



6MWTest – Aerobic Fitness App

Using Mobile GPS to Measure Aerobic Fitness in the Field

Scott Morrison

Supervisor: Dr Liane Lewis

Submitted for the Degree of B.Sc. in Software Engineering 2018

Except where explicitly stated all the work in this report, including appendices, is my own and was carried out during my final year. It has not been submitted for assessment in any other context.

I agree to this material being made available in whole or in part to benefit the education of future students.

Signed: _____

Date: 29/03/2018

Abstract

The 6-minute walk test is a simple and robust test used to measure someone's aerobic fitness levels. A traditional 6-minute walk test is usually carried out in a designated space provided by the instructor, where a participant would be asked to attend. This can quickly become costly both in terms of time and resources needed for both the participant and the instructor.

Mobile applications exist that help reduce the costs associated with a 6-minute walk test but they either are tailored towards the instructor to monitor their participants, or they don't provide enough functionality. The aim of the 6MWTest application is to provide the participant with the means of carrying out a 6-minute walk test in their own place and time, whilst also providing them with functionality that allows them to view their previous results at any given time, along with analysis of their results.

The resulting application stores and displays the results from the user's completed tests which are carried out using a proven accurate and repeatable GPS distance tracker. With the aim to promote behaviour change in the user, the application provides the user with the means to compare their results with their previous results and with results from other users of a similar age. Additionally, the application provides the user with notifications for when they complete a test and when they reach a new personal best result. Furthermore, the application provides the user with a personalised profile and login details, giving them the chance to login and carry out a test on any compatible mobile device.

The 6MWTest application developed during this project provides the benefits and results of a traditional 6-minute walk test, without the additional costs and time associated with the traditional test.

Acknowledgements

I would like to first take the opportunity to thank my supervisor, Dr Liane Lewis for her continued guidance, support, and advice throughout this project. The patience and helpfulness shown by Liane will always be remembered.

Secondly, I would like to thank all the participants who volunteered to help me test and evaluate the 6MWTest application. Their input and time helped greatly to conclude the findings in the evaluation stage of this project and shape the 6MWTest application into what it is today.

Lastly, I would like to thank my partner and family for their support, patience, and encouragement during my time at University and throughout this project.

Table of Contents

Chapter 1.	Introduction.....	1
1.1.1	Main Objectives	1
1.2	Outcome	2
1.3	Structure of Report	2
Chapter 2.	Related Work	4
2.1	Analysis of the Functional Capacity – 6 Minute Walk Test Application	4
2.1.1	Features and Description of Functional Capacity – 6 Minute Walk Test.....	4
2.1.2	Positives of Functional Capacity – 6 Minute Walk Test	6
2.1.3	Negatives of Functional Capacity – 6 Minute Walk Test.....	6
2.2	The Traditional 6-minute Walk Test.....	7
2.2.1	The ATS 6-minute Walk Test Guidelines	8
2.2.2	Suitability of the ATS Guidelines for a Mobile Environment.....	9
Chapter 3.	Problem Description and Specification	11
3.1	Problem Overview.....	11
3.2	Requirements Analysis.....	11
3.2.1	Hybrid Mobile Application vs Native Mobile Application	12
3.2.2	The Ionic Framework.....	13
3.3	Development and Design Approach and Timeline	14
3.3.1	Timeline and Approach of Development.....	14
3.3.2	Timeline and Approach of User Interface Design	15
Chapter 4.	System Design	16
4.1	Architecture of Software	16
4.1.1	Benefits of 3-tiered Architecture	16
4.2	Front-end System Design.....	18

4.2.1	MVC in Angular	18
4.2.2	One-page Applications in Angular	19
4.3	Back-end System Design	21
4.4	Database Design.....	21
Chapter 5.	Detailed Design and Implementation.....	23
5.1	Languages and Frameworks Used.....	23
5.1.1	Ionic (Drifty Co., 2018)	23
5.1.2	HTML5 (mozilla, 2018).....	23
5.1.3	Angular (Google, 2018).....	23
5.1.4	SCSS (Sass, 2018)	23
5.1.5	Cordova (The Apache Software Foundation, 2018)	23
5.1.6	ChartJS (ChartJS, 2018).....	24
5.1.7	NodeJS (Node.js Foundation, 2018).....	24
5.1.8	ExpressJS (Node.js Foundation, 2018).....	24
5.1.9	MySQL (Oracle, 2018)	24
5.2	Additional Tools and Technologies	24
5.2.1	Bitbucket (Atlassian, 2018)	24
5.2.2	Trello (Atlassian, 2018)	25
5.3	Development Environment	25
5.3.1	IntelliJ IDEA (Jetbrains, 2018).....	25
5.3.2	PuTTY (PuTTY, 2018).....	25
5.4	User Interface Design.....	25
5.4.1	Initial Wireframes	26
5.4.2	Initial Prototype and JPMorgan Design Meeting.....	26
5.4.3	Final Design	28
5.5	Features of 6MWTest Application.....	29
5.5.1	Registration and Login Security	29

5.5.2	User Profile	31
5.5.3	Age Group Result Comparison	31
5.5.4	Displaying and Comparing Users' Previous Results	32
5.5.5	6-minute Walk Test.....	34
5.5.6	Background Mode.....	39
5.5.7	Network Connectivity Tracker	40
Chapter 6.	Verification and Validation.....	42
6.1	Testing Environment	42
6.2	Feature Testing.....	42
6.3	Debugging	44
Chapter 7.	Evaluation and Results.....	45
7.1	Aims	45
7.2	Participants and Recruitment	45
7.3	Test Procedures	45
7.4	Data Analysis	47
7.4.1	Application Accuracy	47
7.4.2	Test-re-test-reliability	47
7.4.3	Usability and Acceptance	48
7.5	Results	48
7.5.1	Application Accuracy	48
7.5.2	Test-re-test-reliability	49
7.5.3	Usability and Acceptance	50
7.6	Discussion of Results	50
7.6.1	Accuracy and Repeatability	51
7.6.2	Usability.....	52
7.7	Summary of Evaluation.....	52
Chapter 8.	Summary and Conclusion	53

8.1	Summary	53
8.2	Limitations	53
8.3	Future Work	54
8.3.1	Offline Mode.....	54
8.3.2	Noise Alerts in Silent Mode.....	54
8.3.3	Profile Page Design.....	54
8.3.4	iOS Version.....	55
8.3.5	Accessible Data for Researchers.....	55
8.3.6	Pedometer	55
	References.....	56
	Appendix A.....	61
	Evaluation Results.....	61
	Appendix B	62
	Component-Based Usability Questionnaire	62
	Appendix C	64
	Physical Activity Readiness Questionnaire.....	64
	Appendix D	65
	Project Plan	65
	Project Scope Methodology	66
	Appendix E	67
	Justinmind Initial Wireframes	67
	Wireframe Produced From UI/UX JPMorgan Meeting.....	81
	Appendix G.....	82
	Feature Test Cases.....	82
	User Account Tests.....	82
	Profile Page Tests	83
	Network Tracker Tests	84

Safety and Instructions Dialogue Tests	85
Background Mode Tests.....	85
6-minute walk test	87
Previous Results Tests	90
API Test Cases	91
Account.....	91
Results	94
Appendix H.....	99
User Manual	99
Appendix I	103
Build and Run Instructions.....	103
Building and Running the Ionic Application.....	103
Building and Running the NodeJS Application	104

List of Figures

Figure 2-1 – personal info input pre-test.....	5
Figure 2-2 – test completion screen	5
Figure 2-3 – history page showing previous results	5
Figure 2-4 – medical information popup	5
Figure 2-5 – initial/home screen	5
Figure 2-6 – during test screen.....	5
Figure 4-1 – 3-tiered architecture diagram	16
Figure 4-2 – example query using query parameters.....	17
Figure 4-3 – example query without query parameters	17
Figure 4-4 – MVC design pattern interactions	18
Figure 4-5 – MVC in Angular	19
Figure 4-6 – one-page applications vs multi-page applications.....	20
Figure 4-7 – example API route configuration	21
Figure 4-8 – relationship diagram for user and result database tables.....	22
Figure 5-1- initial menu screenshot	26
Figure 5-2 – initial test screenshot	26
Figure 5-3 – initial home screenshot.....	26
Figure 5-4 – initial profile screenshot.....	26
Figure 5-5 – home screen.....	28
Figure 5-6 – log screen	28
Figure 5-7 – profile screen.....	28
Figure 5-8 – different icons and text for age group comparison.....	32
Figure 5-9 – 6MWTest log tab.....	33
Figure 5-10 – countdown timer template snippet	35
Figure 5-11 - custom Angular pipe for formatting countdown timer	35
Figure 5-12 – calculating distance algorithm representation	36
Figure 5-13 – test completion distance calculator	37
Figure 5-14 – new best distance notification	38
Figure 5-15 – diagram showing haversine distance limitation	38
Figure 5-16 – connection re-established popup	41
Figure 5-17 – no connection popup	41
Figure 7-1 - output of Pearson's Correlation for app vs manual results.....	48

Figure 7-2 - output of paired sample t-test statistics for app results 1 vs app results 2	49
Figure 7-3 – output of paired sample t-test for app results 1 vs app results 2	49
Figure 7-4 – output of intra-class correlation between app results 1 and app results 2	49
Figure 7-5 – bar graph showing response results from Component-Based Usability Questionnaire	50
Figure 7-6 – question responses from Component-based Usability Questionnaire.....	50
Figure 0-1– output of normality test for application results and manual results	61
Figure 0-2 – output of normality test for application result set 1 and application result set 2	61

List of Tables

Table 3-1 – initial requirements	12
Table 3-2 – pros and cons of hybrid and native mobile applications	12
Table 4-1 – structure of user database table.....	22
Table 4-2 – structure of result database table	22
Table 5-1 – height and weight unit letter map	31
Table 6-1 – example test case document for reset password feature.....	43
Table 7-1 - correlation coefficient results scale taken from StatsTutor.....	47

List of Abbreviations

API - Application Programming Interface, 13, 14, 15, 16, 18, 19, 21, 42, 44, 91
apk - Android package kit, 43, 104
async - asynchronous, 29, 35
ATS - American Thoracic Society, 7, 8, 9
CBUQ - Component-Based Usability Questionnaire, 46, 48, 50, 52
CIS - Computer and Information Sciences, 16, 24, 104
CPU - Central Processing Unit, 29
CSS - Cascading Style Sheet, 23
FC - Functional Capacity 6 Minute Walk Test, 4, 5, 6, 7, 32, 52
GPS - Global Positioning System, 1, 2, 4, 13, 14, 35, 36, 39, 40, 42, 43, 45, 50, 51, 52, 53, 54, 55, 85, 86
HTML - HyperText Markup Language, 23
HTTP - Hypertext Transfer Protocol, 18
IDE - Integrated Development Environment, 25
JSON - JavaScript Object Notation, 21, 30
jwt - jsonwebtoken, 30
MVC - Model View Controller, 17, 18
PARQ - Physical Activity Readiness Questionnaire, 45, 64
REST - Representational State Transfer, 15, 16, 21, 42
SCSS - Sass Cascading Style Sheet, 23
SQL - Structured Query Language, 16, 24, 55
SSH - Secure Shell, 25
sync - synchronous, 29
TAM - Technology Acceptance Model, 46
UE - IntelliJ IDEA Ultimate Edition, 25
UI - User Interface, 15, 16, 26, 27, 81
URL - Uniform Resource Locator, 21, 23
UX - User Experience, 15, 27, 81

Chapter 1. Introduction

The 6-minute walk test is a well-established and simple test that allows someone's aerobic fitness levels to be measured (American Thoracic Society, 2002). The test is a derivation of the 12-minute walk test which was developed by Cooper (KH, 1986) to evaluate the fitness levels of healthy individuals. Cooper's 12-minute walk test was then re-evaluated to accommodate those with respiratory diseases, since 12-minutes of walking could be too strenuous for non-healthy individuals. This re-evaluation is where the 6-minute walk test was derived from.

A traditional 6-minute walk test is usually carried out in a designated space provided by the instructor, where a participant would be asked to attend. Participants walk for 6 minutes between two marked distances until the instructor tells them to stop. The results of the test are then calculated and used as a basis for determining the participant's fitness levels.

Although the 6-minute walk test doesn't require many resources to carry out, it can still be costly in terms of time and resources for the instructor, and requires participants to attend a designated place at a pre-scheduled time. To reduce the burden on the participant and save costs, a mobile version of the 6-minute walk test was proposed.

1.1.1 Main Objectives

The aim of this project was to create a mobile application that could mimic the results obtained from a traditional 6-minute walk test.

Several high-level objectives were set out to achieve this aim:

- Develop a mobile application that tracks a user's distance covered during a 6-minute walk test, without the need for an instructor to be present
- Provide the user of the application with a personalised profile which allows the user to enter useful information which could be used for future research purposes
- Allow the application to work as a motivational tool to promote behaviour change in the user by showing comparisons of their results with those of other users of a similar age, as well as with their own previous results
- Show comparisons of results in both textual and graphical formats
- Ensure the user interface is kept minimal, simplistic, and easy to use in order to produce a good overall user experience

- Evaluate the accuracy and repeatability in terms of distance tracking for a mobile version of the 6-minute walk test

1.2 Outcome

The application developed during this project provides the user with the means of carrying out an accurate 6-minute walk test, in their own place and time – making it more convenient for both user and researcher. The application has been developed using the Ionic framework (Drifty Co., 2018), which allows portability for both Android and iOS, with a primary focus on Android for this project.

Utilising GPS, the application tracks and calculates the distance covered by the user over the course of 6 minutes. The accuracy of the GPS distance tracking within the application has been evaluated and tested numerous times by individuals independent of the project and has been deemed statistically acceptable (see sections 7.5.1 and 7.5.2).

With a full login system which requires an email and password, the user can update and view their personal details at any time and retrieve their details regardless of any install/uninstall of the application on a device. Using notifications, textual feedback, and graphical feedback, the application allows the user to see clearly when their results have progressed, backtracked, or stagnated, and gives the user the opportunity to view all their previous results. By allowing the user to track their progress and view their previous results, the application aims to promote behaviour change in the user. Furthermore, the overall layout and usability of the application has been given positive feedback by independent users as seen by the results of a Component-Based Usability Questionnaire (see section 7.5.3).

In its current state, the application is ready to be used in the field for carrying out accurate mobile 6-minute walk tests.

1.3 Structure of Report

Chapter 2 will discuss relevant existing applications as well as covering the traditional 6-minute walk test in more detail. Chapter 3 will look at specifics on the problem defined, objectives that were set out, justification for choosing a non-native platform over a native platform, and discuss the development method used during this project. Following on, chapter 4 will detail the architecture and design of each aspect of the system. In chapter 5, specific implementation and design details will be discussed, including how the GPS calculation algorithm works. Furthermore, chapter 5 will discuss which languages,

frameworks, and tools were used over the course of this project. Chapter 6 and 7 will focus on how the application was tested and evaluated respectively, with chapter 7 providing results on the accuracy of the distance calculator, the repeatability of the application, and the usability of the application. Finally, in chapter 8, the report will be summarised, limitations will be discussed, and future enhancements will be proposed.

Chapter 2. Related Work

On researching work relating to this project, the main focus was on existing applications that provided the same or at least similar functionality to what this project aimed to achieve. Since this project was tailored towards running on an Android device, the emphasis on the search was on the Google play store. There is numerous running, walking and general fitness applications available but the market targeting specific 6-minute walk test applications was found to be slim. Only one application was identified on the Google Play Store, the ‘Functional Capacity – 6 Minute Walk Test’ application (FC) (Solutions, 2015).

There were other 6-minute walk test style applications found during researching related work such as Stefano Picciolo’s Six-Min Walk Test application (Picciolo, 2013) and MedApps S.R.L’s 6 Minutes Walking Test application (S.R.L, 2016) but these were tailored towards a physician or instructor using them to monitor participants rather than for the participants to carry out a test themselves. Therefore, in the next section we will focus on the FC application, provide some details about this application, and discuss limitations that will be addressed with the 6MWTest app developed in this project.

2.1 Analysis of the Functional Capacity – 6 Minute Walk Test Application

2.1.1 Features and Description of Functional Capacity – 6 Minute Walk Test

FC achieves similar functionality to that of the 6MWTest application developed as part of this project; users can carry out a 6-minute walk test with a timer shown and their distance tracked by GPS. Furthermore, users are able to enter some basic information about themselves, and are able to view their previous results. Additionally, FC provides the following features:

- Users are given a predicted walking distance based on the data entered about themselves
- If users are falling below their predicted walking distance then they are given advice that could potentially help with their results
- Safety and instructions are shown to the user before carrying out any test
- On completion of a test a heart icon with a tick inside is shown. The heart is coloured conditionally based on the result of the test – blue for good and grey for poor
- FC continues to run whilst in the background (although this is not apparent)

The overall look and feel of FC is simple and consistent, without many complicated transitions or interactions for the user. The following screenshots show the look, feel, and features of FC:

Note: the initial evaluation of FC was carried out before these screenshots were taken. During the initial evaluation, I forgot to take screenshots before uninstalling the application. I reinstalled it and carried out two tests, one for good functional capacity and one for bad functionality capacity in order to be able to view the different icon results for these screenshots.

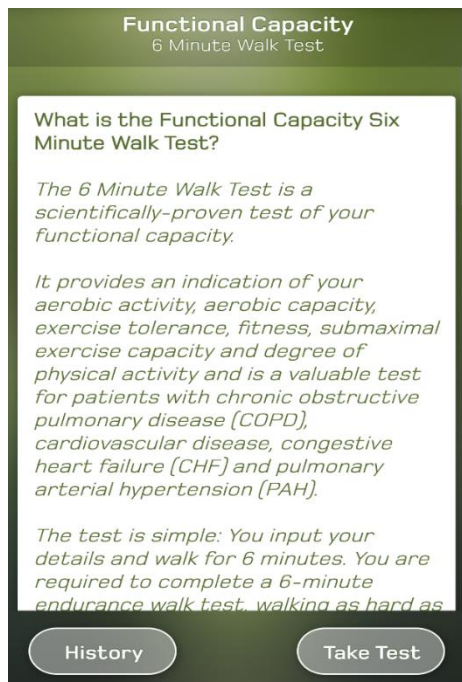


Figure 2-5 – initial/home screen

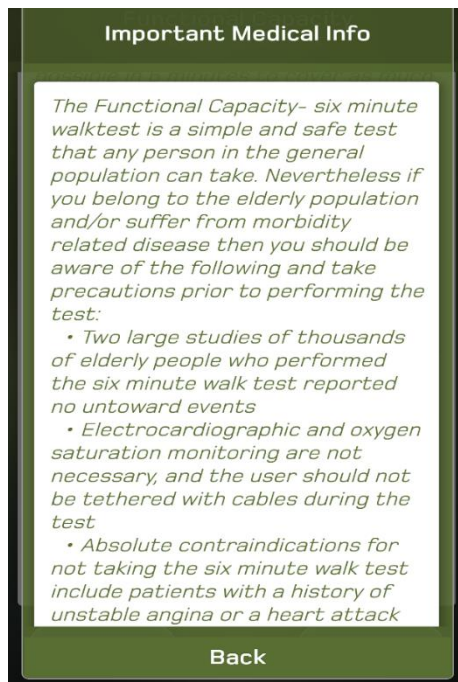


Figure 2-4 – medical information popup

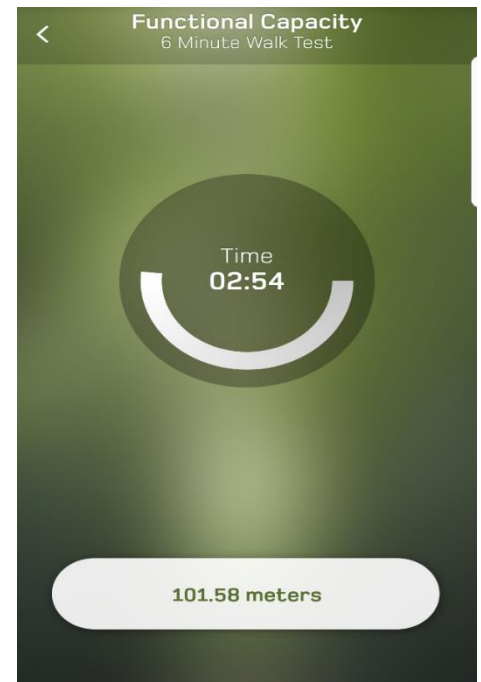


Figure 2-6 – during test screen

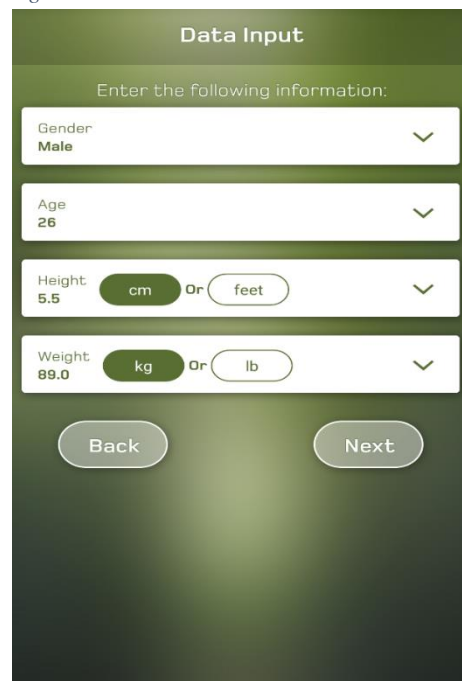


Figure 2-1 – personal info input pre-test
Scott Morrison

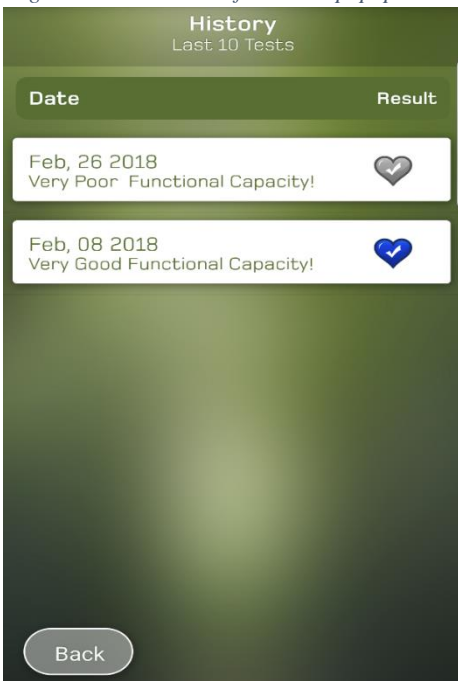


Figure 2-3 – history page showing previous results

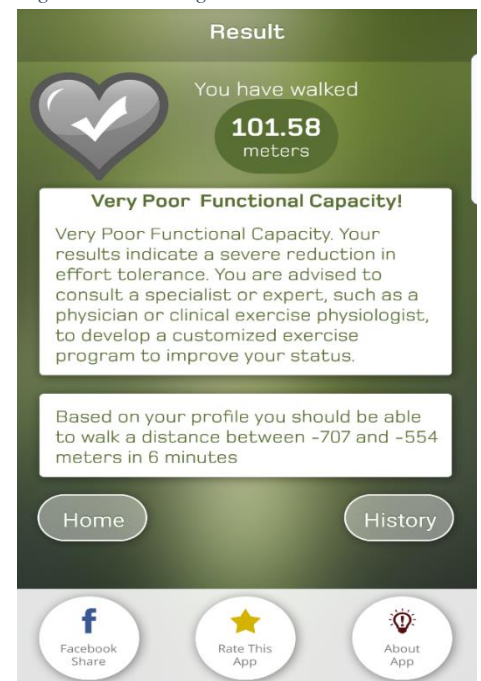


Figure 2-2 – test completion screen

2.1.2 Positives of Functional Capacity – 6 Minute Walk Test

As can be seen in the figures above, FC is well laid out and simply designed. Each of the buttons is understandable and it does not take a lot of effort to get used to using the application. The distance calculator in the application seems to be fairly accurate, although no thorough evaluation was carried out on this part. Whilst performing a 6-minute walk test, FC will show the user the metres they've travelled thus far (see Figure 2-6), giving the user some information about their performance as they carry out the test. Predicted walking distance is a nice feature implemented in FC (see Figure 2-2). It gives the user a target to work towards on their next test and allows the user to see how their actual result compared to their predicted result. FC also allows the user to view information about their previous results, although this is limited to the last ten results only (see Figure 2-3).

The features mentioned above are important to the 6-minute walk test, but FC also provides some useful information for the user, as can be seen in Figure 2-5 and Figure 2-4. The information provided is short and concise enough for the user to read quickly, but also covers the main points of the test and additional medical information required.

2.1.3 Negatives of Functional Capacity – 6 Minute Walk Test

Although FC is simply and cleanly laid out, the choice of colours for the heart icon is confusing. On completion of a test, the heart icon next to the result will either be grey for poor performance, or blue for good performance, making it not very apparent on initial look which colour indicates a good result or a poor result. This can be seen in Figure 2-3 where there is one result saved for a poor performance and one for a good performance. It would be more meaningful to use standard colours for good such as green, and possibly a red or amber for poor. Or make use of different icons such as a tick for good and a cross for poor.

FC provides the user with the ability to enter their details before performing a test (see Figure 2-1). The negative of this feature is that the user is prompted to do this before each test rather than allowing the user to enter the data once and edit it in the future if needed. FC also doesn't store the user's personal data or their results permanently, meaning that if you were to uninstall then install the app again, or install the app on a different device, you have no means of logging in and retrieving your previous results or your profile data.

During a test, FC continues to run even when the application is put in the background or the phone is locked but there is no way of knowing that FC is still running in the background as there is no notification given to the user to indicate this. There is also no way for a test to be

stopped after it has been started. The only way to finish a test is to allow the timer to run down or to close the application completely.

The use of granular permissions is used in FC, meaning that there are no permissions required on installing the application and these are requested as and when they are needed. However, to activate location services for FC the user is first prompted to allow this service and is then taken to the Android settings feature to do so, leaving the application in the process. This could be sped up and made easier for the user by prompting the user to allow FC to change the settings itself, without the need for the user to leave the application and change their settings.

