Goldsmiths
UNIVERSITY OF LONDON

A System To Develop Smarter Gyms & Improve User Experience In The Gym Through NFC Technology

by

Thomas McAloon (ID: 33618177)

GitHub: Prototype

GitHub: Final Build

Goldsmiths, University of London

Supervisor: Tara Collingwoode-Williams.

BSc Computer Science
2023

Declaration

This project report is submitted in partial fulfilment of the requirements for the degree of BSc Computer Science. I declare that this thesis was composed by myself, that the work contained therein is my own, except where explicitly stated otherwise in the text, and that it has not been submitted, in whole or in part, for any other degree or professional qualification.

Thomas McAloon (ID: 33618177)

This thesis was conducted under the supervision of Tara Collingwoode-Williams.

Abstract

This thesis aims to develop FitSmart, a fitness application that makes use of Near Field Communication (NFC) technology to provide a more user-friendly and effective interface. By just tapping their NFC-enabled cellphone against appropriate gym equipment, users of the app can quickly and effortlessly enter workouts and measure progress. The thesis also contrasts typical QR code scanning with the use of NFC for tracking gym equipment.

The thesis is introduced with a study of fitness tracking apps in use today and the limitations of conventional QR code reading. The idea of NFC technology and its potential advantages for fitness tracking are then introduced. The construction of a user-friendly interface, the integration of NFC technology, and the use of the Flutter framework were all part of the development process for FitSmart.

The usefulness of FitSmart is then assessed, and its performance is contrasted with that of conventional QR code scanning, during a user testing phase. According to the test's findings, people preferred using the NFC-enabled FitSmart app over QR code scanning since it was more user-friendly and effective. Furthermore, it was discovered that the app's accuracy and speed for tracking gym equipment usage were superior to conventional QR code scanning.

The thesis also discusses future work and the app's limitations. Although FitSmart offers a better user experience, its functionality is currently restricted to gym equipment with NFC capabilities. Future work will involve bringing the app's compatibility with more devices and creating new features to improve the user experience even more.

Overall, this thesis illustrates the advantages of NFC technology and offers a novel method for tracking fitness. The creation of FitSmart and the outcomes of the user testing phase demonstrate the usefulness of the software and its potential for advancement.

Contents

List of Figures

Abbreviations

IoT Internet of things.

MVC Model-View-Controller.

MVVM Model–view–viewmodel.

NFC Near Field Communication.

PII Personal Identifiable Information.

PWA Progressive Web Application.

QR Quick Response.

RFID Radio-frequency identification.

SDK Software Development Kit.

SSL Secure Sockets Layer.

UI User Interface.

UUID Universally Unique Identifier.

UX User Experience.

Note: Author abbreviations are shown in their corresponding reference entry.

Words marked with * indicate abbreviations

Chapter 1

Introduction

1.1 Overview

The main focus of this project is to build a fitness web application based around NFC tags, which will allow the user to tap their device on a piece of gym equipment to log their progress. NFC stands for Near Field Communication, a technology based on RFID* that enables short-range communication between devices using a magnetic field induction. This project will also assess the convenience of NFC tags to similar technologies.

1.2 Motivation

I have been enthusiastic about going to the gym for many years and have recently been heavily invested since returning to Gaelic Football. As I become more educated in fitness and training, the more I understand the importance of science-based training. However, I have found logging exercises to be a second thought when in the gym and have often sought a more convenient way to track progress in the gym.

There are several apps available to log exercises, such as 'MyFitnessPal' and 'Strong'; however, none use NFC technology. In addition, they require the user to input each machine/exercise and each set, which is not convenient when you are focused on working out. Also, these apps require the user to have their phone on their person throughout their workout. NFC tags can solve both problems as this technology can be scanned using any

NFC-compatible device, such as a smartwatch or smartphone (any device that can be used for contactless payments). Furthermore, NFC tags can be programmed to store information about the gym equipment meaning the users will no longer have to input as much information when logging their workout manually.

1.3 Plan

1.3.1 Aims & Objectives

The main focus of this project is to build an intuitive and efficient fitness web application developed in conjunction with NFC technology that focuses on providing the user with essential health/fitness data. The unique selling point will be that users only have to tap their smartphone to an NFC tag, allowing them to quickly log their sets with that machine.

For this project to be successful, the application will need to include the features of its competition and integrate the NFC functionality. Focusing on science-based training is paramount, as it currently needs to be added across many fitness applications. This could be another attractive feature to make the app stand out in a competitive market. The app will also feature the functionality to save past workouts; this allows users to view their previous personal best and be prompted about the next exercise/machine they intend to use. This feature should improve efficiency for the user. Charts and graphs will also be featured within the application so that the user can quickly view their performances, and the visuals will be clear and concise.

This project will utilise the "mobile first" mindset. The application will take advantage of a plethora of technologies when being developed and deployed, such as Visual Studio Code (IDE), React (Front-end), Google Firebase and Node.js (Back-end), and GitHub (Version Control/Deployment). Although the app will be developed for smartphone use, these frameworks are not for native app development. The application will be responsive and function as intended on multiple devices and web browsers. This approach has been decided to narrow this project's scope without compatibility with other devices. A web application reduces the risk of limiting the project to a handful of devices.

1.3.2 Resources & Techniques

- Pre-existing access to resources:
  - Goldsmiths Library (Research Papers)
  - Google Scholar (Research Papers)
  - A Gym/Gym Equipment
  - Smartphones (Compatible with scanning NFC Tags)
  - Raspberry Pi 4 Model B/Server (£40.00)
  - Development Environment (Visual Studio Code, React, Node.js, Firebase, MySQL)
- Resources to be sourced:
  - NFC Tags (£0.60/count)
  - SSD to USB 3.0 Cable for Raspberry Pi (£6)
  - WD Green 240GB 2.5" SSD (£29.90) / Fanxiang S101 256GB SSD (£18.99)
- Techniques used in this project are as follows:
  - Full Stack Development
  - Data Mining
  - Data Visualisation
  - Database Design, Development & Management
  - UI/UX Design

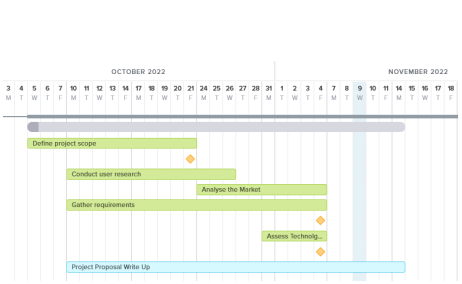
  Knowledge will be drawn from current and past modules, such as Front-End Web, Dynamic Web Applications, Database and the Web, Software Project 2 and Data Visualisation. This knowledge provides a confident base to develop the project and deliver a high-standard product.

  Furthermore, the use of NFC technology and managing sensitive data has been gained from a working environment in which an app featuring the use of NFC technology was

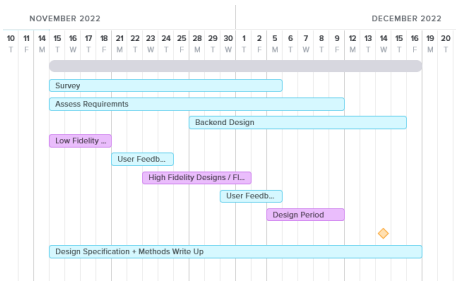
developed. Also, from working within the industry, the importance of handling sensitive data is significant. di Vimercati et al. (2012)

1.3.3 Gantt

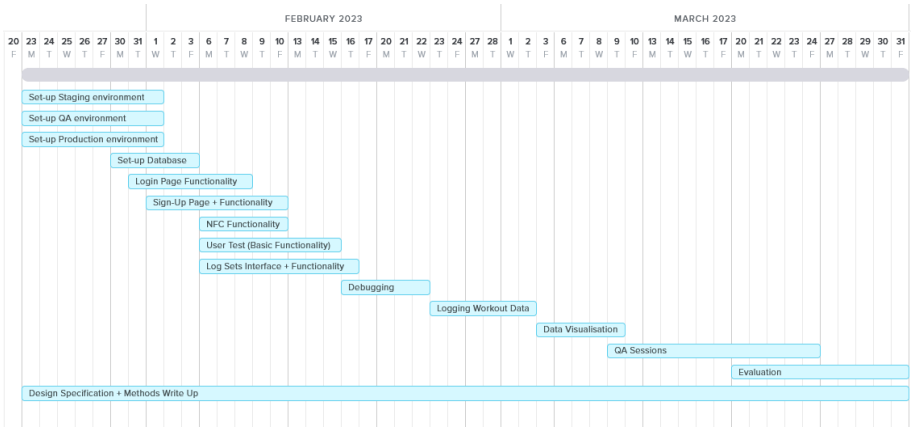
A Gantt chart is a visual representation of the project's activities/tasks displayed against time. It will act as a road map for the project, highlighting each task's start, duration and end, including significant milestones. Constraints and risks within the project should be mitigated through a Gantt Chart, making the project's development efficient and optimal. Having a Gantt Chart (See Figure 1.1) is a form of contingency for the project allowing adjustment if the critical tasks are delayed. Randolph (1998)



(a) Research & Project Proposal



(b) Design Specification & Methods



(c) Implementation & Analysis

Figure 1.1: Gantt Chart

1.4 Section Overview

Chapter 1 (Introduction): This section introduces the project, outlining the motivation behind the project and giving an overview of the concept of the project. A brief plan for the project is discussed, including the aims and objectives, resources and techniques to be used

and a Gantt chart.

Chapter 2 (Background research): This section provides the background research that was conducted for this project. Both qualitative and quantitative research methods were analysed to improve the understanding of the project's intent. By researching existing applications and identifying available technologies, much greater insight was gained into what was possible.

Chapter 3 (Design Specification and Methods): This section outlines the key features and design elements necessary for the project's successful development, including hardware and software requirements, user interface design, authentication methods, and data storage. Additionally, the necessary methods and technologies required for proper implementation of the project, including the components necessary to facilitate data exchange between the app and the user's NFC-enabled device.

Chapter 3 (Design): This section provides design diagrams, that visualise how components of the application will work, as well as explanations as to why certain design decisions were made.

Chapter 4 (Implementation): This section explains how each component of the application works, by providing code examples and explanations.

Chapter 5 (Testing): This section provides detail around all testing, such as technical testing and usability, the rationale for each test will be provided, as well as their results. This section will help identify whether the application met the requirements identified in section 3.

Chapter 6 (Evaluation & Summary): This section presents an overview of the project's major findings, constraints, and future research. It includes a review of the app's advantages and disadvantages, an explanation of the evaluation procedure, and the user testing findings. A reflection on the project's entire development process and its difficulties is also included in this part. The last part, under Future Works, describes prospective areas where the FitSmart app could be developed and improved further, including the addition of new features and device integration.

Chapter 2

Background Research

2.1 Literature Survey

The main goals of the study are to determine whether consumers find using fitness applications at the gym to be inconvenient and whether there is a correlation between user engagement and convenience. To disprove the notion that NFC is a more user-friendly technology, some of the project will concentrate research on NFC vs QR technology. Analysing the reliability of research papers is crucial to assess their ecological validity and the effect this has on the project's progress.

Zhang et al. (2018) studied the application of NFC technology for tracking physical activity. According to the study, NFC technology may be used to precisely track physical activity, with a data accuracy of 10 seconds. The researchers concluded that NFC technology may be utilised to accurately track physical activity.

It has also been investigated how NFC technology can be used to acquire personal fitness data. Zhang et al. (2019) investigated the application of NFC technology to obtain personal fitness data in their study. The authors came to the conclusion that NFC technology can be utilised to safely and swiftly obtain personal fitness data. Furthermore, they discovered that NFC technology can be used to accurately store and access personal fitness data.

Regarding user security and privacy, NFC technology use in fitness applications has repercussions. The authors of a study by Watson et al. (2017) looked at how NFC technology

for fitness applications can affect security and privacy. The authors discovered that employing NFC technology for fitness applications can offer a safe and private way to access one's fitness data, but they also pointed out that there may be security and privacy hazards involved.

2.1.1 Article 1 - Mobile Health Apps to Facilitate Self-Care: A Qualitative Study of User Experiences

This article gives insight into how users perceive health apps. A Qualitative Study of User Experiences on Mobile Health Apps by Curtin University found that consumer interactions somewhat depended on how usable the app was. "User experiences have been described via four themes: Engagement, Functionality, Information Management and Ease of Use. These themes describe concepts such as motivation, customisation, interconnectivity, data inaccuracy, convenience and competitiveness" Anderson et al. (2016). The study mentions "ease of use" and "convenience" as relevant factors behind users' engagement with health apps. If the app is not intuitive, then people will not use it. However, the study also found that some users feel that using their phones in the gym is an unhealthy lifestyle choice and would prefer to use their workout time as a smartphone hiatus. A personalised app experience promotes higher levels of engagement and an overall better user experience, as indicated by Anderson et al. (2016).

This study elucidates an array of contributing factors towards health; however, this needs to be revised when attempting to understand the direct implications of health apps in the gym. Furthermore, this article is potentially considered outdated for this field of research as it was published in 2016. Conversely, this article shows credibility and is apposite to the project as the study outlines the participant's demographics (Figure 2.1(a)), devices and type of apps used by participants (Figure 2.1(b)).

This article needs to explore whether fitness apps can be made more convenient for users and if convenience would engage users more. This is a key area the project will explore via developing a fitness app that is perceived to be less of a chore. In addition, the research will be furthered by recording whether there is a correlation between convenience and user engagement.

Characteristic	Subcategory	Number
Gender	Female	15
	Male	7
Age (years)	18–25	4
	26–35	13
	36–45	2
	46–55	2
	>55	1
Total Participants	Met inclusion criteria	22
	Excluded	4
Interview Duration (minutes)	Mean interview time	20
	Shortest interview	15
	Longest interview	41
Smartphone Operating System	Android	7
	Apple	15
	Windows	0
	Symbian	0
	Linux	0
Main Language	English	17
	Other	5
Recruitment Source	Physical university posters and online staff/student portals	10
	National Asthma Council eNewsletter and social media (Facebook)	3
	Rare Diseases Australia eNewsletter	2
	Celiac Australia eNewsletter	2
	Diabetes Australia eNewsletter	1
	Perth start-up community (posters and daily notices blog)	4
	Pharmacy open 24/7	0
	Curtin University Radio	0
Occupation	Allied health	3 (podiatrist, psychologist and speech therapist)
	University student	5
	Other office-based workforce	9
	Retail	1
	Start-up innovator	4
Highest Education	High school	2
	University	18
	Other	2

doi:10.1371/journal.pone.0156164.t002

(a) Participant Demographics

Type of App	Used by Android Participant Number	Used by iOS Participant Number	Number of Participants
Blood pressure monitoring app (1 type)		P6	1
Diabetes monitoring app (2 types)		P2, P17, P20	3
Migraine management app (2 types)		P5, P8	2
Menstrual cycle monitoring (4 types)	P1, P22	P6, P4	4
Anxiety management app (1 type)		P13	1
Calorie management and weight loss monitoring app (5 types)	P1	P2, P3, P16, P20	5
Celiac disease management app (1 type)	P11		1
Sleep monitoring app (4 types)	P14	P6, P13, P21,	4
Pain management app (2 types)		P8	1
Cycling app (2 types)		P12	1
Fitness App (22 types)	P8, P9, P11, P14, P18, P22	P2, P3, P7, P9, P10, P15, P16, P17, P19, P20, P21	17
Other (saliva analysis kit)		P16	1

doi:10.1371/journal.pone.0156164.t003

(b) Types of Health Apps used by Participants

Figure 2.1: Article Figures

2.1.2 Article 2 - Digital Contact Tracing through the use of NFC on mobile applications as a future viable alternative to QR Code Scanning in the context of Fiji

This article aims to analyse the problems encountered in contact tracing apps and design a solution. The authors propose a Near-Field Communication (NFC) Solution Model as a future viable alternative to QR Code Scanning. Published in 2021 and focused directly on the comparison of NFC Scanning vs QR Scanning, the relevance of this article is significant regarding the background research of the project. The author's findings concerned a lack of usage of the QR functionality due to the QR code failing to scan and users finding the system inconvenient (Jamnadas and Sharma (2021)).

