SmartRep – Android Wear, Motion Sensors & Gestures

CM3203: One Semester Individual Project
Supervisor: Dr. Matthew Morgan
Moderator: Dr. Jianhua Shao

Kieran Mitchell
BSc Computer Science
1433846
mitchellk@cardiff.ac.uk

Abstract

Workout applications are proving extremely popular with smartphone users. Prior to the rise of smartphones and smartphone applications, gym workouts would be logged using pen and paper. Once this functionality became regularly available using Android and iOS applications, many people then decided to record their workout data using a workout journal application which they had installed on their smartphone device.

Currently, the Android market is flooded with various workout journal applications. However, very few of the workout applications on the market implement Android Wear capabilities. This project will focus on the development of a workout journal application built on Android with the added functionality that Android Wear allows for.

Table of Contents

1. Introduction 3
2. Background 7
3. Specification & Design 11
4. Implementation 17
5. Results & Evaluation..... 23
6. Future Work 34
7. Conclusions 32
8. Reflection..... 37
9. References 38
10. Appendices..... 42

Introduction

This section will give an overview of the project. It will discuss the aims of the project, the audience that the project will be targeting, the approach taken to develop the software and any assumptions or constraints that will affect any aspects of the project.

Aims

The primary aim of this project is to produce an Android application which will allow users to log a weightlifting workout in combination with Android Wear functionalities, including motion sensors and heart rate monitors.

The application running on the Android Wear smartwatch will not be standalone. Hence, it will not function without a Bluetooth connection to an Android mobile running the main application. The main objective of the Android Wear application will be automatic-repetition (*auto-rep*) functionality. The auto-rep functionality is defined within this project as repetitions being tracked via an Android Wear device which the user is wearing.

Subsequently, this project will aim to provide the user with heart rate analysis. The heart rate analysis will allow the user to view their heart rate for each exercise and provide insight into which exercises caused their heart rate to reach its highest. This will be a feature for users which have Android Wear smartwatches with heart rate monitors connected.

The project is to be considered a success if the following features are implemented successfully -

1. The number of repetition a user performs is automatically tracked, via Android Wear, for the following exercises:
    - Barbell Deadlift
    - Barbell Squat
    - Barbell Bench Press

2. The user's heart rate will be monitored for select exercises

3. The user's workout is logged and saved within a mobile application including:
    - Exercises completed
    - Sets completed
    - Weight lifted
    - Avg. heart rate
    - Duration of workout
    - Total volume of weight lifted

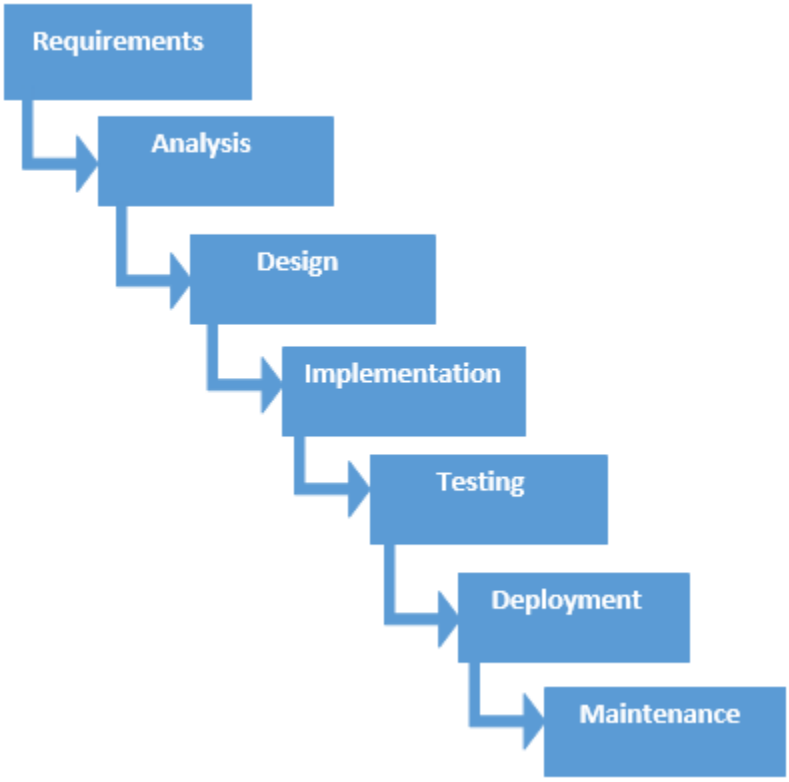
Audience

This application will be targeted toward intermediate & advanced trainers. In terms of intermediate & advanced, this is referring to users who have experience in weight training, capable of creating their own workouts and do not require instructions/examples of how to perform the exercises provided within the application. This is because the application will not provide workout templates or pictures / videos of how to perform said exercises.

The application will be aimed at the audience stated above because there will not be any tutorials or pre-set plans within the application which will show the user the most efficient or correct way of working out. The application will be developed in this fashion because this will prevent the menu and options the user has from becoming complicated and cluttered. Additionally, the timescale of the project will not allow for tutorials and instructions to be developed and included within the application. Therefore, this could be an area to look at in the future of the application if it were to be developed further outside of this project.

Approach

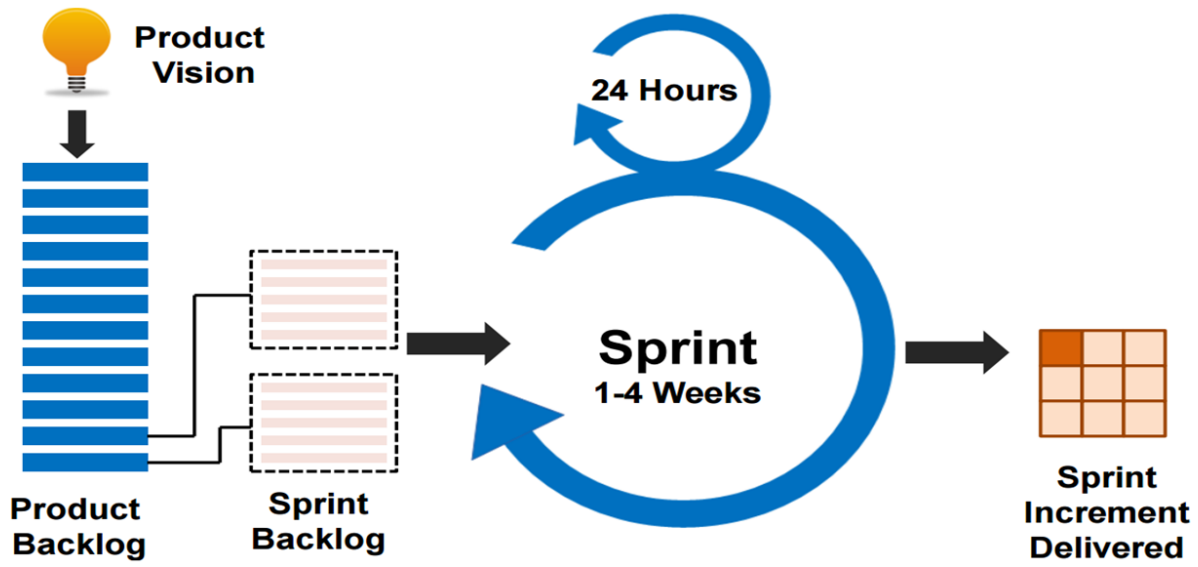
A variety of methodologies have been researched to be used within this project. However, the primary two methodologies considered are The Agile Method and The Waterfall Method. The reason for considering the use of The Waterfall Method is due to the previous years' experience with this method. Additionally, this method is relatively simple and easy to understand. It is also a methodology that produces very clear phases because every area is completed one at time. Hence, nothing overlaps.



Appendix 28: A diagram displaying the flow of The Waterfall Method (TestingFreak, 2017).

The primary issue with The Waterfall Method is that it does not leave any room for flexibility. This is because the process is not iterative meaning that the methodology does not allow for an evolving design which software development often requires. Once an application reaches the testing phase of development, it is extremely difficult to revisit and change something which was not well-thought out within the concept stage. This is illustrated above in Appendix 28.

The Agile Method allows for a test-driven development approach. This means that test cases are written prior to the programming beginning. Test cases can then be integrated with sprints. A sprint is defined as "a set period of time during which specific work has to be completed and made ready for review" (TechTarget, 2017). Therefore, the use of sprints will allow the project to be broken down and will mean there is a consistent sense of urgency as there will always be upcoming deadlines. This can be seen below in Appendix 29.



Appendix 29: A diagram displaying The Agile Model workflow (Lambert, 2016).

The methodology used within this project will be The Agile Model. This will be the most suitable approach as there will be features of the application that will change as the project develops. In turn, this will mean that the design will require changes. For example, the method used to track repetitions can be implemented in multiple ways. However, until adequate testing has been completed, understanding which method will be the most accurate is difficult. Therefore, Agile will allow more flexibility and allow for modifications where necessary throughout this project.

Scope

The combination of the workout journal and Android Wear functionality will make up the scope of this project. The auto-rep functionality will be implemented for three exercise – Barbell Deadlift, Barbell Squat and Barbell Bench Press. The heart rate monitor will also only be implemented for the three exercises mentioned above.

Although, the scope of this project is primarily focussing on three exercises, the auto-rep functionality and the heart rate monitor could be integrated with a larger number of exercises. This would require a larger timescale than this project allows for due to the amount of testing that will be needed.

Assumptions

The following assumptions are what the project will be built upon:

1. Any Android Wear devices included in the project will be running version 1.5+.
2. Android Wear 2.0 will allow the application to run as expected with Android Wear 1.5.

3. The Android Wear device used for testing has the functionality required by the project.
4. All required resources for the project will be obtained.
5. There will be no hardware issues with the accelerometer.
6. All exercises performed testing the auto-rep functionality will be performed in the correct manner.

Constraints

The following constraints are to be considered:

1. Some current Android Wear devices are being updated to Android Wear 2.0 but not all of them.
2. Accuracy of an accelerometer can vary from one Android Wear device to another.
3. Not all Android Wear devices contain accelerometers.
4. Not all Android Wear devices contain heart rate monitors.
5. The small-time frame in which to complete the project (~15 weeks).

Background

This section will provide context into terminology that may be unknown to persons who do not workout or take an interest in fitness. Additionally, this section will discuss the problem area that the project is focussing on, any relevant work that has been performed in order to solve the issue and the solution in which this project will look to implement.

Context

In the following section, insight will be provided in to the terminology and knowledge that will allow for a greater understanding of features within the project.

Regarding '*tracking*' or '*logging'* a weightlifting workout, this is referring to the amount of weight lifted for a specific exercise and for how many repetitions. An exercise performed for one or more repetitions is referred to as a '*set*.' Therefore, repetitions are recorded as a set. A set can be written in shorthand. For example, 10 repetitions of 50kg may be written as *10x50kg.*

Due to the time constraints of this project, the auto-rep functionality within the application will only be implemented for three main exercises –

1. Barbell Bench Press [1]
2. Barbell Deadlift [2]
3. Barbell Squat [3]

These three exercises have been chosen as they are used by powerlifting organisations such as British Powerlifting. Thus, they are very popular exercises by nature. Additionally, they are very popular exercises because they target multiple muscle groups. Exercises which target multiple muscle groups are known as compound movements (Rini, 2014). Any other exercises within the application will require user input for the number of repetitions completed.

Within this project, a repetition will require the user to move through a full range of motion (ROM). This means that in regard to the auto-rep functionality, performing half a repetition on a Barbell Squat will not be recorded by the device. For example, if a person were to be executing a Barbell Bench Press, a full ROM would be considered to be moving the barbell from a where the arms are extended (locked elbows), down to the chest, and back up to an extended position. If the barbell did not touch the chest, this would not be considered full ROM.

Analysis of a user's workout can be presented in the form of *volume lifted*. In order to calculate the volume a person has lifted for a given workout, the following algorithm can be applied:

Training Volume (V) = Sets (S) x Reps (R) x Weight (W) (M&S Team, 2011)

[1] http://www.exrx.net/WeightExercises/Quadriceps/BBSquat.html
[2] http://www.exrx.net/WeightExercises/ErectorSpinae/BBDeadlift.html
[3] http://www.exrx.net/WeightExercises/Quadriceps/BBSquat.html

For example, if a user performed 2 sets of 10 reps using a weight of 50kg, the total volume would be 1000kg (*2 * 10 * 50)*.

A one repetition maximum (one rep max or 1rm) can be defined as "*the most weight you can lift for a defined number of exercise movements. It is a good measure of your current strength level as you follow your weight training program*" (Rodgers, 2016).

However, performing a one rep max is not advised due to the risk of injury and how taxing it is to a person's body. "*Using a 2-3RM allows for better recovery, safer training, more frequent sessions, and better muscle gains than 1RMs."* (Thibaudeau, 2015)

Therefore, a one rep max can be estimated using various algorithms. This is known as a predicted one rep max. Predicted one rep maxes are frequently used within weightlifting and powerlifting because it can give a person an idea of the most weight they would be able to lift without having to actually perform a one rep max.

Problem Area

Currently, there are not any applications in the Google Play store which enable a user to record a gym workout with almost zero interaction. This leaves a variety of problems for any user trying to track a gym workout with their Android smartphone. In the following section, these issues will be addressed.

Initially when performing high intensity exercises, such as the main three previously mentioned, it is very easy to lose count of how many repetition you have performed. The auto-rep functionality which the project is aiming to implement would solve this issue. Assuming the auto-rep functionality tracks repetitions accurately, there would be no need for the user to keep a count of repetitions while performing an exercise.

Furthermore, when performing barbell and dumbbell weight lifting exercises, a person's hands can get dirty. For example, if the barbell were dirty this would be transferred to the user's hands. Alternatively, if a person is using chalk on their hands in order to improve their grip, this can also make their hands dirty. Therefore, many people do not like to interact with their smartphone when their hands are in this state. The auto-rep functionality would significantly reduce the amount of interaction required with a smartphone device whilst working out.