Previous results are shown in FC but as mentioned above, it is limited to the last 10 results only. It is also not very easy for the user to see what the difference was between the previous results without having to click on each one individually and compare the results between each of the tests themselves. Because of this, it is not quite as apparent to the user if their results are progressing in a positive manner or not. Each previous result also shows a date that the test was carried out but there is no way to distinguish between two tests that were carried out on the same day.

2.2 The Traditional 6-minute Walk Test

As well as evaluating applications relating to this project, the traditional 6-minute walk test was also reviewed.

According to the American Thoracic Society (ATS) (American Thoracic Society, 2002), the 6-minute walk test is “a practical simple test that requires a 100-ft hallway but no exercise equipment or advanced training for technicians.”

Participants of the test walk for 6 minutes on a flat surface of a designated length, going back and forth between two marked distances. The instructor of the test will monitor and record how the participant responds to the test, as well as their overall distance covered. The aspects of the participant’s responses that are of interest and the detailed steps of the test carried out are discussed below in section 2.2.1.

The test is predominantly used on people with some form of disease or illness that affects their respiratory system and/or who are unable to carry out long durations of physical activity. The results of the test allow the assessment of a person’s submaximal level of functional capacity. Repeated results are then used to assess the person’s progress in their

functional capacity with possibly one test pre-treatment and one test post-treatment, for example.

The 6-minute walk test has been deemed the “test of choice when using a functional walk test for clinical or research purposes” (Solway S, 2001).

The following description of the test procedure is based on the guidelines provided by the ATS.

2.2.1 The ATS 6-minute Walk Test Guidelines

The “Measurements” section of the ATS guidelines includes steps for measuring the participant’s pulse, blood pressure, and pulse oximetry. The section also includes that the participant’s baseline dyspnea (their level of shortness of breath) should be rated and overall fatigue measured before carrying out a test. For the purpose of this evaluation the steps mentioned have been retracted as they were not part of the requirements or scope of this project. Although the steps are potentially feasible to implement on a mobile device, the requirements of this project, with regards to the 6-minute walk test itself, were set out to solely be able to carry out the actual distance tracking and calculation section of the 6-minute walk test on a mobile device.

For concision and clarity, the following steps are shortened versions of the steps found in the ATS guidelines. No details that would alter the steps have been removed:

1. Tests should be performed around the same time of day
2. A “warm-up” period before the test should not be performed
3. A distance of 50 metres should be measured out in a hall with clearly marked start and end points being placed
4. Participants should be instructed to walk between the marked points
5. Instructors should count the number of times the participant walks back and forth between the two marked points and keep track of the number of times travelled
6. The timer should begin when the participant starts to walk
7. During the walk, the following should be considered and carried out:
 - a. Standard phrases of encouragement should be used (no other phrases should be used):
 - i. After 1 minute: “You are doing well. You have 5 minutes to go.”
 - ii. After 2 minutes: “Keep up the good work. You have 4 minutes to go.”

- iii. Halfway through the test: “You are doing well. You are halfway done.”
- iv. 2 minutes remaining: “Keep up the good work. You have only 2 minutes left.”
- v. 1-minute remaining: “You are doing well. You have only 1 minute to go.”
- vi. If the participant stops at any time: “You can lean against the wall if you would like; then continue walking whenever you feel able.”
- vii. 15 seconds from completion: “In a moment I’m going to tell you to stop. When I do, just stop right where you are and I will come to you.”
- b. If the participant stops and can’t continue for any reason then stop the walk and note the reason for stopping, the distance travelled and the time taken.
- 8. On completion of the walk, record the distance travelled, rounding to the nearest metre
- 9. Congratulate the participant on their effort and offer a refreshment

2.2.2 Suitability of the ATS Guidelines for a Mobile Environment

Steps 1 and 2 of the above are instructional and can easily be presented to the user by having some form of pre-test instructions/guidelines page or dialogue within a mobile application.

Step 3 could be carried out in the way it is described above by utilising the phone’s accelerometer to work out when a user has started walking but it is more feasible and accurate to switch this step around to suit a mobile device. That is, the user should start walking when the timer begins. This instruction for when to start walking can again be made clear in some form of dialogue or page.

Step 4 is when the walking is taking place. The guidelines for words of encouragement could be implemented in the forms of alerts or notifications but the issue here is that this may distract the user during their walk and have an impact on the results. When carrying out a traditional test, the encouragement is coming from the instructor whereas with a mobile version the participant may have their phone in their pocket and would then have to retrieve the phone and possibly unlock it to view the notification/alert. This could potentially cause the user to stop walking whilst they view the notification. Part b of step 4 is certainly suitable for a mobile application. The mobile version could have a way to stop the test and if the test is stopped before the full 6-minutes then the relevant data could be stored.

Overall calculation and recording of the distance travelled in step 5 could be carried out by a chosen calculation algorithm and using some storage solution to record the distance calculated. Storage could either be completed on the phone locally or in some external storage solution.

Step 6 could again be shown using an alert or notification and has room for customisation in a mobile application, for instance showing if the user had progressed from their previous results, or what their result was for that specific test.

Overall a mobile version of the 6-minute walk test is suitable as a replacement to the traditional test albeit with possibly some minor adjustments. The adjustments that would potentially be made aren't enough for the test to be deemed incomplete and even with the adjustments in place, the mobile version should yield the same results.

Chapter 3. Problem Description and Specification

This chapter will cover the problems to be considered for this project and the initial requirements gathered. The chosen development platform will be briefly discussed before ending with discussing the development method used throughout this project.

3.1 Problem Overview

Firstly, the biggest challenge was to ensure the accuracy of the distance tracker within the application is high enough to be able to mimic the results of a traditional 6-minute walk test. Secondly, the application developed should allow the user to easily track their test results, view any of their previous results, and be able to clearly see any progress they have made. This part of the problem should also address the need for promoting behaviour change in the user by making the user aware of their progress through textual feedback, graphical feedback, and additional means such as local notifications. Furthermore, the application layout should be simple and consistent, with a clear understanding on what the application's features are and how to use them.

By solving the above challenges, the application developed can be used as a replacement to the traditional 6-minute walk test.

3.2 Requirements Analysis

Requirements for this project were set out by the supervisor, Dr Liane Lewis, who was also acting as the client. The supervisor's plan is to use the 6MWTest application developed to aid in her research. Therefore, as well as supervising the project, the supervisor was also the main source of information when gathering requirements and making decisions. The original requirements are shown below in Table 3-1.

Requirement	Further Details
Provide the user with ability to create an account and have a modifiable profile page.	Profile should include: Full name, date of birth, height (in m or feet, based on users preference), weight (in kg or lb based on users preference) and gender.
Provide the user with a timer that counts down from 6 minutes.	Alarm when the time is up.
Provide functionality that accurately tracks the user's distance covered during the 6 minutes walked.	Should work indoors and outdoors.
Provide the user with instructions on how to best carry out a test.	Sample instructions were provided.

Display test results after completion of the 6 minutes- results expressed as distance (meters) walked in 6 minutes	Show result immediately on test finishing. User should be able to then see how this result compares to previous results (graph). Limit number of results in graph to not oversaturate data.
Provide the user with ability to view their previous results at any time through independent feature.	Textual and graphical formats to be displayed.
Provide the user with safety instructions for when carrying out a test.	Sample instructions were provided.
Provide the user with a menu of well-defined and easy to understand features.	Menu should include: Profile, instructions, safety, test, results/progress and possibly a map option.

Table 3-1 – initial requirements

Due to the agile nature of the development approach (see section 3.3), these requirements were simply a starting point and were refined continuously over the course of developing the application. The refinements and implementation of each requirement is discussed in detail in Chapter 5.

The mobile development platform for the 6MWTest application was left as a decision to the developer. The main decision was whether to create a hybrid mobile application or a native mobile application.

3.2.1 Hybrid Mobile Application vs Native Mobile Application

There are pros and cons for both hybrid applications and native applications as shown in Table 3-2.

Hybrid	Native
Code can be easily reused on multiple platforms with minimal code changes.	Code is specific to the native platform e.g. Android and needs to be re-written for each new native platform.
Less of a learning curve as the focus is on one language for multiple platforms.	Steeper learning curve as developer needs to know intricacies of multiple languages such as Swift, and Java for Android.
Less access to features on native device which could be problematic for certain application needs (although usually a library is provided to allow access to these features).	Full access to platform targeted by the native language without the need for additional plugins.
Quicker development cycle, saving time and money due to most hybrid platforms being web-based. No need for additional software products – only browser and favourite text editor as a basic setup.	Longer and more costly development requiring specialised software platform and sometimes specific hardware (iOS development is feasible but not recommended on Windows).

Table 3-2 – pros and cons of hybrid and native mobile applications

3.2.2 The Ionic Framework

It was decided that the best approach would be to create a hybrid mobile application. The framework used for this task was the Ionic framework.

The reason for choosing a hybrid approach and using the Ionic framework was primarily due to the portability provided by hybrid applications. Using Ionic, a future enhancement could be to create an iOS version of the application, which would be achievable with minimal changes to the codebase. If a native approach was chosen then this future enhancement would entail developing a whole new application.

Ionic has a large online community and is a well-documented framework. As mentioned above, most hybrid frameworks have a library of tools and plugins that can be used and Ionic is no different. The core library is named Ionic Native and it provides developers with packages that can be imported in to your codebase to enable or help the enablement of a device's features. The main feature to be considered was accessing the device's GPS location. On navigating the Ionic Native documentation, the Ionic Native Geolocation plugin (Drifty Co., 2018) was found to be well-suited for the needs of this project.

The Geolocation plugin, the Diagnostic plugin (Drifty Co., 2018) and the Location Accuracy plugin (Drifty Co., 2018) were some features within Ionic Native that were identified to be important for the development of the 6MWTest application.

The Geolocation plugin provides an API which allows you to retrieve and track a user's location. The API is based on the W3C Geolocation API Specification (W3C, 2018) and provides two methods for retrieving a user's location. Firstly, it provides the *getCurrentPosition()* method which returns the device's current position at that point and time. Secondly, it provides the *watchPosition()* method which when called will continuously return the device's position every time the position changes. Both these methods accept an object as a parameter which holds different configuration options:

- **maximumAge (number):** a positive value which indicates how long to hold on to a cached position. If set to 0 a new position will be returned every time the device's location is requested. If set to infinity then the device must return a cached position regardless of its age.
- **timeout (number):** a positive long value indicating the maximum length of time a device can take to return a position.

- `enableHighAccuracy` (boolean): if set to *true* then the device will use all available ways to gather the location regardless of power consumption. If set to *false* then the device will use less power and save resources which will result in a less accurate location.

The return type of both methods is another object which contains information about the device's location as well as the time at which the location was received. The location includes the latitude, and longitude values as well as the perceived accuracy of the location in metres. The returned object also includes information on the altitude, the altitude accuracy, the direction in which the device is travelling, and the velocity of which the device is travelling at.

The input and output of the above methods provided by the API was adequate to use for accessing the device's GPS location.

The Diagnostic plugin allows one to check if one can retrieve the device's GPS location as well as checking if they are able to retrieve the GPS location using high accuracy. The Location Accuracy plugin allows one to check whether or not the high accuracy setting has been enabled for the application and if not, allows the setting to be changed without making the user leave the application.

The Geolocation API paired with the Diagnostic plugin and the Location Accuracy plugin, as well as the additional resources available through Ionic Native, meant that the Ionic framework could be used confidently as the choice for developing the 6MWTest application.

3.3 Development and Design Approach and Timeline

3.3.1 Timeline and Approach of Development

The chosen development approach for this project was to adopt agile methods such as prototype development, weekly sprints, and task boards to keep track of the progress of the features to be or that have been implemented.

Sticking to the initial project plan (see Appendix D), the weeks leading up to the end of semester one consisted of analysis, requirements gathering, and setting up the initial database tables. Prior to the project poster day, a prototype was worked on using similar methods as described for the main implementation below. This meant that semester two commenced and the application was in the prototype stage and ready to have more features added.

For the remainder of semester two, weekly meetings with the supervisor were scheduled to demonstrate and discuss the latest features that had been implemented and to discuss any problems or setbacks. During these meetings, the following week's features were decided. On the start of each new feature a branch was created on the Bitbucket repo, branching off the main development branch. On completion of each feature, the application was tested using the below test cases and on success, the feature branch was merged back in to the main development branch. A Trello board was used to keep track of what features were outstanding, in progress, and complete.

3.3.2 Timeline and Approach of User Interface Design

On top of iterating through the implementation of features, the layout and design of the user interface was considered. A preliminary design was created for the sake of the poster day prototype and in semester two, the design was looked at again and changed. The design was changed with the help of two UI (User Interface)/UX (User Experience) Analysts from J.P. Morgan whom had volunteered their time to help with the project. Around mid-January we met and went over the design, coming up with a better overall layout. They were then consulted at various stages of the development process to gather feedback on the layout.

Development of the application was complete in mid-February which is when the evaluation stage of the project started. The process during the evaluation stage is discussed below in the Evaluation and Results section.

Chapter 4. System Design

4.1 Architecture of Software

The architecture of the 6MWTest application is a three-tier web-application:

- Tier 1: The front-end of the application.
- Tier 2: The back-end of the application.
- Tier 3: The database used in the application.

The front-end of the application, which is where the Ionic framework lies, is what the user sees and interacts with. This tier calls upon tier 2 to utilise the defined REST API, retrieving login and account details, users' results, and comparison statistics.

The back-end of the application has been implemented in NodeJS. This tier produces the REST API which is called by the front-end. The back-end tier makes a connection with tier 3, where the database lies, and queries the database to retrieve required data.

Tier 3 consists of the MySQL database, as provided by the Computer and Information Sciences (CIS) department. Handling the database structure and fields was done using phpMyAdmin, which provides a web-based UI showing the contents and structure of one's database schema and tables. phpMyAdmin is also provided by the CIS department.

4.1.1 Benefits of 3-tiered Architecture

A tiered architecture provides clear, definitive meaning to each section of the system, what its purpose is, and how it should communicate with the others. Figure 4-1 shows a diagram representing the communication of the architecture.

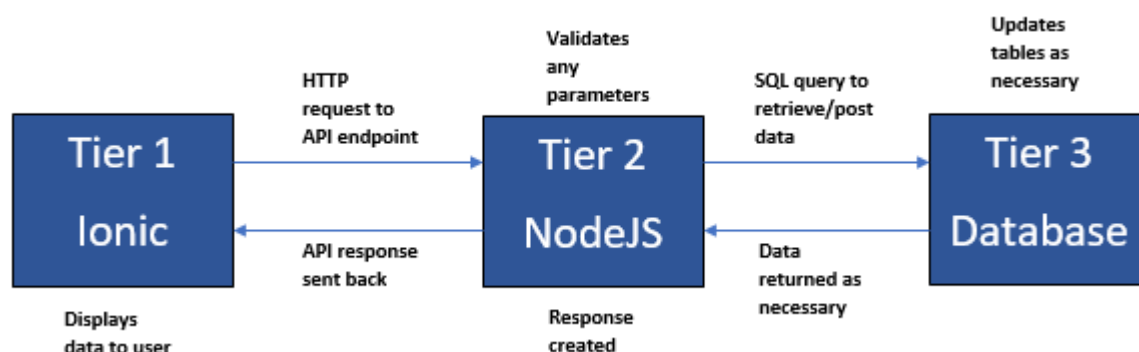


Figure 4-1 – 3-tiered architecture diagram

Having a tiered architecture also ensures there's a level of security between the requests coming from the front-end to what is being saved in the database. On posting a request, the 6MWTest back-end uses different techniques for ensuring the data being passed from the front-end is non-malicious before saving any data to the database. All back-end queries utilise query parameters and strip unwanted characters, as provided by the mysql NodeJS package that is used within the application.

Query parameters are one defence mechanism for ensuring the values being passed in don't carry malicious intent such as a SQL injection attack. An example of the use of query parameters in the 6MWTest application is shown below in Figure 4-2:

Figure 4-2 – example query using query parameters

```
connection.query('insert into user (forename, surname, dob, email, password) values  
(?,?,?,?,'), [forename, surname, dob, email, hash],
```

Here we are creating a query to register a new user. The first section is the query itself with the query parameters denoted by a '?' symbol as the values to insert. The actual values passed in are held in variables and stored in an array as the second parameter of the connection query. Using this method, the query will try to execute first before looking at the values passed in. Therefore, the query knows exactly what it is trying to do and what to expect of the passed in values. If the value passed in isn't what it expected then the query won't execute. The values are also stripped of any unwanted characters. On the contrary, a query without query parameters may look something like Figure 4-3:

Figure 4-3 – example query without query parameters

```
connection.query('insert into user (forename, surname, dob, email, password) values  
('+forename+', '+surname+', '+dob+', '+email+', '+hash+')')
```

The above query would execute with the values as they are passed in before knowing what goal it was actually trying to achieve. This means an attacker, for example, could pass in a drop table command as the *hash* value and potentially delete a table from the database. Of course, further knowledge of the database would need to be had before being able to carry out such an attack but the crucial point is that the vulnerability is present.

4.2 Front-end System Design

The Ionic framework is what makes up the front-end of the application. Angular, the chosen development language for Ionic, by design follows a pattern that could be described as the Model View Controller (MVC) pattern.

MVC is an architectural design pattern consisting of three different components: the Model, the View, and the Controller (tutorialspoint, 2018). The Model is the component that handles the retrieval and manipulation of data from the database. The model is entirely separate from the other two components and is only used as a datastore. The View is what the user sees and interacts with. Users use the view to interact with and manipulate the data, with requests to the controller. The controller listens for requests from the view and issues the requests to the model to update/retrieve data. The controller also provides the view with the data retrieved from the model.

Figure 4-4 shows the interactions between the components in the MVC pattern.

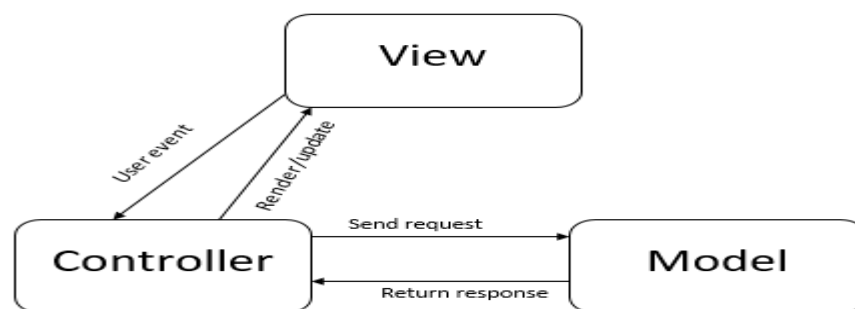


Figure 4-4 – MVC design pattern interactions

4.2.1 MVC in Angular

The Angular ecosystem consists of a slightly adapted version of MVC. Each page within an Angular application consists of a provider (Model), template (View), and a component (Controller). The interactions between each of the components is as follows:

The provider (model) is typically a list of HTTP calls using Angular's HTTP service which make calls to the API defined in the backend of the system. The API returns a response from the HTTP calls. On receiving a response, the provider (model) notifies the component (controller) which in turn updates the template (view).

The template (view) is what the user sees and interacts with. References to functions within the component (controller) are available in the template (view) and are called after user interaction.

The component (controller) listens for these calls and notifies the provider (model) of a request. On receiving a response from the provider (model), the controller updates/notify the template (view) with any changes.

To represent the interactions between the Angular ecosystem, a graphical representation can be found in Figure 4-5.

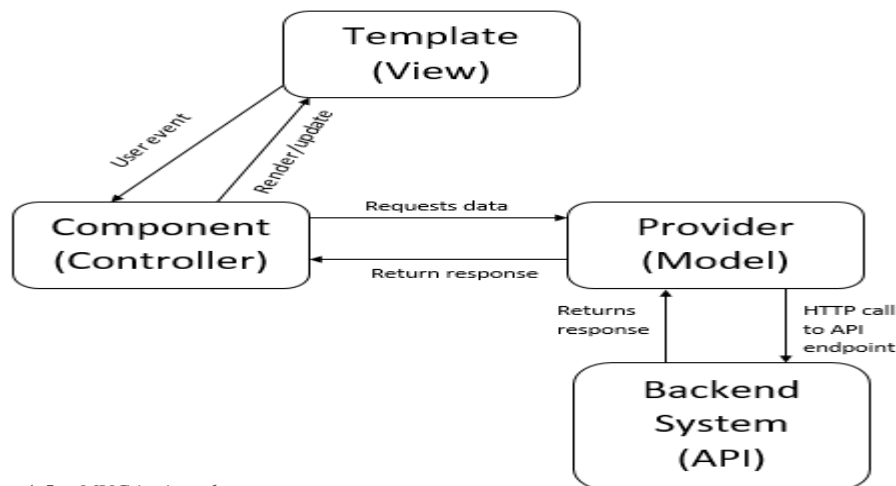


Figure 4-5 – MVC in Angular

This design has the advantage of being able to reuse the provider (model) in multiple components (controllers) throughout the system as it is completely independent of the view. This enables the developer to define reusable functions inside providers (models) such that they do not need to repeat function creation.

This design is not enforced in Angular and is sometimes adapted in such a way that the component acts as both the controller and the model, removing the provider entirely. This results in the component (controller and provider at this point) making calls to the API, waiting on the response, and notifying the view on return. This approach is used mainly when the API calls are specific to that page, and won't be used anywhere else in the system.

4.2.2 One-page Applications in Angular

Angular is used to create one-page web applications. There is one main container page which defines the global application layout. The main container page has a section which displays the page that the user has selected. This differs from a multi-page application where the same layout would be repeated in each separate page. A visual representation showing the difference between these two approaches is found in Figure 4-6.

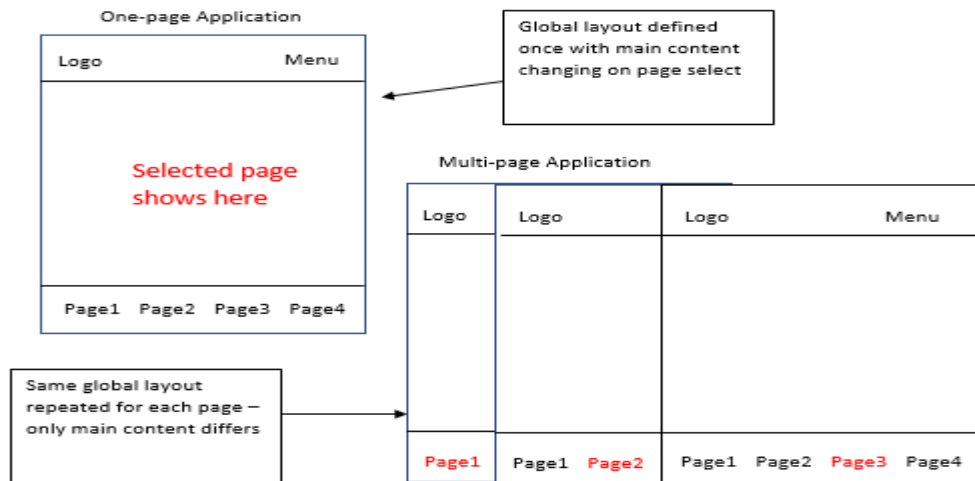


Figure 4-6 – one-page applications vs multi-page applications

The backbone of the front-end of the application are the files that reside within the *app* folder. The *app* folder is what holds all the files that make up the main component of the one-page approach. In this folder there is a component, template, style-sheet, main typescript file, and most importantly a module.

The component and style-sheet are used for adding functionality and style that will be used globally throughout the application, regardless of a specific page's implementation. In the 6MWTest application, the *app* template holds a reference to the *rootPage* variable found in the component - where the *rootPage* refers to which page the user has selected. This variable allows the *app* template to know which page to show depending on the user selection. By default, in the 6MWTest application, the *rootPage* variable is set to show the login page of the application.

As mentioned briefly above, the most important aspect of the *app* folder is the module found in the file named *app.module.ts*. The module is where, amongst other things, the components and packages used within the application are declared. Each component should be declared under the *declarations* array and the *entryComponents* array. This then allows the pages associated with each component to provide functionality to the user. The components and providers within the application make use of imported packages for different tasks. These imported packages must also be declared within the module under the *imports* or *providers* arrays, depending on the type of package. These declarations in the module are used when the application is being packaged up for deployment.

4.3 Back-end System Design

Unlike the front-end of the application, the back-end didn't require any large frameworks to implement. NodeJS was used along with ExpressJS to act as a middleware, since ExpressJS provides a lot of out of the box functionality that reduces boilerplate code.

The layout of the back-end application is small and simplistic since its primary function is to provide the REST API. The files that make up the REST API are as follows:

- Controller files have been created which hold the implemented functions that call and query the database. Each controller file is specific to creating functions dealing with either the user and account details or the results details. The functions defined in these files are small and well-defined, performing only the task required of them. Each function handles the returned data from the database in its own way and returns the response as a JSON object.
- A *routes.js* file has then been created which maps the API endpoints to the functions defined in the controller files. For example:

Figure 4-7 – example API route configuration

```
app.route('/api/register').post(accounts.register);
```

Figure 4-7 defines the API endpoint *api/register* as a post request and maps it to the register function found in the accounts controller. In this example, *app* is a reference to the imported ExpressJS package and *route* is a function provided by the package which routes the defined URL to the defined function.

To tie the back-end up, the *index.js* file calls the *routes.js* file, passing in the imported ExpressJS package. This registers the routes to be used in the application. Furthermore, the *index.js* file declares the port for the application to run on and again using ExpressJS initiates the *listen* function which essentially starts the back-end service, passing in the declared port.

4.4 Database Design

The data required for the 6MWTest application is the users' profile information and their 6-minute walk test results. Due to the small amount of data required, the database for the application is simplistic.

The final database consists of two tables: user and result. The structure of both tables is shown in Table 4-1 and Table 4-2 respectively.

