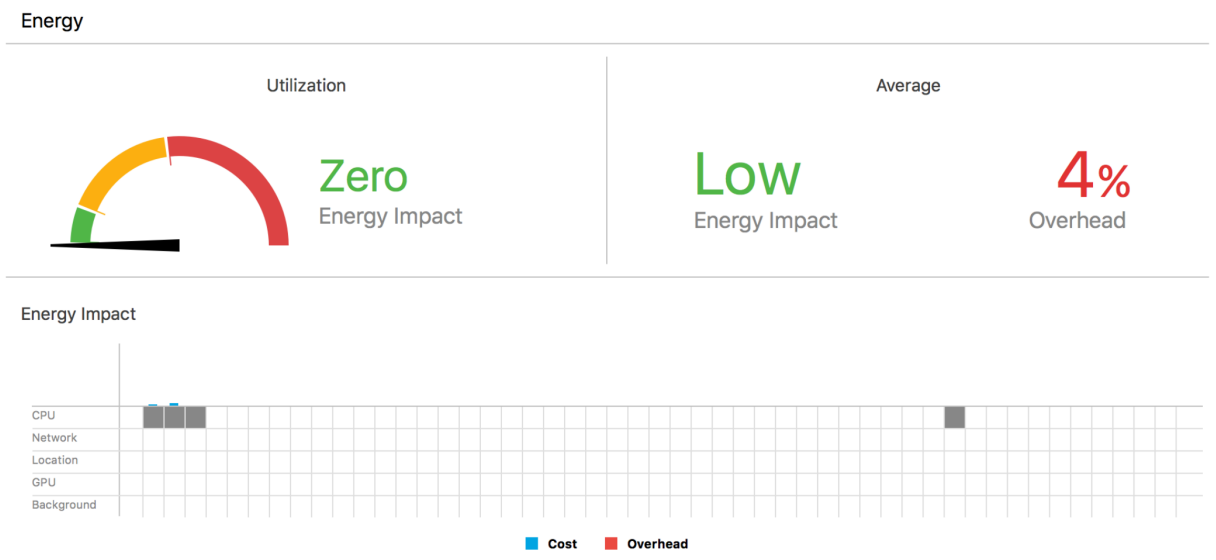


Figure 8.3: Graphical representation of energy usage

8.5 User Experience Testing

User experience testing is of particular importance for this project, as it is a user-focused development. Gaining user feedback will be crucial to not only the success of the project but the development also. To cover both design and functionality multiple rounds and types of user testing should be carried out. These tests should be carried out in a black box manner where the tester will have no information regarding the design or implementation of the product.

8.5.1 Strategy

To allow a full depth of testing user feedback will be sought out from early on in the development with a test-first approach in keeping with the agile aspects of the development methodology. For this testing, the development will not have finished, and features will be missing; this should not affect the results of these tests. The first user tests will be based on reviewing the system design concerning look and interaction. These will be conducted periodically over the course of the development. Once the system has reached a feature complete stage, user testing within the correct environment will be carried out with the developer present. A test subject with knowledge of the gym is desirable to avoid injury. Additionally, the developer will be present for support if any recommendation seems inappropriate for the tester.

8.5.2 Results

From the first round of user testing the most common comment regarding the design was the colour scheme used; the users commented that it "lacks colour, and appears boring"(Hannay, R). In response to this, the menu bar was changed to represent the blue colour scheme used throughout, as well as using a varying grey scale for labels and headings throughout the application. This softened the look of the application and reduced the harsh contrast. The implementation of this can be seen in the right-hand image of Figure 7.4, in comparison to the central image. The renewed design was shown to users after implementation and the new feedback received was much better.

A common comment from testers viewing the nutrition screen was the lack of details supplied in regards to their current settings. It was explained how the calories are adapted to their goal, but some users were unsure of their goal and thought displaying this on the screen also was useful. A solution to this was created and re-shown to the testers; the testers then stated that they would like to be able to change these from the nutrition screen, therefore, further updates were required. The results of this feedback were discussed earlier in Section **??**.

The first issue raised during the in-gym testing spanned from the recommendations, after multiple completed workouts the tester found that one exercise was producing estimations that were slightly too high and they could not reach the desired rep count. The tester then asked how they could reduce this number manually. At that point in the implementation there existed no way for the user to change this value as it was always programmatically generated upon workout completion. To solve this problem, the user suggested a reset button be added to the exercises to enable them to be recalculated next time to more accurately assess the user's current level of strength. This suggestion was implemented, and the feature will be particularly useful going forward for users who have stopped going to the gym for a prolonged period and their strength has dropped. Before this feature, the user would not be able to get accurate estimations, whereas now they have the option to reset an exercise without losing their other information.

During the testing within the gym, the user asked why the weight recommended changed between workouts. This change was due to the decay calculation incorporated, which was then explained. The tester then asked if this was also used reversely to produce a higher 1-RM estimation if an exercise is completed at the end of a workout. It was explained that for safety reasons the current implementation was more reliable. The tester seemed to think that a higher 1-RM should be reversely calculated, therefore with the further development of the application a safe way to incorporate this should be considered.

Another issue was uncovered during the user testing within the gym; it occurred when a user deleted text from a Text Field during a workout and then selected out of the Text Field, without inputting any new data. This de-selection crashed the application and was due to an attempted null write in the database. A solution for this was to revert the field back to the previous value when the field was deselected and the field had no value, hence, preventing null write. This solution was later implemented

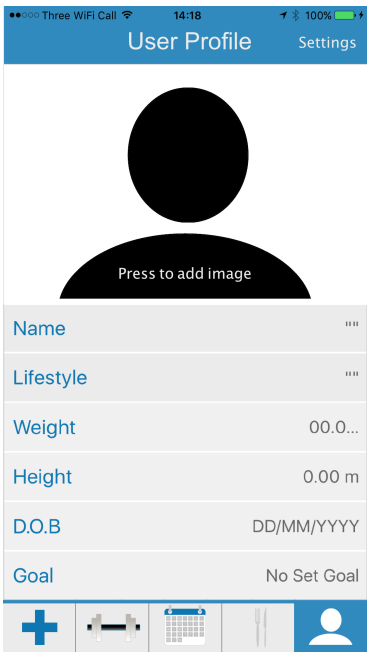


Figure 8.4: Example design mock-up 1 for a new home screen.

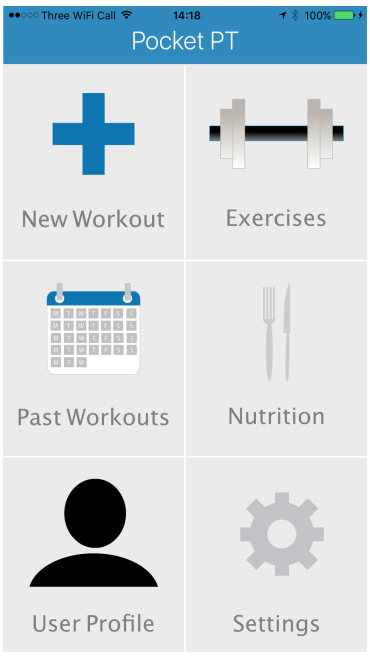


Figure 8.5: Example design mock-up 2 for a new home screen.

to avoid any future problems; a similar implementation was also applied to other Text Fields throughout the application to prevent null write crashes in other components.

Menu

The first problem with the menu screen was not brought up by any tester, but it was noticed by the developer. As the home screen has a table look, most users try and scroll on this screen, even though the screen is not scrollable and all content is static. Despite this not being a major issue it breaks the design conventions users are familiar with when using this type of device and therefore could be altered to match the conventions better and avoid confusion further. One issue that was raised was the fact that the images are located after the text on the home screen, meaning that a user would read the text first then look at the icon as they follow the left-to-right reading convention. Therefore the use of images as metaphors to increase system use speed was in fact redundant. A redesign to avoid this would be of significant benefit. As a solution to this some preliminary home screen mock-ups have been included in Figures 8.4 and 8.5.

Each of these screens has its advantages; the largest benefits come from the first design, as this makes navigation around the application much quicker. This speed up is due to each of the menu options being accessible with only one touch from any screen;. Additionally, these screens can be switched between enabling multi-tasking functionality. There are overheads to having this type of system design approach; using this screen style would significantly increase memory usage as each view is effectively in memory at the same time, and each with a navigation stack to be maintained in memory. For this to be implemented the full application would have to be modified to allow for the new bottom bar within the user interface, as well as the new style of navigation requiring more detailed functional modifications also. Additionally, because of the limited space on the bottom bar, the settings screen had to be moved within the user profile; this could cause confusion for some users. The use of this design would have to be seriously considered before implementation, due to the significant increase in memory consumption.

The second design would have less of an impact on the full system. However, it would also provide little benefits. This design would solve the issue that was raised by the user, and the overhead to complete this should not be significant.

Due to the project schedule and the time it would take to reimplement a home screen, this modification has not been applied at this time. Additionally deciding which solution to use out of the two suggested would require feedback from users. This will be a good area to focus on with further development of the application, which will be discussed later in Section **??**.

From these different test methods, a thorough examination of the system has been completed; next an evaluation of the system will be conducted to compare the system against the original goals set out.

CHAPTER 9

Evaluation

Design, Implementation and testing of the system have now taken place, therefore evaluation of the system can commence, comparing the system against the what was originally planned. This will include requirements; project management and ethical; social, legal and professional issues and finally an evaluation from the developer will be included.