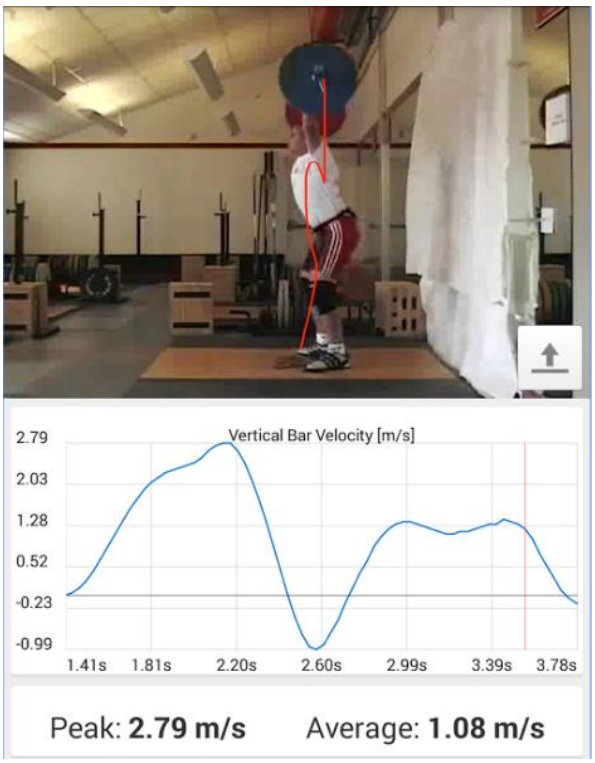
Moreover, to perform to a high level whilst working out, a person is usually in a focussed state. Interacting with a smartphone frequently whilst exercising can detract from a person's focus. Hence, many people do not wish to interact heavily with their device during a workout.

Additionally, there are many methods of monitoring your heartrate as a smartphone user. There is also a wide variety of cardiovascular exercise type applications on the Google Play Store which allow users to monitor their heart rate whilst performing exercise, such as running. There are currently no workout journal type applications on the Google Play store which allow users to monitor their heart rate whilst performing weight lifting exercises, such as the Barbell Squat, Barbell Deadlift and Barbell Bench Press.

Relevant Work

At this moment in time, there are Android applications which track a user's movement when performing weight lifting exercises, such as the Barbell Deadlift. An example of an application of this kind is BarSense[4]. However, the problem with applications of this type is that they are not designed to log workouts as well.

Additionally, with the specific example BarSense, the user must load in a pre-recorded video and mark the object to record the movement of (e.g. the barbell). Thus, the user cannot use this application in real time in order to track movement and store workout data.



Appendix 1: BarSense App Preview

Subsequently, there are also workout log applications which have very minimal interfaces. Due to the minimalistic interface, there are less activities for the user to pass through in order to record their workout data. Hence, slightly less interaction is required by the user. An example of this type of application would be RepCount[5].

However, applications of this type do not offer motion sensor functionality. Consequently, the user would still have to interact with the application every time they wish to record a set. There is zero automation with these kinds of applications.

[4] https://play.google.com/store/apps/details?id=com.barsense.main&hl=en
[5] https://play.google.com/store/apps/details?id=sp.repcount&hl=en

Solution

The solution which has been proposed is unique due to the integration of Android Wear. The Android Wear application will track user movement using motion sensors, more specifically, an accelerometer. This accelerometer will mean repetitions can be automatically tracked and recorded in the Android mobile application without user input.

The position of the Android Wear smartwatch will depend on the exercise being performed. Hence, in theory the accelerometer will track acceleration along different axes dependent on the exercise being performed. However, due to the fact that the accelerometer will only be monitored for the three exercises mentioned, the accelerometer will only need to track acceleration along the X axis for the current level of this project. This is because in all three exercises, the user's arm is always perpendicular to the ground. In turn, this means the Android Wear smartwatch will always be rotated 90° from the default position. However, if this feature were to be implemented with a larger variety of exercises, the accelerometer would also need to monitor acceleration along the Y axis and Z axis.

Furthermore, the application that will be designed within this project will monitor a user's heart rate for any exercises in which repetitions are being automatically tracked. The heart rate monitor will be started in synchronisation with the accelerometer and will also stop monitoring the user's heart rate when the monitoring of the accelerometer finishes at the end of a user's set.

Specification & Design

This section will discuss the route taken in order to develop the functional and non-functional requirements. It will also clearly list the functional and non-functional requirements. Additionally, this section will give insight into the system architecture. It will illustrate the model of the system and how various modules within the system interact.

Requirements

The following functional and non-functional requirements provide the specification into the what the project will be required to do upon completion. These project requirements have been developed using the MoSCoW (Waters, 2009) method.

MoSCoW can be defined using the following requirement categories:

- **Must have** – "Features that must be included before the product can be launched. It is good to have clarity on this before a project begins, as this is the minimum scope for the product to be useful." (Waters, 2009)
- **Should have** – "Features that are not critical to launch, but are considered to be important and of a high value to the user." (Waters, 2009)
- **Could have** – "Features that are nice to have and could potentially be included without incurring too much effort or cost. These will be the first features to be removed from scope if the project's timescales are later at risk." (Waters, 2009)
- **Won't have** – "Features that have been requested but are explicitly excluded from scope for the planned duration, and may be included in a future phase of development." (Waters, 2009)

The requirements below will be tested using a requirement based testing method. Requirement based testing is defined as "a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and non-functional attributes such as performance, reliability or usability" (Tutorials Point, 2017). This testing will involve the designing of test cases, execution of test cases and the documentation of test case results. Upon documenting each test case results, any failing tests will be noted and corrected where possible.

This requirement based testing will take place at multiple stages during the development phases. Initially, tests will be performed at the mid-stage of the project – week 6. This will ensure the development of the application is taking place as planned. Subsequently, the testing will also be performed in the final stages of the development – week 10. This will ensure that all required features have been implemented and that no negative changes have taken place which cause previously passing test cases to fail.

Functional Requirements

The system **must:**

- Allow the user to create a workout consisting of user selected exercises
- Allow the user to record workout data such as weight lifted, repetitions completed.
- Track completed repetitions via Android Wear for:
    - Barbell Squats
    - Barbell Deadlifts

    - Barbell Bench Press
- Monitor a user's heart rate when performing sets of:
    - Barbell Squats
    - Barbell Deadlifts
    - Barbell Bench Press
- Allow completed workouts to be saved locally via a SQLite database.
- Run on Android devices using SDK 23+ / Android 6.0+
- Run on Android Wear devices running Android Wear 1.5+.

The system **should:**

- Allow the user to initiate the start/stop of an exercise with a gesture on the android wear device.
- Allow the user to view previously completed workouts in the form of exercises performed.
- Allow the user to modify data in previously completed workout such as weight lifted and repetitions completed.

The system **could:**

- Allow the user to create custom barbell and dumbbell exercises
- Provide the option of tracking weight in kg or lbs (KG by default)
- Calculate the user's one rep max and present it to them
- Allow the user to enter personal information (height, weight, age, gender)
- Present the most intense exercise the user has performed using heart rate analysis
- Provide post-workout analysis to the user – duration of work out, reps completed, volume lifted

The system **won't:**

- Track completed repetitions via Android Wear for isolation movements
- Run on Android devices running < SDK 20 / Android 4.3 (non-compatible with Android Wear)
- Run on Android Wear devices running < Android Wear 1.5
- Save data externally i.e. on a cloud server

Non-Functional Requirements

The system **must:**

- Allow the user to store as many workouts as their local storage allows
- Track repetitions with a minimum of 90% accuracy of that set
- Send repetitions tracked via Android Wear to the mobile application in under 2 seconds

The system **should:**

- Ask the user if they wish to save/cancel current workout upon exiting application
- Warn the user if their Android Wear device is not compatible
- Allow the user to create a maximum of 50 exercises

The system **could:**

- Provide instructions to the user
- Show an icon in the Android status bar when running

The system **won't:**

- Notify the user if their Android Wear device disconnects

System Architecture

Prior to starting the implementation phase of the project, a vital factor of the development process is to ensure all of the work is backed up regularly. Prior personal experience meant Git functionality would be used within the project. "Git is a free and open source distributed version control system designed to handle everything from small to very large projects" (Git, 2017)

Hence, the next decision was to choose between the two main competitors in terms of web based hosting for source code – GitHub and BitBucket.

"Github and Bitbucket are two of the largest web-based hosting services for source code and development projects" (UpGuard, 2017).

Ultimately, the decision was made to work with Bitbucket primarily due to the pricing structure. Bitbucket provides a free user with an unlimited number of private repositories which would be useful during this project and when working on future projects. However, Github charges each user per repository. Additionally, Bitbucket provides both Windows and Mac clients with an application called SourceTree. "SourceTree simplifies how you interact with your Git and Mercurial repositories so you can focus on coding" (SourceTree, 2017). This would be extremely useful if at any point, during this project or the future, a decision is made to work on a different platform. BitBucket also provides an application called BitBeaker which is an Android BitBucket client.

The decision is supported in the following statement – "If you have many private projects and small numbers of users per project, Bitbucket may be a cheaper option because of its per-repo pricing. If you have large teams collaborating on just a few projects, Github may be the better option." (UpGuard, 2017)

Initially, the system is split into two main segments - Android Mobile and Android Wear. This is illustrated in Appendix 2 which is a diagram displaying the overall design. A choice was made to divide the system in this way because the Android Mobile area of the system will be providing very different functionality to the Android Wear area of the system. Therefore, there is no need for modules within the two areas to be developed closely together.

At the current implementation level of the system, the system would function in a similar manner had it not been divided into the two segments. Although, if the system were to be scaled up in the future, there is no need for the Android Wear development to be included with the Android Mobile area. This is because the Android Wear modules would be already fully implemented. Hence, there would be little development to take place in the future on the Android Wear area of the system. However, the Android Mobile section would be open to much further development in regard to scalability. Therefore, it will be more organised and less confusing for future development if the two areas are kept separate.

The User Interface module, inside of the Android Mobile area, contains every resource that will provide interaction for the user. The user interface will allow the user to navigate throughout the application and apply the functionality that they require. For example, the user interface will allow the user to enter workout data manually,

via the on-screen keyboard. In turn, the user interface will also provide the user with an option to use their Android Wear device in order to automatically track repetitions via motion sensors.

Additionally, the user interface for the Android Mobile application will be much more expansive than the interface presented on the Android Wear device. The user interface for the mobile application will have to allow the user to interact with the whole application. The user interface for the Android Wear application will offer very little interaction. The primary objective for the Android Wear interface will be to present the number of repetitions complete to the user, if they wish to view that detail on their smartwatch.

Developing the Mobile user interface separately from the Android Wear user interface will help to avoid confusion when it comes to future development. Subsequently, the types of activities used to provide user interfaces on the two devices are very different. Therefore, there is no need for them to be developed inside the same module or to be closely related.

Appendix 3 provides an overview of how the objects inside of the application will function together. The system is capable of running in a dynamic fashion depending on what options a user selects. Appendix 3 clearly shows that the object *SmartSet* will inherit from *Set,* as well as providing some additional fields. When a user selects a specific exercise, only one of the previous *SmartSet/Set* objects will be created. Which object is created depends upon whether the user chooses to implement a *SmartSet* or not when given the option. For example, when a user selects the exercise Barbell Squat, the user interface will provide the user with an option of whether they would like to track the repetitions on their Android Wear device (*SmartSet*) or manually input them on their Android Mobile device (*Set*). Hence, the system will dynamically create objects following user choices within the user interface of the mobile application.

The reasoning for designing the system in this way is that it will allow the application to reach a bigger audience. Giving the user an option of a *SmartSet* or a regular *Set* means that the software will not be limited to users who only have Android Wear devices. If a user only has an Android Mobile they are still able to use the application in order to log their workouts because they have the option to manually input their repetition data. Developing the system in a non-dynamic fashion would have severely limited the scalability of the project in the future. This is because at this moment in time, Android Wear devices are still not a frequent item for the everyday Android Mobile user. In the future, Android Wear could well be replaced by something else produced by Google but as long as Android is still being used, the application will still be relevant and useable.

The application passes data through the entire system is a variety of ways. Initially, data must be sent from the Android Wear device to the Android Mobile device. This is due to the fact that no data is stored with the Android Wear smartwatch, only recorded. Thus, as soon as the user informs the Android Wear device they have finished their set and the accelerometer should stop recording, the repetition data must be sent to the Android Mobile device.

The repetition data, i.e. the number of reps, is recorded as an integer data type. This integer is then sent to the Android Mobile module via Bluetooth. The ListenerService file within the Android Mobile module is implemented in order to listen for new repetition data being broadcast from the Android Wear device. Sending the repetition data using this method means that a higher amount of resources is being used than if the application were to send the repetition data once the user has complete his set. However, this design has been chosen because it will

reduce any lag in regards to the Android Wear device sending data to the Android Mobile device. This is because the Android Mobile device will be receiving the repetition data as soon as it has been updated instead of waiting for the user to mark the set as complete.

Subsequently, the Android Mobile modules of the project pass data around the application using Intents (see definition below). The relevant data is placed inside of an intent and can be read and used by the next Activity.

"An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed." (Android.com, 2017)

The majority of data inside of the Android Mobile modules are integer and string types. Integer and string types are very easy to pass around an Android application. Hence, these two data types are used as often as possible. They are also very easy to implement with the XML modules of the Android application.

Furthermore, the SQLite database that is implemented in the design does not support all data types. However, SQLite does support String and Integer data types. Thus, the more of the data which can remain as these two data types would mean less resources would be needed when it came to storing the data inside the database. For example, once the application is finished with the relevant user data (i.e. the workout is complete and overview has been shown) the data can be stored straight into the database with very little, if any, conversion of data types needed.

However, it is not as simple to pass data around inside Intents in terms of custom objects, such as the Exercise object. Therefore, any custom objects which are passed throughout the application are using the Parcelable module provided by Android - "Interface for classes whose instances can be written to and restored from a Parcel" (Android.com, 2017). Using Parcelable means that efficiency is increased throughout the application. The reason for this is once a custom object was integrated with the Parcelable module, it could be passed around multiple times, without having to repeat large amounts of code.

There are other data types used within the application, such as Doubles. In regards to the weight lifted by the user, the data is stored as a double. This is required because, more often than not, weights being used are not whole numbers. For example, in the United Kingdom, frequently used weight plates are 1.25kg and 2.5kg. Hence, there are times were the weight that needs to be recorded will have to be a decimal number. The Integer data type does not allow this.

Moreover, an algorithm being used within the system where integers and doubles are being used is the algorithm for calculating a user's one rep max. There are various algorithms that can be used to calculate a user's one rep max. However, they are all similar to a certain degree - "You multiply the amount of weight you can handle for a given number of reps by a certain coefficient that corresponds to that number of reps" (Butt, 2017).

The one rep algorithm chosen in this project is known as the Brzycki (Mac, 2017) calculation. This algorithm has been used because this is the algorithm favoured by a well-respected scientific journalist in the weight training and nutrition industry, Eric Helms. He uses it within his own training program stating that "there are a few different

ways to calculate 1RM. I've found Brzycki calculation to be the most accurate in trained lifters, and so that's the one I use with my clients. However, it gets far less accurate the higher the reps go" (Helms, 2016).

