

Faculty of Technology, Design and Environment
Oxford Brookes University
School of Engineering Computing and Mathematics

BSc (Single Honours) Degree Project

Programme Name: COMP6013: BSc Computing Project

Module No. COMP6013

Surname: Blake

First Name: Daniel

Project Title: Nutrition and training platform with trainer - client interaction as focus

Student No: 16051879

Supervisor: Arantza Aldea

Date submitted: 17/04/2021

A report submitted as part of the requirements for the degree of BSc (Hons) in Computer Science

At

Oxford Brookes University

Student Conduct Regulations:

Please ensure you are familiar with the regulations in relation to Academic Integrity. The University takes this issue very seriously and students have been expelled or had their degrees withheld for cheating in assessment. It is important that students having difficulties with their work should seek help from their tutors rather than be tempted to use unfair means to gain marks. Students should not risk losing their degree and undermining all the work they have done towards it. You are expected to have familiarised yourself with these regulations.

<https://www.brookes.ac.uk/regulations/current/appeals-complaints-and-conduct/c1-1/>

Guidance on the correct use of references can be found on www.brookes.ac.uk/services/library, and also in a handout in the Library.

The full regulations may be accessed on-line at <https://www.brookes.ac.uk/students/sirt/student-conduct/>

If you do not understand what any of these terms mean, you should ask your Project Supervisor to clarify them for you.

I declare that I have read and understood Regulations C1.1.4 of the Regulations governing Academic Misconduct, and that the work I submit is fully in accordance with them.



Signature

Date 17/04/2021

REGULATIONS GOVERNING THE DEPOSIT AND USE OF OXFORD BROOKES UNIVERSITY MODULAR PROGRAMME PROJECTS AND DISSERTATIONS

Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Library.

I agree that this dissertation may be available for reading and photocopying in accordance with the Regulations governing use of the Library.



Signature

Date 17/04/2021

NUTRITION AND TRAINING PLATFORM WITH TRAINER - CLIENT INTERACTION AS FOCUS

**BSc project
submitted to**

**School of Engineering, Computing and
Mathematics**

OXFORD BROOKES UNIVERSITY

**Supervisor
Arantza Aldea
Daniel Blake
APRIL 2021**

Table of contents

Keywords.....	6
Abstract.....	6
1. Introduction.....	7
1.1 Background.....	7
1.2 Rationale.....	7
1.3 Aims.....	7
1.4 Objectives.....	7
1.5 Deliverable Overviews	8
2. Literature Review.....	9
3. Methodology.....	13
3.1 Introduction	13
3.2 Overview	13
3.3 Basis for Agile Approach	13
3.3.1 Reduced Technical Debt	13
3.3.2 Better Product Control and Reduced Risk	13
3.3.3 Agile Development for Mobile Applications.....	14
3.5 Requirements.....	14
3.5.1 Functional Requirements.....	15
3.5.2 Non-Functional Requirements.....	16
3.5.3 Optional Requirements.....	16
3.6 Use Case Diagram	17
4. Design	18
4.1 GUI Design.....	18
4.2 Activity Flow Diagram	21
4.3 Database Design.....	21
5. Implementation.....	24
5.1 Object Orientated programing	24
5.2 Android Studio	24
5.3 Use of an Emulator	25
5.4 Software Development.....	25
5.5 MySQL vs SQLite	27
5.6 Navigation	29

5.7 Sprint 1 Summary.....	30
5.8 XML	30
5.9 Workouts.....	31
5.10 Session Handler.....	33
5.11 Nutrition.....	33
5.12 Log Calendar	34
5.13 User Testing	35
6. Results	35
6.1 Technical Achievement	35
6.2 Evaluation and Discussion	36
6.2.1 Key Questions	36
6.2.2 Question 4.....	36
6.2.3 Question 5.....	37
6.2.4 Question 6.....	38
6.2.5 Question 7.....	39
6.3 Constraints and Limitations	40
7. Professional Issues	41
7.1 Project Management	41
7.2 Risks Assessment	42
7.3 BCS Code of Conduct and Practices	43
7.4 Legislation	44
7.4.1 Data Protection Act.....	44
7.4.2 Android	44
7.5 Ethical Issues	44
7.5.1 Offensive Content	44
7.5.2 Verifying Trainers.....	45
7.6 Data Management	45
8. Conclusion	45
8.1 Future development	45
8.2 Reflection	46
9. Bibliography	47
10. Repositories and Links	49

Keywords

PTs	- Personal Trainers
Clients	- Clients specifically of Personal Trainers
OOP	- Object Oriented Programming
GUI	- Graphical User Interface
UI	- User Interface

Abstract

An application whose purpose is to facilitate a closer interaction between personal trainers and their clients. This is to be achieved by allowing personal trainers direct access to their clients' fitness related data for example calorie intake, macronutrient goals and workout regimes.

1. Introduction

1.1 Background

There are many calorie and nutrition tracking apps available in today's app market. Notable big names include: 'My Fitness Pal' and 'Yazio'. Both of which, focus heavily on individuals tracking their nutrition. Investigations into the effectiveness of these apps have since shown that, users with a lower sense of self-efficacy for fitness and healthy eating result in greater goal achievement rates after using these apps (Bracken, ML. and Waite, BM. 2020). Coincidentally, the same is often true for clients of personal trainers (here after referred to as PTs). Although many of these apps allow users to independently log their training. Few provide a space for freelance PTs to collaborate with these apps.

1.2 Rationale

After my time working in a gym, I noticed that many PTs were interacting with their clients through nonspecific messaging services and producing training plans in Microsoft Excel. A method that could surely be improved upon. Likewise, if individuals are searching for Personal trainers they may search through word of mouth, local gyms, or websites such as: 'TopLocalTrainer'. With this platform I hope to effectively encapsulate the key aspects of each process mentioned above, combining them into one app.

1.3 Aims

I would create an App that allows PTs to directly interact with client nutrition tracking, create and adapt workout plans, as well as connect and advertise their services to those searching for trainers.

1.4 Objectives

Complete a review of existing services for each user requirement (Nutrition tracking, creating a Training Plan, Personal trainer advertising), identifying the pros and cons of each.

Periodically Interviewing three consenting PTs as part of an Agile development strategy to discover user requirements.

Create an app that delivers the following primary requirements:

- Training log and planning.
 - Where PTs can create training regimes (by a selection of possible/created exercises within the app) and directly upload to

users account with any personalised adjustments, comments, or notes.

- Nutrition tracking with access granted to PTs.
 - Users can track their calorie intake (subcategorized into macro nutrients) by either inputting nutritional data, searching specific food items from a database, or scanning an item's barcode.
 - PTs can view client calorie intake, comment, suggest, and communicate with clients regarding nutrition.
- Advertising for verified community reputable PTs.
 - Build a database of verified PTs.
 - Allow PT profiles to provide links to their social media.

1.5 Deliverable Overview

Product vision	In today's current climate of social distancing, this app aims to consolidate all other aspects of fitness and personal training into one place (excluding necessary face to face interactions and training), through an app that hopes to deliver on the already stated primary requirements. (see. 1.1 Objectives)
Approach	As I will be collaborating closely with PTs as means for testing and identifying new user story's I intend to make use of an Agile deployment approach to delivering an artifact. This will allow for greater flexibility and better iterative testing.
Key deliverables	<p>PTs must be able to comment and suggest appropriate nutrition for their clients (for example, meal suggestion)</p> <p>PTs must be able to communicate directly through the app.</p> <p>Users must be able to track their nutrition by scanning bar codes/ looking food up in a database or manually inputting nutritional information.</p> <p>Users must be able to see the profiles of registered PTs.</p>

2. Literature Review

Currently, there many applications that allow users to track training and nutrition. Many of which do not boast intricate design, with some regarded as no more than a notebook rather than offering any support through the world of physical education. Those designs may be enough for an individual with an experienced training level but offers no guidance to those without. However Before looking into the science of either field, it is important to ask the question; **How does one assess the quality of mobile applications designed for training and nutrition?** Wiemeyer introduced a modular interdisciplinary framework for the evaluation of such mobile apps. In which he specifically targets people with a low level of training experience. There are four main modules: Training, Behavioural change techniques, Technology, Evaluation of effects (Wiemeyer, 2019). When designing this app, specific points within the framework of these modules stand out for example: iterative training control, goals, instructions, feedback, feasibility 'in the wild', risk of bias etc. An obvious conclusion to be made would be that considering this framework may improve the quality of creating and evaluating mobile applications of this type.

In a similar light, María Villasana conducted a systematic review of mobile applications for the promotion of healthy nutrition and physical activity habits. In the analysis of 73 different apps, 19 were referenced in scientific studies, however the apps were only every referred to and lack supplementary validation. The review also discovered and categorised the applications, separating and comparing features. This leads into the question; what are the most common requirements of successful apps in this market? The review discoverers the distribution of features between the mobile applications. Concluding that the most essential features were some of the following: Weight/height, age, gender, goals, diet dairy, food database etc. (Villasana 2019). From this list we can infer that anyone with prior use of these apps may develop an expectation of what is required, and therefore it is important to cover most of the features mentioned when developing a new if deemed necessary. In addition, the popularity and usage of such apps have been recorded by their respective marketplace. In a study showcased by Villasana's work, it shows that the app 'Calorie counter- My Fitness Pal' vastly outperforms the next most popular app in respect to usage and user rating as seen below.

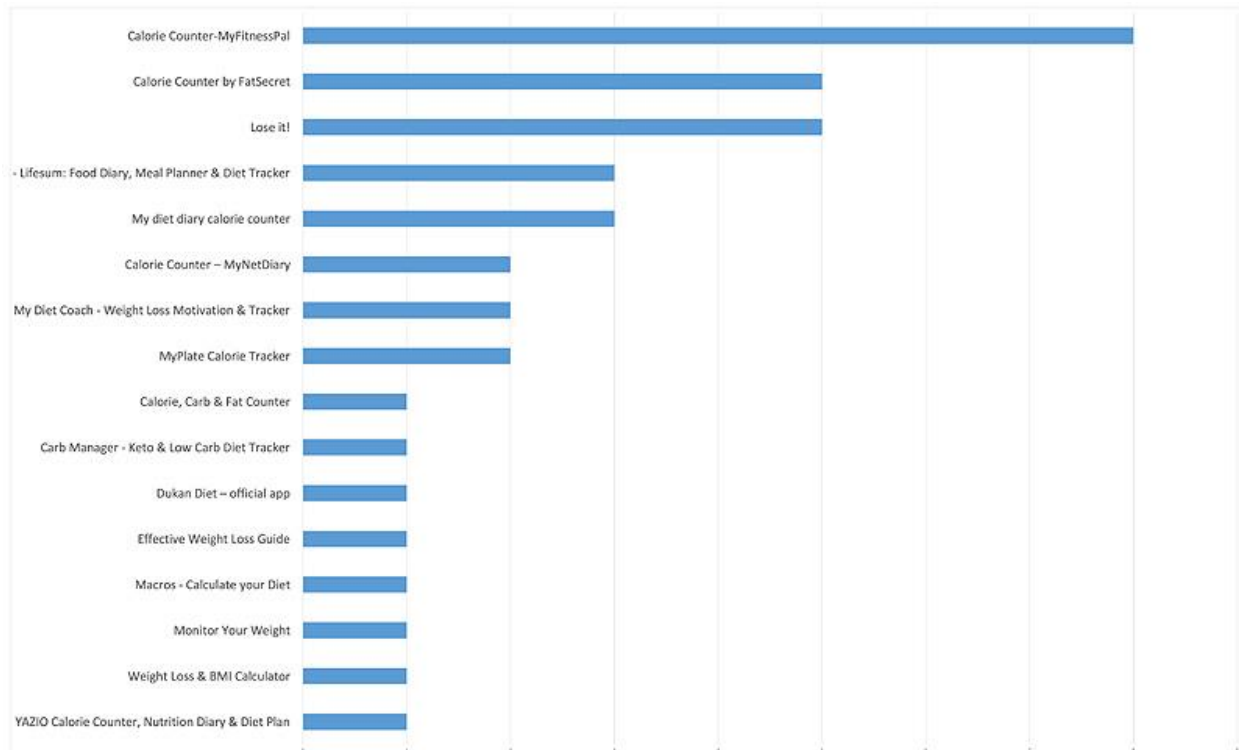


Figure 1 showing fitness app ranking (Villasana, 2019)