9.1 Functional Requirements

Below the updated requirements for the project have been shown with any unnecessary or outdated requirements removed, based on the reasoning previously discussed in Section 4.2.1. Any requirement that the project has successfully met will be coloured in green to represent that it has been met, all those which have not been met will be coloured in red. A further detailed evaluation of each specific requirement will then be discussed in Table 9.1

F1: *The system should allow a user to enter details about a workout, such as exercises completed, number of sets, number of reps, weight lifted and workout review*
F2: *The system should store data entered by the user in a persistent form*
F3: *The system should allow user to select a preferred fitness object, e.g. weight loss, muscle gain*
F5: *Integrate with apple health to retrieve user information as well as writing workout to data*
F7: *A flow should be created on Octoblu to handle sentiment analysis of a user's message review of workout*
F8: *The system should have an option to send information to the previously specified Octoblu flow*
F9: *Create a set of messages to send depending on result of sentiment analysis*
F10: *The system should employ some basic inference techniques to suggest workouts for a user*
F12: *The system should recommend a user's daily calorie intake, corresponding to the user's lifestyle, which should be backed by scientific research*
F13: *The system should employ some inference techniques to suggest the specific parameters of a workout, such as number of reps, weight to be lifted and number of sets to complete for the exercise*
F14: *The system should feature a number of inbuilt workout routines, to allow the system to gather metrics about a user*
F15: *There should exist a number of inbuilt workouts tailored for each of the specific fitness objectives, which are backed by scientific research*
F16: *Employ some learning technique to stop the user excessively using a specific workout routine*
F17: *The base calorie intake set out in 'F12' should be modified to be specific to the user's fitness goal*
F18: *The system should adhere to the characteristics of a Modal View Controller (MVC) system development*
F19: *There should be a View Created for each function of the system, which should be linked to an appropriate model*

F20: *The calorie intake set out in 'F12' should be used to calculate the recommended percentage consumption of macronutrients*
F21: *Maintain a constant styling throughout the UI of the application*

Requirement	Comment	Result
F1	The system allows the user to create workouts and select the exercises to complete, with the ability to also enter the weight lifted and a number of reps for each set. After the workout is complete, they can also enter a personal text and star review of the workout	Pass
F2	All data entered by the user is stored in a persistent manor within the realm database.	Pass
F3	On sign-up the user selects their desired fitness goal, this can also be updated within the user profile section of the application.	Pass
F5	Unfortunately, this requirement was not met due to delays in other areas of the project. Despite this requirement being unmet, no core functionality was lost as all of the apple health benefits were additions for integration within the apple platform, as well as registration speed up. All the fields this would add already exist within the apps platform and sign-up	Fail
F7	A flow was created to handle the workout review sentiment analysis, within Octoblu.	Pass
F8	There is an option both on sign up and within the settings that allow the user to decide whether or not to enable the workout feedback using Octoblu.	Pass
F9	Two messages were created for the different values of sentiment established based on both positive and negative workout reviews. This could be increased further to give a wider range of feedback responses, but in its current state the requirement is still met	Pass
F10	The system does not suggest workouts for a user, hence it does not meet the initial requirements. This feature is not completely lost though as the workout routines in-built are tailored to specific fitness goals and labelled appropriately. The system does therefore still gives user-focused workouts, just not in the way initially intended.	Fail
F12	The system can recommend a user's daily calorie intake, based on their personal information, by using formulae from scientific and medical research.	Pass
F13	Once a baseline for an exercise is established for a given user, the system then infers an appropriate rep count and weight for each exercise in a workout.	Pass
F14	There are a number of workout routines inbuilt to the application; any of these can be used to gather a baseline for future exercise generation.	Pass
F15	The inbuilt workouts are focused on different fitness goals and have been constructed based on established methods.	Pass
F16	The system does not limit successive repeated workouts, therefore this requirement was unmet. However, this feature wasn't implemented through choice, as this feature would limit the user's actions which could be frustrating for the user. To resolve this issue, a warning message could be used instead. This should be considered in future development.	Fail
F17	The suggested calorie intake is adapted for the user's specific fitness goal, based on scientific and medical journals.	Pass
F18	Throughout the system, the model view controller model is followed.	Pass

F19	All Sections of the system have corresponding views and controllers	Pass
F20	Macronutrients are calculated for each user, based on the user's fitness goal and calorie intake. Again this backed by scientific and medical research.	Pass
F21	The application has constant styling throughout using a set colour scheme and Apple design principles.	Pass

Table 9.1: Detailed evaluation of the functional requirements

9.2 Non-functional Requirements

As before with functional requirements all of the requirements which were met have been coloured in green and those which were unmet are coloured red. Again, a further detailed evaluation of each specific requirement will then be discussed in Table 9.2

NF1: *The system should have a responsive UI, such that the user is never unaware of system processes*
NF2: *The system should have a simple UI so it can be used by users with a range of technical abilities*
NF3: *The system should be modular to allow for further or improved functionality*
NF4: *The system should have a low time solution to recording exercises, reps, sets and weight*
NF5: *The developer should have a greater knowledge of iOS development*
NF6: *The application should limit hardware resources used, to a level that is in-line with other applications of this type*

Requirement	**Comment**	**Result**
NF1	Animations are added to show the transitioning of screens, also there is a loading spinner within the workout screen to show when it is generating the workout.	Pass
NF2	The user interface is minimal and uncluttered to promote ease of use; additionally, no feedback was gathered from testing to suggest that there were any difficulties with this.	Pass
NF3	The code for the application is separated within components, and each of these components is broken down further into separate units; this allows individual units to be replaced without a knock on effect to the rest of the system.	Pass
NF4	During a workout weight and repetition, values can be altered by clicking on the value and inputting the new desired value.	Pass
NF5	The developer now has a larger knowledge regarding iOS development and the surrounding technologies	Pass
NF6	As shown in testing the system is extremely efficient, requiring minimal use of the the hardware resources.	Pass

Table 9.2: Detailed evaluation of the non-functional requirements

9.3 Ethical, Social, Legal and Professional Issues Review

The project will now be evaluated against the ethical, social, legal and professional issues which were initially established for the development in Chapter 3. Each of these issues will be discussed individually in their relevant subsections.

9.3.1 Ethical

The main ethical consideration regarding the storage of personal information was met as all user information is stored on their personal device. The user also has the decision to share their information to

enable the motivation feature. Even if this feature is selected the shared information is not stored by Octoblu and all transmissions are protected to avoid the information being accessed by outside sources during transmissions. Additionally the parameters sent to this service have been minimised to only those required, which are workout review and phone number. These measures ensure that user details are not shared with external sources without user consent.

9.3.2 Social

The social issues which relate to this application have been considered throughout development; the application is un-intrusive and is aimed at providing an aid to increase health, which socially is deemed to be good. The most import point to make is that usage of the application and specific features are user choice, hence they take the social responsibility of using the application.

9.3.3 Legal

To ensure the project did not breach and copyright or patent laws, any external material which was chosen for use was thoroughly examined to confirm that it was not covered by copyright or patented.

The application also meets all requirements of the "Data Protection Act 1998" [42]. This is because: only personal details that are required for processing are used; the details are only kept as long as necessary, and the user has the option to delete all stored data at any time; the data is only stored on the user's device in a secure form, preventing unlawful access; the user has to approve any sharing of information.

As the schedule changed and the focus of app was not aimed at production by the deadline, correct terms and conditions were not placed within the project, therefore, liability was not covered for the project. Going forward consulting legal aid to help write a correct terms of use document for the project would be beneficial; this would guarantee that the app would not be held liable for any injury incurred while using the application.

9.3.4 Professional

Throughout the development the British Computing Society code of conduct [89] has been followed, to ensure that the application meets the standards expected.

9.4 Project Management

This section will breakdown the management decisions chosen for the project and analyse how effective the employed solution was, in addition any issues will also be covered.

9.5 Design Approach

The bottom-up approach chosen matched the development perfectly, as planned; not only did it allow the system to grow in a natural way, it prevented any issues being caused from the adapting requirements. This was because no component was ever left redundant due to a change in schedule or requirements, as all the components implemented already had a function in the system.

9.5.1 Software Development Methodology

This hybrid development style has allowed the developer to easily adapt to the challenges that have arisen throughout the project; the change in iOS development course is a prime example of this. The swapping of courses highly benefited the developer and provided an overall benefit to the progress of the development as it gave the developer sufficient knowledge to begin creating a custom application; knowledge which was not supplied by the first course taken. It was also important to maintain small plan driven goals to focus the developer's attention on the time constraints imposed on the project. These small goals allowed the different aspects of the project to stay on schedule, whilst maintaining the features that were expected to be produced.

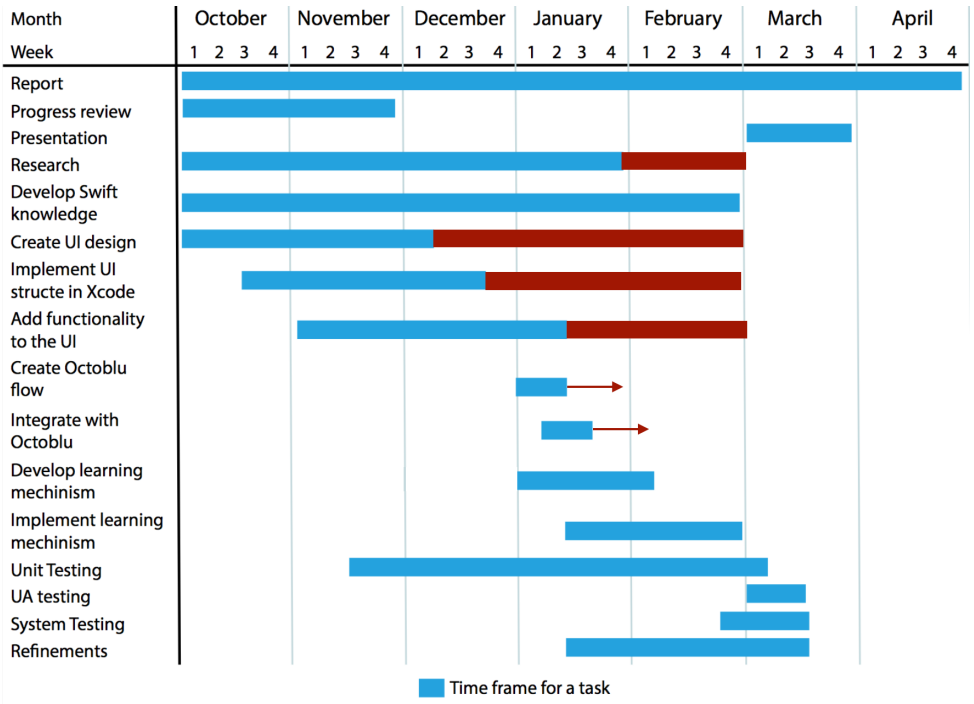


Figure 9.1: Representation of the actual schedule of the project, showing alterations to the updated schedule

9.5.2 Project Schedule

The final schedule for the project strayed from the updated schedule set out in Figure 5.2, as expected. This was due to various reasons. The two main reasons were issues with iOS learning and developer oversight. As stated earlier in the project certain aspects of the development where unresearched when creating the plan. As more knowledge was gained in these areas it became apparent that some of the planned features were not feasible within the time-frame allocated; an example of this was regarding the workout inference. At first the plan was to generate a full workout including exercise schedule; it was later realised that this would not be possible within the time frame. Another area that caused delays was the issue with the developers learning; the change of course caused set backs and delays in other areas of the project. The biggest effect this had was in the reduction in available time to produce an inference mechanism for workouts; this was a necessary sacrifice to ensure that the remaining features of the application could still be implemented, as well as some form of inference. The final schedule for the project has been summarised in Figure 9.1, with the delays and extensions highlighted.