Additional research also supports the claim that Helms makes regarding the accuracy of the Brzycki calculation. "The results suggest that the Brzycki equation may be considered an alternative fairly attractive for the estimation of the values of maximal load (1-RM) for the bench press, from the performance of submaximal tests of 7-10 RM, at least in sedentary or moderately active male adults." (Matheus Amarante do Nascimento, 2007)

The Brzycki equation is as follows:

$$Weight \div (1.0278 - (0.0278 \times Number\ of\ repetitions))$$

An example implementing the Brzycki equation is if a person is able to bench press 90kg for 5 reps, then the calculation for their 1 rep max is: 90 ÷ (1.0278 – (0.0278 x 5)). This would calculate that the person's estimated 1 rep max is 101kg.

Moreover, regarding design, the application includes include a pseudo-distance monitor by tracking the number of changes in acceleration monitored by the Android Wear device. This will be explained in further depth in Implementation section of the report. However, the more changes in acceleration will essentially mean a greater distance moved. Hence, the application can monitor a metric which is similar to distance in order to determine the amount of movement required for one complete repetition. The minimum amount of movement required will be different depending on the exercise being performed by the user.

An issue with implementing the accelerometer as a main factor in calculating repetitions complete, is that it has potential to pick up anomalies. For example, slight movements by the user prior to starting the exercise may be monitored by the device and contribute towards a repetition. Thus, the application requires a lower-bound which will act as a minimum amount of acceleration required. This eliminates the issue of small accelerations being picked up.

Implementation

This section will discuss key features within the system – the accelerometer, heart rate monitor and synchronisation of data. It will provide understanding into how features have been implemented and why they have been implemented in such a way.

Accelerometer

Throughout the development of the project, there have been many unforeseen issues. Initially, a problem which was unexpected was that the implementation of the accelerometer would not allow the monitoring of distance that the Android Wear device had moved since it's starting position. Monitoring of distance via the Android Wear device was one of the methods stated as a possible solution in the Initial Plan.

However, the application required a distance metric to use as parameters for the accelerometer. This issue was overcome by implementing a pseudo-distance monitor. The accelerometer implementation monitors the amount of acceleration changes per movement. Hence, the application uses the acceleration changes as a parameter for each Exercise object. In other words, an exercise that has a greater range of motion will require a greater number of minimum acceleration changes to increment the repetition counter than an exercise that has a smaller range of motion.

An example of this is the Barbell Squat vs the Barbell Deadlift. A correctly performed Barbell Squat should always have a greater range of motion than a Barbell Deadlift. Thus, the minimum acceleration changes required for a Barbell Deadlift have been set lower than the minimum acceleration changes required for a Barbell Squat.

```
getAccelerometer(Exercise) (
        read and set accelerationValue from accelerometer
        read minimumPositive and minimumNegative from data layer
        read count value
        int positiveAcceleration = 0;
        int negativeAcceleration = 0;

        if (accelerationValue > 2) { increase positiveAcceleration by 1; }
        if (accelerationValue < -2) { increase negativeAcceleration by 1; }

        if (positiveAcceleration > minimumPositive AND negativeAcceleration >
mimmumNegative) {
                increment repetition count by 1;
                send repetition count to data layer;
                reset positiveCount and negativeCount to 0;
        }

        if (absolute value of accelerationValue < 0.01)
                positiveCount = 0;
                negativeCount = 0;
        }
)
```

Appendix 4: Pseudocode of the getAcceleration() method used in Android Wear module of code.

Appendix 4 shows how code has been built on top of the accelerometer features provided by Android and how the implementation of the pseudo-distance metric is working. The implementation shown above is critical to the functionality of the application. Without the implementation shown in Appendix 4, the auto-rep functionality would not work. Thus, the application would be missing a key area and, in turn, not satisfy one of the primary objectives stated in the Introduction section.

This is an innovative way of monitoring *"distance"* moved by the Android Wear device. It is not possible to have an increased number of acceleration changes recorded without the device having to have moved a certain distance. The above pseudocode illustrates that each time an acceleration value is recorded which is greater than 2 or less than -2, the number of positive or negative acceleration changes should be increased accordingly. Thus, any acceleration value, that does not seem to be an anomaly (>1), will cause one of the acceleration changes to be incremented.

The current number of positive and negative acceleration changes is then compared to the minimum acceleration changes required by the specific Exercise object being performed. This causes the count to be incremented by 1. Hence, the number of repetitions complete can be updated on the Android Mobile device.

Heart Rate Monitor

Initially, the heart rate monitor functionality proved to be simpler to implement than the accelerometer functionality. Therefore, less issues during development of this feature were discovered. This feature was simpler to implement because Android provide an API which allows the heart rate monitor to be activated.

```
getHeartRate(Array heartBeats) {
        totalHeartBeats = 0;
        for (heartbeat in HeartBeats) {
                totalHeartBeats += heartbeat
        }
        heartRate = totalHeartBeats / size of heartBeats array
        return heartRate;
}
```

Appendix 5: Heart rate monitor pseudo code

The heart rate monitor is simply gathering data regarding the user's heart rate. However, the more complex side of this feature is calculating the user's average heart rate for each set. In order to do this, the application stores each heart rate the Android Wear device monitors for the set the user is performing in an array. It will then calculate the user's average heart rate by finding the sum of all the heart rates stored in the array and dividing this number by the size of the array.

The same methodology is then applied to calculating the user's average heart rate for each whole workout. Again, this is done by finding the sum of all the average heart rates stored for each completed set. This number is then divided by the total number of sets in order to calculate the average heart rate for the entire workout.

Syncing Data

A subsequent problem encountered throughout the development process of the application was finding a suitable method of sending data from the Android Wear device to the Android Mobile device. The initial problem grew from the fact that there is very little documentation provided by Android.com and elsewhere showing how to correctly send data from Android Wear to Android Mobile. This is because most documentation illustrates how to send data from Android Mobile devices to Android Wear devices and not the other way around. This is due to the fact most applications function in the sense that they look to push notifications to the Android Wear device.

Therefore, in order to implement the functionality required, various documentation regarding how to send data from Android Mobile to Android Wear was studied. During this process, reverse-engineering the code that the documentation was providing proved useful. Following that, how to do the opposite operation became clear – sending data from Android Wear to Android Mobile. There are also certain GitHub repositories providing an insight in how to perform the functionality required, such as AndroidWearable-Samples (Mauimauer, 2014). However, examples such as this one come with very little explanation.

The ListenerService module inside the Android Mobile modules goes hand in hand with receiving the data sent by the Android Wearable device. The Listener Service module works on top of the GoogleApiClient API provided by Android. This is shown below in Appendix 6.

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.round_activity_main);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        googleClient = new GoogleApiClient.Builder(this)
                .addApi(Wearable.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();
    }

    @Override
    public void onStart() {
        super.onStart();
        googleClient.connect();
    }


   public void run() {
        // Construct a DataRequest and send over the data layer
        PutDataMapRequest putDMR = PutDataMapRequest.create(path);
        putDMR.getDataMap().putAll(dataMap);
        PutDataRequest request = putDMR.asPutDataRequest();
        DataApi.DataItemResult result = Wearable.DataApi.putDataItem(googleClient,
request).await();
        if (result.getStatus().isSuccess()) {
            Log.v("SendDataLayer", "DataMap: " + dataMap + " sent successfully to data
layer ");
        }
        else {
            // Log an error
            Log.v("myTag", "ERROR: failed to send DataMap to data layer");
        }
    }
```

Appendix 6: Implementation used to send data from Wearable to Mobile

The above code in Appendix 6 shows the implementation used alongside the ListenerService. The above code functions efficiently by placing the relevant data into the Data Layer used by Android. The data inside of the Data Layer can then be sent to the Android Mobile device and called upon to be used when needed.

Initially, the GoogleApiClient object is instantiated. It is at this point where all the relevant data is given to the object prior to it attempting to connect to the data layer. Inside of the onCreate method, which is called every time the application starts, the googleClient object is connected to the Data Layer. Following a successful connection, the run() method places the relevant data into a DataMap object.

The data to be placed inside the Data Layer is placed into a DataMap object prior to be sent to the Data Layer. Once the DataMap object is created with the relevant data inside it, the data is sent to the Data Layer using the putDataItem() method.

At this point in time there is no alternative way of sending data from an Android Wearable device to an Android Mobile device without making use of the Data Layer. However, there are alternative wrappers that can be used instead of the DataMap object chosen. The reason for choosing to use the DataMap is that it is recommended by Android in the following statement – "When possible, use the DataMap class. This approach lets you work with data items in the form of an Android Bundle, so the system does object serialization and deserialization for you, and you can manipulate data with key-value pairs." (Android, 2017)

The previous statement means that because the data items were placed inside an Android Bundle, the application would be able to call that bundle inside of the Android Mobile class. Thus, the data is passed from Android Wearable to Android Mobile.

The ListenerService module then broadcasts the DataMap layer every time that data is changed. This is shown below in Appendix 7.

```
    @Override
    public void onDataChanged(DataEventBuffer dataEvents) {

        DataMap dataMap;
        for (DataEvent event : dataEvents) {

            // Check the data type
            if (event.getType() == DataEvent.TYPE_CHANGED) {
                // Check the data path
                String path = event.getDataItem().getUri().getPath();
                if (path.equals(WEARABLE_DATA_PATH)) {
                    dataMap = DataMapItem.fromDataItem(event.getDataItem()).getDataMap();

                    int dm = dataMap.getInt("repetitions"); // Convert datamap to int
before putting in intent
                    Log.v("ReceiveDataLayer", "DataMap received on mobile: " + dataMap);

                    // Broadcast message to wearable activity for display
                    Intent dataIntent = new Intent();
                    dataIntent.setAction(Intent.ACTION_SEND);
                    dataIntent.putExtra("repetitions", dm);

                    LocalBroadcastManager.getInstance(this).sendBroadcast(dataIntent);