Column Name	Column Type	Nullable?	Default	Extra
user_id (PK)	int(11)	No	None	AUTO_INCREMENT
forename	varchar(255)	No	None	
surname	varchar(255)	No	None	
email	varchar(255)	No	None	
password	varchar(255)	No	None	
dob	date	No	None	
height	int(11)	Yes	0	
weight	int(11)	Yes	0	
gender	varchar(1)	Yes	NULL	
illness	varchar(255)	Yes	NULL	
height_unit	varchar(1)	Yes	F	
weight_unit	varchar(1)	Yes	S	

Table 4-1 – structure of user database table

Column Name	Column Type	Nullable?	Default	Extra
result_id (PK)	int(11)	No	None	AUTO_INCREMENT
user_id (FK)	int(11)	No	None	
date	datetime	No	None	
distance	int(11)	No	None	
city	varchar(255)	Yes	NULL	
street	varchar(255)	Yes	NULL	

Table 4-2 – structure of result database table

The relationship between the two tables is a one-to-many relationship where one user can have from zero to many results. The tables are joined on the user_id column, where the user_id column is stored as a foreign key in the result table. This relationship allows easy retrieval for a user's or multiple users' results.

Figure 4-8 shows the relationship diagram between the two tables.



Figure 4-8 – relationship diagram for user and result database tables

Chapter 5. Detailed Design and Implementation

5.1 Languages and Frameworks Used

5.1.1 Ionic (Drifty Co., 2018)

Ionic is the framework used for the front-end of the 6MWTest application. Ionic allows you to build hybrid mobile applications using typical web technologies together with Ionic's library of plugins. At the time of writing, Ionic supports any devices running iOS 8 and above, Windows 10 Universal App, and Android 4.4 and above. The version of Ionic used for this project was 3.7.1, which was the latest stable build at the time of starting the project.

5.1.2 HTML5 (mozilla, 2018)

HTML5 is the latest version of the HyperText Markup Language and is used for creating the elements and templates that are shown to the user. Element support has drastically improved over the years with HTML5 now supporting input elements such as URL, datetime, and search. Semantic and structural elements have also been introduced in HTML5 with examples such as article, aside, header, and footer.

5.1.3 Angular (Google, 2018)

Angular is the language choice for Ionic and according to the 2018 StackOverflow Survey (stackoverflow, 2018), the most popular framework used by developers. It is a JavaScript framework with a lot of built-in features for both the JavaScript side and the HTML side, including out-the-box scripts for preventing web attacks such as cross-site scripting. For this project, Angular version 4.4.3 was used.

5.1.4 SCSS (Sass, 2018)

SCSS is an extension of the Cascading Style Sheet “language”, which is used alongside HTML for styling the application. SCSS builds on CSS by adding additional features and syntax making styling more robust and powerful.

5.1.5 Cordova (The Apache Software Foundation, 2018)

Cordova is used alongside Ionic to provide plugins and packages that allow the Ionic front-end to access features of the device that a hybrid application wouldn't normally have access to, such as controlling notifications and running the application in the background. Cordova is also used when porting the web-based hybrid application over to a native executable. Cordova version 7.0.1 is used in this project.

5.1.6 ChartJS (ChartJS, 2018)

ChartJS is a small and easy to use JavaScript charting library. It is used in the project to display the last ten 6-minute walk test results to the user in the log tab. ChartJS is easy to setup, with simple and accessible documentation.

5.1.7 NodeJS (Node.js Foundation, 2018)

NodeJS is a JavaScript framework that can be used to run front-end code on the back-end. Using NodeJS as the choice of language for the back-end means that an application can be written entirely in the one language. NodeJS has grown in popularity over the years with companies such as Netflix now using it in production (Netflix, 2014). The version of NodeJS used in this project is 6.11.2.

5.1.8 ExpressJS (Node.js Foundation, 2018)

ExpressJS is a NodeJS framework used for producing a middleware between the front and back-end of the application and for easily creating APIs. ExpressJS is easily configurable with many settings and features which includes supporting additional packages that can be used with it such as the ExpressJS CORS (Goode, 2018) package. The version of ExpressJS used in this project is 4.16.2.

5.1.9 MySQL (Oracle, 2018)

MySQL is a database provided by Oracle which is used by many large enterprise companies including Google, Facebook, and Adobe. It allows the creation and support of relational databases, using SQL (Structured Query Language) as its implementation language. The CIS department offer a MySQL instance to students for use in their projects. The version of MySQL offered by the CIS department is version 5.7.21.

5.2 Additional Tools and Technologies

5.2.1 Bitbucket (Atlassian, 2018)

Keeping a backlog of versions and being able to track features is a crucial aspect of agile development. Version control software allows you to see changes that have been made to one's codebase, create copies of the codebase to add new features whilst maintaining the original codebase, amongst other features. Bitbucket is an online version control platform powered by Git and created by Atlassian. Both the front-end and back-end of the 6MWTest application reside in a private Bitbucket repository.

5.2.2 Trello (Atlassian, 2018)

Trello, another Atlassian tool, provides software to organise the features and issues raised during the development process. The software used for the 6MWTest application was Trello's boards, which were integrated directly with the Bitbucket repository mentioned above. A feature board is another key part of agile development. The categories on the board were "To Do", "Doing", and "Done". Using these categories, features were added and moved into the relevant category after each weekly catch-up meeting with the supervisor.

5.3 Development Environment

5.3.1 IntelliJ IDEA (Jetbrains, 2018)

All code produced for the 6MWTest application was written using IntelliJ IDEA Ultimate Edition version 2017.2.2. JetBrains, the company behind IntelliJ, offer the Ultimate Edition as well as some of their other products to students free of charge. The reason for choosing IntelliJ Ultimate Edition (referred to from now on as UE) wasn't due to the fact that it was free but due to the features that it offers. Firstly, UE provides full JavaScript and TypeScript support which includes syntax highlighting and dynamic error warnings, amongst a wealth of other features. Secondly, UE offers great support for debugging JavaScript code directly in the IDE, whilst the application is running in the browser. This enables one to interact with the application on the browser but have a more robust and easy-to-use debugger within the IDE. Lastly, UE links directly with any version control systems which meant that the Bitbucket repository being used for this project was easily maintainable without having to leave the development environment to do so.

5.3.2 PuTTY (PuTTY, 2018)

Most of the development was out with the University's network. Therefore, to access the back-end Node server required a SSH client. The SSH client used to access the network was PuTTY. By using PuTTY, the terminal that would be accessible through a University computer is emulated on the computer that PuTTY is running on. This meant that the Node server could be interacted with without having to be present at University.

5.4 User Interface Design

The focus of the design was to ensure that the user was provided with a user interface that was not cluttered with features and that each feature of the application was distinct and clear in its purpose. The user interface for each specific page was tweaked as and when features were added but the overall general layout was decided in three separate iterations.

5.4.1 Initial Wireframes

The first iteration of the user interface (UI) was created as a set of wireframes. Wireframes provide a quick but useful resource to discuss the layout of the application and get a feel for how the application is going to look. The wireframes were created using a free trial of the wireframing and prototyping software Justinmind (justinmind, 2018). The set of wireframes created can be viewed in Appendix E.

As shown in the initial wireframes in Appendix E, this was a rough draft for the UI. It provided a starting point for discussing ideas with the supervisor and for getting a feel on which pages would link and how each page would be presented.

5.4.2 Initial Prototype and JPMorgan Design Meeting

Iteration two consisted of building on the wireframes to create the first physical prototype. Sample screenshots of the prototype are shown in Figure 5-2, Figure 5-3, Figure 5-1, and Figure 5-4.

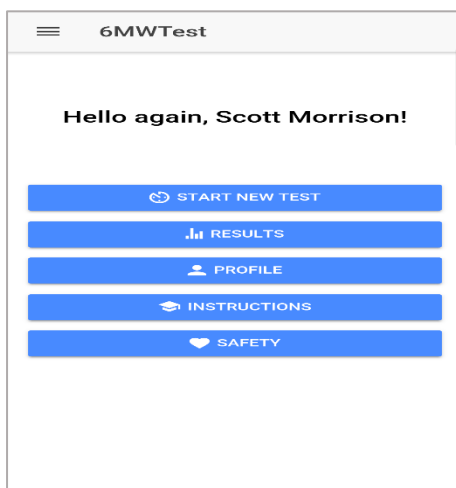


Figure 5-3 – initial home screenshot

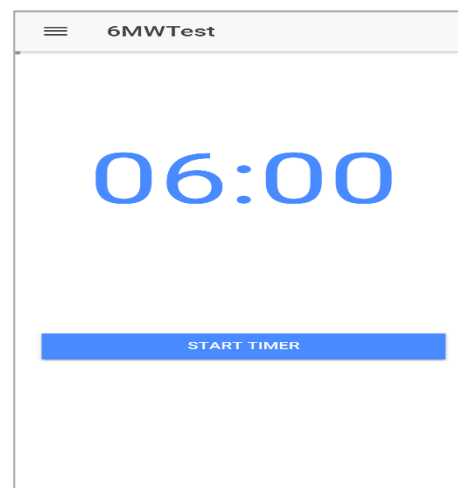


Figure 5-2 – initial test screenshot

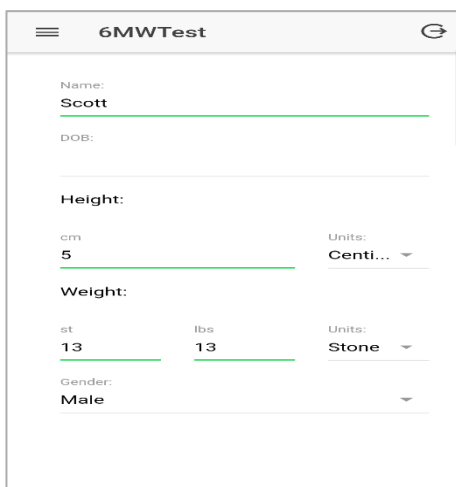


Figure 5-4 – initial profile screenshot

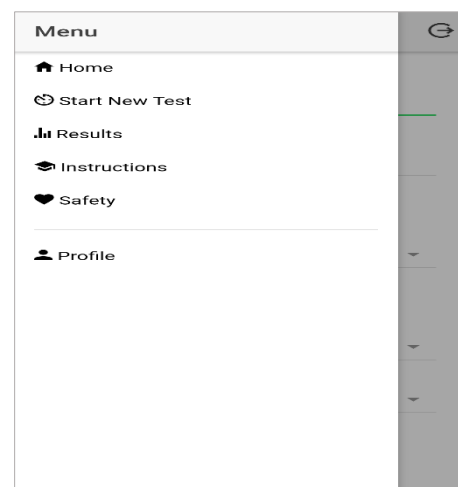


Figure 5-1- initial menu screenshot

In the first physical prototype, the design was kept very basic and simple. The user was provided with a link to each page both on the welcome screen and on the “hamburger” style side menu – giving them easy access to the pages within the application. The links were appropriately named and well-defined, with relevant icons to signify their purpose. Sticking to the default colour scheme as provided by the Ionic framework, the colours were kept simple and minimal.

This prototype was non-functional, with only page links and some elements defined in the profile page. The next stage of retuning the design drew on the experience of UI/UX analysts from JPMorgan. The analysts were sent the initial requirements (see Table 3-1) prior to the consultation meeting to give them an idea of what user interactions would be expected in the application. The two analysts had volunteered their time to discuss the design considerations with reference to the requirements and offered any advice and criticisms on the current physical prototype. The outcome of the meeting was a hand drawn wireframe (see section 0) which would be used as a guideline for the final design.

During the meeting several points were raised and acted upon. Firstly, the different pages defined in the application could be simplified. Rather than have a ‘Home’ page which served little purpose, the decision was to make the ‘Start New Test’ page the initial screen and rename it to ‘Home’. This brought more meaning to what the point of the application was and removed the need for the user to have multiple interactions with the application before being able to access the 6-minute test. The ‘Safety’ and ‘Instructions’ pages were also combined and rather than have them as a separate page, it was decided to have them accessible through the ‘Profile’ page. This brought the page count down to just 3: ‘Home’ (which is where the user would start a 6-minute walk test), ‘Log’ (where the user would view their results), and ‘Profile’ (where the user’s profile information was stored as well as access for safety and instructions). Since the application now consisted of only three pages, the side menu was removed and instead replaced with three distinct tabs along the bottom of the application. Removing the side menu and replacing with the tab layout gave the application a more distinguished mobile application look and feel rather than a web-based look and feel.

To provide the user with the ability to view their results both graphically and textually, a single ‘Log’ tab was created. The ‘Log’ tab would provide the user with a graph at the top of the page showing their last ten results, and a list of all their previous results. The user then could select one of their previous results and be taken to a separate view within the ‘Log’ tab

which would give them specific details on this result including a comparison against those from a similar age group. The plan for the specific result page was then to also use it for when the user has completed a 6-minute walk test. When the user completed a test, they would be automatically re-directed to the specific result page which would show their result and their comparison. This ensures that there is familiarity and consistency for the user when using the application.

The design of the profile page wasn't altered much from the physical prototype. An idea was to have a user profile photo in the profile page as well as a distinct 'Edit Profile' button for when the user wished to make changes but neither of these features made it to the final design (section 5.4.3). The main addition to the profile page, as mentioned above, was including the safety and instructions for the 6-minute walk test.

5.4.3 Final Design

Taking in to consideration the design guidelines in Appendix E, section 0 and after adding the functionality to the application, the final iteration of the design of the application was created. Example screenshots of the home, profile and log pages are shown below in Figure 5-5, Figure 5-6, and Figure 5-7. The remaining screenshots can be found in Appendix H.

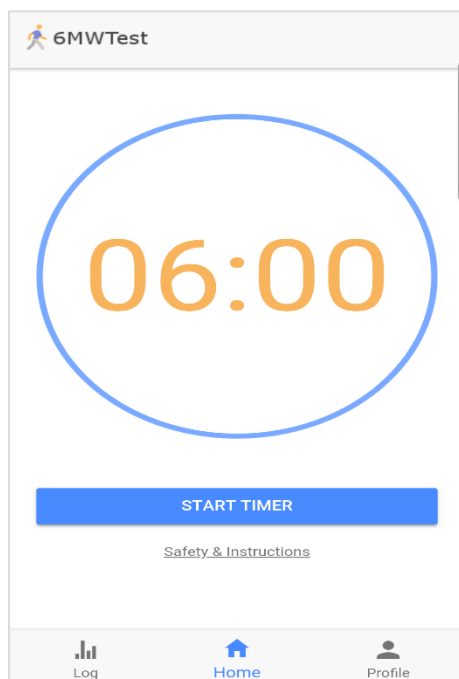


Figure 5-5 – home screen

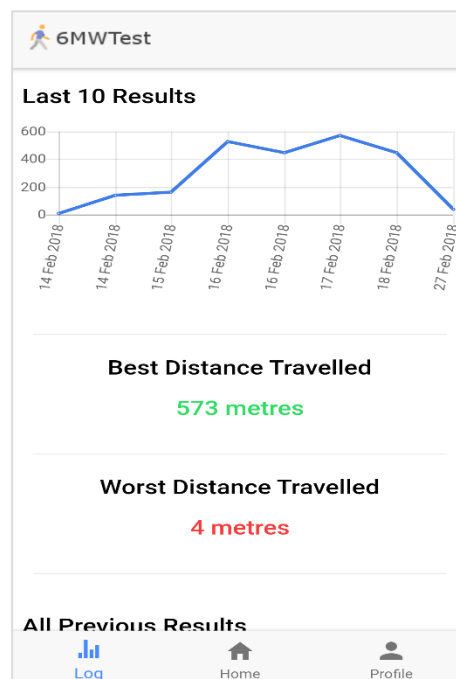


Figure 5-6 – log screen

Figure 5-7 – profile screen

As can be seen from comparing the final design to the initial prototype, the user interface is now much cleaner and concise in its features. The colour scheme hasn't changed from the default Ionic colour scheme except for the additional orange that has been used to match the logo. During development it was decided with the supervisor that rather than have the safety and instructions accessible through the profile, to have these accessible on the home page as this seemed to be a more fitting location for the safety and instructions. Hence, the safety and instructions is now featured on the home page and opens as a dialog box when clicking the link under the start button, or before the user carries out their first 6-minute walk test. The profile resembles more the design produced from the initial wireframes than the design guidelines proposed by the JPMorgan analysts. Adding a profile photo wasn't deemed relevant for the application. The results page provides the user with both graphical and textual formats for quickly viewing their previous results and tracking their progress. The addition of the face icons on the specific results page gives a clear and obvious guideline for the user to see straight away how their last result compares to previous results and to those in a similar age group, without overloading the user with information.

Overall, the look and feel of the application is now thought to be clear, consistent, easily usable, and offers a more mobile-like environment than the initial design. Each page will be described in more detail in the following section.

5.5 Features of 6MWTest Application

5.5.1 Registration and Login Security

User registration is mandatory for the 6MWTest application. To keep the registration process easy for the user, the data gathered during registration has been kept to a minimum before being able to perform a 6-minute walk test. Registration entails the user entering their name, email, date of birth, and password. On registering, the application will send the data to the back-end system which confirms if the user has already registered (by checking their email) or if they are a new user.

Each time a new user is registered, the password received by the back-end is both hashed and salted using the NodeJS package bcrypt (Source, 2018). bcrypt provides both an async and a sync *hash* function. The async version is the recommended approach as the sync version can be CPU intensive therefore using the sync version could lead to complications with processing other requests coming in at the same time on the server. The async *hash* function takes three parameters: the plain text password, the number of salt rounds, and a callback

function that returns the hashed password or an error. The `bcrypt hash` function supports promises therefore the 6MWTest implementation doesn't specify the third parameter and instead uses the returned promise for retrieving the hashed password. Assuming the hash has been returned successfully, the hash is then stored in the database and is used for any comparisons or any other need for the user's password from there on. Plain text passwords are at no point stored in the database of the 6MWTest application.

When a user logs in to the 6MWTest application, the credentials are sent to the back-end and are verified. First, the back-end checks if the user has provided a valid email address. The email validation is returned by passing the email in to a function which compares the email against a regular expression. Although regular expressions are not a comprehensive way of checking email addresses, they provide enough to validate that the email follows the correct format expected. Secondly, the back-end checks if the user's email exists in the database. If the email does not exist then a response of *"Invalid email or password provided."* is sent to the front-end to present to the user. If the user's email does exist in the database then the password is hashed using the approach explained above and is then compared to the hashed password retrieved from the database, using `bcrypt's compare` function. The `compare` function takes the two hashed passwords and returns a boolean value. The returned value will either be true if the hash passwords do match or false if they don't. If the result is false then the same response of *"Invalid email or password provided."* is sent to the front-end. If the result is true then the user's data that matched the provided email in the database is sent back, along with a token.

The token sent back is a JSON web token generated by a HMAC-SHA256 hashing algorithm. The algorithm implementation is provided by the NodeJS `jsonwebtoken (jwt)` package created by Auth0 (Auth0, 2018). Using the `sign` function provided by `jwt`, a token is generated by passing in the payload (the data to be hashed – the user id in the 6MWTest case) and a secret key (used for generating the hash). The returned value is then a unique string of characters that represents the payload. This unique string is returned to the front-end application and stored locally on the device along with the logged in user's email address.

After these steps have been carried out, anytime the user opens the application after having already logged in, the front-end will send both the email address and the token that are stored locally on the device to the back-end. The back-end verifies that the email belongs to a valid user and then verifies the token that was passed back. The token is verified by using `jwt's`

verify function, which accepts the given token as the first parameter, the secret key as the second parameter, and a callback function as the third parameter. The callback function returns a boolean which signifies if the token is valid or not. If the token is valid then the back-end returns the logged in user's data and automatically allows the user access to the system. If the token has expired then the user is notified and is asked to login again.

Each form of the login, registration, and reset password undergo email validation checks against the regular expression mentioned above. Further details on the error messages returned can be found in the test cases in Appendix G.

5.5.2 User Profile

Each user of the 6MWTest application has their own designated profile. The mandatory information provided by the user on registration is their forename, surname, and date of birth. This information is automatically populated when the user views their profile page. Further optional information can then be provided by the user which is their height, weight, and gender. Both the height and weight fields are customisable in terms of which units they display. For height, the user can choose either to display the figure in centimetres or in feet and inches. The weight unit can also be changed with the options for either stone and lbs, or kilograms.

These units are held in the database as a single letter as follows:

Unit Value	Meaning
F	Feet and inches
C	Centimetres
S	Stone and lbs
K	Kilograms

Table 5-1 – height and weight unit letter map

By default, when a new user is registered, the database has the height unit set to *F* and the weight unit set to *S*.

All user data is stored permanently and therefore enables any user to use the application on different mobile devices with the use of their login credentials.

5.5.3 Age Group Result Comparison

On selecting a specific previous result from the log page or when the user has completed a 6-minute walk test, they are directed to the specific result page. On this page, amongst other details discussed in the following section, is a section which shows how the user's results compare to the average result from other users in their age group.

The user's age group is determined by a query executed on the back-end. The query looks at all users who have a date of birth that is two years greater than the current logged in user's date of birth and two years lesser than the current logged in user's date of birth. All distances covered by the users who match these conditions are then retrieved and the mean of the distances is calculated and returned to the front-end.

The mean returned to the front-end is then used in a function to calculate the percentage difference between the age group mean provided and the distance covered for the result that is being viewed. On gathering the percentage difference, different icons and text are used to display the result to the user. The different icons are displayed depending on how the percentage difference calculated above compares with the set average threshold of 15%. Figure 5-8 shows the different icons and text that is displayed.

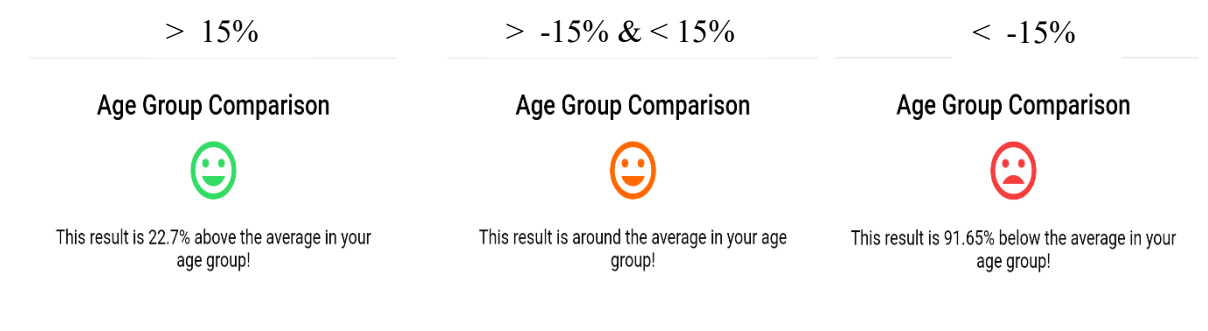


Figure 5-8 – different icons and text for age group comparison

5.5.4 Displaying and Comparing Users' Previous Results

A feature that was lacking in the application evaluated in the Related Work section (FC) was that it only provided the user with the last ten previous results. Furthermore, the results listed by the evaluated application only provided the user with the date they carried out the test, meaning there was no way for the user to distinguish between two tests carried out on the same day. The 6MWTest application provides the user with all their previous results and in more detail. Each result in the 6MWTest application includes the date, time, and location of the test. If the application is unable to find the location where the test was carried out, it defaults to “Unknown” as to make the user aware. The 6MWTest application also provides the user with many ways to compare their previous results, unlike FC. On the log tab the user can view their best result and their worst result. Furthermore, a graph shows the performance of their last ten results (see Figure 5-9).

Having the best result and worst result clearly shown on the log page provides the user with an easy and quick way to determine improved or decreased their waking distance. Both values are also distinct in colour to add an additional visual cue for the user.

The line graph, which was created using the ChartJS JavaScript library, also easily highlights the progress that the user has made in their previous ten results. On pressing a point in the line graph, the user is presented with a tooltip which gives more detail on the test time and specifically how many metres the user travelled.

Below the graph and the best/worst distance sections is the list of the user's previous results. On selecting a specific previous result, the user is taken to a page which shows the date, time, and location of their test. Furthermore, it provides the result of the chosen test, the average distance travelled per minute during the test and a comparison of how this result compares to their previous results.

Similar to the age group comparison above, the user is presented with a section showing an icon and a message that both indicate how this result compares to their previous results. The same function used in the age group comparison is used to retrieve the percentage difference of the mean returned from the back-end, and the same criteria is used to switch icons, colours, and text as is shown in Figure 5-8. The difference between the two comparisons is the back-end query that retrieves the mean of results. Since this result is based on the previous results of the current logged in user only, the query is simpler than that of the age group comparison. Joining the user and result tables is not required as the user id is provided by the front-end, therefore the query simply retrieves all distance results where the *user_id* column contains the provided user id. Assuming there are previous results in the database, the mean is calculated and returned for use by the front-end. If there are no previous results then a response of *"No previous results found."* is sent to the front-end and used to notify the user.

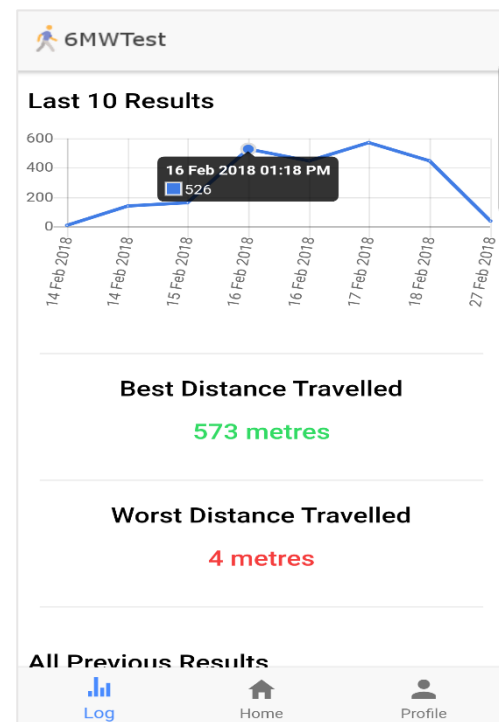


Figure 5-9 – 6MWTest log tab

5.5.5 6-minute Walk Test

On opening the application, the user is presented with the ‘Home’ screen which is where they would begin a 6-minute walk test. When this page has loaded, a request is sent to the back-end with the id of the current logged-in user as a parameter to check if they have any results stored already in the database. On first use of the test, without no previous tests completed, safety and instructions are displayed. They will not be displayed on any subsequent tests, but can be viewed in the ‘home’ screen at any time.