So, what makes MyFitnessPal the most popular calorie tracking app on the market at this moment? Evans explored the functionality of this app and identified key aspects that may set it apart from others. Aspects that have subsequently promoted its clinical use by elite athletes in sports. Evans goes on to list the good features of this app as being:

Easy to search values for reference, Favourites list allow quick inputting, Extensive food database, Detailed charts and graphs, Default or personalised daily goals based on weight and aims, Integrated calorie functions to further personalise diet. (Evans, 2017)

In contrast, he then goes on to list some of the negative features of the app including: Values inputted by other users give scope for incorrect data, need to scan, weigh, or measure every new or varied meal, Risk of obsession, Search function can be arduous. (Evans, 2017)

Although it may not be possible to avoid every negative feature, it would be wise to learn from the app's successful ones. A list of written pros and cons is extremely useful when developing an app in the same field. Also, the use of 'MyfitnessPal' in clinical practice and sports science is a testament to the category of health and fitness apps to which it belongs.

This testament was further developed when Bracken and Waite formed an investigation into whether the use of nutrition tracking app 'MyFitnessPal' had any effect on a user's ability to achieve their goals. The journal in which results are presented asks first whether; it is possible for users to improve their ability to achieve their goals through such apps, and more importantly second; it discusses the possible reasons as to why this may be case. The results show that participants that recorded an initial lower sense of self-efficacy for fitness and healthy eating, reported a significantly elevated ability to achieve their goals. In contrast, participants that recorded an initially higher sense of self-efficacy for fitness and healthy eating presented insignificant change to their ability to achieve goals (Bracken and Waite, 2020). However, Bracken and Waite also disclaimed that testing was limited and that it could be further validated by an increase in diversity within test subjects.

It is important to consider this sort of Psychological effect when creating a health and fitness app and this concept has appeared previously in the literature review when Wiemeyer mentioned the relevance of 'Behavioural change techniques' (Wiemeyer, 2019). In Ansel's study of 'personal training', he explains the fundamentals of personal training, however chapters 2,4 and 12 delve into Programming essentials, motivational psychology, essential nutrition, and personal training as a business. In places going into detail about important options regarding the creation of a fitness programme. As well as invaluable psychological strategies to motivate clients that could be in part transferred to a digital platform (Ansel, 2008). Simple things such as rewarding user's who have completed sets of exercises with gratifying ticks, or displaying graphs depicting user progress thus encouraging self-affirmation. The use of detailed graphs and charts was also a key member of Evan's pros when evaluating the market leading, MyFitnessPal (Evans, 2017).

The next topic of importance follows on from Bracken and Waites studies into the effectiveness of behavioural change and self-belief. The topic being, what behaviours/ideas do we want to promote as a health and fitness app? In the Journal of Clinical Psychology in Medical Settings, they examined the relationship between two dietary mindsets and their effect in the instance of a dietary lapse. The two behavioural approaches were: Prevention, in which the participant was encouraged to think that they shouldn't eat 'bad foods' that would otherwise hinder their diet. The other was Promotion, in which the participant was encouraged to eat healthy foods that would comply with their diets. The investigation found that the participants that held a 'prevention focus' were more likely to adhere to the constraints of their daily caloric intake. It was hypothesised that this could be because they are stricter with their caloric consumption. However, although the a 'prevention focus' yielded participants who held a stricter diet. In the event of a dietary lapse, the 'prevention focus' group were shown to consume a greater number of calories in violation of their diet (Testa and Brown, 2015). To conclude, a 'prevention focus' approach may yield a stricter dietary ethic, however it had also

proved detrimental for dietary maintenance following a dietary lapse, whereas the 'promotion focus' encouraged dietary longevity. After reviewing this investigation, I can infer that a combination of both focuses in moderation may produce the most successful results. A method in which I could encourage a 'prevention' mindset using the app would be – the number of calories turning red if in excess. However, because I want to prioritise positive encouragement, a better solution may be – the number of calories turning green if within 5% above or below their daily caloric goal. In contrast, a method in which I could encourage a 'promotion' mindset may be – allowing trainers to suggest specific healthy meals or an implementation that allows the app to make healthy suggestions based on self-identified options you may have previously had.

We know that self-monitoring strategies such as weighing oneself, planning meals, tracking calories and exercising are key behavioural factors of successful fitness goals thanks to Kruger, Blanck and Gillespie's Dietary and Physical Activity Behaviors (Kruger, 2006). However, how can we encourage the habitual use of these factors through a fitness app? One method of motivating these habits is by generating a positive emotional response to completing them. Petrescu's 'Viral marketing and social networks' examines the success of marketing strategies. Petrescu concludes that the appeal of a certain product is often increased if it can incite a positive emotional response (Petrescu, 2014). With this in mind, connotating key behavioural factors of successful fitness goals to satisfying digital eye candy e.g. gratifying ticks, thumbs up, green text, stars etc. would incite a positive correlation to the use of these features, providing motivation until a longer-term habit is formed.

3. Methodology

3.1 Introduction

In this section, I will discuss the use of 'Agile Development' over the course of this project's technical progress. As such, I will identify the contributing factors as to why an 'Agile' approach would be superior to other relevant methodologies. I will then continue by listing key requirements, design, and specification strategies.

3.2 Overview

The Agile Software Development methodology focuses on incrementing the completion of a product through a number of time constrained iterations called 'Sprints'. The function of a sprint is to deliver a predefined number of features based on a selected portion of the project's functional requirements/user stories. After each sprint, an analysis of the product is conducted by the various stake holders and the overall requirements/development strategies used in the project may be altered accordingly in preparation for the next sprint. This incremental approach allows for greater flexibility overall during development hence the name, 'Agile'.

3.3 Basis for Use of Agile Development

3.3.1 Reduced Technical Debt:

Technical debt makes reference to the maintenance tasks created as a consequence of attempting to prioritize client value and or time constraints over technical design considerations. These tasks may include additional testing, debugging and refactoring code. An 'Agile' approach helps to promote minimal technical debt, as any maintenance tasks that arrive from a sprint are added to an overall 'backlog' of tasks. This provides an opportunity during the planning stage of the next sprint to focus on addressing said tasks, unlike the 'waterfall' method which uses a more holistic approach to coding. I will be developing this project using Java in the IDE Android Studio, however as I do not have an abundant extent of experience with either, it is essential that the project does not accumulate an excess of technical debt.

3.3.2 Better Product Control and Reduced Risk

Unlike a 'Waterfall' approach, in which a larger list product features are implemented in the same phase, the 'agile' method works in tandem with stake holders during product iterations, focusing on producing key deliverables at the end of each sprint. In this way, it is not only easier to identify any technical debt, but also to document the exact functionalities of each sprint's artifacts and or quickly adapt to any change in requirements formed after a review of that

sprint. During 2020's Covid-19 pandemic climate, an initial list of all user requirements was difficult to compile due to fragmented instances of social distancing regulations and self-isolation. Because of this, an agile approach would be better as it reduces the risk of overrunning deadlines as one can begin coding earlier whilst embracing any adaptations to user stories in later sprints and limiting the negative effects on time constraints.

3.3.3 Agile Development for Mobile Applications

When developing mobile applications, emphasis is often given to the user's interaction and satisfaction with the project rather than only meeting an initial requirement. This is because in the crowded market of mobile applications, the success of a product is heavily reliant on its popularity. For this reason, it is important to work alongside user stakeholders in order to develop a cycle implementation, testing, and feedback. Most mobile applications also evolve over subsequent years following their initial public release in the form of updates. An infrastructure practiced in agile development strategy is also better suited for this post release product evolution.

3.4 Requirements

Before any technical development had begun, a list of initial requirements was compiled. The requirements in this list were ascertained by using two methods:

1. An analysis of popular fitness applications and other existing systems (such as 'MyFitnessPal' and 'Yasio') using both primary and secondary research.
2. Interviews conducted both online and in person, with primary stakeholders (Personal Trainers, Clients) regarding the efficacy of existing systems and fundamental user requirements.

The requirements have been separated into two classes, clients and PTs. Then further categorised into different areas of functionality (Nutrition, Exercise, Trainer contact). From this, a list of a list of Functional, Non-functional, and optional requirements have been resolved as follows:

3.4.1 Functional Requirements

Clients

Nutritional

1. Clients can create and search food items.
2. Clients can add said food items to a daily list of food items that compares their intake of calories with their goal intake of calories.
3. Client calories and total macronutrients must be calculated and displayed.
4. Clients can view their food diary Logs, that keep a history of food items they have had on specific days.
5. Clients can set and adapt their intake goals at any time.
 - a. Clients must be able to input their current weight, goal weight, Calorie goals, age and gender.
6. Clients delete and update food items in their daily log, as well make comments as in their dairy.

Workouts

7. Clients can create, update, and delete Workouts and exercises.
8. Clients can view a list of their workouts.
9. Clients can log to their diary that they have done a specific workout.
10. Clients can make notes under their log of a workout that can be viewed later.

Trainer Contact

11. Clients can view a PT profile and make a request into their services.
12. Clients must be able to revoke access from a trainer to their account at any time.

Personal Trainers

Nutritional

13. PTs must be able to change the calorie intake goals of the clients to whom they are connected.
14. PTs must be able to view a client's log of food items and make any comments or suggestions below.

Workouts

15. PTs must be able to create, update, and delete Workouts and exercises.
16. PTs must be able to send created workouts to their clients of their choosing.
17. PTs must be able to adapt and update Client workouts to better support their clients.

18. PTs must be able to view and comment below a client's workout log and or respond to the client's comments.

Trainer Contact

19. Trainers must be able to accept or reject client requests.
20. Trainers must be able to see the profile of a client who has requested their services.
21. PTs must be able to link to multiple client accounts to virtually monitor their training.

Other

22. Users must not be able to delete or edit objects that were not created by them.
23. Users must be able to log onto and create accounts unique to the user.

3.4.2 Non-Functional Requirements

1. The system must be a mobile application.
2. The application should be compatible on most smart phones.
3. The System should be quick and easy to navigate.