                }
            }
        }
    }
}
```

Appendix 7: Broadcast change in data layer

The above code updates every time there is a change in the data layer. It checks for changes in the Data Layer using the DataEvent.TYPE_CHANGED method. The class then checks that the change in data is coming from the

correct path. In other words, the code is ensuring that the data which has been modified is being sent from the Android Wear device running the same application. Providing this check means that the correct data will always be being broadcast to the rest of the application.

Once all checks have been passed, the repetition count inside of the DataMap is converted to an integer. This is then placed inside an intent with a key named “repetitions”. The intent is then broadcast. This means that any activity within the Android Mobile module can receive the Extra by using the getExtra() method and providing the correct key – “repetitions”. This same method is used for passing the heart rate data from the Android Wear device to the Android Mobile device with the key “heartbeat”.

The application could have used an alternative route and only sent the Intent to a specific activity. This would mean that the intent did not have to be broadcast. However, the decision was made to forgo the option of saving resources because the repetition count is a key feature in the application. Therefore, it needs to be able to be accessed from anywhere inside the application and not just sent to a specific activity.

Results & Evaluation

This section will discuss areas of testing within the project such as physical user testing which took place and the test case results. It will analyse the impact that each test case result has on the project. Additionally, this section will provide a project appraisal assessing the project via the aims provided initially.

User Testing

Once the auto-rep functionality is implemented, it requires user testing to ensure there was a high level of accuracy for users of different heights. Testing the functionality with users of different heights would check whether the auto-rep feature still provided accuracy despite each user's range of motion being different for every exercise.

Initially, 9 volunteers agreed to use the application for testing purposes. All 9 volunteers were of different heights, ranging from 5ft6 up to 6ft2. Gathering users of different heights was a crucial part of the user testing phase.

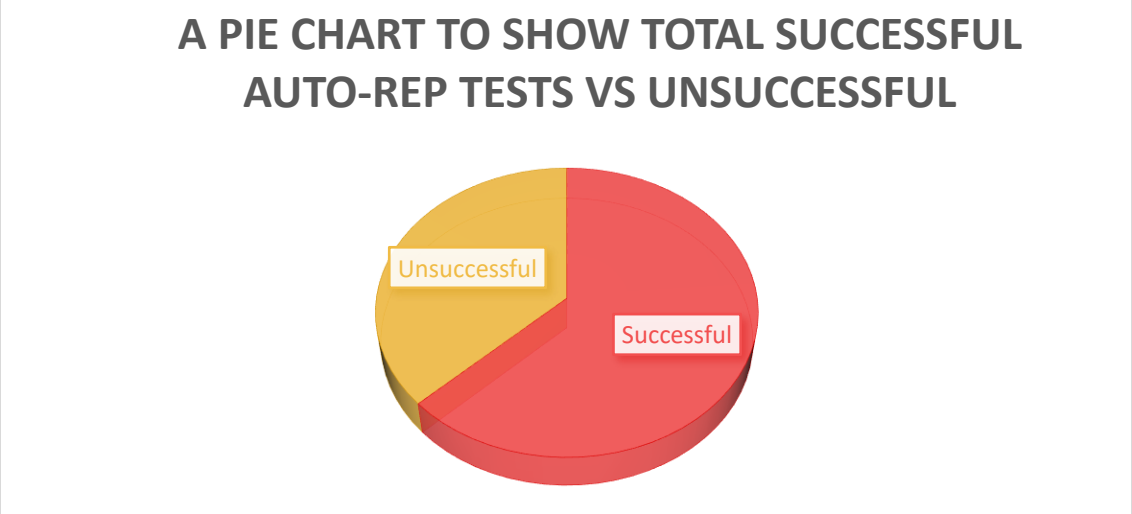
The user testing required each user to perform:

- 10 repetitions of the Barbell Squat
- 10 repetitions of the Barbell Deadlift
- 10 repetitions of the Barbell Bench Press

Each time 10 repetitions were performed by the user, the accuracy required is 90% (as defined by the non-functional requirement). If the accuracy of the auto-rep functionality was less than 90% (i.e. less than 9 of the 10 reps were tracked) then the user test result is a fail.

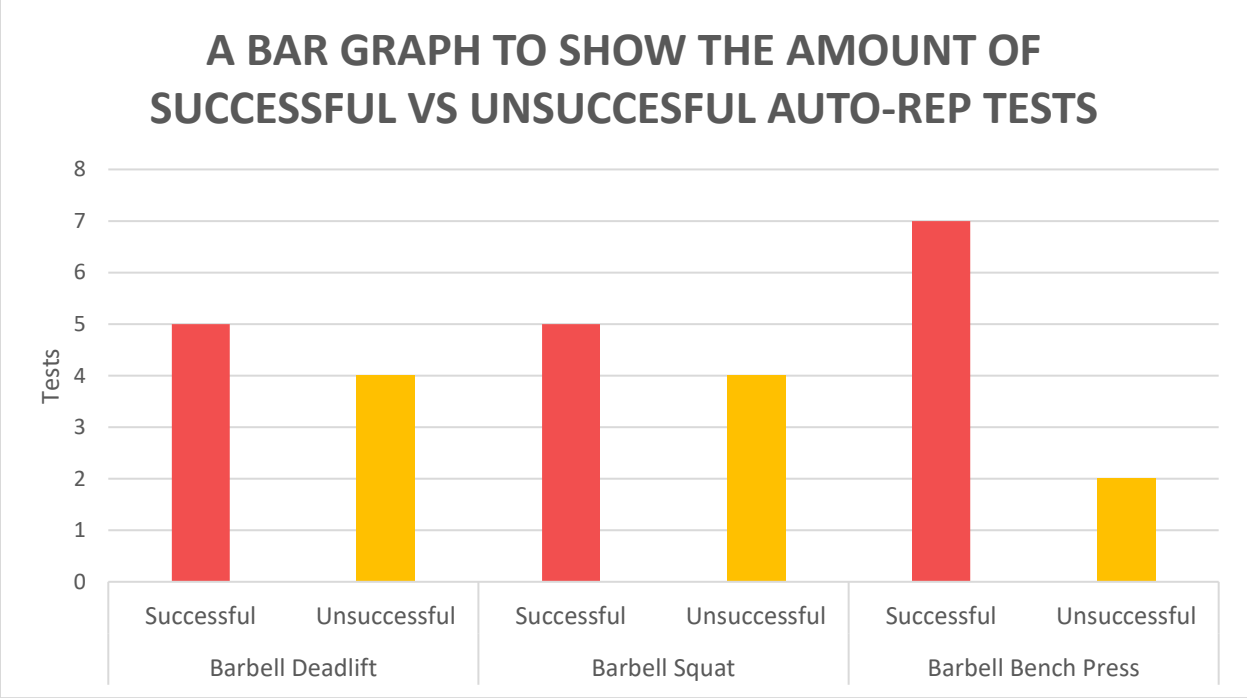
All the user tests were performed under the same test conditions using the same equipment. The Android Wear device used to perform these tests was the Moto 360 Sport[6].

[6] https://www.motorola.com/us/products/moto-360-sport



Appendix 8: A pie chart to show the total successful auto-rep tests vs total unsuccessful auto-rep tests

Initially, Appendix 8 clearly shows that the majority of user tests regarding the auto-rep functionality proved to be successful. This indicates that the application is functioning correctly for the majority of users. However, more analysis is required.

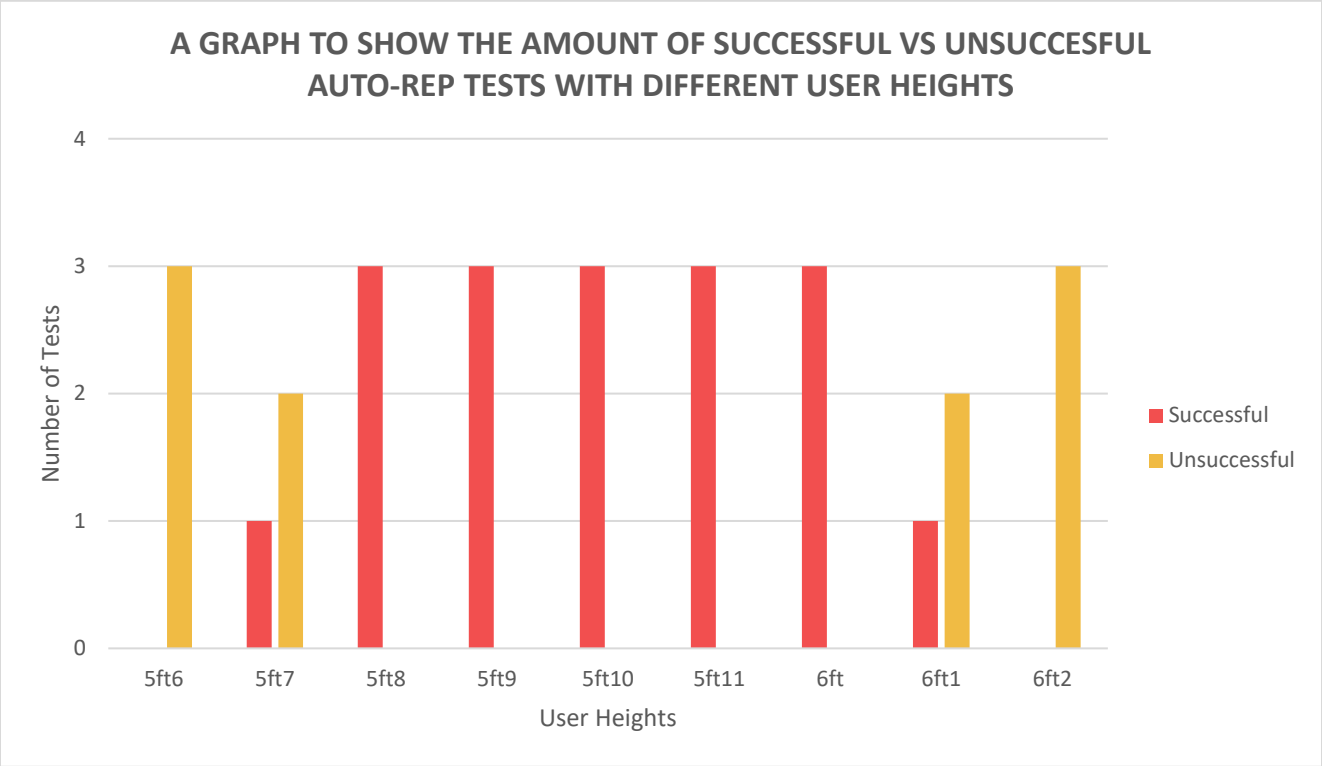


Appendix 9: A bar graph to show the amount of successful vs unsuccessful auto-rep tests with all exercises

Upon further analysis, Appendix 9 illustrates the fact that the Barbell Bench Press proved to be the most successful exercise in terms of the auto-rep implementation. This is because despite the number of users with varying heights, the range of motion of the Barbell Bench Press differs far less than that of the Barbell Squat and Barbell

Deadlift. For example, a 5ft 9 user performing a Barbell Bench Press will have a similar range of motion to a 6ft user performing a Barbell Bench Press.

Appendix 9 also concludes that the Barbell Deadlift and Barbell Squat provided identical test results in terms of successful vs unsuccessful tests. This is because both of these exercises are primarily lower body exercises that require a similar range of motion.



Appendix 10: A bar graph to show the amount of successful vs unsuccessful auto-rep tests with all user heights

Ultimately, Appendix 10 demonstrates that users in the height range of 5ft8 to 6ft all passed with 90% accuracy or more. However, Appendix 10 concludes that the application loses accuracy for users of heights less than 5ft8 or greater than 6ft. This is due to the fact that users of these heights either have a significantly greater or significantly smaller range of motion than the other users when performing all of the exercises. This is an issue that came of little surprise and would be dealt with had the project had a greater time scale. The plan of how this issue would be dealt with is discussed in the Future Work section of this report.

Test Case Results

Presented in the following section, will be the test cases executed: ID, descriptions, result. All of the test cases have been implemented in order to test for the “must-have” functional requirements stated earlier in this report. Hence, listed within the test cases will be the functional requirement they are testing against. This section of the report will also explain how the results of each test case demonstrate that the system is functioning as expected or not and what impact this has on the project.

Note: The full test cases, presented in table format, can be found in the appendices at the end of this report from appendix 11 and onwards.

ID: 01
Functional Requirement: Allow the user to create a workout consisting of selected exercises
Title: Create workout of selected exercises
Description: Test if a user can create a workout by selecting a list of exercises
Result: Pass

This test case is testing whether a user can create a workout through selecting exercises from the pre-set exercises within the application. This test case is given a priority of "High" due to the fact that without a pass result, the rest of the system would not be able to function. A pass result on this test case means that the user will be able to go on to use the primary functionality within the application. For example, if the user can select from a list of exercises, they can then go on to use the auto-rep functionality for some of their chosen exercises.

ID: 02
Functional Requirement: Allow the user to record workout data such as weight lifted, repetitions completed.
Title: User can record workout data
Description: Test whether a user can enter their workout data (weight lifted & repetitions complete)
Result: Pass

This subsequent test is ensuring a user is able to record their own workout data (weight lifted, repetitions complete) for every exercise they perform. Obtaining a pass result for this test case indicates that the user will be able to use deeper functionality within the application, such as save their workout into a local database.

ID: 03
Functional Requirement: Track completed repetitions via Android Wear for the Barbell Squat
Title: Test auto-rep functionality for the Barbell Squat
Description: The application automatically tracks user repetitions for the Barbell Squat exercises via the Android Wearable smartwatch.
Result: Pass

ID: 03_b
Functional Requirement: Track completed repetitions via Android Wear for the Barbell Deadlift
Title: Test auto-rep functionality for the Barbell Deadlift
Description: The application automatically tracks user repetitions for the Barbell Deadlift exercises via the Android Wearable smartwatch.
Result: Pass

ID: 03_c
Functional Requirement: Track completed repetitions via Android Wear for the Bench Press
Title: Test auto-rep functionality for the Barbell Bench Press
Description: The application automatically tracks user repetitions for the Barbell Bench Press exercises via the Android Wearable smartwatch.
Result: Pass

Test Cases 03, 03_b and 03_c are checking whether or not the application on the Android Wear device is able to automatically track user repetitions for the Barbell squat, Barbell Deadlift and Barbell Bench Press. This is the primary functionality within the application. Thus, it is crucial to the success of the project that these tests consistently provide pass results. It is also vital to the system that these test cases pass due to the fact that without the correct implementation of the auto-rep functionality, the application would be extremely similar to many

other workout journal applications on the Google Play store. It is this feature which makes this system unique. Therefore, all three of these test cases have been given a priority of “High”.

ID: 04
Functional Requirement: Allow completed workouts to be saved locally via a SQLite database.
Title: Test if the user workout data saves to an SQLite database
Description: The application saves user recorded workout data into an SQLite database stored locally on the user device.
Result: Pass

Test Case 04 is checking whether or not the application saves the workout data, which has been recorded by the user, into an SQLite database stored locally on the user’s smartphone. If this test case were to fail, it would mean that a user of the application would essentially be wasting their time by recording workout data into it. This is because upon exiting the application, none of their data would be saved. It is for this reason that this test case is given this test case a priority of “High.”

ID: 05
Functional Requirement: Allow the user to initiate the start/stop of an exercise with a gesture on the android wear device.
Title: Test if Android Wear gesture initiates exercise
Description: Test whether the “arm-flick” gesture initiates the Android Wear accelerometer. In turn initiating the auto-rep functionality.
Result: Pass

This test case is testing if the “arm-flick” gesture, applied to the Android Wear device, will initiate the accelerometer which will in turn start the auto-rep functionality. If this test case were to fail, the user would still be able to initiate the start/stop of an exercise using the “Start Exercise” and “Complete Exercise” switch on the Android Wear device’s screen. Therefore, the primary functionality of the application would still work whether this test case passed or failed. Hence, this test case is given a priority of “Medium”.

ID: 06
Functional Requirement: Allow the user to view previously completed workouts in the form of exercises performed.
Title: View previous workout
Description: Test if the user is able to view the name of exercises performed in previously saved workouts
Result: Pass

Test case 06 is testing whether the user is able to view previously completed workouts. It is crucial that this test case passes when executed. This test case is ensuring that there is an area within the application which allows the user to view their workout history. Without being able to view their workout history, a user would essentially be wasting their time in using this application as a workout journal. This is since upon re-launching the application at a later date, there would be no way for the user to view any completed workouts. Hence, this test case has been given a priority of “High”.

ID: 07
Functional Requirement: Allow the user to modify data in previously completed workout such as weight lifted and repetitions completed.
Title: Modify previous workout
Description: Test whether the user can modify previous workout data (weight lifted, repetitions completed)

Result: Fail

This test case is checking if the user is able to modify previously entered workout data. It is an important aspect of the application because the user may realise they have incorrectly entered workout data at a later date than when they marked the workout complete. It is for this reason that the test case has been assigned a priority of "Medium". Although this feature is an important aspect of the application and this test case, the user is still presented with an opportunity to change any incorrect data at the workout overview screen when they mark the workout as complete. This overview activity gives a good opportunity for the user to spot any incorrect data they have entered before closing the application.

ID: 08
Functional Requirement: Provide post-workout analysis to the user – duration of work out, volume lifted, most intense exercise
Title: Give post-workout analysis
Description: Test if whether the user is presented with a post-workout analysis upon workout completion
Result: Pass

This test case is checking if a post-workout analysis is presented to the user upon completion of their workout. It is important that this test case passes because this is the section of the application where the user can view their workout. It will provide information to the user that cannot be accessed until the workout is marked as complete, such as the duration of the workout.

ID: 09
Functional Requirement: Monitor a user's heart rate and record it for Barbell Squats
Title: Record average heart rate for Barbell Squat
Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Squat
Result: Pass

ID: 09_b
Functional Requirement: Monitor a user's heart rate and record it for Barbell Deadlifts
Title: Record average heart rate for Barbell Deadlift
Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Deadlift
Result: Pass

ID: 09_c
Functional Requirement: Monitor a user's heart rate and record it for Barbell Bench Press
Title: Record average heart rate for Barbell Bench Press
Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Bench Press
Result: Pass

Test cases 09, 09_b and 09_c are all testing if a user's heart rate is recorded for the three exercises the Android Wear functionality is integrated with. The heart rate monitor is a vital area of this project. Therefore, it is extremely important that all three of the above test cases pass when executed. If any of the three test cases above were to fail it would indicate that the heart rate monitor is not functioning as it should. This would then have an impact of the aims of the project and essentially prevent the project from being marked as successful. Additionally,

this would have an impact on other areas of the application if the heart rate monitor was malfunctioning due to the fact that the user's heart rate for each workout is presented in various areas of the application.

ID: 10
Functional Requirement: Allow the user to create custom barbell and dumbbell exercises
Title: Create custom exercises
Description: Test whether the user is able to create their own custom exercises to use within a workout
Result: Fail

Test Case 10 is testing whether or not the user is able to create their own custom exercises within the application. Although this test case fails, the application will still perform the key functionality. The most popular barbell and dumbbell exercises have been implemented into the system for the user to perform upon start up. Therefore, there would not be a high number of cases where the user would feel the need to implement their own custom exercise. It is for this reason that the failing of this test case is not a major issue.

ID: 11
Functional Requirement: Provide the option of tracking weight in kg or lbs. (KG by default)
Title: Track in KG or Lbs.
Description: Test if the user can select the weight unit they wish to track their exercises with (KG or Lbs.)
Result: Fail

Test Case 11 is testing if the user can select between KG or Lbs regarding weight units. This test case is given a priority of "Medium" because the main functions within the system would still perform as expected regardless of this test passing or failing. However, ensuring that this test case passes consistently would mean that the application is more appealing on a global basis. This is because the application would appeal more to users outside of Europe, such as American users whose preference would be to track weight in pounds (lbs).

ID: 12
Functional Requirement: Calculate the user's one rep max and present it to them
Title: Calculate 1 rep max
Description: Test if the user can calculate their 1 rep max within the application
Result: Pass

This is testing if the user is able to calculate their one repetition maximum provided they know an amount of weight they can lift for a certain amount of repetitions. This test case is given a priority of "Low". This is because the system would still provide all of the key functionality without this test passing. Additionally, users would not feel a feature was missing if this feature were not included. This is because one rep max calculators are very easily accessible online and there are literally hundreds that users can choose from as long as they have an internet connection.

ID: 13
Functional Requirement: Allow the user to enter personal information (height, weight, age, gender)
Title: Enter personal details
Description: Test if a user can enter their own personal details into the application
Result: Fail

This test case is testing if the user is able to enter in their own personal details within the application. Test Case 13 is checking that the "could have" functional requirement is met – "Allow the user to enter personal information (height, weight, age, gender)". The fact that this test case fails is not a major flaw in the application. This is because

the above details are of little use to any of the features included in the application at its current state. If the application were to be scaled in future and implement further functionality, the functionality described in this test case would be prioritised to give a pass result. It is for this reason that it is given a priority of “Low”.

Project Appraisal

The aim of the project was to produce an Android application which will allow users to log a weightlifting workout in combination with Android Wear functionalities, including motion sensors and heart rate monitors.

The Initial Plan of this project (submitted 31/01/2017) outlines that the following three features will be present in project:

1. Store all workout data in a database
2. Track each workout
3. Record repetitions using an Android Wear smartwatch

Therefore, after further planning and research took place, the project was to be considered a success if the following features are implemented successfully -

1. The number of repetition a user performs is automatically tracked for:
    - Barbell Deadlift
    - Barbell Squat
    - Barbell Bench Press

2. The user’s heart rate is monitored for select exercises

3. The user’s workout is logged and saved within a mobile application including:
    - Exercises completed
    - Sets completed
    - Weight lifted
    - Average heart rate
    - Duration of workout
    - Total volume of weight lifted

All three of the above features have been implemented within the application. It is for that reason that the project can be viewed as an overall success.

The number of repetitions a user performs is tracked automatically via Android Wear for the Barbell Deadlift, Barbell Squat and Barbell Bench Press. As already stated, these exercises were chosen due to the popularity of them and their use within powerlifting competitions. However, following physical testing phases of this project, they also proved to be exercises that required the most stability from a user’s point of view. This meant the automatic tracking of the repetitions via the accelerometer had less noise to deal with and a high level of accuracy was achieved. This also completes the target set in the initial plan of developing a project which can “Record repetitions using an Android Wear smartwatch”.

The Initial Plan states that the auto-rep functionality can be implemented in multiple ways by “tracking distance moved by smart watch” or by “monitoring the spikes in acceleration”. Through the development of this project, the realisation was made that the smart watch has no method of monitoring the distance from the ground. Hence, the method chosen for implementation was monitoring spikes in acceleration. The Initial Plan was a useful resource due to a variety of options being considered in terms of the auto-rep functionality.

Although the accelerometer functionality produces pass results for the relevant test cases (Appendix 13, Appendix 14, Appendix 15), as the user testing shows, it does not meet the non-functional requirement of achieving 90% accuracy for each set when a user is below 5ft 8 or above 6ft. Thus, despite the accelerometer functionality passing all the relevant functional test case criteria, necessary changes would be required to improve the application and allow the project to scale for a greater number of users. This is discussed in the Future Work section of this report.

The user’s heart rate is also monitored for the same three exercises that the auto-rep feature has been implemented with. This has been implemented with the same exercises because the user will most likely always be wearing their Android Wear device when performing these exercises. Therefore, it is most logical to record the user’s heart rate when the Barbell Squat, Barbell Deadlift and Barbell Bench Press exercises are being performed in combination with the auto-rep functionality. In addition to passing the success criteria regarding the heart rate monitor, the heart rate monitor module of my project also passes all of the relevant functional requirement test cases (Appendix 21, Appendix 22, Appendix 23).

The application also provides the user with the capability to record their entire workout. This is the case for whether the user is performing an exercise with auto-rep functionality or an exercise that requires complete user input. This data is then saved locally on the device through the implementation of an SQLite database. Hence, the user’s workout is able to be logged and saved within a mobile application. This indicates that the relevant functional test cases again all produce pass results (Appendix 11, Appendix 12). Moreover, this is implemented in such a way that satisfies the original aim from the Initial Plan – “Store all workout data in a database”.

Ultimately, when comparing the results of this project with the aims and objectives stated in the Initial Plan, as well as the aims in this report, the project can again be considered a success. This is because all of the aims and objectives have been achieved.

Conclusions