Although NFC is commonly used in payment systems, this project aims to demonstrate the potential of NFC synergy with web-based applications. The results show that NFC Scanning requires less time and less user interaction than QR Scanning (see Figure 2.2). The significance of this is that the user will have "a more timely, convenient, and easier to use option", which should result in greater user engagement with the app, according to Jamnadas and Sharma (2021). These findings align with this project's theory that QR scanning is inconvenient and indicate that NFC scanning is the right technology for future development. This article supports the need for this project as NFC scanning appears neglected despite being a superior method of scanning compared to QR, which is widely used.

The project will develop on research from this paper as it will aim to assess the advantages and

	QR Scanning		NFC Scanning
	careFiji	Sample Contact Tracing App	Sample Contact Tracing App
User Interactions	Unlock Phone, Launch App, Press Scan Button, Move phone to scan	Unlock Phone, Launch App, Press Scan Button, Move phone to scan	Unlock Phone, Move phone to scan
Number of interactions	4	4	2

(a) Number of users interaction for QR or NFC per app

	QR Scanning		NFC Scanning
	careFiji	Sample Contact Tracing App	Sample Contact Tracing App
Approximate average time taken in seconds (based on 8 scans)	8.4175	9.46	4.89625

(b) Average time taken for QR or NFC per app

Figure 2.2: Article Figures

disadvantages of NFC scanning against other fitness apps lacking this functionality. As fitness apps are more regularly used, user feedback will yield greater insight into the effectiveness and efficiency of NFC technology against QR technology.

2.1.3 Article 3 - Visualization and Usability issues involved in building a Fitness-training application for mobile device

This article assesses the current visualisation and usability issues in developing a fitness-training application on mobile devices. This article focuses on the development of front-end usability regarding fitness apps; however, this needs to align with the project's aims, which focuses on all aspects of fitness apps and the introduction of new technologies/techniques when logging workouts. The relevance of this article is significant as it addresses one of the leading research questions; what impacts user engagement of fitness apps?

The author stresses that if the application is not intuitive, users will perceive it as a chore, which can demotivate their entire fitness regime. The literature article summarises "that by considering the normal usability criteria on top of the PACMAD model, the effectiveness of the design approach for mobile fitness apps can be fairly effectively evaluated." Zheng (2015). The project will build upon this article by designing a user-friendly front end using the PACMAD model to ensure the application is convenient and intuitive for users.

Overall, the literature suggests that NFC technology can be used for a variety of fitness applications, from tracking physical activity to providing access to personal fitness data. The literature also suggests that NFC technology can be used to securely and privately access personal fitness data. Additionally, there are potential security and privacy risks associated with using NFC technology for fitness applications.

2.2 Analysing the Market

This section aims to identify potentially overlooked features and establish what aspects are currently being neglected and utilised by alternative fitness apps.

2.2.1 MyFitnessPal

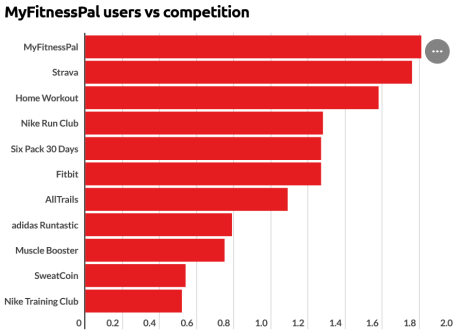
'MyFitnessPal' is a popular web-based exercise and fitness app that helps users keep track of their daily food and drink intake. Nutrients, calories, and vitamins are automatically calculated for the user; this data is then formatted into visual charts and graphs. The 'Workout Routines' feature allows users to track their workouts, including sport-specific exercises and programs covering multiple disciplines.

With a 300% increase in users (50 Million to 200 Million) since 2015 (see Figure 2.1), 'MyFitnessPal' is on an upward trajectory. It would be wise to draw inspiration from 'MyFitnessPal' when developing the project as their continued growth demonstrates that the users welcomed their features.

MyFitnessPal annual users 2015 to 2021 (mm)

Year	Users (mm)
2015	50
2016	75
2017	85
2018	150
2019	180
2020	200

(a) MyFitnessPal Users between 2015-2020



(b) Comparison of Users across Fitness Apps (31/10/2021)

Figure 2.3: Article Figures

'MyFitnessPal' is a free service which could explain why it is so popular; however, the user has to tolerate advertisements and restricted access which can become frustrating. The app features QR scanning, useful for scanning barcodes on food packaging, reducing the amount of manual input by the user.

2.2.2 Strong

'Strong' is a fitness app that gives users an intuitive workout planner. Strong's simple approach stands out as it achieves what it sets out to do. The UI* is sleek, simple, and offers a friendly User Experience that is approachable to anyone. When logging workouts, the app focuses on the essentials, such as exercises, target weight, and target repetition range. Once the user has entered these values, the only input required throughout the rest of the workout is to tick off exercises. Another point to highlight is the app's compatibility, as it features on the Google Play Store (Android) and App Store (IOS).

On greater analysis, Strong presents flaws users may find frustrating. The app excludes many exercises, forcing the user to input the exercises manually. Non-paying users can only access a calendar of when they trained; whilst, paying users can access quality data from their workouts. This accessibility difference may discourage potential users.

2.2.3 Strava

Strava is a fitness app for multi-sport tracking that emphasises community. Users record and upload their workouts which friends can give out kudos to show support. Strava can sync directly with smartwatches and other fitness devices, making the whole experience seamless. One of Strava's strongest features is that users can join various challenges set up by Strava or communities; challenges are a great incentive to use the app and actively log workouts. Strava's unique selling point is its sense of community, which aims to promote personal development and mutual encouragement among its users. Strava also sets a high standard with its UI*, offering a straightforward, aesthetically pleasing design.

On the contrary, some users may not like the community aspect of Strava, and it is not a feature users can turn off. Social media can be a negative place, and users may refrain from posting workouts due to fear and lack of confidence. Safety concerns may arise due to location exposure when using the app.

2.2.4 Analysing the Market Summary

All of these applications offer something unique and are popular for a multitude of reasons. Despite this, they can still be inconvenient for the user, and none offers a solution like NFC

technology. Therefore, this project identifies a gap in the market, making it unique and justifiable.

2.3 Technologies

Researching and evaluating current technologies and frameworks ensures the project is built using the correct tools; it also helps set the scope of the project.

2.3.1 Front-End

Framework	Advantages	Disadvantages
React	• Virtual DOM • Reusable Components • Creating cross-platform products • Quick development • Backward compatibility • Complete SSR/SSG frameworks • React Native	• Poor Documentation • Learning JSX syntax can be difficult
Vue.js	• Can be used for the development of native mobile apps • Easy to learn • Inspired by Angular and React • MVVM* Architecture • Virtual DOM Performance and Rendering	• Lack of scalability • Not as established as Angular and React • Rapidly Evolving & Changing • Issues with iOS & Safari
Angular	• Server-side performance • Ease of prototyping and iterative development • MVVM* Architecture • Maintained by Google	• Heavy framework compared with React & Vue.js • JavaScript support is mandatory • Rapidly Evolving & Changing

2.3.2 Back-End

Framework	Advantages	Disadvantages
Node.js	• Cross-Platform Development • High-Performance for Real-time Apps • Cost effective with Full-stack JavaScript • Scalability for Modern Applications	• A lot of code changes due to Unstable API • Difficulties to Maintain Code
PHP	• PHP-based applications can run on UNIX, Linux, Windows • Open-source and free • Built-in database connection modules • Stable with continuous support • Allows effective storing and retrieval of data	• PHP is not built to handle errors • Open-source leaves the system open to hackers • Hard to maintain and utilise for bigger applications
Firebase	• Supported by Google • Firebase does not require a server • Allows monitoring of crashes • Secure environment of Firebase servers • Faster App Development Speed	• Only NoSQL databases are available • Queries slow down the database performance • Expensive and pricing can be difficult to forecast

2.4 Methodology

The research methods of this project were action research and experimental. The purpose of the research was to identify and extract information apposite to aiding in the development of the project. The research focused on the consensus of health apps, the convenience of using health apps and assessing whether NFC technology was more efficient than QR technology. Sources such as 'Google Scholar' and the 'Goldsmiths Library' were imperative to providing credible information, helping establish a remit for this project. Creating a methodology for the design of this project has enabled appropriate ethical considerations to be embedded into this research process, such as item addictive design, personal data ownership, Weak security and PII* protection. A combination of research, interviews, surveys, and data analysis to collect information and develop insights into the project's research. Also, creative techniques such as brainstorming, storyboarding, and prototyping may generate ideas and solutions.

Chapter 3

Design Specification and Methods

This chapter aims to provide a design specification for a web-based gym app that uses NFC (Near Field Communication) technology. The web app aims to be intuitive and efficient and focuses on NFC technology that provides the user with essential health/fitness data. A clear list of requirements for the project is vital to ensure that all of the aims and objectives are met. Re-assessing the aims and objectives of the project aids in developing a design specification.

Aims:

1. To provide an easy and convenient way for users to track their fitness progress.
2. To provide users with a secure and reliable way to store and access their fitness data.
3. To provide users with a personalised fitness experience based on their individual needs.
4. To promote healthy lifestyle habits and encourage users to stay active.

Objectives:

1. To develop a web-based fitness app that uses NFC technology to efficiently record a workout
2. To integrate the app with smartphones and wearable-technology.
3. To create a user-friendly interface that is intuitive and easy to use.

3.1 Design Specification

3.1.1 Requirements

A web-based fitness app that uses NFC technology needs to be designed with several requirements. These features include tracking and recording user activity and integrating it with existing health and fitness platforms. Furthermore, the app must be able to store and transmit data securely. It should also be able to support various NFC-enabled devices, such as smartphones and wearable devices, and support the latest security protocols to ensure personal data is kept safe. Finally, the app should also have an intuitive, user-friendly interface to make it easy for users to access and interact with the app.

User Requirements

User requirements are the needs and expectations of the end user. The user needs the features, functions, and characteristics to achieve their desired outcome. User requirements include usability, performance, security, scalability, compatibility, and accessibility.

1. The ability to track physical activity and fitness goals using NFC technology.
2. The ability to store and track data from multiple devices with NFC technology.
3. The ability to set and track fitness goals and progress.
4. The ability to create custom workouts and routines.
5. The ability to connect with friends and family to share progress and compete.
6. The ability to view detailed analytics and reports on progress.
7. The ability to sync data with other fitness tracking apps.
8. The ability to access the app on multiple devices.

Requirements Specification

The requirements specification for the project includes robust security protocols, the ability to communicate with both NFC-enabled and non-NFC-enabled devices, a user-friendly interface that allows users to track their fitness goals quickly, and the ability to customise their profile

to track their individual fitness goals. Additionally, the app should be able to save and analyse data to give users insights based on their fitness goals. Furthermore, the app should allow users to access their data from any device securely and should be able to sync with other fitness-tracking devices. Lastly, the app should be compatible with both iOS and Android devices.

The web app needs to be able to do the following:

- Allow users to create an account
- Secure authentication system for users to log in and out of the app.
- Allow users to login and access their personal workout and nutrition profiles
- Allow users to scan gym equipment
- Allow users to log their workout
- Allow users to track their progress

Functional Specification

NFC Technology: The app should be able to utilise NFC technology to track user fitness data. NFC technology should be used to detect and read data from NFC tags embedded in fitness equipment.

User Profiles: The app should allow users to create profiles that store their fitness data. Users should be able to store information such as their age, weight, height, and other relevant fitness data.

Data Collection: The app should be able to collect data from NFC tags and store it in the user's profile. This data should include information such as the type of exercise, duration, intensity, and other relevant data.

Data Analysis: The app should be able to analyse the collected data and provide users with feedback. This should include information such as calories burned, progress over time, and other relevant metrics.

Security: The app should have secure authentication and encryption to protect user data.

3.1.2 System Design

User Interface

The user interface will be sleek, intuitive, and easy to use. The home page should feature a simple, attractive design that displays the user's current fitness level and goals. In addition, the home page should provide easy access to relevant data for the user. The user should also be able to access their fitness profile easily and track their progress. Finally, users should be able to share their fitness achievements on social media easily. The app will also feature a dashboard displaying the user's current status and upcoming activities.

User Experience

The user experience of a web-based fitness app that uses NFC technology will be unparalleled. NFC technology provides an excellent user experience for web-based fitness apps, giving users a more efficient and secure fitness experience. With NFC technology, users can access their data in real-time without requiring manual input. They can use their smartphones to instantly track their progress and results and sync their data with the app. This makes it easier for users to stay motivated and on track with their fitness goals. Furthermore, NFC technology allows for a more secure data transfer, as all information is encrypted. This ensures that users feel safe and secure when using the app. User experience is at the heart of any successful web-based fitness app, and NFC technology is a great way to enhance the user experience. NFC technology makes it easier for users to access the app by simply tapping their phone on the NFC-enabled device. This eliminates the need to enter passwords and makes logging in and out of the app more efficient and intuitive. NFC technology also makes it easier for users to store and manage their fitness data, and with the ability to securely store and share data, users can stay on top of their fitness goals. Additionally, NFC technology allows users to access special promotions and discounts, which can help to motivate them to stay active and reach their fitness goals. The user interface will be designed to be intuitive and user-friendly. Navigation will be clearly labelled and easy to follow. All forms and inputs will be clearly labelled and easy to understand. The app will also be responsive so users can access it from any device.

Features

The project will provide users with the tools and resources they need to reach their fitness goals. It will enable users to set and track their goals, access personalised workout plans, and access health-related data such as body measurements and exercise performance.

Features of the project will include:

- NFC-enabled check-in and out: Users can use their NFC-enabled device to check in and out of the gym.
- View membership information: Users can view their membership information, including their current status, any upcoming activities, and any discounts or promotions.
- Track workout progress: Users can track their workout progress, including the number of workouts completed, calories burned, and any goals they have set.
- Log sets efficiently by tapping their device on an NFC-tagged machine
- Reminders: Users can set reminders for upcoming activities or workouts.

Security

A web-based fitness app must incorporate robust security measures to protect user data and ensure the safety of user information. This should include authentication and authorisation protocols, encryption, and secure data storage. Authentication protocols should require users to create secure passwords, use two-factor authentication, or utilise bio-metric or facial recognition technology. Authorisation protocols should ensure that only authorised users can access the app and that users cannot access data or features they are not authorised to view. Encryption should be used to protect data in transit and storage, and secure data storage should be used to keep stored user data safe from unauthorised access. The app should also include other measures such as monitoring for suspicious activity, logging user activities, and regularly implementing security patches and updates.

- **Data Encryption:** All data stored and transmitted by the app should be encrypted to protect user data from unauthorised access.
- **Authentication:** The app should require users to authenticate their identity before

granting access to any data.

