

# Personal Trainer Manager App

CS39440 Major Project Report

Author: Jay Kirkham (jak77@aber.ac.uk)

Supervisor: Dr/Prof. Chris Loftus (cwl@aber.ac.uk)

15<sup>th</sup> April 2024

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BEng degree in  
Software Engineering (with integrated year in industry) (G600)

Department of Computer Science  
Aberystwyth University  
Aberystwyth  
Ceredigion  
SY23 3DB  
Wales, UK

## Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name Jay Kirkham

Date 15/04/2024

## Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Jay Kirkham

Date 15/04/2024

## Acknowledgements

I give my thanks to Chris Loftus for his helpful support and guidance during the development of the project, including the weekly meetings that kept me on-track to complete the tasks. I further extend my thankfulness to my 2<sup>nd</sup> marker, Edore Akpokodje, for the helpful words of wisdom during the mid-project demonstration, which guided the project in the right direction.

I also give my thanks to my parents, who have been supportive throughout my entire degree, and Savannah Raymond, who has been by my side throughout the whole year, helping me stay positive and focused (and fed).

## Abstract

The Personal Trainer Manager App project has developed a mobile app for Android devices that helps personal trainers and their clients interact with each other to develop the most optimal workouts for the client. It provides an interface for personal trainers to create workout routines of variable length for their clients, and allows clients to view these workouts. Clients are also able to create workouts for themselves, which the trainers will have access to.

This project includes different interface variations for personal trainers and clients, giving specific features for each group. It also features the ability for trainers to “link up” with clients through searching and accepting “linking” requests that both parties can send to each other.

This project also provides a way to create and modify workout routines and set them to repeat weekly, so that all possible client needs are catered for. This includes workouts containing multiple different exercises.

This project allows clients to progress through their workouts in real-time, providing live information and timing for exercises where possible. This will allow clients to easily keep track of what they are meant to be doing at the time on their app. The personal trainer is able to see this progress on their app, so they can keep track of the client and ensure the workout is being followed correctly.

## Contents

<b>1. BACKGROUND, ANALYSIS &amp; PROCESS .....</b>	<b>8</b>
<b>1.1. Background .....</b>	<b>8</b>
<b>1.2. Analysis.....</b>	<b>8</b>
1.2.1. Storage Cloud Solution Providers.....	8
1.2.1.1. Microsoft Azure – CosmosDB .....	8
1.2.1.2. Amazon Web Services – SimpleDB .....	8
1.2.1.3. Cloudflare – R2 Storage Bucket.....	9
1.2.1.4. Google Firebase – Cloud Firestore.....	9
1.2.1.5. Wasabi – Hot Cloud Storage .....	9
<b>1.3. Process.....</b>	<b>9</b>
1.3.1. Methodology.....	9
1.3.2. Version Control .....	10
<b>2. PROJECT REQUIREMENTS.....</b>	<b>11</b>
<b>3. DESIGN.....</b>	<b>12</b>
<b>3.1. Overall Design .....</b>	<b>12</b>
<b>3.2. Framework.....</b>	<b>12</b>
<b>3.3. Authentication .....</b>	<b>12</b>
<b>3.4. App Theme.....</b>	<b>13</b>
<b>3.5. App Navigation .....</b>	<b>13</b>
<b>3.6. Database Design.....</b>	<b>13</b>
<b>4. DEVELOPMENT PROCESS.....</b>	<b>15</b>
<b>4.1. Weekly Meetings .....</b>	<b>15</b>
<b>4.2. Week 1 (29<sup>th</sup> January).....</b>	<b>15</b>
<b>4.3. Week 2 (5<sup>th</sup> February).....</b>	<b>15</b>
<b>4.4. Week 3 (12<sup>th</sup> February).....</b>	<b>16</b>
<b>4.5. Week 4 (19<sup>th</sup> February).....</b>	<b>16</b>
<b>4.6. Week 5 (26<sup>th</sup> February).....</b>	<b>16</b>

4.7.	Week 6 (4 <sup>th</sup> March).....	16
4.8.	Week 7 (11 <sup>th</sup> March).....	17
4.9.	Week 8-11 (18 <sup>th</sup> March – 14 <sup>th</sup> April) Easter Break .....	17
4.10.	Week 12 (15 <sup>th</sup> April) .....	17
4.11.	Week 13 (21 <sup>st</sup> April).....	17
4.12.	Week 14 (28 <sup>th</sup> April) – Final week.....	17
5.	IMPLEMENTATION .....	19
5.1.	Authentication .....	19
5.2.	App Navigation .....	19
5.3.	Data Storage .....	20
5.4.	Data Structure.....	21
6.	TESTING .....	21
6.1.	Test Table.....	21
6.2.	Test Evidence .....	26
7.	CRITICAL EVALUATION.....	29
7.1.	Project Requirements.....	29
7.2.	Methodology.....	30
7.3.	Documentation .....	30
7.4.	Testing .....	30
7.5.	What to do next .....	31
8.	REFERENCES.....	32
9.	APPENDICES .....	35
A.	Use of Third-Party Code, Libraries and Generative AI .....	35
A.1.	Third-Party Code.....	35
A.2.	Libraries.....	35
B.	Cloud Storage Solutions Analysis.....	37
B1	Requirements.....	37

<b>B2</b>	<b>Candidates .....</b>	<b>37</b>
<b>B3</b>	<b>Analysis Table.....</b>	<b>38</b>
<b>B4</b>	<b>Conclusion.....</b>	<b>38</b>

# 1. Background, Analysis & Process

## 1.1. Background

The main focus of this project is to develop an Android [1] application, using the Kotlin [2] programming language, to be used by both personal trainers and client to work together and produce a workout routine that works best for the client.

Android is an operating system unveiled in November 2007 [3] that was developed to be used on touchscreen mobile devices produced by hundreds of manufacturers, such as Samsung, Huawei and HTC. In recent times, Android has expanded to operate on other devices such as smart watches, TVs and even cars. Android is based on a modified version of the Linux kernel, together with other open-source software sources. The official language used for the user interface of Android is Kotlin, which is why Kotlin is the language chosen for this project.

Kotlin is a cross-platform, statically typed programming language [4] that began development in 2011 under the management of JetBrains [5], a Czech software development company. The first official version, Kotlin 1.0, was released early 2016, and since then Kotlin has received numerous updates, with the latest version, Kotlin 1.9, releasing July 6<sup>th</sup> 2023. Kotlin was designed to be a “better language” than Java, whilst also being compatible with Java code, which allows companies to easily transition between Java and Kotlin.

## 1.2. Analysis

### 1.2.1. Storage Cloud Solution Providers

As preparation for this project, I carried out research on possible solutions for cloud storage. This was to establish which cloud storage solution would be most suitable for the project and the app, and for the best price.

5 providers were taken into consideration:

- Microsoft Azure [6]
- Amazon Web Services (AWS) [7]
- Cloudflare [8]
- Google Firebase [9]
- Wasabi [10]

#### 1.2.1.1. Microsoft Azure – CosmosDB

Microsoft Azure’s storage solution, CosmosDB [11], is a NoSQL cloud storage solution hosted by Azure, which is Microsoft’s software development tool providing service that promises strong security and cloud networking. It provides a free service with lowered rate limits and a storage cap of 25GB, a generous amount given its unpaid for, as well as free backups. However, it provides limited tools for app development.

#### 1.2.1.2. Amazon Web Services – SimpleDB

Amazon Web Services, or AWS, is Amazon’s cloud-based computing service platform that is used by thousands of internet services worldwide. Amazon Web Services’ main guarantee is its reliability, scalability and affordability, with a “pay only for what you use” strategy. [7]



SimpleDB [12], Amazon's NoSQL cloud storage solution, provides 1 free GB per month, with a rate limit of 25 hours per month. It also provides a decent selection of app development tools. Furthermore, I have experience with AWS's services before thanks to my industrial year.

#### 1.2.1.3. Cloudflare – R2 Storage Bucket

Cloudflare's R2 Storage Bucket [13] is a NoSQL object bucket-style cloud storage solution that provides generous storage caps (10GB/month) and rate limits (1 million writes, 10 million reads/month) on their free service. However, this service is not backed up and does not provide any app development tools.