Pre-countdown Steps

Prior to the timer beginning its countdown, the application carries out a sequence of events:

- Ionic’s Diagnostic plugin (discussed in section 3.2.2) is first used to check if the application is able to access high-accuracy location services.
- If high-accuracy isn’t enabled then the application uses the Diagnostic plugin’s *isLocationAuthorized()* function to check whether or not the application has permission to access the location of the device.
- If the application doesn’t have location permission then the user is prompted to allow the permission. The permission is set automatically by the Diagnostic plugin’s *requestLocationAuthorized()* function.
- If the application does have location permission, or has been given permission in the previous step, then the Location Accuracy plugin is used to ensure that the application has access to high-accuracy location services. If it doesn’t, then the Location Accuracy plugin’s *request()* function is used to automatically set high-accuracy to be enabled.
- If high-accuracy is enabled or has been enabled by the Location Accuracy plugin then the application requests the initial location of the device using the *getCurrentPosition()* function from the Geolocation plugin. The position object returned by this call is then used by the Geocoder plugin to retrieve the street name and city for where the test is being carried out. This information is stored for later use when saving the result.

After all steps are complete, the *startTest()* function is called, initiating the countdown.

6-minute Countdown Timer

The countdown timer is implemented using an rxjs (ReactiveX, 2018) *Observable* timer. The timer subtracts 1 from an initial *counter* variable set to 361, every second. For showing the formatted time to the user, the *counter* variable is referenced in the template with an *async* Angular pipe (which tells the template that this value will be asynchronously updating), and a custom *timer* pipe defined which formats the time. Figure 5-10 shows the template mark-up which provides the user with the formatted counter.

Figure 5-10 – countdown timer template snippet

```
<div class="timer-circle">
  <p class="timer-text" ion-text text-center>{{countDown | async | timer}}</p>
</div>
```

Angular allows the creation of custom pipes hence the custom *timer* pipe was added to this project. The implementation of the pipe was taken and adapted from a user's answer on a thread on stackoverflow (Vega, 2017). Figure 5-11 shows the created pipe. A conditional was added to ensure that the timer displayed the formatted six minutes in the case of a null value.

Figure 5-11 - custom Angular pipe for formatting countdown timer

```
export class TimerPipe implements PipeTransform {
  transform(value: number): string {
    if(value === null) {
      value = 360;
    }
    const minutes: number = (Math.floor(value/60));
    return ('00' + minutes).slice(-2) + ':' + ('00' + Math.floor(value-minutes *
60)).slice(-2);
  }
}
```

GPS Distance Tracker

By far the most important aspect of the 6-minute walk test feature is the GPS distance tracker. The resource used for retrieving the GPS location was Ionic Native's Geolocation plugin.

The Geolocation plugin, as mentioned in section 3.2.2, provides two functions for retrieving the device's location: *getCurrentPosition()* and *watchPosition()*. *getCurrentPosition()* returns the device's location at that point and time only once. *watchPosition()* returns an *Observable* which returns the device's location every time the location changes. On first starting the project, up until early February, the implementation for the GPS distance tracker involved

using the *getCurrentPosition()* function. Field tests were carried out continuously to test the accuracy of this approach but the results were sporadic and weren't showing signs of being able to provide an acceptable enough accuracy to be used in the application. The implementation was therefore changed to use the *watchPosition()* function for retrieving the device's location.

When the timer begins to countdown, the *watchPosition()* function is called which stores the returned position in an array of positions every time the position changes. After every twenty seconds an algorithm is run which takes the array of positions and calculates the distances between each of them, accumulating the distance as it calculates using the Haversine formula (discussed below). The latitude and longitude produced by the positions are passed in to the implementation of the Haversine formula to calculate the distance.

A high-level graphical representation of how the algorithm works is shown in Figure 5-12.

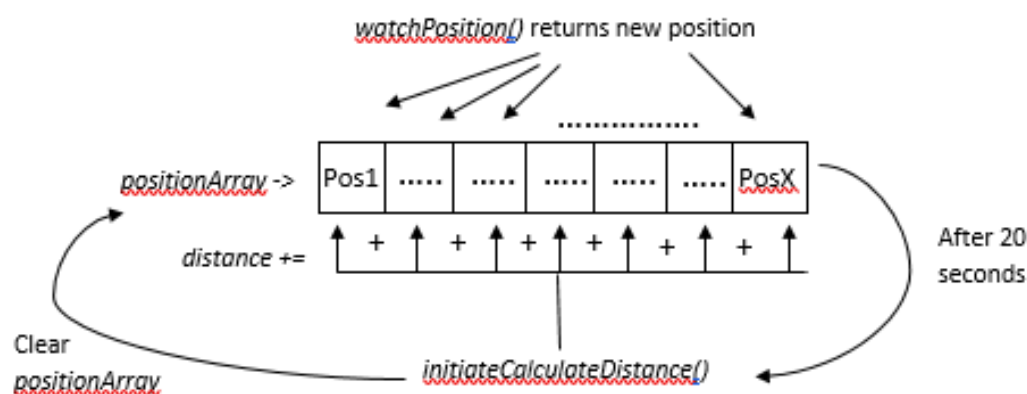


Figure 5-12 – calculating distance algorithm representation

Mobile GPS can at times produce positions that are far from where the device is. Therefore, to smooth out the data once there are no positions left to calculate in the twenty second block, the accumulated distance is checked against a 50-metre threshold. The threshold is used to ensure that the accumulated distance isn't over what is feasible within a 20-second walking timeframe.

The algorithm steps for calculating the distance is as follows:

- If the accumulated distance within the twenty second interval is below the 50m threshold then the distance is added to an overall distance array and added on to a variable which holds the overall distance covered.
- If the accumulated distance is greater than the threshold and there are previous distances recorded in the overall distance array, then the algorithm calculates the

average distance covered so far in the test and overrides the accumulated distance with the result. This result is then added on to the overall distance covered. Additionally, the result is added as an element in the overall distance array.

- If the accumulated distance is greater than the threshold but there are no previous distances recorded then the average cannot be calculated correctly. In this scenario, the algorithm adds 1 to a variable which holds the number of times this scenario occurs. The accumulated distance at this point is cleared for the next set of positions.
- Once all calculations are complete, the array of positions and the accumulated distance are cleared, ready for the next set of positions.

This algorithm continues to run until the countdown timer reaches zero.

Test Completion

On the countdown timer reaching zero, the *Observable* returned from the *watchPosition()* function is destroyed and the final results are calculated. As explained above, the application keeps a count of how many times the average cannot be calculated correctly. Before saving the results the application checks if there were any calculation errors stored and if so, makes up the difference. The first calculation is the average of the overall distance by dividing the current overall distance by the current length of the overall distances array. This figure is then multiplied by the amount of calculation errors received during the six minutes and is added on to the overall distance covered (see Figure 5-13).

```
if (this.countZeroAvgDistances > 0) {  
  if (this.overallDist > 0) {  
    let overallAvg = this.overallDist / this.newDistarray.length;  
    this.overallDist += overallAvg * this.countZeroAvgDistances;  
  }  
}
```

Figure 5-13 – test completion distance calculator

The distance covered during the test is saved to the database, along with the date, time, location, and id of the user. Once the data has been saved, the user is redirected to the results page. The user is notified on the completion of their test via a local notification. The device will also vibrate and make a sound if the user doesn't have their device set to silent. Furthermore, at the stage of completion, the application retrieves the user's previous best result from the back-end and compares it with the current result. If the current result is greater

than the previous best result then the user is given a second notification congratulating them on achieving a new best distance.

A screenshot showing the new best distance notification is shown in Figure 5-14.

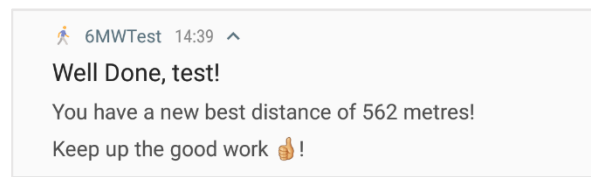


Figure 5-14 – new best distance notification

Haversine Formula

The Haversine formula implementation in the 6MWTest application was taken from an adaptation of Chris Veness' JavaScript Haversine formula found at Movable-Types (Venesse, 2002). The implementation provided by Chris Veness' has been adapted to cater for multiple languages by the development community, including an optimised JavaScript version (Dali, 2014). The optimised JavaScript version of Chris Veness' formula has been altered for this project to be compatible with the TypeScript language, and to ensure the formula returns the distance calculated in metres.

The Haversine formula produces the great-circle distance between two points which is the shortest distance over the earth's surface. This means there lies a potential limitation with the formula. This limitation is easier explained in a graphical format as shown in Figure 5-15.



Figure 5-15 – diagram showing haversine distance limitation

Although the limitation is present, the algorithm should not be affected by it. Since the algorithm uses the *watchPosition()* function, the great-circle distance is being calculated between the newly returned point and the previous returned point each time the user's position changes. These points are returned very frequently and thus are less likely to be affected by this limitation.

Stopping the Test

The user has the ability to stop the test at any point using the ‘Stop Timer’ button that is shown when the test is running. On pressing the button, the user will be presented with a popup which explains that their result won’t be saved if they stop the test and that they should rest if required. Results are only saved if the user completes a full 6 minutes of walking as that is the time required for a valid test.

The popup provides the user with two options: ‘I wish to stop’ (which will close the popup, stop the test, and reset the timer) and ‘I wish to continue’ (which will close the popup and continue the test as normal). The timer continues counting down throughout this scenario and will only stop when the user chooses the ‘I wish to stop’ option.

5.5.6 Background Mode

Ensuring the application could run in the background was a later requirement brought in during one of the weekly meetings with the supervisor in the development period. This requirement was to allow the user to walk freely without having to hold their phone and keep it awake. Due to the agile approach taken, this feature was easy to plan and develop.

This requirement was implemented using the Background Mode plugin from Ionic’s library of plugins (Drifty Co., 2018).

The Background Mode plugin allows the application to run in the background of the device as long as the application hasn’t been shut down completely. This means that the application will continue to run if the screen goes to sleep, the user switches between apps, and when the user locks the device.

On starting a new 6-minute walk test, the plugin’s *enable()* function is called which tells the device that the application has permission to run in the background. When the test is complete, and the application has redirected to the results page, the plugin’s *disable()* function is called, removing the permission to run in the background. The *disable()* function is also called if the user decides to stop the test before it is complete.

When evaluating the related application in the Related Work section it was noted that the Functional Capacity application did allow the GPS to continue running in the background but there was no indication whether or not the application was indeed still running. It is important to ensure the user is aware that their test is still running to not confuse them. There is potential for the user to stop walking to check that their application is still running, which

Scott Morrison

would significantly impact their results. By default, the Background Mode plugin provides a persistent notification any time the application moves in to the background after the *enable()* function has been called. The notification can be customised calling the *setDefault()* function which takes in an object with values that, for instance, can change the title, main text, and icon of the notification. The 6MWTest notification uses its own icon and meaningful text to indicate to the user that the application is continuing to run. When tapping the notification, the user is taken back to the application and both the notification and toolbar icon are removed until the user either completes/stops the test, or they move the application to the background again.

GPS tracking can be problematic when an application is running in the background, even with Background Mode enabled but the plugin provides the *disableWebViewOptimizations()* function to address that problem. The reason behind the problem is the operating system will try and optimize the resources produced by an application when the application isn't in the foreground. For instance, GPS and media. This optimization can lead to the GPS not running at all or not running as expected.

5.5.7 Network Connectivity Tracker

Network connectivity is required for the 6MWTest application to function thus the application tracks when the user has a network connection or not. To track the connection, the application utilises the Ionic Native Network plugin (Drifty Co., 2018). The Network plugin provides two main functions, the *onConnect()* function and the *onDisconnect()* function. Both functions return an *Observable* that will constantly watch for changes in the network connection until the *Observable* is destroyed. Since the application requires network connectivity throughout, both the *onConnect()* and *onDisconnect()* functions are called within the parent *app* component. This ensures that regardless of the page the user has in use, globally the network tracker will still be monitoring the device for changes.

If the device loses network connection then a popup is shown to the user. Similarly, if the device's network connection is then re-established after being lost, a different popup is shown to the user. Both popups are shown below in Figure 5-16 and Figure 5-17.

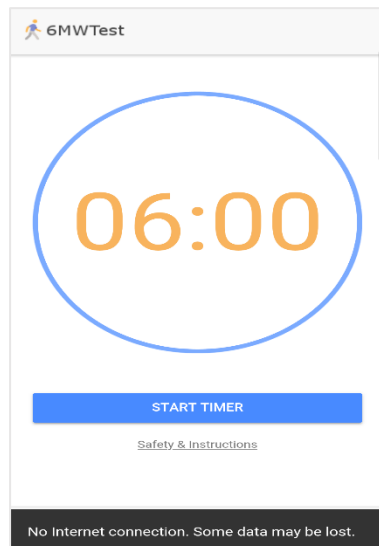


Figure 5-17 – no connection popup

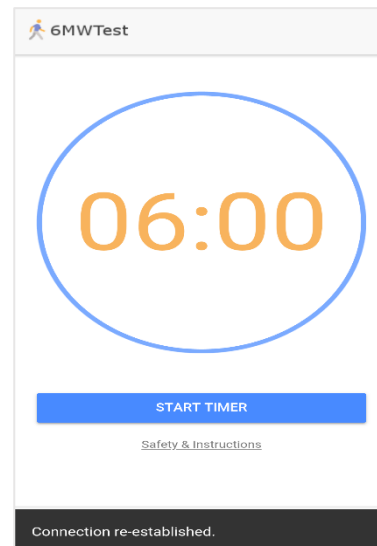


Figure 5-16 – connection re-established popup

The difference between both popups, with the exception of the message shown, is that the no connection popup will stay in the view until the connection has been re-established. Whereas the connection re-established popup is configured to stay in the view for two seconds before automatically being removed.

Chapter 6. Verification and Validation

Due to the focus being on the GPS distance tracker within the application, the primary form of testing was field tests. However, after implementing each feature of the application, use case tests for the specific features were also carried out.

6.1 Testing Environment

Ionic is a web-based development framework therefore testing could be performed on Chrome's mobile device emulator. Most features were tested either through Chrome's mobile device emulator or by emulating on a Samsung Galaxy S8 device, although during evaluation different Android devices were used for carrying out the tests with no problems arising.

Since the GPS tracker is problematic to test indoors, the major form of testing was performed outdoors in the same environment that was used for the evaluation and on a Samsung Galaxy S8 device, running version 7.0.

The REST API was tested using Postman version 5.5.0. Postman allows APIs to be tested by providing a desktop app which can be used for entering your API end-point along with any parameters.

6.2 Feature Testing

Since an agile development method was being followed, most features were implemented and completed within the week that they were discussed. Rather than testing small individual features one at a time, features were grouped together and tested to gain a better picture of the overall flow of the system.

Each set of features, apart from the distance tracker, was tested against a use case test document specific to that set of features. An example test document is shown in Table 6-1 which shows the cases carried out for the reset password feature. The rest of the features, along with their associated test documents, can be found in Appendix G, Feature Test Cases.

Feature: Reset Password				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Entering a valid user email and matching passwords	Email: test@case.com Password: "password1" Confirm Password: "password1"	Alert: "Password successfully reset. Please now login." Password changed in DB.	Alert: "Password successfully reset. Please now login." Password changed in DB.	As expected
Enter a non-valid user email and matching passwords	Email: test@invalidcase.com Password: "password1" Confirm Password: "password1"	Alert: "No account found with that email." No changes to DB.	Alert: "No account found with that email." No changes to DB.	As expected
Enter a non-valid email format and matching passwords	Email: test@case Password: "password1" Confirm Password: "password1"	Alert: "Invalid email address entered. Please enter a valid email." No changes to DB.	Alert: "Invalid email address entered. Please enter a valid email." No changes to DB.	As expected
Enter a non-valid email format and matching passwords	Email: 12345 Password: "password1" Confirm Password: "password1"	Alert: "Invalid email address entered. Please enter a valid email." No changes to DB.	Alert: "Invalid email address entered. Please enter a valid email." No changes to DB.	As expected
Enter a valid email and passwords don't match	Email: test@case.com Password: "password1" Confirm Password: "password2"	Alert: "Please ensure both passwords match." Password not changed in DB.	Alert: "Please ensure both passwords match." Password not changed in DB.	As expected

Table 6-1 – example test case document for reset password feature

Table 6-1 shows that normal inputs were tested as well as non-normal inputs, to gauge how the features responded to different inputs and scenarios. Using this approach, the features were tested in different scenarios giving an idea of which features had to be re-worked depending on their response.

As mentioned previously, the GPS distance tracker was tested outdoors and mainly on a Samsung Galaxy S8 device by installing the Android apk file produced from the Ionic and Cordova deployment. The same environment was setup each time tests were carried out, similarly to the evaluation stage. The GPS tracker results were then checked each time

Scott Morrison

against the result of using a metre wheel to measure the distance walked for 6 minutes. This allowed the tracker to be continually monitored as development took place to see what factors made the accuracy better or worse.

Using Postman to test the API, a collection of requests was created which had the end points for the API with valid and invalid parameters supplied if relevant. The tests were run throughout development of the API and at the final stage of development. Details of the tests run and the end results can be found in Appendix G, API Test Cases.

6.3 Debugging

Debugging the application was carried out using the Chrome console and IntelliJ's built-in debugger that was discussed in section 5.3.1. Using the debugger along with the console meant that the device could be emulated and debugged simultaneously, giving a real-world scenario when attempting to solve any issues that were found.

Break-points were used to stop execution of the application at certain points to be able to investigate the variables and objects concerned during the debugging process. Both Chrome and IntelliJ enable the use of stepping over and stepping in to a break-point execution to analyse associated variables and objects also.

Making use of console logs and alerts was a quick way to establish what state a variable or object was in at a given time. This approach was used at times when the issue wasn't critical and a quick verification of the data was required.

Using these tools available, any issues were easily identifiable which aided tremendously in finding a fix for them.

Chapter 7. Evaluation and Results

This chapter will cover the methods and procedures used for evaluating the 6MWTest application. At the end of this chapter the results from the evaluation will be discussed and summarised.

7.1 Aims

There were three aims for this evaluation: testing the accuracy of the GPS distance tracker with reference to the distance covered during a manual test, testing the repeatability of the GPS distance tracker, and assessing the usability and acceptance of the 6MWTest application.

7.2 Participants and Recruitment

Participants were recruited through the means of Facebook and through a network of friends. The inclusion criteria for participants were that they had to be aged between 18-40years - and not have any health conditions that prevented them from engaging in physical activity safely, as determined by the Physical Activity Readiness Questionnaire (PARQ) (verywellfit, 2018) found in Appendix C.

Each participant was sent an email with a zipped file which included an information sheet explaining the purpose of the project, what was expected of them, and an outline of any risks that were involved. Additionally, the zip file included a consent form and the PARQ mentioned above. Participants were asked to sign and complete the consent and PARQ forms and return at their earliest convenience.

Ten participants returned their completed forms and were then recruited for this evaluation.

The evaluation study received approval from the Department of Computer and Information Sciences Ethics Committee on the 13th of December 2017.

7.3 Test Procedures

Prior to carrying out any tests, participants were sent account details for the application to allow them to register a new account and to login. This allowed the participant to interact with the application independently and saved time for when the participant arrived for their designated test slot.

Each test slot lasted approximately 45 minutes and was carried out between mid-morning and late afternoon, although no specific time criteria was required for carrying out a test. The test environment was the same for each participant: a straight, clear, and flat area in Kelvingrove

Park, Glasgow. The environment was setup prior to the participant arriving, following the guidelines as set out by the ATS, discussed in section 2.2.1 above (American Thoracic Society, 2002). A distance of 50 metres was measured between a clearly marked starting point and a clearly marked end point. The participants were asked to walk the distance between the starting and end points for the purpose of the test.

On arriving for their test slot, participants were given further instructions on what was expected of them. Firstly, each participant was asked to ensure they had no running instances of the 6MWTest application on their mobile device. They were also asked to turn their location services off and their mobile data on to ensure that each participant's device was starting the test under the same conditions. Participants were then asked to carry out two 6-minute walk tests using the 6MWTest application. With the application running, participants were asked to walk between the start and end distance markers, walking as fast as they could but without running, until the built-in timer in the 6MWTest application reached zero. On completion of the application test, participants were given a short break if they so required and were given time to use the application to view their results. Participants were then asked to carry out two manual 6-minute walk tests using their own phone's built-in timer, walking between the two marked points, as fast as they could but without running, until the displayed timer reached zero. Once their timer reached zero, participants were asked to remain still whilst the remaining distance was verified using a metre wheel. Participants were then given a short period to rest and asked if they would like to know their results from the manual test.

Following the physical tests, participants were sent an email containing a Component-Based Usability Questionnaire (CBUQ) (Willem-Paul, 2011). The CBUQ is a validated questionnaire based on the Technology Acceptance Model (TAM) and Davis' six Perceived Ease-of-Use (PEOU) statements (Davis, 1989). The CBUQ consists of six simple questions with answers from a 7-point Likert-scale ranging from "extremely unlikely" to "extremely likely". A copy of this questionnaire can be found in Appendix B. Participants were asked to complete and return this questionnaire at their earliest convenience.

On gathering the results from the above tests, statistical analysis was carried out using IBM's SPSS Software (IBM, 2018) version 25, downloaded freely from Pegasus using the licence provided by The University of Strathclyde.

7.4 Data Analysis

Depending on the normality of data, different approaches to analysis can be carried out. Before doing any analysis, a test for normality was performed. Due to the small sample size of the data, the Shapiro-Wilk was performed as it is more suited for smaller sets of data. The test revealed that the data was normally distributed (see Figure 0-1 and Figure 0-2). Due to this result, the accuracy of the 6MWTest application was tested with the Pearson correlation and test-retest-reliability was tested with both a paired-sample t-test and an intra-class correlation.

7.4.1 Application Accuracy

As mentioned above, the accuracy of the application was analysed using a Pearson's Correlation since the data was deemed parametric. The purpose of carrying out a Pearson's Correlation was to determine how close the set of results obtained from the application tests were to the results obtained from the manual tests. The results of Pearson's Correlation coefficient can be interpreted from the following guidelines by StatsTutor (StatsTutor, 2018):

Result of Correlation	Strength
.00-.19	Very weak
.20-.39	Weak
.40-.59	Moderate
.60-.79	Strong
.80-1.0	Very strong

Table 7-1 - correlation coefficient results scale taken from StatsTutor

As shown in Table 7-1 above, the higher the correlation coefficient, the more the application results and manual results are correlated.

7.4.2 Test-re-test-reliability

For testing the repeatability of the application, a paired-sample t-test was used, followed by an intra class correlation coefficient.

The results produced by a paired-sample t-test include the mean of each set of data and the p-value. A p-value greater than .05 would indicate that the results from the repeated tests are not significantly different from each other.

The output from an intra-class correlation is used to show the strength of the correlation between the first set of application test results and the second set of application results. Table 7-1 can be used as a reference when determining this strength.

7.4.3 Usability and Acceptance

The aim of the CBUQ used during this stage is to gauge the perceived usability of a system, or part of a system. Using the instructions given on the CBUQ page (Willem-Paul, 2011), the average of all answers from the CBUQ is calculated and compared against the norm value of 5.29. The instructions in the page state that if the average is greater than 5.29 then this shows that the application is comparable to easy to use applications. If the average is lower than 5.29 then the application is more comparable to less than easy to use applications. As all questions were answered by all participants, the highest average that could be produced by the responses is 7, and the lowest average is 1. The closer the average is to 7, the higher the perceived usability is, and the closer the average is to 1, the lower the perceived usability is.

7.5 Results

The number of participants recruited for analysis was ten, but due to an error during data collection for the first participant, the number of participants' data actually analysed in the end was nine.

Out of the nine participants used for this evaluation, seven were male (78% of participants) and two were female (22% of participants). All participants were aged between 21-30.

7.5.1 Application Accuracy

The correlation coefficient result produced by the Pearson Correlation between the application tests and the manual tests, as shown in Figure 7-1, was .89 ($p < .001$). Following Table 7-1, this result shows there is a very strong correlation between the application results and the manual results.

Pearson's Correlation for Application Results vs Manual Results

		App_results	Manual_results
App_results	Pearson Correlation	1	.899**
	Sig. (2-tailed)		.000
	N	18	18
Manual_results	Pearson Correlation	.899**	1
	Sig. (2-tailed)	.000	
	N	18	18

Figure 7-1 - output of Pearson's Correlation for app vs manual results

7.5.2 Test-re-test-reliability

Looking at the below figures in Figure 7-2 and Figure 7-3, we can see that participants walked on average 575 meters (Mean = 575.0) on the first set of application tests and 573 meters (Mean = 573.4) on the second set of application tests. There was no significant difference between these two sets of tests (difference between the means = 1.56m, $p = 0.78$).

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	App_result_1	575.0000	9	71.70774	23.90258
	App_result_2	573.4444	9	78.19864	26.06621

Figure 7-2 - output of paired sample t-test statistics for app results 1 vs app results 2

Paired Samples Test

		Paired Differences							
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
Pair 1	App_result_1 - App_result_2	1.55556	15.85174	5.28391	-10.62917	13.74028	.294	8	.776

Figure 7-3 – output of paired sample t-test for app results 1 vs app results 2

The output of the intra class correlation is shown in Figure 7-4. The result from the intra class correlation shows that there is a very strong correlation between the first set of application results and the second set of application results (Average Measure Intra class Correlation = .99, $p < .001$).

Figure 7-4 – output of intra-class correlation between app results 1 and app results 2

Intraclass Correlation Coefficient

	Intraclass Correlation ^b	95% Confidence Interval		F Test with True Value 0			
		Lower Bound	Upper Bound	Value	df1	df2	Sig
Single Measures	.980 ^a	.915	.995	88.598	8	8	.000
Average Measures	.990 ^c	.955	.998	88.598	8	8	.000

Two-way mixed effects model where people effects are random and measures effects are fixed.