9.5.3 Tools

The tools sourced for this project greatly enhanced the product that was possible to develop. The tool that was most useful was Xcode although this was necessary to build the application; once the developer was familiar with the application this surpassed expectations and gave many additional benefits that greatly eased development. Cocoapods also proved to be highly useful in the sourcing of packages, as well as maintaining them as originally planned; this allowed further time to be focused on the development. Trello was a tool that was not originally planned, but the use of this when required was invaluable to the development as all the project tasks and bugs would have been increasingly difficult to track without this, which would have reduced the time available to complete the tasks and fix any issues.

9.5.4 Management Approach

The communication methods utilized enabled fluid conversation between the developer and supervisor, enabling issues to be resolved quicker than they would have been if only scheduled meetings were used. The scheduled meetings were useful in monitoring progress of the project and ensuring that the plan-

driven goals were met. Hence the management approach proved to be effective in helping the project establish an achievable set of goals within the strict time-frame.

9.5.5 Risk Management

The plans and guidelines set out for the project were invaluable. Without these redundancies in place the project would not be at the stage it is now. An issue occurred when committing a merge between the development branch and the master branch of the project resulting in a crash of Xcode. Upon relaunch it was realised that the failed merge had not only corrupted the main branch, but the development branch could no longer be accessed due to unresolved merges. This left the developer without a project weeks before the presentation. Thankfully the original guidelines had been followed and a Time machine back-up of the folder existed from the previous day; the folder was then reverted back to this version, which resolved the merge issues and allowed the project to progress with a minor one day delay. This delay was insignificant compared to the loss of the whole project with only 2 weeks remaining until the presentation date and shows the benefit of the initial risk management plan.

9.6 Authors Reflection

Why should this contribution be considered relevant and important to the subject of your degree?

This project brought together aspects from across the degree and enhanced the developer's knowledge in these further. The core development and programming modules provided a level grounding to learn a new language and development style. Aspects form modules undertaken in the final year were also utilised, combining the practices from Social Informatics into developing a user-centered interface. Emerging technologies were also utilised, based on the preliminary material taught in Advanced Databases. This was researched further to allow a NoSQL database to be incorporated effectively. Principles of Advanced Computer Architecture were also followed, to ensure a highly efficient and performant system was produced. Focusing less on actual details, the ethical and social aspects behind the completing of the application are also very relevant. As mentioned in the background to this project, the advancements in technology are promoting the population to do less. Technology has always been driven to make lives easier, but practices need to be put in place so that technology does not create an obese population. Therefore the use of technology as a fitness aid is an area which should be pushed further from within the Computer Science field, to help maintain a healthier society. Finally, the project represents the change in direction for applications, as mobile devices are now the predominantly used technological device for most users, pursuing development in this area is fitting of the modern usage of applications.

How can others make use of the work in this project?

The system could be used a baseline application, to support further research into how technology can be utilised to aid the estimation of workout metrics. This estimation could be done using more advanced machine learning techniques. Additionally, the work conducted could be modified to produce more accurate recommendations, as recommendation based systems are constantly being modified, there will always exist areas that can be explored to give more precise results. The exploration of these could be carried out using this application thanks to the degree of modularity applied especially in the workout generation area. The obvious area others could make use of the application is through usage, the system is at a level where it could be used by others and provide a suitable degree of guidance in becoming active, users should be made aware of the risks involved with strength training before usage.

What are the limitations of this project?

The inference mechanism used is still rudimentary and could be expanded upon to provide a more thorough way of estimating metrics for the user; some methods for this will be covered in Section **??**. Additionally, the application could be more informative to new users, supplying further information on using the gym; this is suggested through the use of a guide, and workout execution videos within exercises. The final area there are limitations, is regarding the number of exercises included within the application. The information for each specific exercise has to be sourced and manually entered into a data loading script, the number of exercises it was possible to complete during the project's timeframe was limited;

this was an arduous task for the developer, which provided little benefit once a suitable exercise catalogue was produced.

Why should this project be considered an achievement?

The project brings together three different academic disciplines to produce a system that has functionality never before implemented into an application, all within a limited timeframe. This required a full-scale development including a significant research section, to tie the three disciplines together successfully. The project can be thought of as a success through the use of quantitative measures, achieving almost all of the requirements for the project; the initial overarching aims for the project were also achieved.

The level of advancement of the developer's knowledge shows the real success of the project; with no prior experience with app development before the project, to the level required to produce this solution, requires the project be thorough and challenging. The confidence gained through the hours of work dedicated to this project and the challenges faced along the way have provided great insight into a realistic application development. This has proved essential in the options it has provided the developer with, leading to the developers successful hiring.

A summary of the development will be covered next, based on the evaluations made in this chapter; the further work to be carried out on this system will also be included in the summary.

CHAPTER 10

Conclusion

This chapter will outline the overall project, highlighting the key areas of accomplishment. This outline will be followed by the further work to be completed for the project, alongside explanations of why this work is beneficial to the product.

10.1 Project Summery

The final product meets almost all of the original requirements, providing the user with an application that: gives the user a fitness goal tailored workout routine; updates workouts based on previously completed exercises; allows experienced users to create their own personal workouts; provides feedback based on users workout review; recommends calorie intake and macronutrients based on specific fitness goal; provides users the ability to track their past workouts; gives a detailed catalogue of exercises including image and text descriptions. All of these are available through the iOS application and the building of this significantly increased the developer's knowledge of the language and development tools used. There still exists a broad spectrum of enhancements and additions to the product, which will be discussed in the next section. The lack of these enhancements and additions does not subtract from the current products achievements, especially given the time constraints enforced upon the project.

10.2 Future Work

For the ongoing development, both breadth and depth of the application will be increased. This increase is because the implemented features require more detail and enhancement, as well as the addition of new features which would further enhance the products ability to represent a personal trainer. There is a number of implementations that are planned, however, not all of these can be discussed in detail, therefore some of the more important issues are discussed below.

10.2.1 Remaining Work

The most apparent areas for further work to be completed link to the requirements that were not met within the time-frame of the project; these were discussed previously in Sections 9.1, 9.2 and 9.3. Additionally, there are also the issues discovered from user testing in Section 8.5, that could not be resolved within the projects time-frame. These incomplete requirements would be the work that would be completed first, before progressing onto additional features.

10.2.2 Health Kit Integration

This area also spans from unmet requirements; there are areas linking to this which could be extended further than originally set out at the projects start. The primary benefit would be easier sign-up as base

metrics about the user could be gathered from details already entered into Apple Health leaving less sign up options and would also allow the fields within the app to stay up to date. By frequently checking for updated user data from Apple Health, more accurate nutritional estimations would be given as the estimations use weight as a factor within the calculation and up to date information is necessary. The other benefit to Apple Health is that the past workouts section could gather all previous workout data to give a more informed breakdown of how active the user has been.

10.2.3 Apple Watch Integration

The main benefit of using the Apple Watch is the ability to utilise the additional sensors within the watch itself. The first of these that could be used is the heart rate sensor; a user's heart rate can be used to quantify how hard their body is working, hence if their heart rate is weak while completing an exercise, the weight is too low as they do not have to work enough to lift it. On the other hand, if their heart rate is significantly above the active level, it is likely that this weight is too heavy and that the user is overly straining themselves to lift the set weight. Both of these principles could be employed to ensure that the exercise is being conducted safely and also effectively.

Additionally, using the accelerometer within the Apple Watch would allow a different estimation to 1-RM to be used, based on methods discussed in Section 2.2.5. The velocity at which an exercise is completed along with the weight being lifted can give an accurate approximation to 1-RM [74]. This method could be used in addition to the formula currently employed to provide Watch users more precise estimations of 1-RM. If a repetition of the exercise is completed quickly, this would suggest it is easy for the user to complete and they could lift a heavier weight, counter to this if the user lifts the weight slowly it is likely that they are struggling and the weight may be too heavy. These are some basic inferences that could further aid a more accurate weight prediction for the user.

The accelerometer could also be used to measure the form of the exercise being completed. For instance, if the user is shaking a considerable amount during the exercise and they are not stable, this would suggest the current weight is too difficult for the user. Using this information the weight could be lowered accordingly. Having this ability to detect from could help prevent injury from incorrect completion of an exercise.

10.2.4 Beginners Guide

Despite the efforts made to make the application accessible for users who have never undertaken strength training before, it became evident at the presentation that more could be done to make the application more informative for newcomers. A two-phase approach will be used for this. The first will involve the addition of a welcome guide and the second will be the expansion of the exercise information to include video demonstrations.