3.4.3 Optional Requirements

1. Food database can be pre-populated with food items from known public databases.
2. A Client can use their camera to scan their food's barcode for quick selection, rather than manually searching for it in the food database.
3. The System GUI must provide positive reinforcement by providing textual/ visual affirmations of hard work or achievement.
4. *Clients and PTs must be able to enter their location, and GUI map display of local PTs must be shown.
5. PT's must be able to upload media to their profiles including images or videos.
6. Clients must be able to rate PT's and PT ratings must be openly displayed.

3.5 Use Case Diagram

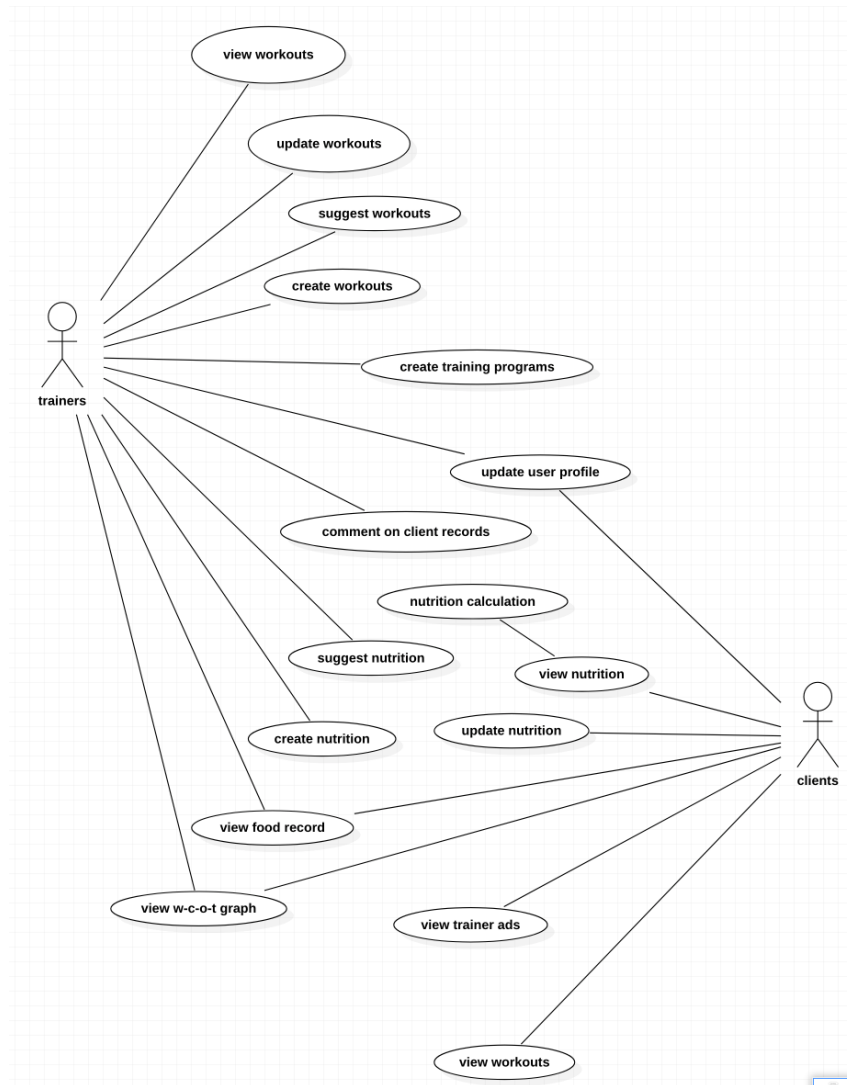


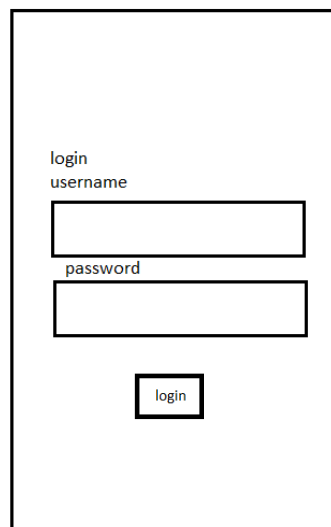
Figure 2 Above describes the use cases of each user in the system.

4. Design

4.1 GUI Design

Intuitive GUI can be selling point of the simplest mobile applications. Whereas bad GUI design is often the crux of user discontentment. Consequently, it may be assumed that only impressive GUI design is ubiquitous throughout the most popular mobile applications. However, these apps are frequently produced by larger development teams in contrary to a single developer. Thus, in the interest of time management, it was important that this application feature an effective but simple GUI. As the appeal of its appearance is not one of the functional requirements as previously mentioned (see 3.4.1 Functional Requirements).

To ensure that users are separated by type, and only have access to their unique instance of the system; The first page a user is greeted with upon opening the app is the login page, as shown in figure 3.



login
username

password

login

Figure 3 Displays the App tile design of the Login page

Once the user has inputted the correct details, they will subsequently be faced with an initial home screen. As of yet, this home screen serves little purpose other than acting as an anchor for navigation and all other user activities. However, upon a professional release, the home screen is capable of holding information such as user-based notifications, software development update logs or general user targeted content/advertisement e.g. blog posts.

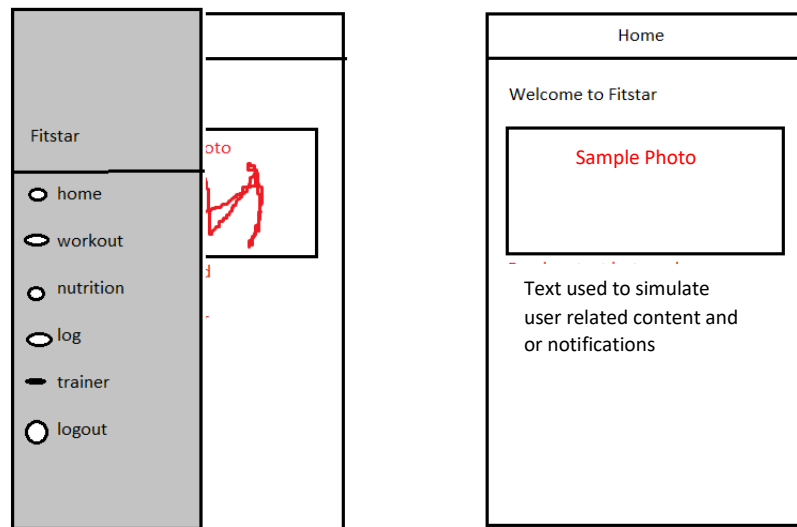


Figure 4 displays the navigation drawer and home page designs.

The drawer style navigation appears regularly in popular modern mobile applications as it accommodates an unobstructed user experience. The navigation options displayed in figure 4 are those in accordance with a 'client's' use case. Whereas the navigation options given to a 'PT' are limited to: Home, Workout, Client and Logout. Following on from this the user can pursue any available user pages for example:

Workout

My workouts

Legs

Add workout

Workout

Name

legs

comment/ description

Squat

lunes

leg press

Add exercise

exercise

Name

Calf raises

sets

weight

reps

Set 1

20

set 2

18

set 4

Add Set

NUTRITION

CALORIES

GOAL

CURRENT

REMAINING

Date:dd/mm/yy

X

Y

X-Y

Breakfast

Lunch

Dinner

Snacks

Add Food

Food

search food here

database

my foods

Potatoes

broccoli

beans

Add food

Food

name

nuts

number of servings

x

grams

serving size

100

grams

per serving size

protein

carbohydrates

fats

calories

Add food

Figure 5 A GUI design sample through various pages of the application.

4.2 Activity Flow diagram

The user activity flow diagram as shown below in figure 6 shows a map of the UI, the path in which users may take to get to different areas of the system and examples of the actions that can occur.

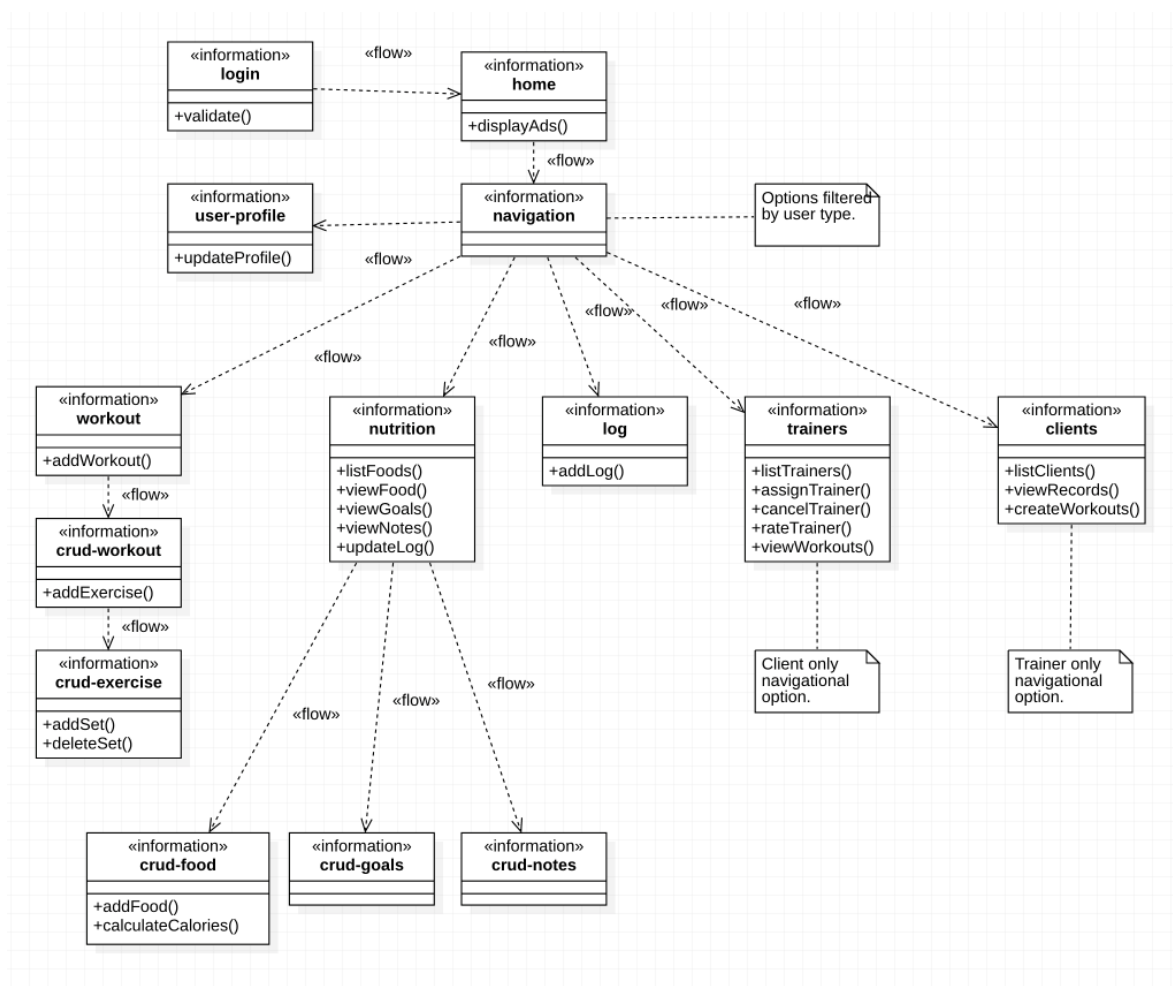


Figure 6 displays an activity flow diagram of the proposed system.