- **Authorisation:** The app should ensure that only authorised users can access specific features or data.
- **Access Control:** Access to data within the app should be restricted to users with the appropriate permission levels.
- **Secure Socket Layer (SSL):** The app should use SSL to ensure secure data transmission across the network.
- **Input Validation:** All user inputs should be validated to protect against malicious attacks such as SQL injection.
- **Auditing:** The app should be audited regularly to ensure that all security measures function correctly.
- **Secure Coding:** All code should be reviewed and tested for security vulnerabilities before being released into production.
- **Security Updates:** The app should be regularly updated with the latest security patches to protect against the latest security threats.

3.1.3 Use Case Diagrams

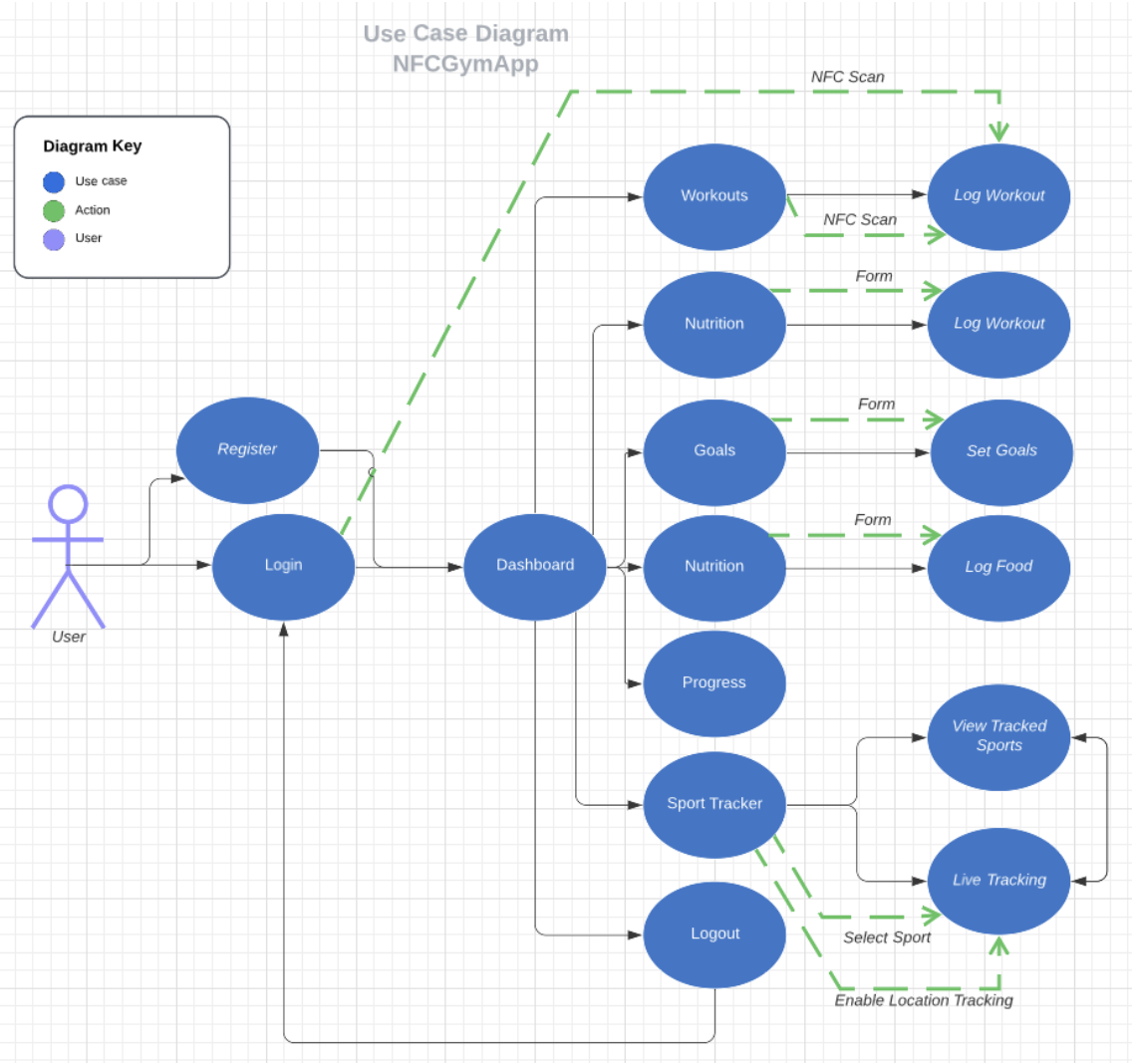


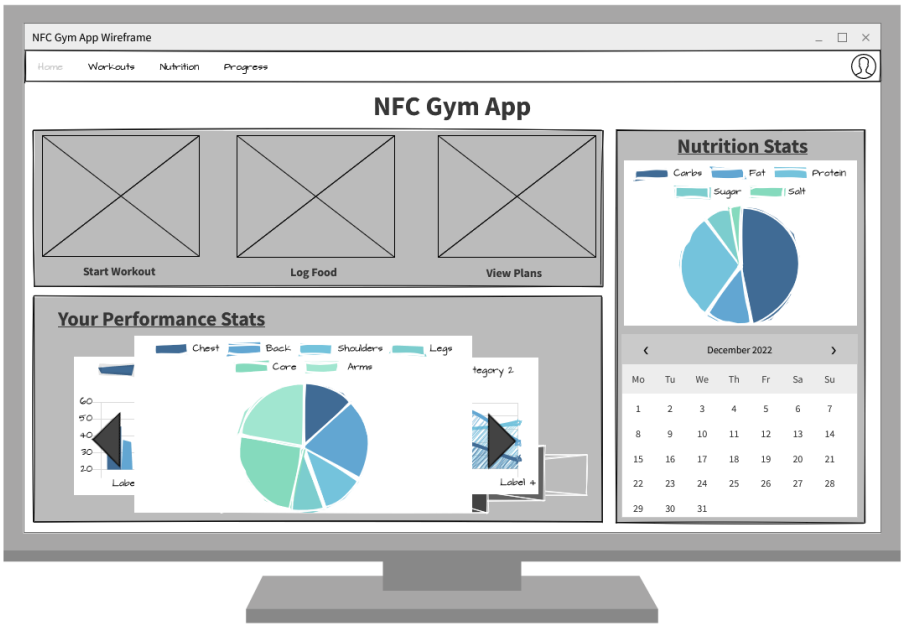
Figure 3.1: Use Case Diagram

3.1.4 Wire Frames

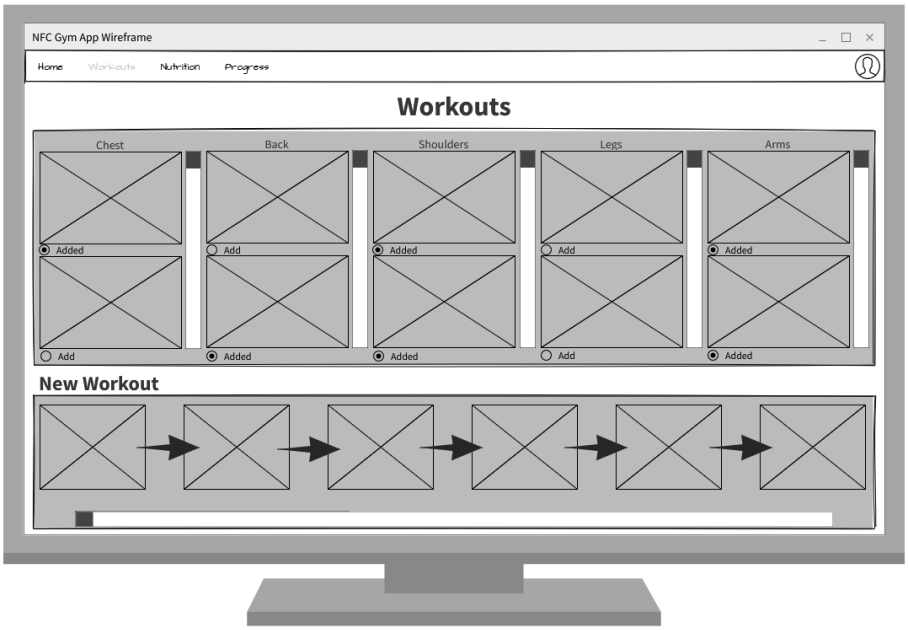
(a) Sign Up Page

(b) Sign In Page

Figure 3.2: Sign In & Sign Up Website Wireframe



(a) Home Page



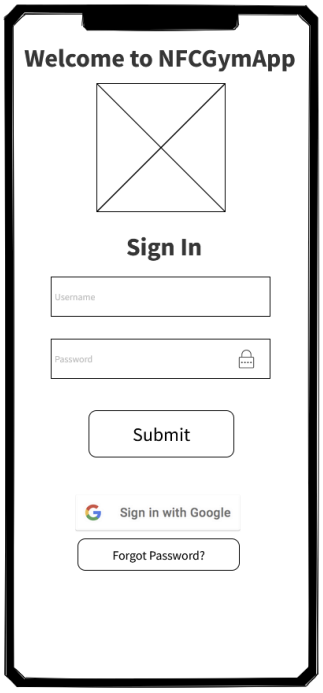
(b) Workout Page

Figure 3.3: Home & Workout Website Wireframe

(a) Nutrition Page

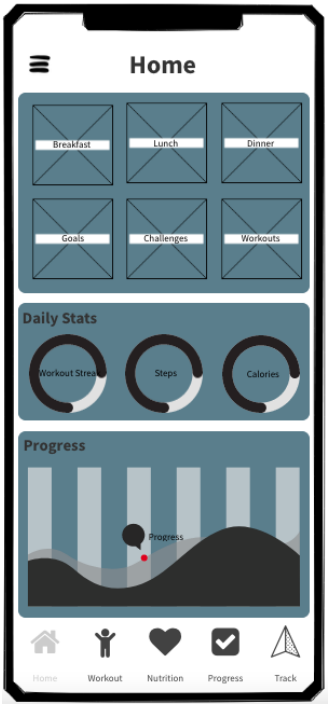
(b) Progress Page

Figure 3.4: Nutrition & Progress Website Wireframe

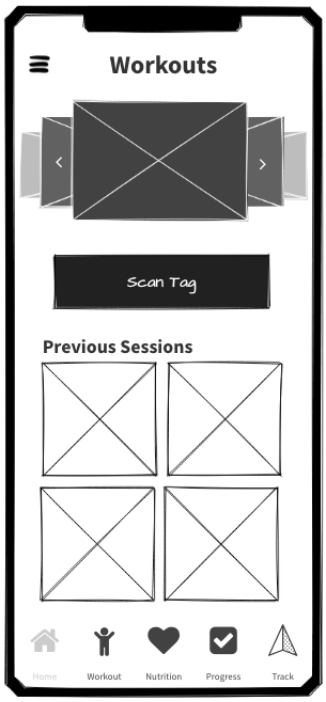


(a) Sign In Page

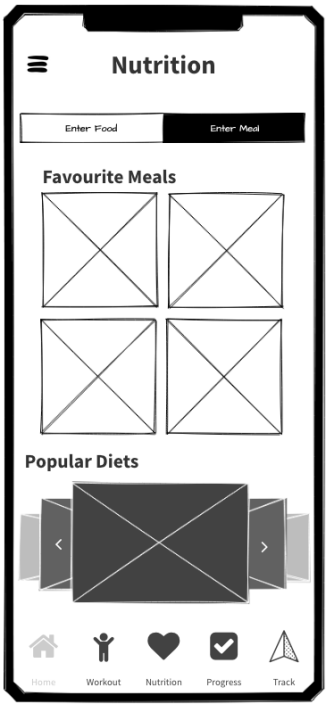
(b) Sign Up Page



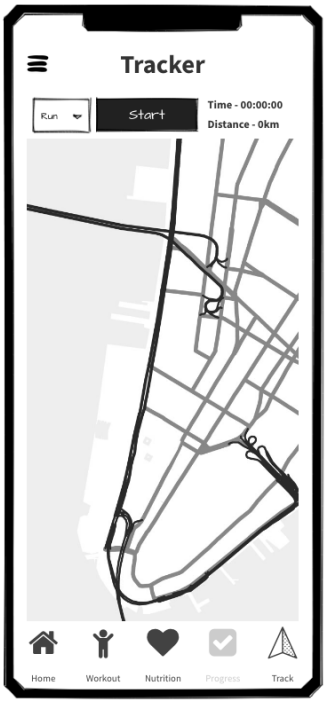
(c) Home Page



(d) Workouts Page



(e) Nutrition Page



(f) Tracker Page

Figure 3.5: App Wireframe

3.1.5 Design Methodology

The web app will be built using HTML, CSS, and JavaScript. It will also leverage a database for user accounts and progress tracking. The app will be organised into distinct sections for each feature, such as a section for tracking progress and interacting with other members. The first step of the design methodology for the project is to create a functional prototype designed to meet the user's needs. This should include features such as tracking a user's exercise, allowing the user to sync their data with other fitness tracking apps, and demonstrate the NFC functionality. After the prototype is created, the app must be thoroughly tested and evaluated to ensure that it functions properly and meets the user's requirements. This includes testing the app's performance, device compatibility, and user experience. The app must be regularly updated and monitored for feedback to ensure that the app continues to meet the user's needs. Once the app is tested and evaluated, it can be released to the public.

Testing

The web app will be thoroughly tested prior to launch. This will include both manual and automated tests, such as functional testing, usability testing, security testing, and performance testing. Once the prototype testing is complete, the app should be tested in a real-use case environment to verify that the app, specifically the NFC functionality, is working correctly. Finally, the app should be deployed to a production environment, such as a web server, and tested in a real-world environment with an NFC reader.

Deployment

The application will be built using the npm run build command. This process will create a production-ready version of the project in the build folder. A server will need to be set up to host the application. Nginx, a web server, will be set up to serve the application and be deployed onto this web server. The server will be configured to serve the application and allow for final tests. Once the application is deployed and tested, the project will be available for users to access. Using a Raspberry Pi to host a web app is possible but limited by the Pi's hardware.

3.2 Ethical Considerations

3.2.1 Project Development

An ethical consideration the project must consider is data privacy and security. The app should have appropriate data protection and privacy policies to ensure that personal data is protected and not misused. Furthermore, ensuring that the app does not violate any user's rights to privacy or any other ethical considerations is crucial. Another ethical consideration for the project is user consent, a clear user consent policy must be in place, which outlines the user's rights and choices regarding the use of their data and the app's data collection and processing activities. In addition, the app should not exploit any of its users. This includes not providing false or misleading information about the app's effectiveness or offering unscientifically backed services.