- a. The estimator is the same, whether the interaction effect is present or not.
- b. Type A intraclass correlation coefficients using an absolute agreement definition.
- c. This estimate is computed assuming the interaction effect is absent, because it is not estimable otherwise.

7.5.3 Usability and Acceptance

The graph below in Figure 7-5 shows the responses from the CBUQ.

Question Map:

Q1: Learning to operate the Aerobic Fitness App would be easy for me

Q2: I would find it easy to get the Aerobic Fitness App to do what I want it to do

Q3: My interaction with the Aerobic Fitness App would be clear and understandable

Q4: I would find the Aerobic Fitness App to be flexible to interact with

Q5: It would be easy for me to become skilful at using the Aerobic Fitness App

Q6: I would find the Aerobic Fitness App easy to use

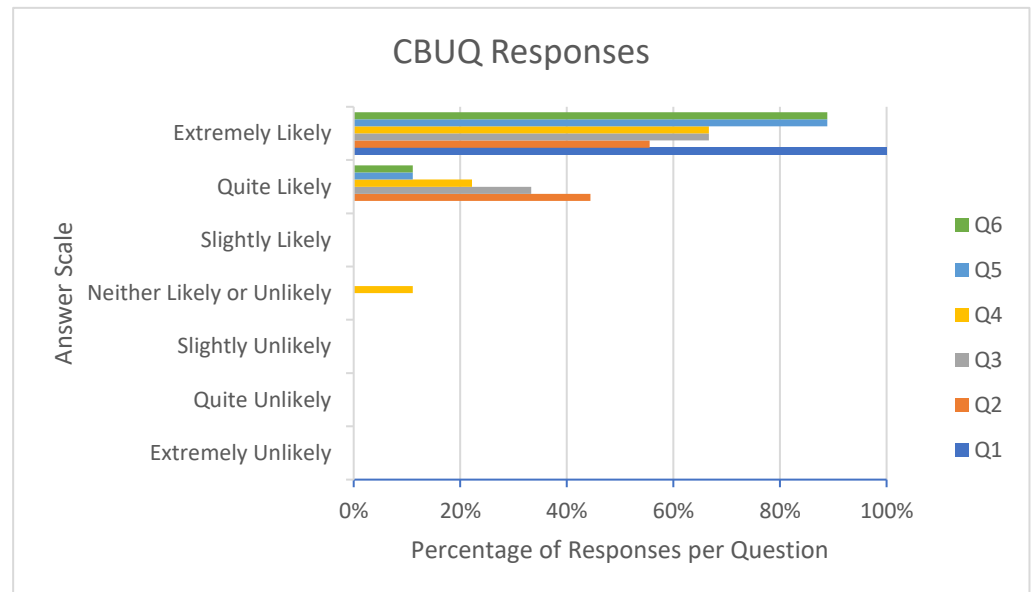


Figure 7-5 – bar graph showing response results from Component-Based Usability Questionnaire

The overall score of the CBUQ was 6.74, as shown in Figure 7-6, which is above the recommended norm of 5.29.

6MWTest						
Questionnaire Responses						
Participant ID:	Question1	Question2	Question3	Question4	Question5	Question6
1	7	7	7	7	7	7
2	7	7	7	7	7	7
3	7	7	7	7	7	7
4	7	7	7	7	7	7
5	7	7	7	7	7	7
6	7	6	6	4	6	7
7	7	6	7	7	7	6
8	7	6	6	6	7	7
9	7	6	6	6	7	7
Score (Avg of all answers): 6.740740741						

Figure 7-6 – question responses from Component-based Usability Questionnaire

7.6 Discussion of Results

The aim of this evaluation was to prove that the result returned by the GPS distance tracker in the application was accurate and comparable to the result returned from the original 6-minute walk test. Furthermore, the aim was to show that the application results are repeatable after

continued use of the 6-minute walk test. Lastly, the aim included showing that the usability and acceptance of the application was good.

7.6.1 Accuracy and Repeatability

The results have shown that the distance tracked with the 6MWTest application is very good compared to the results of a traditional 6-minute walk test. Although the accuracy of the GPS distance tracker isn't 100% accurate (as Pearson correlation was .89) this result is still more than acceptable. Results further demonstrated that the test-re-test-reliability of the application is very good.

Combining the accuracy results and the test-re-test-reliability results, we can observe that there is a possibility that the application will produce a margin of error with the distance tracked compared to what the actual distance would be. Furthermore, we can observe that the margin of error will not be significant and will be repeatable. These results show that the application is likely to provide results that aren't truly accurate to what the user has travelled, but the very strong repeatability of the application indicates that the inaccuracy of the result will hold for further tests. Thus, the user can continue to track their progress of the 6-minute walk tests using the mobile application.

Although these results are positive, they are based on a small sample size of participants in one controlled environment. The application is readily usable in any environment that provides a GPS signal therefore further evaluation could be carried out to provide a higher confidence level in the accuracy of the application. Evaluation participants were given the opportunity to use the application prior to their time slot but they had no prior experience with the 6-minute walk test. Further experience with the test and the application could lead to an increase in their walking distance due to the participant feeling more confident with the test procedure. Due to this observation, the test results of the participants may increase significantly from the first tests to recent test afterwards before potentially stagnating. Again, further evaluation could be carried out with a larger participant group consisting of people with mixed experience of the 6-minute walk test. The age-group and lifestyle of the participants is also a factor to consider with this evaluation. All participants were aged between 21-30 years and could walk for 6-minutes with ease. An older or mixed group of participants with differing abilities would be beneficial to further test the accuracy and repeatability of the application.

7.6.2 Usability

The results from the CBUQ showed that participants perceived the usability of the application to be comparable to a highly easy to use application. Although, this result is based on an even smaller sample size than the accuracy and repeatability tests since each participant only had one CBUQ to complete.

Further evaluation could be carried out in this section to prove the usability of the 6MWTest application is good. Rather than have the participants only evaluate the 6MWTest application, they could also evaluate and fill in a CBUQ for related applications such as FC. These results can then be compared to prove that the 6MWTest application is an easier to use application than others that perform the same task. Furthermore, the participant group were all comfortable with using technology and mobile applications. Other individuals may not be as comfortable with this technology and could possibly require further assistance to use the application. A participant group of mixed capabilities and ages would be beneficial for further testing the application's usability.

7.7 Summary of Evaluation

In the first section of the results in this evaluation it was shown that the GPS distance tracker within the application is very strongly accurate and acceptable for the task required. Secondly, it was shown that the GPS distance tracker within the application is very strongly repeatable. Using the results from these two sections, it was demonstrated that the GPS distance tracker within the application is acceptable for carrying out and tracking the progression of a user's 6-minute walk tests. Lastly, the application as a whole was shown to be easily usable, with a clear, clean, and understandable layout. Putting the results of each section of the evaluation together, the 6MWTest application created as part of this project was found to be comparable to the original 6-minute walk test. Although, further research with a larger participant group with differing abilities would provide a more confident conclusion.

Chapter 8. Summary and Conclusion

8.1 Summary

The application produced during the course of this project has met the initial and additional requirements set. Missing functionality from related work analysed has been implemented in the application, giving a more robust and complete mobile 6-minute walk test.

Developing the project with an agile approach has enabled additional features to be added without any redesign, and enabled quick development. This approach also allowed the project plan to be adhered to which meant each stage of the process ran smoothly.

The project has been developed using modern technologies that are easily maintainable and allow portability for future work.

Third party volunteers have helped to ensure the user interface and usability of the application met the standards expected and as such, the overall look and feel of the application has been deemed highly easy to use. Furthermore, the application has been deemed very strongly accurate and very strongly repeatable in its task to measure the distance travelled using mobile GPS. Although, further analysis could be carried out for this stage.

Overall, the 6MWTest application developed during this project is ready to be used in the field as a replacement for the traditional 6-minute walk test.

8.2 Limitations

Some limitations of the application have to be considered:

- At times when the connectivity tracker is running, the 'no connection' box will stay present even when the device has re-established a connection to the network. Closing the application down and opening it again or turning the connection on and off will fix this issue, however it is still present. It should be noted that this is a common issue amongst most mobile applications but an issue that could be looked at nonetheless for future enhancements.
- The results found during the evaluation stage were adequate enough to draw conclusions that the application results are comparable to a traditional 6-minute walk test, but it can't be said with too highly a confidence that the application is truly comparable. Further evaluation with different participant groups and in different environments would allow

more definitive analysis of the accuracy, repeatability, and usability of the 6MWTest application.

- It was noticed during evaluation that the kg to stone conversion was off for certain weight values. This was demonstrated that for some values the conversion was off by 1lb. Selecting 10st, 11lbs and converting to kg would come out at 68kg (rounded by the application). The user then navigated back to their profile page at a later time and changed the weight unit back to stone. The conversion this time showed their weight to be 10st, 10lbs. This is correct for the conversion from 68kg but the problem arises from the initial rounding of the kg value. If the kg were to be stored with the decimal point then the conversion would come out as 10st, 11lbs.
- Forename and surname are mandatory fields when registering but it was noted that when updating the profile page, the values in these fields can be left blank and saved as such. This does not affect the user's results in any way and the user can still easily be identified either by their id or their email.

8.3 Future Work

8.3.1 Offline Mode

A feature that would have been good to fit in to the development stage would be allowing the application to work offline. Local storage could be utilised as a means to hold a cached version of the results and previous results and then saved to the database when a connection was re-established. Thankfully, this feature would be easy to implement with the current infrastructure and would enhance the application greatly.

8.3.2 Noise Alerts in Silent Mode

Although the application does emit a noise and vibration to the user when a test is over, this relies on the device not being on silent mode. There are ways to enable sound and vibration even when device is on silent mode. Enabling this would give the user a lot more freedom to continue walking without having to check if the application is still running or not.

Furthermore, it would enhance the accessibility of the device for users who may be visually or hearing impaired.

8.3.3 Profile Page Design

Due to the focus being primarily on the results and GPS tracker features, some features of the application could have been tested more thoroughly. The limitations with the profile page would have been picked up if more time was spent testing this feature. The profile page

would also benefit from having cleaner user inputs for certain fields such as the height and weight selectors.

8.3.4 iOS Version

An iOS version of the application would mean the application could be used with a much wider pool of users. Since the application has been developed as a hybrid-application, this is certainly nowhere near as infeasible as a completely new application would be. However, due to some of the plugins used, it would still take considerable work to ensure the application was fully compatible with running on iOS before creating an Ionic build from the current version.

8.3.5 Accessible Data for Researchers

The current methods for accessing the data from a research point of view would entail querying the database either through a console or through the phpMyAdmin web interface. This is not an ideal scenario for researchers who may not be skilled with SQL and is not a time-efficient way of retrieving data in general.

A future enhancement could be to create a web application that is viewable by the researcher and outlines the data retrieved from the results. Possible ideas would be to give the data filters and different exportable formats to provide the researcher with the freedom required to analyse the data.

8.3.6 Pedometer

Although the GPS tracker within the application has proven to be accurate, it could be optional for the user to either use GPS to measure their distance or a pedometer to measure their distance. This would allow the user to use the 6MWTest application indoors and outdoors, at the user's leisure. Current methods for counting steps on mobile devices aren't quite as accurate as GPS, depending on the implementation and the device's hardware but it is still achievable as a future enhancement.

References

- American Thoracic Society, 2002. *atsjournals*. [Online]
Available at: <https://www.atsjournals.org/doi/pdf/10.1164/ajrccm.166.1.at1102>
[Accessed 26 02 2018].
- Atlassian, 2018. *Home*. [Online]
Available at: <https://bitbucket.org>
[Accessed 12 02 2018].
- Atlassian, 2018. *Home*. [Online]
Available at: <https://trello.com/>
[Accessed 09 01 2018].
- Auth0, 2018. *JsonWebToken*. [Online]
Available at: <https://github.com/auth0/node-jsonwebtoken>
[Accessed 19 01 2018].
- ChartJS, 2018. *Home*. [Online]
Available at: <http://www.chartjs.org/>
[Accessed 16 01 2018].
- Dali, S., 2014. *Calculate distance between two latitude-longitude points? (Haversine formula)*. [Online]
Available at: <https://stackoverflow.com/a/21623206>
[Accessed 03 01 2018].
- Davis, F. D., 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3), pp. 319-340.
- Drifty Co., 2018. *Background Mode*. [Online]
Available at: <https://ionicframework.com/docs/native/background-mode/>
[Accessed 03 02 2018].
- Drifty Co., 2018. *Diagnostic*. [Online]
Available at: <https://ionicframework.com/docs/native/diagnostic/>
[Accessed 08 01 2018].

Drifty Co., 2018. *Geolocation*. [Online]

Available at: <https://ionicframework.com/docs/native/geolocation/>

[Accessed 09 12 2017].

Drifty Co., 2018. *ionicframework*. [Online]

Available at: <https://ionicframework.com/>

[Accessed 23 10 2017].

Drifty Co., 2018. *Location Accuracy*. [Online]

Available at: <https://ionicframework.com/docs/native/location-accuracy/>

[Accessed 09 01 2018].

Drifty Co., 2018. *Network*. [Online]

Available at: <https://ionicframework.com/docs/native/network/>

[Accessed 14 02 2018].

Goode, T., 2018. *Node.js CORS middleware*. [Online]

Available at: <https://github.com/expressjs/cors>

[Accessed 18 03 2018].

Google, 2018. *Home*. [Online]

Available at: <https://angular.io/>

[Accessed 20 03 2018].

IBM, 2018. *IBM SPSS Software*. [Online]

Available at: <https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software>

[Accessed 20 02 2018].

Jetbrains, 2018. *idea*. [Online]

Available at: <https://www.jetbrains.com/idea/>

[Accessed 02 11 2017].

justinmind, 2018. *Home*. [Online]

Available at: <https://www.justinmind.com/>

[Accessed 07 11 2017].

KH, C., 1986. A means of assessing maximal oxygen intake. Correlation between field and treadmill testing.. *JAMA*, p. 203:201–204.

mozilla, 2018. *HTML5*. [Online]

Available at: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>

[Accessed 20 03 2018].

Netflix, 2014. *Netflix technology blog*. [Online]

Available at: <https://medium.com/netflix-techblog/node-js-in-flames-ddd073803aa4>

[Accessed 18 03 2018].

Node.js Foundation, 2018. *Home*. [Online]

Available at: <https://nodejs.org/en/>

[Accessed 18 11 2017].

Node.js Foundation, 2018. *Home*. [Online]

Available at: <https://expressjs.com/>

[Accessed 23 11 2017].

Oracle, 2018. *Why MySQL?*. [Online]

Available at: <https://www.mysql.com/why-mysql/>

[Accessed 18 03 2018].

Picciolo, S., 2013. *Six-Min Walk Test*. [Online]

Available at: https://play.google.com/store/apps/details?id=com.stepic.sixminwt&hl=en_GB

[Accessed 09 10 2017].

PuTTY, 2018. *Home*. [Online]

Available at: <https://www.putty.org/>

[Accessed 07 10 2017].

ReactiveX, 2018. *Rxjs*. [Online]

Available at: <http://reactivex.io/rxjs/>

[Accessed 04 01 2018].

S.R.L, M., 2016. *6 Minutes Walking Test*. [Online]

Available at: https://play.google.com/store/apps/details?id=com.medapps.TM6M&hl=en_GB

[Accessed 09 10 2017].

Sass, 2018. *Home*. [Online]

Available at: <https://sass-lang.com/>

[Accessed 20 03 2018].

Scott Morrison

Solutions, S. M. S., 2015. *6MW Test - Functional Capacity*. [Online]
Available at: <https://play.google.com/store/apps/details?id=com.walkapp&hl=en>
[Accessed 10 10 2017].

Solway S, B. D. L. Y. T. S., 2001. *A qualitative systematic overview of the measurement properties of functional walk tests used in the cardiorespiratory domain.*, s.l.: Chest..

Source, O., 2018. *bcrypt*. [Online]
Available at: <https://www.npmjs.com/package/bcrypt>
[Accessed 02 02 2018].

stackoverflow, 2018. *Frameworks, Libraries, and Tools*. [Online]
Available at: <https://insights.stackoverflow.com/survey/2018/#technology-frameworks-libraries-and-tools>
[Accessed 08 03 2018].

StatsTutor, 2018. *Spearman's correlation*. [Online]
Available at: <http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf>
[Accessed 05 03 2018].

The Apache Software Foundation, 2018. *Home*. [Online]
Available at: <https://cordova.apache.org/>
[Accessed 20 03 2018].

tutorialspoint, 2018. *MVC Framework - Introduction*. [Online]
Available at:
https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
[Accessed 20 03 2018].

Vega, 2017. *Timer countdown Angular 2*. [Online]
Available at: <https://stackoverflow.com/a/46316755>
[Accessed 07 01 2018].

Venesse, C., 2002. *Calculate distance, bearing and more between Latitude/Longitude points*. [Online]
Available at: <https://www.movable-type.co.uk/scripts/latlong.html>
[Accessed 03 01 2018].

verywellfit, 2018. *PAR-Q (Physical Activity Readiness Questionnaire) for Safe Exercise*.

[Online]

Available at: <https://www.verywellfit.com/physical-activity-readiness-questionnaire-3120277>

[Accessed 18 11 2017].

W3C, 2018. *Geolocation API Specification 2nd Edition*. [Online]

Available at: <https://www.w3.org/TR/geolocation-API/>

[Accessed 18 01 2018].

Willem-Paul, 2011. *Questionnaires*. [Online]

Available at: http://mmi.tudelft.nl/willem-paul/index.php/Questionnaires#Component-Based_Usability_Questionnaire_.28CBUQ.29

[Accessed 21 11 2017].

Appendix A

Evaluation Results

Tests of Normality for App Results and Manual Results

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
App_results	.165	18	.200*	.864	18	.014
Manual_results	.130	18	.200*	.930	18	.191

Figure 0-1 – output of normality test for application results and manual results

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Tests of Normality for App Results 1 and App Results 2

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
App_result_1	.222	9	.200*	.888	9	.192
App_result_2	.199	9	.200*	.857	9	.088

Figure 0-2 – output of normality test for application result set 1 and application result set 2

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Appendix B

Component-Based Usability Questionnaire

6MWTest - Aerobic Fitness App Questionnaire

Thank you for taking part in the investigation of the 6MWTest - Aerobic Fitness App. The final stage of the investigation is a short questionnaire that will allow the researcher to gauge the usefulness and usability of the implemented mobile application.

Please answer the following questions honestly. If you have any questions then feel free to ask.

Statement	Unlikely	1	2	3	4	5	6	7	Likely
		extremely	quite	slightly	neither	slightly	quite	extremely	
Learning to operate the Aerobic Fitness App would be easy for me		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
I would find it easy to get the Aerobic Fitness App to do what I want it to do		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
My interaction with the Aerobic Fitness App would be clear and understandable		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
I would find the Aerobic Fitness App to		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	

be flexible to interact with								
It would be easy for me to become skilful at using the Aerobic Fitness App	0	0	0	0	0	0	0	0
I would find the Aerobic Fitness App easy to use	0	0	0	0	0	0	0	0

Appendix C

Physical Activity Readiness Questionnaire

6MWTTest - Aerobic Fitness App

Pre-screening (PARQ: Physical Activity Readiness Questionnaire)

Please use the text highlighter tool to answer Yes or No to each of the following questions:

1. Has your doctor ever said that you have a heart condition and that you should only do physical activity recommended by a doctor?
Yes ☐ **No** ☐
2. Do you feel pain in your chest when you do physical activity?
Yes ☐ **No** ☐
3. In the past month, have you had chest pain when you were not doing physical activity?
Yes ☐ **No** ☐
4. Do you lose your balance because of dizziness or do you ever lose consciousness?
Yes ☐ **No** ☐
5. Do you have a bone or joint problem that could be made worse by a change in your physical activity?
Yes ☐ **No** ☐
6. Is your doctor currently prescribing drugs (for example, water pills) for your blood pressure or heart condition?
Yes ☐ **No** ☐
7. Do you know of any other reason why you should not do physical activity?
Yes ☐ **No** ☐

If you have responded with “yes” to any of the above questions then you won’t be able to participate in this investigation.

(PRINT NAME)	
Signature of Participant:	Date:

Appendix D

Project Plan

Semester 1

Week 6 – Week 7

- Research the location tracking and graph library technologies in both the Ionic framework and Android native.
- Decide on which approach to take non-native vs native.
- Contact systems support to setup mySql instance

Week 8 – Week 9

- Create more detailed requirements specification.
- Create class diagram for database.

Week 10 – Week 12

- Create project design diagrams including wireframes and meet with supervisor to refine these
- Create initial prototype app with basic wireframe layout
- Create project poster

Xmas Break

- Create DB tables, relationships and insert prototype data
- Begin implementation of application

Semester 2

Week 0

- Project poster submission and presentation

Week 1 – Week 6

- Continue implementation of application

Week 6 – Week 8

- Evaluation and testing of application

Week 8 – Week 11

- Finalize report

The above plan includes adding to my report every fortnight as well as weekly meetings with supervisor.

Project Scope Methodology

The methodology to be used during the project will predominantly be based on an Agile approach. I will look to follow one week sprints with specific features as the sprint goal. After each sprint I will be producing a new prototype of the application which will contain the new features from the previous sprint. The prototype can then be fine-tuned after each sprint with the end goal being a fully functioning app. I will be meeting with my supervisor every week and during these meetings we will go over the prototype in order to ensure requirements and design are being met correctly. I also plan on involving friends in the User Experience and User Interface fields during sprints where I will seek their feedback and improve the app based on the responses.

My tasks will be split up in to three categories: open, in progress, and complete. The idea is that the tasks that are in progress should be those that are being implemented in the current sprint, the ones that are open are those that have still to be implemented, and the closed tasks are those that have been completed and approved. I will also hold a backlog of less important tasks that can be added at the end if there is sufficient time.

The requirements will be mostly gathered at the start as well as designing the majority of the app's look and feel so it is not strictly Agile, however I will leave room for these in each weekly meeting to be fine-tuned.