4.3 Database design

The premise of this system is heavily reliant on storing large databases populated with information relevant to the user. The underlying method used to assist this system will be a relational data model. Using a relational database reduces the build-up of unnecessary data and

allows for more efficient methods of ascertaining specific information. This can be achieved through the process of normalization.

Database normalization is the process in which we can use to split large databases into smaller ones. By doing so, systems can reduce data redundancy and traffic when querying. Data redundancy is the repetition of similar data. This not only incurs a cost on maintenance and storage but also provides an opportunity for data inconsistent between the similar data as mention. Thereby Normalization also improves data integrity by holding a reference the data held in other tables rather than storing it again. This is particularly relevant in this system as users can create a multitude of different objects, however, to repeat the entirety of user information in every entry would increase the size required to store it to that which is completely unwarranted. In this manner, inserting, updating, and deleting data are also far simpler.

Tables	Data fields
Users	First_name, Last_name, User_Type, Workouts, Workout_Names, Exercises, Exercise_Names, Sets, Set_Conditions, Set_Values, Food, FoodName, Date_Created, Calories, Protein,Carbohydrates, Fats, Clients, Trainers

Table 1 displays a sample of raw data fields the system must manipulate.

Table 1 displays the sample of raw data fields that the system needs to collect. The process of database normalization will be used to structure this unnormalized form in accordance with a series of normalization requirements called 'normal forms'. These 'normal forms' are used to specify different levels of database normalization. They are as follows:

1st Normal Form (1NF) – The elimination of repeating data groups.

2nd Normal Form (2NF) – The creation of separate tables for sets that apply to multiple records.

The data must also already be in 1NF and all non primary key attributes must be uniquely identified by, and dependant on a primary key.

3rd Normal Form (3NF) – The elimination of any remaining data fields that do not depend on the key. The data must also already be in both 1NF and 2NF.

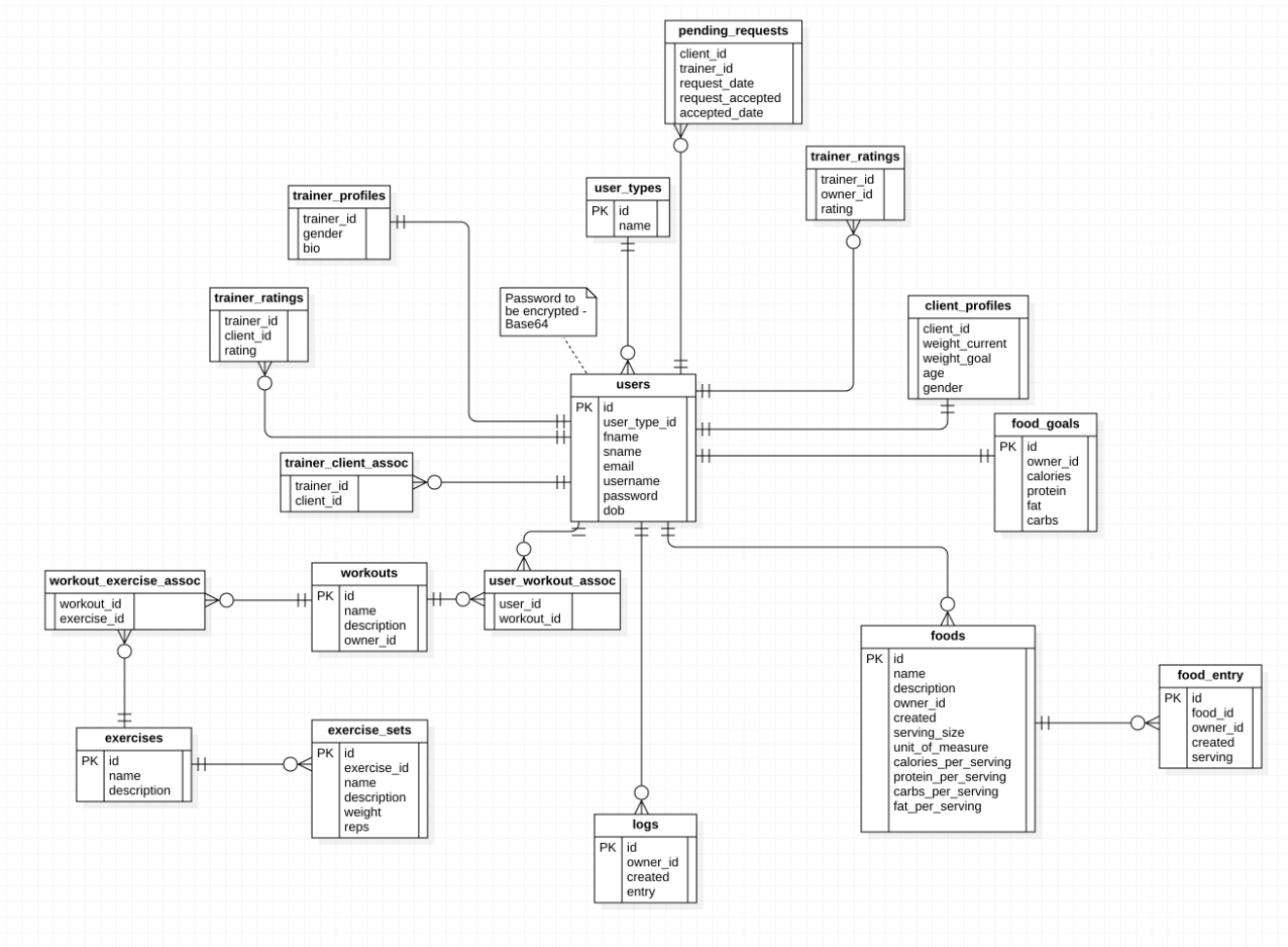


figure 7 displays the database model diagram showing the relational database in 3rd normal form.

In Figure 7, 1NF was first achieved by separating data sets such as workouts, foods, exercises etc. The reason for this was that, if this form were not reached, the database would have repeated User information e.g first_name, last_name, after every new object that belonged to that user had been created. Which may have intern left some data fields empty creating data anomalies. 2NF and 3NF were then achieved by identifying each data entry by a primary key and then eliminating any data fields that were not reliant on that key.

5 Implementation

Having outlined the design constraints and compiled the requirements of the proposed system in 3.4 Requirements. Some key implementation discussions and methods will be discussed as follows.

5.1 Object Orientated Programming

Object oriented programming (OOP) is a programming system based around the use of 'Objects'. An object may contain data fields known as attributes and procedures called as methods. Example attributes include name, age, gender, and example methods may be 'get age' or 'calculate age from date of birth'. This is important because the incorporation of many instances of the same class i.e., an object can occur many times throughout the proposed fitness system.

The use of OOP also allows the system to employ other strong features related to OOP for example Abstraction, Encapsulation, Inheritance and Polymorphism. These practices aid the implementation phase by organising existing functions and reducing the amount of repeated code. For example, coding implementation began by forming a basic navigation system, by inheriting properties and methods from a parent class 'Fragments', further development could take place without the need to repeat methods in every child instance of the fragments class.

5.2 Android Studio

Android Studio is the official IDE for Google's Android operating system. At present, more than 76.6% of Smartphones including popular brands such as Samsung, Sony, and HTC models use Android as their operating system (Rajput M, 2020). Besides Smartphones, using Android studio also provides a streamlined path towards compatibility with android wearable smart tech like smart watches that are frequently used to record physical data that may be used in fitness applications. Android studio has recently become one of the main IDEs for android application development for the reasons mentioned above. It also offers an opportunity to standardise features with the android's custom widget features leading to systems inheriting aspects of Androids GUI style. It was for these reasons that Android studio was the IDE of choice for this system rather than other popular IDEs such as Eclipse.

5.3 Use of an Emulator

Android Studio offers a default emulator, however the disk space required to run it resulted in multiple system crashes. Therefore downgrading its specification was necessary before development began. The System was built using a Nexus S running Android 8.0 Oreo / API 26. This is important and further explained in 6.3 Constraints and Limitations

5.4 Software development

As an available language inside of Android Studio, initial development began using Java and XML files alongside MySQL, which is an open-source relational database management system written in C and C++. Android Studio offers its users a range of templates to avoid coding fundamental tasks speeding up early development however given the circumstances; this project began from an empty activity class.

During the early phases of development, it became evident that this system had the possibility of containing a multitude of classes. Therefore, it was important, as it is in all systems to keep an organised structure from the start. This was achieved by creating place holder classes and ordering them into separated packages with other similar classes as seen below in figure 8:

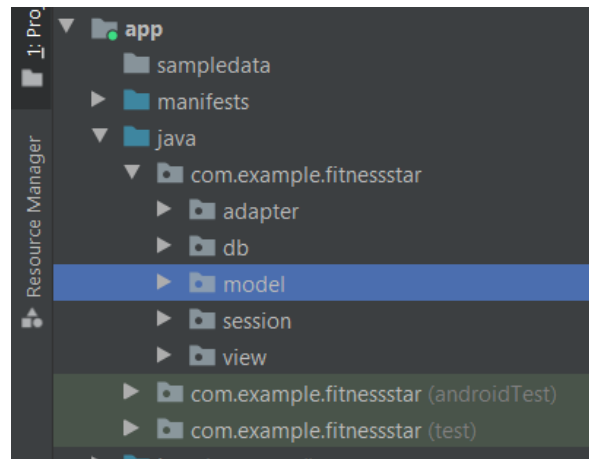


Figure 8 displays package organisation.

The initial programming task in the first agile sprint centred around separating the Clients sub system from the PT's sub system by use of a login and validate function. This function would determine first the type of user then return the id of user to be called in later methods. From this function, the system defined the appropriate navigation options shown.

To achieve this, required the implementation of two Database tables in MySQL, User_Types and Users. The tables would have to be seeded with preliminary data however currently the only important fields were UserName, Email, and password.

From doing this, it quickly became obvious that storing data in its raw form offered no security against data theft. Thereby offering no such protocol in alignment with the data protection act. Therefore, in order to mitigate data theft, the password was encrypted using 'base64' encryption. See figure 9 below:

```
public User validateLogin(String username, String password) {  
    /*  
     * Ensure there are valid values.  
     */  
    if ((username == null) || (password == null)) {  
        return null;  
    }  
    /*  
     * Base64 Encode password.  
     */  
    String encodedPassword = Base64.encodeToString(password.getBytes(), Base64.DEFAULT);  
    /*  
     * Validate login details with the backend.  
     */  
    return db.validateUser(username, encodedPassword);  
}
```

Figure 9 shows the method used to encrypt passwords on login validation.

Unfortunately, during this early testing phase, the system suffered multiple inconsistencies in the reliability of MySQL. The emulator used to run the code encountered persistent connection issues to MySQL and often crashed as a result. As a temporary solution for testing purposes, example user values were hardcoded into the system. However, this did not follow the previously design database structure and therefore had to be addressed.