The following section of this report will analyse what has been achieved throughout the duration of this project. It will also provide insight into the decisions which were made, why these decisions were made and what better decisions could have been made regarding the software development phase of this project.

In regard to the functional requirements of this project, all of the key functional requirements (i.e. the “must have” requirements) have been met. In addition to that, the initial project aims have also been achieved.

From a wider perspective, Android was an excellent platform to build this project on in terms of Android Mobile. There are a variety of reasons for this. Throughout the development of the project, there were many times where I was unsure how to implement a specific function. However, due to the extensive library that Android provides which goes hand in hand with the very detailed documentation, I was able to overcome every obstacle I came across. Some issues took much longer than others to resolve. Nevertheless, I was always able to find information regarding how to get around the problem somewhere online.

On the other hand, I feel that because Android Wear is still a relatively young module that Android has produced, the documentation was limited. This meant that when I faced any issues regarding the Android Wear module within the application, it was much more difficult to find the answer I wanted. Hence, because the Android Wear documentation was limited, I often found myself searching Git libraries and various other resource websites in order to find the answers I required. Therefore, it took much longer than I thought it would to find the answer I required due to the fact that it is much harder trying to find the correct answer when searching a variety of sources as opposed to the official documentation.

In addition to that, using Android my project presented another issue for myself in regard to XML. The heavy use of XML within Android took me a few weeks to develop a familiarity with it. This is because I am more comfortable developing for the web and using HTML, CSS and JavaScript. Therefore, in the initial weeks of development the designing of things took longer than I initially planned which added to the pressure that the small-time frame was already applying. I do not believe XML was the wrong language to use because the only other option would be to implement a web application within Android. However, I have no experience in doing that. Hence, I did not intend to use that option within my project.

The methodology I chose to use for automatically counting repetitions via Android Wear works well. One of the non-functional requirements for this project is to ensure 90% accuracy with the auto-rep functionality. The results upon testing the system with all three exercises combined with users of different heights demonstrates the system has met this requirement for most users. However, there were extreme instances where the auto-rep functionality lost the required accuracy. These instances occurred for users of a height less than 5 foot 8 or greater than 6 foot. A method of dealing with this issue is outlined in the Future Work section.

I believe that using the accelerometer, which is implemented in almost all Android devices, was the correct choice for this project. In order to track repetitions, the method that has been applied within the system means that the device does not need to know the exact amount of distance that it has travelled. The system only requires the knowledge that the user has travelled through a minimum number of acceleration changes. It is for this reason

that the accelerometer performs to the required level within the application and provides the necessary functionality with users of different heights.

The auto-rep functionality described above is a unique implementation. There is no other application on the Google Play store offering the automatic tracking of repetitions through the use of an Android Wear device and an accelerometer. Implementing a feature that has not been developed in a similar fashion elsewhere, meant that there were very few examples to base this module of code on. Hence, this proved to be a very challenging process and resulted in a much greater understanding of Android by the end of the project. For example, the development process of this application has meant I now have a deeper understanding of Android Sensors and the different functionality offered by Android Wear.

The gesture to initiate the exercise essentially signals the Android Wear device to begin monitoring the built-in accelerometer. Implementing the gesture functionality was the correct decision because it makes the application more accessible. It is more accessible due to the fact that when getting into position it is quicker and more efficient for the user to apply a gesture to the Android Wear device as opposed to having to reach for their Android Mobile device or Android Wear device with their other hand.

The heart rate monitor and the functionality if offers has been a vital piece of this project from the beginning. This is due to one of the aims of the project being to successfully implement the heart rate monitor and the heart rate monitor implementation also being a functional requirement. The decision to implement a heart rate monitor proved to be astute because the development time took less than had been originally assigned to it. Hence, the implementation of a key feature within the project was developed efficiently and resulted in there being extra time to invest elsewhere. In addition to that, the heart rate functionality adds something extra to the application that is not offered elsewhere. There are many Android applications that offer the tracking of user heart rates when performing cardio exercises. Although, there are not any applications on the Google Store that allow a user to monitor their heart rate within a workout journal whilst they are performing weight lifting exercises.

The implementation of the SQLite database was implemented in the time I allotted for this module. As ensuring that user workout data is saved is a key functionality requirement, I am of the opinion that implementing the SQLite database was the correct decision given the short time frame. This is because it was relatively easy to implement and work with due to the documentation and examples provided by Android. From another perspective, I do not believe that implementing a local database was the correct decision in terms of scalability. This will be discussed further in later sections of the report.

Future Work

In regard to future development of the system, there are many ideas which could be implemented. Some of these ideas have been in mind since the initial planning stages of this project. However, some of these ideas have also come to light throughout the development stages of the project. These ideas and the reason they should be implemented in the future will be discussed within this section.

Initially, the primary feature of the application, the auto-rep functionality, should be implemented on more exercises throughout the application. If more time were available, the auto-rep functionality would be implemented on all major compound exercises from the start. This would allow the auto-rep functionality of the application to be more recognisable. At the start of this project, there was belief that the auto-rep functionality could be implemented on more than three exercises. However, given the time frame there was just not enough time to implement auto-rep with a greater number of exercises. This is because each exercise would require significant testing to ensure that a satisfactory level of accuracy would be reached.

Subsequently, an idea for the future is to modify the way that user data is being saved within the application. The implementation of the local SQLite database would be changed to a cloud server to store the data, such as Google's Firebase[7]. This would improve the application because it would ensure that user data is safe. By safe, this is referring to reducing the risk of losing or losing the user device. If the user were to lose their device with the current application installed, all of their workout data would also be lost. However, if a cloud server, such as Firebase, were to be implemented within the application, all the user data would be recoverable onto a new device. In addition to the data being recoverable, it would also allow the user to remove their workout data from being in the wrong hands, if they were to lose their device. This is because the data could be remotely deleted from the server.

Saving user data on to a cloud server would also mean that, if the application were to be scaled and pushed onto different platforms, such as IOS and web applications, the user data would be easily transferable on to alternate devices. Hence, it would make the transition from one device to another seemingly hassle free for the user. A cloud server platform such as Firebase, would also provide the user with the ability to export their workout data. This would allow the user to perform their own analysis on their workout data and come to their own personal conclusions.

Furthermore, a new feature within Android Wear 2.0, which was released in early 2017, means that Android Wear applications can now run individually. This means that an Android Wear application no longer needs to be partnered with an Android Mobile device running the equivalent of that application. Hence, Android Wear applications now have the capability to be stand-alone applications. Upon realising this new feature, the aim would be to develop the system so that it can be run solely on an Android Wear device. This would mean that the user would only have one device to worry about when performing their workout in the gym. Therefore, the equipment required for the user to have on hand within the gym would be even more minimal. However, the Android Mobile version of the system would have to also remain functional in order to take precautions. For

[7] https://firebase.google.com/

example, Android Wear might not grow to the popularity that Google are hoping which would require keeping the options open for this system.

Moreover, the user should be presented with the option to enter their own personal details within the application. These user details would include gender, weight, age and height. Providing the application with the details mentioned would mean that the users could be presented with a more in-depth post workout analysis. For example, the application would be able to provide the user with a relatively accurate number for the calories they have burned during their workout. This is because many calories burned algorithms not only require the user's heart rate but also their height, weight, age and gender.

Adding to the previous point, the addition of user details within the application would also allow the accuracy of the auto-rep feature to be improved further. This is due to the fact that if the user's height is known, the application could build on the auto-rep functionality to allow it to take into account the user's height. For example, if User A is much taller than User B, then User A would have a much greater range of motion when performing an exercise. Hence, User A may find the application less accurate than User B. This is a problem that was considered from the very beginning of this project but became apparent during the user testing phase. Although, if the application were able to take into account the user's height, this issue could be solved. This feature was originally planned to be implemented within the system and is one of the "could have" functional requirements. However, there was not enough time to include this feature.

Additionally, in order to remove further potential errors and implement dynamic programming into the application, the user could be presented with a range of motion test upon the initial launch. The addition of this feature would mean that the user would be asked to perform 10 repetitions of the Barbell Squat, Barbell Deadlift and Barbell Bench Press. The amount of positive and negative acceleration changes for each exercise could then be monitored and stored. These acceleration changes could then be used as parameters within the getAcceleration() function. Hence, the application accuracy would be improved vastly as the positive and negative acceleration change minimum values would be set on a user-by-user basis. This would again solve the issue presented in the user testing phase of this report. Also, this would resolve any issues caused by Android Wear devices having varying accuracy levels in terms of their accelerometers.

In terms of the exercise options within the application, the application would be improved greatly if it offered a cardio option for the user. The cardio functionality would be combined with the heart rate sensor in order to provide the user with an estimate of the calories that they have burned during their cardio session. The addition of this feature within the system would mean that it appeals to a greater audience. This is because the application would become more than just a weight lifting journal. Therefore, users who have an interest in exercising through cardio but not weight lifting would also be able to use the application.

Additionally, the user could have the ability to add their own weight lifting exercises within the application. These exercises would not be able to be implemented with the auto-rep functionality. However, purely from a workout journal perspective, allowing the users to create their own exercises would be an appealing addition to the application. This feature was originally one of the requirements but the relatively small-time frame meant that there was not enough time to implement all of the features. This feature currently being absent from the application would not present any major issues. Therefore, this is a feature to implement in the future.

Ultimately, in the future this application could be integrated with other major Android applications within the fitness industry, such as Fitbit[8] and MyFitnessPal[9]. Both Fitbit and MyFitnessPal provide the user with their own personal food journals. Thus, a user can track the food they eat. This means that the user is essentially tracking their caloric intake. Integrating this system with an application such as the two mentioned here, would mean that the calories burned calculated within the system could be added into the user's food journal. This would mean that the user's caloric intake would be reduced. Implementing the application with major applications such as Fitbit and MyFitnessPal would greatly approve the appeal of the application because food journal applications are extremely popular with Android users.

[8] https://www.fitbit.com/
[9] https://www.myfitnesspal.com/

Reflection

This section of the report will discuss any self-development which has taken place. Specifically, it will provide an insight into what has been learnt personally and which skills have been gained that will be used in future work.

Through the entirety of this project, there was a steep learning curve in terms of over-ambitious goal setting. The key, “must have”, functional requirements took up a lot of the time allotted for the development of the application. This is due to inexperience when developing within Android and the difficulty of the features. This meant that the time allocated for the smaller features within the functional requirements, such as the option for the user to decide between using KG or Lbs as a weight unit, was smaller than planned. Hence, there were occasions throughout this whole process where there was not enough time to implement some of the features that were initially planned. On the other hand, this has not discouraged myself from aiming high in future projects. However, it has enabled me to realise that I must be more realistic in relation to setting goals within a time frame.

Furthermore, there were certain instances throughout this project where I feel I should have performed more research prior to implementing specific features. For example, I am of the opinion that more research should have been performed by myself in regards to the most popular weightlifting exercises (besides Barbell Squat, Barbell Deadlift and Barbell Bench Press). Not carrying out a high level of research in this area meant that I had to go back multiple times to revise the exercises I had included within the application. Thus, I have realised that where there is time, research should always be performed to as high a level as possible. Initially, this will take up more time at the start but it will allow for greater efficiency at later stages of the project.

Moreover, I intentionally left an amount of time as unassigned when planning this project. This was done as a precautionary measure to reduce the impact of any features taking longer than the time that had been allowed. This proved to be an intelligent decision because there were stages throughout the development phase were the time assigned was not enough. This extra time meant that, initially, I did not have to take any time from other tasks in order to cover for the over-time required. Although, the extra time proved not to be enough. Therefore, in future, along with more intelligent planning, I would plan to leave even more unassigned time to deal with the issue of running over time on certain tasks further.

To be more specific in relation to the problems I faced, one of the major issues involved implementing the accelerometer in the manner required. Initially, I attempted to try and find various examples of the implementation I was looking to use. However, there were very few suitable examples. Hence, I ended up wasting a lot of my time attempting to search for examples that were nowhere to be found. I solved my issue by reading the documentation and gaining a much greater understanding of the accelerometer and the Android Wear modules in general. Therefore, I have educated myself to read the documentation on anything that I do not understand prior to trying to find examples of code. This will in turn allow myself to use my time more effectively.

Ultimately, I am of the opinion that in order to improve myself as a developer, I should have prioritised the modules of code I was developing. Although I had a clear hierarchy designed by the functional requirements I wrote, there were times throughout this project where I would develop less important features prior to implementing the key “must have” requirements. Had I run out of time to develop the key features, such as the auto-rep functionality, this would have been the reason for doing so. Therefore, although not prioritising all of my

work in this instance did not have an impact on the outcome, I have realised that in future that the work should be developed on a high-low priority basis.

References

Agile Methodology, 2008. *The Agile Movement.* [Online]
Available at: http://agilemethodology.org/
[Accessed 11 03 2017].

Android.com, 2017. *Intent.* [Online]
Available at: https://developer.android.com/reference/android/content/Intent.html
[Accessed 01 04 2017].

Android.com, 2017. *Parcelable.* [Online]
Available at: https://developer.android.com/reference/android/os/Parcelable.html
[Accessed 01 04 2017].

Android, 2017. *Syncing Data Items.* [Online]
Available at: https://developer.android.com/training/wearables/data-layer/data-items.html
[Accessed 03 04 2017].

Butt, C., 2017. *Formulas And Coefficients That Predict Your 1-Rep Maximum.* [Online]
Available at: http://www.weightrainer.net/training/coefficients.html
[Accessed 02 04 2017].

Git, 2017. *Git -- everything is local.* [Online]
Available at: https://git-scm.com/
[Accessed 21 03 2017].

Helms, E., 2016. *1 Rep Max Calculator.* [Online]
Available at: https://muscleandstrengthpyramids.com/1rm-calculator/
[Accessed 01 04 2017].

Lambert, D., 2016. *Combining Agile Methodologies And Business Architecture.* [Online]
Available at: https://www.batimes.com/articles/combining-agile-methodologies-and-business-architecture.html
[Accessed 01 05 2017].

M&S Team, 2011. *Beyond Sets and Reps.* [Online]
Available at: https://www.muscleandstrength.com/articles/beyond-sets-and-reps-look-at-training-volume.html
[Accessed 25 03 2017].

Mac, B., 2017. *Maximum Load (1RM).* [Online]
Available at: https://www.brianmac.co.uk/maxload.htm
[Accessed 05 04 2017].

Matheus Amarante do Nascimento, E. S. C. F. Y. N., 2007. *Validation of the Brzycki equation for the estimation of the 1-RM in the bench press,* s.l.: 42e.

Mauimauer, 2014. *AndroidWearable-Samples.* [Online]
Available at: https://github.com/mauimauer/AndroidWearable-Samples/blob/master/AgendaData/Application/src/main/java/com/example/android/wearable/agendadata/MainActivity.java
[Accessed 11 02 2017].

Powerlifting, B., 2017. *Powerlifting.* [Online]
Available at: https://www.britishpowerlifting.org/powerlifting
[Accessed 06 03 2017].

Rini, D., 2014. *Compound Exercises Bring Compound Results.* [Online]
Available at: https://www.bodybuilding.com/fun/powerful_workout_exercises.htm
[Accessed 25 03 2017].

Rodgers, P., 2016. *What Is Repetition Maximum and 1RM?.* [Online]
Available at: https://www.verywell.com/what-is-repetition-maximum-and-1rm-3498379
[Accessed 26 03 2017].

SourceTree, 2017. *SourceTree.* [Online]
Available at: https://www.sourcetreeapp.com/
[Accessed 22 03 2017].

TechTarget, 2017. *Definition Sprint.* [Online]
Available at: http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint
[Accessed 15 04 2017].

TestingFreak, 2017. *What is The Waterfall Model.* [Online]
Available at: http://testingfreak.com/waterfall-model-software-testing-advantages-disadvantages-waterfall-model/
[Accessed 16 04 2017].

Thibaudeau, C., 2015. *The 1 Rep Max is Dead.* [Online]
Available at: https://www.t-nation.com/training/1-rep-max-is-dead
[Accessed 26 03 2017].

Tutorials Point, 2017. *Requirement Based Testing.* [Online]
Available at: https://www.tutorialspoint.com/software_testing_dictionary/requirements_based_testing.htm
[Accessed 13 04 2017].

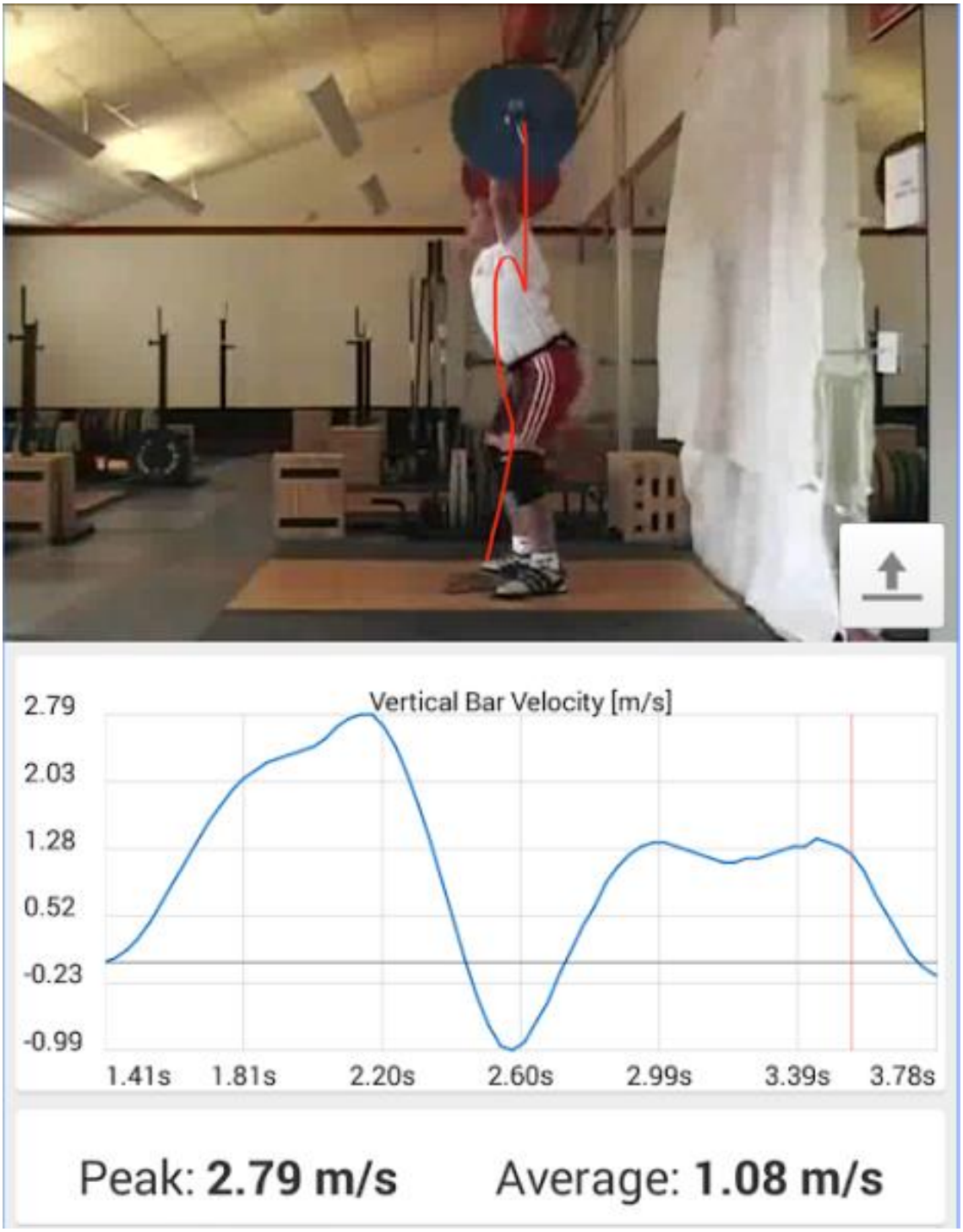
UpGuard, 2017. *Github vs Bitbucket.* [Online]
Available at: https://www.upguard.com/articles/github-vs-bitbucket
[Accessed 03 04 2017].

Waters, K., 2009. *Prioritization Using MoSCoW.* [Online]
Available at: http://www.allaboutagile.com/prioritization-using-moscow/
[Accessed 11 03 2017].

Appendices

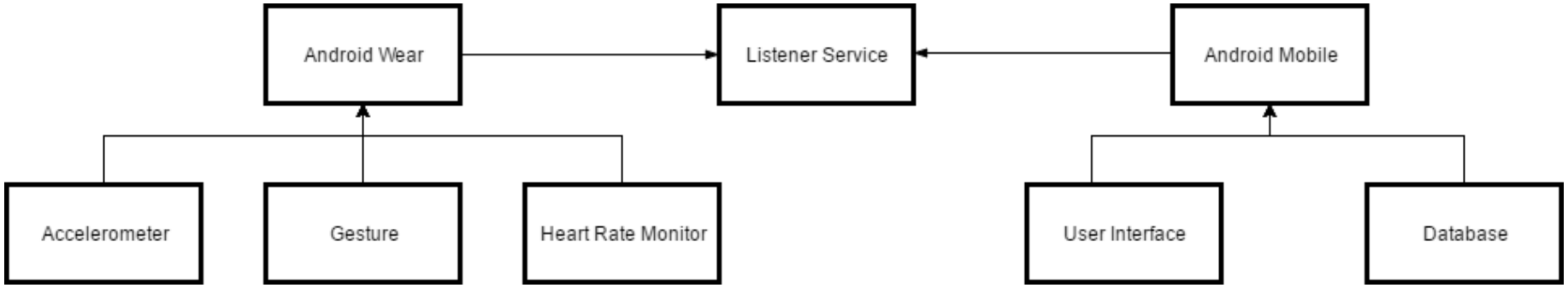
Appendix 1

BarSense application preview



Appendix 2

Diagram displaying the overall design of the system



Appendix 3

Class diagram of the system

Appendix 4

Accelerometer pseudo code