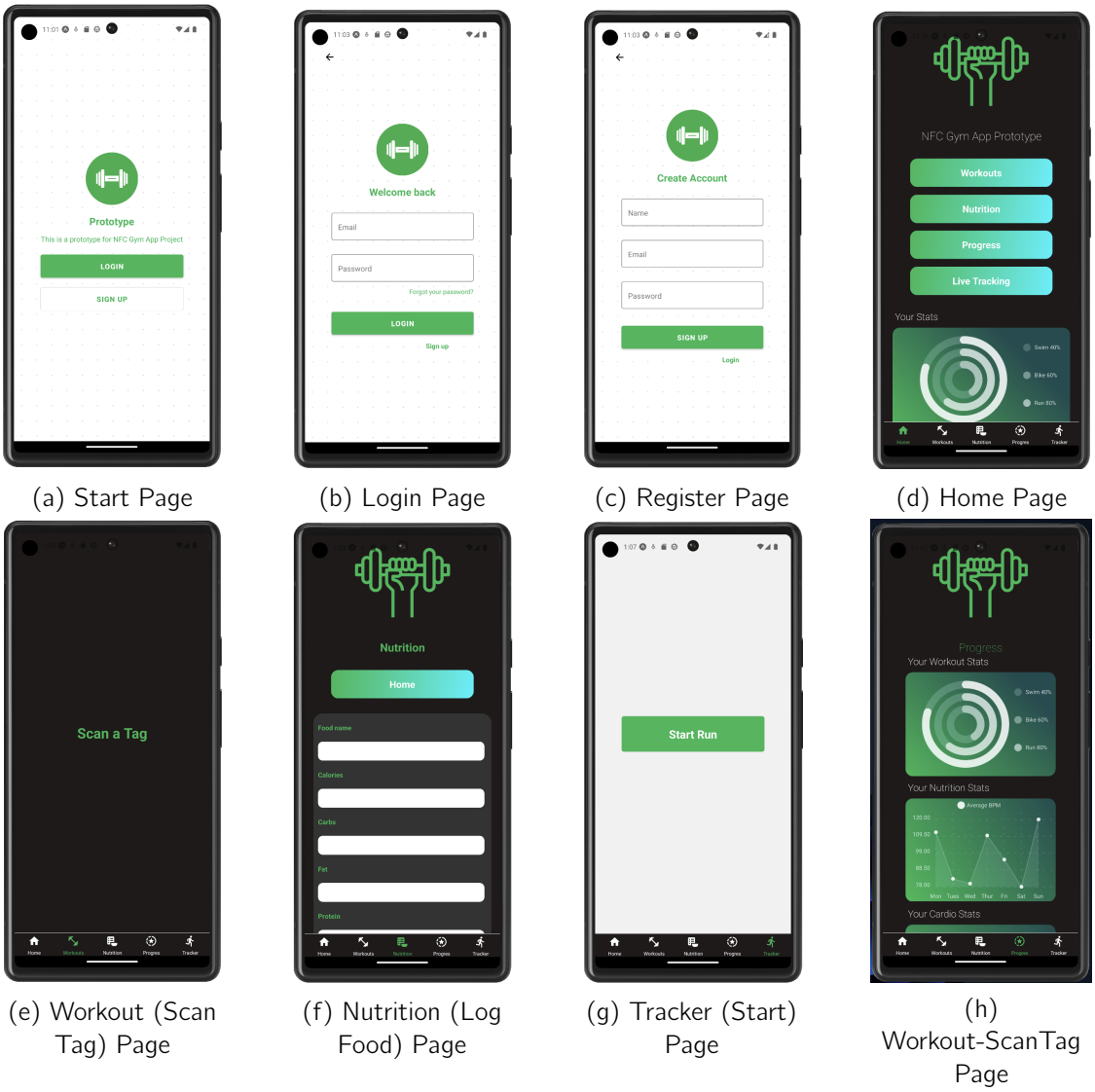
Additionally, the app should not pressure users to purchase products or services in order to use the app. The app must be designed to promote healthy habits, not encourage unhealthy behaviours. It should not offer any advice or guidance that could be potentially dangerous to the user. Furthermore, it should not contain any advertising or links to harmful websites.

3.2.2 Complying with GDPR

The General Data Protection Regulation (GDPR) is a set of laws put in place to protect the privacy of individuals and their personal data. The project will include appropriate security measures, such as encryption and authentication, to protect users' data. Additionally, clear, concise information about how the user's data will be used will be available through a privacy policy or terms of service agreement. Users will be able to update, delete, or export their data as needed. This includes providing the user with the ability to opt-out of data collection and giving them the ability to access, correct, or delete their data. All data-handling processes will be documented, as this will help ensure that the data-handling practices are consistent and comply with the GDPR.

3.3 Prototype Documentation

3.3.1 React Native App Prototype



(a) Start Page (b) Login Page (c) Register Page (d) Home Page

(e) Workout (Scan Tag) Page (f) Nutrition (Log Food) Page (g) Tracker (Start) Page (h) Workout-ScanTag Page

Figure 3.6: Prototype on Android Video Demonstration Here

3.4 Progress Report

This report captures the progress made on the project. The project development is currently at 25% completion. The initial designs and structure have been completed, and the first set of features have been implemented. The prototype demonstrates how users can track their daily fitness activities, log nutrition and view their progress. In addition, the concept of the project has been proven to work, however the development of the project will have to be

built using React Native.

The next step in the development of the fitness app is to begin the implementation stage. The user interface of the app has been designed and the necessary features have been identified. The user interface has been coded using Flutter and the NFC technology has been integrated into the app. The app has been tested on both Android and IOS platforms with the results meeting satisfactory requirements.

3.5 Updated Plan

The plan has been updated with the progress made so far on the project.

1. Research: Conduct continued research on Flutter and NFC technology and their application to the development of the fitness app.
2. User Interface Design: Create a user interface that is visually appealing, easy to use and provides the necessary features.
3. Database Design: Design the database structure that will store the user information, workout data, and other fitness information.
4. User Authentication: Set up user authentication within the app allowing users to sign up and login
5. Development: Begin to pull user data from the database into the app so that the users can see their fitness data
6. Testing: Test the application for errors and bugs.
7. Deployment: Deploy the application to the appropriate app stores.
8. Maintenance: Monitor and maintain the application to ensure it is running smoothly. Update the application as needed.

3.5.1 Reassessing the Projects Scope

After encountering challenges during the prototyping phase of the project, it became clear that the initial scope of the project was too vast. In order to meet the timescale of the

project, the need to break down the project into smaller and more manageable components, as well as reevaluate the goals and objectives of the project was essential.

What features are essential to the project and what features would be nice additions given the time. Understanding the goals of the project gives greater insight into what wold be considered essential for the project to be a success. Going into finer detail by breaking down the project into smaller sub-projects can aid in focusing the projects scope to meet the time-frame. What specific tasks and features need to be completed or implemented in order to create this fitness app? The user interface, the back-end development, and the integration of any third-party services that may be needed.

Updated Scope

1. Objectives - The minimum goal of this project is to create a fitness mobile application using Flutter that allows users to track and monitor their fitness activities and progress, implementing NFC technology to improve user experience.
2. Features -
    - Login/Registration
    - Activity Tracking
    - Data Visualisation
    - Customisable Workouts
    - NFC Functionality
    - Fitness Challenges (Additional)
    - Social Interaction (Additional)
    - Reminder Alerts (Additional)
    - Meal Planning (Additional)
    - Nutrition Tracking (Additional)
3. Requirements -

- Design and develop a mobile fitness application using Flutter
- Develop a user-friendly and intuitive interface
- Integrate fitness tracking features and data visualisation
- Integrate NFC features allowing users to quickly log workouts
- Develop features for fitness challenges and social interaction (Additional)
- Integrate customisable workouts and meal planning (Additional)
- Implement reminder alerts and nutrition tracking (Additional)

4. Risks -
    - Technical difficulties
    - Learning curve
    - Unforeseen issues with the development process
    - User acceptance

3.5.2 Updated Timeline

Week 1 - January 16th:

- Set up development environment
- Set up Trello

Week 2 - January 23rd:

- Implement user authentication using Firebase Auth
- Set up Firestore to store user credentials
- Create a basic user interface

Week 3 - January 30th:

- Implement logging of workouts with Firestore
- Set up Firestore to store workout data

- Test user authentication

Week 4 - February 6th:

- Implement tracking of runs
- Set up Firestore to store run data
- Test logging of workouts

Week 5 - February 13th:

- Implement NFC scanning of machines
- Set up Firestore to store NFC data
- Test tracking of runs

Week 6 - February 20th:

- Implement tracking of gym visits
- Set up Firestore to store gym data
- Test NFC scanning of machines

Week 7 - February 27th:

- Test tracking of gym visits
- Finalise the project plan
- Finalise the user interface

Week 8 - March 6th:

- Debug and test all features
- Set up a testing environment
- Test user authentication

Week 9 - March 13th:

- Debug user interface
- Test logging of workouts

- Test tracking of runs

Week 10 - March 20th:

- Debug tracking of gym visits
- Test NFC scanning of machines
- Finalise user interface

Week 11 - March 24th:

- Final test of all features

3.5.3 Adapting the project to development challenges

React had been initially decided upon due to my understanding of the technology and it's offering of a reliable and powerful platform for the development of web applications.

However, as the project progressed, several challenges began to hinder the progress. Unfortunately, React was not able to provide the necessary solutions to these issues. After researching potential solutions, it was clear that a switch to a native tool such as Flutter or React-Native was the best way forward.

Flutter or React Native?

Flutter and React Native are two popular mobile app development frameworks used to build native applications.

Flutter is a newer mobile app development framework created by Google. It uses the Dart programming language and features a reactive programming model, allowing for faster development and better performance. Flutter is also optimised for building user interfaces that are both beautiful and performant. Flutter also supports the use of NFC technology and is able to take advantage of its features within an app.

React Native, on the other hand, is a JavaScript-based framework developed by Facebook and is the most popular option for mobile app development. It is a cross-platform framework that allows developers to build applications for both iOS and Android platforms. React Native also supports the use of NFC technology, but the setup is more complex than with Flutter.

When comparing the two frameworks for developing the project, Flutter has several advantages. It is faster and easier to develop with, has better performance, and is optimised for building beautiful user interfaces. Additionally, it is easier to set up NFC technology with Flutter than with React Native. Adapting the project to resolve the challenges faced provided a greater opportunity to achieve the expectations and requirements of the project.

Chapter 4

Implementation

4.1 Project Management Tools

4.1.1 Trello

Trello played a crucial role in managing the development of the FitSmart App (see Fig 4.1). The intuitive drag-and-drop feature provided a flexible and efficient way to prioritise tasks based on their status, ensuring that high-priority items received the necessary attention and that the development process remained well-organised from start to finish. In addition, Trello's visual interface and customisable board settings allowed for a high level of flexibility in adapting to the needs of the development process. I could quickly and easily adjust settings to reflect changes in project scope, timeline, or priority.

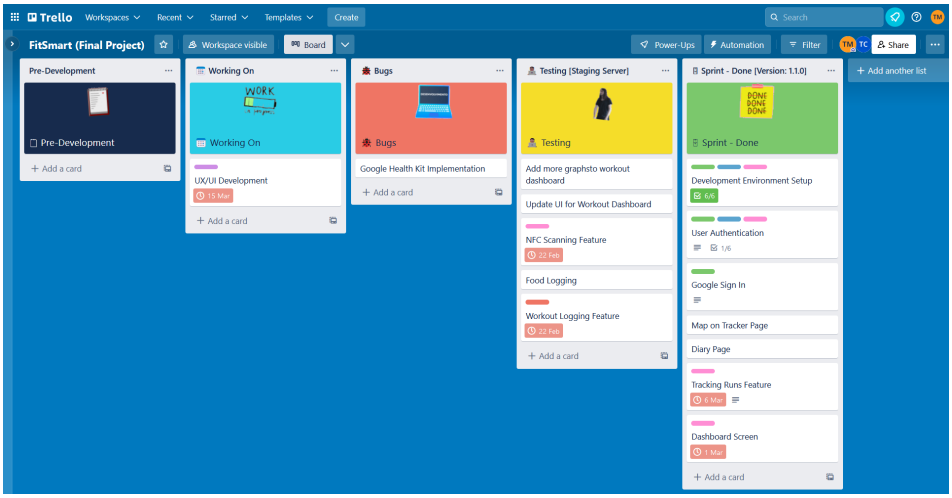


Figure 4.1: FitSmart Trello Board

4.1.2 Github

GitHub is a website that offers collaboration and version control solutions for software developers. Github was utilised in the context of the project to keep track of codebase modifications and facilitate concurrent iOS and Android platform development. For various platforms or features, distinct branches can be established on Github that can later be merged back into the main core. By doing so, the project can be developed for many platforms while keeping managing a single codebase for the project.

4.2 Setting up the Development Environment

The first step in the development process was to set up the Flutter environment. This required the installation of several components, including the Flutter SDK, an integrated development environment (IDE), and the Android SDK and its dependencies. The Flutter SDK was downloaded and installed by visiting the Flutter website and downloading the appropriate version of the SDK for the desired platform. Next, Android Studio was installed from the Android Developers website, as this is the recommended IDE for Flutter. Finally, the Android SDK and its dependencies needed to be installed from the Android SDK page on the Android Developer website.

4.2.1 Setting up Firebase

Firebase is essential to the project since it offers a safe and simple authentication process for users to download and log in to the app. Additionally, it enables developers to store user data using Firestore, a real-time NoSQL document database that syncs with all devices simultaneously. Firebase Analytics gives developers insights into user behaviour and app usage, enabling them to monitor user engagement and enhance the app's functionality.

Setting up Firebase Authentication and Firestore for the app involved a few steps. A Firebase account and a project were created through the Firebase website. Once the project had been created, Firebase needed to be added to the app by generating a configuration file containing the app's unique credentials. After the configuration file was generated, the app was registered with Firebase.

Firebase Authentication and Firestore had to be enabled within the project. This was done by selecting the Authentication and Firestore options from the navigation menu in the Firebase console (See Fig 4.2). After these features had been enabled, the appropriate dependencies needed to be added to the app's pubspec.yaml file. Once the dependencies had been added to the pubspec.yaml file, the app needed to be connected to Firebase. To do this, an instance of the FirebaseApp class must be created and initialised with the configurations provided in the configuration file. The FirebaseApp instance could now be used to obtain a FirebaseAuth instance which can be used to authenticate users of the app.

Finally, an instance of the Firestore class was created. This instance allowed for access to the Firestore database which could now be used to create, read, update, and delete data from your Firestore database.

4.2.2 Challenges with setting up the development environment

Developing apps for iOS requires a Mac computer as the iOS SDK and Xcode are essential tools for developing and debugging apps. They are only available on the Mac App Store, making them difficult to obtain. Fortunately, I had access to Mac equipment through Goldsmiths' facilities and by being lent a friend's older Macbook Air throughout the development of the project. Also, Provisioning profiles are required to deploy apps on iOS devices for testing and distribution. These profiles must be created and managed in the Apple Developer Portal, which some may find to be a cumbersome process.

Furthermore, when developing for Android, there are additional challenges that need to be addressed, such as configuring environment variables and path settings, setting up an Android emulator, or connecting a physical device to your computer if one is available. Additionally, the Android Studio project must be configured to use the Flutter SDK.

4.3 User Authentication

Firebase Authentication allows users to sign in to an application by providing their email address and password, or by using one of the many popular third-party identity providers, such as Google, Facebook, or Apple. It provides secure authentication, user management,

and other authentication-related services for applications.