5.5 MySql vs SQLite

MySql was chosen for this project as it is a suitable database management service often used to deploy applications and services specifically for a cloud environment. However due to this early setback and considering the scope of this system, it decided that SQLite would be sufficient to simulate the core functionalities of this design. SQLite is also a C language library that implements a small and fast SQL database engine. SQLite is also built into all mobile phones and most computers. However, the important change is that by using SQLite, the SQL databases are built and then stored to a local drive rather than uploaded and accessed from the cloud.

This means that the database is currently confined to one system, which is obviously in contrast to its functional requirements and would not be the case with a full-scale system release.

```
DATABASE_NAME( value: "fitness_star.db"),

USER_TYPES_TABLE_NAME( value: "user_types"),
USER_TYPES_TABLE_SQL_CREATE( value: "create table if not exists user_types ( " +
    "id integer primary key autoincrement, " +
    "name text not null " +
    // "primary key (id) " +
    ");"),
USER_TYPES_TABLE_SQL_DROP( value: "drop table if exists " + USER_TYPES_TABLE_NAME.getValue()),

USERS_TABLE_NAME( value: "users"),
USERS_TABLE_SQL_CREATE( value: "create table if not exists users ( " +
    "id integer primary key autoincrement, " +
    "user_type_id integer, " +
    "fname text not null, " +
    "sname text not null, " +
    "email text not null, " +
    "username text not null, " +
    "password text not null, " +
    "dob date not null, " +
    // "primary key (id), " +
    "foreign key (user_type_id) references user_types (id) " +
    "on delete cascade " +
    "on update no action " +
    ");"),
USERS_TABLE_SQL_DROP( value: "drop table if exists " + USERS_TABLE_NAME.getValue()),
```

Figure 10 displays the database constants used to initialise the database name and column names.

```
db.execSQL(USER_TYPES_TABLE_SQL_CREATE.getValue());
db.execSQL(USERS_TABLE_SQL_CREATE.getValue());
```

Figure 11 displays the functions used to create the tables if they do not already exist.

```
if (DatabaseUtils.queryNumEntries(db, USERS_TABLE_NAME.getValue()) < 2) {
    db.execSQL("delete from " + USERS_TABLE_NAME.getValue() + ";");
    db.execSQL("insert into " + USERS_TABLE_NAME.getValue() + " (id, user_type_id, fname, sname,
email, username, password, dob) values (1, 1, 'John', 'Strongman', 'john@strongman.com',
'strongjohn', 'cGFzc3dvcmQ=', '1990-01-01');");
    db.execSQL("insert into " + USERS_TABLE_NAME.getValue() + " (id, user_type_id, fname, sname,
email, username, password, dob) values (2, 2, 'James', 'Winsor', 'james@winsor.com', 'jamesw',
'cGFzc3dvcmQ=', '1990-01-01');");
}
```

Figure 12 displays the function used to seed example User data used for testing.

Figures 10, 11, 12 above show how the SQL Databases are implemented using android Studio and SQLite. The Databases are only created if they do not already exist. They also feature auto incrementation of their primary keys to ensure that two data entries may not occupy the same primary key.

```
public User validateUser(String username, String password) {
    SQLiteDatabase db = this.getReadableDatabase();
    String sql = "select * from " + USERS_TABLE_NAME.getValue() + " where ((lower(username) = '" + username.trim().toLowerCase() + "' or (lower(email) = '" + username.trim().toLowerCase() + "' and lower(password) = '" + password.trim().toLowerCase() + "'))";
    Cursor results = db.rawQuery(sql, null);
    Log.d(LOG_TAG, sql);
    Log.d(LOG_TAG, "Results Count: " + results.getCount());

    User user = null;
    if ((results != null) && (results.getCount() == 1)) {
        results.moveToFirst();

        user = new User();
        user.setId(results.getInt(results.getColumnIndex("id")));
        user.setUserTypeId(results.getInt(results.getColumnIndex("user_type_id")));
        user.setFname(results.getString(results.getColumnIndex("fname")));
        user.setSname(results.getString(results.getColumnIndex("sname")));
        user.setDob(results.getString(results.getColumnIndex("dob")));
    }
    return user;
}
```

Figure 13 shows how the user found in the database and returned to the system.

Figure 13 above shows the method which users are found in the SQL database. SQL databases may be queried and or manipulated using several known commands. Here an entry is obtained using the 'SELECT' command and 'WHERE' command to specify the location and search parameters. This is important because these Commands will be reused throughout many operations in this system.

5.6 Navigation

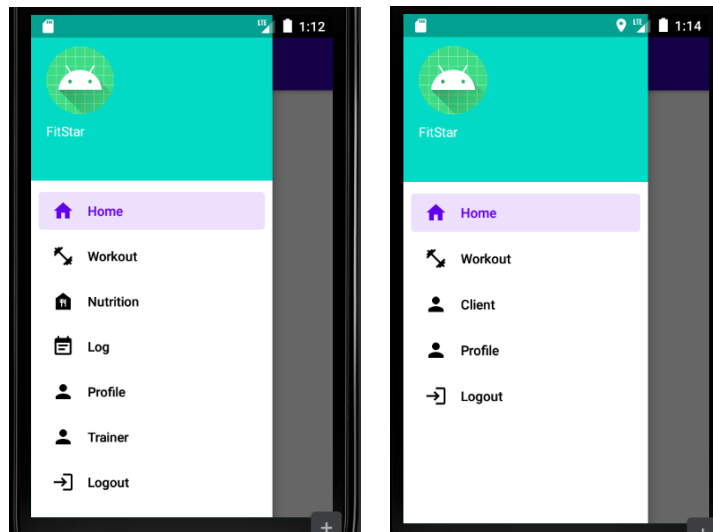


Figure 14 displays the navigation after a client logs in (left) vs after a PT logs in (right)

Once that was completed, progress into the navigation stage could be made in accordance with the system design. The design required two separate navigation paths. This was achieved by accessing the User object SQLite had gathered from the database and comparing it's User_type value.

1 = Trainer

2 = Client

These values were declared in the enum 'SessionConstants.java'. An Enumerated Type (enum for short) is a type that has a fixed set of constant values. E.g. enum Size {small, medium, large}.

```
NavigationView navigationView = findViewById(R.id.nav_view);

Menu menu = navigationView.getMenu();
if(user.getUserTypeId() == Integer.parseInt(USER_TYPE_ID_CLIENT.getValue())){
    menu.findItem(R.id.nav_client).setVisible(false);
    menu.findItem(R.id.nav_trainers).setVisible(true);
    menu.findItem(R.id.nav_trainerProf).setVisible(false);
}else {
    menu.findItem(R.id.nav_client).setVisible(true);
    menu.findItem(R.id.nav_trainers).setVisible(false);
    menu.findItem(R.id.nav_nutrition).setVisible(false);
    menu.findItem(R.id.nav_log).setVisible(false);
    menu.findItem(R.id.nav_trainers).setVisible(false);
    menu.findItem(R.id.nav_profile).setVisible(false);
    menu.findItem(R.id.nav_trainerProf).setVisible(true);
}
```

Figure 15 displays the way in which different navigations are displayed.

We achieved figure 14 by assigning visibility values to different menu items shown in figure 15.

5.7 Sprint 1 Summary

As this initial sprint faced a larger issue of databases other planned features such as workout use case stories were saved for the second sprint.

5.8 XML

Extensible Markup Language is a language that defines a set of rules for encoding documents in a format that is both readable by humans and machines. In this way it was used to represent all of the visual features of this app as well as construct formulaic layouts for each Fragment app screen. Android studio offers a range of visual development tools to much akin to 'Visual Basic'. It also offers a range of different layout designs which define the way objects are anchor in a scene. Androids most recent addition to these options would be the 'Constraint layout' as seen in Figure 16. The Constraint layout allows you to position and size other view objects in a flexible way. This is used to control the way users of different size devices view the application.

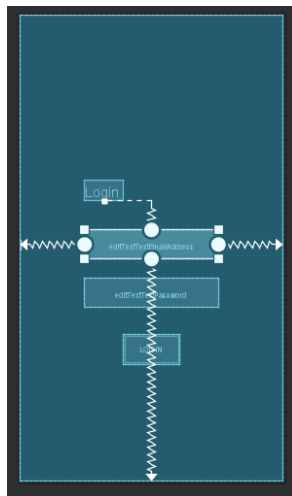


Figure 16 shows a GUI representation of the constraint layout acting on a text view.

This system implements a range of different layouts. The parent layout is always the Constraints layout. However, on pages that display a list I have implemented a scroll view layout with a linear layout contained. This is so that objects do not extend beyond the parameters of the view. The linear layout view appends each new object vertically or horizontally to the end of the last object, depending on the layout's configuration. This is important because it helps to keep user created objects in a consistent layout style.

5.9 Workouts

Sprint 2 incorporated the implementation of workout creation and viewing. As part of this, the user's workouts are listed using a custom button constructor after the user selects the workout navigation item. This was achieved in a similar method to the users however user-workout association information is first gathered from the User_Workouts_association table as shown in figure 17 below:

```
public ArrayList<Workout> listWorkouts(User user) {
    SQLiteDatabase db = this.getReadableDatabase();

    String sqlw = "select * from " + USER_WORKOUTS_ASSOCIATION_TABLE_NAME.getValue() + " where ((user_id = '" + user.getId() + "'))";
    Cursor nresults = db.rawQuery(sqlw, null);

    ArrayList<Workout> workoutlist = new ArrayList<Workout>();

    for (int j = 0; j < nresults.getCount(); j++) {
        nresults.moveToNext();

        String sql = "select * from " + WORKOUTS_TABLE_NAME.getValue() + " where ((id = '" + nresults.getInt(nresults.getColumnIndex("workout_id")) + "'))";
        Cursor results = db.rawQuery(sql, null);

        for (int i = 0; i < results.getCount(); i++) {
            results.moveToNext();
            Workout workout = new Workout();
            workout.setId(results.getInt(results.getColumnIndex("id")));
            workout.setName(results.getString(results.getColumnIndex("name")));
            workout.setDescription(results.getString(results.getColumnIndex("description")));
            workout.setOwnerId(results.getInt(results.getColumnIndex("owner_id")));
            workoutlist.add(workout);
            Log.d(LOG_TAG, workout.getName());
        }
    }
    return workoutlist;
}
```

Figure 17 displaying a method that accesses the database and returns a list of workout objects.

The custom button then constructor creates a new linear layout and adds a new button to that layout that matches its size. That linear layout then gets appended to a list of linear layouts within a scroll view (**See 5.8 XML**). The constructor also makes use of methods inherited from the fragments class such as shown in figure 18 below, where the method passes the retrieved workout object to a View workout fragment for the purpose of quickly referencing the workout_id.

```

private void addbutton(Workout workout1, View view) {
    LinearLayout layout = (LinearLayout)view.findViewById(R.id.svLL_createExercise);
    newbtn = new Button(view.getContext());
    newbtn.setText(workout1.getName());
    layout.addView(newbtn);

    newbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            FragmentTransaction fragmentTransaction1 = getFragmentManager().beginTransaction();
            fragmentTransaction1.replace(R.id.fragment_container, new ViewWorkoutFragment(user,workout1));
            fragmentTransaction1.addToBackStack(null);
            fragmentTransaction1.commit();
        }
    });
}

```

figure 18 displays the implementation of a custom button constructor that calls methods inherited from the fragments class.