#### 1.2.1.4. Google Firebase – Cloud Firestore

Google's Firebase Firestore [14] cloud solution is a NoSQL document-style cloud storage solution that offers 1 GiB (1024mb) storage and 50k reads, 20k writes and 20k deletes per day. It does provide backups, but only as a paid service. It also allows for integration with Google accounts, and provides app development tools tailored to Android, which is what the project is based on.

This solution is the one that was chosen as a result of its Android-tailored nature and generous rates, as well as how well it suits the demands of the project.

#### 1.2.1.5. Wasabi – Hot Cloud Storage

Wasabi [15] provides a paid-only service that is based on object buckets, which comes with a free 30-day trial that gives a generous 1TB storage and rate limits of 1000 GET, 100 PUT, 100 POST and 10 DELETE per minute. It also has no development tools, which limits its potential as a development service.

This solution was decided against early on in the analysis process as it is essentially a paid-only solution, which was not an option for this project as there is no funding available. Furthermore, the 30-day trial period was deemed far too short for the project lifespan, and would cause major issues during the development process as a new provider would be required.

### 1.3. Process

#### 1.3.1. Methodology

A wide range of methodologies were considered for this project, with a focus towards Lean [16], Scrum [17] and Kanban [18]. As a result of many factors, such as the singular team size and the limited timeframe, as well as my own personal development methods, it was chosen to use Kanban, as it was the easiest way of tracking current progress and tasks, and didn't limit me to certain timeframes so I didn't rush sections and reduce the quality of the overall project.

To work alongside this methodology, a kanban board was setup using the GitLab issues feature, which allows you to create "tickets" with descriptions and tags to track progress.

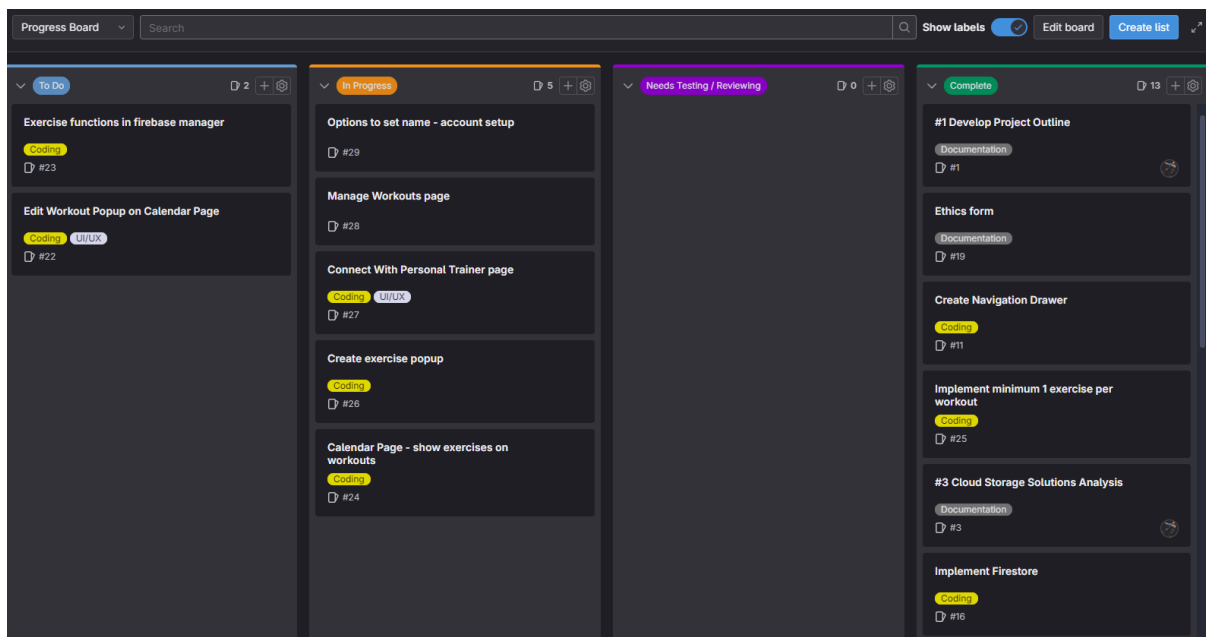


Figure 1 Kanban Board for the project

Kanban is a lean methodology designed to help manage and improve workflows by aiming to manage work by creating a balance of demands and capacity using the visualisation of work items, typically on an interactive board called a Kanban board. Kanban, which directly translates to “signboard” or “billboard” in Japanese, was pioneered by Toyota in the late 1940s to help them match production of vehicles to customer demand and identify potential limitations in material stocks. Another benefit of Kanban is that it allows the developers to limit how much work they are doing at a given time, ensuring that they aren’t overloaded with tasks and can avoid burnout. Kanban relies on the concept of “user stories”, which are targets and end goals for the development of a software system written from the perspective of an end user in an informal and natural language context. Kanban boards also are frequently used with “epics”, which are collections of smaller tasks or “user stories” designed to help break down the development process into more manageable pieces that can help spread the workload across a development team.

The reason Kanban was chosen over other methodologies such as Scrum or Lean is because most other methodologies tend to require additional ceremonies, such as weekly sprint planning, reviews and retrospectives, which are less effective during solo development and can be quite time consuming, particularly on a time-limited project such as this one. Furthermore, Kanban can be more adaptive to less predictable workflows due to its lack of restriction on time, which makes it more suitable for this project than other methodologies as the limited timeframe and lower experience levels can lead to frequent changes in plans.

### 1.3.2. Version Control

As there will be multiple devices used to develop the application, as well as the improved code safety and security in case of hardware failure, it was decided to store the code on a cloud-based version control service.

It was decided that GitLab [19] would be the chosen version control, as it comes with various beneficial features, such as issues boards as well as pipelines for continuous implementation. Furthermore, the university has locally hosted secure instances of the service. The code is hosted privately on the university’s service, which guarantees even more security and features that wouldn’t normally be available.

GitLab is an alternative to GitHub [20] that is more focused on the development process for a team rather than as a public interface for developers to showcase and share their code. Because of this, GitLab provides useful development tools such as issues, issue board and Continuous Integration & Continuous Development (CI/CD) pipelines [21]. GitLab was chosen to host the code over GitHub thanks to its issue boards, which were crucial to the project as it was used as the Kanban board.

## 2. Project Requirements

The following table contains the list of project requirements outlined for the project. These high-level requirements are the goals in which the project should achieve, with the useful priority standard known as MoSCoW [22] (Must have, Should have, Could have, Won't have), which is a recognised method of prioritization that is applicable to many situations, including software development. "Must have" represents the highest priority, representing the core required features of the project. "Should have" represents the features that would be almost fundamental but not mandatory. "Could have" project requirements are the additional requirements for the project that are better described as secondary or tertiary features, and are not critical to the output. "Won't have" represents the project requirements that will not be addressed, or won't be implemented in the near future or until the requirements are changed.

ID	Description	Priority
PR1	The user should be able to access the app through an authentication method, and be able to easily sign in and out of the application.	SHOULD HAVE
PR2	The user must be able to select what kind of user they want to be, between Client and Personal Trainer.	MUST HAVE
PR3	The user, whilst using the app as a Client, must be able to create, modify and delete workouts.	MUST HAVE
PR4	The user, whilst using the app as a Client, should be able to add and remove exercises from workouts.	SHOULD HAVE
PR5	The user, whilst using the app as a Client, should be able to view their workouts using a calendar view.	SHOULD HAVE
PR6	The user, whilst using the app as a Client, should be not be able to see workouts created by other users.	SHOULD HAVE
PR7	The user, whilst using the app as a Client or as a Personal Trainer, must be able to create a "link" with another user that is using the other user type. A Client user should be able to link with a Personal Trainer user, and vice versa.	MUST HAVE
PR8	The user, whilst using the app as a Client or as a Personal Trainer, should be able to communicate with other users that they have linked with using a form of text-based communication within the application.	SHOULD HAVE
PR9	The user, whilst using the app as a Personal Trainer, must be able to create, edit and delete the workouts of Client users that they have created a link with.	MUST HAVE

<b>PR10</b>	The user, whilst using the app as a Personal Trainer, should be able to add, edit and remove exercises from the workouts of Client users that they have created a link with.	SHOULD HAVE
<b>PR11</b>	The user, whilst using the app as a Client, could be able to work through their workouts in real-time.	COULD HAVE
<b>PR12</b>	The user, whilst using the app as a Personal Trainer, could be able to watch the progress of the client's workouts in real-time.	COULD HAVE

## 3. Design

### 3.1. Overall Design

The main focus of the app design is that it is fast, responsive, and user-friendly. This is to ensure that users would be able to carry out their exercises and workouts efficiently and with minimal delays caused by the app. It also aims to be a core part of a user's exercise routine.