The guide will be based around introducing the user to using a gym, running through different types of exercises and the equipment associated. This guide aims to provide familiarity with the gym environment. Next a brief run through of how workouts are structured to explain how primary and secondary muscles are involved in exercises; this will reassure the user that some exercise will cause aches to other muscles also and this is to be expected. Finally, an explanation of good form for exercises will be explained. This explanation will cover how exercises should be performed, and that a steady pace with correct form is more important than lifting heavy weights with bad form when improving their physique. This explanation will also state that completing an exercise too fast is not beneficial.

The videos will aim to provide more support in explaining and demonstrating how an exercise should be completed; ideally these should have a spoken description alongside the actual physical video of the exercise being completed. To further enhance the videos function multiple angles of completion could be shown to ensure the user fully understands how to complete the shown exercise.

Hopefully this would be enough to make the app properly accessible to newcomers; user feedback should be gathered to discover if these additions are beneficial. Additionally, any areas not already covered by the application could be added based on the gathered feedback. To further aid these features an introductory session from a personal trainer could be taken to find the information they give and this could also be incorporated. As the application is designed for any fitness level these additions should be skip-able for the users who already have experience, as this will be unnecessary as well as time-consuming and intrusive.

10.2.5 Searching

Frequently throughout the app there are lists of items, be that workouts or exercises. Currently, there is no way to filter through the lists or search for a specific exercise quickly. The addition of a search box at the top of any table would allow quick searching of the table allowing more experienced users to find what they are looking for more quickly. To begin with, this could be a simple key term search checking if any if the item strings contain the searched string. This search could be further enhanced to give a more optimal solution that accounts for slight errors in the terms entered. This feature would become even more useful as the number of exercises and workouts grow, as finding a single exercise in a list of hundreds, even when sorted would prove to be difficult and time-consuming when compared with text searching.

10.3 Commercialising the Application

As this project was not aimed at developing a finalised product by the deadline, marketing of the product was not included within the bulk of the report. As this will eventually be an objective of the project, possible marketing strategies for the app will follow alongside reasoning to the benefits and downfalls of each approach. This discussion will focus solely on different pricing strategies, and will not include other marketing angles, such as advertising.

Paid App

A paid application is the most simple model and has been established for some time. The premise is that the user pays a one-off fee for the application and then has full access to that version of the application forever. Commonly with apps, all future updates are also given for free with this model, although that is not essential. The benefit of this model is that revenue is made from every user; however, it requires a steady stream of new users to continue to be profitable. Additionally, users may not purchase the application because of the initial cost, this is likely because there are free alternatives and the user has not yet had a chance to experience the app before payment.

Advertisements

This model has become dominant with mobile applications, as it allows users to use applications without any financial outlay. The premise is that adverts are added within the application; if the adverts are selected a small payment is paid to the producer of the application. The obvious benefit of this model is that the user can try the application without any fee. However, this can cause an issue where none of the users select the adverts, and therefore no revenue is made. Equally this model requires a high continuous user count to make a profit, but it does not rely on a constant stream of new users. To avoid the unselected advert issue some applications force the user to watch advertisements at set intervals, to ensure that revenue can still be made.

Subscription

This model requires the user to pay a regular fee to continue to use the application and is commonly referred to as SaaS (Software as a Service). The fee frequently is charged monthly, with some applications giving discounts to users who subscribe to a longer period of time, where the discount increases the longer the subscription period. Also, a set amount of time can be given to the user free of charge. This time is assigned to let the user test the application before they agree to a subscription. The obvious benefit of this is the ongoing revenue. However, many users dislike this style of marketing model, as they dislike the regular payments associated, especially when running costs are low. Another point worth noting is that the monthly subscription fee is expected to be significantly less than a one-off payment would be. If the users were not encouraged to keep using the app this would result in a substantial loss in revenue compared to the paid model; therefore a guarantee of regular paying users is necessary.

Freemium Model

As with advertisement based approach, the Freemium model also rose with the rise of mobile applications. It works on the premise that the application is given to the user free of charge, but with limited function-

ality. The user can then purchase the additional functionality at an extra cost. The benefit of this model is that the users can try the application for free. If they like the app they are then encouraged to buy the additional functionality. The obvious flaw with this plan is that there is no revenue guaranteed, as all users could simply use the application in its limited state. Some manufacturers overcome this by limiting the application greatly to ensure it does not function optimally without the paid extras. Although this reduces the risk of users only using the free functionality, it may prevent the user from fully experiencing the application, and they therefore not feel inclined to purchase the extras.

Pocket PT's Marketing Strategy

There is no guaranteed way to make an application profitable, however having the wrong strategy can ensure that it is not. All of the models above have positive and negative aspects, and these are also dependent on how the application will be used. Due to the nature of the application, it is likely that users will use the app regularly, but there may not be a high level of frequent new users. As this is the case, a model that focuses on frequent users would be better suited than a one-off payment model as the new users are not guaranteed to bring enough maintainable income. The subscription-based approach will also be avoided as it is likely to deter more customers. Therefore both Freemium and advertising are left. Both of these models fit well, as the Freemium model allows users to purchase additional features, which for this application could be more workouts or a greater exercise catalogue. Both of these would target users who have used the application for a long time and seek new variety. Additionally, the advertisement model would allow revenue from continued use, which will be the main usage style of the application.

A hybrid approach between these will be employed where the initial product will come with both adverts and slightly limited features. The idea behind this is an individual could easily use the app without the purchase of the add-ons and still generate revenue without financial outlay, while users who require more functionality can buy it, again increasing app revenue. The removal of adverts could even be an upgrade for those who find the ads intrusive; this would offset the loss of income from removing adds. There are still areas where this strategy could fail, but it is the most appropriate fit for the applications current level. This is the best fit because it encourages users to try the application for free, while maintaining revenue at all stages either via advertising or through paid features.

Bibliography

[1] Apple. http://www.apple.com/uk/, 2016.

[2] AMAC. Level 2 exercise and fitness knowledge (gym instructor / exercise to music instructor). http://amactraining.co.uk/resources/handy-information/free-learning-material/level-2-exercise-and-fitness-knowledge-index/.

[3] Apple. Fitbit: App store page. https://itunes.apple.com/gb/app/fitbit/id462638897?mt=8.

[4] Apple. Start developing ios apps (swift). https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html.

[5] Apple. ios. http://www.apple.com/uk/ios, 2016.

[6] Apple. Messages. https://support.apple.com/explore/messages, 2016.

[7] Apple. Myfitnesspal. https://itunes.apple.com/gb/app/calorie-counter-diet-tracker/id341232718?mt=8, 2016.

[8] Apple. Swift. https://developer.apple.com/swift/, 2016.

[9] Apple. Watchos. http://www.apple.com/uk/watchos/, 2016.

[10] Apple. Xcode. https://developer.apple.com/xcode/ide/, 2016.

[11] Apple. icloud. https://www.apple.com/uk/icloud/, 2017.

[12] Apple. Keynote. https://www.apple.com/uk/keynote/, 2017.

[13] Apple. Quicktime. https://support.apple.com/kb/DL923?viewlocale=en_US\&locale=en_US, 2017.

[14] Apple. What is core data. https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html?utm_source=iosstash.io, 2017.

[15] R. W. Artur Direito, Yannan Jiang and R. Maddison. Smartphone apps to improve fitness and increase physical activity among young people: protocol of the apps for improving fitness (aimfit) randomized controlled trial. *BMC Public Health*, 15(635), July 2015.

[16] K. Beck. Principles behind the agile manifesto, 2001.

[17] D. T. O. T. Ben Asher, Dimitris Koutsogiorgas. Cocoapods. https://cocoapods.org/, 2017.

[18] Bodybuilding.com. Workouts. https://www.bodybuilding.com/category/workouts.

[19] Bodybuilding.com. Macronutrient intake. January 2017.

[20] M. Brzycki. Strength testingpredicting a one-rep max from reps-to-fatigue. *Journal of Physical Education, Recreation & Dance*, 64(1):88–90, 1993.

[21] G. E. Campos, T. J. Luecke, H. K. Wendeln, K. Toma, F. C. Hagerman, T. F. Murray, K. E. Ragg, N. A. Ratamess, W. J. Kraemer, and R. S. Staron. Muscular adaptations in response to three different resistance-training regimens: specificity of repetition maximum training zones. *European Journal of Applied Physiology*, 88(1):50–60, 2002.

[22] D. G. Candow and D. G. Burke. Effect of short-term equal-volume resistance training with different workout frequency on muscle mass and strength in untrained men and women. *The Journal of Strength & Conditioning Research*, 21(1):204–207, 2007.

[23] E. CB, S. JF, F. HA, and et al. Effects of dietary composition on energy expenditure during weight-loss maintenance. *JAMA*, 307(24):2627–2634, 2012.

[24] Citrix. Citrix. `https://www.citrix.co.uk/`, 2017.

[25] G. Coleman. Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, 81(5):772–784, May 2008.

[26] R. Cox. How to use tech to improve your fitness. Online, September 2013.

[27] T. L. DELORME. Restoration of muscle power by heavy-resistance exercises. *The Journal of Bone & Joint Surgery:*, 27, October 1945.

[28] t. A. o. N. Dietitians of Canada, Dietetics, and the American College of Sports Medicine;. Nutrition and athletic performance. December 2016.

[29] A. Dix. Human computer interaction. *Encyclopedia of database systems*, pages 1327–1331, 2006.

[30] DRopbox. Dropbox. `https://www.dropbox.com/`, 2016.

[31] B. Egan. Protein intake for athletes and active adults: Current concepts and controversies. *Nutrition Bulletin*, 41(3):202 – 213, 2016.

[32] N. H. S. England. Diets weighed up. february 2009.

[33] P. H. England. Adult obesity. `http://www.noo.org.uk/NOO_about_obesity/adult_obesity`, 2016.

[34] P. H. England. Adult obesity international comparisons data factsheet, September 2016.

[35] P. H. England. Child obeisity: Uk prevalence, 2016.

[36] P. H. England. Patterns and trends in adult obesity. Public Health England, April 2016.