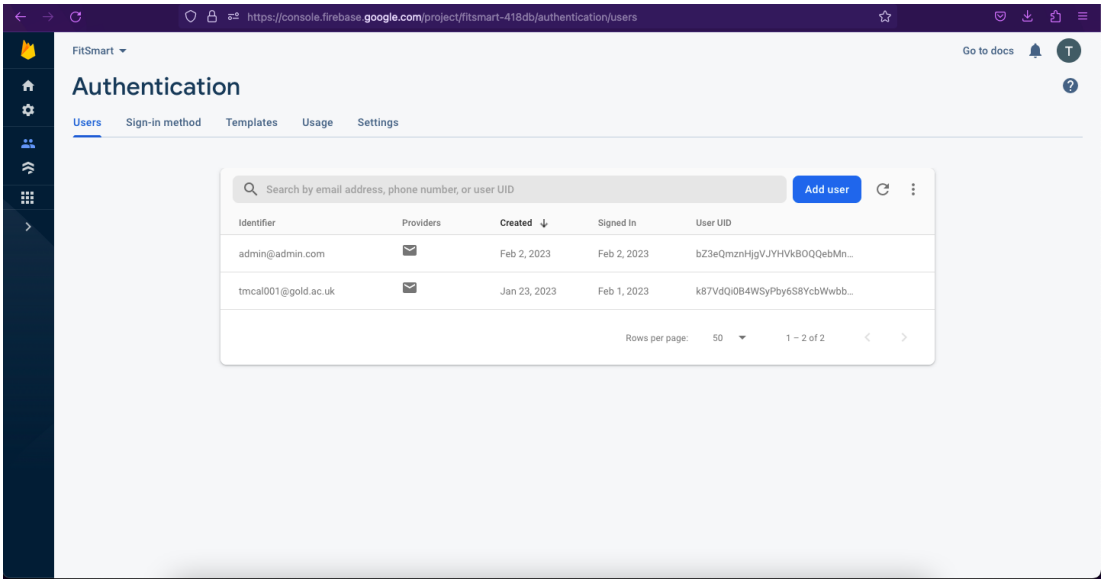
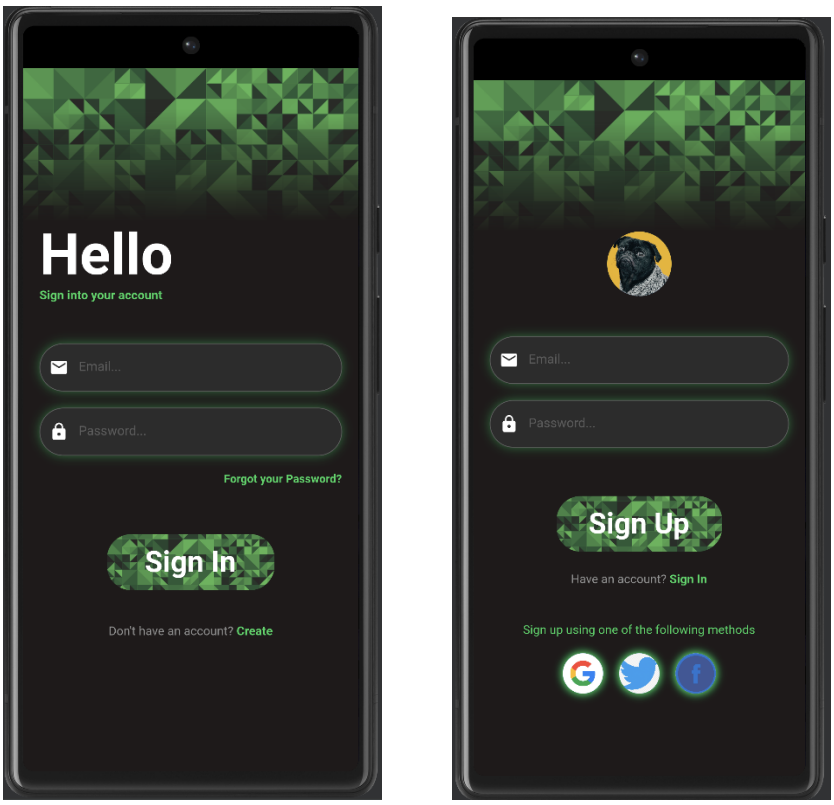


Figure 4.2: Firebase Console

The Firebase Authentication API provided a set of methods and classes that make it easy to build authentication flows into the app. The API also provided methods for managing existing user accounts, such as updating user information or deleting user accounts. In addition, the Firebase Authentication API provides a set of classes for managing users' authentication states. Also, the API provides a set of security measures to protect user data, such as, implementing two-factor authentication and encrypting user data. The UI for the Auth screens can be seen in Fig 4.3.



(a) Sign In Page (b) Sign Up Page

Figure 4.3: Auth Screens

When a user authenticates their identity in Firebase, the SDK securely stores the user's credentials and generates an authentication token. This token is then used to access the Firebase Database, and the user is granted access to the resources they need to perform the requested action. The Firebase Authentication SDK also includes features such as password reset and email verification, which allows users to securely reset their passwords and verify their identity when needed.

The full code can be found within the 'auth' folder on Github here

Register Code

```
void register(String email, password)async{
  try{
    await auth.createUserWithEmailAndPassword(email: email, password: password);
  }catch(e){
    Get.snackbar("About User", "User Message",
    backgroundColor: Colors.redAccent,
    snackPosition: SnackPosition.BOTTOM,
    titleText: const Text(
      "Account creation failed",
      style: TextStyle(
        color: Colors.white
      ),
    ),
  }
}
```

Sign In Code

```
void signIn(String email, password)async{
  try{
    await auth.signInWithEmailAndPassword(email: email, password: password);
  }catch(e){
    Get.snackbar("About Sign In", "Sign In Message",
    backgroundColor: Colors.redAccent,
    snackPosition: SnackPosition.BOTTOM,
    titleText: const Text(
      "Login failed",
      style: TextStyle(
        color: Colors.white
      ),
    ),
  }
}
```

4.4 FitSmart App Structure

4.4.1 Folder Structure

The app's file structure is organised as follows:

- db: This folder contains code connecting the app to the Firestore Database.
- models: This folder contains data models used throughout the app for working with data from the Firestore database or API.
- pages: This folder contains all the pages of the app, organised into sub-folders by feature. Each sub folder contains sub-folders for widgets specific to that page. For example, the auth folder contains a sign-in page and a sign-up page, while the dashboard folder contains the main dashboard page and sub-folders for widgets such as the graph widget etc.
- utils: This folder contains utility functions for handling dates or formatting text and generating a UUID (universally unique identifier) for database entries.
- widgets: This folder contains reusable widgets that are used across multiple pages/features of the app. The bottom navigation is located here as it remains the same throughout the app.

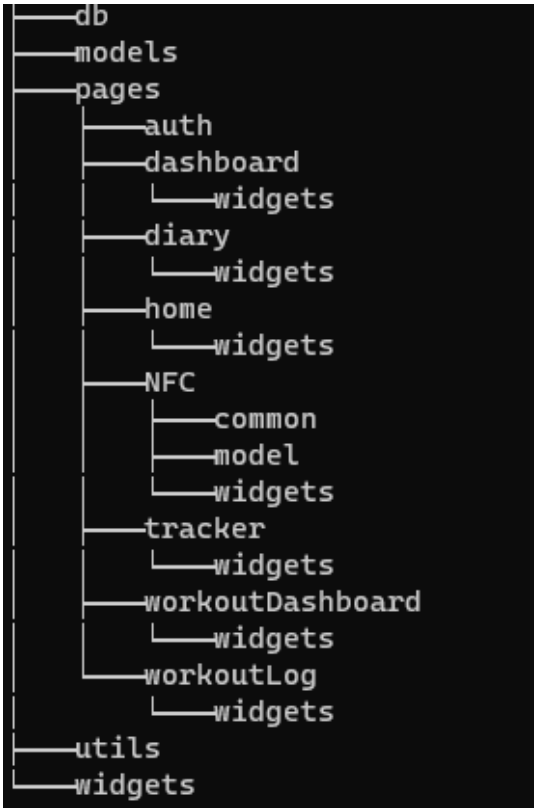


Figure 4.4: Project Directory

4.4.2 Database Structure

The database structure of FitSmart is designed to store data related to exercises, food tracking, and GPS tracking. Firestore is used as the database, which is a cloud-based NoSQL database offered by Firebase.

There are three main collections in FitSmart: gpsTracker, foodTracker, and exercises. Each collection contains documents that correspond to individual entries in the app.

GPS Tracker Collection

The gpsTracker collection stores information about the user's GPS-tracked activities, such as running or cycling. Each document in the gpsTracker collection (see Fig 4.4) contains the following fields:

- date: The date on which the activity was performed.
- distance: The distance covered during the activity.
- duration: The duration of the activity.
- speed: The average speed during the activity.

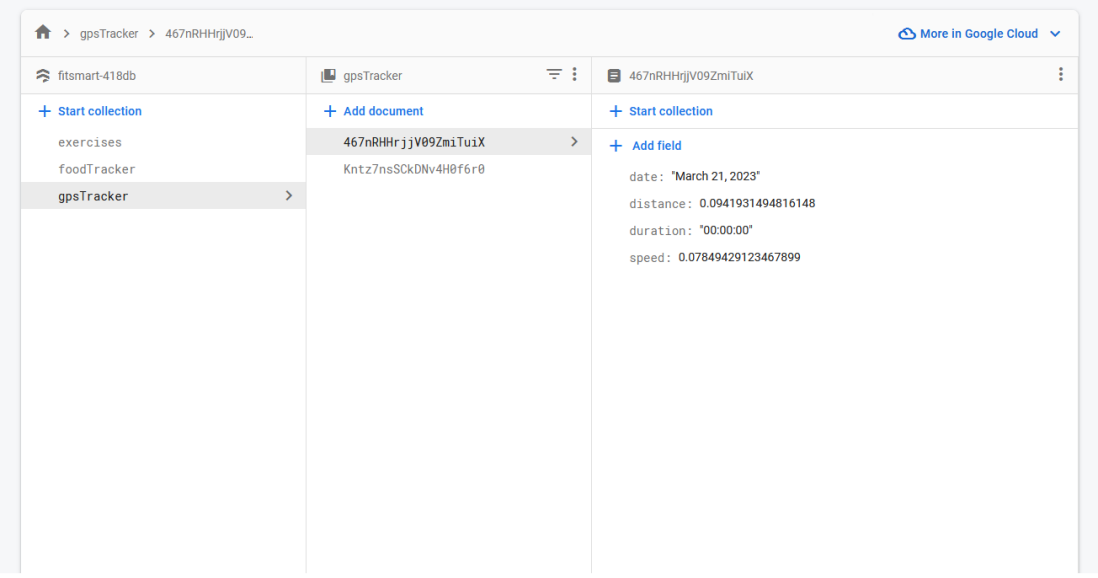


Figure 4.5: Firestore - GPS Tracker Database Structure

Food Tracker Collection

The foodTracker collection stores information about the food that a user consumes. Each document in the foodTracker collection (see Fig 4.5) contains the following fields:

- calories: The number of calories consumed.
- carbs: The number of carbohydrates consumed.
- createdOn: The timestamp of when the entry was created.
- fat: The number of grams of fat consumed.
- food_name: The name of the food consumed.
- grams: The total weight of the food consumed in grams.
- protein: The number of grams of protein consumed.

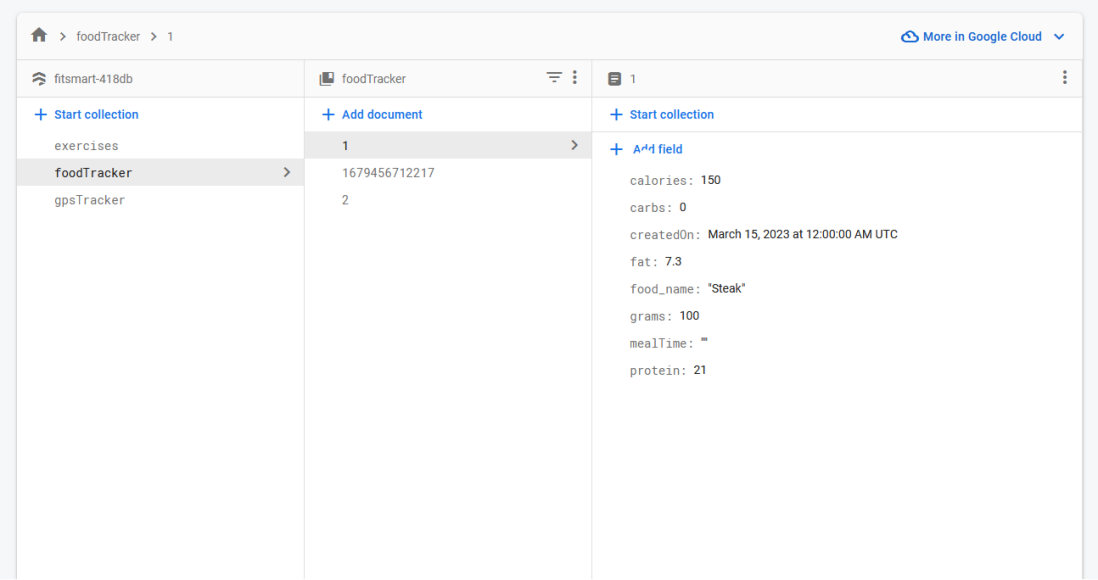


Figure 4.6: Firestore - Food Tracker Database Structure

Exercises Collection

The exercises collection stores information about the exercises that a user performs. Each document in the exercises collection (see Fig 4.6) contains the following fields:

- name: The name of the exercise.
- timestamp: The timestamp of when the exercise was performed.
- sets: An array of sets, each of which contains the number of reps and weight lifted.

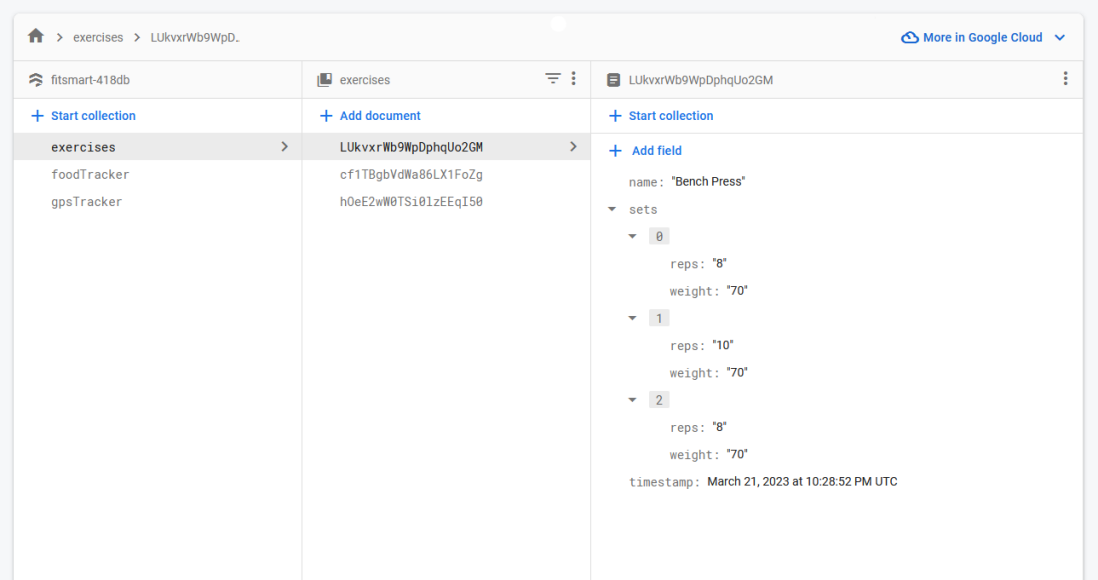


Figure 4.7: Firestore - Exercise Database Structure

In summary, the database structure is designed to store and organise the different types of data that are collected by the app. This structure enables efficient storage, retrieval, and analysis of data, enabling users to track their fitness goals more effectively.

main.dart

The 'main.dart' is the main entry point for any Flutter app. It is the first file that is executed when the app is launched. Within the FitSmart App, the main function is an asynchronous function that ensures Flutter is initialised, initialises Firebase, and sets up the app's orientation and UI mode. Additionally, it creates an instance of the 'AuthController', which is used for authentication-related functionality within the app.

```
Future<void> main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp().then((value) => Get.put(AuthController()));
SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersive);
SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown
]);
  runApp(const MyApp());
}
```

The structure and theme of the app are defined by the 'MyApp' class, which is a StatelessWidget. It uses the GetMaterialApp package to conveniently manage the app's state and configure its title, theme, routes, and initial route. The app's various pages are mapped to strings in the routes, with each page being defined as an object. The available pages include the home page, sign-in page, sign-up page, dashboard, workout tracker, workout log, NFC scanner page, and diary.