It was decided early on that the app would effectively function as two very similar apps, with one side catering to clients and the other targeted towards the personal trainers. This was to ensure that both user types could use the functionality of the application without having to develop two separate applications. To accompany this, it shall be as easy as possible to switch between the two app configurations for ease of use.

### 3.2. Framework

As this project is to develop an Android app using Kotlin as the primary programming language, a choice was to be made between using Jetpack Compose [23], a Kotlin-focused framework for Android UI development, and the older, traditional XML-style of app design. After some decision making, it was decided that Jetpack Compose would be the core Android framework that is used for this project. This is because it is a much more modern approach and has many more features. It is also more intuitive to use as all code is written in Kotlin as opposed to half-Kotlin/Java and half-XML. Furthermore, I am more experienced with using Jetpack Compose as opposed to XML as a result of a previous module in the course. Furthermore, Jetpack Compose allows for dynamic, animated displays as opposed to the static nature of XML. [24]

One drawback of using Jetpack Compose as opposed to XML is that the separation of app design and app logic is lost with Jetpack Compose. However, this has more of an impact on larger teams where the designers and developers are separate groups and trying to work together on the same code space can cause various issues.

It was also decided that the build tool used for the application will be Gradle [25]. This is because it is the official build tool for Android development and for Android Studio [26].

### 3.3. Authentication

For the best security, it was decided that the application would feature an authentication method that relies on OAuth2 (Open Authorization), which is a kind of access token that allows you to

connect using the same account across multiple applications. [27] For example, because of OAuth2 it is possible to sign into Spotify with a Google account.

As the application is an Android application, most users will be running the device using a Google account. As a result, using Google as a form of authentication would be very beneficial in providing a smooth and seamless experience for users, both first-time and returning.

### 3.4. App Theme

The aim of the theme is to present the user with punchy, energising colours that encourage positivity and activity. As a result, the theme uses various shades of blue and also provides a variant for dark theme to cater to various user wants. The theme is toggleable in the user settings between Dark, Light, and System, System being derived from the user's current device theme.

The theme is automatically generated using an online material3 browser-based service that produces the Kotlin Color.kt, Theme.kt and Type.kt files for the ui.theme package [28]. The Color & Theme files contain the data for the colours used within the app, and the Type file contains a default typography that is used throughout the app to control the displayed text.

### 3.5. App Navigation

It was intended from the start of the project that the app would feature a sophisticated and straightforward method of app navigation. As a result, a decision was made that the app would feature a variety of different methods to navigate through the screens.

One method decided on was the bottom navigation bar, which is a common practice among android applications and can be found in many applications across the web. Originally, the application was planned to have 4 options in this bottom bar: A homepage, which displays all the information that a user would want to see when they first open the application, a calendar page, which gives a display of all the upcoming workouts, a chat page, which allows the user to communicate with other users, and a settings page, which allows the user to configure various options such as appearance. However, it was deemed bad practice to include the settings in the bottom navigation bar, so it was moved to the other main navigation method, which is the modal navigation drawer [29]. A modal navigation drawer is a type of drawer that overlaps existing content rather than pushes it aside like a standard navigation drawer. Navigation drawers typically contain navigation options that are not as high level as those found on the navigation bar, therefore it acts as a secondary method of navigation and features methods such as settings and signing out.

### 3.6. Database Design

The database used for this project is the Firebase Firestore [14], which is a NoSQL cloud storage solution that uses a document and collection storage method. This method is more suitable for the project than SQL-type storage methods as it allows for far more flexibility as the storage methods are not limited to relational tables, so can be finetuned to fit the context of the project more accurately. Furthermore, as there is no normalisation of tables, meaning that you don't need to join data together, queries for items in the database are a lot more efficient and execute much faster. [30]



Figure 2: Database Design, created on Figma

## 4. Development Process

### 4.1. Weekly Meetings

There was a weekly 30-minute meeting with the project supervisor held on Mondays. During this meeting, various topics would be covered, including the progress made last week, the plans for the next week, and any future goals or tasks to be taken later on. These meetings were also used for reviewing any produced documents for the project, such as the project outline or the cloud storage analysis. The feedback provided during these meetings was greatly beneficial and vastly improved the quality of the documents before they were finalised, as well as the quality of the application.

Another benefit of these meetings was that I was able to express my plans and problems to the supervisor. Thanks to this, I was able to get feedback on these plans, as well as any tips and advice that could be provided to solve the problems. For example, an issue had appeared where calling database functions just once as every time the data was returned to the composable function, it would recompose and re-call the function, retrieving the data a second time, and would occasionally loop if I was not using a map. The project supervisor was able to provide a solution that was the Compose function `LaunchedEffect` [31], one of many side effects, which ignores recomposition when calling functions and instead relies on a manual key that triggers every time the value set to the key is changed.

### 4.2. Week 1 (29<sup>th</sup> January)

The first week of the project was spent planning with the supervisor what the project should aim to produce, and starting the development of the Project Outline, one of the documents required to be submitted for the project. Included in this was a thought process on what methodology that would be used for the development of the project. Along with this, the development environment was also considered, which was a fairly straightforward decision due to the specific nature of the project. As a result, it was decided early-on that Kotlin was the language to be used, with the AndroidStudio [26] IDE developed by JetBrains [5].

During this week, various systems were setup in advance for the development of the project. One of the first things done was that a GitLab project was configured to act as a version control service for the project. Furthermore, the AndroidStudio project was initiated and various Firebase plugins and dependencies were added in advance in preparation for later development.

### 4.3. Week 2 (5<sup>th</sup> February)

During the second week of development for the project, it was decided that an analysis of possible cloud storage solutions would be advantageous, as the project intends to use cloud-based storage as one of its various features. As a result, various storage solutions were analysed throughout the week, but the completed document would not be completed until the next week.

During this week, the project outline specification document was also developed further to the point of completion so it could be turned in before the deadline date of the 12<sup>th</sup> of February. Included in this further development was a discussion and review with the supervisor during the weekly meeting, which enhanced the quality of the final document. The project outline contained the project description, proposed tasks and project deliverables, and was submitted on time on the afternoon of the Sunday this week, the day before the deadline.

#### 4.4. Week 3 (12<sup>th</sup> February)

This week, the development of the technical submission began, starting with some rough concepts of data classes for things such as workouts and exercises. They were in no way final, but they gave a good example of what was planned. This week also contained some work towards other parts of the app structure, such as the packages and layout of the code. Furthermore, some work was done towards the development of the authentication of the app, including research into the documentation of the Firebase Auth [32] service.

On the side, the cloud storage solutions analysis document was also further improved, with more comparisons and a conclusion being drafted, but the majority of development was focused on the application itself, so the changes to the analysis document were minor.

#### 4.5. Week 4 (19<sup>th</sup> February)

This week marked a major step forward in the development of the project, as the authentication UI was successfully added to the project, thanks to the provided pre-made user interface from Firebase known as FirebaseUI [33]. This UI dependency was hugely beneficial as it meant that crucial development time did not need to be focused on producing a log-in screen, as the UI provides both the graphics and the functions to handle the authentication itself. At this stage, there were still some issues with the authentication itself, as there were limitations relating to Facebook and Twitter authentication that caused a setback. These limitations were that Facebook and Twitter require a business account to be set up in order to use their authentication APIs, which required a business address, website and email address, three things that did not exist as there is no business involved with the production of this project. However, at this point in the development, I had not yet discovered this implementation issue.