The view workout section offers the option to delete or edit a workout, however it only does so if the workout was originally created by the user. Although, if delete is chosen, it removes that user_workout association regardless.

Further development into the use case - creating a workout meant users had to be capable of creating exercises and sets to correspond. Regarding Exercise sets, by use of edit text boxes, the system allows for users to input their own measures. For example, Users are not constrained to simply 'weight' and 'reps', they can choose to measure themselves in anyway that they can think ie. Distance and seconds.

To update the values of workout, associations, exercises, and sets. Save buttons have been implemented on each page. On clicking said buttons, using an onclick listener, the system retrieves the information stored in the relevant edit text fields and prompts an update method that calls an SQL 'UPDATE' command. This command also must specify data changes using the key word 'SET'. As seen below in figure 19:

```

public void updateWorkout(Workout workout) {
    SQLiteDatabase db = this.getReadableDatabase();

    db.execSQL("update " + WORKOUTS_TABLE_NAME.getValue() + " set name='" +
workout.getName() + "' where ((id = '" + workout.getId() + "')) ");
}

```

Figure 19 displays the use of 'set' to 'update' a workouts set of values 'where' the values meet the parameters

5.10 Session Handler

At this point, I had determined that the system would be better if the fragment classes were only used to establish front end operations such as initializing button operations. The reason for this was that - holding all of the methods in one location would greatly reduce future maintenance issues and simplify debugging.

To begin doing so, an interface called 'SessionHandler' was implemented. By using Interfaces in OOP programs, class instances of the interface can achieve multiple inheritances. They can also achieve a greater level of abstraction. Abstraction makes applications extendable in easier way. It also makes refactoring much less demanding. In this System, all of the methods were planned and listed in the SessionHandler. The class 'SessionHandlerImpl' implements all of the methods held in sessionhandler and provided the medium to which fragment classes could interact and call operations from DBHelper class to the database. This made defining new methods and testing much easier through development.

5.11 Nutrition

User stories regarding nutrition required a range of surface level functions, such functions include listing today's entries and calculating the sum of calories consumed that day compared to the displayed goal amount. When listing this data it was important to query the table and select entries that first – Only belong to that user. And Second – match the date in which the current date occupies. This was achieved by adding a WHERE statement to the SELECT Sql query in the FOOD_ENTRIES table. Then by using an AND operation to return only values that meet those specified requirements. A for loop is then used to list through received entry objects to calculate the total consumed calories.

Other functions implemented relating to nutrition User stories include: Searching for foods by name in a food database, listing foods that belong to the use, creating new food, creating new entries and many more.

```
public void createEntry(Food food, int userId, double servings, String date ) {
    SQLiteDatabase db = this.getReadableDatabase();
    db.execSQL("insert into " + FOODS_ENTRY_TABLE_NAME.getValue() + " (food_id, owner_id, name, date,
my_serving, serving_desc, calories, protein, carbs, fats) values (" + food.getId() + ", " + userId +
", '" + food.getName() + "', '" + date + "', " + servings + ", '" +
food.getServingSize()+food.getUnitOfMeasure() + "', " + (food.getCalories()*servings) + ", " +
(food.getProtein()*servings) + ", " + (food.getCarbs()*servings) + ", " + (food.getFats()*servings) +
");");
}
```

Figure 20 showing the calculation of nutritional values based on food object and serving information.

When the user creates a new entry the macronutrient values such as calories, Proteins etc are calculated at the point in which they are added to the FOOD_ENTRY database (see figure 20 above). The FOOD_ENTRY database and FOOD_database now have similar fields.

However, it was implemented in this way so that in the occasion that Food data is lost, updated or deleted from the Food data base, a user whom has used that Food date to create an entry will not be affected by this change. Future iterations of the system may clean the Food_entries database of foods that do not hold the current date.

5.12 Log Calendar

The log calendar makes use of Android studio's calendar widget (see figure 21). After implementing, it could be manipulated to call functions that correspond to specific dates. Those functions include displaying the contents of a log and or updating that contents. If no log exists, a method is called that creates a new log and immediately returns it.

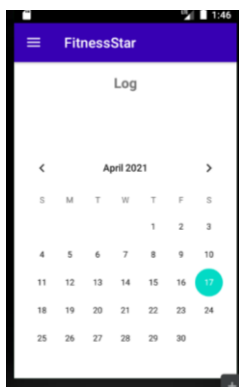


Figure 21 shows Android studio's calendar widget

Logs themselves are editable strings used to represent the fitness diary of a user. In this way Users may edit them at will or, a trainer to whom the user is connected may edit them. Workout and food entry log changes are automatically recorded. For example figure 22 shows the implementation of a food entry log:

```
if(entries != null) {
    for (FoodEntry entry : entries) {
        currentC += entry.getCalories();
        currentP += entry.getProtein();
        currentCrb += entry.getCarbs();
        currentF += entry.getFats();
    }
}

com.example.fitnessstar.model.Log l1 = new com.example.fitnessstar.model.Log();
l1.setOwnerId(user.getId());
l1.setLogDate(today);

l1.setEntry("[ + Food Entry] : "+food.getName() + "\n" +
    "[Daily Nutrition] \nCalories : " + currentC + "/" + goalC + "\n" +
    "P: " + currentP + "/" + goalP + "\n" +
    "C: " + currentCrb + "/" + goalCrb + "\n" +
    "F: " + currentF + "/" + goalF
```

Figure 22 shows the format and method in which new food entries are appended to the log for that date

5.12.1 Trainer Functions

The last sprint saw the implementation of trainer functions based on the functional requirements listed in section 3.4 Requirements alongside functions previously implemented. For example, a Trainer may send his/her client's workouts they have created or personally create editable workouts the clients. A trainer may now also edit client goals and suggest Workouts, nutrition or leave comments using the log files.

5.13 Testing

Testing phase was carried out at the end of sprint 2 and sprint 3 with the help of user stakeholders. Functional Testing was also carried out throughout the implementation where an expected result of a test case was compared to the actual result of that test case. This was mostly achieved by use of LOG_TAGS which printed string values to the IDEs run log. For Example

```
Log.d(LOG_TAG,user.getEmail());
```

-Prints the email address of user.

All features that are present in the final build have gone through this kind of functional testing and user testing.

6 Results

6.1 Technical Achievement

Over the development of this project, 91% of functional requirements were achieved. Despite an initial delay in progress, the final project could simulate real life usage and was capable of user lead field testing. In this time of social distancing the program facilitates a closer interaction between Personal Trainers and clients that it was intended for. Therefore, although the program requires further development, it may be considered a success.

In addition, throughout the phases of this project I have undergone significant personal development in regard to software development and skills related to this system. Prior to its inception I possessed only a brief experience using these methods of implementation. However, now I am confident that this experience has significantly improved my proficiency and will benefit all future projects.

6.2 Evaluation and Discussion

Due to covid restrictions it was difficult to connect with a multitude of testers. This was because all testing was done over video calls in which I carried out the actions based on the users instructions. Having said that 15 interviews consisting of 3 personal trainers 1 personal trainers' client and 11 non associated participants took place. Out of those 15, 9 had previously used fitness applications in the past. The user ages ranged from 21 to 49.

During the interviews, interviewees participated first in a demo phase, before then being asked a selection of questions and 4 key questions. They were then free to give any informal feedback in which many important topics were raised. The questions were as follows:

1. How old are you?
2. Have you any experience with trainer and or personal fitness?
3. Have you previously used any other fitness applications?

6.2.1 Key questions

4. Do you think fitness applications work in helping you to achieve fitness related goals? Why?
5. Is this system easy to use? If no why?
6. Do you think using this service would make interactions with a personal trainer and or client more convenient?
7. Do you think this system could help you to achieve your fitness goals? Why?

6.2.2 Question 4

'Do you think fitness applications work in helping you to achieve fitness related goals? Why?'

In 2020 Bracken and Waite investigated the behavioural change of participants and their perceived ability to achieve goals after using fitness applications. Here, participants were asked if fitness applications work in helping you to achieve fitness related goals and the results are as follows:

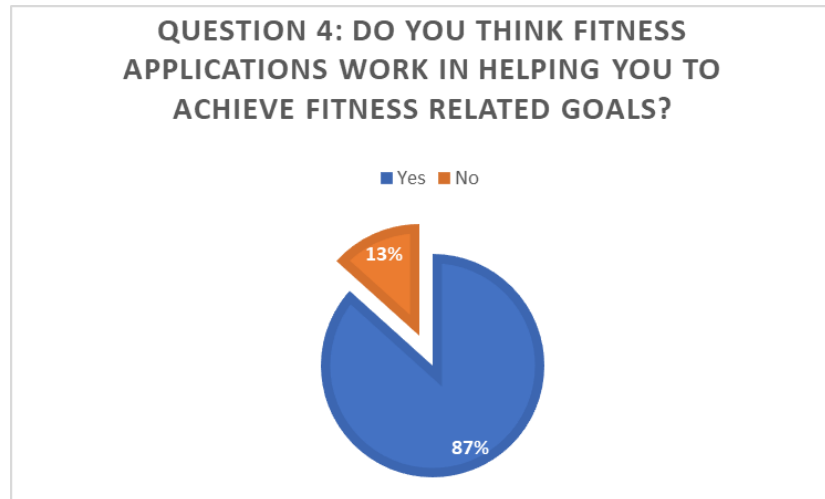


Figure 23 displays the chart corresponding to question 4 'Do you think fitness applications work in helping you to achieve fitness related goals?'

13 out of 15 participants respond 'yes' which would also reflect the findings of Bracken and Waite to which part of this fitness application had been premised. When participants were asked why they thought that the general consensus was that fitness applications support the user in their fitness endeavours by digitizing otherwise tedious tasks. One gave the example as follows:

'Fitness application are as much a tool a dumbbell or treadmill is, they're not necessary but they make things easier'.

Of the two that answered no, one of them gave the response that – 'fitness related achievements are only as a direct result of the users determination'. The other also said that fitness applications had no effect on the ability to achieve goals. Coincidentally, of the two that replied no, one was the oldest participant. However, because of the small sample size there is no way of conclusively investigating whether this affected their response.

6.2.3 Question 5

Is this system easy to use? If no why?

The decision to implement a very simple UI was with the intention of keeping the system easy to use. However, the feedback from this question uncovered some common flaws with its fundamental design. Of the 15 participants, 10 answered yes and 5 answered no.