[37] P. J. F. Eric R Helms, Alan A Aragon. Evidence-based recommendations for natural bodybuilding contest preparation: nutrition and supplementation. *Journal of the International Society of Sports Nutrition*, 11(20), May 2014.

[38] Fitbit. Fitbit. `https://www.fitbit.com/`, 2016.

[39] F. Fitness. Full fitness: Exercise workout trainer. `https://itunes.apple.com/gb/app/full-fitness-exercise-workout/id536049508?mt=8`, 2016.

[40] S. J. Fleck and W. Kraemer. *Designing Resistance Training Programs, 4E*. Human Kinetics, 2014.

[41] Google. Creating a sentiment analysis model — prediction api documentation — google cloud platform. `https://cloud.google.com/prediction/docs/sentiment_analysis`.

[42] B. Government. Data protection act 1998. Legislation, 2005.

[43] IBM. Natural language understanding. `https://console.ng.bluemix.net/catalog/services/natural-language-understanding`.

[44] H. Inc. 7 minute workout. `http://7minworkoutapp.net/`.

[45] H. Inc. 7 minute workout app store page. https://itunes.apple.com/gb/app/7-minute-workout-challenge/.

[46] Jay. Citrix. https://patchthecode.github.io/, 2017.

[47] A. Karkach and R. AA. Modern approaches to the analysis and forecasting of population health with the help of mathematical models. *Science and Information Technology*, (1), 2014.

[48] M. C. Kelley. The impact of fitness technology on health outcomes, 2014.

[49] G. J. Kim. *Human–Computer Interaction*. CRC Press.

[50] K. M. Knutzen, L. R. Brilla, and D. Caine. Validity of 1rm prediction equations for older adults. *The Journal of Strength & Conditioning Research*, 13(3):242–246, 1999.

[51] R. Koch. Texshop. http://pages.uoregon.edu/koch/texshop/, 2017.

[52] L. L. E. W. J. Lambert, Charles P. Frank. Macronutrient considerations for the sport of bodybuilding. *Sports Medicine*, 34(5):317–327, 2004.

[53] LeisureDB. 2016 state of the uk fitness industry report, May 2016.

[54] Life-Fitness. Use technology to improve your health. Online, April 2015.

[55] T. B. T. Limited. Citrix. http://www.goodnotesapp.com/, 2017.

[56] T. Liscensing. The changing ways were watching the box. http://www.tvlicensing.co.uk/ss/Satellite?blobcol\=urldata\&blobheadername1\=content-type\&blobheadervalue1\=application%2Fpdf\&blobkey\=id\&blobtable\=MungoBlobs\&blobwhere\=1370006220727\&ssbinary\=true, 2011.

[57] R. Marchese and A. Hill. *The Essential Guide to Fitness: For the Fitness Instructor*. Pearson Australia, 2011.

[58] W. Materko, C. E. B. Neves, and E. L. Santos. Prediction model of a maximal repetition (1rm) based on male and female anthropometrical characteristics. *Revista Brasileira de Medicina do Esporte*, 13(1):27–32, 2007.

[59] MensFitness. Workout routines. http://www.mensfitness.com/training/workout-routines.

[60] MensHealth. Workout. http://www.menshealth.co.uk/workout.

[61] Microsoft.

[62] Microsoft. Text analytics api preview. https://azure.microsoft.com/en-gb/services/cognitive-services/text-analytics/.

[63] H. Miranda, S. J. Fleck, R. Simão, A. C. Barreto, E. H. Dantas, and J. Novaes. Effect of two different rest period lengths on the number of repetitions performed during resistance training. *The Journal of Strength & Conditioning Research*, 21(4):1032–1036, 2007.

[64] L. M.L., D. V.O., V. J.M., L. J.R.P., R. V.M., B. J.P., and F. J. Fernandes. Precision of 1-rm prediction equations in non-competitive subjects performing strength training. *Motricidade , Vol 6, Iss 3, Pp 31-37 (2010)*, (3):31, 2010.

[65] MyFitnessPal. Myfitnesspal. https://www.myfitnesspal.com/, 2016.

[66] Netflix. Netflix. https://www.netflix.com/gb/, 2016.

[67] NHS. Health and fitness trackers. Online, May 2014.

[68] NHS. Obesity. http://www.nhs.uk/conditions/Obesity/Pages/Introduction.aspx, June 2016.

[69] Octoblu. Octoblu. https://www.octoblu.com/.

[70] OFCOM. The uk is now a smartphone society. https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2015/cmr-uk-2015, August 2015.

[71] E. M. Paul Branco. Cvcalendar. https://github.com/cvcal, March 2017.

[72] R. Percival. The complete ios 10 developer course. https://www.udemy.com/complete-ios-10-developer-course/.

[73] . L. K. P. Phil Block, M.S. Tailoring nutrient intake to exercise goals.

[74] P. Picerno, D. Iannetta, S. Comotto, M. Donati, F. Pecoraro, M. Zok, G. Tollis, M. Figura, C. Varalda, D. Di Muzio, F. Patrizio, and M. F. Piacentini. 1rm prediction: a novel methodology based on the force–velocity and load–velocity relationships. *European Journal of Applied Physiology*, 116(10):2035–2043, 2016.

[75] D. C. G. Pierre-Marc Airoldi. Charts. https://cocoapods.org/pods/Charts, 2017.

[76] A. practical approach to strength training. *Brzycki, Matt.* McGraw-Hill, 1998.

[77] A. Qidwae. Technology negatively affecting our health, study shows. *Journal Sentinal*, July 2012.

[78] Realm. Realm mobile database. https://realm.io/products/realm-mobile-database/, 2017.

[79] L. Roberts. Are personal trainers worth the price? *The Telegraph*, March 2011.

[80] F. M. Sacks, G. A. Bray, V. J. Carey, S. R. Smith, D. H. Ryan, S. D. Anton, K. McManus, C. M. Champagne, L. M. Bishop, N. Laranjo, M. S. Leboff, J. C. Rood, L. de Jonge, F. L. Greenway, C. M. Loria, E. Obarzanek, and D. A. Williamson. Comparison of weight-loss diets with different compositions of fat, protein, and carbohydrates. *New England Journal of Medicine*, 360(9):859–873, 2009. PMID: 19246357.

[81] Serif. Affinity designer. https://affinity.serif.com/en-gb/designer/, 2017.

[82] Serif. Affinity photo. https://affinity.serif.com/en-gb/photo/, 2017.

[83] J. Shakeshaft. The ultimate guide to gym lingo, July 2011.

[84] B. P. Shirley Gerrior, WenYen Jaun. An easy approach to calculating estimated energy requirements. *Preventing Chronic Disease*, 3(4), October 2006.

[85] P. B. Shirley Gerrior, WenYen Juan. Metaphors and the user interface. http://www.katalinszabo.com/metaphor.htm, 1995.

[86] T. Sjsten. Calculate your one rep max (1rm). https://www.athlegan.com/calculate-1rm/, May 2016.

[87] B. Skwarecki. How much does genetics really affect your fitness?, October 2015.

[88] Skype. Skype. https://www.skype.com/en/, 2016.

[89] B. C. Society. Bcs, the chartered institute for it trustee board regulations - schedule 3 code of conduct for bcs members, June 2015.

[90] S. Text. Sublime text. https://www.sublimetext.com/, 2017.

[91] TheHutGroup. The hut group. https://www.thehutgroup.com, 2017.

[92] TheySay. Preceive. urlhttp://www.theysay.io/product/preceive/.

[93] P. M. C. M. N. G. D. B. Thomas P Wycherley, Lisa J Moran. Effects of energy-restricted high-protein, low-fat compared with standard-protein, low-fat diets: a meta-analysis of randomized controlled trials. *The American Journal of Clinical Nutrition*, October 2012.

[94] Trello. Trello. https://trello.com/, 2017.

[95] A. Tremblay, J.-A. Simoneau, and C. Bouchard. Impact of exercise intensity on body fatness and skeletal muscle metabolism. *Metabolism*, 43(7):814–818, 1994.

[96] S. University. Cs139p iphone application development. `http://web.stanford.edu/class/cs193p/cgi-bin/drupal/`.

[97] C. Williams and S. Ratel. *Human Muscle Fatigue*. Taylor & Francis, 2009.

[98] L. Williams. A survey of plan-driven development methodologies, 2014.

[99] O. C. Witard, S. L. Wardle, L. S. Macnaughton, A. B. Hodgson, and K. D. Tipton. Protein considerations for optimising skeletal muscle mass in healthy young and older adults. *Nutrients*, 8(4), 2016.

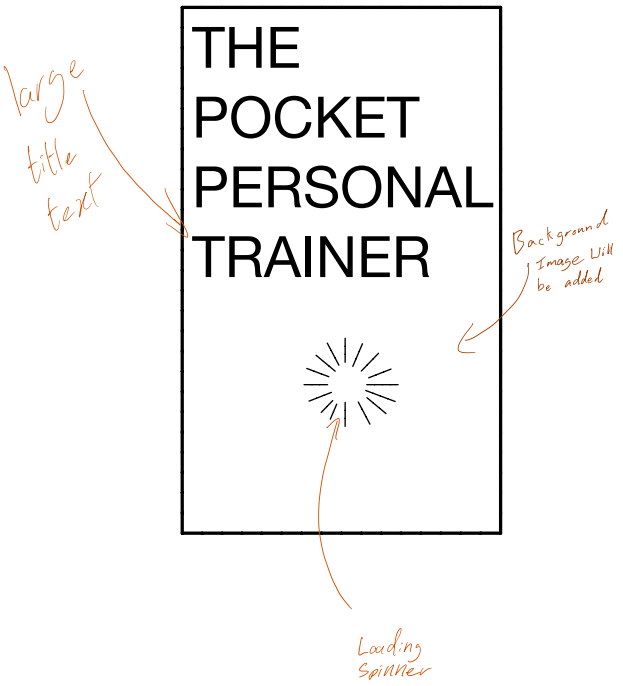
[100] L. Z and H. D. Overeating and overweight: Extra calories increase fat mass while protein increases lean mass. *JAMA*, 307(1):86–87, 2012.