During this week, the cloud analysis document had been completed, so all efforts could be focused on the development of the app itself.

#### 4.6. Week 5 (26<sup>th</sup> February)

The main focus of this week was to develop the navigation aspects of the app. It was decided that the app should feature both a navigation bar at the bottom of the page, as well as a navigation drawer with additional lower-ranking options that are not as critical to the application as those on the bottom navigation bar. Both features were successfully implemented with support for future changes. Furthermore, the authentication process was finalised, with the exception of Facebook due to their requirements. options due to their business requirements. This week was also the first week where it was possible to demonstrate the app working on a physical Android device.

#### 4.7. Week 6 (4<sup>th</sup> March)

The main focus of this week was introducing the functionality of Firebase Firestore cloud storage. This included a rough implementation of creating workouts, as well as a Calendar view that displayed the workouts across the next two weeks. During this week, the decision was also made to drop Facebook and Twitter as authentication options, as they require various variables such as business location and website, factors which cannot be provided for as mentioned in week 4. It was not deemed necessary to create a whole business for the sake of authentication, nor was it considered right to “fake” the details.



#### 4.8. Week 7 (11<sup>th</sup> March)

This week's big task for the first few days was preparation for the mid-project demonstration, which was held on Wednesday 13<sup>th</sup> with the 2<sup>nd</sup> marker, Edore Akpokodje. The demonstration was very successful, and as a result the project was graded full marks for each section for a total score of 5/5. During the demonstration, some beneficial feedback was provided on how some possible problems could be overcome.

The rest of the week was spent putting the demonstration feedback to good use and reinforcing the quality of the app in its current state. This included improving the display of the Calendar screen, as well as adding various new features such as the ability to edit the workouts.

#### 4.9. Week 8-11 (18<sup>th</sup> March – 14<sup>th</sup> April) Easter Break

This section contains the 3 weeks of the Easter holiday as well as the week before. As this is the holiday period, the development of the project slowed down, but some minor tasks were still completed, such as the creation of the composable functions that will be used as the screens for connecting to personal trainers and clients. During this period, work begun on developing the implementation for the linking of the personal trainers and the clients, allowing them to work together, particularly the addition of the new screens and their corresponding functionalities, in a rough form. This included allowing the personal trainers to create workouts for other people.

#### 4.10. Week 12 (15<sup>th</sup> April)

During the 12<sup>th</sup> week, the main focus was to begin the implementation of the final features of the application. This included finalising the linking system, and adding the exercises feature to the workouts to make them more practical. Neither of these features were completed during this week, but both were nearly complete and would be easily completed the next week.

This report document was also created during this week, with the structure being implemented and various sections being completed, such as the Abstract and the Acknowledgements, and the Background, Design and Implementation sections were started.

#### 4.11. Week 13 (21<sup>st</sup> April)

It was decided that this week would be the last week where project development and coding take place. As a result, most if not all time was focused on finalising the app. This includes finishing the client/trainer linking implementation, including the searching with filter feature and the ability for the trainers to see the workouts for multiple different clients. It also includes the completion of the exercises feature for the workouts, allowing for creation, editing and deletion of exercises when creating and editing a workout. Furthermore, this week was also used to create the chat feature that allows clients and trainers to communicate with each other.

During this week, not much report work was carried out as most of the efforts were dedicated to completing the technical submission, however some changes were made to the layout and styling of the document.

#### 4.12. Week 14 (28<sup>th</sup> April) – Final week

With the final implementations complete on the technical side of the project, the final week has been dedicated in its entirety to completing this report in time for the deadline, which is the 3<sup>rd</sup>

of May. This includes completing and finalising all the sections and submitting the completed work before the deadline that is scheduled on Friday. This week also contained major reconstructions, such as including a new top-level section dedicated to the high-level project requirements, as well as both starting and finishing the testing section and the critical evaluation section. The final day, set as Thursday, was dedicated to bulking out the document and adding all the references to the document, which was made much easier thanks to Microsoft Word's citation feature, something I had neglected to use on previous documents.

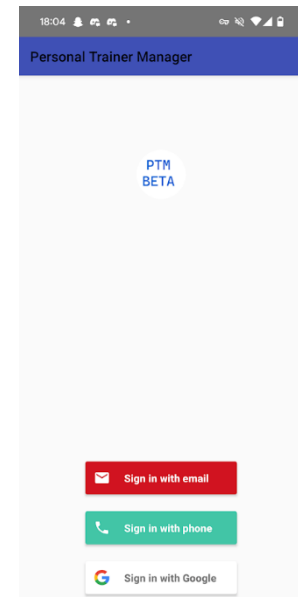
## 5. Implementation

### 5.1. Authentication

It was decided early on that, to achieve the goals of having the application have an authentication method, it would be best suited to have a cloud-based service that provides the easiest and most dynamic authentication solution. This focus was one of the main reasons Firebase [32] was chosen as the cloud storage provider, as they also provide an authentication service that can use various forms of authentication such as Google, Facebook, Twitter, Phone number or Email.

One limitation that occurred as part of this implementation was certain technical requirements from Facebook and Twitter, where they required a business location and email address to allow access to the OAuth APIs, neither of which existed for this project. As a result, these login options could not be featured on the application.

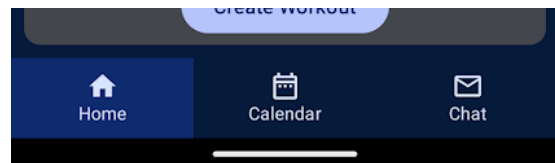
The login screen used is a prebuilt library provided by Google Firebase. This was to save time on the project so that more critical features could be worked on. However, it does not match the presentation and theme of the app, and with more time the app would be better presented with its own login screen. Furthermore, this user interface is outdated and is no longer maintained by Google, and as a result may feature bugs and security risks.



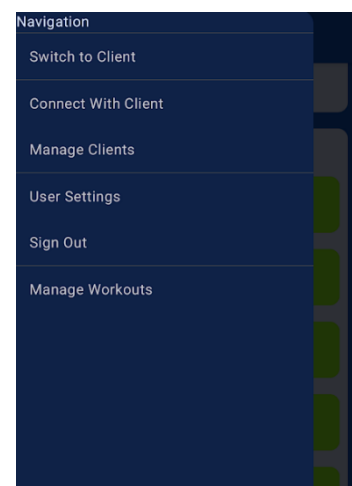
### 5.2. App Navigation

The app can be navigated through multiple methods, giving the users maximum control over what they want to see or do.

The first navigation method is the bottom navigation bar, which shows the 3 core elements of the app: the homepage, which shows the user various information such as their exercises/workouts for the day or next steps in setting up their account; the calendar page, which shows the client all their workouts for the next 2 weeks and shows the personal trainer all the exercises and workouts for their clients over the same period; and the chat page, which allows clients and personal trainers to communicate with each other in real time using text messages.



The second navigation method is the navigation drawer, which gives the user various other pages to navigate to, such as – but not limited to – the user settings page, and the connect with trainers or clients page, which lets users search for their desired counterpart. This drawer slides out from the side of the screen via a button in the top left that is always present, and also gives options such as to switch to the client/personal trainer version of the app, and to sign out of the app.



### 5.3. Data Storage

All data is stored on the Firebase Firestore, a cloud storage solution created and hosted by Google, using its document and collection style NoSQL database. The methods used to access the database are provided by the Firebase Firestore dependency (`com.google.firebase:firebase-firestore`).

To reference the database, a variable instance is created with the value `Firebase.firestore`. This database reference is then used with `.collection(id)` and `.document(id)` to create a query that navigates the file storage and get to the collection or document required for the query. Then, if performing a type of search, a form of `where()` clause is included. For example, `.whereEqualTo(path, thing)` will compare a thing with the value stored in path, and will rule out any documents where the comparison returns false. A limitation was found with the `where()` clauses as they are unable to search for sections of string within strings, something that SQL does have the functionality for. As a result, any time the application needs to filter through a list of clients or trainers, it has to request the entire collection of the requested type and sort through it on the device as opposed to through the query. This can impact the performance of the query as well as the application, and can also have an effect on the Firestore rate limits.

