|  |
| --- |
| COM667 Computing Systems Project |
| Final Report |
| Food and Fitness Tracker - FFT |

|  |
| --- |
| Christopher Jordan – B00554456 Project Supervisor: Leo Galway |

Abstract

The initial motivation behind the ‘Food & Fitness Tracker – FFT’ project was to create a nutritional and exercise tracking application which had all the advantages of existing solutions but none of the drawbacks. To enable this, the role of mobile health, nutrition and exercise tracking applications in helping to tackle obesity was investigated with the aim being to discover and implement an innovative solution in which a user is encouraged to track their nutrition and exercise activities, in the hope that the end user would be encouraged into adopting a healthier lifestyle.

The project followed the Modified Waterfall software development lifecycle, with functionality testing occurring alongside the development process.

Testing of the entire application was completed after to the development process had been completed. The main technique employed was Black Box testing (i.e. manual testing).

The design phase of this project explains the data structure used to store and retrieve data. A flow map has also been included to show the thinking behind how a user may use the application. UI design have also been included to outline what the completed application should look like alongside a colour palette and app badge.

The project Implementation section of this project contains code sections that were explained in greater detail so that their purpose could be understood.

Whilst the completed application did not meet all the requirements originally stated, however, due to the ‘MOSCOW’ approach to requirement prioritisation being employed for this project and the requirements that were not met having been listed as a ‘Could Have’ it has not adversely affected outcome of this project.

Acknowledgments

I would like to take this opportunity to offer my thanks to everyone who was involved in making this project a success.

I would like to firstly offer my thanks to all the lecturers and staff at the University of Ulster for the four years of excellent teaching that has been provided to me. Thank you for the consistent support and learning provided along my academic journey here at Ulster University.

I would like to express my thanks and appreciation to my project supervisor, Leo Galway for time and guidance during this final year of study. Leo thank you for constantly challenging me so that I complete my work to the highest standards and for taking the time lout of your busy schedule to meet with me.

Special thanks must also go to Matthew Hobson and Kourosh Pourgholi who helped with the Requirements gathering process and the user acceptance testing aspects of this project.

I would also like to thank ‘Allsop – Intelligent Solutions’ for their support over the last year as member of their staff, with special thanks going to Richard Campbell (CEO) and Matt McNabb (Senior Front-End Developer) who provided me with guidance on how to best manage this project as well as mentoring me in the Iconic Framework at the start of my project.

Finally, I would like to thank my friends and family, who have been patient and supportive throughout the year whilst I was working on this project. Thank you for your unwavering belief in everything I set out to do in my career, both academic and professional.

Table of Contents

[1 Chapter 1: Introduction 1](#_Toc5558491)

[1.1 Project Background 1](#_Toc5558492)

[1.2 Project Aim 1](#_Toc5558493)

[1.3 Copyright, Intellectual Property Rights or Commercial Sensitivity 1](#_Toc5558494)

[1.4 Hardware and Software 1](#_Toc5558495)

[1.5 Dissertation Overview 2](#_Toc5558496)

[2 Chapter 2: Literature Review 3](#_Toc5558497)

[2.1 Background Research 3](#_Toc5558498)

[2.1.1 Obesity 3](#_Toc5558499)

[2.1.1.1 Risks of Obesity 4](#_Toc5558500)

[2.1.1.2 Causes of Obesity 5](#_Toc5558501)

[2.1.1.3 Treating Obesity 5](#_Toc5558502)

[2.1.2 Mobile Health 5](#_Toc5558503)

[2.1.2.1 Benefits of Mobile Health 5](#_Toc5558504)

[2.1.2.2 Mobile Health Trends 6](#_Toc5558505)

[2.1.2.3 Activity Tracking 6](#_Toc5558506)

[2.1.2.4 Diet and Nutritional Tracking 6](#_Toc5558507)

[2.2 Software Development Life Cycles 6](#_Toc5558508)

[2.2.1 What is a SDLC? 7](#_Toc5558509)

[2.2.2 Stages of a SDLC 7](#_Toc5558510)

[2.2.2.1 Planning and Requirement Analysis 7](#_Toc5558511)

[2.2.2.2 Defining Requirements 8](#_Toc5558512)

[2.2.2.3 Design 8](#_Toc5558513)

[2.2.2.4 Building the Project 8](#_Toc5558514)

[2.2.2.5 Testing 8](#_Toc5558515)

[2.2.2.6 Deployment to the Market and Maintenance 9](#_Toc5558516)

[2.2.3 Software Development Models 9](#_Toc5558517)

[2.2.3.1 Waterfall Model 9](#_Toc5558518)

[2.2.3.1.1 Advantages 10](#_Toc5558519)

[2.2.3.1.2 Disadvantages 10](#_Toc5558520)

[2.2.3.2 Modified Waterfall. 10](#_Toc5558521)

[2.2.3.2.1 Advantages 10](#_Toc5558522)

[2.2.3.2.2 Disadvantages 10](#_Toc5558523)

[2.2.3.3 Scrum Model 10](#_Toc5558524)

[2.2.3.3.1 Advantages 11](#_Toc5558525)

[2.2.3.3.2 Disadvantages 11](#_Toc5558526)

[2.2.3.4 Spiral Model 11](#_Toc5558527)

[2.2.3.4.1 Advantages 12](#_Toc5558528)

[2.2.3.4.2 Disadvantages 12](#_Toc5558529)

[2.2.3.5 Conclusion 12](#_Toc5558530)

[2.3 Technical Background 12](#_Toc5558531)

[2.3.1 iPhone Operating System 12](#_Toc5558532)

[2.3.2 Android 13](#_Toc5558533)

[2.3.3 Chosen Platform 13](#_Toc5558534)

[2.3.4 Development Tools and Languages 13](#_Toc5558535)

[2.3.4.1 Xamarin 13](#_Toc5558536)

[2.3.4.1.1 Requirements 14](#_Toc5558537)

[2.3.4.2 Ionic 14](#_Toc5558538)

[2.3.4.2.1 Requirements 14](#_Toc5558539)

[2.3.4.3 Chosen Development Tool and Language 14](#_Toc5558540)

[2.3.4.4 Data Storage Options 15](#_Toc5558541)

[2.3.4.4.1 SQLite 15](#_Toc5558542)

[2.3.4.4.2 Firebase 15](#_Toc5558543)

[2.3.4.4.3 Local Storage 15](#_Toc5558544)

[2.3.4.4.4 Chosen Data Storage Option 15](#_Toc5558545)

[2.3.5 Resources Required 15](#_Toc5558546)

[3 Chapter 3: Requirement Analysis 17](#_Toc5558547)

[3.1 Requirement Gathering 17](#_Toc5558548)

[3.1.1 Focus Group 18](#_Toc5558549)

[3.1.2 Observations 18](#_Toc5558550)

[3.1.2.1 Apple Health 18](#_Toc5558551)

[3.1.2.2 Nike Run Club 19](#_Toc5558552)

[3.1.2.3 MyFitnessPal & MapMyRun 19](#_Toc5558553)

[3.2 Requirements 19](#_Toc5558554)

[3.2.1 Functional Requirements 19](#_Toc5558555)

[3.2.2 Non-Functional Requirements 20](#_Toc5558556)

[3.2.3 Requirement Prioritisation Strategy 21](#_Toc5558557)

[3.3 Risk Analysis 22](#_Toc5558558)

[3.3.1 Risk Mitigation Strategy 23](#_Toc5558559)

[3.3.1.1 Pre-Controls 24](#_Toc5558560)

[3.3.1.2 Post Controls 25](#_Toc5558561)

[3.4 Requirements Gathered During Development 25](#_Toc5558562)

[4 Chapter 4: Design 26](#_Toc5558563)

[4.1 Design Principles 26](#_Toc5558564)

[4.2 Flow Map 26](#_Toc5558565)

[4.3 Prototyping 27](#_Toc5558566)

[4.3.1 Colour Palette 28](#_Toc5558567)

[4.3.2 App Badge 29](#_Toc5558568)

[4.3.3 Details Page 29](#_Toc5558569)

[4.3.4 Diet Home Page 30](#_Toc5558570)

[4.3.5 Main Menu Pane 30](#_Toc5558571)

[4.3.6 Select Date Page 31](#_Toc5558572)

[4.3.7 Food Diary Page 32](#_Toc5558573)

[4.3.8 Add Meal Page 32](#_Toc5558574)

[4.3.9 Exercise Home Page 33](#_Toc5558575)

[4.3.10 New Run Page 34](#_Toc5558576)

[4.3.11 Previous Runs Page 35](#_Toc5558577)

[4.3.12 Run Details Page 36](#_Toc5558578)

[4.4 Data Storage 36](#_Toc5558579)

[5 Chapter 5: Implementation 38](#_Toc5558580)

[5.1 System Architecture 38](#_Toc5558581)

[5.1.1 Views 38](#_Toc5558582)

[5.1.2 Controllers 39](#_Toc5558583)

[5.1.3 Model 39](#_Toc5558584)

[5.2 Code Overview 39](#_Toc5558585)

[6 Chapter 6: Testing and Evaluation 40](#_Toc5558586)

[6.1 Quality Assurance/ Testing 40](#_Toc5558587)

[6.1.1 What is Testing? 40](#_Toc5558588)

[6.1.1.1 Verification 41](#_Toc5558589)

[6.1.1.2 Validation 41](#_Toc5558590)

[6.1.1.3 White Box Testing (Development) 42](#_Toc5558591)

[6.1.1.4 Black Box Testing (Manual) 42](#_Toc5558592)

[6.1.1.5 Software Defects 43](#_Toc5558593)

[6.1.1.6 User Acceptance Testing 44](#_Toc5558594)

[6.1.1.6.1 Issues raised by UAT 45](#_Toc5558595)

[6.2 Evaluation 46](#_Toc5558596)

[6.2.1 Technologies Used 46](#_Toc5558597)

[6.2.1.1 Visual Studio Code 46](#_Toc5558598)

[6.2.1.2 The Ionic Framework 47](#_Toc5558599)

[6.2.1.3 Local Storage 47](#_Toc5558600)

[6.2.1.4 Nutritionix API 47](#_Toc5558601)

[6.2.2 Modified Waterfall Life Cycle 47](#_Toc5558602)

[7 Results 48](#_Toc5558603)

[7.1 Requirements Not Met 49](#_Toc5558604)

[7.2 Additional Work Completed 50](#_Toc5558605)

[8 Conclusion 51](#_Toc5558606)

[8.1 Project Reflection 51](#_Toc5558607)

[8.1.1 Project Scope 51](#_Toc5558608)

[8.1.2 Application Produced 52](#_Toc5558609)

[8.1.3 Source Control 52](#_Toc5558610)

[8.1.4 Project Maintenance 52](#_Toc5558611)

[9 References 54](#_Toc5558612)

[10 Appendix A 58](#_Toc5558613)

[10.1 Supporting Documentation 58](#_Toc5558614)

[11 Appendix B 58](#_Toc5558615)

[12 Appendix C 62](#_Toc5558616)

Table of Figures

[Figure 2.1 Adult Obesity levels from 1993 to 2016 in England (Baker, 2019) 4](#_Toc5737315)

[Figure 2.2 Obesity levels by age group(Baker, 2019) 4](#_Toc5737316)

[Figure 2.3 *Stages of a Software Development Life Cycle* (Wisdomjobs.com, 2018) 7](#_Toc5737317)

[Figure 2.4 Waterfall Model(Citta, 2017) 9](#_Toc5737318)

[Figure 2.5 Modified Waterfall Model (Powell-Morse, 2016) 10](#_Toc5737319)

[Figure 2.6 Scrum Model (Intelegain Technologies, 2017) 11](#_Toc5737320)

[Figure 2.7 Mobile Operating Market Share (StatCounter, 2018) 13](#_Toc5737321)

[Figure 3.1 Probability-Impact Matrix before controls were implemented 24](#_Toc5737322)

[Figure 3.2 Probability-Impact Matrix after controls were implemented 25](#_Toc5737323)

[Figure 4.1 FFT User Flow 27](#_Toc5737324)

[Figure 4.2 App Badge 29](#_Toc5737325)

[Figure 4.3 Details Page 29](#_Toc5737326)

[Figure 4.4 Diet Home Page 30](#_Toc5737327)

[Figure 4.5 Main Menu Pane 30](#_Toc5737328)

[Figure 4.6 Select Date Page 31](#_Toc5737329)

[Figure 4.7 Food Diary Page 32](#_Toc5737330)

[Figure 4.8 Add Meal Page 32](#_Toc5737331)

[Figure 4.9 Exercise Home Page 33](#_Toc5737332)

[Figure 4.10 New Run Page 34](#_Toc5737333)

[Figure 4.11 Previous Runs Page 35](#_Toc5737334)

[Figure 4.12 Run Details page 36](#_Toc5737335)

[Figure 4.13 Database Models 37](#_Toc5737336)

[Figure 5.1 Model-View-Controller (Margam, 2018) 38](#_Toc5737337)

[Figure 5.2 Ionic ‘View’ example (McGivery, 2014) 38](#_Toc5737338)

[Figure 5.3 service/database.ts code part 1 40](#_Toc5737339)

[Figure 5.4 service/database.ts code part 2 42](#_Toc5737340)

[Figure 5.5 Details.html code 44](#_Toc5737341)

[Figure 5.6 Details.ts code 45](#_Toc5737342)

[Figure 5.7 CheckFordetails() Method 46](#_Toc5737343)

[Figure 5.8 Date.html Code 47](#_Toc5737344)

[Figure 5.9 Date.ts Code 47](#_Toc5737345)

[Figure 5.10 Food-Log.html code (selectedDate variable) 48](#_Toc5737346)

[Figure 5.11 food-Log.ts code (selectedDate variable) 48](#_Toc5737347)

[Figure 5.12 Meal type html code 48](#_Toc5737348)

[Figure 5.13 Meal Type Logic 49](#_Toc5737349)

[Figure 5.14 Meals table call 49](#_Toc5737350)

[Figure 5.15 FormatRows() Method 50](#_Toc5737351)

[Figure 5.16 Code for HealthKit availability 51](#_Toc5737352)

[Figure 5.17 loadstepData() Method 51](#_Toc5737353)

[Figure 5.18 Step Count html code 52](#_Toc5737354)

[Figure 5.19 Search Food html code snippet 52](#_Toc5737355)

[Figure 6.1 Causes of Software Defects 56](#_Toc5737356)

[Figure 12.1 Original method for recording Meals 75](#_Toc5737357)

List of Tables

[Table 1 Functional Requirements 20](#_Toc5737358)

[Table 2 Non-Functional Requirements 20](#_Toc5737359)

[Table 3 MOSCOW Rating for Functional and Non-Functional Requirements 22](#_Toc5737360)

[Table 4 Project Risk Analysis 23](#_Toc5737361)

[Table 5 Scope Creep 25](#_Toc5737362)

[Table 6 Colour scheme of project application 28](#_Toc5737363)

[Table 7 Manual Test Cases Executed 56](#_Toc5737364)

[Table 8 Software Defects found during Testing 57](#_Toc5737365)

[Table 9 UAT Issue Resolutions 59](#_Toc5737366)

[Table 10 Requirements Fulfilment Status 62](#_Toc5737367)

[Table 11 Additional Work Completed 63](#_Toc5737368)

[Table 12 ST-001 – Android App Installation & Uninstall 71](#_Toc5737369)

[Table 13 ST-002 – iOS App Installation & Uninstall 71](#_Toc5737370)

[Table 14 ST-003 – Add Personal Details 72](#_Toc5737371)

[Table 15 ST-004 – Add Meal 73](#_Toc5737372)

[Table 16 ST-005 – View Current Day (Food Diary) 73](#_Toc5737373)

[Table 17 ST-006 – View Previous Day (Food Diary) 73](#_Toc5737374)

[Table 18 ST-007 – Add Run 74](#_Toc5737375)

[Table 19 ST-008 – View Previous Run 74](#_Toc5737376)

# Chapter 1: Introduction

## Project Background

Over the last 6 months I have gained an increased interest in my personal health and wellbeing. To help me keep a record of what I am eating and what exercises I am doing on daily basis, I employ the use of several mobile applications, as I have not been able to find an app which will do the same job as all three apps. These applications are “Apple Health”, “MyFitnessPal” and “Nike Run Club”. The main issue that I have with using these three applications is that whilst each app does record different information, and each have their own advantages such exercise route tracking, calories eaten in day and projected weight loss. They are also inefficient due the amount of crossover information that is stored on each of the application such as the total number of steps taken in a day and calories burned. The second issue I have with these apps is the number of adverts that appear (i.e. MyFitnessPal and Nike Run Club) when they are used. The third and final problem with them is that the information breakdowns tend to be quite complex and therefore not very helpful to a normal user. This app would initially be used by myself and a few of my friends who have encountered similar issues.

Due to the above reasons, I desire to create an application that will keep track of a user’s daily eating and fitness habits. The initial features for the proposed solution will allow the user to enter their daily diet and any exercises they have completed. As well as being able to track the current route a user is taking for an exercise that they are currently completing. A breakdown of the users’ daily diet and fitness habits will also be provided to the user.

## Project Aim

My project aim is to create a cross-platform application that will allow the user to track their daily diet and exercise activities and provide a breakdown of the information in an easily interpreted graphical representation.

## Copyright, Intellectual Property Rights or Commercial Sensitivity

The only area of concern that I will need to ensure that I comply with, when creating the application would be data protection as set by the General Data Protection Regulation (GDPR) by the EU. This law requires me to ensure that all personal data and the privacy of EU citizens is protected.

## Hardware and Software

The only hardware necessary for me to create the app would be my own personal pc and access to mac which contains the XCode. As well as this at this stage of my initial project planning, I am intending to create the application using the Ionic Framework.

## Dissertation Overview

Below is an overview of what is contained in the remaining chapters of the report:

**Chapter 2**

This chapter contains all research surrounding the project. This includes reviews on software development lifecycles, evaluations of possible target platforms, development environments/technologies, development methods. This section ends in a conclusion with the most appropriate approach being selected.

**Chapter 3**

Within this section, the application requirements are outlined with information on techniques used to create functional and non-functional requirements. Also included in this section is the risk analysis of the overall project.

**Chapter 4**

Presented in this chapter are the designs of the system, this includes high-level UI designs as well as database designs.

**Chapter 5**

Implementation is documented in this chapter with an in-depth insight into how all elements of the system were developed. Requirements outlined in Chapter 3 and designs included in chapter 4 are used as references throughout this section.

**Chapter 6**

Testing is contained within this chapter which details the strategies used during and after implementation was complete. All testing methods are defined and explained with reference to how they were used.

**Chapter 7**

This chapter evaluates the completed mobile application. Project planning and maintenance is reviewed alongside the feedback from end users. Requirements outlined in Chapter 3 are reviewed against the end-product.