[101] D. Zamora. The absolute beginner's guide to exercise. `http://www.webmd.com/fitness-exercise/guide/fitness-beginners-guide#3`.

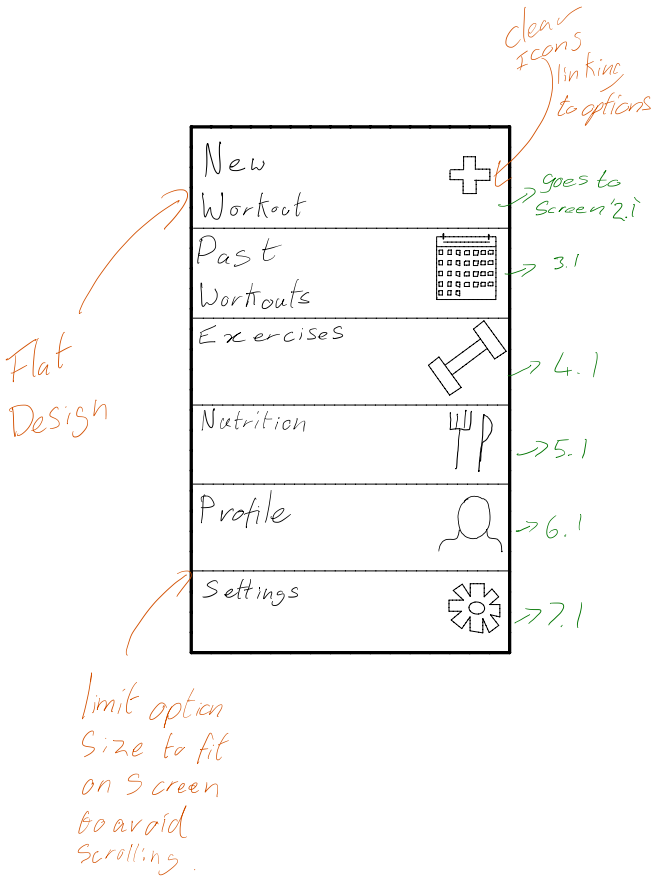
APPENDIX A

User Interface Design Mockups

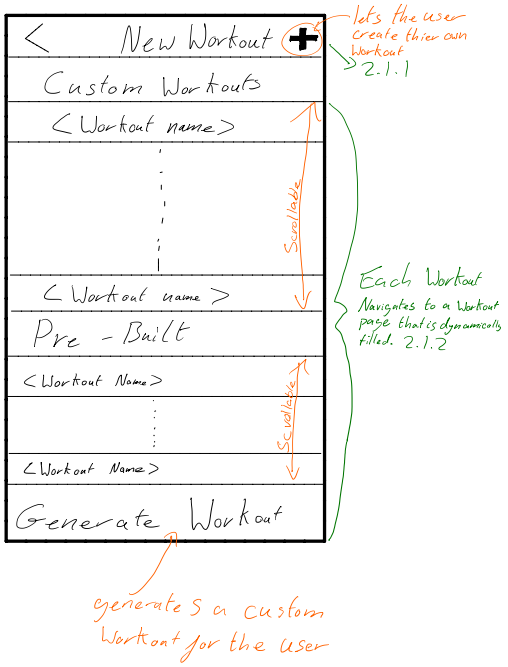
Welcome Screen



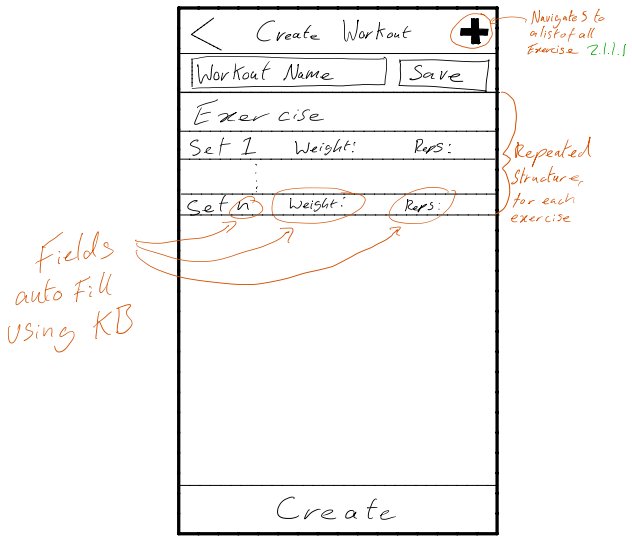
1. Start Screen



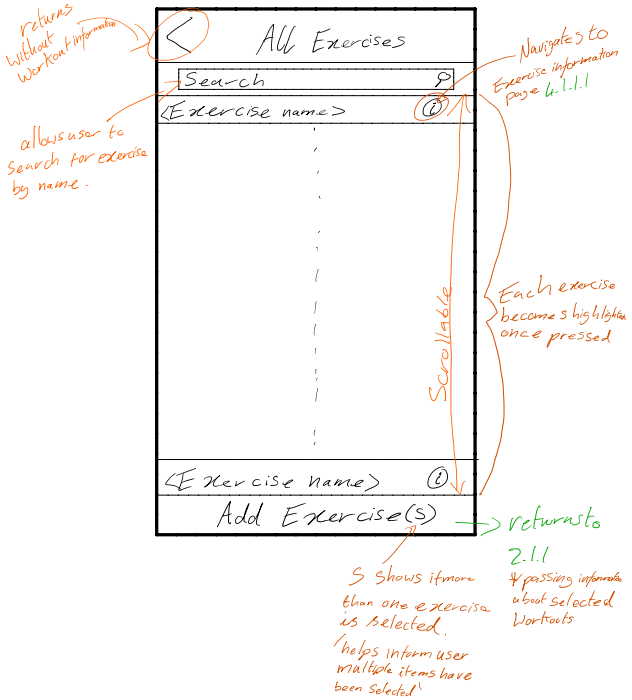
2.1 New Workout Screen



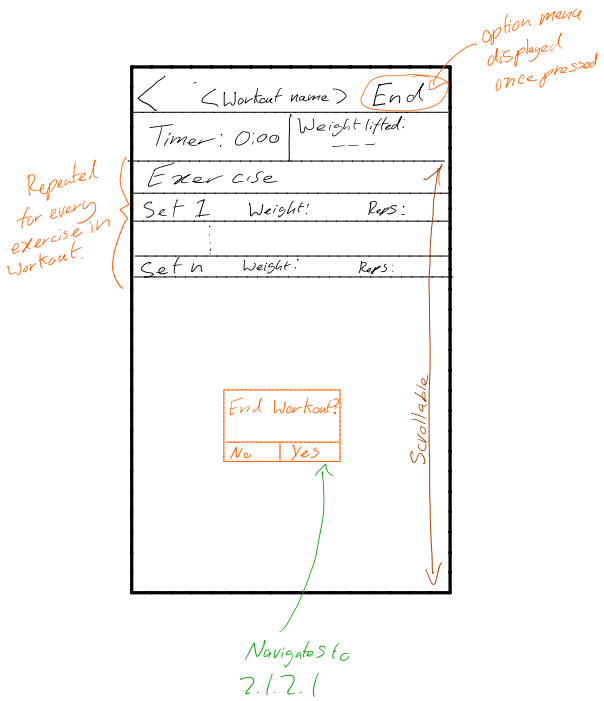
2.1.1. Create Workout



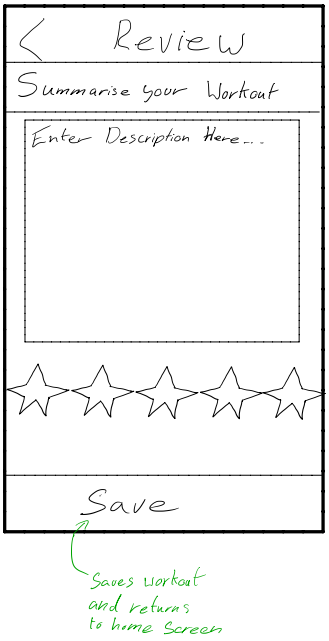
2.1.1 Add Exercise



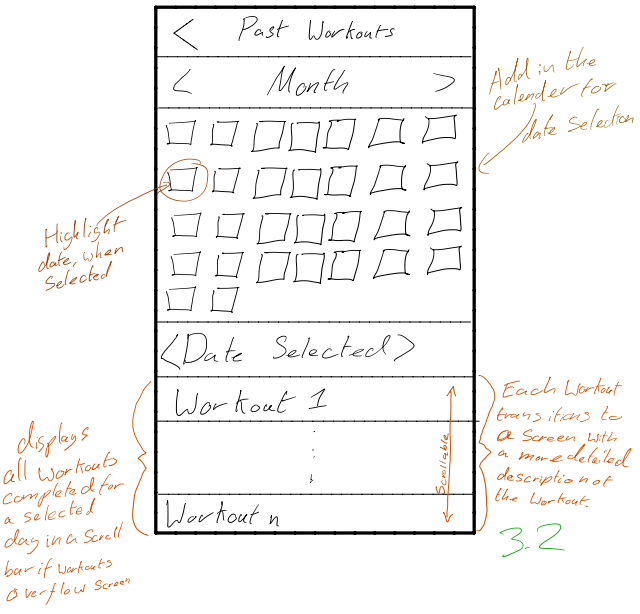
2.1.2 Workout Page



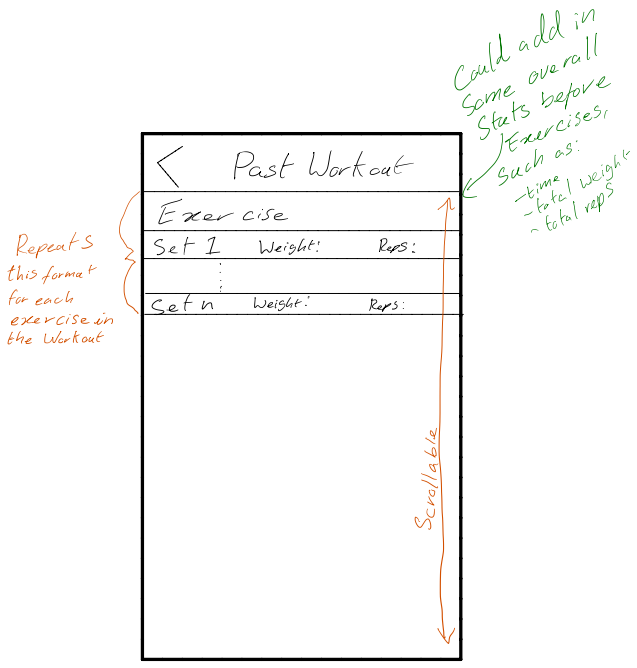
2.1.2.1 Workout Review



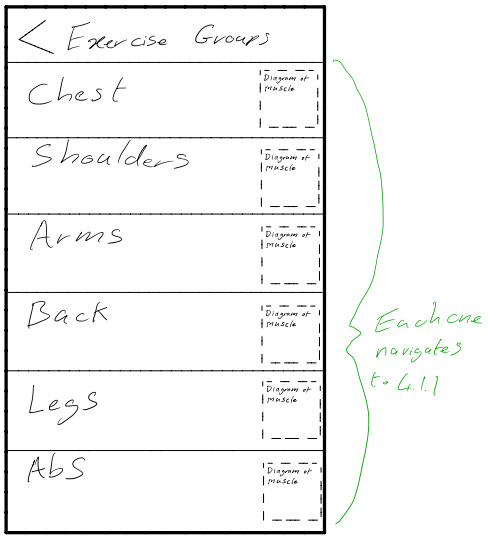
3.1 Past Workouts



3.2 Past Workout

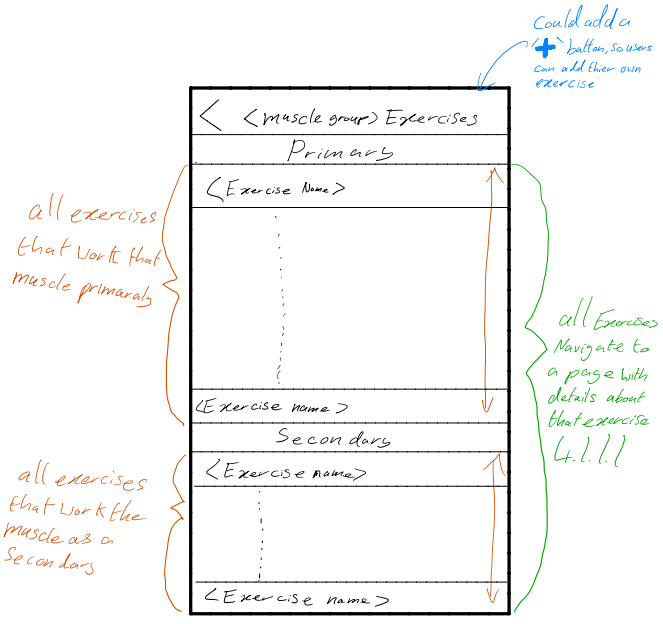


4.1 Exercises

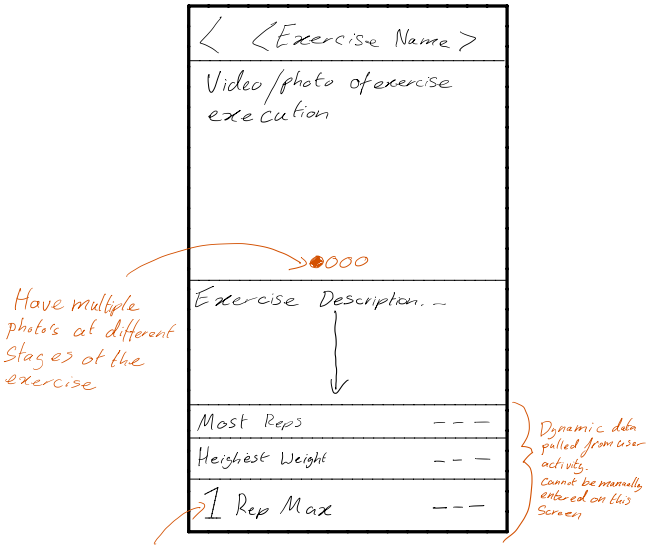


Could add an all section → this would require a new transition screen without primary/secondary dividers.