After adding any optional clauses for filtering the results, a command is added to the query. Some commands include `.get()` [33], which retrieves the document or all the documents in the collection that match the `where()` filters, and `.update()`, which can be used to change the values of an existing document but will not work if the document does not exist.

Finally, listeners are added to the end of the query if the data is being retrieved. The `.addOnSuccessListener{}` function allows the retrieved data to be manipulated or other tasks to be completed if the query completes without any errors, and the `.addOnFailureListener` can be added to detect any errors in the query and act accordingly. Listeners can also be used as opposed to commands to retrieve the data any time it is changed [34]. This was particularly useful when creating the chat feature as it allows the user to retrieve the new chats in real time rather than have to manually update the chat messages.

The above code functions were learnt about using the Firebase tutorials found on the Firebase website. These tutorials can be found using the following links:

- Get data with Cloud Firestore [33]
  - This tutorial provided directions on how to write a query that requested documents or collections, and how to interact with that data once retrieved to use it as needed.
- Get real-time updates with Cloud Firestore [34]
  - This tutorial gives beneficial instructions on how to attach listeners to documents and collections so that any changes to the database are automatically sent to the listening function.
- Perform simple and compound queries with Cloud Firestore [35]
  - This tutorial builds upon the first tutorial about getting data from the database by introducing filters to the query, such as `whereEqualTo()`
- Add data to Cloud Firestore [36]
  - Information from this tutorial included a description on each possible method to add data to the database, such as how `update()` only works on existing data whereas `add()` only works with hashmaps of data.
- Delete data from Cloud Firestore [37]
  - This tutorial gave useful instruction on how to remove data from the database. It also addressed a limitation with Firestore on Android, where you cannot delete collections. This is likely due to the risk of errors that can occur during the process, as well as the risk of mistakes when handling large collections of data.

## 5.4. Data Structure

The structure of the data follows a package format, with code separated into packages for an easier to read code base. Each package represents one of the screens. For example, there is a package dedicated to the chat screen, the calendar screen and the settings screen. Furthermore, there is also a package for components, which mainly consists of Enums and Data Classes and are elements of the application that are used in multiple sections. There is also a package for the theme of the UI, which contains the Color.kt, Theme.kt and Type.kt files that control various parts of the UI presentation.

A UML diagram has been produced to represent the flow of connections between files, but its size is too large to be legible in this document and thus a link has been produced so that the diagram is accessible online at [https://www.figma.com/file/XyluW5braKRDSWDfUzoC2f/UML-diagram-\(Copy\)?type=whiteboard&node-id=5442%3A481&t=7bBp318VXEYHb3Za-1](https://www.figma.com/file/XyluW5braKRDSWDfUzoC2f/UML-diagram-(Copy)?type=whiteboard&node-id=5442%3A481&t=7bBp318VXEYHb3Za-1). An alternative way to view the UML diagram is by accessing the png image file named "UML Diagram" saved within the "Documentation" file stored within the technical submission.

## 6. Testing

It was initially planned to include automated testing within the application. However, due to time constraints, it was decided that all of the tests will be manually carried out and the results will be tracked on a test table. With more time to further develop the project, it is likely that automated testing would have been implemented, likely using JUnit [38], Cucumber [39] or other similar testing frameworks suitable for Kotlin.

### 6.1. Test Table

The following test table was created as an alternative to automated testing. The tests were carried out on the latest version of the project application, to eliminate any possible inconsistencies between the table and the final demonstration.

Test No.	Test goal	Test Input	Expected Output	Actual Output	Pass/Fail
1	Assert that user is able to create an account using an email address	Open app Sign up using email address "jak77@aber.ac.uk"	App successfully signs user up and logs user in with provided email address	App successfully signs user up and logs user in with provided email address	Pass
2	Assert that user is able to create an account using a phone number	Open app Sign up using phone number "07546 969715"	App successfully signs user up and logs user in with provided phone number	Log-in interface displays "An unknown error occurred." And sign-in is unsuccessful.	Fail
3	Assert that user is able to create an account using a Gmail account	Open app Sign up using Gmail account	App successfully signs user up and logs user in with provided Gmail account	App successfully signs user up and logs user in with provided Gmail account	Pass

4	Assert that user is able to log out of the application successfully	Open app Sign in using any method Sign out via navigation drawer	App successfully signs user out and returns to the log-in screen	App successfully signs user out and returns to the log-in screen	Pass
5	Assert that user remains logged in after closing the app	Open app Sign in using any method Close app Reopen app	App successfully re-opens and navigates straight to the home screen with the user still signed in	App successfully re-opens and navigates straight to the home screen with the user still signed in	Pass
6	Assert that user is able to log into an existing email account using their email	Open app Log out of app Sign in using existing email	App successfully logs user back into the app	App successfully logs user back into the app	Pass
7	Assert that user is prompted to enter a username when signed in and with no username set	Open app Sign in with an account that has not yet set a username	Homepage contains a prompt asking the user to set a username with a clickable button to do so	Homepage contains a prompt asking the user to set a username with a clickable button to do so	Pass
8	Assert that a user is able to open the user settings	Open app Sign in with an account Navigate to settings page via navigation drawer and button "User Settings"	Settings page opens successfully	Settings page opens successfully	Pass
9	Assert that user is able to change their username via the user settings	Test 7 input Open "change username" dialog via "change username" button Enter different username Press "save"	Username changed successfully	Username changed successfully – always changes the name of the personal trainer even when using the app as a client, never changes the name of the client	Fail
10	Assert that user cannot set their username as an empty string	Test 7 input Open "change username" dialog via "change username" button Enter empty string Press "save"	User is prevented from being able to change their username	User is prevented from being able to change their username	Pass

11	Assert that user can change the application theme in the user settings to Dark Theme	Test 7 input Disable "Use System Theme" switch Enable "Dark Theme" switch	App theme is changed to a dark theme	App theme is changed to a dark theme	Pass
12	Assert that user can change the application theme in the user settings to Light Theme	Test 7 input Disable "Use System Theme" switch Disable "Dark Theme" switch	App theme is changed to a light theme	App theme is changed to a light theme	Pass
13	Assert that user can change the application theme in the user settings to their system theme	Test 7 input Enable "Use System Theme" switch	App theme is changed to the theme of their Android device	App theme is changed to the theme of their Android device	Pass
14	Assert that user's saved theme persists when closing and reopening the application	Test 11 input Close application Re-open application	App theme should remain as a dark theme	App theme should remain as a dark theme	Pass
15	Assert that user is able to switch to "Personal Trainer" mode from "Client" mode	Open navigation drawer whilst using the app in "client" mode Tap "Switch to Personal Trainer"	User mode changes to "Personal Trainer" and app updates its displays accordingly	User mode changes to "Personal Trainer" and app updates its displays accordingly	Pass
16	Assert that user is able to switch to "Client" mode from "Personal Trainer" mode	Open navigation drawer whilst using the app in "Personal Trainer" mode Tap "Switch to Client"	User mode changes to "Client" and app updates its displays accordingly	User mode changes to "Client" and app updates its displays accordingly	Pass
17	<b>Assert that user is able to successfully create a workout whilst using the app.</b>	-	-	-	-
17.1	Assert that user is able to open the "Create Workout" screen via the "Create Workout" button on the homepage	Navigate to homepage Press "Create Workout" button	App navigates to "Create Workout" screen	App navigates to "Create Workout" screen	Pass