To render the app's pages and create its basic material design structure, the 'MyApp' class's build method returns a MaterialApp widget. This widget uses the routes to navigate to the appropriate page widgets when requested by the user.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'FitSmart',
      theme: ThemeData(
        scaffoldBackgroundColor: const Color.fromRGBO(40, 40, 40, 1),
        fontFamily: 'Roboto',
        textTheme: const TextTheme(displayLarge: TextStyle(
          fontSize: 25,
          color: Color.fromRGBO(255, 255, 255, 1),
          fontWeight: FontWeight.w900
        )
        )
      ),
      debugShowCheckedModeBanner: false,
      routes: {
        '/': (context) => const HomePage(),
```

```
        '/signin':(context) => const SignInPage(),
        '/signup':(context) => const SignUpPage(),
        '/dashboard':(context) => const DashboardPage(),
        '/tracker':(context) => const TrackerPage(),
        '/workoutDashboard':(context) => const WorkoutDashboard(),
        '/workoutlog':(context) => const LogWorkoutPage(),
        '/tag_read':(context) => const NFCScanPage(),
        '/diary':(context) => const DiaryScreen(),
      },
      initialRoute: '/signin',
    );
  }
}
```

pubspec.yaml

The pubspec.yaml file is a configuration file used in Flutter projects to specify the project's metadata, dependencies, and other settings.

In the 'FitSmart' pubspec.yaml file, the project's name is 'FitSmart', and includes a short description. The version field specifies the version number of the package, and it includes a build number, which is useful for continuous integration and deployment workflows. The environment field specifies the minimum Dart SDK version required to run the package.

The dependencies field includes a list of packages that this project depends on, along with their version numbers. These packages include Flutter itself and various third-party packages used in the project. The dependencies used is this project are as follows:

```
dependencies:
  flutter:
    sdk: flutter
  intl: ^0.18.0
  google_maps_flutter: ^2.2.4
  get: ^4.6.5
  firebase_auth: ^4.2.2
  firebase_core: ^2.4.1
  google_sign_in: ^6.0.1
  cloud_firestore: ^4.4.2
  nfc_manager: ^3.2.0
  provider: ^6.0.5
  syncfusion_flutter_charts: ^20.4.52
  geolocator: ^9.0.2
```

```
  stop_watch_timer: ^2.0.0
  google_fonts: ^4.0.3
  sqflite: ^2.2.5
  health: ^4.5.0
  path: ^1.8.2
  firebase_database: ^10.0.15
  fl_chart: ^0.61.0
  url_launcher: ^6.1.10
  json_annotation: ^4.8.0
  permission: ^0.1.7
  flutter_launcher_icons: ^0.12.0
```

The 'dev_dependencies' field lists packages that are only used during development and not required for the project to run. The 'dev_dependencies' used is this project are as follows:

```
dev_dependencies:
  dependency_validator: ^3.0.0
  flutter_launcher_icons: "^0.12.0"
  test: ^1.22.0

flutter_icons:
    android: "launcher_icon"
    ios: true
    image_path: "assets/fitsmartIcon.png"

flutter_test:
    sdk: flutter

flutter_lints: ^2.0.0
```

4.5 Workout Log

The workout log was a valuable feature in the development of FitSmart, allowing users to track their progress and intuitively log workouts. The NFC scanning feature was designed to make logging workouts quick and easy, and the validation checks ensured data accuracy and completeness. The progress tracking features aimed to help users stay motivated and committed to their goals.

To build the workout log feature, the data to be collected needed to be defined. This included the date and time of the workout, the exercises performed, the number of sets and reps and the amount of weight lifted. The user interface was designed to be intuitive and easy to use, so gym members could quickly log their workouts using the NFC scanning feature.

Retrieving & Displaying Data from Firestore

To display a user's workout data, the app queries the Firestore API using the user's ID as a parameter. The retrieved data, including workout history, date, type, and metrics, is then shown in a ListView Widget (see Fig 4.7 (b)). Firestore listeners are used to keep the data up-to-date by automatically updating the app when changes occur.

The 'ViewWorkoutPage' widget is a StatefulWidget that uses a StreamBuilder child to listen for changes in the Firestore collection of exercises. If there's available data, it groups the workouts by day and displays them in a ListView. Otherwise, a CircularProgressIndicator is shown.

Firestore's default instance and the exercises collection are accessed using the code:

```
1 FirebaseFirestore.instance.collection('exercises').
```

The 'groupByDay' method groups the workouts by their completion date and returns a Map with DateTime keys and lists of DocumentSnapshots as values.

The 'buildDayTile' method creates a Card widget for a specific day with a list of workouts. Each workout is displayed using the 'buildWorkoutTile' method, which shows its name, total sets, and individual set details.

Tapping on a day tile navigates to a new page that displays the workouts for that day in a vertical list, using the 'onDayTileTap' method.

The full code can be found within the 'workoutLog' folder on Github here

NFC Feature

The 'TagReadModel' class defines a model that stores information about the NFC tags read by the app, including the tag itself and any additional data associated with the tag. The 'handleTag' function is called when the app detects an NFC tag, and it adds the tag to the tags list and any additional data to the additionalData map.

The 'TagReadPage' class defines a widget that displays the app's user interface (see Fig 4.7 (c)). It creates a ChangeNotifierProvider to provide the 'TagReadModel' to its child widgets. The user interface includes a button that starts an NFC session and displays the information about the tag that was read. The user interface also includes an ElevatedButton that navigates the user to another page with a workout log.

The '_TagInfo' class defines a widget that displays the information about an NFC tag on the app's user interface. The widget utilises the 'Ndef.from' method from the nfc_manager package to extract the NDEF record from the tag, provided that it exists. Afterwards, the widget generates a list of FormRow widgets to exhibit details regarding each record present in the NDEF message.

The full code can be found within the 'NFC' folder on Github here

Logging & Sending Data to Firestore

The user input data is collected in the app using various input widgets such as text fields, dropdown menus, and sliders. The inputs are then formatted into the appropriate data structure such as a map or a class object that matches the data model of Firestore documents. The formatted data is then sent to Firestore using the Firestore API. This involves creating a new document in the appropriate collection, or updating an existing document if it already exists. Using the appropriate Firestore API methods, such as 'set()' or 'update()', the data is then written to Firestore. If any errors occur during the write operation, appropriate error

handling has been implemented to alert the user and handle the error gracefully.

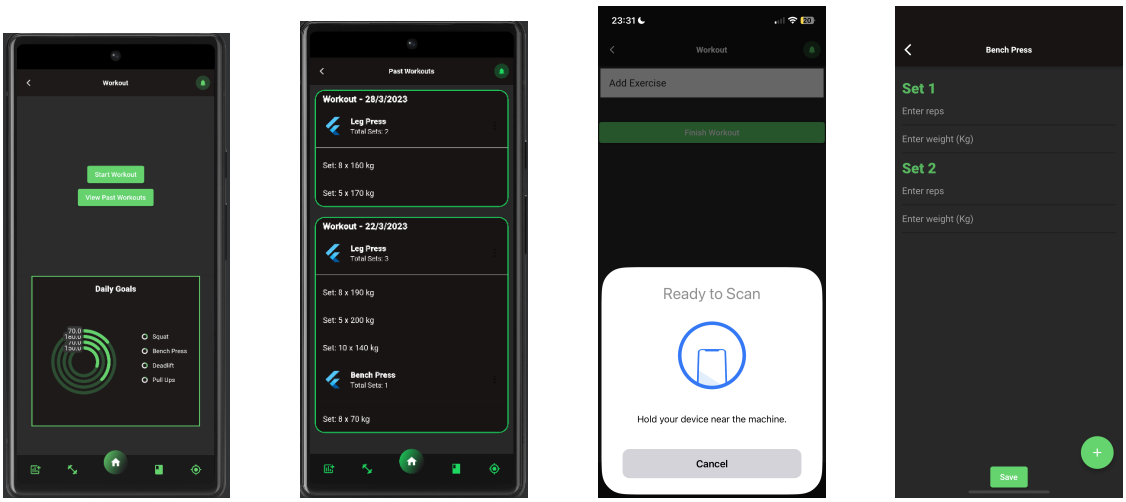
The '_NdefRecordPageState' class is a StatefulWidget that holds the state of the page. The page displays the name of the exercise, a list of sets, and a button to save the exercise data to Firestore (see Fig 4.7 (d)).

The build method of the '_NdefRecordPageState' widget constructs the user interface of the page using a Scaffold widget. The title of the page is set to the name of the exercise, which is extracted from the 'NdefRecord' object passed to the widget. The body of the page is a Column widget that contains a list of sets represented by a 'ListView.builder' widget. Each set is displayed using two TextFields that allow the user to enter the number of reps and weight for the set.

The '_saveExercise' function is called when the user presses the 'Save' Button. It creates a new document in the 'exercises' collection in Firestore and saves the exercise data entered by the user. If the operation is successful, a success message is displayed using a SnackBar widget. If an error occurs, an error message is displayed instead.

The 'NdefRecordInfo' class is a utility class that converts an 'NdefRecord' object to a human-readable format. The 'fromNdef' method of this class extracts the exercise name and other details from the 'NdefRecord' object and returns a new 'NdefRecordInfo' object that contains this information.

The full code can be found within the 'ndef_record.dart' file on Github here



(a) Dashboard/Menu Screen

(b) Past Workouts Screen

(c) Start Workout Screen

(d) Workout Log Entry Screen

Figure 4.8: Workout Log UI

4.6 GPS Tracker

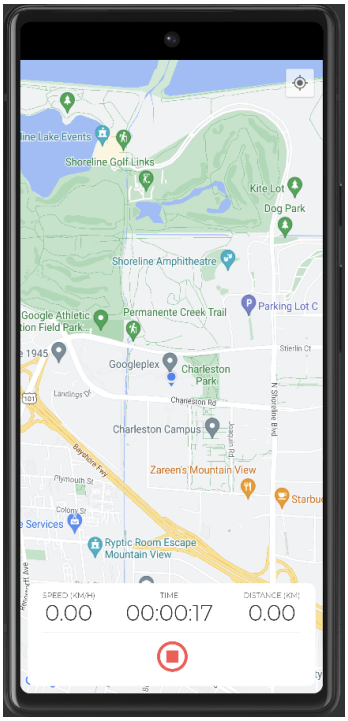


Figure 4.9: Map Page

The GPS Tracker works by displaying a map with the user's current location, tracks their route using GPS, and displays real-time information such as their speed, distance, and time spent exercising. When the user stops the exercise, the application saves the exercise's details, such as date, duration, speed, and distance, and returns to the previous screen where they can see logged GPS based exercises.

Map Widget & Geolocator

The MapPage (see Fig 4.8) widget uses the geolocator package to retrieve the user's location data, and the google_maps_flutter package to display a map with a polyline tracking the user's route. The geolocator package uses the device's GPS hardware to retrieve the user's location data, such as their latitude and longitude coordinates. The google_maps_flutter package uses the Google Maps API to display a map with a polyline tracking the user's route. The polyline is created by connecting a series of latitude and longitude points, which are retrieved from the geolocator package at regular intervals. The widget also includes a timer using the stop_watch_timer package to track the time elapsed, and calculates the user's speed and distance travelled based on their location data. When the user presses the stop button, the widget creates a 'TrackerEntry' object and navigates back to the previous page with the entry as a result.

EntryCard Widget

The 'EntryCard' widget displays a card containing some information about a 'TrackerEntry' object. The 'EntryCard' takes an entry object as a required parameter and displays the date, distance, duration, and speed of the entry. The card has a dark background colour, a green border, and rounded corners.

The full code can be found within the 'tracker' folder on Github here

4.7 Food Diary

The Diary Screen (see Fig 4.9) is a widget that lets users track their food intake. It manages it's own state and interacts with a database for food tracking.

Figure 4.10: Diary Page

Within the 'initState' function, 'DiaryScreen' creates a 'FoodTrackTask' object to hold details of a food item, and the 'resetFoodTrack' function sets this object's values to default. The '_calorieCounter' function displays stats on calorie intake for a given date, while '_addFoodButton' displays an icon button that lets the user add a new food item. '_selectDate' allows the user to pick a new date and '_stateSetter' checks if the selected date is within one day of the current date.

When the user clicks the 'Add Food' Button, the function first checks if the input data is valid. If it is, the function sets the 'createdOn' field of the food item to the current date and time, and adds the food item to the Firestore database using the 'databaseService.addFoodTrackEntry' method. Finally, it calls 'resetFoodTrack' to clear the form. If the input data is invalid, an error message is shown with the ScaffoldMessenger widget.

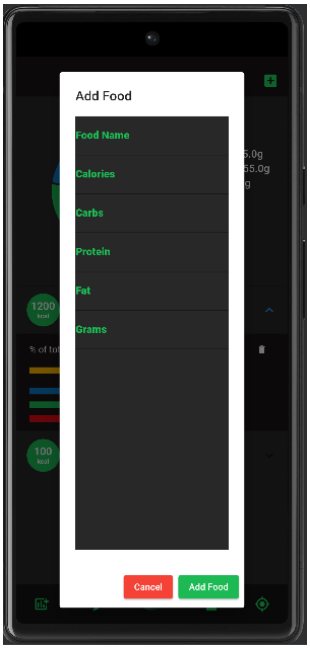


Figure 4.11: Food Entry Modal

Calorie Stats Widget

The Calorie Stats widget displays a pie chart (see Fig 4.9) of the macro-nutrient breakdown (fat, carbohydrates, and protein) of food items tracked by a fitness app. The pie chart is based on the data from the current date or the date selected by the user. The widget calculates the total calories, carbohydrates, fat, and protein of the food items and then displays the percentage of each macro-nutrient as a pie chart section. It also displays the total amount of each macro-nutrient in grams below the chart.

The full code can be found within the 'diary' folder on Github here

4.8 Health Data

Figure 4.12: Dashboard Page

To access a user's health data, the Flutter Health package needed to be installed. This package provides a platform-agnostic API for accessing health data, allowing developers to access data like step count, distance, and heart rate on both Android and iOS platforms. It supports features like reading and writing health data, as well as listening for changes. It can also be used to access data from various sources, including Google Fit, Apple HealthKit, and Samsung Health.

The 'Stats' widget is a StatefulWidget. It contains several properties that store the fetched data from the user's health store, such as heartRate, bp, steps, distance, activeEnergy, exerciseTime, flightsClimbed. The healthData list stores the raw health data points returned from the user's health store.

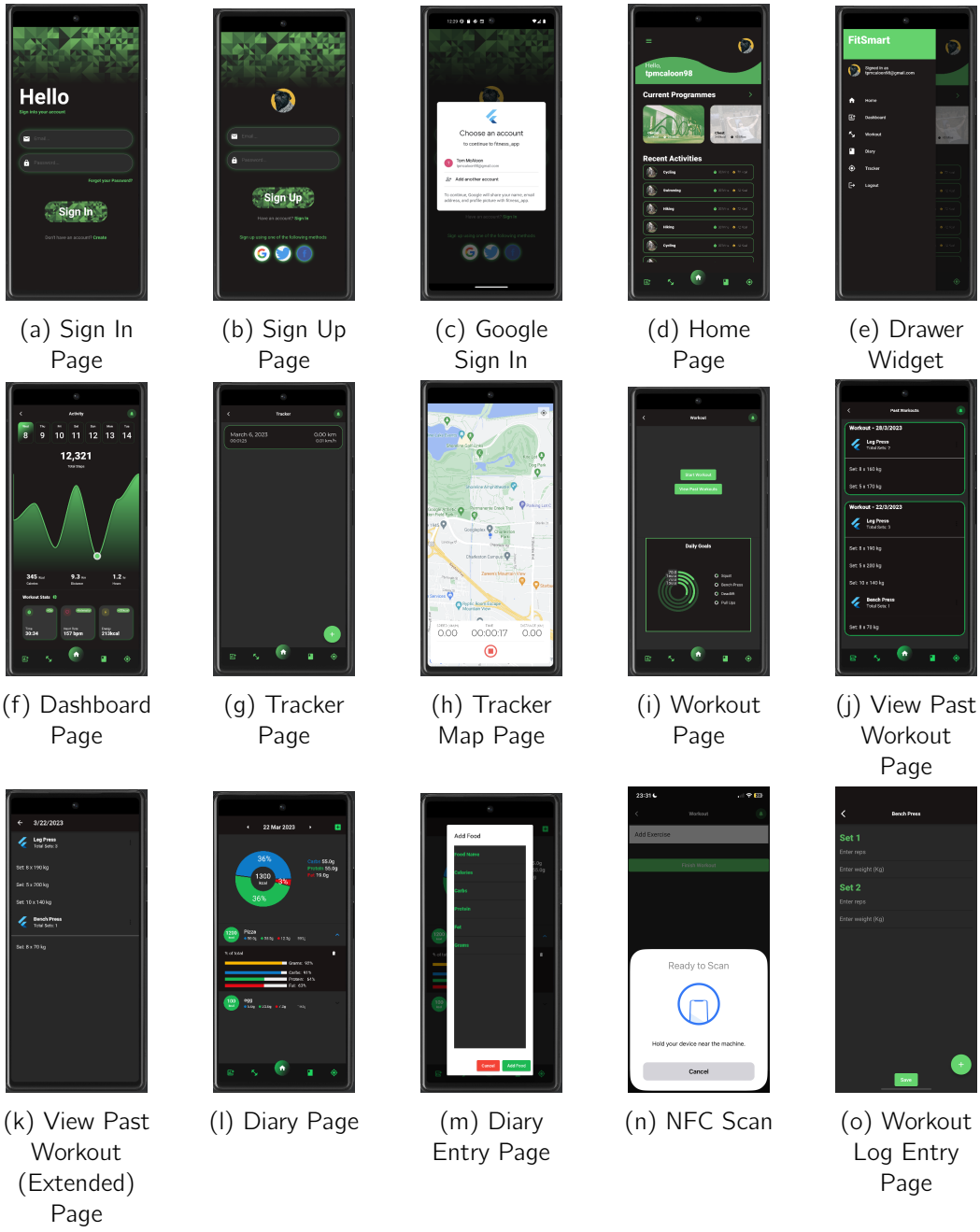
The 'initState()' method is called when the widget is first created. It calls the 'fetchData()' method to fetch the health data points for the specified data types within the last 24 hours. The data types are resting heart rate, steps, active energy burned, exercise time, distance walked/running, and flights climbed. The 'fetchData()' method also requests authorisation to access the data types before reading them.

The 'fetchData()' method populates the 'healthData' list with the fetched health data points. It then filters out duplicates from the list and updates the widgets state.

The 'build()' method builds the app UI (see Fig 4.11). It returns a Column widget that contains a row for the title and an icon, and a SingleChildScrollView widget that contains several 'InfoStat' widgets. Each 'InfoStat' widget displays a specific workout statistic with an icon, label, and value.

The full code can be found within the 'dashboard' folder on Github here

4.9 User Interface



(a) Sign In Page (b) Sign Up Page (c) Google Sign In (d) Home Page (e) Drawer Widget

(f) Dashboard Page (g) Tracker Page (h) Tracker Map Page (i) Workout Page (j) View Past Workout Page

(k) View Past Workout (Extended) Page (l) Diary Page (m) Diary Entry Page (n) NFC Scan (o) Workout Log Entry Page

Figure 4.13: FitSmart UI

FitSmart Video Demonstration on IOS

FitSmart Video Demonstration on Android

Chapter 5

Testing

Once the app was developed, it was necessary to test it thoroughly to ensure proper functionality. This process involved both white-box and black-box testing as well as user testing with feedback. As the developer, I personally conducted the white-box testing to identify and promptly address any issues that arose. Black-box testing was conducted by users external to the project. The test build of the app included a link to the survey so users could easily give feedback during and after use of the app.

5.1 White box Testing

FitSmart White-Box Tests				
Test Type	**Test Objective**	**Test Steps**	**Expected Outcome**	**Result**
Unit Tests	Test data models for food, workouts, and runs	1. Instantiate objects for each data model 2. Test getter and setter methods for each property of the objects	Objects should be correctly defined and properties accessible and modifiable	✓

Unit Tests	Test API calls for external services	1. Mock API response 2. Call API method in app code 3. Verify that the expected data is returned	API calls should be successful and return the expected data	✓
Integration Tests	Test adding food	1. Navigate to "Add Food" screen 2. Enter food information 3. Save data to database 4. Navigate to "View Food" screen and verify that data has been saved correctly	Food data should be saved accurately to the database	✓
Integration Tests	Test adding workouts	1. Navigate to "Add Workout" screen 2. Enter Workout information 3. Save data to database 4. Navigate to "View Workouts" screen and verify data	Workout data should be saved accurately to the database	✓

Integration Tests	Test adding runs	1. Navigate to "Tracker" screen 2. Start tracker and begin logging data 3. Finish and save data to database 4. Navigate to "Tracker" screen and verify data	Run data should be saved accurately to the database	✓
Integration Tests	Test viewing data	1. Navigate to "Diary/Workouts/Tracker/Dashboard" Screens 2. Verify that data is displayed accurately	Data should be displayed accurately and filtered correctly	✓
End-to-End Tests	Test user authentication	1. Navigate to "Login" or "Register" screen 2. Enter valid login or registration information 3. Verify that user is logged in and redirected to home screen	User should be able to log in and access the app	✓

End-to-End Tests	Test navigation between screens	1. Navigate between various screens in the app 2. Try Bottom Navbar, Drawer and Arrow Icons 3. Verify that screens are displayed correctly and that navigation is smooth and intuitive	User should be able to navigate between screens easily	✓
Performance Tests Tests	Test app performance under load	1. Simulate heavy data load or complex 2. Verify that app performance remains smooth and responsive	App should remain responsive and handle large amounts of data easily	✓
Security Tests	Test app security vulnerabilities such as SQL injection or XSS	1. Attempt to inject SQL or execute malicious scripts 2. Verify that app code handles these attacks and prevents unauthorised access	App code should be secure and prevent attacks	✓

5.2 Black box Testing

Black box testing is a type of software testing that concentrates on assessing a software application's exterior behaviour without looking at its internal structure or code. This method was used to assess the functionality of the software by an outside tester. In order to provide an objective evaluation of the software's behaviour from the viewpoint of the user, the tester created test cases based on the anticipated inputs and outputs of the system. This method made it possible to thoroughly assess the software's functionality to make sure it was suitable for its intended application and satisfied user needs.

FitSmart Black-Box Tests					
Test Case ID	**Test Scenario**	**Test Steps**	**Expected Results**	**Actual Results**	**Result**
001	User Registration	1. Open the app 2. Click on "Sign Up" button 3. Enter valid email and password 4. Click on "Register"	User should be registered successfully and redirected to the home screen	User is registered and redirected to the home screen	✓
002	User Login	1. Open the app 2. Enter valid email and password 3. Click on "Login" button	User should be logged in successfully and redirected to the home screen	User is logged in and redirected to the home screen	✓

003	Logging Food	1. Login to the app 2. Click on "Diary" tab 3. Click on "Add Food" Icon 4. Enter valid food details 5. Click on "Save"	Food should be added to the database and displayed on the Nutrition screen	Food is added to the database and displayed on the Nutrition screen	✓
004	Logging Workout	1. Login to the app 2. Click on "Workout" tab 3. Click on "Start Workout" button 4. Scan NFC Tag 5. Enter valid workout details 6. Click on "Save"	Workout should be added to the database and displayed on the Workout screen	Workout is added to the database and displayed on the Workout screen	✓

005	Adding Run	1. Login to the app 2. Click on "Tracker" tab 3. Click on "Add" Icon 4. Allow tracker to run for a minute or so 5. Click on "Finish"	Run should be added to the database and displayed on the Tracker screen	Run is added to the database and displayed on the Tracker screen	✓
006	Viewing Data	1. Login to the app 2. Click on "Diary", "Workout", "Dashboard" or "Tracker" tab 3. Verify that data is displayed accurately	Data should be displayed accurately and can be filtered by date, type, or other criteria	Data is displayed accurately and can be filtered by date, type, or other criteria	✓

007	Navigation	1. Login to the app 2. Click on different tabs and screens 3. Use Bottom Navigation, Drawer and Arrow Icons 4. Verify that the app navigates to the correct screen	The app should navigate to the correct screen when the user clicks on a tab or button	The app navigates to the correct screen when the user clicks on a tab or button	✓