```
getAccelerometer(Exercise) (
        read and set accelerationValue from accelerometer
        read minimumPositive and minimumNegative from data layer
        read count value
        int positiveAcceleration = 0;
        int negativeAcceleration = 0;

        if (accelerationValue > 2) { increase positiveAcceleration by 1; }
        if (accelerationValue < -2) { increase negativeAcceleration by 1; }
        // eliminate small movement anomalies
        if (positiveAcceleration > minimumPositive AND negativeAcceleration >
mimmumNegative) {
                send repetition count to data layer;
                increment repetition count by 1;
                reset positiveCount and negativeCount to 0;
        }

        // attempt to remove noise
        if (absolute value of accelerationValue < 0.01)
                positiveCount = 0;
                negativeCount = 0;
        }
)
```

Appendix 5

Heart rate monitor pseudo code

```
getHeartRate(Array heartBeats) {
        totalHeartBeats = 0;
        for (heartbeat in HeartBeats) {
                totalHeartBeats += heartbeat
        }
        heartRate = totalHeartBeats / size of heartBeats array
        return heartRate;
}
```

Appendix 6

Listener module code

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.round_activity_main);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        googleClient = new GoogleApiClient.Builder(this)
                .addApi(Wearable.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();
    }

    @Override
    public void onStart() {
        super.onStart();
        googleClient.connect();
    }


   public void run() {
        // Construct a DataRequest and send over the data layer
        PutDataMapRequest putDMR = PutDataMapRequest.create(path);
        putDMR.getDataMap().putAll(dataMap);
        PutDataRequest request = putDMR.asPutDataRequest();
        DataApi.DataItemResult result = Wearable.DataApi.putDataItem(googleClient,
request).await();
        if (result.getStatus().isSuccess()) {
            Log.v("SendDataLayer", "DataMap: " + dataMap + " sent successfully to data
layer ");
        }
        else {
            // Log an error
            Log.v("myTag", "ERROR: failed to send DataMap to data layer");
        }
    }
```