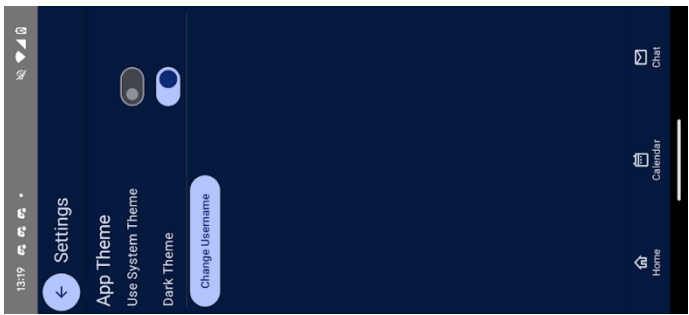
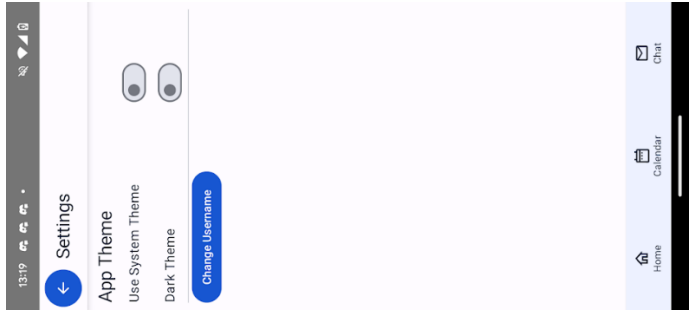
17.2	Assert that user is able to assign a name and a description to the workout	Test 17.1 input Enter name for workout in "Workout name" text box  Enter description for workout in "Workout description" text box	Workout name and description are shown to the user as inputted	Workout name and description are shown to the user as inputted	Pass
17.3	Assert that user is able to change the date of the workout	Test 17.1 input Press "Change Date" button  Set a different date to that which is default	Workout date is changed and shown to the user accurately	Workout date is changed and shown to the user accurately	Pass
17.4	Assert that user is able to change the time of the workout	Test 17.1 input Press "Change Time" button  Set a different time to that which is default	Workout time is changed and shown to the user accurately	Workout time is changed and shown to the user accurately	Pass
17.5	Assert that user is able to toggle whether the workout repeats weekly	Test 17.1 input Press "Repeating" switch	Switch correctly changes to represent if the workout is repeating	Switch correctly changes to represent if the workout is repeating	Pass
17.6	Assert that the user is able to create an exercise for the workout	Test 17.1 input Press "Create Exercise" button  Enter "name" as "Name" in exercise dialog  Toggle "Add Distance" switch to on  Insert 50 into the appearing text box  Press "Confirm"	Dialog closes and created exercise is shown on the "Create Workout" page  Exercise shows the data added to it when tapped	Dialog closes and created exercise is shown on the "Create Workout" page  Exercise shows the data added to it when tapped	Pass
17.7	Assert that, when creating an exercise, the exercise cannot be created without a name	Press "Create Exercise" button  Leave "name" text box blank  Toggle "Add Distance" switch to on  Insert 50 into the appearing text box	"Confirm" button is greyed out and cannot be pressed	"Confirm" button is greyed out and cannot be pressed	Pass

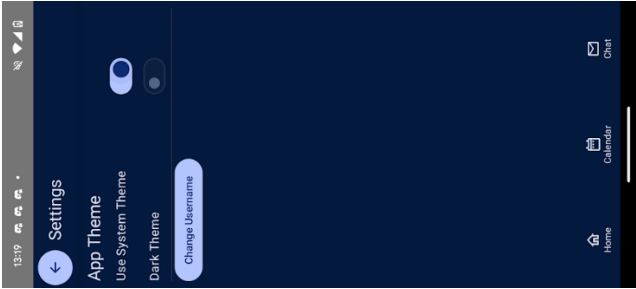

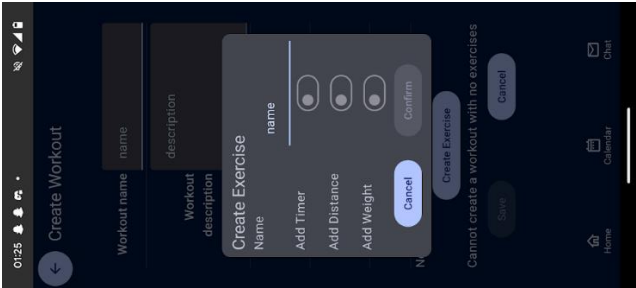
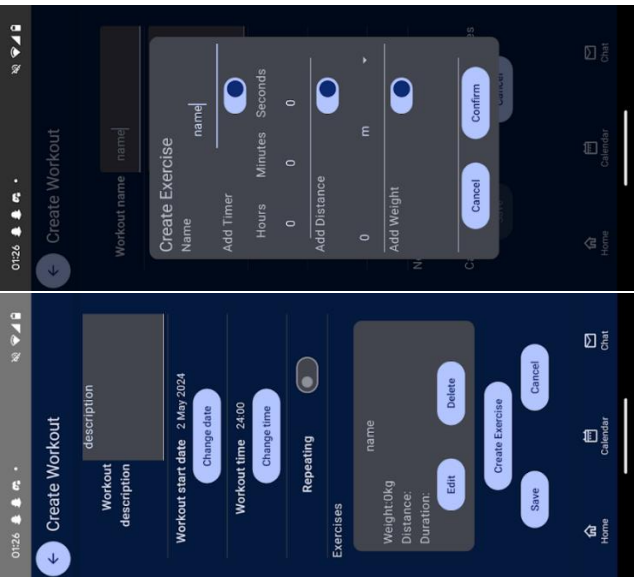


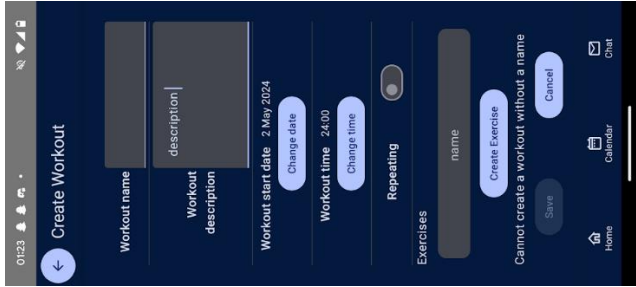
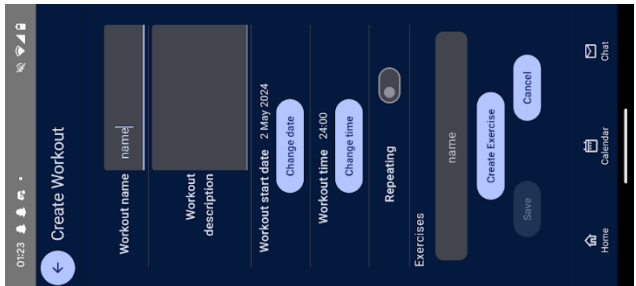

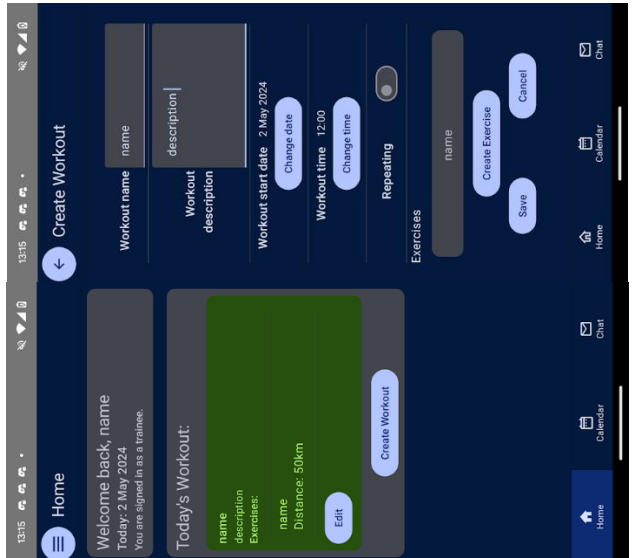
17.8	Assert that, when creating an exercise, the exercise cannot be created without either a timer, distance or weight value set	Press "Create Exercise" button  Enter "name" as "Name" in exercise dialog  Do not toggle any options to on	"Confirm" button is greyed out and cannot be pressed	"Confirm" button is greyed out and cannot be pressed	Pass
17.9	Assert that, when creating an exercise, any values with enabled switches but no values entered are automatically set to 0	Test 17.6 input  Enter "name" as "Name" in exercise dialog  Toggle "Add Distance" to on  Toggle "Add Weight" to on  Toggle "Add Time" to on  Press "Confirm"	Exercise is successfully saved, "Distance", "Weight" and "Time" values are shown on the exercise as 0	Exercise is successfully saved. "Weight" is shown as 0. "Distance" and "Duration" are shown as empty strings.	Fail
17.10	Assert that you cannot create a workout without a set name	Test 17.1 input  Enter "description" in description text box  Test 17.6 input  Do not put text in "Workout Name" text box	"Save" button on workout page is greyed out and inactive, text "Cannot create a workout without a name" is displayed to the user	"Save" button on workout page is greyed out and inactive, text "Cannot create a workout without a name" is displayed to the user	Pass
17.11	Assert that you cannot create a workout without a description	Test 17.1 input  Enter "name" in workout name text box  Test 17.6 input  Do not put text in "Workout description" text box	"Save" button on workout page is greyed out and inactive, text "Cannot create a workout without a description" is displayed to the user	"Save" button on workout page is greyed out and inactive  No text is displayed to the user	Fail
17.12	Assert that you cannot create a workout without an exercise	Test 17.1 input  Enter "name" in workout name text box  Enter "description" in workout description text box  Do not add an exercise to the workout	"Save" button on the workout page is greyed out and inactive, text "Cannot create a workout with no exercises" is displayed to the user	"Save" button on the workout page is greyed out and inactive, text "Cannot create a workout with no exercises" is displayed to the user	Pass