**Chapter 8**

The conclusion of the report evaluates the success of the project and includes the proposal of future enhancements.

# Chapter 2: Literature Review

## Background Research

### Obesity

The term obese can be used to describe any person who is very overweight (i.e. with a lot of body fat). This is becoming a common problem in the UK with it currently being estimated that obesity affects around 1 in every 4 adults and around 1 in every 5 children aged 10 to 11 in the UK.

Whilst there is no definitive way to diagnose someone as obese, the most commonly used method to indicate if someone can be classified as healthy weight, overweight or obese is by measuring a person’s Body Mass Index (BMI).

BMI is used to determine if a person is of a healthy weight for their height. The NHS has created a simple tool that can be used to determine you BMI and if you are of a Healthy weight. It can be accessed via <https://www.nhs.uk/live-well/healthy-weight/bmi-calculator/>.

“For most adults, a BMI of:

* 18.5 to 24.9 means you're a healthy weight
* 25 to 29.9 means you're overweight
* 30 to 39.9 means you're obese
* 40 or above means you're severely obese”

(NHS, 2016)

In more general terms men with a waist circumference of 94cm (37in) or more and women with a waist circumference of 80cm (about 31.5in) or larger are more at risk to develop obesity-related health problems in the future than someone who either of a healthy weight or just overweight.

The figure below shows the rise obesity levels in men and women since 1993 in England.

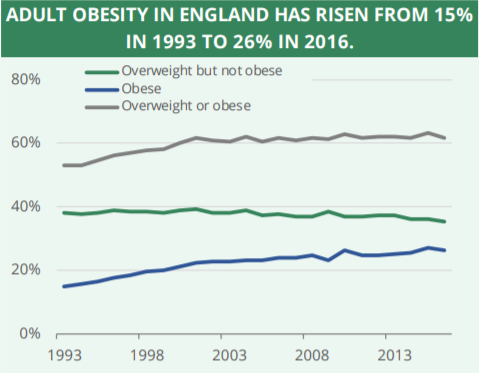


Figure 2.. Adult Obesity levels from 1993 to 2016 in England (Baker, 2019)

The figure below shows the obesity levels for both men and women in the UK.

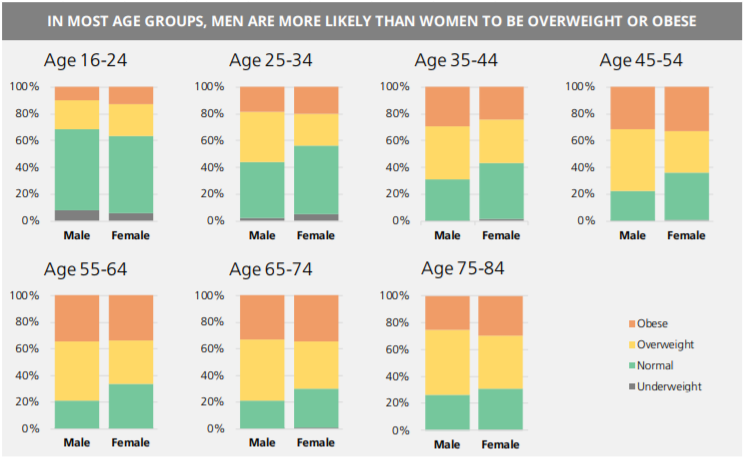


Figure 2.. Obesity levels by age group(Baker, 2019)

#### Risks of Obesity

Whilst being obese will lead to the obvious physical changes, it can also lead to several serious and potentially life-threatening conditions if it is not tackled. These conditions include Type 2 diabetes, Coronary Heart Disease, some types of cancer (i.e. breast and bowel cancer) and stroke. As well as these conditions obesity can also affect a person quality of life and could also result in a person developing depression and low self-esteem.

#### Causes of Obesity

Whilst there is sometimes an underlying health condition that can attribute to weight gain such as underactive thyroid gland (hypothyroidism), the most common cause of obesity is generally caused by consuming more calories, particularly those in fatty and sugary foods, than you burn off through physical activity. This excess energy which is not burned off is then stored by the body as fat, resulting in weight gain.

Our modern lifestyles are also an attributing factor to the increased number of people being diagnosed as obese. This is because for many people, modern living involves eating excessive amounts of cheap, high-calorie food and spending a lot of time sitting down, either at desks in work, on sofas or in cars.

#### Treating Obesity

The NHS recommends that the best way for a person to treat obesity is by eating a healthy, reduced calorie diet and to exercise regularly. Is suggested by the NHS that in order to do this that they should:

1. “eat a balanced, calorie-controlled diet as recommended by your GP or weight loss management health professional (such as a dietitian)
2. join a local weight loss group
3. take up activities such as fast walking, jogging, swimming or tennis for 150 to 300 minutes (2.5 to 5 hours) a week
4. eat slowly and avoid situations where you know you could be tempted to overeat”

(NHS, 2016)

The app that will be created as part of this project, will help users tackle their obesity problem allow them to track the foods that they are eating and their daily calorie intake. The app will also allow them to track the amount of exercise that they do.

### Mobile Health

Mobile Health (mHealth) is the general term which is used to describe the use of mobile phones and other wireless technology in medical care. Mobile Health is most commonly applied to mobile devices so that the user can be educated about preventative healthcare services. The application that will be created as a result of this project can be described as a Mobile Health application due to the aim of the application to educate the user about their daily diet and exercise routine which in turn should help them to try tackle their Obesity/weight issues.

#### Benefits of Mobile Health

The main benefit behind Mobile Health for consumers is the matter of convenience. As mobile applications and wearable devices allow users to constantly track and manage data without having to manually collect data which can take a long time. Another benefit is the sheer number of Mobile Health applications that available to consumers with there being 325,000 mHealth apps available for download from app stores as of 2017, according to digital health consulting firm research2guidance.

#### Mobile Health Trends

A consumer survey carried out by Accenture in 2018 found that 46% of respondents used their mobile phone or tablet to manage their health, compared to 36% in 2016. Further, 36% said technology was very important for managing health, up from 30% in 2016. 90% of respondents from the same survey were also willing to share information from their wearable devices with their physician.

#### Activity Tracking

Mobile Health applications have a proven record of increasing a user’s activity levels, with a study conducted by Sherry Boschert using control group, finding that those who used Mobile Health applications to track their activity levels increased their step count by on average 22 (half a mile) after 8 weeks compared to those who did not use a Mobile Health application to track their activity levels.

#### Diet and Nutritional Tracking

In a study conducted by West, et al. examining the potential of behavioural change theory integrated into mobile nutrition apps. The results of this study discovered the following results:

“The majority of study participants agreed or strongly agreed with statements regarding app use increasing their motivation to eat a healthy diet, improving their self-efficacy, and increasing their desire to set and achieve health diet goals. Additionally, majority of participants strongly agreed that using diet/nutrition apps led to changes in their behaviour, namely increases in actual goal setting to eat a healthy diet (58.5%, 127/217), increases in their frequency of eating healthy foods (57.6%, 125/217), and increases in their consistency of eating healthy foods (54.4%, 118/217).”

(Joshua H West, 2017)

From the results found by this study it can be clearly seen that a Mobile Health application that is used to track a user diet also ends up motivating the user to eat healthier due to them not releasing just how bad some of the foods that they originally ate our for the body. By increasing a users motivation to eat healthier means that it will also likely help people to lose weight which is aim of my app in helping to reduce worldwide obesity levels.

## Software Development Life Cycles

A Software Development Life Cycle (SDLC) also known as a Software Development Process is a process used by people who work in the software industry to help them design, develop and test software to ensure that the finished software is of a high quality. A SDLC aims to create high-quality software that either meets or exceeds the customers’ expectations whilst ensuring that the end software is completed on-time and within cost estimates.

To ensure that software is completed to the same standard when utilising a SDLC the international standard ‘ISO/IEC 12207’ was developed. This standard for software development processes aims to be the standard that defines all the tasks that are required for developing and maintaining software. (Tutorialspoint, 2018)

### What is a SDLC?

A SDLC is a framework which defines the tasks to be performed at each stage of the software development process. It consists of a detail plan describing how to develop, maintain, replace and alter or enhance specific software. The SDLC defines a methodology that can be used to improve the quality of the software and the overall development process. (Tutorialspoint, 2018)

A close up of text on a white background

Description automatically generatedThe following figure is graphical representation of several of the stages in a typical SDLC:

Figure 2.. *Stages of a Software Development Life Cycle* (Wisdomjobs.com, 2018)

### Stages of a SDLC

The typical Software Development Life Cycle will be comprised of the following stages:

#### Planning and Requirement Analysis

Requirement Analysis is the most important stage of the SDLC, as it is this stage that determines if a project should proceed or if it should be cancelled. The Requirement Analysis stage should be carried by senior members of the development team and any key stakeholders such as the customer, sales team and any experts on the project area. The information gathered is the used to help plan how the project will be approached as well as to conduct a feasibility study into the operational, economic and technical areas of the project and if it should go ahead. (Wisdomjobs.com, 2018)

Planning for the quality assurance will also usually be carried out at this stage. This includes quality assurance requirements and the identification risks. This allows for a software project to be implemented with minimum risks. (Wisdomjobs.com, 2018)

#### Defining Requirements

Upon the Requirement Analysis has been completed the next stage in a life cycle is to define and document the product requirements and for them to be approved by either the customer or group who requested the project to be developed. Requirements a documented in a Software Requirement Specification (SRS) which will contain all the project requirements that are to be designed and development as part of the project life cycle.

#### Design

A Document Design Specification (DDS) will then usually be produced, which uses the SRS created earlier as a reference. A DDS can be either a very high-level document or a very low-level document. A high-level DDS may only contain screenshots of what the user interface should look and where requirements are implemented. Whereas a low-level DDS will also describe of the software architecture that should be implement such as any databases, APIs and the classes and modules the code will include. Typically, more than one design will be produced for this document. (Tutorialspoint, 2018)

The DDS will then be reviewed by the key stakeholders involved in the project and depending on various the project parameters such as risk, cost and development time the most appropriate design will be chosen for implementation.

#### Building the Project

It is at this point in the SDLC that the actual development of the projects programming code is started. The code that is developed will follow the chosen design laid out in the DDS that was approved for implementation. It is important during the development process that the code written follows the organisations coding guidelines such as their choice of programming language and development tools.

#### Testing

Whilst testing should be carried out during all stages of the SDLC, this stage refers to the testing of the proposed completed project. Due to this stage of testing all bugs/issues should be reported, tracked and retested until the product is deemed to be stable and of a high enough standard as defined in the SRS that it can be released.

#### Deployment to the Market and Maintenance

Upon the completion of the Testing stage the product is then released formally into its intended market. Occasionally, the product will be released in a limited capacity to a User Acceptance Testing (UAT) environment. So that the intended end-user/customer can test the product and decide if they are satisfied with the product or if further development is required before formal release.

### Software Development Models

There are numerous approaches that can be followed for a software development lifecycle, with each methodology having their own advantages and disadvantages. The following software methodologies where reviewed and evaluated as part of my project, so that I could determine the most suitable methodology to follow for my project development life cycle.

* Waterfall Model
* Modified Waterfall Model
* Scrum Model
* Spiral Model

#### Waterfall Model

The Waterfall Model was developed in 1970 by Dr Winston W. Royce, it was the first process model that was introduced. The Waterfall model is very simple to understand and use with it emphasising a logical progression through the various stages of a SDLC. As such it requires that work is completed in a sequential manner which means a new piece of work cannot be started until the previous task has been completed. (TryQA.com, 2013)

Recently the popularity of the Waterfall model has fallen, due to the rise of the agile methodologies. This however does not mean that it is obsolete, as the logical nature of the waterfall model still retains its desirability for many software developers.



Figure 2.. Waterfall Model(Citta, 2017)

##### Advantages

* Adapts to shifting teams
* Forces a structured organisation
* Allows for early design changes
* Suited for milestone development

##### Disadvantages

* Nonadaptive design constraints
* Lack of user/client feedback mid-process
* Delayed Testing Period

#### Modified Waterfall.

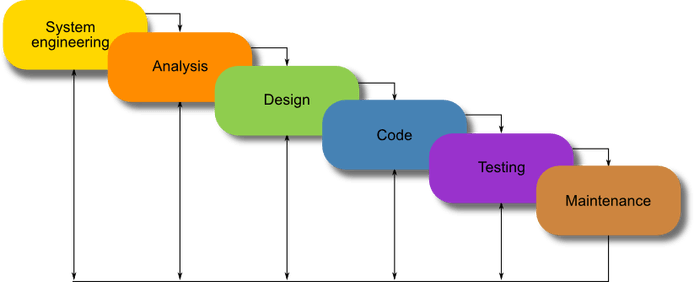
The modified waterfall model is almost identical to the waterfall model which it is based off. The modified waterfall model allows a developer to go back to an earlier task and update it if they release that something needs to be changed instead of going back starting all over again from the start.

Figure 2.. Modified Waterfall Model (Powell-Morse, 2016)

##### Advantages

* Adapts to shifting teams
* Forces a structured organisation
* Allows for early design changes
* Suited for milestone development
* Adaptive to Design Constraints

##### Disadvantages

* Lack of user/client feedback mid-process
* Delayed Testing Period

#### Scrum Model

The Scrum model is an agile methodology that suggests that work should be carried out in series of sprints with each sprint lasting for an agreed upon period of time. In the Scrum model at the start of each sprint the team members will determine the number of items they can commit to. The team members will then create a sprint backlog which is a list of all the tasks that are to be performed during the sprint.

As a part of the Scrum model that are two key roles which support the scrum teams, these the ScrumMaster, who could be thought of as the coach or captain of the team. It is the ScrumMasters’ role to help the team members involved in a sprint to use, the Scrum process, to allow them to perform at the highest level. (Mountain Goat Software, 2018)

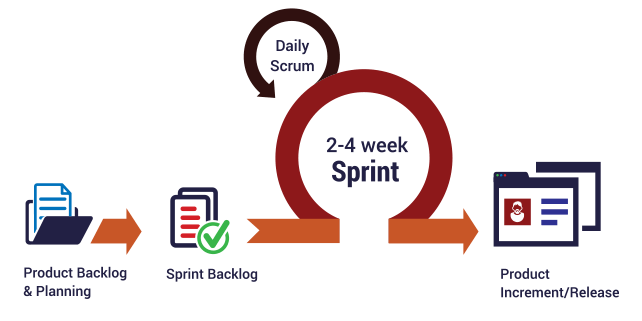


Figure 2.. Scrum Model (Intelegain Technologies, 2017)

##### Advantages

* Ensures effective use of time and money
* Projects are divided into easily managed sprints
* Development work is tested in-sprint
* Feedback from customers and stakeholders
* Adaptive to Design Constraints

##### Disadvantages

* Scope creep, due to lack of a definite end-date
* Requires experienced team members
* Team member can become frustrated due to daily team meetings
* Difficult to adopt in large teams

(Chandana, 2018)

#### Spiral Model

The Spiral model is an incremental Software Development Life Cycle which focuses on risk analysis. The Spiral model was first described by Barry Boehm in his 1986 paper "A Spiral Model of Software Development and Enhancement". The spiral model combines combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. This allows for the incremental release of the product or incremental refinement through each iteration around the spiral. In the spiral model the risk determines the amount effort required for each task as well as the amount of details required for a task to completed. (PAL, 2018)

##### Advantages

* Risk handling
* Good for large projects
* Flexibility in requirements
* Client/user feedback
* Adaptive to Design Constraints

##### Disadvantages

* Difficult to implement
* Expensive
* Highly dependent on Risk Analysis
* Difficult to manage time

#### Conclusion

Having reviewed the advantages and disadvantages of each of the four software development life cycles that are detailed in this report, I decided that the best approach for my project would be to follow the modified waterfall model.

The reason that I chose the modified waterfall model is that like I like the logical process that it follows when completing tasks. The reason that neither the Scrum or Spiral models were chosen is due to belief of the Project Lead that they are better suited to projects that are being developed by a team and not as a solo project.

During the project it is important that I fellow the project plan that will be created and watch out for slipping into the habit continuously going back and forth between the various stages in the plan to make small incremental changes which could result in a delay in the project’s planned completion date.

## Technical Background

### iPhone Operating System

Originally called the iPhone Operating System due to the release of the first iPhone in 2007 and later renamed to IOS with introduction of the Apple iPad. Nowadays iOS is used on the iPhone, iPad and the iPod Touch. Apple requires all their devices. Apps can be installed onto a device that runs on iOS via Apples’ AppStore which currently has approximately 2,000,000 apps available for download. In order to publish an app to the AppStore an Apple Developer account and Apple XCode are both required. It is important to note that Apple XCode can only be run on an Apple device running MacOS (i.e. Mac or MacBook). An Apple Developer Account has yearly fee for £79. From the period of November 2017 to November 2018 around 20.31% of all devices where running on iOS. (Nations, 2019)

### Android

Android is the mobile operating system that is most commonly found installed on most mobile devices. It was first released in 2009 by Google. To install an app on a device which utilises Android as its mobile operating system via the Google Play store. The Google Play store has approximately 2,600,000 apps available for download. In order to publish an app to Google Play a Google Developer account is required. This developer account has a one of fee of $25. From the period of November 2017 to November 2018 around 75.08% of all devices were running on Android.

### Chosen Platform

The solution developed for this project should work on both iOS and Android. The reason for this is that these two platforms have the largest market share with them having a total market share of over 95%, from November 2017 to November 2018.

A screenshot of a cell phone

Description automatically generated

Figure 2.. Mobile Operating Market Share (StatCounter, 2018)

### Development Tools and Languages

#### Xamarin

Xamarin is a cross-platform framework for mobile app development which brings .NET and C# to both the Android and the iOS platforms. It is an optional add-on feature which is offered for free as part of Microsoft Visual Studio. Xamarin allows the developer to create applications that have a native look and feel to them.

By using Xamarin a developer can access native APIs and SDKs that are constantly kept up to date by Microsoft to ensure that the application being developed will work on the latest target platform version.

Xamarin also allows the developer to shar large parts of their code across different target platforms (usually around 70-80%). This means that the developer does not have to waste time rewriting code to meet certain platform needs.

##### Requirements

* Visual Studio
* Xamarin add-on feature

#### Ionic

The Ionic Framework is an open source UI toolkit which can be used to develop and build high-quality mobile and desktop apps using existing web technologies such as HTML, CSS, and JavaScript. The Ionic Framework is currently integrated with Angular, and it focuses primarily on the frontend user experience or UI interaction of an app with aim being for an application which has been developed in the Ionic Framework to have the look and feel of a native application.

The Ionic Framework offers the advantage of being able develop native looking applications for multiple platforms simultaneously without needing to write parts of code for specific platforms.

The current version of the Ionic Framework in ‘Ionic 3’ and whilst ‘Ionic 4’ is also available for use it is currently in beta testing and is therefore not being considered as development option.