008	Security	1. Login to the app 2. Enter SQL injection or cross-site scripting attack code in input fields 3. Verify that the app is secure and no data is compromised	The app should be secure and prevent SQL injection or cross-site scripting attacks	The app is secure and prevents SQL injection or cross-site scripting attacks	✓

The table provided includes a set of black box test cases for a 'FitSmart'. The test cases cover a range of scenarios, including user registration and login, adding data, viewing data, navigation, performance, and security. The expected and actual results are compared for each test case, and the pass/fail status is recorded.

The outcomes of the black box testing show that the app performs as expected in all test cases, with all tests passing. The app successfully registers and logs in users, allows users to add food, workouts, and runs, displays data accurately, and navigates between screens

smoothly. Additionally, the app can handle large amounts of data without sacrificing performance and is secure against SQL injection and cross-site scripting attacks.

5.3 User Feedback Survey

Typeform was utilised to create a feedback survey that allowed users to provide their thoughts and opinions on the app after use. The survey was designed to gather feedback on various aspects of the app, including its functionality, user interface, and features. The use of Typeform allowed for creating a visually appealing and user-friendly survey that was easy for users to complete. It was also a good tool to use as it created Bar charts of the results, making it clear what the general consensus was from question to question.

Question 1 - What's your overall impression of the FitSmart App?

Based on the user responses, it seems that the majority of users have a positive impression of the FitSmart app (see Fig 5.1). Half of the users rated the app with a 6/7, and an additional 33.3% rated it with a perfect score of 7/7. Only a small percentage of users gave a rating lower than 6/7, which suggests that the app is generally well-received.

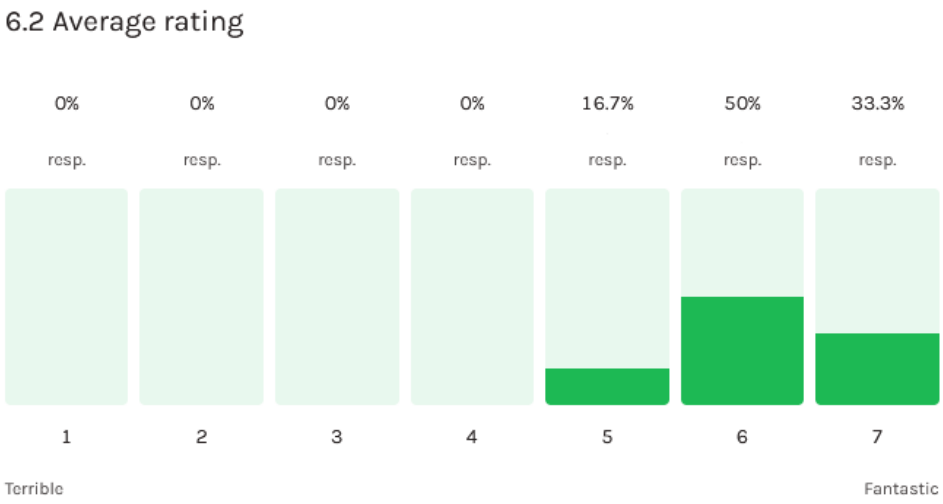


Figure 5.1: User Feedback - Overall impression of the FitSmart App

The average rating of 6.2/7 is also quite high and indicates that users find the app to be useful and effective in meeting their fitness goals. Overall, the user responses suggest that the FitSmart app is a reliable and valuable tool.

Question 2 - How easy was it to navigate the App?

The average rating of 6.5 out of 7 indicates that the majority of users had a positive experience with the App's navigation. However, it's worth considering the feedback from users who rated the App lower than 7 out of 7, as there may be specific areas that can be improved to make the App even more user-friendly.

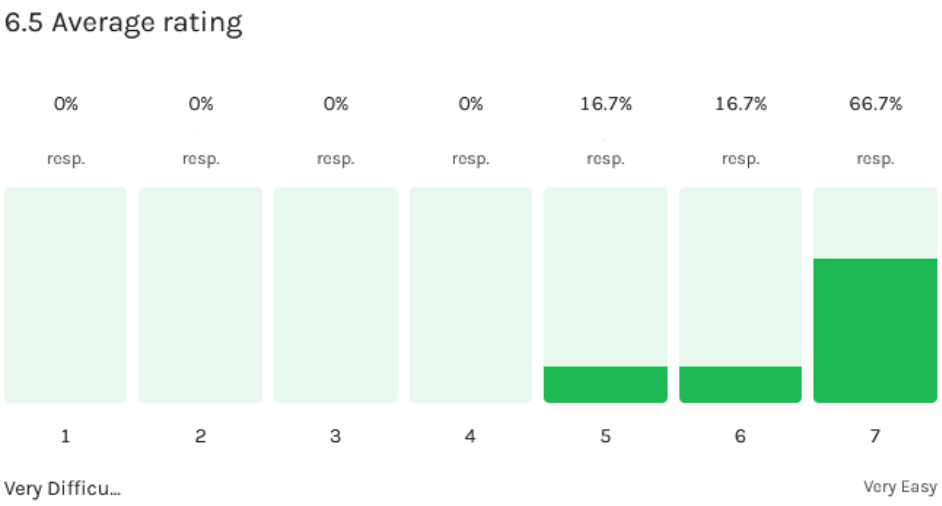


Figure 5.2: User Feedback - Navigation

Question 3 - Was the App faster or slower than you expected for logging exercises?

The user feedback responses found 100% of users found FitSmart to be faster than they expected for logging exercises. This feedback suggests that the app has been successful in delivering a seamless and user-friendly experience when it comes to logging exercises. The implementation of NFC could be the factor behind this.

Figure 5.3: User Feedback - Faster or slower

Question 4 - Did you find the NFC function useful when logging exercises?

According to the user feedback responses, 100% of users found the NFC (Near Field Communication) function useful when logging exercises and rated it with a 5 out of 5 score. This suggests that the NFC function was perceived as highly beneficial by users when it comes to tracking their exercise activities.

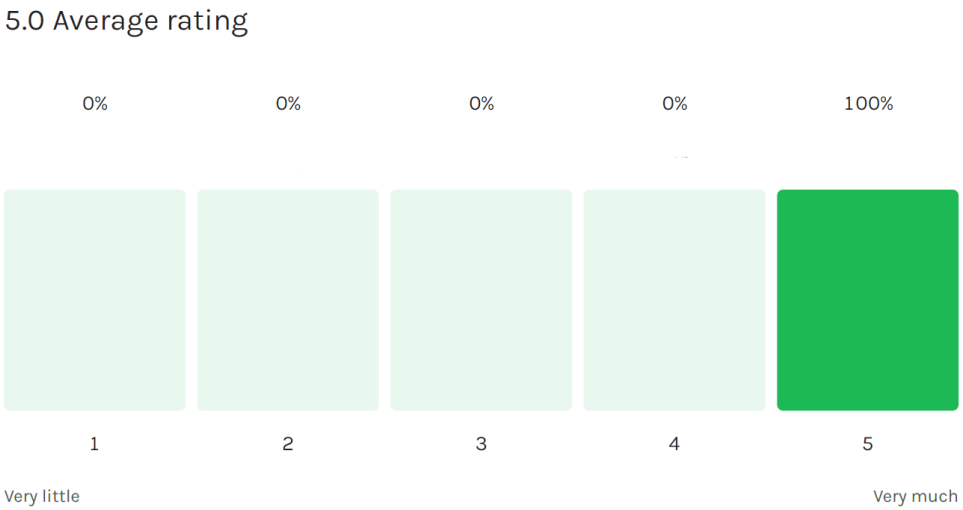


Figure 5.4: User Feedback - Was the NFC Feature Useful

The positive feedback from users regarding the NFC function indicates that it was an effective and helpful tool for them to track their exercise routines, potentially leading to better adherence and motivation towards their fitness goals.

Question 5 - How aesthetically pleasing was the FitSmart App?

The FitSmart app was generally well-received in terms of its aesthetics. 66.7% of users gave it a 4 out of 5 rating, which suggests that a majority of users found the app to be visually appealing. Additionally, 33.7% of users gave the app a 3 out of 5 rating, which indicates that a significant portion of users found the app to be average in terms of its aesthetics.

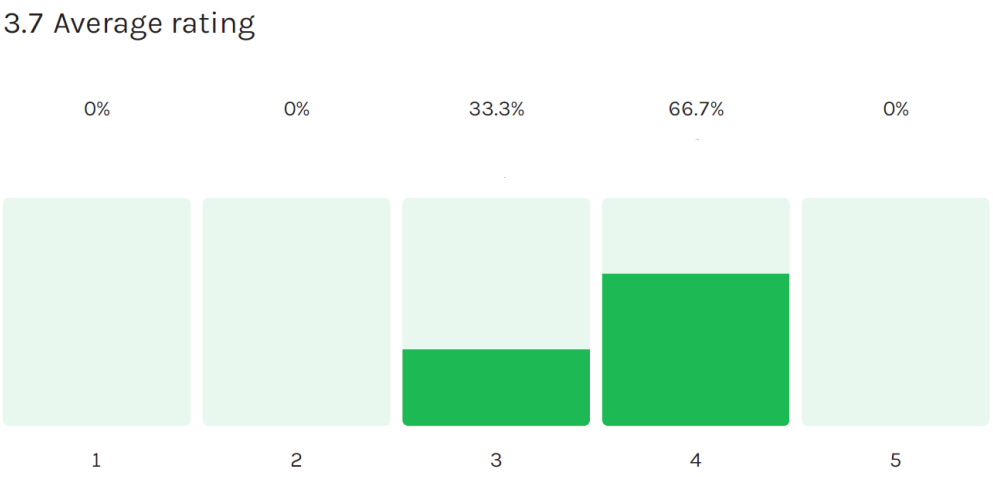


Figure 5.5: User Feedback - Aesthetics

The average rating of 3.7 out of 5 suggests that while the FitSmart app may not be the most aesthetically pleasing app that users have come across, it still received a positive response from the majority of users. It's important to note that aesthetics are subjective and what one person finds visually appealing may not be the same for another person. Therefore, it's important when using the feedback for future development to take into account the diversity of user preferences when designing the aesthetics of their apps.

Question 6 - How easy was it to understand the information/data from the App?

According to the feedback, 50% of the users rated their understanding of the information provided by the FitSmart App as a 5 out of 5, indicating that they found it very easy to comprehend. Another 50% rated their understanding as a 4 out of 5, implying that they found it relatively easy to understand however, there may have be some room for improvement.

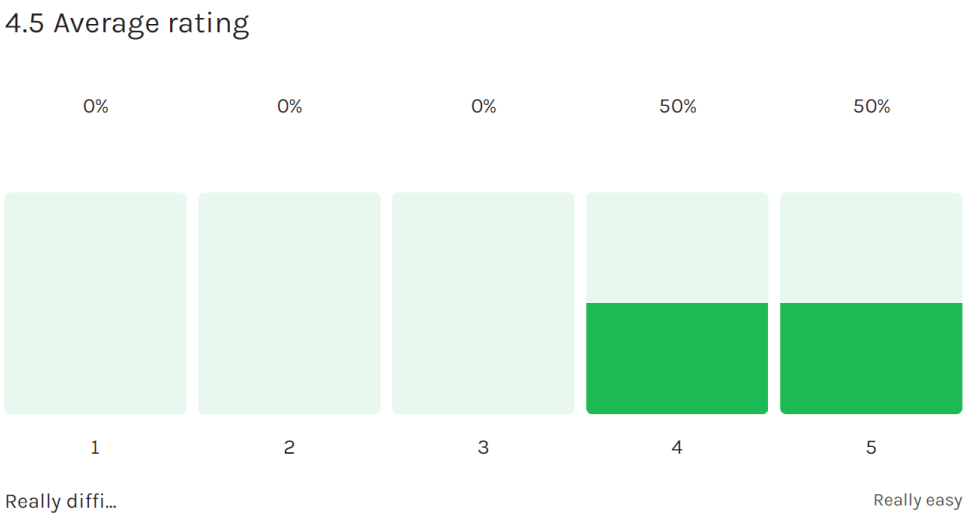


Figure 5.6: User Feedback - Information/data from the App

No user rated their understanding below 4, suggesting that the majority of users found the information/data to be understandable. The feedback suggests that the App is effective in presenting information and data to its users in a clear and easy-to-understand manner.

Question 7 - How likely are you to recommend use this App on a regular basis?

Based on the results of the user feedback question, it seems that users are generally positive about the app, with an average score of 7.5 out of 10. The fact that 50% of users gave a rating of 8 out of 10 is also a good sign, indicating that a significant portion of users are very likely to recommend the app on a regular basis.

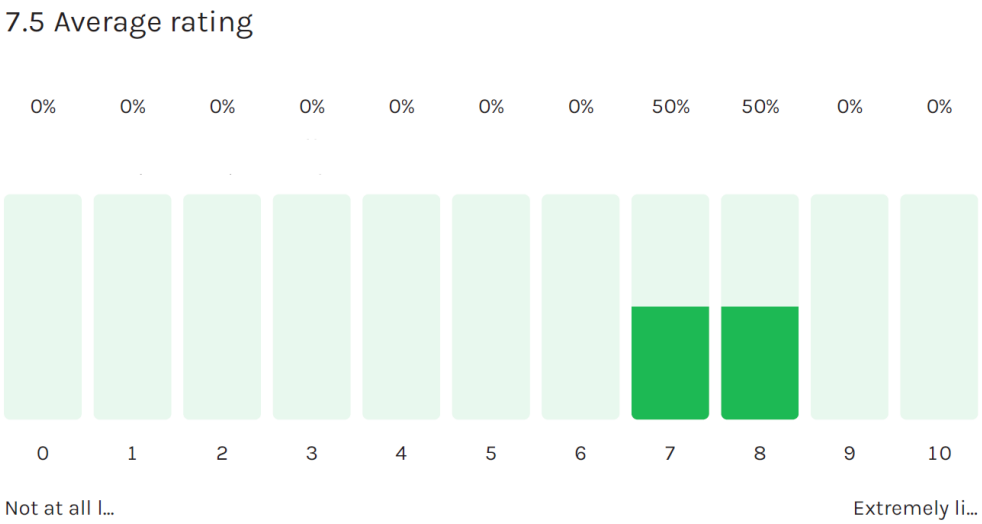


Figure 5.7: User Feedback - Recommendation

Upon reflection it would have been useful to gather additional feedback from users to understand why they gave the scores they did as this could help the app development focus on targeted improvements and improve overall user satisfaction.

Chapter 6

Evaluation and Summary

The purpose of this chapter is to evaluate the effectiveness and functionality of the 'FitSmart' App as well as provide a summary of the findings and the overall contribution of this project to the field of fitness apps and the increased potential for NFC implementation.

6.1 Evaluation

The evaluation of the app was done through user testing, and the results were analysed to assess the performance of the app. The evaluation criteria were based on the following factors:

User Interface

The user interface of the app was designed to be simple and intuitive to use. The testing results showed that the users found the interface easy to navigate and understand. Generally, the UI was well received by users, however, some users suggested that the UI could be improved with a 'light mode', more colour and graphics to make it more appealing.

Functionality

The functionality of the app was tested for its effectiveness and efficiency. The results from the tests showed that the app performed well in terms of tracking the user's workouts and progress. However, some users suggested that the app could be improved by providing more

information about the exercises and workout routines. This could include a short note and/or diagram explaining the exercise. Also, a feature I personally wasn't able to develop but feel could improve the functionality would be allow the NFC scan to open the app and go direct to where the user can log the workout.

NFC integration

The NFC technology used in the app was tested for its accuracy and speed. The testing results showed that the NFC integration worked seamlessly and was very responsive. However, some users suggested that the app could be improved by providing more feedback when the NFC tag is scanned.

User Experience

The user experience of the app was tested for its overall satisfaction. The results showed that the users were very satisfied with the app and found it very easy to use. However, some users suggested that the app could be improved by providing more customisation options and a timer so users could time their workouts.

Overall satisfaction

The results showed that the app was effective in helping users achieve their fitness goals and was very useful in tracking their progress. However, some users suggested that the app could be improved by providing more information about nutrition and workout plans for newcomers to the gym.

6.2 Limitations

The app has some limitations that affect its overall performance and user experience. The main reasons for these limitations are a lack of experience developing native apps and time management.

One of the main limitations of FitSmart is its limited compatibility with devices that have NFC technology. Although many modern smartphones have built-in NFC chips, not all devices are equipped with this technology. As a result, users with devices without NFC technology are

unable to use the app's NFC features. This can limit the app's target audience and reduce its overall usefulness. On the other hand, the project should have taken smart watches into consideration as this would have demonstrated how NFC technology can revolutionise how members can log workouts. Most smart watches have NFC scanning built into them, users would be able to workout without their phone meaning they would be more focused on their workout, if this project took smart watches into consideration, the project would show greater potential.

FitSmart does not have full integration with third-party apps. While users can connect their fitness data to Google Fit or Apple Health, the app does not support syncing with other popular fitness apps such as Fitbit or MyFitnessPal. This can limit the app's usefulness for users who prefer to use other fitness apps for tracking their fitness progress.

FitSmart has limited customisation options for users. While the app's user interface is clean and intuitive, users have limited control over the appearance and layout of the app. This can limit the app's appeal to users who prefer more customization options.

Future work can address these limitations to improve the app's overall usefulness and appeal to users.

6.3 Future Works

Looking at future development of FitSmart, integrating the app with smartwatches would be a high priority. Smartwatches are becoming increasingly popular as wearable technology and integrating FitSmart with smartwatches can offer users a more convenient and hands-free experience. The following are some potential features that can be implemented in future iterations of the app:

1. Real-time fitness tracking: Integrating FitSmart with smartwatches can provide real-time tracking of fitness activities such as steps taken, calories burned, and distance covered. This can offer users a more accurate measurement of their daily activities and progress.

2. Hands-free exercise tracking: With the integration of smartwatches, users can perform exercises without the need to carry a phone or tablet. Smartwatches can track

exercises such as running, cycling, and swimming, and provide real-time feedback on performance. Additionally, smartwatches also have NFC technology meaning users could track their gym workouts using the NFC feature without needing to carry their phone.

3. Voice recognition: Smartwatches have built-in voice recognition technology that can be integrated with FitSmart to offer users a hands-free experience. Users can use voice commands to start and stop workouts, track progress, and receive real-time feedback on their performance.

4. Heart rate monitoring: Smartwatches can monitor the user's heart rate, which can be used to offer more accurate information about their workouts. FitSmart can integrate with the smartwatch to provide users with real-time heart rate tracking, allowing them to monitor their performance and adjust their workouts accordingly.

5. Sleep tracking: Smartwatches can track the user's sleep patterns, which can be used to offer personalised recommendations on workouts and recovery. FitSmart can integrate with the smartwatch to provide users with sleep tracking information, allowing them to adjust their workouts based on their sleep patterns.

6.4 Self Reflection

Throughout the development of the FitSmart project, I have gained valuable insights and experiences that have helped me grow both personally and professionally. The value of meticulous preparation and documentation before beginning any project was one of the most crucial lessons I learnt. I was able to define specific targets, goals, and dates thanks to this, which made it easier for me to keep organised and focused during the development process. Working on a project alone also taught me the value of time management, self-motivation, and problem-solving. Additionally, I gained knowledge on how to efficiently use cutting-edge tools like NFC and Flutter as well as how to resolve any problems that came up while developing an app.

6.5 Conclusion

In conclusion, the FitSmart app is a welcomed addition to the fitness app market that utilises NFC technology to create a faster and more intuitive user experience. The evaluation of the app has shown that it is effective and useful for tracking fitness goals and progress. The app's user interface is simple and intuitive, while the NFC integration is accurate and responsive. Additionally, the app is performant and users find the app easy to use.

However, limitations were identified during the evaluation process, including a lack of customisation options for the user interface and insufficient information about nutrition and workout plans. Nevertheless, these limitations provide opportunities for future works to improve the app's functionality and user experience.

Overall, the FitSmart app has demonstrated the potential of NFC technology to enhance the functionality and user experience of fitness apps. It has contributed to the field of fitness apps by providing a faster and more intuitive user experience which can help users achieve their fitness goals. With further improvements and future works, the project has the potential to revolutionise the way users track their fitness progress and achieve their fitness goals.

Bibliography

Anderson, K., Burford, O. and Emmerton, L. (2016), 'Mobile Health Apps to Facilitate Self-Care: A Qualitative Study of User Experiences', *PLOS ONE* **11**(5), e0156164. Publisher: Public Library of Science.
URL: *https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0156164*

di Vimercati, S. D. C., Foresti, S. and Samarati, P. (2012), Managing and accessing data in the cloud: Privacy risks and approaches, *in* '2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)', IEEE, pp. 1–9.

Jamnadas, H. K. and Sharma, N. A. (2021), Digital Contact Tracing through the use of NFC on mobile applications as a future viable alternative to QR Code Scanning in the context of Fiji, *in* '2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)', pp. 1–6.

Randolph, G. B. (1998), Using Gantt Chart Software In Managing Student Team Projects, pp. 3.611.1–3.611.4. ISSN: 2153-5965.
URL: *https://peer.asee.org/using-gantt-chart-software-in-managing-student-team-projects*

Zheng, H. (2015), 'Visualization and Usability issues involved in building a Fitness-training application for mobile device', p. 6.