For version control I will utilise Atlassian's BitBucket, following a branch structure of:

Master (the current complete working version of the app)

Develop (The state of the application as it's being worked on in each sprint)

Feature-X (where X is a feature belonging to the current sprint)

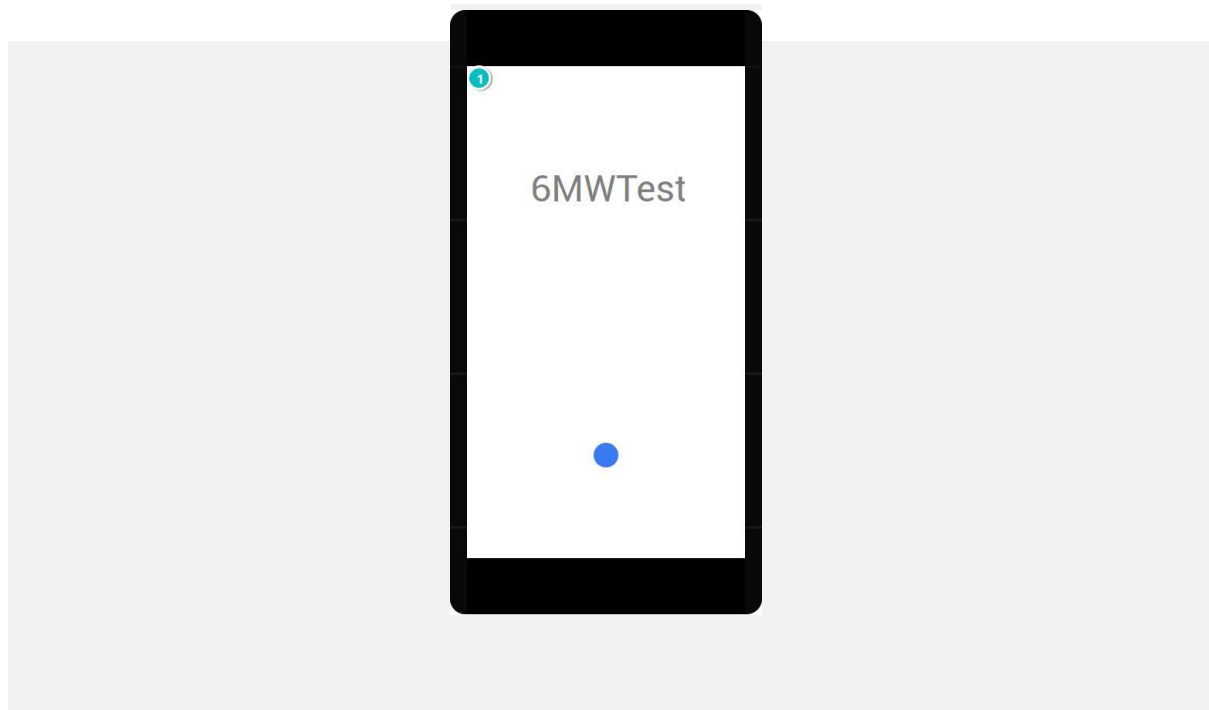
Appendix E

Justinmind Initial Wireframes

02. Screens / [initial_wireframe](#)



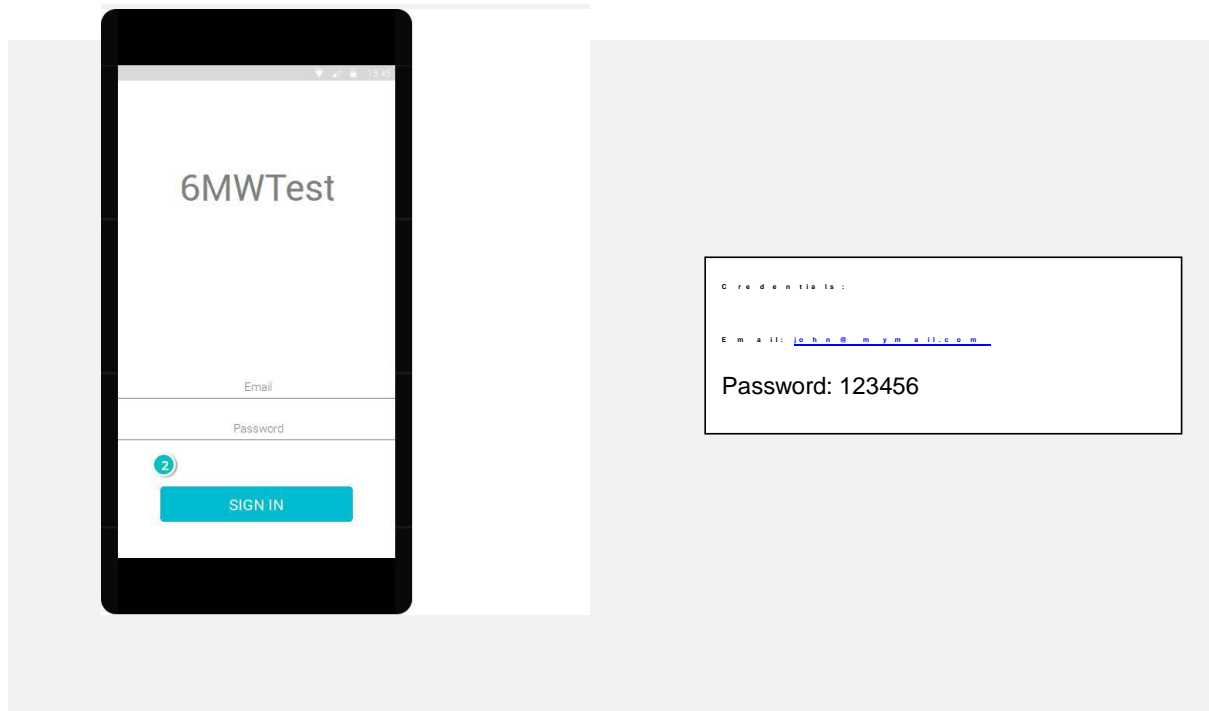
Splash screen



Interactions

- 1 on Page Load: goes to 'login screen' with effect: fade →

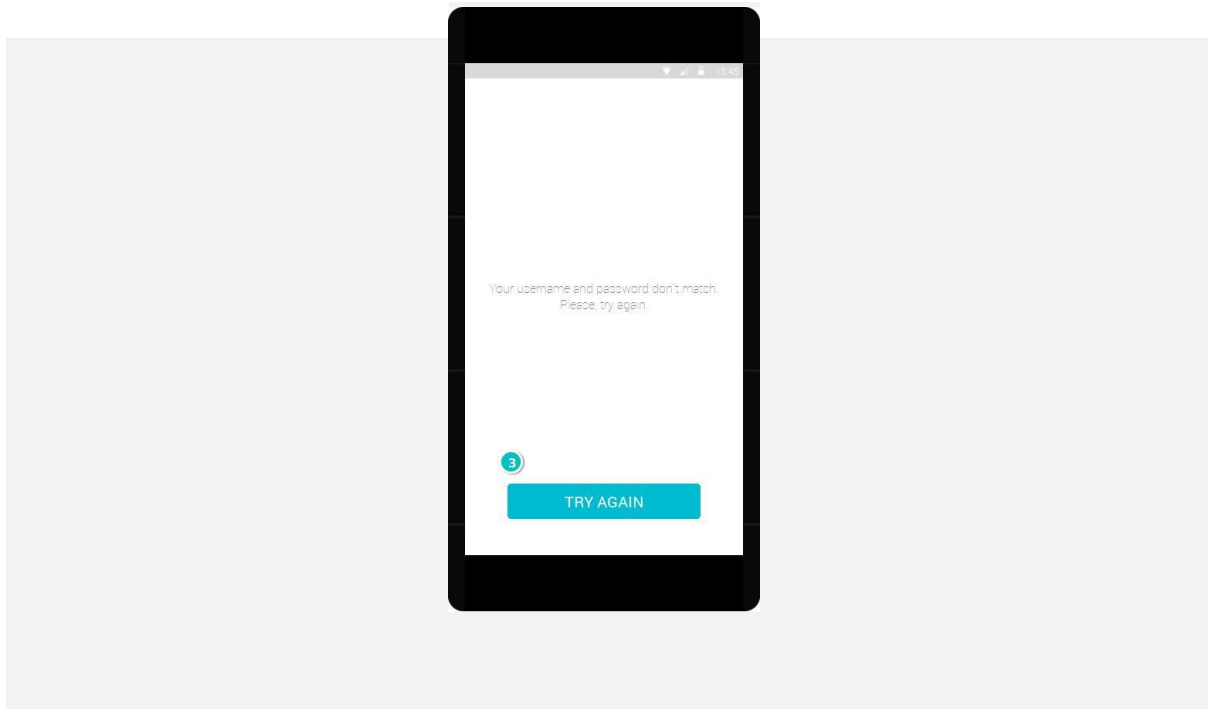
login screen



Interactions

- 2 **on Click:** When ((Input_3.value = 'john@mymail.com') and (Input_2.value = '123456')) goes to 'home' → Else goes to 'Error screen' with effect: slide left →

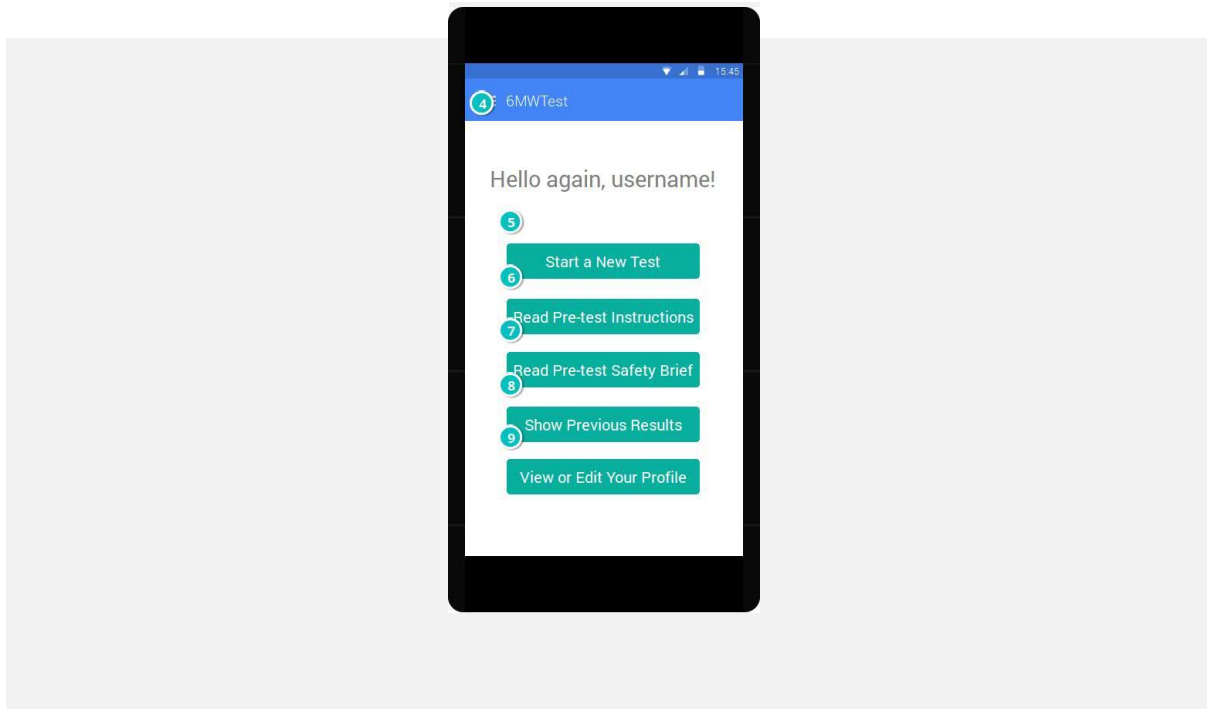
Error screen



Interactions

- 3 on Click: goes to 'login screen' with effect: slide right →

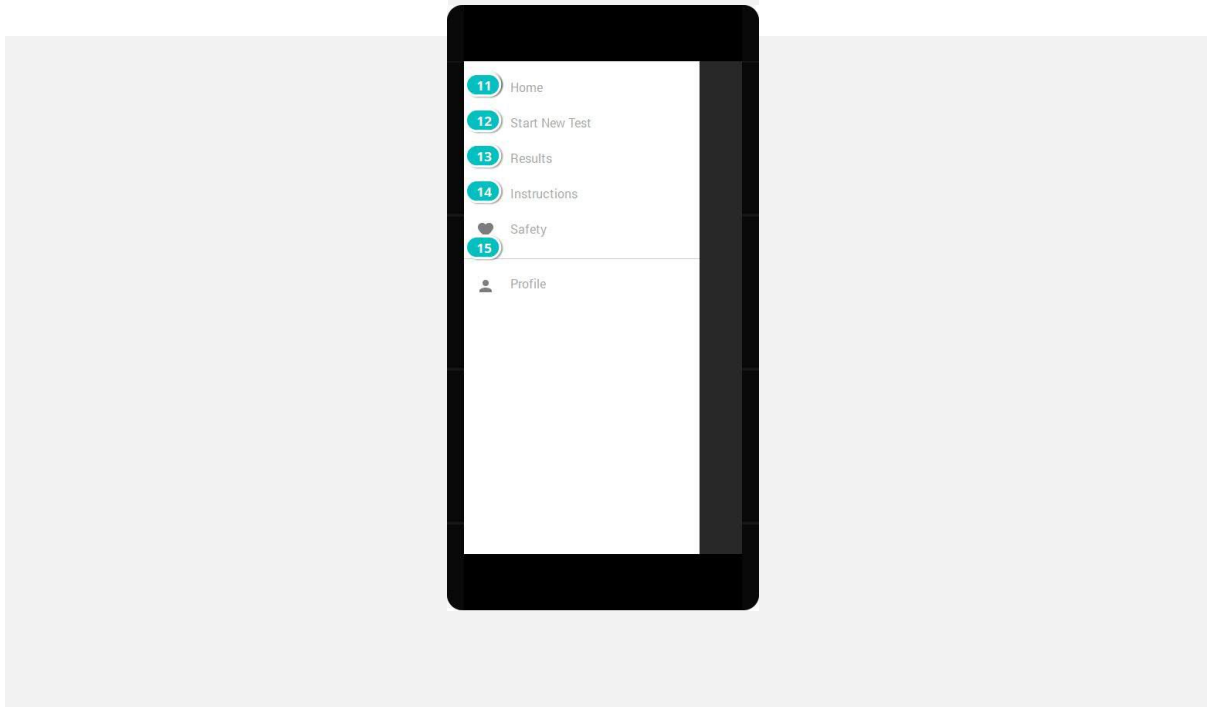
home



Interactions

- 4 on Click: goes to 'menu' →
- 5 on Click: goes to 'test' → goes to 'test' →
- 6 on Click: goes to 'instructions' →
- 7 on Click: goes to 'safety' →
- 8 on Click: goes to 'results' →
- 9 on Click: goes to 'profile' →