##### Requirements

* Node.js
* Git/GitHub Desktop
* Cordova
* Ionic CLI
* NPM

#### Chosen Development Tool and Language

After reviewing the possible development tools out there, it was decided that this project shall make use of the Ionic Framework and more specifically Ionic 3. The main reasoning behind this choice was that Ionic 3 is currently used by my current employer. This therefore means that I have access to the company’s documentation and best practices which are currently used when creating an application for any of our clients.

#### Data Storage Options

After a deciding upon the development tool that would be used for this project, it was important to determine which database option would be best suited for use in this project. Three options were considered.

##### SQLite

This is the default database option for both the Android and IOS platforms. SQLite has the advantage of being a lightweight, transitional database which requires very little storage space. SQLite databases can only be accessed by the device that it is currently installed on. This means that there is little risk to its use when it comes to storing personal information about a user.

##### Firebase

Firebase provides users with a real-time database and backend as a service. The service allows for developers to access an API which allows for application data to be synchronised across multiple devices and shared with multiple users. All data stored on Firebase is stored on Firebase’s cloud. It was first developed in 2011 by Firebase Inc. but it was later acquired by Google in 2014. Firebase also allows a develop to view analytics regarding app usage and user engagement. Firebase has a developer community of than more than 450,000 developers.

##### Local Storage

Local Storage differs from SQLite and Firebase in that you do not tell the application to use a specific database type. Instead Local Storage will use the default data storage option for the device in which the application is being ran on. This means that this option is perfectly suited to cross-platform development as there is no need to worry about if the device that the application is being run on, has the chosen data storage option available. The one drawback with Local Storage is that you not able to create a record that has multiple fields to each record, but this can be resolved by simply saving all the required data as JSON. An advantage using Local Storage is that it can only be accessed by the device it was stored on meaning there is a reduced risk of a security or data breach occurring.

##### Chosen Data Storage Option

This project shall use Local Storage for all the data storage needs for the application that will created as part of this project. The reason Local Storage was chosen was because the data that is to be stored via the application does not need to be shared with any other devices or users, and due to it defaulting to the preferred data storage method of the device the application is being run on.

### Resources Required

In order to publish my application to both Google Play and the AppStore the following resources are required:

* Apple Developer Account
* Google Developer Account
* XCode
* Android SDK Manager
* Adobe Experience Design (XD) – UI design
* RapidAPi – API Hub (intending to use the Nutrionix API)
* Git/GitHub Desktop (Document Control)
* Visual Studio Code

# Chapter 3: Requirement Analysis

## Requirement Gathering

As part of any project or project management it is essential that requirements gathering is carried out as part of it. This allows for stakeholders to develop a full understanding of what it will deliver that is critical to its success. It is important that this stage of a project is not overlooked, so that further down the line it is not realised that the customers’ needs have not been adequately met due to the customers’ requirements not being fully understood.

To give this project a greater chance at success the following rules will be applied to help ensure successful requirement gathering is carried out:

1. Don't assume you know what the customer wants - always ask.
2. Involve the users from the start.
3. Define and agree on the scope of the project.
4. Make sure requirements are SMART - specific, measurable, agreed upon, realistic and time-based.
5. Gain clarity if there is any doubt.
6. Create a clear, concise and thorough requirements document and share it with the customer.
7. Confirm your understanding of the requirements alongside the customer (play them back).
8. Avoid talking technology or solutions until the requirements are fully understood.
9. Get the requirements agreed with the stakeholders before the project starts.
10. Create a prototype, if necessary, to confirm or refine the customer's requirements.

(Haughey, 2014)

During the requirements gathering process, there some commonly found mistakes that should be avoided. These mistakes are:

* Basing a solution on complex or cutting-edge technology and then discovering that it cannot easily be rolled out in the 'real world'.
* Not prioritising the requirements, for example, 'must have', 'should have', 'could have' and 'would have' - known as the MoSCoW principle.
* Insufficient consultation with real users and practitioners.
* Solving the 'problem' before you know what the problem is.
* Lacking a clear understanding and making assumptions rather than asking.

(Haughey, 2014)

Two different methods where utilised when it came to requirements gathering for this project to help ensure that no requirements were missed. The two methods employed for this project was a Focus Group and Observation. Both techniques were carried out over a period of time.

The collated requirements where then separated out into Functional and Non-Functional Requirements.

### Focus Group

Focus Groups are used to help provide a large variety of possible requirements, by getting a group of people together to discuss the problem and think of possible solutions and desirable functionality. This technique is useful in that it allows people to bounce ideas of each other and therefore creating broader range of ideas.

In this case I gathered four people together who had an interest in keeping fit and tracking their diet and briefed them on the project background and my aims for the completed solution. Upon being briefed I then asked them about how they currently keep track of this information and what they would like to see in the proposed solution.

### Observations

Observations is another useful technique for requirement gathering. Observations work by a person observing the environment that the solution will used in or by observing solutions that are already on the market.

Due to the fact my proposed solution is something that I personally would use and have prior experience when it comes to solutions already on the market, this meant that I had a unique perspective when it came to gathering requirements. As part of this requirement gather technique, I looked at apps that I have already used previously, which are already on the market and I have used in the past. These apps are:

* Apple Health
* Nike Run Club
* MyFitnessPal & MapMyRun

#### Apple Health

Apple health is an app that becomes pre-installed on all iPhones. It was first released in 2014 upon the release of iOS 8.0. Upon review I found that Apple Health was the most basic in terms of overall design with the app simply keeping track of the user sleep patter, total, steps, total distance and weight. However, it is useful for storing medical data and if linked to an Apple iWatch it is even able to tell if there are any irregularities with a wearer’s heartbeat, which the user should see a medical professional about. The data that the app stores is viewable on daily basis and can also be viewed in a graph that allowed the user to keep track of changes in their daily life. Apple Health can also be integrated with a large number of third-party apps that are available for download on the Apple Appstore.

#### Nike Run Club

Nike Run Club is an activity tracking app that is produced by Nike. The app is available for free on both the Google Play store and the Apple AppStore. Nike Run Club had what I feel to be the best user interface out of all the existing solutions that where reviewed, with the app being very easy to use and understand. The app offers guided runs and exercise by professional athletes that was a nice feature as well as an interactive map which your route was tracked on. Also included is the ability to set yourself challenges create training plans to reach a certain distance on your runs. Also included in any training plans that are created, are links to exercises in the Nike Training Club which can guides you through many exercise routines. However, despite these positive features there was one big drawback in that it was not able to keep track of your daily diet.

#### MyFitnessPal & MapMyRun

MyFitnessPal and MapMyRun are both owned by the same company, in this case ‘Under Armour’, upon being purchased in 2015. A free version of both apps is available for download on the Google Play and Apple AppStore however, to get access to all the features available on both apps, a yearly subscription fee of £39.99 is required, making it one of the more expensive Mobile Health applications on the market. Due to them being owed by the same company a user is allowed to like the account they use on the apps together, this allows a user to view the combined data from the two apps however, I felt that this was a complicated process and the apps contain almost too much information, which made them quite complex to use and understand. Another downfall for these apps was the number adverts and restricted features which aren’t included unless they are bought by the user.

## Requirements

### Functional Requirements

A functional requirement is basically a requirement that specifies what a system ‘should’ do. A functional requirement will usually identify the behaviour or function of a feature which the end user will require the system to do. Table 1 lists all the functional requirements defined for this project:

|  |  |
| --- | --- |
| Number | Functional Requirement |
| FR-01 | The application shall work on both iOS and Android platforms |
| FR-02 | A new user shall be asked to set up an account |
| FR-03 | The app shall remember a user after first-time set up |
| FR-04 | The user shall be able to navigate the app via a Navigation menu pane |
| FR-05 | The app shall allow a user to be able to search for meals. |
| FR-06 | A user shall be able to log a meal for breakfast |
| FR-07 | A user shall be able to log a meal for lunch |
| FR-08 | A user shall be able to log a meal for dinner |
| FR-09 | A user shall be able to log a snack |
| FR-10 | A user shall be able to view all the food they have logged in the last 7 days |
| FR-11 | A user shall be able to view a calorie break down for all the food they have eaten for the current day. |
| FR-12 | The app shall allow the user to start a run |
| FR-13 | The app shall allow the user to end a run |
| FR-14 | The app shall record the time ran by a user |
| FR-15 | The app shall record the distance covered during a run |
| FR-16 | The app shall record the calories burned during a run |
| FR-17 | Upon a run ending the app shall show the route a user ran on a map |
| FR-18 | Upon a run ending the app shall show the distance a user ran |
| FR-19 | Upon a run ending the app shall show the calories burned by a user |
| FR-20 | Upon a run ending the app shall show the average pace of the user |
| FR-21 | The app shall show statistics on all the runs a user has completed |
| FR-22 | The app shall display the step count for a day |
| FR-23 | The app shall show the average distance walked/ran since the user started using the app |
| FR-24 | The app shall ask the user to select a gender upon initial set up |
| FR-25 | The app shall ask the user to enter their weight upon initial set up |
| FR-26 | The app shall be able to randomly suggest a meal for breakfast |
| FR-27 | The app shall be able to randomly suggest a meal for lunch |
| FR-28 | The app shall be able to randomly suggest a meal for dinner |
| FR-29 | The app shall display the recipe upon selecting a meal |
| FR-30 | The app shall provide a nutritional breakdown of any meals/snacks it returns |
| FR-31 | The app shall allow the user to compare their information against other app users |
| FR-32 | The app shall provide the user with a monthly summary of their workouts |
| FR-33 | The app shall allow the user to view information on their weekly diet in a graph |

Table Functional Requirements

### Non-Functional Requirements

A non-functional requirement simply describes ‘how’ are system should work upon completion. They help to determine the quality of the software and how it should be behave once developed. Table 2 list all the non-functional requirements that my completed projects shall meet.

|  |  |
| --- | --- |
| Number | Non-Functional Requirement |
| NFR-01 | The app shall have a consistent theme throughout the application |
| NFR-02 | The app shall be simple to understand and use |
| NFR-03 | The app shall work on devices of different screen sizes |
| NFR-04 | The app shall not crash |
| NFR-05 | The app shall be capable of running in the background |
| NFR-06 | The app shall react appropriately to changes orientation |
| NFR-07 | The app shall be easy to navigate |
| NFR-08 | The app shall conform to best practice and design principles for the chosen platforms |

Table Non-Functional Requirements

### Requirement Prioritisation Strategy

To ensure that this project is successful a requirement prioritisation strategy will be used. A requirement prioritisation strategy is used to determine which requirements should be implemented first and which ones can be delayed until later in the project’s development lifecycle. For this project the MOSCOW technique was employed.

The MOSCOW technique categorises, requirements into the following 4 categories:

* Must Do
* Should Do
* Could Do
* Won’t Do

The following tables show the MOSCOW rating for each of the non-functional and functional requirements.

|  |  |  |
| --- | --- | --- |
| Number | Functional Requirement | MOSCOW |
| FR-01 | The application shall work on both iOS and Android platforms | Must |
| FR-02 | A new user shall be asked to set up an account | Must |
| FR-03 | The app shall remember a user after first-time set up | Must |
| FR-04 | The user shall be able to navigate the app via a Navigation menu pane | Must |
| FR-05 | The app shall allow a user to be able to search for meals. | Should |
| FR-06 | A user shall be able to log a meal for breakfast | Must |
| FR-07 | A user shall be able to log a meal for lunch | Must |
| FR-08 | A user shall be able to log a meal for dinner | Must |
| FR-09 | A user shall be able to log a snack | Should |
| FR-10 | A user shall be able to view all the food they have logged in the last 7 days | Should |
| FR-11 | A user shall be able to view a calorie break down for all the food they have eaten for the current day. | Must |
| FR-12 | The app shall allow the user to start a run | Must |
| FR-13 | The app shall allow the user to end a run | Must |
| FR-14 | The app shall record the time ran by a user | Must |
| FR-15 | The app shall record the distance covered during a run | Must |
| FR-16 | The app shall record the calories burned during a run | Should |
| FR-17 | Upon a run ending the app shall show the route a user ran on a map | Should |
| FR-18 | Upon a run ending the app shall show the distance a user ran | Should |
| FR-19 | Upon a run ending the app shall show the calories burned by a user | Should |
| FR-20 | Upon a run ending the app shall show the average pace of the user | Should |
| FR-21 | The app shall show statistics on all the runs a user has completed | Should |
| FR-22 | The app shall display the step count for a day | Should |
| FR-23 | The app shall show the average distance walked/ran since the user started using the app | Won’t |
| FR-24 | The app shall ask the user to select a gender upon initial set up | Must |
| FR-25 | The app shall ask the user to enter their weight upon initial set up | Must |
| FR-26 | The app shall be able to randomly suggest a meal for breakfast | Could |
| FR-27 | The app shall be able to randomly suggest a meal for lunch | Could |
| FR-28 | The app shall be able to randomly suggest a meal for dinner | Could |
| FR-29 | The app shall display the recipe upon selecting a meal | Could |
| FR-30 | The app shall provide a nutritional breakdown of any meals/snacks it returns | Could |
| FR-31 | The app shall allow the user to compare their information against other app users | Won’t |
| FR-32 | The app shall provide the user with a monthly summary of their workouts | Won’t |
| FR-33 | The app shall allow the user to view information on their weekly diet in a graph | Won’t |
| NFR-01 | The app shall have a consistent theme throughout the application | Must |
| NFR-02 | The app shall be simple to understand and use | Must |
| NFR-03 | The app shall work on devices of different screen sizes | Must |
| NFR-04 | The app shall not crash | Must |
| NFR-05 | The app shall be capable of running in the background | Must |
| NFR-06 | The app shall react appropriately to changes orientation | Must |
| NFR-07 | The app shall be easy to navigate | Must |
| NFR-08 | The app shall conform to best practice and design principles for the chosen platforms | Must |

Table MOSCOW Rating for Functional and Non-Functional Requirements

## Risk Analysis

For any project to achieve a successful it is important that all project risks are identified early and managed appropriately to avoid time slippage, increased costs, corruption or even the total loss of the project. A risk can be thought as anything that may be a potential problem that may occur during the software development life cycle.

Table 4 below contains all the risks identified for this project.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | RISK | IMPACT | RESOLUTION |
| 1 | Developer illness | Reduced productivity. Project falls behind schedule. | Task buffers and trying to work ahead of schedule. |
| 2 | Lack of productivity | Project will fall behind schedule | Creation and following of a project plan will help ensure productivity. |
| 3 | Compromising on Design | Finished software may not follow the same theme and be complex to user. May lead to further coding being needed in later stages. | Designs for all screens will produced before any development commences. 2 weeks has been dedicated to this. |
| 4 | Data Loss | Project may need to be restarted or falls behind schedule due to having redo work. | Multiple copies of all work carried out will be kept. i.e. Cloud, USB, hard drive |
| 5 | Unidentified resources | Time wasted getting the unidentified resources. Fall behind schedule | All required resources will be identified early in the projects planning stage. |
| 6 | Unavailable resources | Resources may be identified which you may need to pay for or are unable to access. Workarounds could compromise project quality. Project falls behind schedule | All resources will be gathered in advance and checked that they are working correctly. |
| 7 | Unforeseen Circumstances | Both technical and personal issues which may result in work not being completed to schedule. Or having to change project direction. | Buffers are included in the project plan to account for unforeseen circumstances. |
| 8 | Requirement Creep | Constant updating/adding of requirements will result in work not being accounted for in the project plan | Requirements will be finalised and can only be changed should a review carried to determine the feasibility and impact on the projects planned completion date. |

Table Project Risk Analysis

### Risk Mitigation Strategy

Upon all project risks being identified and controls being implemented. All risks where put through a Probability-Impact Matrix to ensure that the risks were within acceptable tolerances.

Figures 3.1 and 3.2 show probability-impact matrices for before and after controls were implemented for al risks identified.

#### Pre-Controls



Figure 3.. Probability-Impact Matrix before controls were implemented

#### Post Controls



Figure 3.. Probability-Impact Matrix after controls were implemented

## Requirements Gathered During Development

During the development stage of the project there was the introduction of some new requirements as well as changes to the MOSCOW prioritisation of some of the existing requirements. In a SDLC these changes to the requirements are called ‘scope creep’, it is called this because the new requirements where not originally scoped and were therefore not planned for. Table 5 below shows this scope creep and their MOSCOW prioritisation (note: all scope creep changes are shown in red).

|  |  |  |
| --- | --- | --- |
| Number | Functional Requirement | MOSCOW |
| FR-26 | The app shall be able to randomly suggest a meal for breakfast | Won’t |
| FR-27 | The app shall be able to randomly suggest a meal for lunch | Won’t |
| FR-28 | The app shall be able to randomly suggest a meal for dinner | Won’t |
| FR-29 | The app shall display the recipe upon selecting a meal | Won’t |
| FR-34 | The app shall be integrated with the Apple Health application | Could |
| FR-35 | The app shall be able to save data to the Apple Health application | Could |
| FR-36 | The app shall be able to retrieve data from the Apple Health application | Could |

Table Scope Creep

# Chapter 4: Design

## Design Principles

When designing a new application, it is important to adhere to common design principles which the user will already be familiar with from other apps and websites. By employing common design principles, it means that the applications end user should be able to quickly understand the app by using their knowledge of existing applications. The common design patterns that will be followed in this project will be the use of the ‘Hamburger’ icon for applications menu pane. All app icons will be taken from [ionicons.com/](https://ionicons.com/) which, contains icons that were designed and created by the Ionic Framework team. By using these icons, it will ensure the final application feels like it is native app and is therefore familiar to the end user. It is also important for the application to be internally consistent. This means that the apps colour scheme, components and layout shall be consistent throughout the application to help ensure that the end user has a smooth experience when navigating through and using the application whilst avoiding inconsistencies which could result in the user to stop using the application.

To help ensure that application created as part of this project offers users a smooth user experience and is consistent throughout, Schneiderman’s 8 golden rules will be followed during its design phase. These 8 rules are:

1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialog to yield closure.
5. Offer simple error handling.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

## Flow Map

Flow maps are a technique which are employed by user interface and user experience designers to help them try and understand how the end user is likely going to interact with the application and what their decision-making processes will be based on. Figure 4.1 as seen below shows the flow map of how user may wish to interact with application for this project.

A picture containing text, whiteboard

Description automatically generated

Figure 4.. FFT User Flow

## Prototyping

The User Interface (UI) Design is the link between users and an application/website. A good UI design will include the basic design elements that need to be present so that someone can successfully navigate an application and make the appropriate decisions.

During the UI design stage, it is important to consider the User Experience (UX). The UX design is important due to the objective of UX design being to create a frictionless and enjoyable experience for the end user. By considering UX Design during the prototyping stage means that customer satisfaction should be boosted by providing better usability, accessibility and pleasure in the interaction they have with the application.