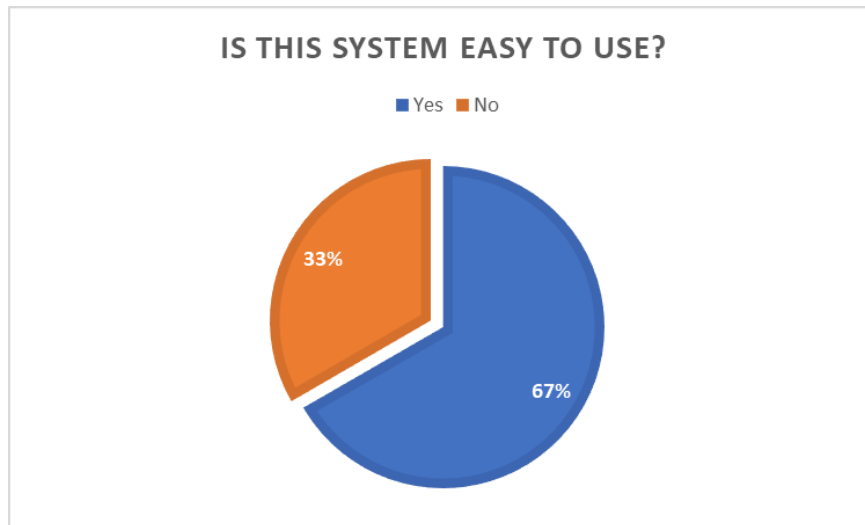


Figure 24 displays the chart corresponding to question 5; is the system easy to use?

Although all participants considered the GUI to be easy. 5 users took issue over the idea of user set calorie and macro nutrient goals with a common answer being-

‘I don’t know much about the number of calories, protein, carbs and or fats I should be consuming’.

Reflecting on this, the program could be improved by incorporating an automatic system of calculating user calories as an alternative to asking users to fill in all fields alone. However, as this system stresses the assistance of Personal trainers to do this for Clients, it did not reach the functional requirements the design phase see 3.4.1 Functional Requirements

6.2.4 Question 6

Do you think using this service would make interactions with a personal trainer and or client more convenient?

More convenient trainer - client interactions were the aims of this system. During this time of covid 19 restrictions many trainers and other social professions have struggled to maintain clients. It was my hope that an app of this design could mitigate these disadvantages to some degree. The results found that all 15 participants thought that this service would improve the interactions between clients and trainers.

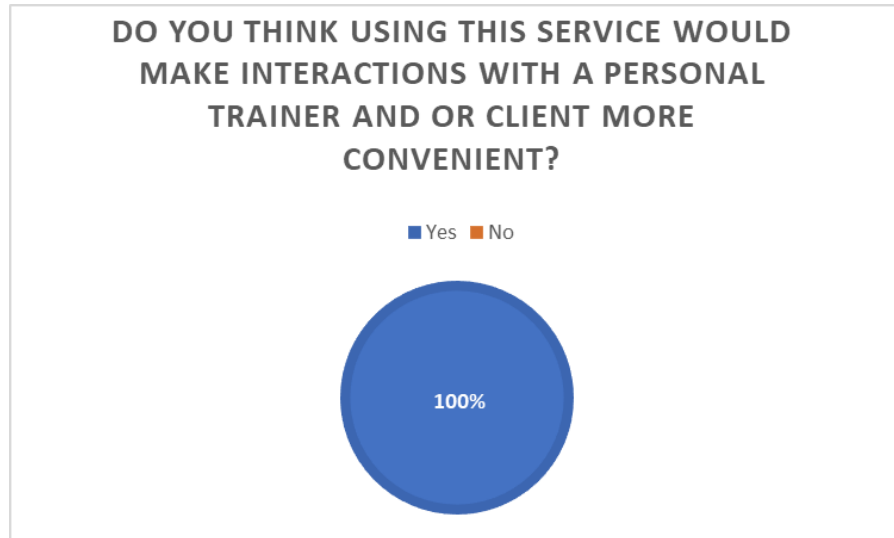


Figure 25 chart that corresponds to question 6; Do you think using this service would make interactions with a personal trainer and or client more convenient?

Although all participants agreed, many gave suggestions as to how it could be furthered, such as: 'Access to google maps and proximity based trainer sorting', 'Searching for exercises already created by other users much like food'. This feedback was incredibly useful and could be used to evolve the software in further agile sprints by means of software updates.

6.2.5 Question 7

Do you think this system could help you to achieve your fitness goals? Why?

This question is directly in regard to the proposed system; however, it may also be influenced by whether or not the participant has or has had access to a personal trainer. The results are as follows.

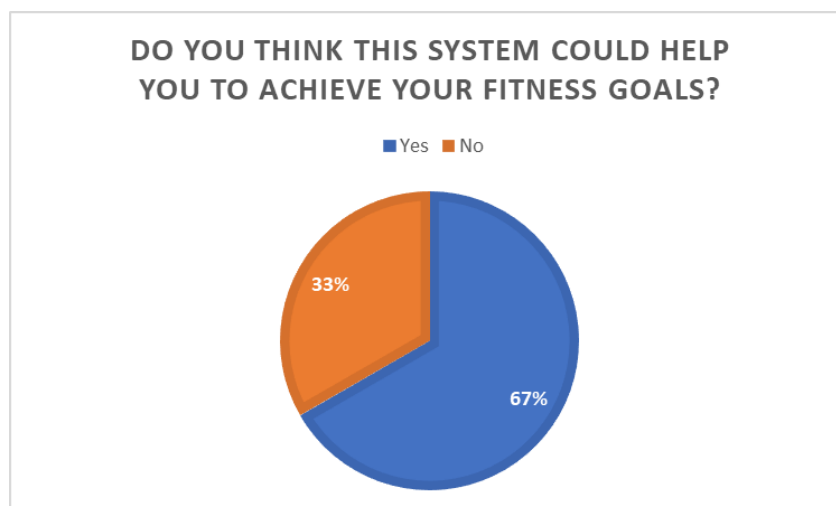


Figure 26 displays the chart that corresponds to the question; Do you think this system could help you to achieve your fitness goals?

10 out of 15 participants said yes. This is important because, 3 less participants answered yes in comparison to question 4 where 13 people answered yes. Although 10 participants still answered yes, this decline would indicate that the system had shortcomings that other proposed systems did not. When asked why, the participants once again suggested that implementing features that automize nutrition goal calculations based on desired weight would vastly improve the user experience should they not have access to a personal trainer.

6.3 Constraints and limitations

The use of Android studio with little to no prior experience led to the surfacing of many fundamental issues as development began. Such issues include the use of SQL stored on a local system rather than accessed from the cloud. This constraint is obviously in direct conflict with the core proposed environment of the system as the databases must be stored remotely and accessed through the internet in a professional system.

Another core constraint of the current system was that all testing was done using the same emulator. During the layout design process testing was not given to mobile devices of different screen sizes therefore I would predict that the GUI would not meet the same aesthetic as it does on the current emulator. This would have a guaranteed negative effect on the users experience. The current Emulator is a:

NEXUS S running Andriod 6.0 Oreo / API 26

In addition as the user data base was stored in onto a local drive implemented no current way of adding new users to the system and only hard coded a limited amount for testing purposes. Those users are as follows and can be found in the Testing hints documentations found in the repositories:

First Name	Last Name	Username	E-mail	Password	User Type
John	Strongman	Stronjohn	john@strongman.com	password	Trainer
James	Winsor	jamesw	james@winsor.com	password	Client

Table 2 displays seeded data required to access the system.

Constraints in regard to the proposed professional system include connection to the internet in order to search and receive data from the larger databases. This can be mitigated by

storing an instance of user related objects on the device however it would not allow users to for example search foods, add foods to the public database, or engage in any trainer related functions which are at the systems core.

In addition, system infrastructure such as the databases must be online at all times, this would incur financial and maintenance costs which have not been considered in this design. The users may also face significant reduction in performance if traffic is not handled correctly on the system. Furthermore, as there is no limit on the size of databases, particularly the log database, it is difficult to predict how this may affect the performance of the system and its professional feasibility.

7 Professional Issues

7.1 Project Management

During the planning phase, a Gantt chart was produced to forecast technical progress and project deadlines. However, after experiencing several technical issues as well user testing restrictions as result of covid-19, predicting intended deadlines became increasingly unreliable. Due to this reason the Gantt chart was only updated once, and then later disregarded altogether as it became progressively redundant. Instead, progress was made more flexibly around technical achievement and Stakeholder availabilities. This type of incorrect project time estimation was a key risk and is later mention in the Risk table R.10 in 7.2 Risk Assessment.

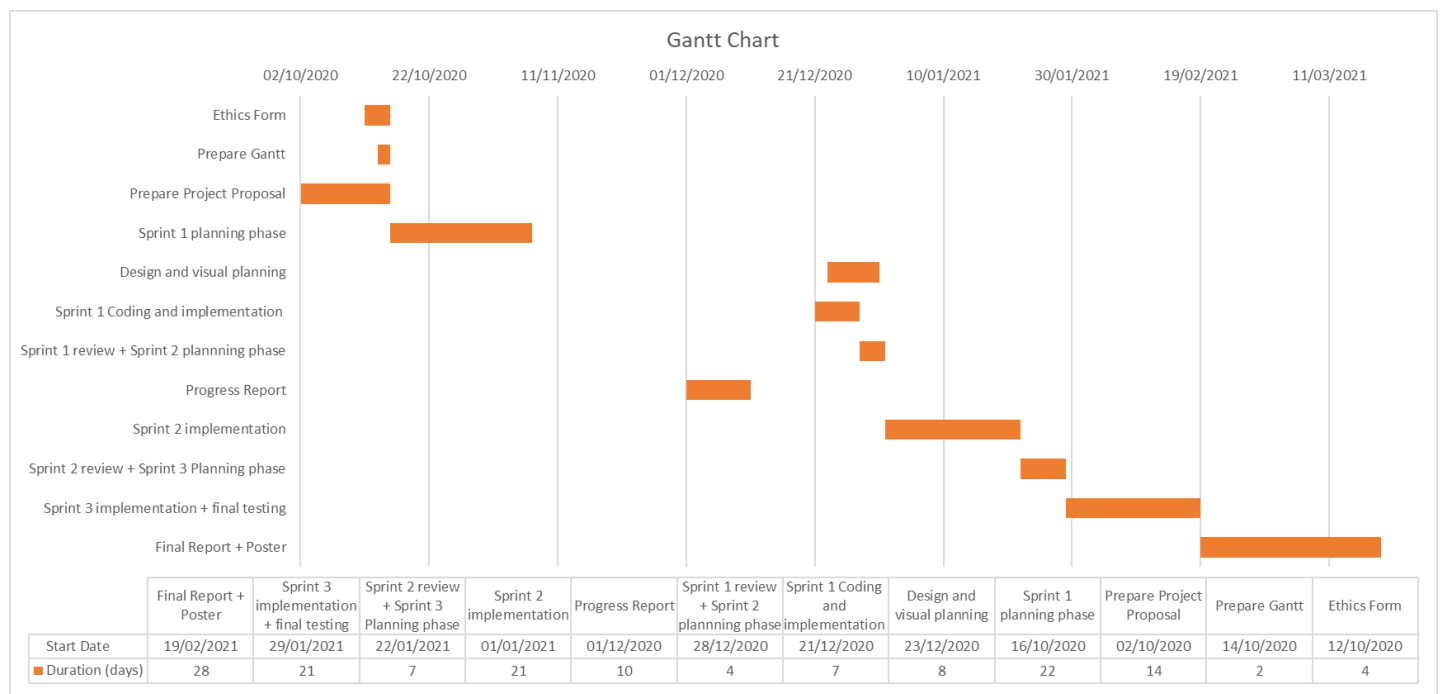


Figure 27 displays a project management Gantt chart.