menu



Interactions

10 on Click: goes to 'home' →

11 on Click: goes to 'test' →

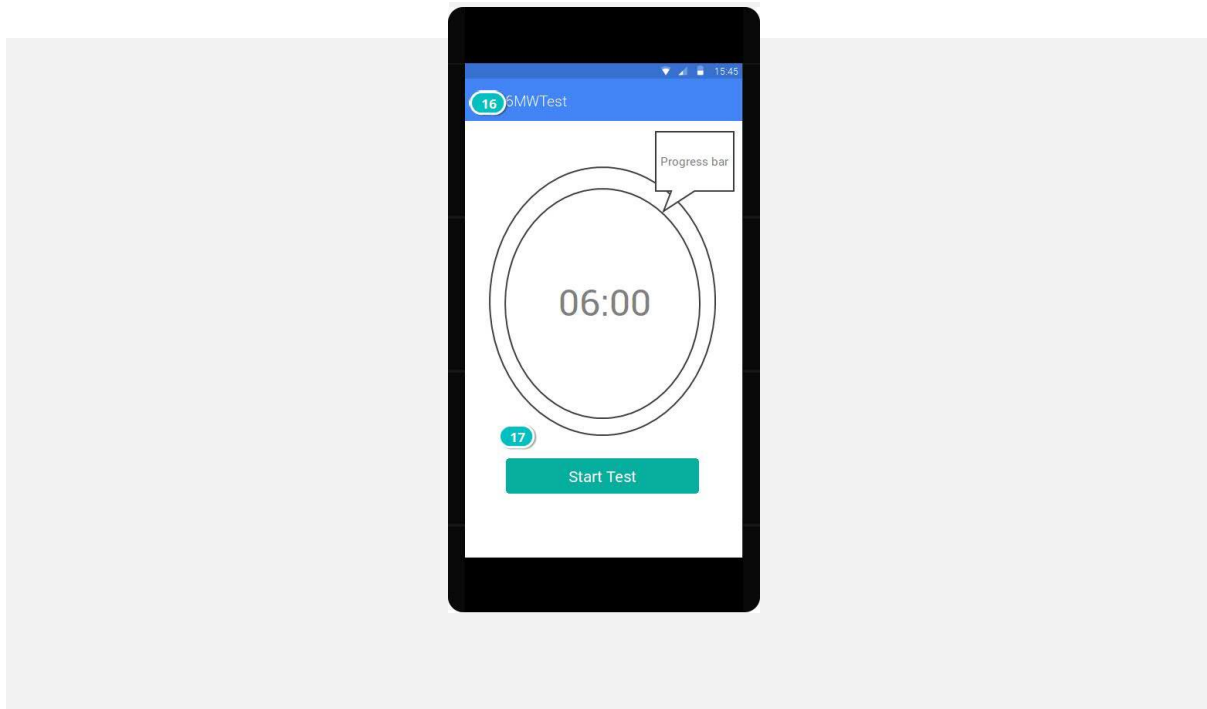
12 on Click: goes to 'results' →

13 on Click: goes to 'instructions' →

14 on Click: goes to 'safety' →

15 on Click: goes to 'profile' →

test

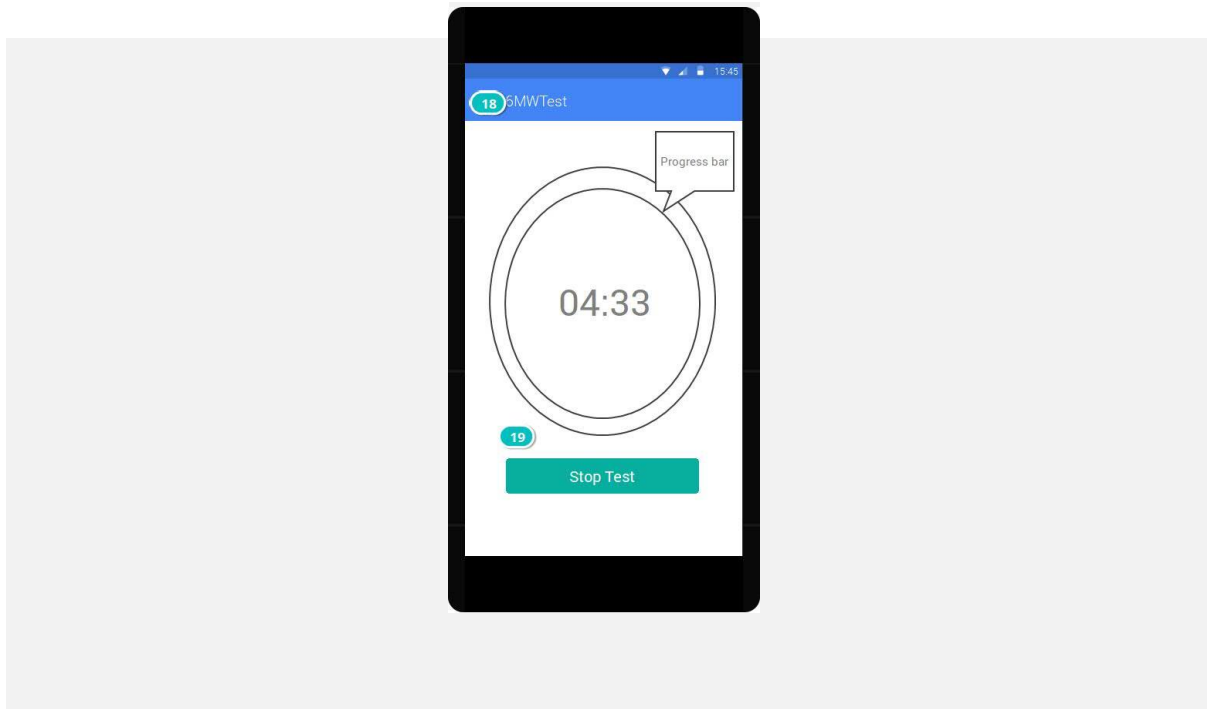


Interactions

16 on Click: goes to 'menu' →

17 on Click: goes to 'stop-test' →

stop-test

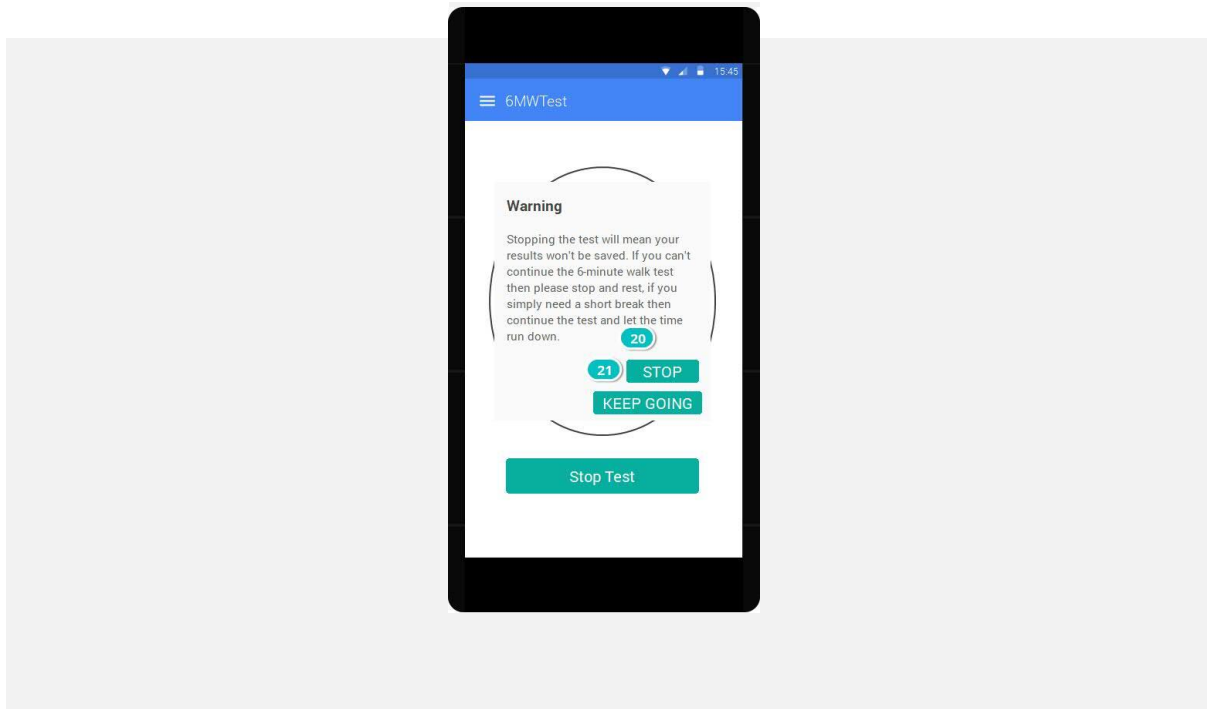


Interactions

18 on Click: goes to 'menu' →

19 on Click: goes to 'stop-test-message' →

stop-test-message

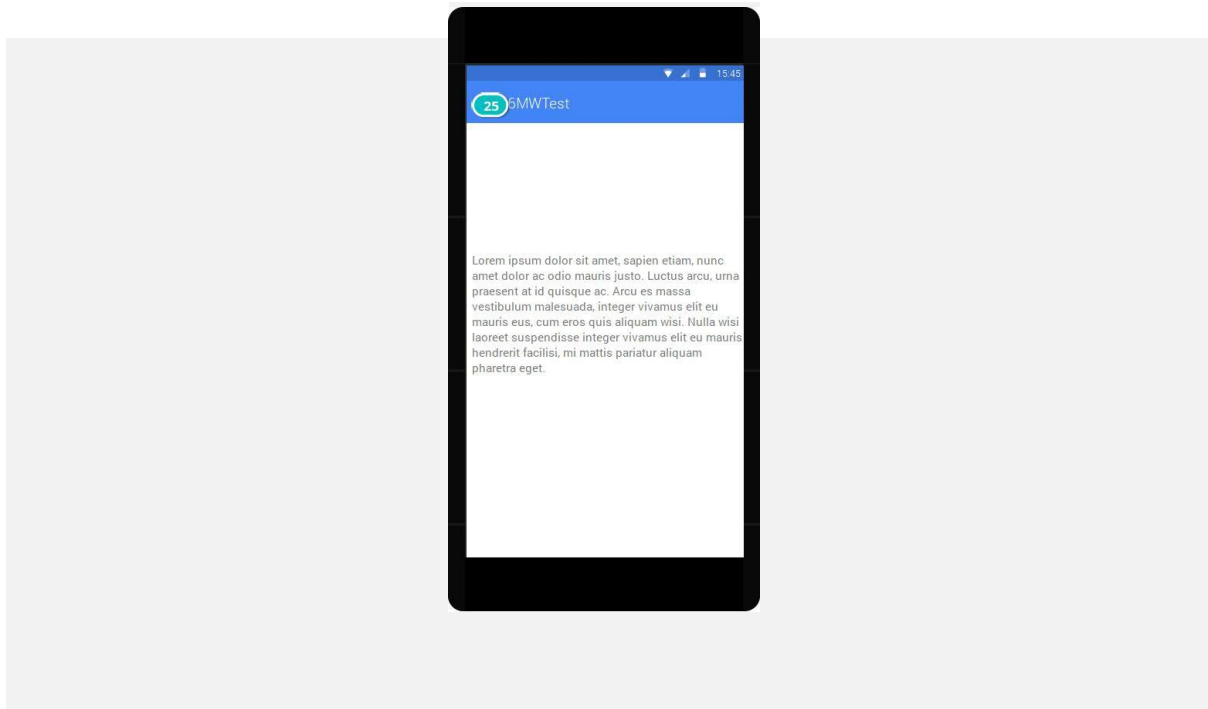


Interactions

20 on Click: goes to 'home' →

21 on Click: goes to 'stop-test' →

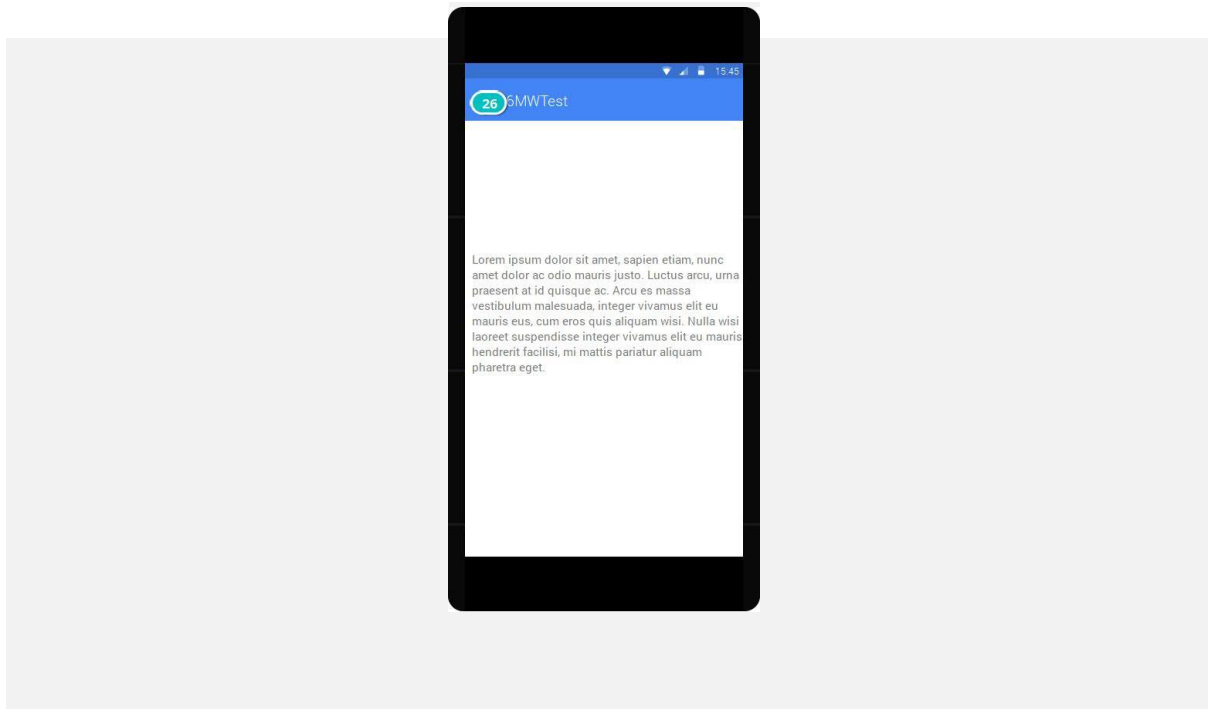
instructions



Interactions

25 on Click: goes to 'menu' →

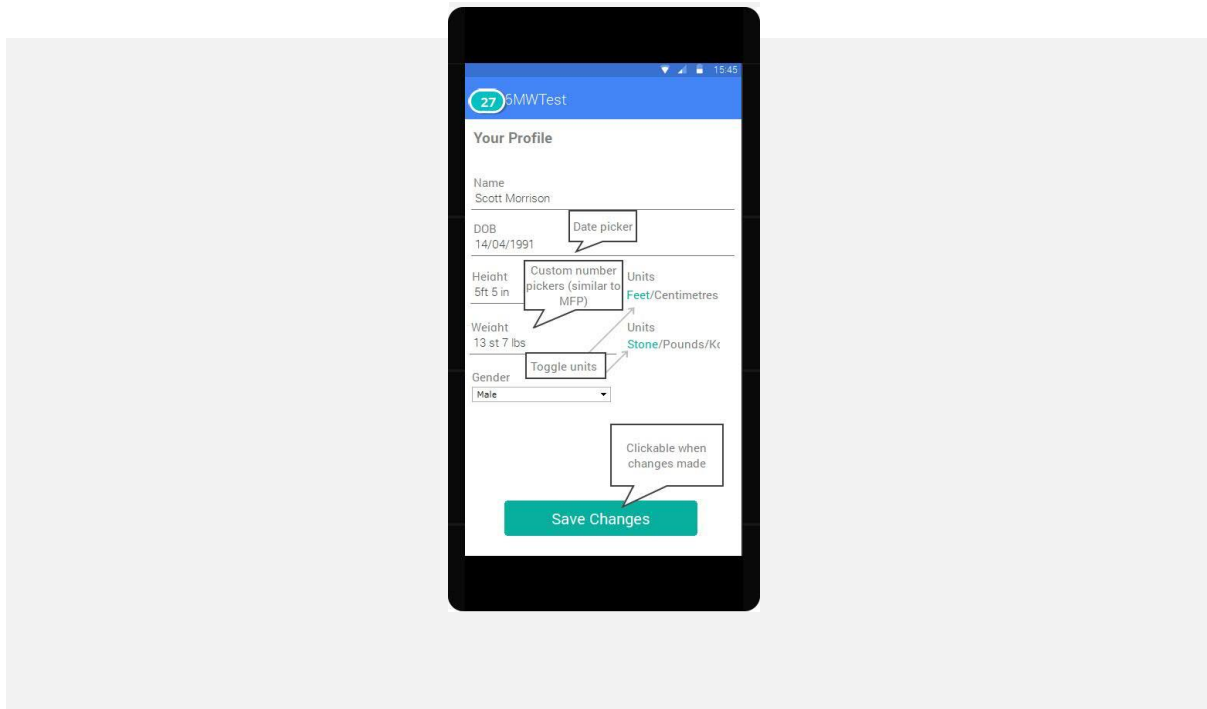
safety



Interactions

26 on Click: goes to 'menu' →

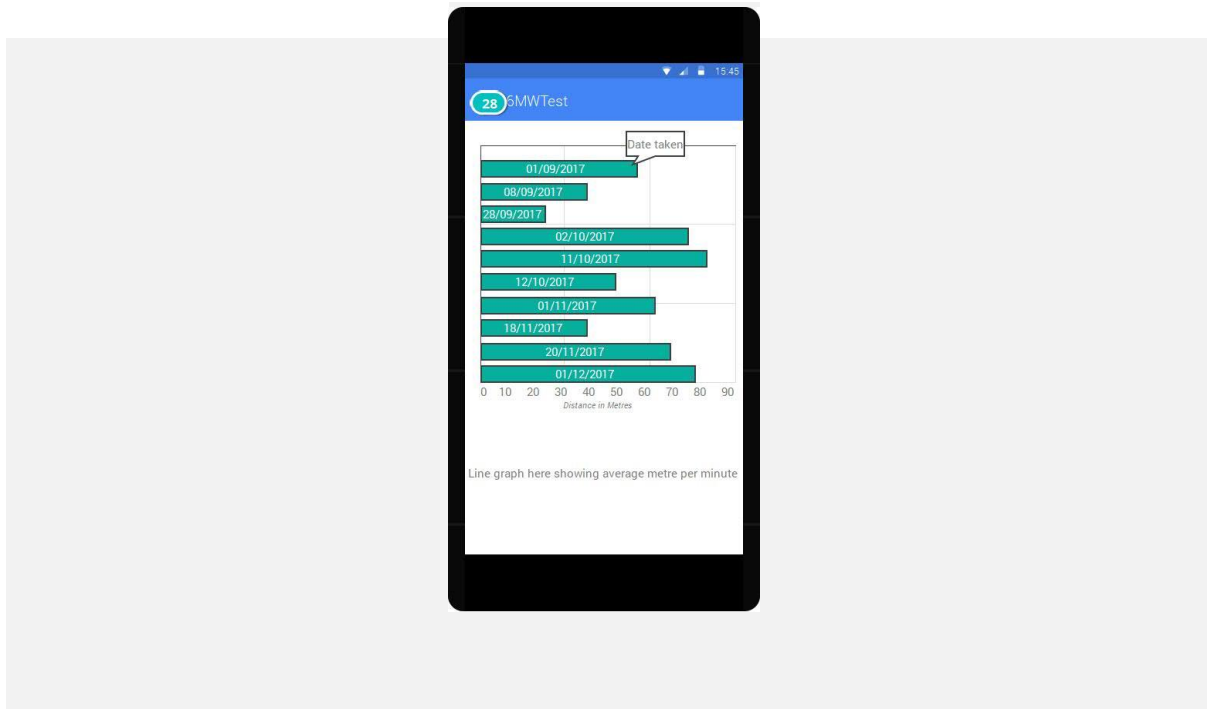
profile



Interactions

27 on Click: goes to 'menu' →

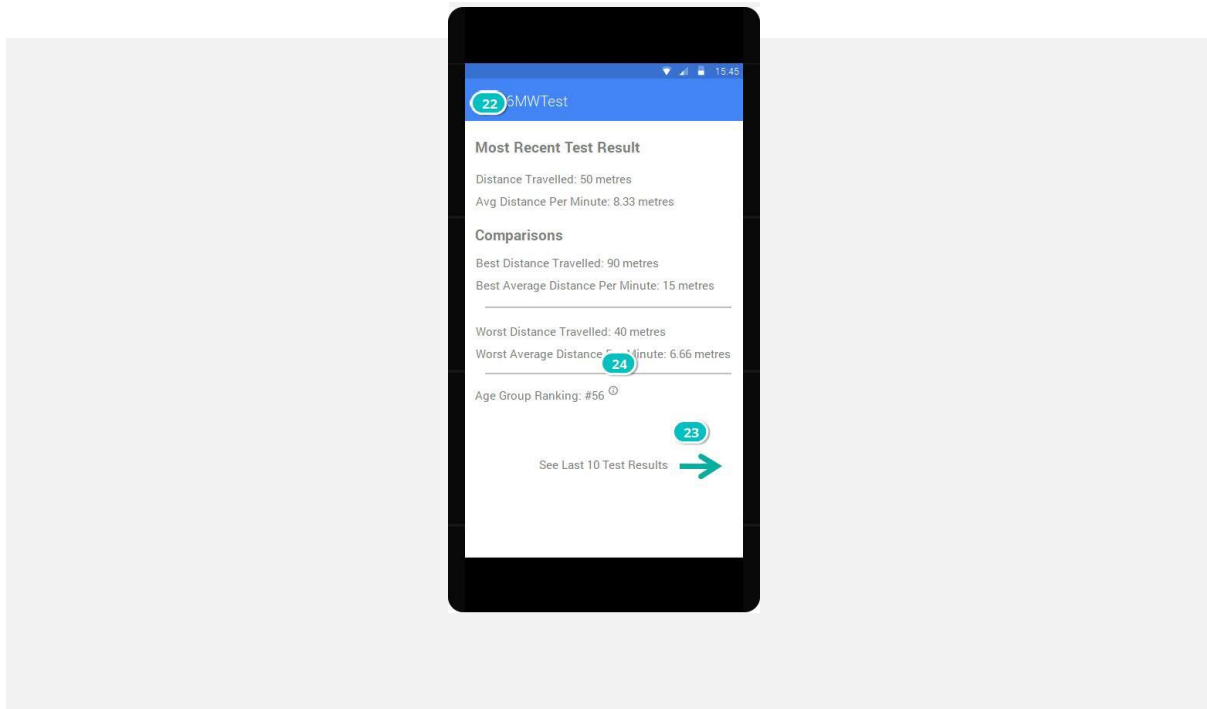
results-graphs



Interactions

28 on Click: goes to 'menu' →

results



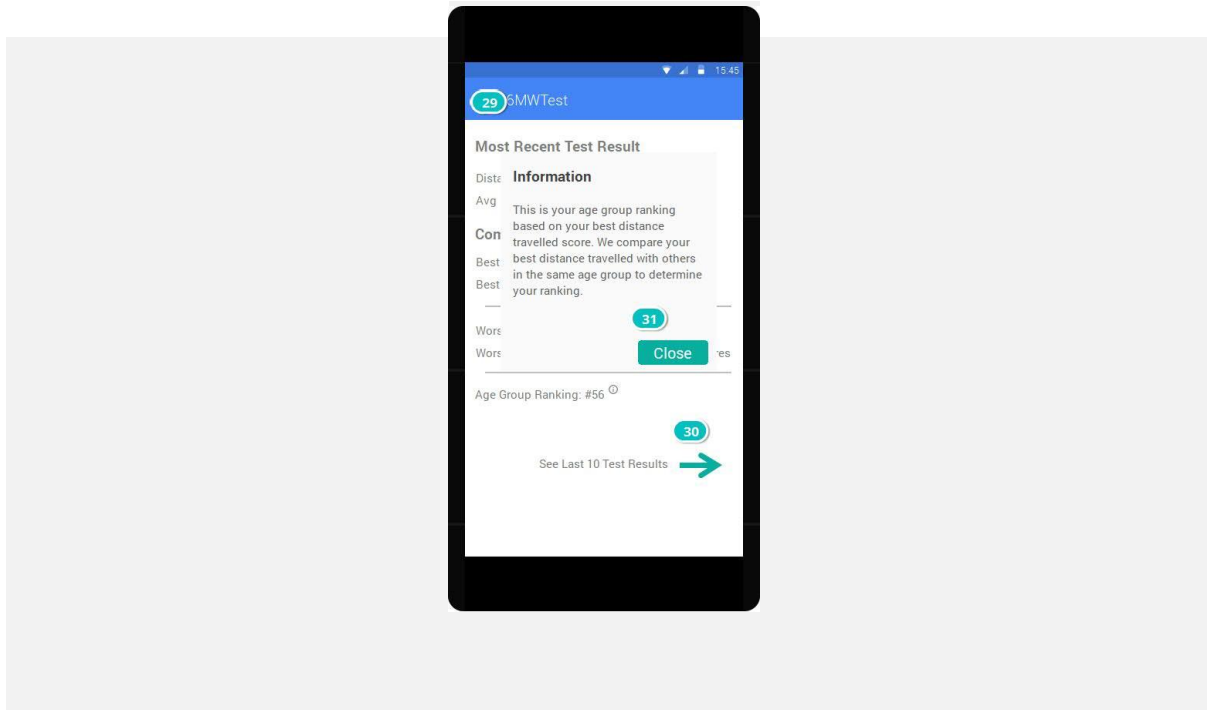
Interactions

22 on Click: goes to 'menu' →

23 on Click: goes to 'results-graphs' →

24 on Click: goes to 'results-info' →

results-info



Interactions

29 on Click: goes to 'menu' →

30 on Click: goes to 'results-graphs' →

31 on Click: goes to 'results' →

Wireframe Produced From UI/UX JPMorgan Meeting



Appendix G

Feature Test Cases

User Account Tests

Feature: Account Register				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Register for a new account using valid details	Forename: "Test" Surname: "Case" DOB: select 31,12,1990 Email: test@case.com password: "password"	Alert: "Account created. Please now login." User added to DB.	Alert: "Account created. Please now login." User added to DB.	As expected
Register for a new account using invalid email format	Forename: "Test" Surname: "Case" DOB: select 31,12,1990 Email: test@case. password: "password"	Alert: "Invalid email address entered. Please enter a valid email." No user added to DB.	Alert: "Invalid email address entered. Please enter a valid email." No user added to DB.	As expected
Register for a new account using invalid email format	Forename: "Test" Surname: "Case" DOB: select 31,12,1990 Email: 12345 password: "password"	Alert: "Invalid email address entered. Please enter a valid email." No user added to DB.	Alert: "Invalid email address entered. Please enter a valid email." No user added to DB.	As expected
Register for a new account with existing user email	Forename: "Test" Surname: "Case" DOB: select 31,12,1990 Email: test@case.com password: "password"	Alert: "An account already exists with that email." No user added to DB.	Alert: "An account already exists with that email." No user added to DB.	As expected

Feature: Account Login				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Login with valid email and password.	Email: test@case.com Password: "password1"	Alert: "Logging you in..." > redirect to 'Home' page.	Alert: "Logging you in..." > redirect to 'Home' page.	As expected
Login with valid email and invalid password.	Email: test@case.com Password: "password"	Alert: "Invalid email or password provided."	Alert: "Invalid email or password provided."	As expected

		No page re-direct.	No page re-direct.	
Login with invalid email and any password	Email: test@invalidcase.com Password: "password"	Alert: "Invalid email or password provided." No page re-direct.	Alert: "Invalid email or password provided." No page re-direct.	As expected
Login with non-valid email format and any password.	Email: 12345 Password: "password"	Alert: "Invalid email or password provided." No page re-direct.	Alert: "Invalid email or password provided." No page re-direct.	As expected

Profile Page Tests

Feature: Profile				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Profile information should be correct after creating new user.	User creates new account with the following details: Forename: "Test" Surname: "Case" DOB: select 31,12,1990 Email: test@case.com password: "password" User logs in with registered email and password. User navigates to profile page.	Profile page shows: Forename: Test Surname: Case DOB: 31/12/1990 Height: 0 ft, 0 in Height Units: Feet Weight: 0 st, 0 lbs Weight Units: Stone Gender: none selected	Profile page shows: Forename: Test Surname: Case DOB: 31/12/1990 Height: 0 ft, 0 in Height Units: Feet Weight: 0 st, 0 lbs Weight Units: Stone Gender: none selected	As expected
Profile information should change after user update.	User changes profile info to show: Forename: Test1 Surname: Case1 DOB: 30/11/1991 Height: 5 ft, 5 in Height Units: Cm Weight: 10 st, 6 lbs Weight Units: Kg Gender: Male	Profile page shows: Forename: Test1 Surname: Case1 DOB: 30/11/1991 Height: 165 cm Height Units: Cm Weight: 66 kgs Weight Units:	Profile page shows: Forename: Test1 Surname: Case1 DOB: 30/11/1991 Height: 165 cm Height Units: Cm Weight: 66 kgs Weight Units:	As expected

	User logs out by pressing logout icon User logs back in using valid registered credentials. User navigates to profile page.	Kg Gender: Male User data updated in DB.	Kg Gender: Male User data updated in DB.	
Height should convert correctly after changing unit	User changes height unit from Cm to Feet	Height value shows: 5 ft, 5 in User data updated in DB.	Height value shows: 5 ft, 5 in User data updated in DB.	As expected
Weight should convert correctly after changing unit	User changes height unit from Kg to Stone	Weight value shows: 10 st, 6 lbs User data updated in DB.	Weight value shows: 10 st, 6 lbs User data updated in DB.	As expected

Network Tracker Tests

Feature: Network Tracker				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Test network tracker popup for losing connection	Open the application Navigate to home page Turn off all network connections on device	Popup at bottom of screen: "No Internet connection. Some data may be lost." Remains visible whilst connection is lost	Popup at bottom of screen: "No Internet connection. Some data may be lost." Remains visible whilst connection is lost	As expected
Test network tracker popup for network re-established	Open the application Navigate to home page Turn off all network connections on device Observe network lost popup Turn network connection back on	Loss of connection popup removed. New popup at bottom of screen: "No Internet connection. Some data may be lost." Disappears after 2 seconds.	Loss of connection popup removed. New popup at bottom of screen: "No Internet connection. Some data may be lost." Disappears after 2 seconds.	As expected

Safety and Instructions Dialogue Tests

Feature: Safety and Instructions Dialogue				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Safety and instructions shown automatically for user before carrying out first test	Register a new account using valid details. Login using new account details. Press 'Start Timer' button on home screen.	Safety and Instructions popup shown automatically.	Safety and Instructions popup shown automatically.	As expected
Safety and instructions not shown for user with previous test(s)	Login with the following credentials: Email: test@resultscase.com Password: "password" Press 'Start Timer' button on home screen.	Timer starts counting down without showing safety and instructions popup.	Timer starts counting down without showing safety and instructions popup.	As expected

Background Mode Tests

Feature: Background Mode				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Timer continues to count down whilst application is in the background.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started and note the time. Navigate away from application without shutting application down. Wait 10 seconds. Navigate back to application.	Timer shows 10 seconds less than noted time before closing.	Timer shows 10 seconds less than noted time before closing.	As expected
GPS continues to run whilst application is in the background.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started. Observe GPS icon in device's toolbar. Navigate away from application without shutting application down.	GPS icon in device's toolbar still visible.	GPS icon in device's toolbar still visible.	As expected
User is notified of background mode enabled	Navigate to the home page and press the 'Start Timer' button.	6MWTest icon observed in device's toolbar.	6MWTest icon observed in device's toolbar.	As expected

when application is in the background.	Observe timer has started. Navigate away from application without shutting application down.	Persistent notification observed in notification area. Main text: "6MWTest is running in the background."	Persistent notification observed in notification area. Main text: "6MWTest is running in the background."	
Timer continues to count down whilst device is locked.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started and note the time. Lock the device. Wait 10 seconds. Unlock the device.	Timer shows 10 seconds less than noted time before device lock.	Timer shows 10 seconds less than noted time before device lock.	As expected.
GPS continues to run whilst device is locked.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started. Observe GPS icon in device's toolbar. Lock the device. Wake the device.	GPS icon in device's toolbar still visible.	GPS icon in device's toolbar still visible.	As expected
User is notified of background mode enabled when device is locked.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started. Lock the device. Wake the device.	Persistent notification observed in notification area. Main text: "6MWTest is running in the background."	Persistent notification observed in notification area. Main text: "6MWTest is running in the background."	As expected.
Background mode disabled after application is back in foreground.	Navigate to the home page and press the 'Start Timer' button. Observe timer has started. Navigate away from application without shutting application down. Observe 6MWTest icon and notification in device's toolbar and notification area respectively. Navigate back to application.	6MWTest icon no longer visible in device's toolbar. Persistent notification no longer visible in notification area. Timer continuing as normal.	6MWTest icon no longer visible in device's toolbar. Persistent notification no longer visible in notification area. Timer continuing as normal.	As expected
Background mode disabled after test has complete.	Navigate to the home page and press the 'Start Timer' button.	6MWTest icon no longer visible in device's toolbar.	6MWTest icon no longer visible in device's toolbar.	As expected.

	<p>Observe timer has started.</p> <p>Navigate away from application without shutting application down.</p> <p>Observe 6MWTest icon and notification in device's toolbar and notification area respectively.</p> <p>Navigate back to application.</p> <p>Wait for test to complete and results page to show.</p> <p>Navigate away from application without shutting application down.</p>	<p>Persistent notification no longer visible in notification area.</p> <p>GPS icon no longer visible in device's toolbar.</p>	<p>Persistent notification no longer visible in notification area.</p> <p>GPS icon no longer visible in device's toolbar.</p>	
<p>User returned to application on pressing background notification</p>	<p>Navigate to the home page and press the 'Start Timer' button.</p> <p>Observe timer has started.</p> <p>Navigate away from application without shutting application down.</p> <p>Observe notification in device's notification area.</p> <p>Press icon.</p>	<p>Application brought back to the foreground.</p> <p>Timer continuing as normal.</p>	<p>Application brought back to the foreground.</p> <p>Timer continuing as normal.</p>	<p>As expected</p>

6-minute walk test

Feature: 6-minute walk test				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
<p>User redirected to results page after completing test.</p>	<p>Navigate to home page.</p> <p>Press 'Start Timer' button.</p> <p>Walk until timer reaches zero.</p>	<p>Alert: "Hold on a sec while your results are finalised and saved..."</p> <p>Redirects to result page showing distance walked, date, time, location and comparisons.</p> <p>DB updated with result.</p>	<p>Alert: "Hold on a sec while your results are finalised and saved..."</p> <p>Redirects to result page showing distance walked, date, time, location and comparisons.</p> <p>DB updated with result.</p>	<p>As expected</p>

Notification shown after completion of test.	Navigate to home page. Press 'Start Timer' button. Walk until timer reaches zero.	Notification of completion shown with result of test in metres. Result of test matches result shown in redirected result page. DB updated with result.	Notification of completion shown with result of test in metres. Result of test matches result shown in redirected result page. DB updated with result.	As expected.
Notification showing best distance after first test completion.	Register account as new user. Login as new user. Navigate to home page. Press 'Start Timer' button. Confirm safety and instructions have been read. Press 'Start Timer' button again. Walk until timer reaches zero.	Notification of completion shown with result of test in metres. Notification of best distance achieved shown in metres. Result of test matches result shown in redirected result page. DB updated with result.	Notification of completion shown with result of test in metres. Notification of best distance achieved shown in metres. Result of test matches result shown in redirected result page. DB updated with result.	As expected
Warning popup when trying to stop test.	Navigate to home page. Press 'Start Timer' button. Observe timer counting down. Press 'Stop Timer'	Popup showing explanation that test won't be saved if stopped.	Popup showing explanation that test won't be saved if stopped.	As expected
Test stopped after confirming to stop.	Navigate to home page. Press 'Start Timer' button. Observe timer counting down. Press 'Stop Timer' Observe popup. Press 'I wish to stop' text in popup.	Test is stopped with timer reset to 06:00 and 'Start Timer' text shown on button. No test results saved in DB.	Test is stopped with timer reset to 06:00 and 'Start Timer' text shown on button. No test results saved in DB.	As expected
Test continued after confirming to continue.	Navigate to home page. Press 'Start Timer' button. Observe timer counting down.	Test continues with as normal.	Test continues with as normal.	As expected

	Press 'Stop Timer' Observe popup. Press 'I wish to continue' text in popup.			
Timer continues to countdown when popup shown.	Navigate to home page. Press 'Start Timer' button. Observe timer counting down and note time. Press 'Stop Timer' Observe popup. Wait 10 seconds. Press 'I wish to continue' text in popup.	Timer shows 10 seconds less than before popup shown. Test continues as normal.	Timer shows 10 seconds less than before popup shown. Test continues as normal.	As expected.
Ensure test starts only after enabling location services.	Disable device's location. Ensure application has Location under 'permissions' on device's settings. Navigate to home page of application. Press 'Start Timer' button. Observe system message asking to enable location services. Press 'Ok'	Timer only begins counting down after enabling location.	Timer only begins counting down after enabling location.	As expected
Ensure test doesn't start after rejecting location services.	Disable device's location. Ensure application has Location under 'permissions' on device's settings. Navigate to home page of application. Press 'Start Timer' button. Observe system message asking to enable location services. Press 'Cancel'	Timer doesn't start counting down.	Timer doesn't start counting down.	As expected.
Ensure test starts only after allowing location permissions.	Enable device's location. Ensure application doesn't have Location under 'permissions' on device's settings.	Timer only begins counting down after allowing permissions.	Timer only begins counting down after allowing permissions.	As expected

	Navigate to home page of application. Press 'Start Timer' button. Observe popup asking for location permissions. Select 'Yes'			
Ensure test doesn't start after rejecting location permissions.	Enable device's location. Ensure application doesn't have Location under 'permissions' on device's settings. Navigate to home page of application. Press 'Start Timer' button. Observe popup asking for location permissions. Select 'No'	Timer doesn't start counting down.	Timer doesn't start counting down.	As expected

Previous Results Tests

Feature: Previous Results				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
No results shown in log for new user.	Register an account as a new user. Login with new credentials. Navigate to log tab.	"No previous results found" for both last 10 results graph and all previous results.	"No previous results found" for both last 10 results graph and all previous results.	As expected
Results shown after carrying out test as new user.	Register an account as a new user. Login with new credentials. Navigate to log tab. Observe no previous results found. Navigate to home tab. Carry out steps to start test. Walk until timer reaches zero. Observe application redirecting to result page. Navigate to log tab.	Graph shown with one point. List of previous results shown one result. Best distance matches last result. Worst distance matches last result. DB updated with test result.	Graph shown with one result. List of previous results shown one result. Best distance matches last result. Worst distance matches last result. DB updated with test result.	As expected.
Previous results shown in descending order by date.	Login using new credentials for previous test. Navigate to home tab.	Graph shows line with two points.	Graph shows line with two points.	As expected

	Carry out steps to start test. Walk until timer reaches zero. Observe application redirecting to result page. Navigate to log tab.	New result shown at top of list above previous result. DB updated with test result.	New result shown at top of list above previous result. DB updated with test result.	
--	---	--	--	--

API Test Cases

Account

Endpoint: /api/login Request body: {email, password} Type: POST				
Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Login with valid user credentials	{ "email": "test@case.com", "password": "password" }	User's data returned as correct from DB along with token for persistent login.	User's data returned as correct from DB along with token for persistent login.	As expected
Login with invalid email	{ "email": "test@casenotvalid.com", "password": "password" }	Message returned: "Invalid email or password provided."	Message returned: "Invalid email or password provided."	As expected
Login with invalid password but valid email	{ "email": "test@case.com", "password": "notvalidpassword" }	Message returned: "Invalid email or password provided."	Message returned: "Invalid email or password provided."	As expected
Login with invalid email format	{ "email": "test@case.", "password": "notvalidpassword" }	Message returned: "Invalid email or password provided."	Message returned: "Invalid email or password provided."	As expected

Endpoint: /api/get-details
Request body: {email, token}
Type: POST

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Get details with valid email and token	{ "email": "test@case.com", "token": "eyJhbGciOiJIUzI1NiJ9.NDc.Z3Yaf3O97tQ6EtRkAgsN1H8g_pwi159PdonG1izgZoo" }	User's data returned as correct from DB.	User's data returned as correct from DB.	As expected
Get details with invalid email and valid token	{ "email": "test@casenotvalid.com", "token": "eyJhbGciOiJIUzI1NiJ9.NDc.Z3Yaf3O97tQ6EtRkAgsN1H8g_pwi159PdonG1izgZoo" }	Message returned: "Invalid email or password provided."	Message returned: "Invalid email or password provided."	As expected
Get details with valid email and invalid token	{ "email": "test@case.com", "token": "notvalidtoken" }	Message returned: "Token expired."	Message returned: "Token expired."	As expected
Get details with missing email and valid token	{ "email": "", "token": "eyJhbGciOiJIUzI1NiJ9.NDc.Z3Yaf3O97tQ6EtRkAgsN1H8g_pwi159PdonG1izgZoo" }	Message returned: "Invalid email or password provided."	Message returned: "Invalid email or password provided."	As expected