Using Adobe XD, a simple prototype design was created for this project, to help visualise what the application may look like upon completion of this project (Note the final application may differ in design). The following prototype design have developed on an iPhone X template.[[1]](#footnote-1)

### Colour Palette

One of the easiest and best ways to achieve consistency in any application or web site is to ensure that the application or website follows a set colour scheme. Table 6 below shows the colour scheme that was used throughout the application that was created as part of this project.

|  |  |  |  |
| --- | --- | --- | --- |
| Colour | Hex | RGB | Usage |
|  | #4A41C5 | 74, 65, 197 | Primary: used for the header bar and buttons |
|  | #32db64 | 50, 219, 100 | Secondary: used if for some reason the primary colour is not suitable |
|  | #f53d3d | 245, 61, 61 | Danger: used to show any errors that occur |
|  | #f4f4f4 | 244, 244, 244 | Light colour: used as the screen background colour and for the text colour if it is inside an object with the ‘Primary’ colour. |
|  | #222 | 34, 34, 34 | Dark: Used to show normal text which is shown on top of the ‘Light’ colour |

Table Colour scheme of project application

### App Badge



Figure 4.. App Badge

This is the app badge that will be displayed on a devices home screen for the application. This app badge will be the same for both iOS and Android platforms.

### Details Page

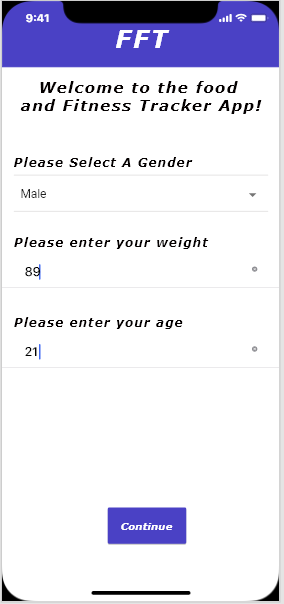


Figure 4.. Details Page

The Details page is where the user enters their personal details such as gender, age and weight. This data is then stored in an SQLite database upon pressing the continue button. At the top of the application is the app header bar. The header bar will always contain the text ‘FFT’, however the Details page will not contain the ‘Hamburger’ icon because the Details page will be displayed upon the app being opened for the very first time. The ‘Continue’ button will be disabled until each of the three fields on the Details page are populated with valid data. Upon data being populated into all three fields and the ‘Continue’ button will become enabled, which, upon being pressed will redirect the user to the ‘Diet Home’ page. This page can also be accessed via the ‘Personal Details’ menu option found in the ‘Main Menu’ pane.

### Diet Home Page

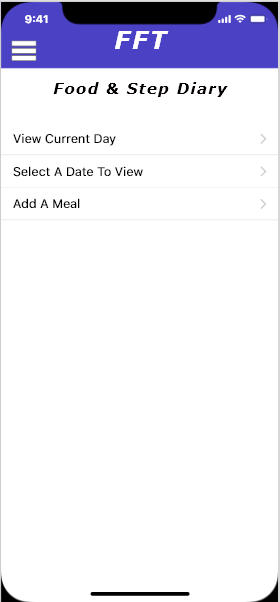


Figure 4.. Diet Home Page

The ‘Diet Home’ page is the main home page for the app. As such it will be the 1st page that will appear to the user if they have already set their personal details. From this page that a user can add a meal, view the food eaten for a date in the past or view their food diary for the current day. In the header bar the ‘Hamburger’ icon will be displayed. Which upon being pressed will open the main menu pane for the application.

### Main Menu Pane

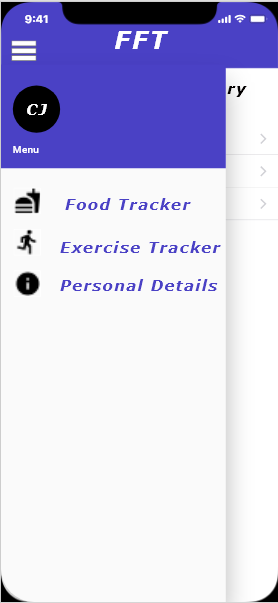


Figure 4.. Main Menu Pane

This is the Main Menu pane it can be accessed by the ‘Hamburger’ icon found in the header of the application. The Main Menu pane will display the following options ‘Food Tracker’ which will redirect you to the Diet Home page, the option ‘Exercise Tracker’ which will direct you to the Exercise Home page and the option ‘Personal Details’ which will direct the user to the Details page.

### Select Date Page

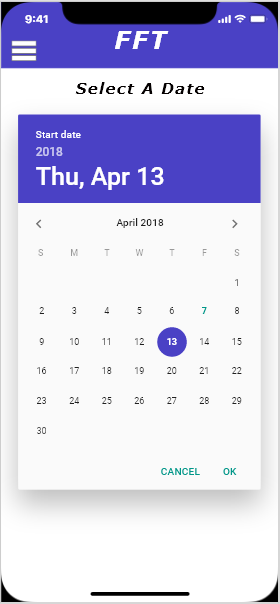


Figure 4.. Select Date Page

This is the ‘Select Date’ page, it is displayed upon the user pressing the option “Select Date to View” found on the ‘Diet Home’ page. This page will display the native date picker field for devices platform. Upon selecting a date and pressing the ‘Ok’ button will then redirect the user to the Food Diary page.

### Food Diary Page

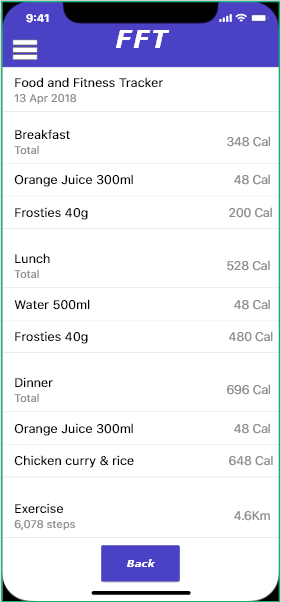


Figure 4.. Food Diary Page

This is the ‘Food Diary’ page, which will be displayed upon the user selecting a date on the ‘Select Date’ page or by the user pressing the ‘View current Day’ option found on the ‘Diet Home’ page. Tis page will display to the user information on what they have eaten for the selected date/current day as well as the total number of steps completed for the day. The ‘Back’ button will redirect the user to the ‘Diet Home’ page.

### Add Meal Page

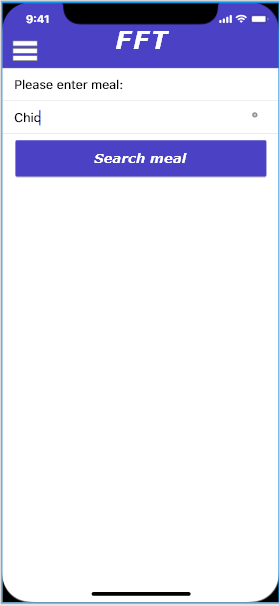
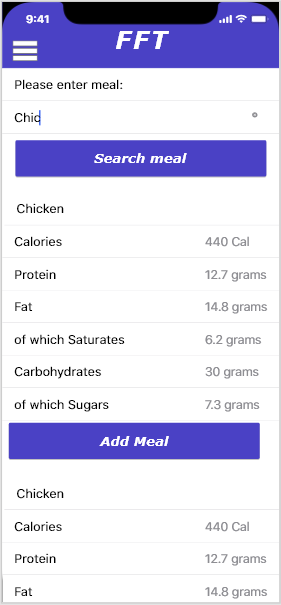


Figure 4.. Add Meal Page

This is the ‘Add Meal’ page, the image on the left shows the screen that will be displayed when the user navigates to this page via the ‘Add A Meal’ button located on the ‘Diet Home’ page. The ‘Add Meal’ page will contain a text input field and a search button that get all the matching results for the inputted text from the Nutritionix API and then display the returned results as seen in the second image. Each of the returned results will display some basic food macro data and an add meal button so that the user can add a meal to their food diary. Upon the page displaying the returned result the text input field and search bar will still be shown at the top of the screen. Should the user not find meal that they wish to add the user can opt to enter greater detail into the text input field and then press the search button again to update the results.

### Exercise Home Page

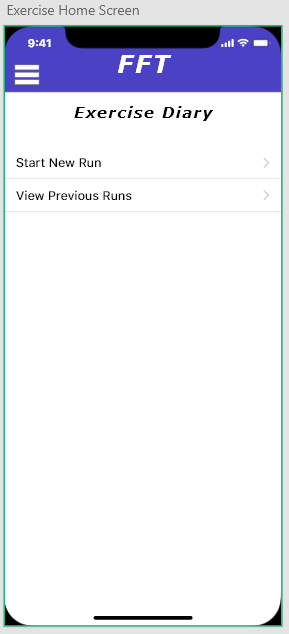


Figure 4.. Exercise Home Page

This is the ‘Exercise Home’ screen, from this page a user can navigate to either the ‘New Run’ page or the ‘Previous Runs’ page. This page is accessed via the option ‘Exercise Tracker’ found on the Main Menu pane. The ‘Hamburger’ icon is present in the left-hand corner of the header bar should the use wish to open the navigation pane.

### New Run Page

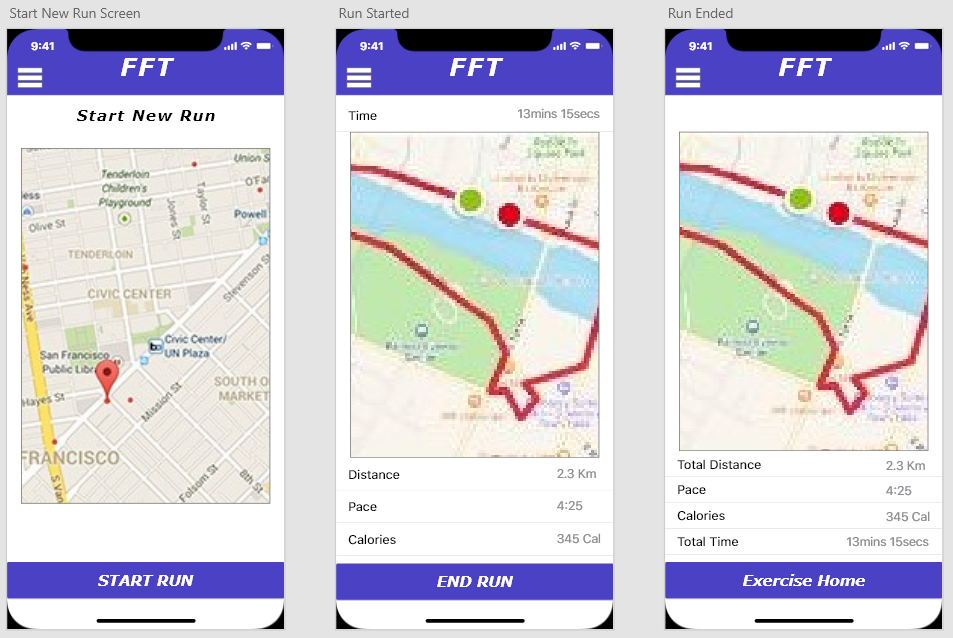


Figure 4.. New Run Page

These are the screens that will display for the ‘New Run’ page. This page can be accessed via the ‘Start New Run’ option found on the ‘Exercise Home’ page. When a user first navigates to this page they will see screen “Start New Run”. This screen will display a map showing their current location and a ‘Start Run’ button located at the bottom of page.

Upon pressing the ‘Start Run’ button the “Run Started” screen will be displayed; this screen will display map showing their current location as well as the route they have currently travelled. Below the map will be some basic information about their run such as the distance travelled, pace and calories burned. Finally, there will be an ‘End Run’ button at the bottom of the screen.

Upon the user pressing the “End Run” button the “Run Ended” screen will be displayed. This screen will display the route they travelled during their workout and some information about their run such total time, total distance, pace and calories. At the bottom of the page will be a button to the ‘Exercise Home’ page.

### Previous Runs Page

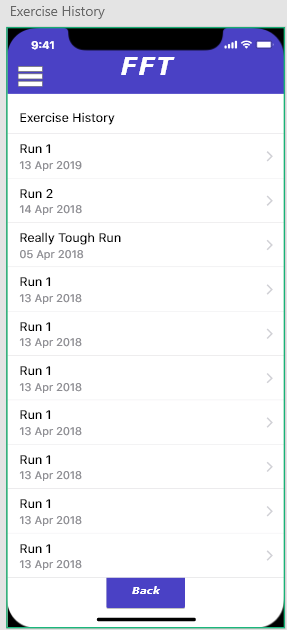


Figure 4.. Previous Runs Page

This is the ‘Previous Runs’ page which can be accessed via the option ‘View Previous Runs’ found on the ‘Exercise Home’ page. This page will display all the runs that a user has previously completed. Upon pressing on one of the previous runs listed that the user has completed, they will be directed to the Run Details page for that run. Upon pressing the ‘Back’ button they will be redirected back to the ‘Exercise Home’ page.

### Run Details Page

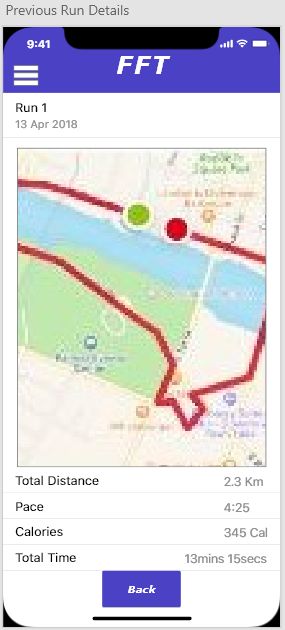


Figure 4.. Run Details page

The “Run Details” page as seen above is accessed by selecting a run listed on the ‘Previous Runs’ page. This page displays information about the selected run. The information that is displayed is the date the run was completed on, the route of the run as displayed on a map, the total distance, pace, calories and the total time the run took to complete.

## Data Storage

Due to the final application using Local Storage and not a set database type (i.e. SQLite or Firebase) means that all data that will saved by the application will be stored in either the ‘Details’, ‘Meals’ or ‘Routes’ table as a single field using JSON. Figure 4.1 shows what the database models would be shown as if SQLite or Firebase was used and what data will be present in the JSON for each table.

The Details table will be used to store the personal information about the applications user. The data that will be stored in the is table is their age, gender and weight. The Details table will only ever contain a single record for a user because the app should only ever be used by one user. Du to this presumption there is no need for any foreign keys in this table.

The Meals table will contain records containing information about all the meals that an app user has eaten and recorded during their time using the application. Each record in the table will contain the following information ‘meal description’, ‘date’, ‘calories’, ‘protein’ and as what meal type it belongs too (i.e. Breakfast, Lunch, Dinner or Snack). Each record will also contain a unique identifier number which will autoincrement upon a new record being added to the table. Due to the Meals table not being linked in any other table a foreign key is not necessary.

The Routes table will contain records containing information about all the workouts that an app user has completed using the application. Each record in the table will contain the following information ‘path route, ‘start date/time, ‘end date/time’, ‘speed’ and the ‘distance’ of each workout. Each record will also contain a unique identifier number which will autoincrement upon a new record being added to the table. Due to the Routes table not being linked in any other table a foreign key is not necessary.

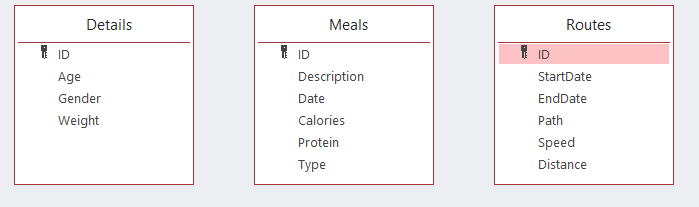


Figure 4.. Database Models

# Chapter 5: Implementation

The following section provides an overview of both the Applications System Architecture as well as a code overview which provides a breakdown of code snippets and its purpose in the application created as part of this project.

## System Architecture

Due to the code being written using the Ionic Framework, the application created by this project can be thought off as following the MVC (Mode-View-Controller) pattern. This is because most ionic apps can be broken down into 5 major areas: Views, Controllers, Data, App Configuration, and Directives. In an ionic app the MVC is made up of the Views, Controllers and the Data.

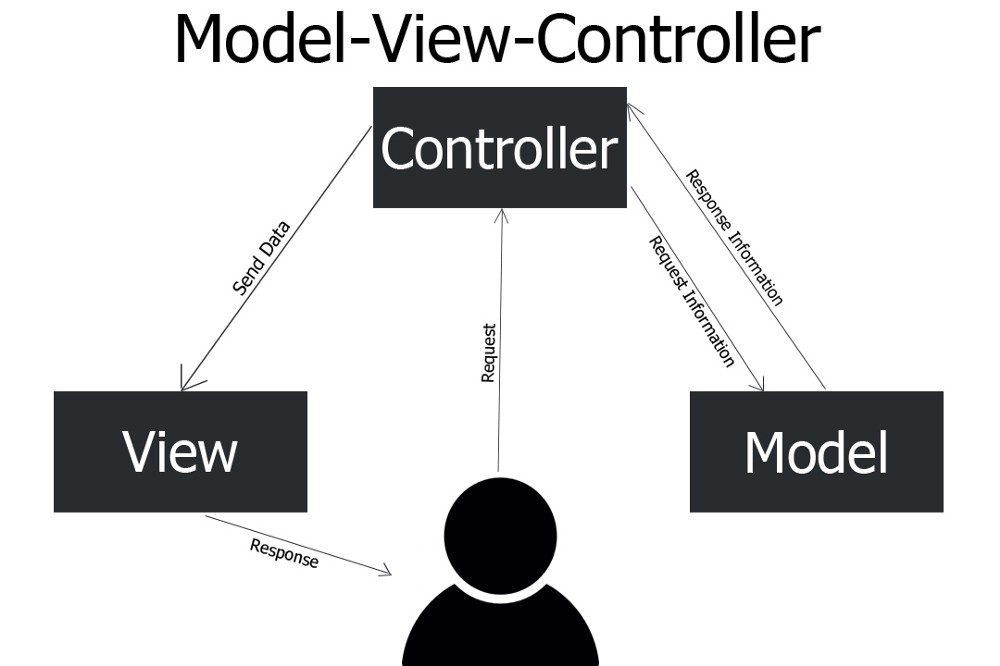


Figure 5.. Model-View-Controller (Margam, 2018)

### Views

In an Ionic App ‘Views’ are often be referred to as templates due to them being referred to as such by Angular controllers. As a result of this, in ionic ‘Views’ are often stored in a ‘/templates’ folder where each ‘View’ is in a separate .html file. It is also possible to include data bindings in an ionic ‘View’. (McGivery, 2014) An example of a ‘View’ with data bindings in ionic can be seen in the code snippet below:

<ion-view title="About">

<ion-content>

This is a view in Ionic! My Name is {{name}}

</ion-content>

</ion-view>

Figure 5.. Ionic ‘View’ example (McGivery, 2014)

In an ionic app the ‘Views’ should be found within the .html file of a page.