17.13	Assert that workout is visible on homepage after being created	Test 17.1 input Enter "name" in workout name text box  Enter "description" in workout description text box  Test 17.6 input Press "Save" button	Newly created workout is visible on the home screen under "Today's Workouts"	Newly created workout is visible on the home screen under "Today's Workouts"	Pass
18	Assert that workout can be edited on the homepage	Navigate to homepage with existing workout  Expand workout by tapping on it  Press "Edit" button	User is navigated to the "Edit workout" page, which looks the same as the "Create Workout" page but the data fields are pre-filled with the existing workout's data	User is navigated to the "Edit workout" page, which looks the same as the "Create Workout" page but the data fields are pre-filled with the existing workout's data	Pass
19	Assert that workout can be deleted from the "Edit Workout" page	Test 18 input Press "Delete" button	User is navigated back to the home screen  Workout is removed from the list	User is navigated back to the home screen  Workout is removed from the list	Pass

## 6.2. Test Evidence

Test	Evidence	Description
11		This screenshot shows the app in dark theme whilst the dark theme switch is toggled to on
12		This screenshot shows the app in light theme whilst the dark theme switch and the use system theme switch are toggled off

13		This screenshot shows the app in the system theme, which is dark theme, whilst the use system theme switch is toggled to on
17.7		This screenshot shows that an exercise cannot be created without a name, as the Confirm button is disabled
17.8		This screenshot shows that an exercise cannot be created without any variables, as the Confirm button is disabled
17.9		These screenshots show that exercises with all 3 values left as their default inputs do not show the intended result on the create workout page, as the distance and duration values are shown as blank instead of 0

17.10		<p>This screenshot shows that you cannot create a workout without a name as the Save button is disabled. It also shows the text displayed that tells the user that they must set a name.</p>
17.11		<p>This screenshot shows that you cannot create a workout without a description as the Save button is disabled. It also shows that the text informing the user of this does not show, failing the test.</p>
17.12		<p>This screenshot shows that you cannot create a workout without any exercises as the Save button is disabled. It also shows the text informing the user of this.</p>
17.13		<p>These screenshots prove that workouts can be created in the Create Workout screen, and then viewed on the homepage screen.</p>

## 7. Critical Evaluation

### 7.1. Project Requirements

The first evaluation to be made is to compare the project requirements with the project outcome, and to make sure as many goals as possible were achieved. The following table records each project requirement, as well as if the goal was achieved. As shown by the table, all Must Have and Should Have requirements were achieved, which makes the project a success. Unfortunately, due to time constraints, the Could Have goals were not achievable, but they were not mandatory for the project, and were simply additional features, as the main goal of the application was to manage the workouts for personal trainers and clients.

ID	Description	Priority	Achieved?
PR1	The user should be able to access the app through an authentication method, and be able to easily sign in and out of the application.	SHOULD HAVE	Yes
PR2	The user must be able to select what kind of user they want to be, between Client and Personal Trainer.	MUST HAVE	Yes
PR3	The user, whilst using the app as a Client, must be able to create, modify and delete workouts.	MUST HAVE	Yes
PR4	The user, whilst using the app as a Client, should be able to add and remove exercises from workouts.	SHOULD HAVE	Yes
PR5	The user, whilst using the app as a Client, should be able to view their workouts using a calendar view.	SHOULD HAVE	Yes
PR6	The user, whilst using the app as a Client, should be not be able to see workouts created by other users.	SHOULD HAVE	Yes
PR7	The user, whilst using the app as a Client or as a Personal Trainer, must be able to create a "link" with another user that is using the other user type. A Client user should be able to link with a Personal Trainer user, and vice versa.	MUST HAVE	Yes
PR8	The user, whilst using the app as a Client or as a Personal Trainer, should be able to communicate with other users that they have linked with using a form of text-based communication within the application.	SHOULD HAVE	Yes
PR9	The user, whilst using the app as a Personal Trainer, must be able to create, edit and delete the workouts of Client users that they have created a link with.	MUST HAVE	Yes
PR10	The user, whilst using the app as a Personal Trainer, should be able to add, edit and remove exercises from	SHOULD HAVE	Yes

	the workouts of Client users that they have created a link with.		
<b>PR11</b>	The user, whilst using the app as a Client, could be able to work through their workouts in real-time.	COULD HAVE	No
<b>PR12</b>	The user, whilst using the app as a Personal Trainer, could be able to watch the progress of the client's workouts in real-time.	COULD HAVE	No

## 7.2. Methodology

Using the Kanban methodology, I was able to work fluidly without having to restrict myself to deadlines, which allowed me to spread my workload out and avoid burnout from overworking myself. However, this did have a negative impact in that I wasn't always able to deliver what was promised weekly between the meetings with the project supervisor. For example, I had initially discussed that I would develop a prototype for the application using the design tool Figma [40]. However, due to the fact that I never set myself an official deadline for this task, it was pushed aside for other tasks, and eventually the task was never completed. It could be argued that this also occurred due to a lack of prioritisation with tasks, something that I would have done in hindsight as it caused me to often focus on the wrong tasks and disregard those of higher importance.

An alternative to Kanban would be to use a Scrumban (Scrum + Kanban) methodology, which combines the usefulness of Kanban with the sprints of Scrum. This is beneficial as it ensures that the right tasks are delivered on the right time. Furthermore, due to my year in industry, I worked on multiple software development projects that employed the Scrumban methodology, so I have experience working in this way.

## 7.3. Documentation

Whilst working on the project, I did not document much information about my progress. The only metric measuring what I achieved each week of the project is the weekly supervisor meeting minutes which are provided by the supervisor after every meeting. In hindsight, it would have been far more forward-thinking to have tracked my progress in another method, such as a daily/weekly blog, ideally with screenshots of the issue board hosted on GitLab. This was suggested by the project supervisor, and I regret not taking their advice. Due to this, the Development Process section is underdeveloped and may contain some inaccuracies.

Another way to improve the documentation of the project is to start on documentation earlier. For example, starting the report earlier would allow for more time to flesh it out and improve the quality and the quantity of the content. However, starting earlier could lead to sections having to be rewritten, costing critical time that could be dedicated towards improving the quality of the technical submission.

## 7.4. Testing

When starting this project, my original intention was to include automated testing alongside the project as a demonstration of my understanding of the subject. However, due to the demands of other tasks, as well as the poor prioritisation discussed in 7.2, I was unable to develop any automated tests for the project. If given the opportunity, I would have liked to have used a combination of JUnit [38] and Cucumber, as Cucumber has an Android branch [39] that can be found on GitHub. The

benefits of automated testing are significant, as it allows development to be made without having to manually re-test every possible instance every time. It also supports continuous development pipelines, a feature of GitLab which could be exploited for further gain when developing new features for the application.