Endpoint: /api/register
Request body: {forename, surname, dob, email, password}
Type: POST

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Register with valid details	{ "forename": "Test", "surname": "Case", "dob": "1990-12-31", "email": "test@casevalid.com", "password": "password" }	Message: "Account created. Please now login" DB updated to reflect addition.	Message: "Account created. Please now login" DB updated to reflect addition.	As expected
Register with invalid email format.	{ "forename": "Test", "surname": "Case", "dob": "1990-12-31", "email": "test@case.", "password": "password" }	Message returned: "Invalid email address entered.Please enter a valid email." DB not updated.	Message returned: "Invalid email address entered.Please enter a valid email." DB not updated.	As expected
Register with missing email.	{ "forename": "Test", "surname": "Case", "dob": "1990-12-31", "email": "", "password": "password" }	Message returned: "Invalid email address entered.Please enter a valid email."	Message returned: "Invalid email address entered.Please enter a valid email."	As expected

	<pre> "password": "password" } </pre>	enter a valid email.	enter a valid email.	
		DB not updated.	DB not updated.	
Register with already registered email.	<pre> { "forename": "Test", "surname": "Case", "dob": "1990-12-31", "email": "test@case.com", "password": "password" } </pre>	<p>Message returned: "An account already exists with that email."</p> <p>DB not updated.</p>	<p>Message returned: "An account already exists with that email."</p> <p>DB not updated.</p>	As expected

Endpoint: /api/reset-password
Request body: {email, password, confirmPassword}
Type: POST

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Reset password with valid details	<pre> { "email": "test@casevalid.com", "password": "password1", "confirmPassword": "password1" } </pre>	<p>Message: "Password successfully reset. Please now login."</p> <p>DB updated to reflect change.</p>	<p>Message: "Password successfully reset. Please now login."</p> <p>DB updated to reflect change.</p>	As expected
Reset password with invalid email.	<pre> { "email": "test@casenotvalid.com", "password": "password1", "confirmPassword": "password1" } </pre>	<p>Message returned: "No account found with that email."</p> <p>DB not updated.</p>	<p>Message returned: "No account found with that email."</p> <p>DB not updated.</p>	As expected
Reset password with differing passwords and valid email	<pre> { "email": "test@casevalid.com", "password": "password1", "confirmPassword": "password" } </pre>	<p>Message returned: "Please ensure both passwords match."</p> <p>DB not updated.</p>	<p>Message returned: "Please ensure both passwords match."</p> <p>DB not updated.</p>	As expected
Reset password with invalid email format and	<pre> { "email": "test@case.", "password": "password1", "confirmPassword": "password1" } </pre>	<p>Message returned: "Invalid email address entered. Please enter a valid email."</p>	<p>Message returned: "Invalid email address entered. Please enter a valid email."</p>	As expected

matching passwords		DB not updated.	DB not updated.	
Reset password with missing email	{ "email": "", "password": "password1", "confirmPassword": "password1" }	Message returned: "Invalid email address entered. Please enter a valid email." DB not updated.	Message returned: "Invalid email address entered. Please enter a valid email." DB not updated.	As expected

Results

Endpoint: /api/previous-results

Expected Parameters: userId

Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/previous-results?userId=48	Response showing correct results from database	Response showing correct results from database	As expected
Invalid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/previous-results?userId=invalid	Message response: "No previous results found"	Message response: "No previous results found"	As expected
Missing userId parameter	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/previous-results?	Message response: "No previous results found"	Message response: "No previous results found"	As expected

Endpoint: /api/last-ten-results

Expected Parameters: userId

Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/last-ten-results?userId=48	Response showing correct results	Response showing correct results	As expected

		from database	from database	
Invalid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/last-ten-results?userId=invalid	Message response: "No previous results found"	Message response: "No previous results found"	As expected
Missing userId parameter	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/last-ten-results?	Message response: "No previous results found"	Message response: "No previous results found"	As expected

Endpoint: /api/max-distance
Expected Parameters: userId
Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/max-distance?userId=48	Response showing correct results from database	Response showing correct results from database	As expected
Invalid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/max-distance?userId=invalid	Null distance response	Null distance response	This could be handled better in future enhancements by providing a meaningful message
Missing userId parameter	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/max-distance?	Null distance response	Null distance response	This could be handled better in future enhancements by providing a meaningful message

Endpoint: /api/min-distance
Expected Parameters: userId
Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/min-distance?userId=48	Response showing correct	Response showing correct	As expected

		results from database	results from database	
Invalid userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/min-distance?userId=invalid	Null distance response	Null distance response	This could be handled better in future enhancements by providing a meaningful message
Missing userId parameter	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/min-distance?	Null distance response	Null distance response	This could be handled better in future enhancements by providing a meaningful message

Endpoint: /api/has-result
Expected Parameters: userId
Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Userid with previous results	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/has-result?userId=48	Message response: "204"	Message response: "204"	Message responses could be clearer. 200 signifies that it is indeed a new user. 204 signifies it isn't.
Userid without previous results	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/has-result?userId=49	Message response: "200"	Message response: "200"	Message responses could be clearer. 200 signifies that it is indeed a new user. 204 signifies it isn't.
Invalid user id	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/has-result?userId=invalid	Message response: "200"	Message response: "200"	This could be handled better in future enhancements by providing a meaningful message.
Missing user id	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/has-result?	Message response: "200"	Message response: "200"	This could be handled better in future enhancements by providing a meaningful message.

Endpoint: /api/user-result-mean
Expected Parameters: userDOB
Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid user DOB	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-result-mean?userDOB=1990-12-31	Response showing correct results from database	Response showing correct results from database	As expected
Invalid user DOB	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-result-mean?userDOB=invalid	Message response: "No previous results found"	Message response: "No previous results found"	As expected
Missing DOB	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-result-mean?	Message response: "No previous results found"	Message response: "No previous results found"	As expected

Endpoint: /api/user-all-result-mean
Expected Parameters: userId
Type: GET

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
User with previous results	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-all-result-mean?userId=48	Response showing correct results from database	Response showing correct results from database	As expected
Userid without previous results	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-all-result-mean?userId=49	Message response: "No previous results found"	Message response: "No previous results found"	As expected
Invalid userid	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-all-result-mean?userId=invalid	Message response: "No previous results found"	Message response: "No previous results found"	As expected
Missing userid	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/user-all-result-mean?	Message response: "No previous results found"	Message response: "No previous results found"	As expected

		results found"	results found"	
--	--	----------------	----------------	--

Endpoint: /api/save-result

Request body: {userId, city, street, distance, date}

Type: POST

Test Description	Input (if applicable)	Actual Result	Expected Result	Comments
Valid result saved	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/save-result body: { "userId": 48, "date": "2018-03-27 10:25:00", "distance": 500, "street": "Dumbarton Road", "city": "Glasgow" }	Message response: "Result saved." DB updated.	Message response: "Result saved." DB updated.	As expected
Result saved with invalid userid	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/save-result body: { "userId": 5, "date": "2018-03-27 10:25:00", "distance": 500, "street": "Dumbarton Road", "city": "Glasgow" }	Message response: "Problem saving result." DB not updated.	Message response: "Problem saving result." DB not updated.	As expected
Result with missing userId	https://devweb2017.cis.strath.ac.uk/pkb14171-nodejs/api/save-result body: { "userId": "", "date": "2018-03-27 10:25:00", "distance": 500, "street": "Dumbarton Road", "city": "Glasgow" }	Message response: "Problem saving result." DB not updated.	Message response: "Problem saving result." DB not updated.	As expected

Appendix H

User Manual

Managing Login/Registration/Password Reset

Logging in – by the user entering their details in the *email* and *password* input boxes and pressing the *LOGIN* button.

Once the *LOGIN* button is pressed the user will either be redirected to the ‘Home’ page or a message will show if their credentials were invalid. On a successful login, any subsequent logins will be handled automatically by the application until the user logs out.

Creating a new account – by pressing the *CREATE NEW ACCOUNT* text.

On a successful registration the user will be presented with a message which on closing will take them back to the login screen where they can then enter their new credentials and login.

On an unsuccessful registration the user will be presented with a message explaining why the registration was unsuccessful. This could be for two reasons: the user tried to register with an email that is already registered, or the user tried to register with an email which is of an invalid format.

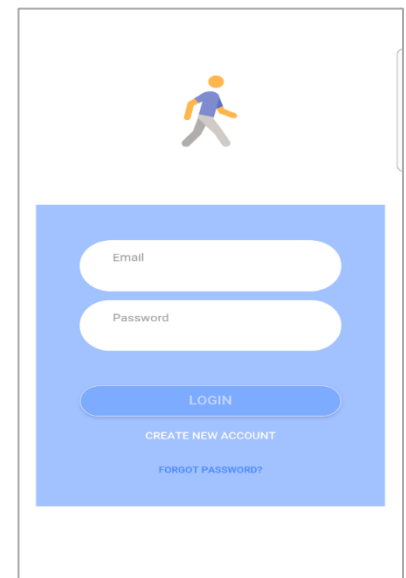
Reset password –by pressing the *FORGOT PASSWORD?* text.

On a successful password reset, the user will be presented with a message confirming so.

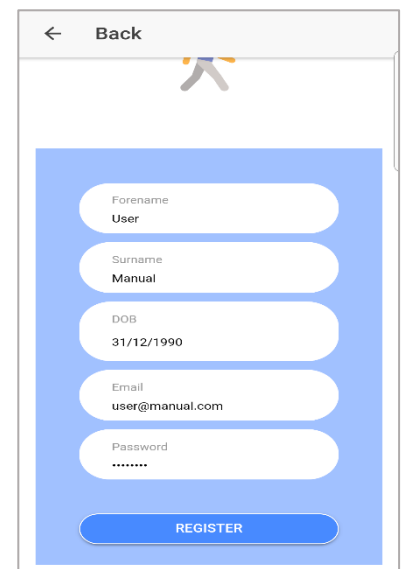
On an unsuccessful password reset the user will be presented with a message explaining why the reset was unsuccessful. This could either be because both passwords supplied don’t match, the email provided doesn’t belong to a valid account, or the email provided isn’t of a valid email format.

The user can go back to the login page at any point by pressing the *Back* text at the top left corner of the screen on both the registration screen and the reset password screen.

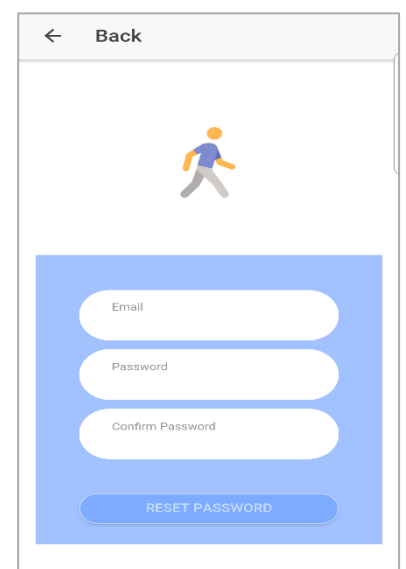
All fields in each of these screens are required therefore the button shown in each screen will remain disabled until the required fields have been filled in.



A mobile app login screen. At the top is a blue header with a white 'Back' button and a user icon. Below the header is a light blue card with rounded corners. Inside the card, there are two white input fields: 'Email' and 'Password'. Below these fields are three blue buttons: 'LOGIN', 'CREATE NEW ACCOUNT', and 'FORGOT PASSWORD?'. The 'LOGIN' button is the largest and is positioned in the middle.



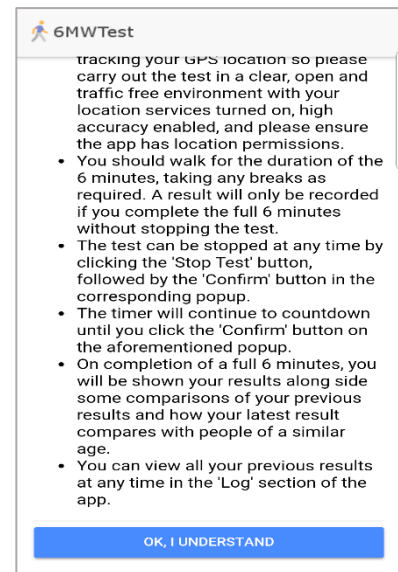
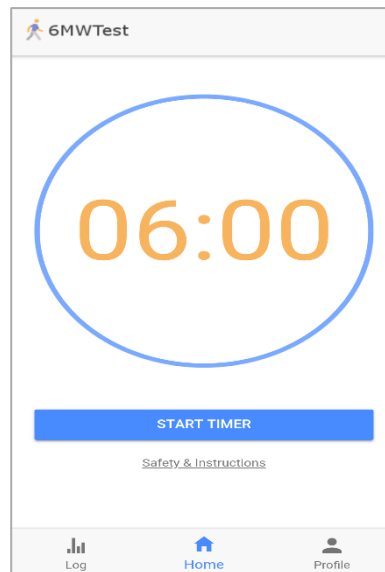
A mobile app registration screen. At the top is a blue header with a white 'Back' button and a user icon. Below the header is a light blue card with rounded corners. Inside the card, there are five white input fields: 'Forename User', 'Surname Manual', 'DOB 31/12/1990', 'Email user@manual.com', and 'Password'. Below these fields is a blue button labeled 'REGISTER'.



A mobile app reset password screen. At the top is a blue header with a white 'Back' button and a user icon. Below the header is a light blue card with rounded corners. Inside the card, there are three white input fields: 'Email', 'Password', and 'Confirm Password'. Below these fields is a blue button labeled 'RESET PASSWORD'.

Carrying out a 6-minute walk test

The 'Home' screen is where the user will carry out a 6-minute walk test. The user can start the test by pressing the *START TIMER* button. If the user has not carried out any previous tests before then prior to the test starting, the safety and instructions will be shown. The user should read these instructions and then press the *OK, I UNDERSTAND* button at the bottom. The user can then begin the test by pressing the *START TIMER* button again. The user can access the safety and instructions at any time by pressing the *Safety & Instructions* text.



Once the test begins the application may prompt the user to allow permission to access their location. On allowing the permission, the application will enable the required settings automatically and the test will then begin. The user does not need to leave the application at any point during this process.

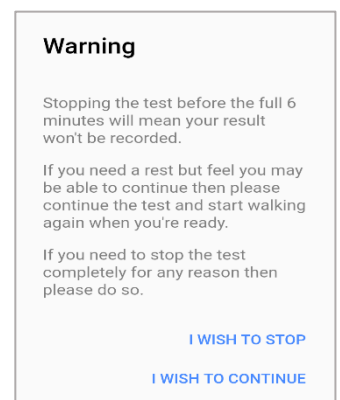
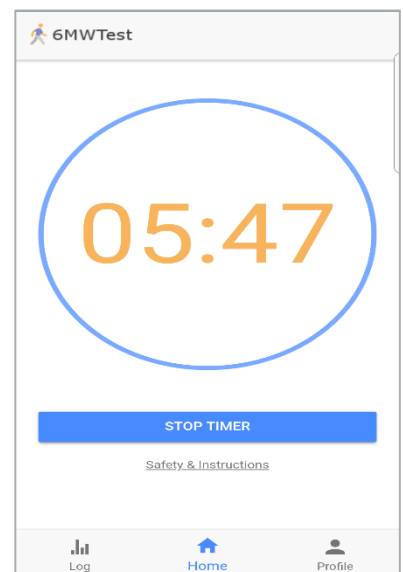
On the test beginning, the displayed timer will start counting down and the *START TIMER* button will change to a *STOP TIMER* button. If the user presses the *STOP TIMER* button at any point during the test they will be presented with a popup explaining that their results won't be saved if they stop the test.

The displayed popup shows two options *I WISH TO STOP* and *I WISH TO CONTINUE*.

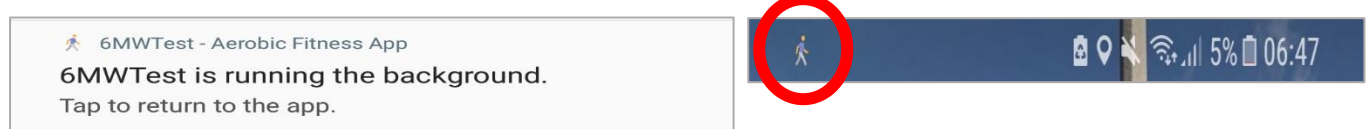
If the user presses the *I WISH TO STOP* option then the popup will close, the displayed countdown timer will return to displaying 06:00, the *STOP TIMER* button will change back to the *START TIMER* button, and the user's results won't be saved.

If the user presses the *I WISH TO CONTINUE* option then the popup will close and the test will continue as normal.

The displayed countdown timer will continue to run until it reaches 00:00 or the user selects the *I WISH TO STOP* option in the popup.

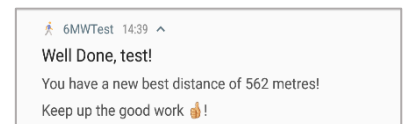


Whilst the test is being carried out, the user can navigate away from the application or walk with their phone in their pocket. The application will continue to run in the background until the test is complete. The application will notify the user that the application is running in the background via an icon in their device's toolbar and a persistent notification. The user can return to the application by pressing the persistent notification.



Test completion

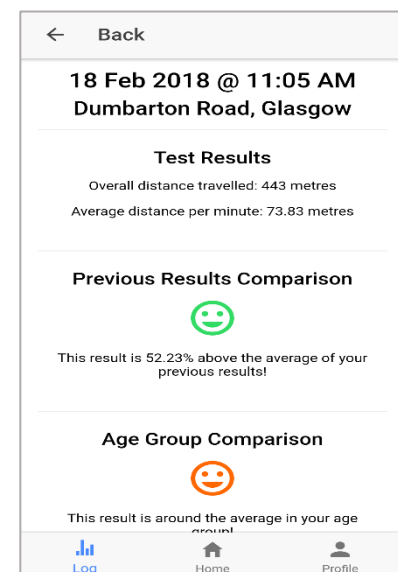
Once the countdown timer has reached zero, the test is complete and the user can stop walking. The application will save the results calculated during the test and redirect the user to their results page. The application will also notify the user that they have completed their test via notification and vibration and sound if their device isn't set to silent. A further notification will also be given to the user if they have achieved a new best distance.






Test result

The results screen shown to the user is the same screen the user can access by selecting a previous result in the 'Log' screen. The results show the date, time, and location of the test carried out; the distance travelled for that test along with the average distance per minute; comparison sections for the user to compare this result with their previous results and the results from users within a similar age group.

The comparison sections show a coloured icon which will change depending on the result in the comparison. If the result is greater than the average of the user's previous result/age group results then the icon will be a green happy face. If the result is less than the average of the user's previous result/age group results then the icon will be a red sad face. If the result is around the average of the user's previous result/age group results then the icon will be an orange happy face.



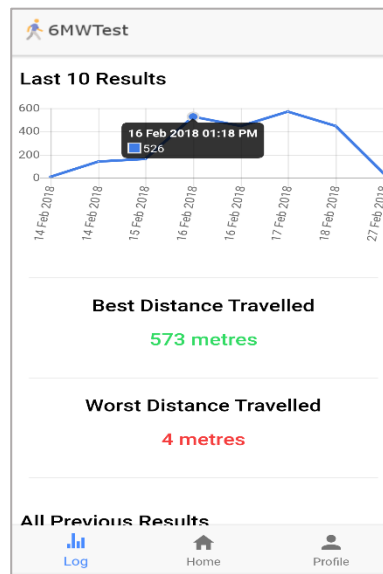
Age Group Comparison	Age Group Comparison	Age Group Comparison
		
This result is 22.7% above the average in your age group!	This result is around the average in your age group!	This result is 91.65% below the average in your age group!

The text will also update to reflect this result and show how much the result is above or below the average, depending on the result.

Previous results

At any given point, the user can view all their previous results by pressing the 'Log' tab.

The log screen shows the user their last 10 previous results, if they so have them, in a graph at the top to allow the user to clearly see their progression. On pressing a point in the graph, the user is presented with a tooltip providing them with more information. The log screen also shows the user's best distance and worst distance travelled which are clearly highlighted with headings and different text colours. Below this is a list of all the user's previous results.



Date	Time	Distance (metres)
14 Feb 2018		
14 Feb 2018		
15 Feb 2018		
16 Feb 2018	01:18 PM	526
16 Feb 2018		
17 Feb 2018		
18 Feb 2018		
27 Feb 2018		

Date	Time	Distance (metres)
14 Feb 2018	10:52 AM	Gray Street, Glasgow
15 Feb 2018	03:20 PM	Unknown, Glasgow
16 Feb 2018	01:18 PM	Dumbarton Road, Glasgow
16 Feb 2018	09:22 PM	Norval Street, Glasgow
17 Feb 2018	12:06 PM	Dumbarton Road, Glasgow
18 Feb 2018	11:05 AM	Dumbarton Road, Glasgow
27 Feb 2018	02:29 PM	Dumbarton Road, Glasgow

On selecting a specific previous result, the user is taken the specific result page as discussed above which shows their result, and some comparison statistics.

If a user has no previous results then text will be shown in place of the graph/list of previous results to make it clear to the user that they have no results to display.

The log screen will refresh every time the user views the screen by either pressing the 'Log' tab or by pressing the 'Back' button in the specific results page. This is to ensure that the user is always viewing their latest results.

Profile

The user can view their profile by pressing the 'Profile' tab at the bottom of the application. In their profile the user can change their forename, surname, dob, height, weight, and gender by pressing on the desired field and editing accordingly. Each change in the profile will be automatically saved. During the save process, a small spinning icon will appear on the top right-hand side of the page to indicate that the changes are being saved.

Both the height and weight fields can be altered to show different units:

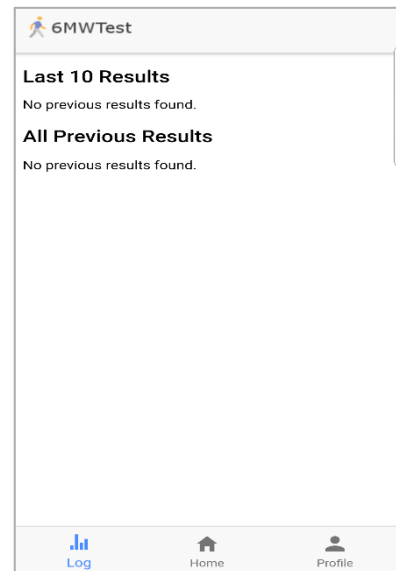
Height – Feet (feet and inches), Cms (centimetres).

Weight – Stone (stone and lbs), Kgs (kilograms).

On switching between these units, the application will automatically convert the height/weight according to the selected unit.

The user can logout of the application at any time by pressing the logout icon found in the top-right corner of the profile screen.

Scott Morrison



Forename:	User
Surname:	Manual
DOB:	31/12/1990
Height:	0 ft, 0 in, Units: Feet
Weight:	0 st, 0 lbs, Units: Stone
Gender:	

Appendix I

Build and Run Instructions

Prerequisites

Install NodeJS and npm (npm comes with NodeJS install):

<https://nodejs.org/en/>

Verify install by running:

```
npm -version  
node --version
```

To ensure you have the latest version of npm run:

```
npm install npm@latest -g
```

Install Ionic and Cordova:

```
npm install -g ionic cordova
```

note: -g signifies a global install therefore administrative rights are required

Building node_modules folder & Ionic plugins

All required node_modules are listed in the project's package.json file for both the Ionic application and the NodeJS application. These need to be downloaded and generated in a node_modules folder within the project before continuing.

To do this run the following command from the project directory (where package.json resides):

```
npm install
```

This may take some time depending on your connection speed.

After creating the new node_modules folder, the Ionic platform and plugins need to be restored from package.json. From the project directory run:

```
ionic cordova prepare
```

Building and Running the Ionic Application

Running Ionic application

On success of all the above, you can now run the Ionic application. To run the application such that it appears in the browser run (from the project directory):

```
ionic serve
```

This will boot a development server and run the application in your browser. However, due to the application using Cordova, some plugins won't function as expected and a 'Cordova not available error' will be thrown, namely when clicking the 'Start Timer' button. Therefore, to get around this it is recommended to run the application using an emulator by running the following command:

```
Ionic cordova run [<platform>]
```

Where [<platform>] would be android/ios.

Stopping Ionic application

Stopping the Ionic serve command can be done by going to the terminal and entering:

```
Ctrl+c
```

Testing on a native device

There are a number of ways to test the Ionic application in a native environment. For instructions please follow the Ionic documentation:

To create either a debug or production apk or ipa file for Android or iOS follow:

<https://ionicframework.com/docs/cli/cordova/build/>

To run the application on a connected native device, follow:

<https://ionicframework.com/docs/cli/cordova/run/>

To run the application on an emulator, follow:

<https://ionicframework.com/docs/cli/cordova/emulate/>

Note: for emulating the application, your environment must be setup correctly. To ensure this is the case please follow:

Android: <https://cordova.apache.org/docs/en/latest/guide/platforms/android/#installing-the-requirements>

iOS: <https://cordova.apache.org/docs/en/latest/guide/platforms/ios/index.html>

Building and Running the NodeJS Application

Due to how the CIS systems support team has setup their Node environment, the NodeJS application must reside in /home/username/DEVWEB/2017/nodejs. For setting this up after the end of this project, it would be suggested to contact the CIS systems support team for help with doing this. Some documentation can be found here:

<https://devweb2017.cis.strath.ac.uk/nodejs.html>

The current NodeJS backend for this application resides in the developer's own *nodejs* directory under the following directory '*/home/pkb14171/DEVWEB/2017/nodejs/6mwtest-backend*'