### Controllers

Controllers meanwhile can be thought of as the brains behind an application, where the flow of logic and data is being controlled. In Angular when you navigate to a page you are actually calling the ‘controller’. The controller will then, use a view as a template for the mark-up it will show to the user and, then make calls to the data layer classes (factories/services) so that the actual data is bound to the template. (McGivery, 2014)

### Model

Finally, the model can be thought of as the data (factories/services) as this provides the data to controller, normally through an external backend or web service. For this project the data layers are the mobile devices local storage and the Nutritionix Library in which API calls are made too.

## Code Overview

### services/database.ts

import { Injectable } from '@angular/core';

import { Storage } from '@ionic/storage';

import \* as uuid from 'uuid/v4';

@Injectable()

export class DatabaseService {

constructor(private \_db: Storage) { }

get meals() {

return this.\_db.get('meals');

}

get details() {

return this.\_db.get('details');

}

get routes() {

return this.\_db.get('routes');

}

insert(table: table, row: tableRow) {

return new Promise((resolve, reject) => {

this[table].then((data) => {

const arr = Array.isArray(data) ? [...data] : [];

const id = this.getUUID(arr);

arr.push({

id,

...row

});

this.\_db.set(table, arr).then(() => {

resolve();

}).catch((err) => {

reject(err);

})

});

})

}

set(table: table, row: tableRow) {

return new Promise((resolve, reject) => {

this.\_db.set(table, row).then(() => {

resolve();

}).catch((err) => {

reject(err);

})

})

}

getUUID(arr) {

const newId = uuid();

const exists = arr.find(f => f.id === newId);

if (exists) {

return this.getUUID(arr);

}

return newId;

}

}

Figure . service/database.ts code part 1

The ‘service/database.ts’ class is a helper service that allows methods and data to be available to various pages/components throughout the application. The ‘service/database.ts’ helper class contains the methods for storing and retrieving data from the local storage database. The reasoning behind the developer adding a helper class to the application was to avoid them having to repeated code in the various pages and components.

#### Get meals() Method

The ‘get meals()’ method when called returns all meals data that have been stored in the database.

#### Get details() Method

The ‘get details()’ method when called returns the user details data that have been stored in the database.

#### Get routes() Method

The ‘get routes()’ method when called returns all routes data that have been stored in the database.

#### insert(table: table, row: tableRow) Method

This method is used to insert any form of new data into the database that is not replacing any exiting data. It does this by creating a new ‘Promise’, a promise is simply a placeholder for a future piece of data that has not yet been determined yet. Promises are best used when a task is being performed asynchronously.

Two constants are then declared ‘arr’ and ‘id’. The constant ‘arr’ checks to see of the data is in the form of an array and if it is null the ‘?’ operator prevents an exception by stopping the project from evaluating the data should the data be either null or undefined. This is important when a program is being run asynchronously as the data will not always be set when it is first ran. If there is data present but is not in the form of an array it will then convert it into an array and assign the array to the constant. The constant ‘id’ meanwhile calls the method ‘get UUID()’ (see section 5.2.1.6 for explanation) and assigns the returned Unique User ID (UUID) to the constant ‘id’.

Upon the constants ‘arr’ and id’ having values assigned to them this data is then push to the local storage database.

#### set(table: table, row: tableRow) Method

The ‘set’ method is only called by the page ‘details’ page with this method being used to create a new personal details record, if one does not already exist or if one does exist then it updates the already stored data with the newly entered data.

#### getUUID(arr) Method

The ‘getUUID’ method is used to either generate a Unique User ID value for each ‘meal’ or ‘route’ that is added by user via the application.

type table = 'routes' | 'meals' | 'details';

type tableRow = RouteIn | MealIn | DetailIn;

export interface DBTable {

id: string

}

export interface RouteOut extends DBTable {

path: string,

startTime: string,

endTime: string,

speed: number,

dist: number

}

export interface MealOut extends DBTable {

meal: string,

desc: string,

calories: string,

date: string

}

export interface DetailOut extends DBTable {

gender: string,

age: number,

weight: number

}

export interface RouteIn {

path: string,

startTime: string,

endTime: string,

speed: number,

dist: number

}

export interface MealIn {

meal: string,

desc: string,

calories: string,

date: string

}

export interface DetailIn {

gender: string,

age: number,

weight: number

}

export interface Database {

details: DetailOut,

meals: MealOut[],

routes: RouteOut[]

}

Figure . service/database.ts code part 2

#### Local Storage

The type variable ‘table’ creates 3 new tables called ‘routes’, ‘meals’ and ‘details’, meanwhile type variable ‘tableRow’ denotes that rows for each table will be in the format as listed in the respective export interface methods ‘RouteIn’, ‘MealIn’ and ‘DetailIn’ this allows for the data in these records to returned as json and parsed. If the format of these rows was not defined by these export methods means that angular will not throw an error along the lines of “Property 'details' does not exist on type Object” which is Angular basically saying that doesn’t ‘understand’ the returned data.

### Details Page

#### Details.html

<ion-header>

<ion-navbar color="primary">

<ion-title>FFT - Setup</ion-title>

</ion-navbar>

</ion-header>

<ion-content padding>

<div text-center>

<h2>Personal Details</h2>

</div>

<form [formGroup]="myForm" (ngSubmit)="SubmitDetails()">

<ion-list>

<ion-item>

<ion-label>Gender</ion-label>

<ion-select placeholder="Select Gender" formControlName="gender">

<ion-option value="f">Female</ion-option>

<ion-option value="m">Male</ion-option>

</ion-select>

</ion-item>

<ion-item \*ngIf="gender.hasError('required')">

<ion-label style="color: red">\*Your Gender Is Required</ion-label>

</ion-item>

<br>

<br>

<ion-item>

<ion-label>Enter your Age</ion-label>

<ion-input formControlName="age" placeholder="Enter Age..." type="number"></ion-input>

</ion-item>

<ion-item \*ngIf="age.hasError('required')">

<ion-label style="color: red">\*Your age Is Required</ion-label>

</ion-item>

<br>

<br>

<ion-item>

<ion-label>Enter weight in KG</ion-label>

<ion-input formControlName="weight" placeholder="Enter weight..." type="number"> </ion-input>

</ion-item>

<ion-item \*ngIf="weight.hasError('required')">

<ion-label style="color: red">\*Your Weight Is Required</ion-label>

</ion-item>

</ion-list>

<br>

<button ion-button full [disabled]="!myForm.valid" type="submit">Set Details</button>

</form>

</ion-content>

Figure . Details.html code

The Details.html code makes is form the requires a user to enter their details for the three required fields. Upon successfully completing the form the method ‘submitDetails()’ shall be executed. Should the user not enter their data the following will occur

* Warning Message: “\*Your Gender Is Required” (in red)
* Warning Message: “\*Your age Is Required” (in red)
* Warning Message: “\*Your Weight Is Required” (in red)
* Set Details button will be disabled

Upon data being entered for each of the fields and passing the regex validation for each field, the ‘Set Details’ will become enabled allowing the user to save the entered data into database via the execution of the ‘submitDetails()’ method. The user will then be redirected to the Home page.

#### Details.ts

import { Component } from '@angular/core';

import { FormBuilder, FormControl, FormGroup, Validators } from '@angular/forms';

import { NavController } from 'ionic-angular';

import { DatabaseService, DetailOut } from '../../services/database';

import { HomePage } from './../home/home';

@Component({

selector: 'page-details',

templateUrl: 'details.html',

})

export class DetailsPage {

myForm: FormGroup;

gender: any;

age: any;

weight: any;

constructor(public navCtrl: NavController, public formbuilder: FormBuilder, private dbServ: DatabaseService) {

this.myForm = formbuilder.group({

gender: new FormControl('', Validators.required),

age: new FormControl('', [Validators.required, Validators.minLength(1), Validators.maxLength(2)]),

weight: new FormControl('', [Validators.required, Validators.minLength(1), Validators.maxLength(3)])

})

this.gender = this.myForm.controls['gender'];

this.age = this.myForm.controls['age'];

this.weight = this.myForm.controls['weight'];

this.dbServ.details.then((data: DetailOut) => {

if (data) {

this.myForm.get('gender').setValue(data.gender);

this.myForm.get('age').setValue(data.age);

this.myForm.get('weight').setValue(data.weight);

}

})

}

ionViewDidLoad() {

console.log('ionViewDidLoad DetailsPage');

}

SubmitDetails() {

const model = {

gender: (this.myForm.get('gender').value) as string,

age: +this.myForm.get('age').value,

weight: +this.myForm.get('weight').value,

};

this.dbServ.set('details', model).then(() => {

this.navCtrl.setRoot(HomePage);

this.navCtrl.popToRoot();

});

}

}

Figure . Details.ts code

##### Constructor

In the constructor a new formgroup is declared which shall consist of the following fields:

* Gender
* Age
* Weight

All of these fields have been marked as required fields as seen by the use of ‘Validators.required’ and the fields ‘Age’ and ‘Weight’ both have further validation. The age field also has validation stating that it must have a minimum length of 1 and maximum length of 2 this means that the value entered must be between 0-99. Whilst the weight field has the validation that the vale entered must be at minimum 1 value in length and a maximum of length of 3 values i.e. 0-999.

The code “this.dbServ.details.then((data: DetailOut)” is checking to see if any data already exists in the details table. If data is found is ford the method ‘DetailOut’ is called from the services/database.ts helper class which denotes the format of the returned json allowing it to parsed and assigned to variables for age, gender and weight.

##### SubmitDetails() Method

This method is called upon the successful completion of the form. It starts by creating a new constant in this case called model which is set as the values for gender (in string format), age and weight. The ‘Set’ helper method is then called for the details table which is updated with the data stored the constant ‘model’. Once this is done the user is then redirected to the ‘Home’ page of the app.

### Home Page

The Home page is simply a page that allows the user to navigate to other page of the application with their only being one piece of logic being applied to the page.

checkForDetails() {

this.dbServ.details.then((details) => {

if (isEmpty(details)) {

this.navCtrl.push(DetailsPage);

}

});

}

Figure . CheckFordetails() Method

The ‘checkForDetails()’ method calls the databaseServer helper class (i.e. ‘dbServ’) and checks to see the details table contains any data via the ‘isEmpty’ function. If it finds that the details table does not currently have any data stored in it then it will redirect the user to the Details page

### Date Page

<ion-item>

<ion-label>Select A Date</ion-label>

<ion-datetime displayFormat="DD/MM/YYYY" picker-format="DD MMMM YYYY" [(ngModel)]="date"></ion-datetime>

</ion-item>

<br>

<br>

<div text-center>

<button ion-button align-center (click)="viewDate()" round>View Date</button>

Figure . Date.html Code

This page simply contains a datepicker field which binds the selected date to the variable ‘date’ and a ‘View Date’ button calls the method ‘viewDate()’.

import \* as moment from 'moment';

date: any;

constructor(public navCtrl: NavController) { }

viewDate() {

this.navCtrl.push(FoodLogPage, {

selectedDate: moment(this.date, 'YYYY-MM-DD').format('DD/MM/YYYY')

});

}

Figure . Date.ts Code

The Date.ts file uses the moment directive of Angular in the ‘viewDate()’ method. This method simply takes the select date and using the moment directive reformats it into the format ‘DD/MM/YYYY’. This date value is assigned to the variable ‘selectedDate’ and which is then pushed to the ‘Food Log Page’ so the selected date value can be utilised in it.

### Food Log Page

#### Selected Date Code

The food log page will display all of the items of food for the each meal type (i.e. Breakfast, Lunch, Dinner, Snack) and if available to the stepcount for the device the application is being run on.

<div text-center>

<h2>Food and Step Diary</h2>

</div>

<div>

<label text-left>Date Selected: {{selectedDate}}</label>

</div>

Figure . Food-Log.html code (selectedDate variable)

The html shown in figure 5.10. is simply displaying the variable selectedDate as set in the food-log.ts file as seen below:

this.selectedDate = navParams.get('selectedDate') || moment().startOf('day').format('DD/MM/YYYY');

Figure . food-Log.ts code (selectedDate variable)

To view the food log page, you must navigate to their either directly from the home page to view the current day or by the date page. If the user has navigated to the page via the date page then it binds the parameter that was sent to it as part of the ‘viewDate()’ method and binds it to the variable via ‘this.selectedDate’. However, should the user have navigated to the Food Log page directly from the Home page then, then current device date is retrieved via the moment directive and formatted such that it shows as ‘DD/MM/YYYY’ and assigned to the variable via ‘this.selectedDate’. This variable is then outputted in the html by it being called via the binding ‘{{selectedDate}}’.

#### Meal Type code

The following code is repeated/executed for each of the meal types listed on the page (i.e. Breakfast, Lunch, Dinner, Snack’).

<ion-list>

<h4>Breakfast<small \*ngIf="break.items.length"> - Total Calories: {{ break.totalCal }}</small></h4>

<ion-item \*ngFor="let item of break.items">

<ion-label text-Left>{{item.name}}</ion-label>

<ion-label text-right>{{item.calories}} Cal</ion-label>

</ion-item>

</ion-list>

Figure . Meal type html code

The line “\*ngIf="break.items.length">” checks to see if any data is returned for the Breakfast meal Type and if so it displays the total calories of all the items via the data binding “{{ break.totalCal }}” that fall under the chosen meal type (in this case ‘Breakfast’).

The for statement “\*ngFor="let item of break.items">” returns and displays the item description and the calories via the data bindings “>{{item.name}}” and “{{item.calories}}” for each item stored in the database for the chosen meal type (in this case ‘Breakfast’). Figure 5.13 below shows the logic behind the html of the meal type data.

getMeals() {

this.dbServ.meals.then((meals: MealOut[]) => {

meals = isEmpty(meals) ? [] : meals;

const byDate = meals.filter(f => f.date === this.selectedDate);

const breakfast = this.formatRows(byDate.filter(f => f.meal === 'Breakfast'));

this.break = {

totalCal: Math.round(breakfast.reduce((a, b) => a + (+b.calories), 0)),

items: breakfast

}

const lunch = this.formatRows(byDate.filter(f => f.meal === 'Lunch'));

this.lunch = {

totalCal: Math.round(lunch.reduce((a, b) => a + (+b.calories), 0)),

items: lunch

}

const dinner = this.formatRows(byDate.filter(f => f.meal === 'Dinner'));

this.dinner = {

totalCal: Math.round(dinner.reduce((a, b) => a + (+b.calories), 0)),

items: dinner

}

const snack = this.formatRows(byDate.filter(f => f.meal === 'Snack'));

this.snack = {

totalCal: Math.round(snack.reduce((a, b) => a + (+b.calories), 0)),

items: snack

}

this.totalCal = byDate.reduce((a, b) => a + (+b.calories), 0);

});

}

Figure . Meal Type Logic

The ‘getMeals()’ method first calls the table meals from the database via the code shown in figure 5.14 shown below and denotes that any data returned will be returned in the json format listed in the helper method ‘MealOut’.

this.dbServ.meals.then((meals: MealOut[]) => {

meals = isEmpty(meals) ? [] : meals;

const byDate = meals.filter(f => f.date === this.selectedDate);

Figure . Meals table call

The ‘isEmpty’ function checks if the meals table if is empty and if so the ‘?’ operator prevents the following error “Property 'meals' does not exist on type Object” by preventing it from evaluating the data should it be null or undefined. The constant ‘byDate’ is then declared and set. The ‘byDate’ constant is set to only be the json data which is filtered by the date value stored in the ‘selectedDate’ variable.

Following this a constant is declared for the meal type ‘Breakfast’ which calls the ‘formatRows’ method, which formats an array of data as shown in Figure 5.15 below. This method maps each of the returned json items as either an item ‘name’ or item ‘calories’. The array ‘byDate’ which is formatted by the ‘formatRows()’ method is also filteredso that only the data which is of the type ‘Breakfast’ is formatted and stored as the constant ‘Breakfast’.

formatRows(arr) {

return arr.map(m => {

return {

name: m.desc,

calories: m.calories

};

})

}

Figure . FormatRows() Method

Following this the total calories is then calculated for each of the items stored for the array of items for the constant ‘Breakfast’.

All of the code that is ran for the meal type ‘Breakfast’ is then repeated for the meal types ‘Lunch’, ‘Dinner’ and ‘Snack’.

#### Step Count Code

The final piece of logic that is ran for the Food Log page is for calculating the step count if available on the user’s device. The logic behind calculating the step count value is shown in figure 5.16 and figure 5.17.

this.plt.ready().then(() => {

if (this.plt.is('cordova')) {

this.healthKit.available().then(available => {

if (available) {

// Request all permissions up front if you like to

var options: HealthKitOptions = {

readTypes: ['HKQuantityTypeIdentifierStepCount', 'HKWorkoutTypeIdentifier', 'HKQuantityTypeIdentifierActiveEnergyBurned', 'HKQuantityTypeIdentifierdistanceWalkingRunning'],

writeTypes: ['HKWorkoutTypeIdentifier', 'HKQuantityTypeIdentifierActiveEnergyBurned', 'HKQuantityTypeIdentifierdistanceWalkingRunning']

}

this.healthKit.requestAuthorization(options).then(\_ => {

this.loadstepData();

})

}

});

} else {

this.loadstepData();

}

})

Figure . Code for HealthKit availability

The code shown in figure 5.16 is ran inside the constructor. It first checks on the status of platform environment, following this it then checks if ‘cordova’ is available. This will only be available if the application is being ran on a mobile device and not as a progressive web application on a pc. Should Cordova be available it will then check to see if the user has allowed the application access to Apple HealthKit. Apple HealthKit will only be available for on iOS devices. Should HealthKit be available on the user’s device, it will then request read and write access various Apple HealthKit variables. Upon this request being authorised the method ‘loadStepData()’ is called. Should the user not have access to Cordova then the ‘loadstepData()’ method will still be called due to the fact dummy data is outputted if this is not available.

//Retrieve the stepcount on the users device if available

loadstepData() {

var stepOptions = {

startDate: moment().startOf('day'),

endDate: moment().endOf('day'),

sampleType: 'HKQuantityTypeIdentifierStepCount',

unit: 'count'

}

if (this.plt.is('cordova')) {

this.healthKit.querySampleType(stepOptions).then(data => {

let stepSum = data.reduce((a, b) => a + b.quantity, 0);

this.stepcount = stepSum;

}, err => {

console.log('No steps: ', err);

});

} else {

this.stepcount = 15000;

}

}

Figure . loadstepData() Method

The ‘loadstepData’ method first creates a new variable called ‘stepOptions’ which sets the parameters that apple HealthKit requires in order to return the step count for the iOS device. It then checks if Cordova is available, if it is available it then queries the HealthKit data using the step options set earlier to get the step count data. Then using ‘let’ variable a new parameter ‘stepSum’ which is of unique scope which is assigned the returned data which is reduced down to a single value. The variable ‘stepcount’ is then set to equal the value stored for ‘stepSum’. Should the step count not be available for the iOS device, possibly because permission to Health data has not been given then a console message shall be displayed. Should Cordova not be available when checked at the start of this method then the stepcount variable will be set to equal ‘15000’ (this is simply dummy data for when using the application as a progressive web app). (Morony, 2018)

Figure 5.18 below shows how the step count value bound to the variable ‘stepcount’ which is declared in the food-log.ts page. This value is marked as a read-only input field to prevent the user from trying to change the displayed data.