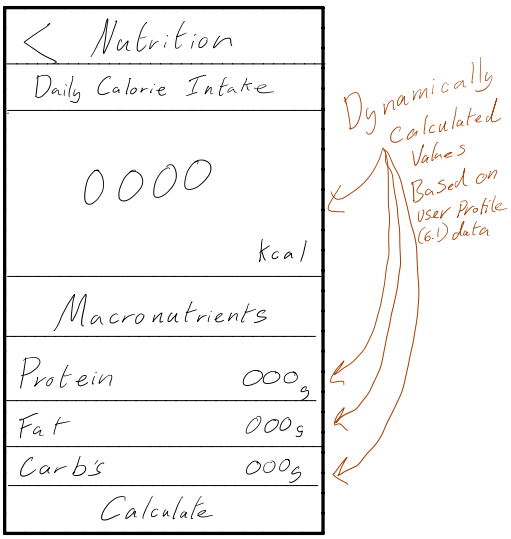
4.1.1 Exercise by Group



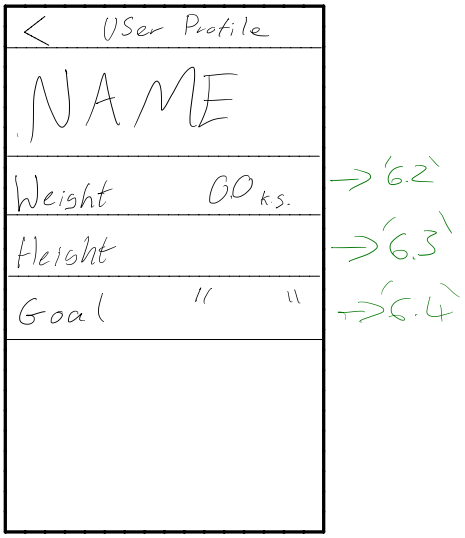
4.1.1.1 Exercise



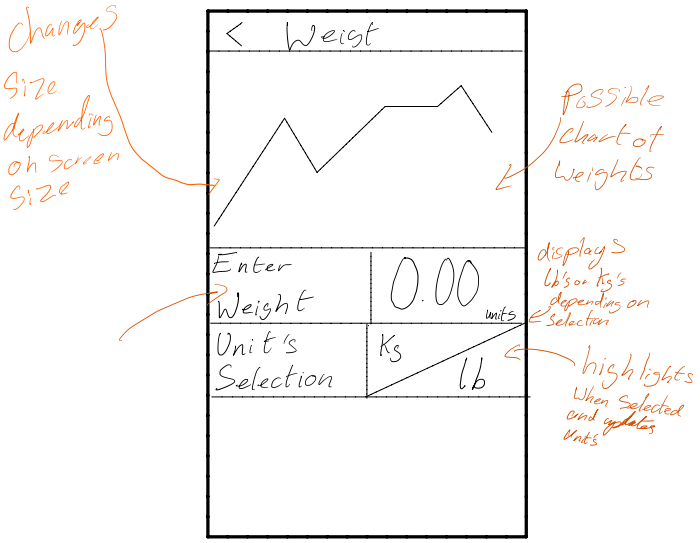
5.1 Nutrition



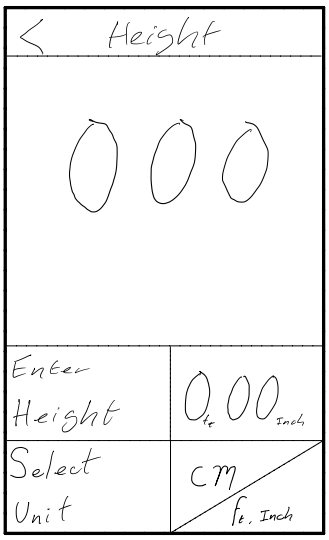
6.1 User Profile



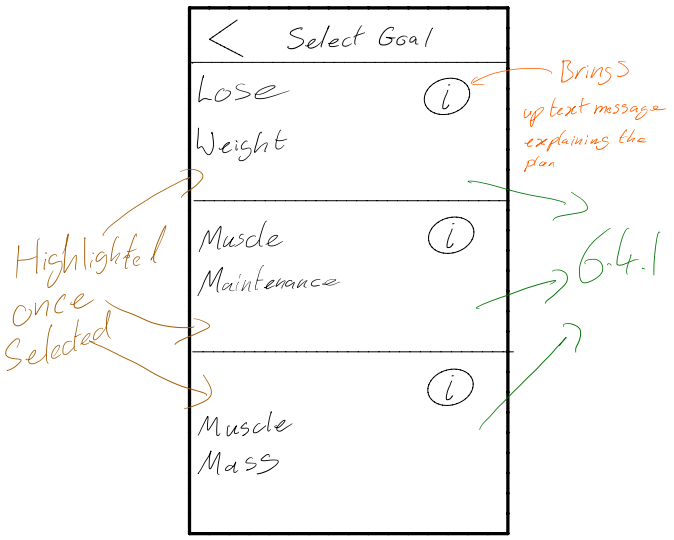
6.2 Weight Screen



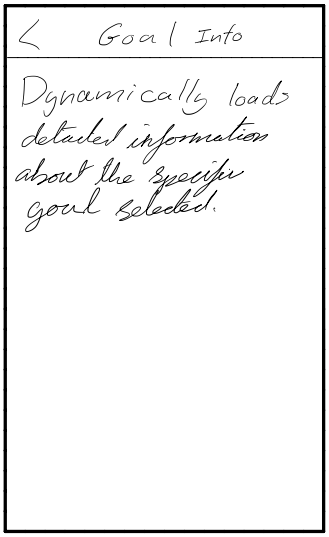
6.3 Height Screen



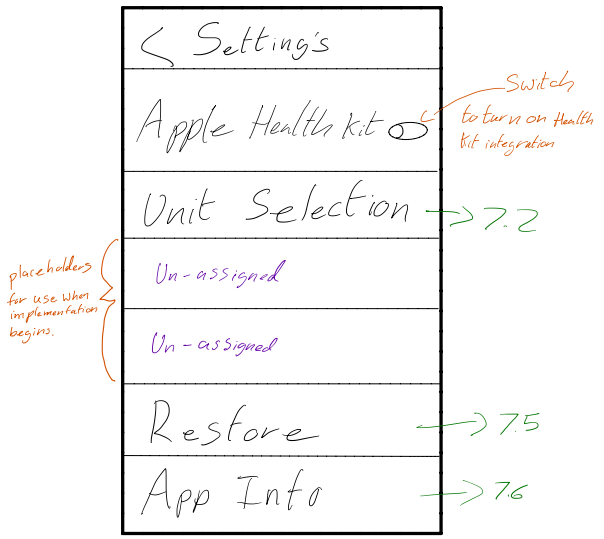
6.4 Goal Selection



6.4.1 Goal Info

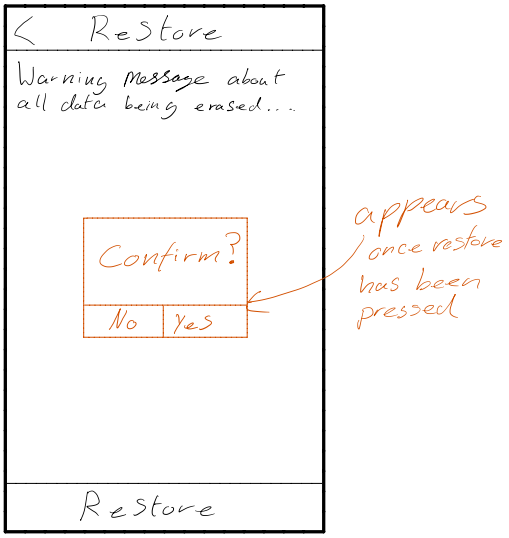


7.1 Settings

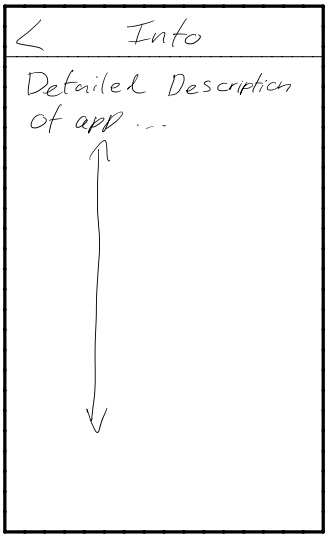


7.2 Unit Selection

7.5 Restore



7.6 App Info



APPENDIX B

Realm Data Storage Objects

B.1 User

B.2 Workout

B.3 Set

B.4 Exercise

```
class UserData : Object{
    dynamic var id = 0
    dynamic var name = ""
    dynamic var height = 0.0
    var weight = List<Weight>()
    dynamic var dob = Date()
    dynamic var goal = ""
    dynamic var profileImagePath = ""
    var Workouts = List<Workout>()
    dynamic var calories = 0
    dynamic var gender = ""
    dynamic var lifestyle = ""
    dynamic var phoneNumber = ""
    dynamic var feedback = false //true if the system should report to octoblu for personalised feedbck
        messages

    override static func primaryKey() -> String? {
        return "id"
    }

}

//additional classs to enable realm ro store doubble values for weigth, as only classes that extend from
    object can be saved in a realm list

class Weight : Object{
    dynamic var value = 0.0
    dynamic var date = Date()
}
```

Figure B.1: Code showing the structure and parameters of the Realm user storage object

```
/*
 * Class to store information about a completed workout schedudle
 */
class Workout : Object{
    dynamic var id = 0
    dynamic var name = ""
    dynamic var rating = 0 //value from 0-5 representing
    dynamic var difficultyRating = 0 //value from 0-5 representing
    dynamic var review = "" //user comments of the workout
    dynamic var date = Date()
    dynamic var time = 0.0 //duration of the workout
    dynamic var totalWeight = 0.0
    dynamic var totalReps = 0
    dynamic var goal : String! //the goal the workout links to most
    dynamic var user = 0 //id of the user
    var sets = List<Sets>()
    dynamic var schedule = "" //name of the schedule the workout is following

    override static func primaryKey() -> String? {
        return "id"
    }
}

/*
 * Class to store information about a workout schedule
 * This is simply the name of the workout the goal it focuses on and which exercises are completed
 * The number of reps and sets for the schedule will be assigned dynmically when a schedule is chosen and
    a 'Workout is begun"
 */
class Schedule : Object{
    dynamic var id = 0
    dynamic var name = ""
    dynamic var goal = ""
    dynamic var userCreated = false
    dynamic var bodyweight = false
    var exercises = List<ExerciseId>()
    var exerciseGroups = List<Groups>()

    override static func primaryKey() -> String? {
        return "name"
    }
}

class ExerciseId : Object{
    dynamic var id = 0
}

class Groups : Object{
    dynamic var group = ""

    override static func primaryKey() -> String? {
        return "group"
    }
}
```

Figure B.2: Code showing the structure and parameters of the Realm workout storage object

```
class Sets : Object{
    // MARK: Properties

    dynamic var number = 0
    var reps = List<Reps>()
    var weights = List<Weights>()
    dynamic var exerciseId = 0
}

class Reps : Object{
    dynamic var count = 0
}

class Weights : Object{
    dynamic var weight = 0.0
}
```

Figure B.3: Code showing the structure and parameters of the Realm set storage object

```
class Exercise : Object{
    // MARK: Properties

    dynamic var id = 0
    dynamic var name = ""
    var imagePaths = List<ExerciseImage>()
    dynamic var muscleGroup = ""
    dynamic var primaryMuscle = ""
    var secondaryMuscles = List<Muscle>()
    dynamic var exerciseDescription = ""
    dynamic var mostReps = 0
    dynamic var mostWeight = 0.0
    dynamic var oneRepMax = 0.0

    override static func primaryKey() -> String? {
        return "name"