7.2 Risks Assessment

Risk ID	Potential Risk	Cause ID	Potential Causes	Severity	Likelihood	Risk	Mitigation ID	Mitigation
R1.1	Missed deadline	C1.1.1	Illness	1	3	3	M1.1.1	Register exceptional circumstances if ill.
		C1.1.2	Cannot choose topic	1	1	1	M1.1.2	Conduct research early and meet supervisor
		C1.1.3	Poor time management	4	3	12	M1.1.3	Make a Gantt plan early
R1.2	Feature creep	C.1.2.1	Over-ambitious project spec.	3	5	15	M1.2.1	Discuss plan with supervisor early. Create basic requirements
R1.3	Hardware failure	C.1.3.1	Broken device	5	2	10	M1.3.1	Buy new device/ use public device and back up data
R.1.4	Gold-plating	C1.4.1	Over-ambitious project spec.	3	4	12	M1.4.1	Refine requirements to essentials
R.1.5	Excessive difficulty	C1.5.1	Over-ambitious project spec.	4	5	20	M1.5.1	Research technical skill requirements.
R1.6	Software bugs	C1.6.1	Non-modular design	1	3	3	M1.6.1	Create highly modular design before implementation
		C1.6.2	Poor test plan	4	3	12	M1.6.2	Create test plan at start
		C1.6.3	Buffer Overflow	4	4	16	M1.6.3	create an extensive test into the capabilities of the buffer overflow
		C1.6.4	Dangling pointers	4	3	12	M1.6.4	test and frisk for dangling pointers, before debugging.
		C1.6.5	Memory leak	4	3	12	M1.6.5	evaluate the memory during testing
R1.7	Real-time performance short falls	C1.7.1	Slowness and poor design	2	3	6	M1.7.1	Carefully considered technical design
R1.8	Loss of data	C1.8.1	Poor version control	4	2	8	M1.8.1	Implement version control strategy at start.
R.9	Test personnel limitations	C1.9.1	Covid-19 pandemic	3	2	6	M1.9.1	Test open areas or virtually
R.10	Incorrect project time estimation	C.1.10.1	Over-ambitious project spec.	4	4	16	M1.10.1	reduce ambitions to feasible task

Table 3 displays a list of anticipated risks associated with this project.

7.3 BCS Code of conduct and practices

There are many relevant BCS Code of conduct and practices I must inherently adhere to, some of which are as follows:

Regarding public interest:

Members shall in their professional practice safeguard public health and safety.

Although it is not directly relevant to the development of this app, in a grander project of this sort there must be a way to verify the Personal trainers to which we advertise.

Members shall ensure that within their chosen fields they have knowledge and understanding of relevant legislation.

See Data protection Laws in 7.4.1 Legislation.

Members shall in their professional practice have regard to basic human rights.

Consent of testing participants must first be given.

Regarding duty to employers or clients

Members shall endeavour to complete work undertaken on time.

Regarding duty to the Profession

Members shall uphold the reputation of the Profession and shall seek to improve professional standard through participation in their development, use and enforcement, and shall avoid any action, which will adversely affect the good standing of the Profession.

7.4 Legislation

7.4.1 The UK Data Protection Act

The UK Data Protection Act, empowers people to take control of their data by passing this bill of regulations, including:

- Implements GDPR across all general data processing.
- Provides restrictions to rights to access and delete data.
This is specifically relevant to this project as a user may hold a profile, that may hold sensitive information e.g., location.

This is Important because currently the system is in violation of this Act as users have no means to delete their User information. However, this is not the case for the proposed system.

For a full overview, visit the following site:

[Data Protection Act 2018 Overview - GOV.UK \(www.gov.uk\)](https://www.gov.uk/data-protection-act-2018-overview)

7.4.2 Android

By using Android studio, it is more than anticipated that this system would be uploaded to Google's Application Market – 'play store'. Therefore it would be necessary that it abides by the regulations of Android Developer distribution Agreement, which may be found at the following address:

<https://play.google.com/about/developer-distribution-agreement.html>

7.5 Ethical

7.5.1 Offensive content

As the databases are filled with user created content, the system risks the propagation of offensive content for example food names or Trainer profiles. This can be minimized by adopting a system of automatically filtering profanity. However this feature is not contained within the original scope of project.

7.5.2 Validated trainers

As with all systems of meeting new people there is a risk to public safety to which this service would have some degree of ethical responsibility over. To address this, stakeholders may benefit from having a method of 'vet'ing Personal Trainers.

7.6 Data Management

All data management documentation and files can be found in the repositories under Individual sprints, See **10.Repositories**.

8 Conclusion

Overall, this project has demonstrated the core functionalities of the proposed systems. A closer relationship with Trainers and Client progress has been achieved. Thereby adhering to the initial aim of creating an app that allows Personal Trainers to interact with Client fitness records more closely and directly.

From the User feedback and research, we can infer that this proposed system of Client-Trainer interactions may serve to benefit the effectiveness of their professional relationship, and thereby improve the efficacy of Personal Trainers that use it better than existing systems.

However, we may also conclude that the automation used by existing systems, in regard to calculating calories and macronutrients, may become the detriment of user satisfaction towards systems that do not incorporate such features.

Fitness applications are set of tools, simply adding more fitness related tools would incontrovertibly increase the effectiveness of that tool set.

8.1 Future work

Most popular mobile applications come under 'Embedded' Type software using Lehman's Classification software. This means they often have opportunity to evolve and can be subject to regular updates regardless. This system is no different however there are also immediate improvements that can be made.

Firstly, the database infrastructure must be transferred to a cloud environment, and the system must be adapted to access that service. This step is essential before any official prototype of the product may be distributed in the public market.

In addition, as suggested by the user feedback, autonomy of Nutrition calculations is a must that is implemented by already popular existing systems.

Furthermore, the GUI of popular mobile applications significantly contribute to their success. Therefore, achieving a more visual pleasing aesthetic to the front end of this application is essential. This would be achieved by further studies into the defining aspects of popular mobile application GUI and its findings would then be applied to this system. Leading on from that, users also suggested relative location-based trainer advertisement via google maps. This functionality would greatly improve relevant trainer advertisement and accessibility. It may also be achieved without great difficulty as a result of Android studio's Google maps integrated widgets.

Other areas of further development include pre-populating food databases with public food databases and allowing users to quickly attain information by using their camera to scan barcodes as seen in the optional requirements.

8.2 Reflections

Despite a gathering of technical issues, this system was able to simulate the core functionalities of its proposed design.

Although there is still much room for improvement, developing this project has provided an excellent learning experience. Since its inception I have significantly increased my independence with Java programming, SQL and Android studio. I am confident that if I was to do this project again, I could achieve a more professional result.

The area of the project that went well was the planning a designing phase, incorporating an agile approach and involving user stakeholders. However, from there I underestimated the original complexity and scope of the project in accordance with my SQL, java, and android studio proficiency, this resulted in unrealistic deadlines.

2021 has also certainly been an unusual year filled with the unpredictability of covid-19. This unpredictability has seen to its own delays in user testing, research, and implementations. Located in London's borough of Newham I underestimated the effect it would have on time management. Reflecting on this I still believe, whilst not boasting success, from the results I was able to develop an artifact that benefited its intended user base thus fulfilling my initial aims.

9 Bibliography

Ansell, M. (2008) Personal training. Exeter: Learning Matters (Active learning in sport). Ch.2,4,12

Bracken ML and Waite BM (2020) "Self-Efficacy and Nutrition-Related Goal Achievement of Myfitnesspal Users," Health education & behavior : the official publication of the Society for Public Health Education, 47(5), pp. 677–681. doi: 10.1177/1090198120936261.

Evans, D. (2017) "Myfitnesspal," British Journal of Sports Medicine, 51(14), pp. 1101–1101. doi: 10.1136/bjsports-2015-095538.

Kruger, J., Blanck, H. M. and Gillespie, C. (2006) "Dietary and Physical Activity Behaviors among Adults Successful at Weight Loss Maintenance," The International Journal of Behavioral Nutrition and Physical Activity, 3, pp. 17–17. doi: 10.1186/1479-5868-3-17.

Maughan, R. J. and Burke, L. (2002) Sports nutrition. Malden, Mass.: Blackwell Science (Handbook of sports medicine and science). doi: 10.1002/9780470757185.

Petrescu, M. (2014) Viral marketing and social networks. New York: Business Expert Press (Digital and social media marketing and advertising collection). Available at: <https://ebookcentral-proquest-com.oxfordbrookes.idm.oclc.org/lib/brookes/reader.action?docID=1684366&ppg=94>

(Accessed: October 16, 2020).

Rajput, M. (2020). Why Android Studio Is Better For Android Developers Instead Of Eclipse, Mobile Zone, Available at: <https://dzone.com/articles/why-android-studio-better> (Accessed: 23 April 2020).

Testa, R. J. and Brown, R. T. (2015) "Self-Regulatory Theory and Weight-Loss Maintenance," Journal of Clinical Psychology in Medical Settings, 22(1), pp. 54–63. doi: 10.1007/s10880-015-9421-z.

Villasana, M. (2019) Mobile Applications for the Promotion and Support of Healthy Nutrition and Physical Activity Habits: A Systematic Review, Extraction of Features and Taxonomy Proposal. Volume 12. TOBIOJ-12-50, page 50-71, Available From: <https://benthamopen.com/ABSTRACT/TOBIOJ-12-50> [Accessed 26 October 2020]

Wiemeyer, J. (2019). Evaluation of mobile applications for fitness training and physical activity in healthy low-trained people - A modular interdisciplinary framework. International Journal of Computer Science in Sport 18, 3, 12-43, Available From: Sciendo <https://doi.org/10.2478/ijcss-2019-0016> [Accessed 26 October 2020]

10 Repositories

Shared Google drive:

<https://drive.google.com/drive/folders/1OPDoP38OrL96siwH3Kqk19A78i7bcp9H?usp=sharing>

Video demo:

<https://youtu.be/h0gdDbrbBvE>

Proposal

<https://drive.google.com/drive/folders/1VeKUvDHRZQeZhFowTgZjyz2fMUHQT7J6?usp=sharing>

Final Artifact:

<https://drive.google.com/drive/folders/1gcFEUtZvLY70XtCTI2NKoqataPBn7krY?usp=sharing>

Designs and Models:

<https://drive.google.com/drive/folders/1j86SuoICTiZKeHtDzdpXWhQb6CSx46X6?usp=sharing>

Progress report

https://drive.google.com/drive/folders/1fnHnUw9PvvnAC3Wd_ckpgJzVnNvHSrQ6?usp=sharing

Sprint plans and reviews

https://drive.google.com/drive/folders/1Kl_eTGIRMz_f-DF7g3pJiiGQEWoOxmbZ?usp=sharing

Sprint version control

<https://drive.google.com/drive/folders/1GDU8xSzPejoEeOiQIX5c9QFIBztRsJbS?usp=sharing>

Interview results

https://drive.google.com/drive/folders/1MyReXXLEWyJlsglGDIQ_cnRgxt3Ol60A?usp=sharing

Ethics Form

https://drive.google.com/drive/folders/1QqxxhaxwrqQ2HQek-6KnUpyKcMHevQ9_W?usp=sharing