<h4>Exercise</h4>

<ion-item>

<ion-label text-left>Steps: </ion-label>

<ion-input text-left type="text" [(ngModel)]="stepcount" readonly></ion-input>

</ion-item>

Figure . Step Count html code

### Search Meal Page

#### Search Meal

<div padding>

<h2>Search Food</h2>

<ion-item>

<ion-input type="text" [(ngModel)]="searchMeal" placeholder="Enter Food item"></ion-input>

</ion-item>

<button ion-button align-center (click)="mealSearch($event,food)" block>Seach For Food Item</button>

</div>

Figure . Search Food html code snippet

The code shown above in Figure 5.19 shows the html code for search for items that can be added to a meal. An input field is present which allows a user to enter data, which is bound to the parameter ‘searchMeal’. There is then a button which upon being pressed will call the method ‘mealSearch’ which passes a parameter called ‘food’.

searchMeal: any;

food = new BehaviorSubject([]); //automatically updates on page upon changes being made

apiBaseURL: string = '';

data: string;

params: string = '?fields=item\_name%2Cnf\_calories%2Cnf\_total\_fat%2Cnf\_protein%2Cnf\_saturated\_fat%2Cnf\_total\_carbohydrate&appId=b15cf509&appKey=+ef180408b108a12f463c2eb4e1ea8a66'; //API params

break: string = 'Breakfast';

lunch: string = 'Lunch';

dinner: string = 'Dinner';

snack: string = 'Snack';

date = new Date().toISOString();

cal: string;

name: string;

toastOptions: ToastOptions;

addToMeal: Array<string> = [];

key: string;

//private database: FoodDatabaseProvider, private storage: Storage

constructor(

public navCtrl: NavController,

public navParams: NavParams,

private http: HttpClient,

private dbServ: DatabaseService,

private toast: ToastController

) {

//API construtor

this.apiBaseURL = "https://api.nutritionix.com/v1\_1/search/";

this.searchMeal = navParams.get('food');

this.mealSearch;

// Method that for the API call

mealSearch(event, food) {

this.http.get(this.apiBaseURL + this.searchMeal + this.params)

.subscribe(data => {

this.food.next(data['hits'].map(m => m.fields));

console.log(JSON.stringify(data)); // outputs the json to the console log

}, err => {

console.log(err); //should the call not work, return an error

});

}

Figure . Search Meal typescript logic

The first action that this method runs is to call the Nutritionix API with a *GET* call which is built using the following variables ‘this.apiBaseURL’, ‘this.searchMeal’ and ‘this.params’. The returned json data is then assigned to the *BehaviourSubject* variable ‘food’ which listens and updates the page upon the data inside it being changed. During the assigning the json data to ‘food’ it manipulated so that only the returned json data which falls under the title ‘fields’ is assigned to the variable. Should the api call fail for some reason an error message shall be displayed in the console. The *BehaviourSubject* array ‘food’ is then outputted to the html view as seen in figure 5.21 so that a user can select an appropriate food item and add it to a meal type or search for a different food item.

<div \*ngFor="let item of food | async">

<ion-card>

<br>

<h2 text-center ngDefaultControl [(ngModel)]="name">{{item.item\_name}} </h2>

<ion-item>

<ion-label text-left>Calories</ion-label>

<ion-label text-right ngDefaultControl [(ngModel)]="cal">{{item.nf\_calories}} Kcal</ion-label>

</ion-item>

<ion-item>

<ion-label text-left>Protein</ion-label>

<ion-label text-right>{{item.nf\_protein}} grams</ion-label>

</ion-item>

<ion-item>

<ion-label text-left>Fat</ion-label>

<ion-label text-right>{{item.nf\_total\_fat}} gams</ion-label>

</ion-item>

<ion-item>

<ion-label text-left>Saturated Fat</ion-label>

<ion-label text-right>{{item.nf\_saturated\_fat}} grams</ion-label>

</ion-item>

<ion-item>

<ion-label text-left>Carbohydrates</ion-label>

<ion-label text-right>{{item.nf\_total\_carbohydrate}} grams</ion-label>

</ion-item>

<button ion-button ngDefaultControl (click)="addMeal(item, 'Breakfast')" block name="break">Breakfast</button>

<button ion-button ngDefaultControl (click)="addMeal(item, 'Lunch')" block name="lunch">Lunch</button>

<button ion-button ngDefaultControl (click)="addMeal(item, 'Dinner')" block name="dinner">Dinner</button>

<button ion-button ngDefaultControl (click)="addMeal(item, 'Snack')" block name="snack">Snack</button>

</ion-card>

Figure . Display Food Results code

The returned json is then put through a for statement that creates a new card item for each returned result and its associated food macro information. The name for each returned food item and the macro information for the food item is then displayed inside of the card. The ‘name’ and ‘calories’ values are also bound to the variables ‘name’ and ‘cal’, which are only utilised if one of the 4 buttons at the bottom of each card is pressed. Each of these buttons represents a different meal type which a meal can be added to in the database. Upon one of the four buttons being pressed the ‘addMeal()’ method is then executed.

#### Add Meal

The ‘addMeal()’ method as shown below in Figure 5.22 is what is used to actually store the selected food item to the database.

constructor(

public navCtrl: NavController, public navParams: NavParams, private http: HttpClient, private dbServ: DatabaseService, private toast: ToastController

) {

//Set the Toast options

this.toastOptions = {

message: 'Meal Successfully Added',

duration: 4000

}

}

addMeal(item, meal: string) {

const model: MealIn = {

meal,

desc: item.item\_name,

calories: item.nf\_calories,

date: moment().format('DD/MM/YYYY')

}

this.dbServ.insert('meals', model).then(() => {

this.toast.create(this.toastOptions);

this.navCtrl.setRoot(HomePage);

this.navCtrl.popToRoot();

});

}

Figure . addMeal() Method

As part of the ‘addMeal()’ method it expects for two parameters to be present when it is called. The first is ‘item’ which the returned API data that is being added to the database. The second parameter is a string and this string should denote the meal type that the food item is being added as (i.e. breakfast, lunch, dinner or snack). The method create a new constant called ‘model’ which uses the format as denoted in the ‘MealIn’ helper method. The method uses the ‘services/database.ts’ helper class again by calling the ‘insert()’ method which inserts the data stored in the constant ‘model’ as a new json record in the database table. A toast message is then displayed, and the user is returned to the Home page.

## Add Run Page

<ion-content padding>

<button ion-button full icon-left (click)="startRun()" \*ngIf="!isTracking">

<ion-icon name="walk"></ion-icon>

Start Run

</button>

<button ion-button full icon-left (click)="stopRun()" \*ngIf="isTracking">

<ion-icon name="hand"></ion-icon>

End Run

</button>

<div #map></div>

<ion-list>

<ion-list-header>Previous Workouts</ion-list-header>

<ion-item \*ngFor="let route of previousRoutes">

{{route.finished | date}}, {{route.path.length}} Waypoints

<button ion-button clear item-end (click)="showHistoricRunRoute(route.path)">View Route</button>

</ion-item>

</ion-list>

Figure . Start/End Run html code

Upon opening the Add Run page of the application the user will be see a Start Run button and a map showing their current location (only if on a mobile device). Upon pressing the ‘Start Run’ the app will start tracking a user’s location and display it on the map. The Start Run button will change to show the ‘End Run’ button. Finally, at the bottom of the page is a list of all the previous runs that user has completed using the application. Each list item displayed under the Previous Runs heading will contain the Date of the run the number of waypoints on the run and a view route button which upon being pressed executes the ‘showHistoricRunRoute(route.path)’ method which upon being pressed display the route of the run on the map.

ionViewDidLoad() {

this.platform.ready().then(() => {

this.loadHistoricRoutes();

if (this.platform.is('cordova')) {

let mapOptions = {

zoom: 13,

mapTypeId: google.maps.mapTypeId.ROADMAP,

mapTypeControl: false,

streetViewControl: false,

fullScreenControl: false

};

this.map = new google.mpas.Map(this.mapElement.nativeElement, mapOptions);

this.geolocation.getCurrentPosition().then(pos => {

let LatLng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);

this.map.setCenter(LatLng);

this.map.setZoom(15);

});

}

});

}

Figure . Add Run Page Load logic

Upon the Add Run page being loaded the method ‘ionViewDidLoad()’ is executed. This method first checks on the environment status that the application is being run on. The method then calls the method ‘loadHistoricRoutes()’ as seen below in figure 5.25.

loadHistoricRoutes() {

this.dbServ.routes.then((historic: RouteOut[]) => {

historic = isEmpty(historic) ? [] : historic;

this.previousRoutes = historic.map((m) => ({ finished: m.endTime, path: JSON.parse(m.path) }));

});

}

Figure . loadHistoricRoutes Method

The ‘loadHistoricRoutes()’ method calls the routes table from the database, should the it be empty the ‘?’ operator will cause it stop evaluating the routes table should it be null or undefined. If there is data present then the json is assigned to the variable ‘historic’. This is then mapped to the the variable ‘previousRoutes’ which define each returned item as either the end time and the path that was followed.

After executing this method the ‘ionViewDidLoad()’ method then checks to see if Cordova is available. If it is the code then declares the map option parameters. Followed by creating a new google map object. The user current location is then retrieved and displayed on the map with the map being centred on this location.

### Start Run method

// Function to start tracking a users workout

startRun() {

this.isTracking = true;

this.trackedRoute = [];

this.startTime = new Date().getTime().toString();

this.positionSubscription = this.geolocation.watchPosition()

.filter(p => p.coords !== undefined) //Filter out any error that occur

.subscribe(data => {

setTimeout(() => {

this.trackedRoute.push({ lat: data.coords.latitude, lng: data.coords.longitude });

this.redrawPath(this.trackedRoute);

this.speed.push(data.coords.speed);

});

})

}

Figure . startRun() method

Upon the ‘Start Run’ button being pressed the ‘startRun()’ method is executed. First it set the variable ‘isTracking’ to ‘*true’* which causes for the ‘Start Run’ button to no longer be shown and the ‘End Run’ button to now display in its place. The variable trackedRoute is declared as an empty array and the variable Start time is set as the current time in string format. The code the tracks and records the user’s current location with it filter out any errors that are returned. These changes in position are then pushed to the array ‘trackedRoute’. The method ‘redrawPath()’ (figure 5.27) is then called which draws a path between each set of coordinates stored in the array ‘trackedRoute’ and displays the path on the map on the application page. The users speed at each set of coordinates is also recorded.

// Draws the path the user took on their workout in the map

redrawPath(path) {

if (this.currentMapTrack) {

this.currentMapTrack.setMap(null);

}

if (path.length > 1) {

this.currentMapTrack = new google.maps.Polyline({

path: path,

geodesic: false,

strokeColor: '#ff00ff',

strokeOpacity: 1.0,

strokeWeight: 3

});

this.currentMapTrack.setMap(this.map);

}

}

Figure . redrawPath() method

### Stop Run Method

Finally, when the user presses the ‘Stop Run’ button the method ‘stopRun()’ is executed. The ‘stopRun()’ method as seen below in figure 5.28.

stopRun() {

let newRoute = { finished: new Date().getTime(), path: this.trackedRoute };

this.finishTime = new Date().getTime().toString();

this.previousRoutes.push(newRoute);

//Calculate Average Speed

let totSpeed = 0,

avgSpeed = 0;

if (this.speed.length && this.trackedRoute.length) {

for (let i = 0; i <= this.trackedRoute.length; i++) {

totSpeed = totSpeed + (this.speed[i] || 0);

}

avgSpeed = totSpeed / this.speed.length;

}

//Calculate total distance

let dist = 0;

for (let i = 0; i <= this.trackedRoute.length; i++) {

dist = dist + get(this.trackedRoute, '[i].distance.value', 0);

}

const model: RouteIn = {

path: JSON.stringify(this.trackedRoute),

startTime: this.startTime,

endTime: this.finishTime,

speed: avgSpeed,

dist: dist

};

this.dbServ.insert('routes', model).then(() => {

this.isTracking = false;

this.positionSubscription.unsubscribe();

if (this.platform.is('cordova')) {

this.currentMapTrack.setMap(null);

}

});

}

Figure . stopRun() Method

The ‘stopRun()’ method starts by declaring a new object called ‘newRoute’ which is built around the combination of the finish time and the array ‘trackedRoute’ as set in the method ‘startRun()’. Two for loops are then executed so that the average speed and the total distance can be calculated for the run. A new constant called ‘model’ is then declared which formatted by the helper method ‘RouteIn’ and values are assigned to each of the expected parameters that is expected when saving a new route to the database. This data is then inserted into the database and the variable ‘isTracking’ is set to *‘false’*  so that that the ‘End Run’ button changes back to the ‘Start Run’ button.

## Previous Runs

<ion-card \*ngFor="let workout of workouts">

<ion-card-header>{{ workout.calories }}</ion-card-header>

<ion-card-content>

<p>Activity: {{ workout.activityType }}</p>

<p>Duration: {{ workout.duration / 100 }} min</p>

<p>Date: {{ workout.startDate | date:'short' }}</p>

<p>Distance: {{ workout.miles }} miles</p>

</ion-card-content>

</ion-card>

<ion-card \*ngFor="let activity of activities">

<ion-card-header>{{ activity.path.length }} Waypoints</ion-card-header>

<ion-card-content>

<p>Start Time: {{ activity.startTime | date: 'dd/MM/yyyy, HH:mm:ss' }}</p>

<p>End Time: {{ activity.endTime | date: 'dd/MM/yyyy, HH:mm:ss' }}</p>

<p>Speed: {{ activity.speed }} m/s</p>

<p>Distance: {{ activity.dist }} meters</p>

</ion-card-content>

Figure . Previous Runs html

The html for Previous Runs contains two different card items. The first card item will display the details retrieved for any workouts stored via Apple Health and the second card item will be displayed for all of the runs that were stored using the local storage database for the application. A new card item is created for each workout/run that is retrieved from either apple health or the database.

### Apple HealthKit Workouts

constructor(public navCtrl: NavController, private healthKit: HealthKit, private plt: Platform, private dbServ: DatabaseService) {

this.plt.ready().then(() => {

if (this.plt.is('cordova')) {

this.healthKit.available().then(available => {

if (available) {

// Request all permissions up front if you like to

var options: HealthKitOptions = {

readTypes: ['HKQuantityTypeIdentifierStepCount', 'HKWorkoutTypeIdentifier', 'HKQuantityTypeIdentifierActiveEnergyBurned', 'HKQuantityTypeIdentifierdistanceWalkingRunning'],

writeTypes: ['HKWorkoutTypeIdentifier', 'HKQuantityTypeIdentifierActiveEnergyBurned', 'HKQuantityTypeIdentifierdistanceWalkingRunning']

}

this.healthKit.requestAuthorization(options).then(\_ => {

this.loadWorkouts();

})

}

});

} else {

this.loadWorkouts();

}

})

}

Figure . Check Apple HealthKit is available

The code shown in figure 5.30 is ran inside the constructor. It first checks on the status of platform environment, following this it then checks if ‘cordova’ is available. This will only be available if the application is being ran on a mobile device and not as a progressive web application on a pc. Should Cordova be available it will then check to see if the user has allowed the application access to Apple HealthKit. Apple HealthKit will only be available for on iOS devices. Should HealthKit be available on the user’s device, it will then request read and write access various Apple HealthKit variables. Upon this request being authorised the method ‘loadWorkouts()’ is called. Should the user not have access to Cordova then the ‘loadWorkouts()’ method will still be called due to the fact dummy data is outputted if this is not available.

//Load workouts from apple HealthKit

loadWorkouts() {

var stepOptions = {

startDate: moment().startOf('day'),

endDate: moment().endOf('day'),

sampleType: 'HKQuantityTypeIdentifierStepCount',

unit: 'count'

}

if (this.plt.is('cordova')) {

this.healthKit.querySampleType(stepOptions).then(data => {

let stepSum = data.reduce((a, b) => a + b.quantity, 0);

this.stepcount = stepSum;

}, err => {

console.log('No steps: ', err);

});

this.healthKit.findWorkouts().then(data => {

this.workouts = data;

debugger;

}, err => {

console.log('no workouts: ', err);

// Sometimes the result comes in here, not sure why

this.workouts = err;

});

} else {

this.stepcount = 15000;

this.workouts = [];

}

}

Figure . Retrieve Workouts from Apple HealthKit

The ‘loadWorkouts()’ method first creates a new variable called ‘stepOptions’ which sets the parameters that apple HealthKit requires in order to return the step count for the iOS device. It then checks if Cordova is available, if it is available it then queries the HealthKit data using the step options set earlier to get the step count data. Then using ‘let’ variable a new parameter ‘stepSum’ which is of unique scope which is assigned the returned data which is reduced down to a single value. The variable ‘stepcount’ is then set to equal the value stored for ‘stepSum’.

The Apple HealthKit is then queried again this time asking for all workouts it contains for the current date, the returned data is then assigned to the variable ‘workouts’. Should the step count or workouts not be available for the iOS device, possibly because permission to Health data has not been given then a console message shall be displayed. Should Cordova not be available when checked at the start of this method then the stepcount variable will be set to equal ‘15000’ and the variable ‘workouts’ will be set as an empty array (this is simply dummy data for when using the application as a progressive web app). (Morony, 2018)

### Load Runs from database

loadRuns() {

this.dbServ.routes.then((routes: RouteOut[]) => {

routes = isEmpty(routes) ? [] : routes;

this.activities = routes.map(m => {

return {

startTime: m.startTime,

endTime: m.endTime,

speed: m.speed,

dist: m.dist,

path: JSON.parse(m.path)

}

});

});

}

Figure . loadRuns() Method

# Chapter 6: Testing and Evaluation

## Quality Assurance/ Testing

Quality Assurance (QA) or Testing is the final stage of the Modified Waterfall Software Development Life Cycle (SDLC) before the application created as part of this project can be released to the wider public for download on the relevant app stores. As part of this project testing was carried out four different testing techniques. The first technique used was during the actual development stage of the SDLC via the use of Development Testing. The second technique used was upon the completion of the development stage of the SDLC via Manual Testing. The third testing technique used was exploratory Testing and finally, the testing technique that was carried out for this project was User Acceptance Testing (UAT).