    }

}

class Muscle : Object{
    dynamic var muscle = ""
}

class ExerciseImage : Object{
    dynamic var path = ""
}
```

Figure B.4: Code showing the structure and parameters of the Realm exercise storage object

APPENDIX C

Workout Generation Code

```
class WorkoutGenerator {

    var user : UserData!
    let realm = try! Realm()
    var reps = 0
    var weight = 0.0
    var decayValue = 0.95
    var counter = [-1,-1,-1,-1,-1,-1,-1]
    /* [0 : Shoulders
        1 : Chest
        2 : Back
        3 : Arms
        4 : Legs
        5 : Abs]
    */

    func sets(schedule: Schedule) -> List<Sets>{
        let sets = List<Sets>()

        for exerciseId in schedule.exercises{
            let set = Sets()
            let pred = NSPredicate(format: "id == %d", exerciseId.id)
            let exercise = realm.objects(Exercise.self).filter(pred).first
            set.number = calculateNumSets(exercise: exercise!)
            set.exerciseId = exerciseId.id

            //number of exercises that have already worked that muscle group
            let completedExercises = incrementCounter(muscleGroup: (exercise?.muscleGroup)!)

            for _ in 0..<set.number {
                //calculate the values for this set
                let numReps = calculateReps(exercise: exercise!)
                var amountWeight = 0.0

                if (completedExercises <= 0){
                    amountWeight = calculateWeight(exercise: exercise!)
                }else{
                    amountWeight = calculateWeight(exercise: exercise!) * pow(decayValue, Double
                        (completedExercises))
                }

                //add values to set
                set.reps.append(
                    Reps(value: [
                        "count" : numReps
                    ])
                )
                set.weights.append(
                    Weights(value: [
                        "weight" : ceil(amountWeight)
                    ])
                )
            }

            sets.append(set)
        }

        return sets
    }
```

Figure C.1: Code showing the variables and main method to produce a workout containing sets, weights and repetitions; for a supplied workout schedule

```
func incrementCounter(muscleGroup : String) -> Int{
    switch muscleGroup {
    case "Shoulders":
        counter[0] += 1
        return counter[0]
    case "Chest":
        counter[1] += 1
        return counter[1]
    case "Back":
        counter[2] += 1
        return counter[2]
    case "Arms":
        counter[3] += 1
        return counter[3]
    case "Legs":
        counter[4] += 1
        return counter[4]
    case "Abs":
        counter[5] += 1
        return counter[5]
    default:
        return 0
    }
}

func calculateReps(exercise : Exercise) -> Int{
    if (exercise.mostReps == 0){ //this is the first time completeing an exercise and therefore should be performed for as many reps as possible
        return 999 //code to signify maximum
    }

    switch user.goal {
    case "WeightLoss":
        reps = 15
        break
    case "MuscleMaintenance":
        reps = 12
        break
    case "MassGain":
        reps = 8
        break
    default:
        break
    }

    return reps
}
```

Figure C.2: Code showing supporting methods to track muscles used and calculate the correct number of repetitions

```
// method to calculate the weigth that should be lifted on the first attempt at an exercise
func calculateWeight(exercise : Exercise) -> Double{
    if (exercise.mostReps == 0){
        return firstTime(exercise: exercise)
    }

    //using Brzycki formula for 1RM
    weight = exercise.oneRepMax * ( 1.0278 - (0.0278 * Double(reps)) )

    return weight
}


// method to calculate the weigth that should be lifted on the first attempt at an exercise
func firstTime(exercise : Exercise) -> Double{
    return 5.0
}

func calculateNumSets(exercise : Exercise) -> Int {
    if (exercise.mostReps == 0){
        return 1
    }
    return 4
}
```

Figure C.3: Code showing the supporting methods for weight calculations for a given exercise