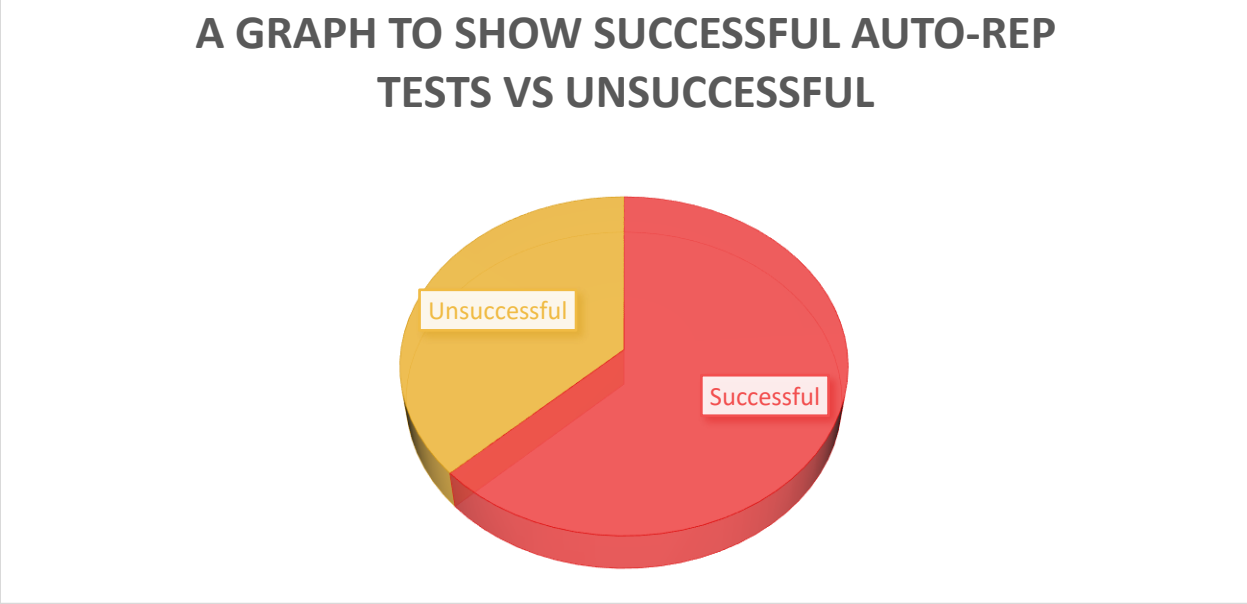
Appendix 7

Broadcast change in data layer code

```
    @Override
    public void onDataChanged(DataEventBuffer dataEvents) {

        DataMap dataMap;
        for (DataEvent event : dataEvents) {

            // Check the data type
            if (event.getType() == DataEvent.TYPE_CHANGED) {
                // Check the data path
                String path = event.getDataItem().getUri().getPath();
                if (path.equals(WEARABLE_DATA_PATH)) {
                    dataMap = DataMapItem.fromDataItem(event.getDataItem()).getDataMap();

                    int dm = dataMap.getInt("repetitions"); // Convert datamap to int
before putting in intent
                    Log.v("ReceiveDataLayer", "DataMap received on mobile: " + dataMap);

                    // Broadcast message to wearable activity for display
                    Intent dataIntent = new Intent();
                    dataIntent.setAction(Intent.ACTION_SEND);
                    dataIntent.putExtra("repetitions", dm);

                    LocalBroadcastManager.getInstance(this).sendBroadcast(dataIntent);

                }
            }
        }
    }
}
```

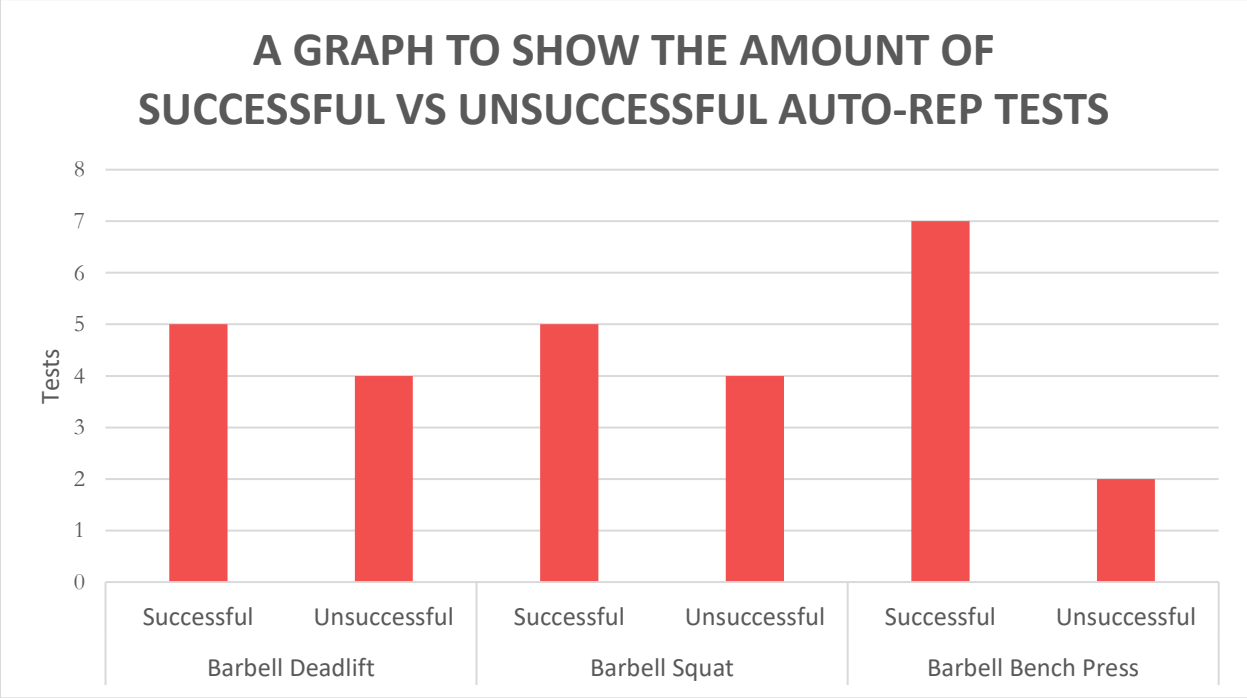
Appendix 8

A graph to show successful auto-rep tests vs unsuccessful



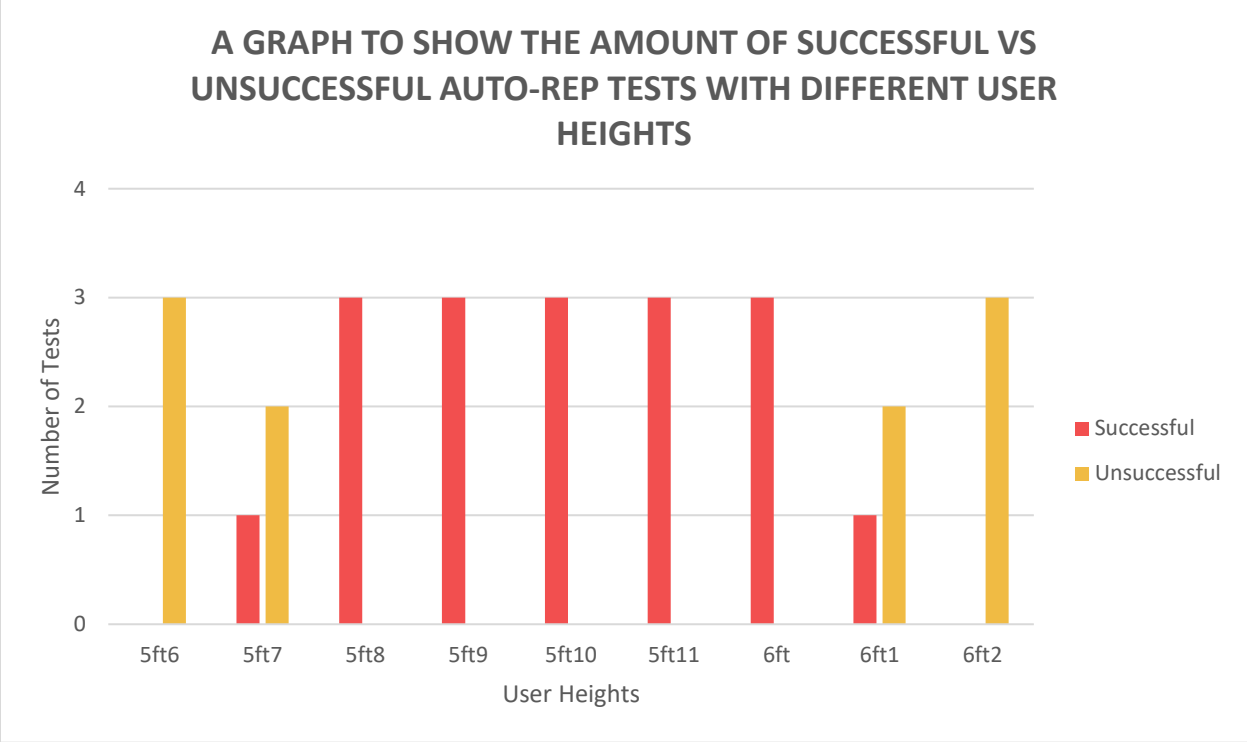
Appendix 9

A graph to show the amount of successful vs unsuccessful auto-rep tests



Appendix 10

A graph to show the amount of successful vs unsuccessful auto-rep tests with different user heights



Appendix 11

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Create workout of selected exercises	**Test ID:** 01
Test Priority (Low/Medium/High): High	
Test Description: Test whether or not a user can create a workout by selecting a list of exercises	
Pre-condition: • The user has the application installed on their Android Mobile device • The application has loaded a selection of exercises for the user to select from	
Dependencies:	
Steps: 1. Press the "Start Workout" button 2. Choose an exercise category 3. Select at least one exercise 4. Press the "Start Workout" button or select the back button and repeat steps 2-4	
Expected Result: The user should be presented with the first exercise they selected with an option to enter in weight lifted and repetitions completed. The user can iterate through all exercises selected by selecting the "next exercise button."	
Actual Result: The user is presented with the first exercise selected with an option to enter in weight lifted and repetitions completed. The user can iterate through all exercises selected by selecting the "next exercise button."	
Pass/Fail: PASS	

Appendix 12

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: User can record workout data	**Test ID:** 02
Test Priority (Low/Medium/High): High	
Test Description: Test whether or not a user can enter their workout data (weight lifted & repetitions complete).	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has selected the exercises they wish to perform from a list of exercises • The user has selected the "Start Workout" button	
Dependencies: • Test case 01 must have passed	
Steps: 1. Press the "+" button alongside inside of the "Weight" section or select the EditText field and enter in desired value 2. Press the "+" button inside of the "Reps" section or select the EditText field and enter in the desired value 3. Press the save button or press the clear button and repeat steps 1-3	
Expected Result: The current activity should update and present the user with a table-like format of the set that they just saved. It will show the user: 1. The set number that they have saved 2. The weight that they lifted in that set 3. The amount of reps that were completed in that set	
Actual Result: The current activity updates and presents the user with a table-like format of the set that they have just marked as complete when pressing save. It includes: 1. The set number that they have saved 2. The weight that they lifted in that set **3.** The amount of reps that were completed in that set	
Pass/Fail: PASS	

Appendix 13

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Test auto-rep functionality for Barbell Squat	**Test ID:** 03
Test Priority (Low/Medium/High): High	
Test Description: The application automatically tracks user repetitions for the Barbell Squat exercises via the Android Wearable smartwatch.	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected a Barbell Squat within their selected exercises list • The user has selected the "Start Workout" button • The use has navigated to the Barbell Squat exercise screen by selecting "Next Exercise" for the required number of times	
Dependencies: • Test case 01 must have passed	
Steps: 1. Select "Yes" when the Android Wear Detected warning appears 2. Once ready to perform exercise, apply the gesture (wrist-rotation) to the Android Wear device or select the begin exercise button on the Android Wear device 3. Perform exercise for desired number of repetitions 4. Apply the same gesture as used in step 2 in order to mark the exercise as complete or select the exercise complete button on the Android Wear device	
Expected Result: The current activity should update the number inside of the Repetition's EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.	
Actual Result: The current activity updates the number inside of the Repetition's EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.	

Pass/Fail: PASS

Appendix 14

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Test auto-rep functionality for Barbell Deadlift	**Test ID:** 03_b
Test Priority (Low/Medium/High): High	
Test Description: The application automatically tracks user repetitions for the Barbell Deadlift exercises via the Android Wearable smartwatch.	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected a Barbell Deadlift within their selected exercises list • The user has selected the "Start Workout" button • The use has navigated to the Barbell Deadlift exercise screen by selecting "Next Exercise" for the required number of times	
Dependencies: • Test case 01 must have passed	
Steps: 5. Select "Yes" when the Android Wear Detected warning appears 6. Once ready to perform exercise, apply the gesture (wrist-rotation) to the Android Wear device or select the begin exercise button on the Android Wear device 7. Perform exercise for desired number of repetitions 8. Apply the same gesture as used in step 2 in order to mark the exercise as complete or select the exercise complete button on the Android Wear device	
Expected Result: The current activity should update the number inside of the Repetition's EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.	

Actual Result: The current activity updates the number inside of the Repetition's EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.
Pass/Fail: PASS

Appendix 15

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Test auto-rep functionality for Barbell Bench Press	**Test ID:** 03_c
Test Priority (Low/Medium/High): High	
Test Description: The application automatically tracks user repetitions for the Barbell Bench Press exercises via the Android Wearable smartwatch.	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected a Barbell Bench Press within their selected exercises list • The user has selected the “Start Workout” button • The use has navigated to the Barbell Bench Press exercise screen by selecting “Next Exercise” for the required number of times	
Dependencies: • Test case 01 must have passed	
Steps: 1. Select “Yes” when the Android Wear Detected warning appears 2. Once ready to perform exercise, apply the gesture (wrist-rotation) to the Android Wear device or select the begin exercise button on the Android Wear device 3. Perform exercise for desired number of repetitions 4. Apply the same gesture as used in step 2 in order to mark the exercise as complete or select the exercise complete button on the Android Wear device	
Expected Result: The current activity should update the number inside of the Repetition’s EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.	