### What is Testing?

Testing can be used to fulfil several purposes such as:

* Confidence Building
* Finding Defects
* Ensuring requirements are met
* A risk management exercise
* Measuring and improving quality
* Preventing defects

However, this is not a definition of what testing is; the International Software Testing Quality Board define test as:

“The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.”

(International Software Testing Quality Board, 2019)

The primary purpose of testing is to force failures (negative testing; i.e. break the software) however, the absence of defects is not a consolation should the software not meet its stated requirements (positive testing; i.e. requirement testing). It is therefore important to carry out both positive and negative testing to ensure that the software is working as expected. Software that does not work correctly can often lead to many problems, including the loss of money, time or even the businesses reputation, and in the worst cases even injury or death.

The two main processes involved in Software Testing are Verification and Validation. These processes are used to check whether a system meets the specification and that it fulfils its intended purpose.

#### Verification

Under the ISO 9000 standard ‘Verification’ can be defined as:

“Confirmation by examination and through the provision of objective evidence that specified requirements have been fulfilled”

(International Software Testing Quality Board, 2019)

Or as suggested by in 1979 by Barry Boehm by asking the question ‘Are we building the product right?’ (i.e. Are we building what is specified?). The main objective of verification is to ensure that a product is built according to the stated requirements and design specifications.

Throughout the implementation of this project, verification was frequently carried out. The projects’ Gantt chart was reviewed frequently to ensure that the projects timeline was being adhered too. The developer also consulted the projects requirements and designs to ensure that they were being followed and met sufficiently after every element was created.

#### Validation

Under the ISO 9000 standard ‘Validation’ can be defined as:

“Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled”

(International Software Testing Quality Board, 2019)

Or as suggested by in 1979 by Barry Boehm by asking the question ‘Are we building the right product?’ (i.e. Is that which is specified really what is required?). The main objective of validation is to ensure that the finished product meets the needs of the intended user’s and that initial specifications were correct.

Validation was incorporated throughout the development stage as seen in the project Gantt chart for this project, through the form of development testing. Black Box (manual) testing was also carried out on both iOS and Android platform before the project’s application was released to UAT for the stakeholders to test.

#### White Box Testing (Development)

Development Testing is a testing technique that is used by developers during the development phase of a project to check if the code they have just written is working as expected. The purpose of Development Testing is not to replace the quality assurance (QA) process but rather to augment it. The aim of development testing is to eliminate construction/build errors before code is sent to the QA stage of the SDLC. By using this strategy, it is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

For this project development testing was carried out after the completion of each new function/method was completed in development stage of the project. The advantage of doing this type of testing is that bugs/issues can be found relatively quickly since any bugs/issues found should be the direct result of an issue/mistake in the code that had just been completed.

#### Black Box Testing (Manual)

Manual testing is one simplest and most commonly employed software testing techniques used today. Manual testing involves a tester manually executing a test case without the use of automation tools. Manual testing often helps to find bugs in software that would not normally be found by automation software due to the tester having a unique view on what is a bug. Manual testing will often require more effort to carry out, but it is important that is done without any manual testing you are unable to tell if automation is possible. One of the key advantages of manual testing is that it does require any additional tools to be carried out.

In the software testing industry one of the key fundamentals that is believed throughout the industry is the statement “100% automation is not possible”, this makes manual testing imperative to the testing stage of any software project. (Guru99, 2019)

For this project as part of the manual testing process, the following steps where followed:

1. Read and understand the software project documentation/guides. Also, study the Application prototype designs.
2. Draft Test cases that cover all the requirements mentioned in the documentation.
3. Review the test cases.
4. Execute the test cases on the application in test
5. Report bugs.
6. Once bugs are fixed, again execute the failing test cases to verify they pass.

To allow for manual testing to occur on both Android and iOS the application created as part of this project was published to an internal test track via a google developer account so that it could be tested on an Android device and it was also published to TestFlight so that it could be tested on an iOS device.

Table 7 below contains a list of all the test cases executed as part of this project. The actual test cases can be found in Appendix B.

|  |  |
| --- | --- |
| **Title** | **Result** |
| ST-001 – Android App Installation & Uninstall | PASS |
| ST-002 – iOS App Installation & Uninstall | PASS |
| ST-003 – Add Personal Details | PASS |
| ST-004 – Add Meal | PASS |
| ST-005 – View Current Day (Food Diary) | PASS |
| ST-006 – View Previous Day (Food Diary) | PASS |
| ST-007 – Add Run | PASS |
| ST-008 – View Previous Run | PASS |

Table Manual Test Cases Executed

#### Exploratory Testing

Exploratory testing is the easiest testing technique that can be implemented for any type of project. Exploratory testing simply involves the tester taking the test system and playing around with it to see what works and what doesn’t work. Exploratory testing does not involve any fixed test structure. The tester thought that exploratory testing was a useful technique for this project due exploratory testing often find the more esoteric bugs and issues that are present in an application which would not normally be found by following structured test cases.

#### Software Defects

Software defects are often the result of a human being making an error (mistake) in a programs code or in the documentation, which in turn will produce a defect. In a programs code, if a defect is present in a programs code this may result in the system failing to do what it is expected to do. It is important to note though, that whilst defects in software, systems or documents may result in failures, but not all defects do so. Figure 6.1 below shows the common causes of software defects.

* Humans are fallible
* Working under pressure
* Working with extreme complexity

A Defect/Fault/Bug

“the result of an error”

An Error

“a mistake by a human”

A Failure

“System does not perform as expected”

causes

causes

causes

Figure 6.. Causes of Software Defects

Table 8 below shows a list of all the software defects found throughout the Development testing and the Manual Black Box testing of this project.

|  |  |  |
| --- | --- | --- |
| **Bug/Issue Number** | **Description** | **Resolution** |
| 1 | Data for a meal is not being saved to local storage | FIXED |
| 2 | ‘Set Details’ button is not disabled if no data is entered for the details page | FIXED |
| 3 | View Current Day option redirecting to wrong page | FIXED |
| 4 | Date picker is sending the wrong date format | FIXED |
| 5 | View previous day is not returning food diary data for selected date | FIXED |
| 6 | Nutritionix API call is failing to return results | FIXED |
| 7 | Results returned by Nutritionix API call need filtered | FIXED |
| 8 | Food diary only displays one meal item for breakfast, lunch dinner, snack | FIXED |
| 9 | Step count is displaying the last 24 hours and not the step count for the current date | FIXED |
| 10 | Formatting issues with results returned for Add Meal | FIXED |
| 11 | Exercise Tracker menu option is not working | FIXED |
| 12 | Workout route is not being displayed | FIXED |
| 13 | Total distance travelled is incorrect | FIXED |
| 14 | Apple Health integration is not working | FIXED |
| 15 | Previous Runs details page always shows the same data | FIXED |
| 16 | Unable to ‘End’ a workout | FIXED |

Table Software Defects found during Testing

#### User Acceptance Testing

User Acceptance Testing (UAT) is last phase of software testing in an SDLC. During UAT, the finished software is handed over to the people who would be using the software daily. These users would then test the software to make sure that it can handle the required tasks in real world scenarios, that the application was originally specified to meet. For any new application or software can be released to the market, it is critical that the UAT process is carried out and should any major issues be raised that they are addressed before being released to the market. (Usersnap, 2019)

As part of this project the following stakeholders agreed to this project’s application:

* Matt McNabb - Android
* Matthew Hobson - iOS
* Kourosh Pourgholi - Android

To allow them to test the application their account details were added to either the internal test track on google or to the list of testers who can download it via TestFlight. Found below is the feedback from their user acceptance testing.

**Matt McNabb**

“Really liking the app. It offers a smooth user experience throughout and the apps design is also consistent which is good to see. The only issue that I have with the app being that it would be nice to see the total number of calories burned during a logged activity.”

**Matthew Hobson**

“Hey Chris tested the app there and it all seems to be working fine for me after playing around with it for a while. The only thing being that it may be nice to be able to delete a meal that has been added to a day, although this not listed in the requirements, so this isn’t a major issue.”

**Kourosh Pourgholi**

“Hey dude just after installing the app there and giving it a wee trail run and everything seems to be working great. Kinda gutted though that the whole recipe idea functionality had to be dropped from the project, as I was really looking forward to see what it would suggest.”

##### Issues raised by UAT

|  |  |  |
| --- | --- | --- |
| **ISSUE** | **Resolution** | **REASON** |
| Calories burned from workout not shown | NOT FIXED | To be able to accurately calculate the number of calories burned during an activity, a person heart rate activity is required. Without this heart rate the equation of calories burned can be out by as much 200 calories which was felt to be counterproductive to the aim of this project in helping to tackle obesity. |
| No functionality for deleting a meal | NOT FIXED | This feature was purposefully not included due to the fact during observation phase of the requirements gathering process it was noticed that by allowing a user delete meals, it meant that a user was able to change the meal to something that was slightly healthier to make them feel better about what they ate that day. |
| No random meal generator functionality | NOT FIXED | This functionality and associated requirements had been changed to a ‘Won’t Do’ in the Moscow prioritisation feature due to this being feature that was included in the Nutritionix API if a $40/month subscription fee was paid. |

Table UAT Issue Resolutions

## Evaluation

The following section evaluates the technologies used to complete this project and their overall effectiveness for the required task they were used to fulfil. Overall the developer felt that all the technologies and tools used to accomplish the projects goals. The chosen technology stack helped to cut down on the time needed for common mundane development tasks and allowed for the time spent developing and testing the system to be spent effectively.

### Technologies Used

#### Visual Studio Code

As an IDE Visual Studio Code was a very effective tool which helped to massively speed up the development process. Visual Studio Codes ‘IntelliSense’ allowed for the automatic completion of code and the automatic importing of the required libraries to allow the use of certain functionality. The tools debugging feature and built-in terminal allowed you to run and debug the code inside the actual editor. Visual Studio Code is also fully integrated with ‘Git’ which allowed you to make git commands straight from the editor and not have to do this via command line or GitHub. Finally, Visual Studio Code offers a wide range of externally developed extensions that can add new languages, themes, debuggers, as well as connect to additional services. These extensions are ran in separate processes, ensuring that they didn’t slow down the editor.

#### The Ionic Framework

The use of the Ionic Framework was the developers real experience in using ionic, thankfully due to the fact the developer starting using it in work around the same time as he started the development work for this project meant that they had minimal trouble in learning how to develop in ionic correctly. The Ionic Framework is predominately based on HTML5 but also makes use of the Angular framework which is installed by default with Ionic. The Ionic Framework allowed the developer to only have to develop the projects application once for both the Android and iOS platforms. Another advantage of Ionic was that it also allows for the application to be used as Progressive Web Application although it does have limited functionality depending on the device it was opened on when used in this format. The Ionic Framework also an app called the ‘Ionic DevApp’ that can be used to test early prototypes of an applications code which speed of the “Development Testing” stages of the SDLC for this application. One drawback that was found with the Ionic Framework by the developer was that plugins that can be used to access certain features on a device are not owned and maintained by the Ionic Framework team but are instead maintained and owed by third-parties. This meant that some of the plugins offered did not work for the latest platform versions, this had negative impact on the developers time spent on this project due to usage of the ‘Health’ plugin which was only discovered to not work on a device running iOS 12 during the final stages of projects development time.

#### Local Storage

Local Storage allowed the developer to write the code and not have to worry about how data shall be saved on the various device types, due to the fact Local Storage will make the application use whatever database storage option the device is running defaults too. Another advantage of using Local Storage was that this allowed for the application to also be ran as a progressive web app which would not have been possible if it had been using SQLite.

#### Nutritionix API

The Nutritionix API has one of the largest food databases in the world that is currently available to developers for free. This meant that the developer did not have to spend a large amount of time writing a new food database that would have been severely limited when it came to the foods/meals that could be added in the application. The Nutritionix API did have one major drawback to it though in that it limited the API calls that you could make and what data was received from the ones you could make unless you paid a $40 per month subscription fee. An example of this is that it will not return any images for any API calls made unless you pay the subscription fee nor will it allow you to use its random meal generator API call.

### Modified Waterfall Life Cycle

The Modified Waterfall Software Development Lifecycle allowed for the effective management of the projects resource and the time allowed for each stage of the project’s life cycle. The Project lead felt this life cycle was the correct choice for this project due to the limited number of people who would be involved in it, meaning that it was to start and finish each aspect of the project before moving onto the next stage.

# Results

Table 10 below shows a list of all the requirements gathered for this project and if they have been fulfilled.

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Functional Requirement | MOSCOW | STATUS |
| FR-01 | The application shall work on both iOS and Android platforms | Must | Fulfilled |
| FR-02 | A new user shall be asked to set up an account | Must | Fulfilled |
| FR-03 | The app shall remember a user after first-time set up | Must | Fulfilled |
| FR-04 | The user shall be able to navigate the app via a Navigation menu pane | Must | Fulfilled |
| FR-05 | The app shall allow a user to be able to search for meals. | Should | Fulfilled |
| FR-06 | A user shall be able to log a meal for breakfast | Must | Fulfilled |
| FR-07 | A user shall be able to log a meal for lunch | Must | Fulfilled |
| FR-08 | A user shall be able to log a meal for dinner | Must | Fulfilled |
| FR-09 | A user shall be able to log a snack | Should | Fulfilled |
| FR-10 | A user shall be able to view all the food they have logged in the last 7 days | Should | Fulfilled |
| FR-11 | A user shall be able to view a calorie break down for all the food they have eaten for the current day. | Must | Fulfilled |
| FR-12 | The app shall allow the user to start a run | Must | Fulfilled |
| FR-13 | The app shall allow the user to end a run | Must | Fulfilled |
| FR-14 | The app shall record the time ran by a user | Must | Fulfilled |
| FR-15 | The app shall record the distance covered during a run | Must | Fulfilled |
| FR-16 | The app shall record the calories burned during a run | Should | NOT MET |
| FR-17 | Upon a run ending the app shall show the route a user ran on a map | Should | Fulfilled |
| FR-18 | Upon a run ending the app shall show the distance a user ran | Should | Fulfilled |
| FR-19 | Upon a run ending the app shall show the calories burned by a user | Should | NOT MET |
| FR-20 | Upon a run ending the app shall show the average pace of the user | Should | Fulfilled |
| FR-21 | The app shall show statistics on all the runs a user has completed | Should | Fulfilled |
| FR-22 | The app shall display the step count for a day | Should | Fulfilled |
| FR-23 | The app shall show the average distance walked/ran since the user started using the app | Won’t | N/A |
| FR-24 | The app shall ask the user to select a gender upon initial set up | Must | Fulfilled |
| FR-25 | The app shall ask the user to enter their weight upon initial set up | Must | Fulfilled |
| FR-26 | The app shall be able to randomly suggest a meal for breakfast | Won’t | NOT MET |
| FR-27 | The app shall be able to randomly suggest a meal for lunch | Won’t | NOT MET |
| FR-28 | The app shall be able to randomly suggest a meal for dinner | Won’t | NOT MET |
| FR-29 | The app shall display the recipe upon selecting a meal | Won’t | NOT MET |
| FR-30 | The app shall provide a nutritional breakdown of any meals/snacks it returns | Could | Fulfilled |
| FR-31 | The app shall allow the user to compare their information against other app users | Won’t | N/A |
| FR-32 | The app shall provide the user with a monthly summary of their workouts | Won’t | N/A |
| FR-33 | The app shall allow the user to view information on their weekly diet in a graph | Won’t | N/A |
| FR-34 | The app shall be integrated with the Apple Health application | Could | Fulfilled |
| FR-35 | The app shall be able to save data to the Apple Health application | Could | Fulfilled |
| FR-36 | The app shall be able to retrieve data from the Apple Health application | Could | Fulfilled |
| FR-37 | The app shall be integrated with the Google Fit API | Could | Fulfilled |
| FR-38 | The app shall be able to save data to the Google Fit API | Could | Fulfilled |
| FR-39 | The app shall be able to retrieve data from the Google Fit API | Could | Fulfilled |
| NFR-01 | The app shall have a consistent theme throughout the application | Must | Fulfilled |
| NFR-02 | The app shall be simple to understand and use | Must | Fulfilled |
| NFR-03 | The app shall work on devices of different screen sizes | Must | Fulfilled |
| NFR-04 | The app shall not crash | Must | Fulfilled |
| NFR-05 | The app shall be capable of running in the background | Must | Fulfilled |
| NFR-06 | The app shall react appropriately to changes orientation | Must | Fulfilled |
| NFR-07 | The app shall be easy to navigate | Must | Fulfilled |
| NFR-08 | The app shall conform to best practice and design principles for the chosen platforms | Must | Fulfilled |

Table Requirements Fulfilment Status

## Requirements Not Met

**FR-16 & FR-19**

To be able to accurately calculate the number of calories burned during an activity, a person heart rate activity is required. Without this heart rate the equation of calories burned can be out by as much 200 calories which was felt to be counterproductive to the aim of this project in helping to tackle obesity.

**FR-26, FR-27, FR-28, FR-29**

Whilst these requirements had originally been given a MOSCOW prioritisation of ‘Could’ this was later changed to ‘Won’t Do’ due to this service being part of the premium package of the Nutritionix API which requires a $40/month subscription fee. Due to these costs being outside the budget available for this project, these requirements were changed to ‘Won’t Do’ and no further action was taken.

## Additional Work Completed

During the Development Stage there was some scope creep that occurred, thankfully for the developer this did not have a serious impact on the projects timescale due to the fact additional slippage time had been left available at the end of the project which could be used as additional development time for the project. The projects Gantt chart can be found in appendix A. The additional requirements that were added to the project can be found below in ‘Table 11’ (for their MOSCOW prioritisation rating please see section ‘3.4 Requirements Gathered During Development’). These requirements where added after some investigation into seeing if they could be added to the project and not have a negative impact on the project’s overall timescale.

|  |  |
| --- | --- |
| Number | Functional Requirement |
| FR-34 | The app shall be integrated with the Apple Health application |
| FR-35 | The app shall be able to save data to the Apple Health application |
| FR-36 | The app shall be able to retrieve data from the Apple Health application |

Table Additional Work Completed

# Conclusion

## Project Reflection

### Project Scope

The overall aim of this project was to develop Food and Fitness Tracker mobile health application. The hope of the project was to create an application which would help to promote healthier lifestyle choices as well as if necessary helping the end user to tackle obesity/weight issues. The introduction of integrating the projects application with both Apple health and Google Fit provides the end user with a greater user experience and increases the user value of the application.

The initial research looked at the examined how obesity effects the people of the UK and how obesity can be best tackled. This research showed that the creation of a mobile health application that would allowing a user to track the nutritional value of the foods that they are eating and the activities that they do in a day. By creating app that allows a user to quickly search for and adds food to their diary means that the user does not have to waste a large amount time trying to find out the information for themselves, which it made more likely for them to give up on living a healthier lifestyle.

It is the belief the Project Lead that one of the most important parts of this project was the research into the various Software Development Life Cycle and the choosing of the methodology that best suited this project. The reason why this was so important to the project is due to the methodology determining how the project will be managed throughout its life cycle. The Project Lead feels that the choice of the Modified Waterfall methodology was the correct choice for this project, as it allowed for a clear plan to be put into place which allowed for easy tracking the projects schedule. The use of the Modified Waterfall methodology also prevented the working on multiple aspects of project at the same time. This helped to maintain morale whilst working on the project as project tasks had a clear completion date and it was easy to see that project progress was being made. If the project was to be done again the developers feels that it may have been worthwhile involving some project stakeholders and playing planning poker to estimate the development time for development tasks as the developer found that some of the estimated times were simply not achievable (i.e. the ‘food log page’ task as seen in the projects Gantt chart in Appendix A).

For the requirement gathering process the Project Lead felt that the process could be improved upon if it was to be carried out again. The Project Lead felt that whilst including potential end users in a focus group to gather their opinions on what they would like to be included in the final application was vital to the projects overall success, they believe that should have been a greater emphasis on researching possible requirements to see if they were visible as if this was carried out in greater depth the issue needing to pay a subscription fee so that access was allowed random recipe/meal generator API call would have been raised and therefore development time would not have been wasted trying to implement this functionality. If not for the availability and use of the time allocated for the project slippage buffer in the project plan then this project would not have been completed in time.

A suggestion for future students who are thinking of undertaking a project of similar size and scale would be to research and decide on an SDLC in the early stages of the project and to create a realistic project plan based on their chosen methodology and stick to it. As tasks can quickly build up when the plan is not followed.

### Application Produced

The developer is proud of the application that they successfully made for this project using the Ionic Framework something that they had no prior experience in before starting this project. The developer feels that the Ionic framework was well chosen to accomplish this task but did feel that the fact that the plugins are not being owned and maintained by the Ionic Framework team was negative drawback and that it would be interesting to do the project again using Xamarin Forms just to see what the differences are. Other than this single issue, the developer is pleased with the application that has been made with accomplish all the Requirements that had been marked as ‘Must Do’ as well accomplishing the majority of the ‘Should Do’ and ‘Could Do’ requirements. The Developer feels that creating UI Designs before any of the development work has commenced was also useful endeavour as it meant they did not have to waste trying to work out what a page should look like but instead just followed the laid-out designs.

The Developer suggests that any future students undertaking a similar project they should take it as an opportunity to learn a new programming language and use new technologies like they have done for this project. They suggest this because by using the Ionic Framework has them new opportunities in the work place where they are moving from the software testing role to a new role which whilst still being predominately a testing role, it will now involve some development work. Also, by learning a new programming language the student will be challenging themselves and not just sticking to what they are comfortable with.

### Source Control

The Project Lead feels that GitHub was very successful method of document source control and would highly suggest that any future students also use it for their source control as it allowed work to safely stored on the web with each commit being easy to push fetch from any PC or location. This allowed work to be carried out on multiple computers. It also allows for earlier pieces of work to be downloaded should a mistake have been made and the developer not be sure when the mistake was introduced.

### Project Maintenance

Thankfully it is quite easy for the application that was created as part of this project to be maintained. As should any changes be made to the project then all that is required is for the developer to publish the changes using the Google and Apple developer accounts and then end-user will get a notification informing them that the app needs to be updated by either Google Play or the Apple AppStore.

# References

176/SC, T., 2005. *ISO 9000:2005, Quality management systems -- Fundamentals and vocabulary.* ed. (): International Organization for Standardization.

Anon., . *ISO 9000 - Quality management.* [Online]   
Available at: http://www.iso.org/iso/home/standards/management-standards/iso\_9000.htm  
[Accessed 5 4 2019].

Baker, C., 2019. *Obesity Statistics,* s.l.: House of Commons Library.

Beaumont, D., 2017. *Adobe XD: Getting Started.* [Online]   
Available at: https://www.youtube.com/playlist?list=PLHjwuoik-ep1Vb4RPXNUbAVHxB1rJRC8W  
[Accessed 22 December 2018].

Boschert, S., 2014. App use increased physical activity. *Clinical Neurology News*, 19 August.

Chandana, 2018. *Scrum Project Management – Pros and Cons.* [Online]   
Available at: https://www.simplilearn.com/scrum-project-management-article  
[Accessed 03 November 2018].

Citta, 2017. *Waterfall Model.* [Online]   
Available at: https://www.cittasolutions.com/waterfall-model  
[Accessed 01 November 2018].

dariosalvi78, 2019. *dariosalvi78/cordova-plugin-health: Cordova plugin for the HealthKit and Google Fit frameworks.* [Online]   
Available at: https://github.com/dariosalvi78/cordova-plugin-health  
[Accessed 23 January 2019].

Google, 2019. *Firebase Guides.* [Online]   
Available at: https://firebase.google.com/docs/guides/  
[Accessed 31 March 2019].

Grimm, S., 2019. *Simon Grimm - YouTube.* [Online]   
Available at: https://www.youtube.com/channel/UCZZPgUIorPao48a1tBYSDgg  
[Accessed 1 Febuary 2019].

Guru99, 2019. *Manual Testing Tutorial for Beginners: Concepts, Types, Tool.* [Online]   
Available at: https://www.guru99.com/manual-testing.html  
[Accessed 06 April 2019].

Haller, A., 2018. *Ionic Tutorial: Building a complete mobile app with Ionic 3.* [Online]   
Available at: https://ionicthemes.com/tutorials/about/building-a-complete-mobile-app-with-ionic-3  
[Accessed 30 December 2019].

Haughey, D., 2014. *REQUIREMENTS GATHERING 101.* [Online]   
Available at: https://www.projectsmart.co.uk/requirements-gathering.php  
[Accessed 14 November 2018].

Intelegain Technologies, 2017. *Scrum Software Development Dallas | Scrum Project Management.* [Online]   
Available at: https://www.intelegain.com/scrum/  
[Accessed 03 November 2018].

International Software Testing Quality Board, 2019. *ISTQB Glossary.* [Online]   
Available at: http://glossary.istqb.org/search/testing  
[Accessed 05 April 2019].

Ionic, 2019. *Android Play Store - Ionic Documentation.* [Online]   
Available at: https://ionicframework.com/docs/publishing/play-store  
[Accessed 23 March 2019].

Ionic, 2019. *Intro.* [Online]   
Available at: https://ionicframework.com/docs/intro  
[Accessed 25 January 2019].

Ionic, 2019. *Ionic Native.* [Online]   
Available at: https://ionicframework.com/docs/v3/native/  
[Accessed 21 January 2019].

Ionic, 2019. *Ionic UI Components and API Customization.* [Online]   
Available at: https://ionicframework.com/docs/v3/components/#overview  
[Accessed 04 January 2019].

Ionic, 2019. *Publishing to app store - Ionic Documentation.* [Online]   
Available at: https://ionicframework.com/docs/publishing/app-store  
[Accessed 23 March 2019].

Ionic, 2019. *Publishing your app - Ionic Framework.* [Online]   
Available at: http://ionicn.com/docs/guide/publishing.html  
[Accessed 23 March 2019].

Joshua H West, L. M. B. R. A. C. F. P. C. H. B. T. C., 2017. Controlling Your “App”etite: How Diet and Nutrition-Related Mobile Apps Lead to Behavior Change. *JMIR Mhealth Uhealth.*

Margam, G., 2018. *Model-View-Controller (MVC).* [Online]   
Available at: https://medium.com/datadriveninvestor/model-view-controller-mvc-75bcb0103d66  
[Accessed 04 April 2019].

Marino, S., 2019. *Benefits of Tracking Your Food Intake.* [Online]   
Available at: https://www.canr.msu.edu/news/benefits\_of\_tracking\_your\_food\_intake  
[Accessed 01 April 2019].

McGivery, A., 2014. *Structure of an Ionic App.* [Online]   
Available at: http://mcgivery.com/structure-of-an-ionic-app/  
[Accessed 04 April 2019].

Montemagno, J., 2018. *Xamarin.Essentials: Cross-Platform APIs for Mobile Apps.* [Online]   
Available at: https://blog.xamarin.com/xamarin-essentials-cross-platform-apis-mobile-apps/  
[Accessed 03 October 2018].

Mountain Goat Software, 2018. *Scrum Methodology and Project Management.* [Online]   
Available at: http://www.mountaingoatsoftware.com/agile/scrum  
[Accessed 02 November 2018].

Nations, D., 2019. *What Is IOS? (iPhone Operating System).* [Online]   
Available at: https://www.lifewire.com/what-is-ios-1994355  
[Accessed 02 04 2019].

NHS, 2016. *Obesity - NHS.* [Online]   
Available at: https://www.nhs.uk/conditions/obesity/  
[Accessed 23 January 2019].

Nutritionix, 2016. *Nutritionix - Nutrition Database API Documentation.* [Online]   
Available at: https://developer.nutritionix.com/docs/v1\_1  
[Accessed 16 January 2019].

PAL, S. K., 2018. *Software Engineering | Spiral Model.* [Online]   
Available at: https://www.geeksforgeeks.org/software-engineering-spiral-model/  
[Accessed 03 November 2018].

Powell-Morse, A., 2016. *Waterfall Model: What Is It and When Should You Use It?.* [Online]   
Available at: https://airbrake.io/blog/sdlc/waterfall-model  
[Accessed 01 November 2018].

RapidApi, 2019. *The RapidAPI Developer Hub.* [Online]   
Available at: https://docs.rapidapi.com/docs/basics-creating-a-project  
[Accessed 16 January 2019].

Rose, M., 2018. *What is mHealth?.* [Online]   
Available at: https://searchhealthit.techtarget.com/definition/mHealth  
[Accessed 02 April 2019].

Sanchez, A., 2018. *Ionic is composed: The MVC (Model-View-Controller or Model-View-Controller).* [Online]   
Available at: https://medium.com/@asanchez\_77574/ionic-is-composed-the-mvc-model-view-controller-or-model-view-controller-e47566a1dd2e  
[Accessed 04 April 2019].

StatCounter, 2018. *Mobile Operating System Market Share Worldwide.* [Online]   
Available at: http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201711-201811-bar  
[Accessed 27 November 2018].

Tong HL, C. E. T. W. W. Y. Q. J. M. P. L. L., 2019. *Efficacy of a Mobile Social Networking Intervention in Promoting Physical Activity: Quasi-Experimental Study,* Sydney: JMIR Publications.

TryQA.com, 2013. *What is Waterfall model- Examples, advantages, disadvantages & when to use it?.* [Online]   
Available at: http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/  
[Accessed 30 October 2018].

Tutorialspoint, 2018. *SDLC Overview.* [Online]   
Available at: https://www.tutorialspoint.com/sdlc/sdlc\_overview.htm  
[Accessed 01 November 2018].

University of Washington, 2018. *Shneiderman's "Eight Golden Rules of Interface Design".* [Online]   
Available at: https://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html  
[Accessed 05 December 2019].

Wisdomjobs.com, 2018. *SDLC OVERVIEW SOFTWARE DEVELOPMENT LIFECYCLE (SDLC).* [Online]   
Available at: https://www.wisdomjobs.com/e-university/software-development-lifecycle-sdlc-tutorial-1513/sdlc-overview-25976.html  
[Accessed 01 November 2018].

# Appendix A

## Supporting Documentation

|  |  |
| --- | --- |
| Prototyping Design Document |  |
| Project Gantt Chart |  |

# Appendix B

|  |  |  |
| --- | --- | --- |
| **ST-001 – Android App Installation & Uninstall** | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | On an Android device navigate to the following URL: https://play.google.com/apps/internaltest/4701669015653843594 | Test install page is displayed |
| **2** | Join the test program | Test program joined successfully |
| **3** | Install the app | App installs successfully |
| **4** | Uninstall the application | Application is successfully uninstalled |

Table ST-001 – Android App Installation & Uninstall

|  |  |  |
| --- | --- | --- |
| **ST-002 – iOS App Installation & Uninstall** | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | On an iOS device  open Apple TestFlight | TestFlight opened and the FFT application is available for download |
| **2** | Install the application | Application installed  successfully |
| **3** | Uninstall the application | Application is successfully installed |

Table ST-002 – iOS App Installation & Uninstall

|  |  |  |
| --- | --- | --- |
| **ST-003 – Add Personal Details** | | |
| This test should be carried out on both an iOS and Android device | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Download and install the FFT application | App installed |
| **2** | Open the application | App opens and the "Details" page is displayed |
| **3** | Click on the 'Set Details' button | Button is disabled and validation showing that the fields are required is shown |
| **4** | Click on the Gender field | The option 'M' and 'F' are displayed |
| **5** | Select a gender | Gender option successfully selected  'Set Details' button is still disabled |
| **6** | Click on the Weight field and enter a weight value | Weight value is successfully entered  'Set Details' button is still disabled |
| **7** | Click on the Age field and enter a weight value | Age value is successfully entered  'Set Details' button is enabled |
| **8** | Click on the 'Set Details' button | The user redirected to the 'Diet Home' page |

Table ST-003 – Add Personal Details

|  |  |  |
| --- | --- | --- |
| **ST-004 – Add Meal** | | |
| This test should be carried out on both an iOS and Android device | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Open the application | App opens and the "Diet Home" page is displayed |
| **2** | Click on the 'Add Meal' option | Add Meal' page is displayed |
| **3** | Click on the 'Search' button | No items are returned |
| **4** | Enter a food item e.g. 'chicken pie' | Results for 'chicken pie' are returned |
| **5** | Click on the 'Lunch' button | Meal is saved as lunch for that date and user is returned to the 'Diet Home' page |
| **6** | Repeat steps 2-5 a total of 3 times ensuring that a different meal option selected each time (i.e. Breakfast, dinner, snack) | Age value is successfully entered  'Set Details' button is enabled |
| **7** | Click on the 'Set Details' button | The user redirected to the 'Diet Home' page |

Table ST-004 – Add Meal

|  |  |  |
| --- | --- | --- |
| **ST-005 – View Current Day (Food Diary)** | | |
| This test should be carried out on both an iOS and Android device | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Open the application | App opens and the "Diet Home" page is displayed |
| **2** | If required run test ST-004 so that meals have been added for the current day | STEP ONLY |
| **3** | Click on the 'View Current Day' option | The Food Log page is displayed |
| **4** | Check that the returned data is correct | The returned data is correct |

Table ST-005 – View Current Day (Food Diary)

|  |  |  |
| --- | --- | --- |
| **ST-006 – View Previous Day (Food Diary)** | | |
| This test should be carried out on both an iOS and Android device Meals should be added for a date in the past | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Open the application | App opens and the "Diet Home" page is displayed |
| **2** | Click on the 'View Previous Day' option | The 'Select Date' page is displayed |
| **3** | Select a date that does not have any meals logged for it and press the 'Continue' button | The Food Log page is displayed; however, no food data is displayed |
| **4** | Click on the 'Back' button located in the header | The 'Select Date' page is displayed |
| **5** | Select a date that has meals logged for it and press 'Continue' | The food log page is displayed showing the meals logged for the selected date |
| **6** | Verify the data is correct | The data is correct |

Table ST-006 – View Previous Day (Food Diary)

|  |  |  |
| --- | --- | --- |
| **ST-007 – Add Run** | | |
| This test should be carried out on both an iOS and Android device Meals should be added for a date in the past | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Open the application | App opens and the "Diet Home" page is displayed |
| **2** | Press the Hamburger icon | The Main Menu pane is displayed |
| **3** | Close the menu pane by swiping to the left | Menu pane closes |
| **4** | Press the Hamburger icon | STEP ONLY |
| **5** | Press on the 'Exercise tracker' menu option | The 'Exercise Home' page is displayed |
| **6** | Press on the 'Add Run' option | Add Run page is displayed, and the map shows the devices current location |
| **7** | Press the Start Run button | New Run is started |
| **8** | Go for a walk with the test device, keeping an eye on the application | The map tracks the user’s location showing their route |
| **9** | Press the 'End Run' button | Run is ended and the device displays a summary of the activity |

Table ST-007 – Add Run

|  |  |  |
| --- | --- | --- |
| **ST-008 – View Previous Run** | | |
| This test should be carried out on both an iOS and Android device Meals should be added for a date in the past | | |
| **STEP** | **Action** | **Expected Result** |
| **1** | Open the application | App opens and the "Diet Home" page is displayed |
| **2** | Press the Hamburger icon | The Main Menu pane is displayed |
| **3** | Press on the 'Exercise tracker' menu option | The 'Exercise Home' page is displayed |
| **4** | Press on the 'Previous Runs' option | Previous Runs page is displayed and displays a list of all the previous runs and their details |
| **5** | Press on a run | Map is displayed showing the route that was ran during the run |

Table ST-008 – View Previous Run

# Appendix C

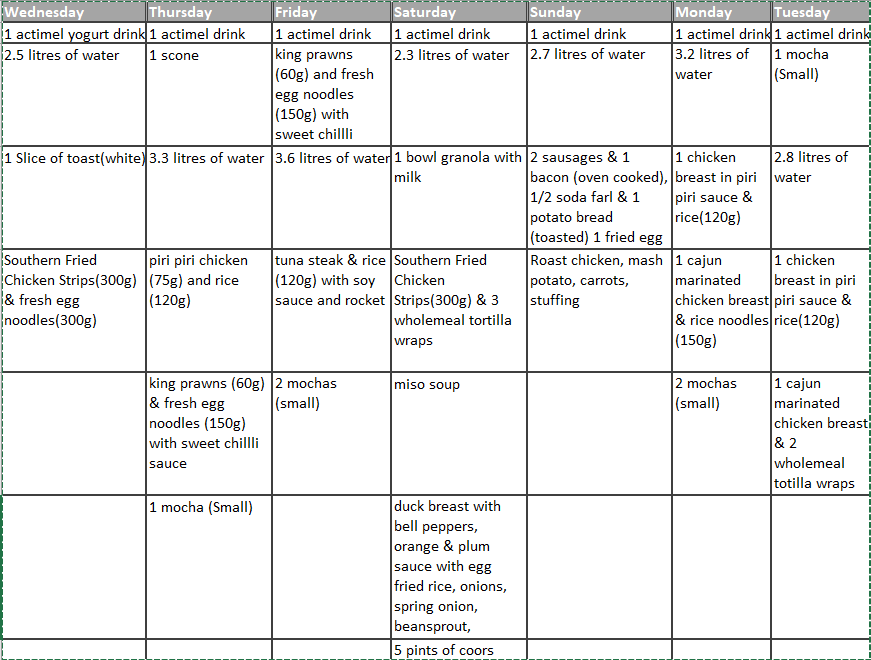


Figure 12.. Original method for recording Meals

1. The application prototype as created in Adobe XD can be found in appendix A. [↑](#footnote-ref-1)