Actual Result: The current activity updates the number inside of the Repetition's EditText field. The number will be updated to the number of repetitions that the Android Wear device registered the user as completing.
Pass/Fail: PASS

Appendix 16

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Test if user workout data saves to SQLite database	**Test ID:** 04
Test Priority (Low/Medium/High): High	
Test Description: The application saves user recorded workout data into an SQLite database stored locally on the user device.	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has selected exercises from the list of exercises provided. • The user has selected the "Start Workout" button • The user has entered in workout data (weight lifted, repetitions completed) for each exercise they selected	
Dependencies: • Test case 01 must have passed	
Steps: 1. On the final selected exercise screen, the user selects the "Complete Workout" button	
Expected Result: The user is presented with a Workout Complete screen. Once the Workout Complete screen is loaded, the user is presented with a notification at the button of the screen displaying "Workout Saved"	
Actual Result: The user is presented with a Workout Complete screen. Once the Workout Complete screen is loaded, the user is presented with a notification at the button of the screen displaying "Workout Saved"	
Pass/Fail: PASS	

Appendix 17

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Test if Android Wear gesture initiates exercise	**Test ID:** 05
Test Priority (Low/Medium/High): Medium	
Test Description: Test whether the "arm-flick" gesture initiates the Android Wear accelerometer. In turn initiating the auto-rep functionality.	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has selected exercises from the list of exercises provided. • The user has selected at least one exercise out of Barbell Squat, Barbell Deadlift and the Barbell Bench Press. • The user has selected the "Start Workout" button • The user navigates to a Barbell Squat, Barbell Deadlift or Barbell Bench Press exercise using the "Next Exercise" button.	
Dependencies: • Test case 01 must have passed	
Steps: 1. Select "Yes" when the Android Wear Detected warning appears 2. Perform "wrist-roll" gesture to signal that you are starting your exercise.	
Expected Result: The Android Wear device updates the screen to show the name of the exercise currently being performed. It also updates the optional button from "Start Exercise" to "Complete Exercise"	
Actual Result: The Android Wear device updates the screen to show the name of the exercise currently being performed. It also updates the optional button from "Start Exercise" to "Complete Exercise"	
Pass/Fail: PASS	

Appendix 18

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: View previous workout test	**Test ID:** 06
Test Priority (Low/Medium/High): High	
Test Description: Test whether the user is able to view the name of exercises performed in previously saved workouts	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has previously completed and saved at least one workout • The user knows the date of their last completed workout	
Dependencies: • Test case 01 must have passed	
Steps: 1. Select the Workout History option from Home screen	
Expected Result: The application will take the user to a new activity. This activity will present a list of previously completed workouts.	
Actual Result: The application takes the user to a new activity. This activity will present a list of previously completed workouts.	
Pass/Fail: PASS	

Appendix 19

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Modify previous workout test	**Test ID:** 07
Test Priority (Low/Medium/High): Low	
Test Description: Test whether the user is able to modify previous workout data (weight lifted, repetitions completed)	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has previously completed and saved at least one workout • The user knows the date of their last completed workout	
Dependencies: • Test case 01 must have passed • Test case 06 must have passed	
Steps: 1. Select the Workout History option from the Home screen 2. Select a previously completed workout 3. Select the “Modify Workout” button 4. Select a specific set (if there is more than one completed set saved) 5. Update the number of repetitions complete or update the amount weight lifted. 6. Select the “Update” button	
Expected Result: The application will present the user with the workout overview screen (using the updated data). It will present the exercises completed and the sets with the weight lifted and repetitions complete.	
Actual Result: There is not a “Modify Workout” button	
Pass/Fail: FAIL	

Appendix 20

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Give post workout analysis	**Test ID:** 08
Test Priority (Low/Medium/High): Medium	
Test Description: Test whether a post workout analysis is presented to the user upon completion	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has selected at least one exercise from the exercise list • The user has completed at least one selected exercise within their workout • The user has chosen the “Start Workout” option	
Dependencies: • Test case 01 must have passed • Test case 02 must have passed	
Steps: 1. Navigate to the last selected exercise by using the “Next Exercise” button 2. Once the last exercise is reached, choose the “Complete Workout” button	
Expected Result: The user should be presented with a workout overview activity which will provide information including the duration of the workout, the exercises completed and the total volume of weight lifted	
Actual Result: The user is presented with a workout overview activity which will provide information including the duration of the workout, the exercises completed and the total volume of weight lifted	
Pass/Fail: PASS	

Appendix 21

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Record average heart rate for Barbell Squat	**Test ID:** 09
Test Priority (Low/Medium/High): High	
Test Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Squat	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected at least the Barbell Squat from the exercise list • The user has chosen the "Start Workout" option	
Dependencies: • Test case 01 must have passed • Test case 05 must have passed	
Steps: 3. Navigate to the Barbell Squat in your selected exercise list by pressing the "Next Exercise" button. 4. Select "Yes" when the Android Wear Detected warning appears 5. Perform "wrist-roll" gesture to signal that you are starting your exercise or select the "Start Exercise" button on the Android Wear device. 6. Perform a Barbell Squat for the desired number of repetitions 7. Perform "wrist-roll" gesture to signal that you are completing your exercise or select the "Exercise Complete" button on the Android Wear device 8. Press the "save" button on the Android Mobile device	
Expected Result: The user should be presented with an average heart rate number for each Barbell Squat performed	
Actual Result: The user is be presented with an average heart rate number for each Barbell Squat set performed	

Pass/Fail: PASS

Appendix 22

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Record average heart rate for Barbell Squat	**Test ID:** 09_b
Test Priority (Low/Medium/High): High	
Test Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Deadlift	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected at least the Barbell Deadlift from the exercise list • The user has chosen the “Start Workout” option	
Dependencies: • Test case 01 must have passed • Test case 05 must have passed	
Steps: 1. Navigate to the Barbell Squat in your selected exercise list by pressing the “Next Exercise” button. 2. Select “Yes” when the Android Wear Detected warning appears 3. Perform “wrist-roll” gesture to signal that you are starting your exercise or select the “Start Exercise” button on the Android Wear device. 4. Perform a Barbell Deadlift for the desired number of repetitions 5. Perform “wrist-roll” gesture to signal that you are completing your exercise or select the “Exercise Complete” button on the Android Wear device 6. Press the “save” button on the Android Mobile device	
Expected Result: The user should be presented with an average heart rate number for each Barbell Deadlift set performed	
Actual Result: The user should be presented with an average heart rate number for each Barbell Deadlift set performed	

Pass/Fail: PASS

Appendix 23

A test case which has been executed on the system

Module Name: Android Wear	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Record average heart rate for Barbell Bench Press	**Test ID:** 09_c
Test Priority (Low/Medium/High): High	
Test Description: Test whether the Android Wear module records an average heart rate while the user is performing a Barbell Bench Press	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device • The user has selected at least the Barbell Bench Press from the exercise list • The user has chosen the “Start Workout” option	
Dependencies: • Test case 01 must have passed • Test case 05 must have passed	
Steps: 1. Navigate to the Barbell Bench Press in your selected exercise list by pressing the “Next Exercise” button. 2. Select “Yes” when the Android Wear Detected warning appears 3. Perform “wrist-roll” gesture to signal that you are starting your exercise or select the “Start Exercise” button on the Android Wear device. 4. Perform a Barbell Bench Press for the desired number of repetitions 5. Perform “wrist-roll” gesture to signal that you are completing your exercise or select the “Exercise Complete” button on the Android Wear device 6. Press the “save” button on the Android Mobile device	
Expected Result: The user should be presented with an average heart rate number for each Barbell Bench Press set performed	

Actual Result: The user is presented with an average heart rate number for each Barbell Bench Press set performed
Pass/Fail: PASS

Appendix 24

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Create custom exercises	**Test ID:** 10
Test Priority (Low/Medium/High): Low	
Test Description: Test whether the user is able to create their own custom exercises to use within a workout	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device	
Dependencies: • Test case 01 must have passed • Test case 05 must have passed	
Steps: 1. Select the “Create Exercises” option 2. Enter in the name of the custom exercise 3. Select the muscles that the exercise targets 4. Press the “Save” button	
Expected Result: The user will be able to see their custom exercise that they have created inside of the muscle category that they have selected as the target muscle.	
Actual Result: There is no “Create Exercise” option for the user to select.	
Pass/Fail: FAIL	

Appendix 25

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Track in KG or Lbs	**Test ID:** 11
Test Priority (Low/Medium/High): Low	
Test Description: Test if the user is able to select the weight unit they wish to track their exercises with (KG or Lbs)	
Pre-condition: • The user has the application installed on their Android Mobile device • The user has the application installed on their Android Wear device	
Dependencies:	
Steps: 5. Select the “Settings” option from the navigation drawer. 6. Press the switch labelled “Weight units”	
Expected Result: The switch should change the weight units being tracked within the system from KG to Lbs. A toast message will appear at the bottom of the screen displaying the current units being tracked.	
Actual Result: There is no settings option within the navigation drawer.	
Pass/Fail: FAIL	

Appendix 26

A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Calculate 1 rep max	**Test ID:** 12
Test Priority (Low/Medium/High): Low	
Test Description: Test if the user is able to calculate their 1 rep max within the application	
Pre-condition: • The user has the application installed on their Android Mobile device • The user knows the weight they can lift for a certain exercise and the number of repetitions they can complete at that weight	
Dependencies:	
Steps: 1. Select the “1RM” option from within the navigation drawer 2. Enter in the weight you can lift in the “Weight” section text field 3. Enter in the repetitions you can manage for that amount of weight in the “Reps” section text field 4. Press the “Calculate 1RM” button	
Expected Result: The activity should update and display the estimated one rep max to user.	
Actual Result: The activity updates and displays the estimated one rep max to the user.	
Pass/Fail: PASS	

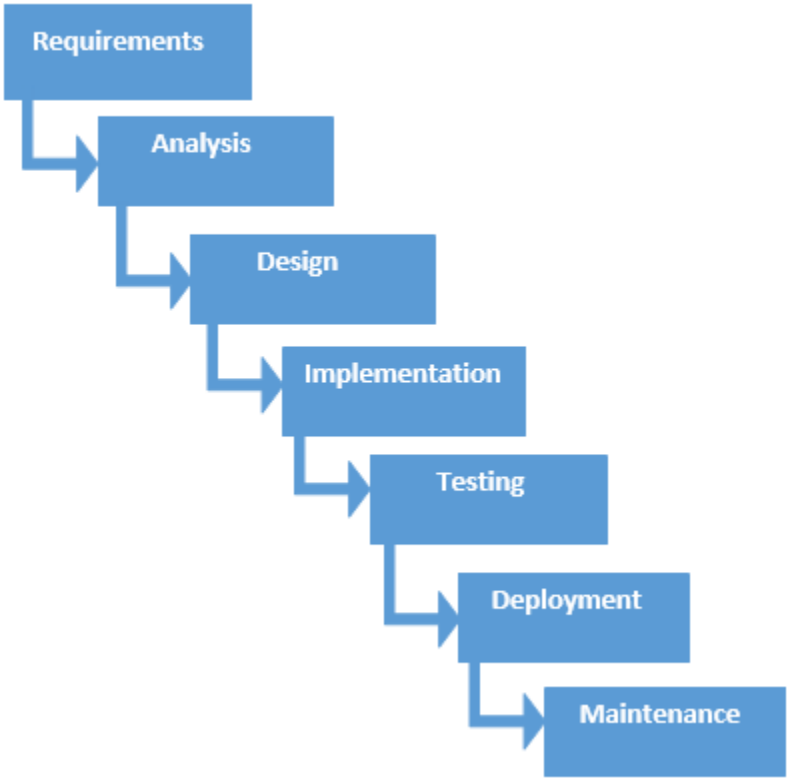
Appendix 27

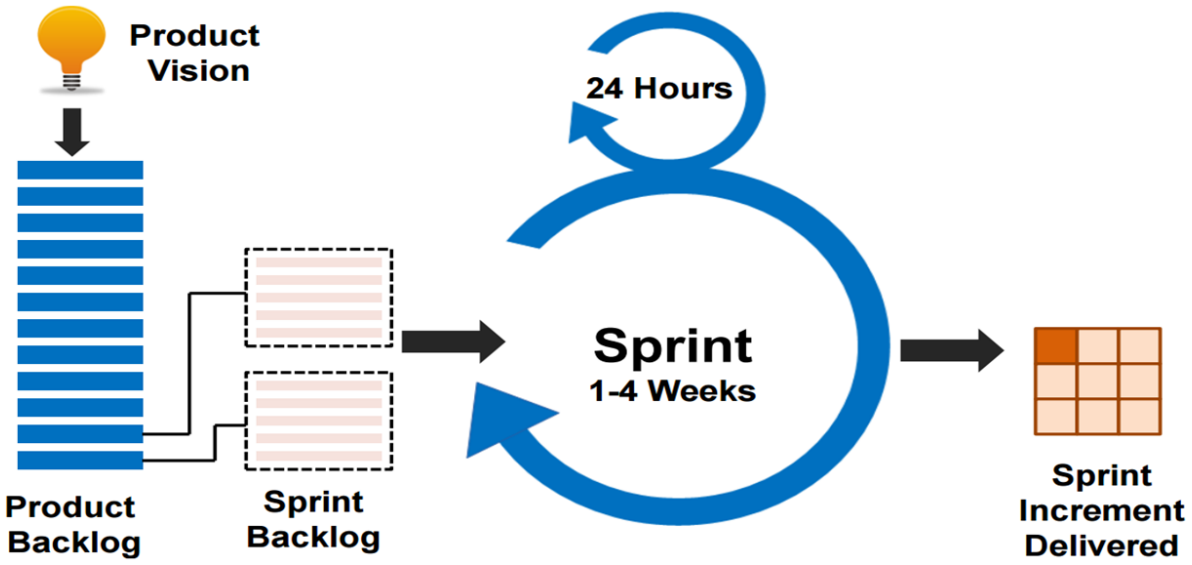
A test case which has been executed on the system

Module Name: Android Mobile	
Designed By: Kieran Mitchell	**Designed On:** 20/03/2017
Executed By: Kieran Mitchell	**Execution Date:** 01/04/2017
Test Title: Enter personal details	**Test ID:** 13
Test Priority (Low/Medium/High): Low	
Test Description: Test if the user is able to enter in their own personal details within the application	
Pre-condition: • The user has the application installed on their Android Mobile device	
Dependencies:	
Steps: 1. Open the application for the first time 2. Select "Yes" to the Provide Details alert dialog 3. Enter your height into the height text field 4. Enter your weight into the weight text field 5. Enter your age into the age text field 6. Select your gender from the gender switch button	
Expected Result: The activity should update and display a message notifying the user that there details have been saved	
Actual Result: There is no "Provide Details" dialogue shown to the user	
Pass/Fail: FAIL	

Appendix 28

A diagram displaying the flow of The Waterfall Method



Appendix 29

A diagram displaying the Agile Model workflow