## 7.5. What to do next

If given the opportunity to continue developing the project, the next objective would be to implement the functions that allow the users using the app as clients to work through the workouts in real-time, and the users using the app as personal trainers to view the clients doing this on their own device.

A second implementation that would be included in the next iteration of development would be the beginnings of automated testing, as that is a feature that benefits future development greatly. Completing this task would improve the accuracy of the testing and save time in the future as automated tests can be applied to CI/CD pipelines, which conveniently is a feature of GitLab, the code hosting and versioning service being used for this project.

Alternatively, and arguably more important, the next route to take would be to fix a lot of the critical bugs. There are some failed tests documented in the test table which could be fixed if given the time, and some data is still not handled correctly throughout the app. Equally, there are some graphical inconsistencies, as some UI elements are not as polished as others and could do with a tune-up to improve the visual quality of the application.

The final decision would be to rework a lot of the existing systems, as my development methods and techniques changed over the course of the development of the project in improvement methods, and as such some functions are outdated and use weaker techniques. For example, during the development of the firebase manager functions, I initially obtained data from the database snapshots using `.getField<String>(path)`. However, a simplified method exists that I previously did not identify, which is `.getString(path)`. This both tidies the code and enforces type-specific methods, which improves the robustness of the code.

## 8. References

- [1] Android, "Android Homepage," 2024. [Online]. Available: [https://www.android.com/intl/en\\_uk/](https://www.android.com/intl/en_uk/). [Accessed 02 May 2024].
- [2] Kotlin, "Kotlin Homepage," 2024. [Online]. Available: <https://kotlinlang.org>. [Accessed 02 May 2024].
- [3] Wikipedia, "Android (operating system)," 30 April 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)). [Accessed 02 May 2024].
- [4] Wikipedia, "Kotlin (programming language)," 19 April 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Kotlin\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Kotlin_(programming_language)). [Accessed 02 May 2024].
- [5] JetBrains, "JetBrains Homepage," 2024. [Online]. Available: <https://www.jetbrains.com>. [Accessed 02 May 2024].
- [6] Microsoft, "Azure Homepage," 2024. [Online]. Available: <https://azure.microsoft.com/en-gb>. [Accessed 09 February 2024].
- [7] Amazon, "AWS Homepage," 2024. [Online]. Available: [https://aws.amazon.com/?nc2=h\\_lg](https://aws.amazon.com/?nc2=h_lg). [Accessed 09 February 2024].
- [8] Cloudflare, "Cloudflare Homepage," 2024. [Online]. Available: <https://www.cloudflare.com/en-gb/>. [Accessed 09 February 2024].
- [9] Google Firebase, "Firebase Homepage," 2024. [Online]. Available: <https://firebase.google.com>. [Accessed 09 February 2024].
- [10] Wasabi, "Wasabi Homepage," 2024. [Online]. Available: <https://wasabi.com>. [Accessed 09 February 2024].
- [11] Microsoft, "CosmosDB," 2024. [Online]. Available: <https://azure.microsoft.com/en-gb/products/cosmos-db>. [Accessed 09 February 2024].
- [12] Amazon, "SimpleDB," 2024. [Online]. Available: <https://aws.amazon.com/simplydb/>. [Accessed 09 February 2024].
- [13] Cloudflare, "R2 Storage Buckets," 2024. [Online]. Available: <https://www.cloudflare.com/en-gb/developer-platform/r2/>. [Accessed 09 February 2024].
- [14] Google Firebase, "Cloud Firestore," 2024. [Online]. Available: <https://firebase.google.com/docs/firestore>. [Accessed 09 February 2024].
- [15] Wasabi, "Hot Cloud Storage," 2024. [Online]. Available: <https://wasabi.com/hot-cloud-storage/>. [Accessed 09 February 2024].



- [16] Wikipedia, "Lean software development," 17 April 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Lean\\_software\\_development](https://en.wikipedia.org/wiki/Lean_software_development). [Accessed 02 May 2024].
- [17] Wikipedia, "Scrum (software development)," 02 May 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Scrum\\_\(software\\_development\)](https://en.wikipedia.org/wiki/Scrum_(software_development)). [Accessed 02 May 2024].
- [18] Wikipedia, "Kanban (development)," 06 March 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Kanban\\_\(development\)](https://en.wikipedia.org/wiki/Kanban_(development)). [Accessed 02 May 2024].
- [19] GitLab, "GitLab Homepage," 2024. [Online]. Available: <https://about.gitlab.com>. [Accessed 02 May 2024].
- [20] GitHub, "GitHub Homepage," 2024. [Online]. Available: <https://github.com>. [Accessed 11 March 2024].
- [21] J. Shah, "GitHub vs GitLab: Which is the Best in 2024?," Radix, 29 February 2024. [Online]. Available: <https://radixweb.com/blog/github-vs-gitlab>. [Accessed 28 April 2024].
- [22] ProductPlan, "MoSCoW Prioritization," 2024. [Online]. Available: <https://www.productplan.com/glossary/moscow-prioritization/>. [Accessed 30 April 2024].
- [23] Android, "Build better apps faster with Jetpack Compose," 2024. [Online]. Available: <https://developer.android.com/develop/ui/compose>. [Accessed 02 May 2024].
- [24] A. Shokoya, "XML Views or Jetpack Compose: Which is The Best Option For Your Next Project?," Medium, 22 September 2023. [Online]. Available: <https://medium.com/@MeenoTeK/xml-views-or-jetpack-compose-which-is-the-best-option-for-your-next-project-ccf38573a82>. [Accessed 28 April 2024].
- [25] Gradle, "Gradle Homepage," 2024. [Online]. Available: <https://gradle.org>. [Accessed 02 May 2024].
- [26] Android, "Android Studio," 2024. [Online]. Available: <https://developer.android.com/studio>. [Accessed 01 May 2024].
- [27] Microsoft, "What is OAuth?," 2024. [Online]. Available: <https://www.microsoft.com/en-gb/security/business/security-101/what-is-oauth>. [Accessed 02 May 2024].
- [28] Material Foundation, "Material Theme Builder," 12 December 2023. [Online]. Available: <https://material-foundation.github.io/material-theme-builder/>. [Accessed 16 February 2024].
- [29] Android, "Navigation Drawer," 30 April 2024. [Online]. Available: <https://developer.android.com/develop/ui/compose/components/drawer>. [Accessed 15 February 2024].
- [30] L. Schaefer, "NoSQL vs. SQL Databases," MongoDB, 2024. [Online]. Available: <https://www.mongodb.com/nosql-explained/nosql-vs-sql>. [Accessed 01 May 2024].

- [31] Android, "Side-effects in Compose," 30 April 2024. [Online]. Available: <https://developer.android.com/develop/ui/compose/side-effects>. [Accessed 03 May 2024].
- [32] Google Firebase, "Firebase Authentication," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/auth>. [Accessed 13 February 2024].
- [33] Google Firebase, "Get data with Cloud Firestore," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/firestore/query-data/get-data>. [Accessed 12 March 2024].
- [34] Google Firebase, "Get realtime updates with Cloud Firestore," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/firestore/query-data/listen>. [Accessed 28 April 2024].
- [35] Google Firestore, "Perform simple and compound queries with Cloud Firestore," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/firestore/query-data/queries>. [Accessed 03 March 2024].
- [36] Google Firebase, "Add data to Cloud firestore," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/firestore/manage-data/add-data>. [Accessed 22 February 2024].
- [37] Google Firebase, "Delete data from Cloud Firestore," 30 April 2024. [Online]. Available: <https://firebase.google.com/docs/firestore/manage-data/delete-data>. [Accessed 24 February 2024].
- [38] Android, "Test your Compose layout," 30 April 2024. [Online]. Available: <https://developer.android.com/develop/ui/compose/testing>. [Accessed 02 May 2024].
- [39] Cucumber, "cucumber-android," GitHub, 22 December 2023. [Online]. Available: <https://github.com/cucumber/cucumber-android>. [Accessed 02 May 2024].
- [40] Figma, "Figma," 2024. [Online]. Available: <https://www.figma.com>. [Accessed 02 May 2024].

## 9. Appendices

### A. Use of Third-Party Code, Libraries and Generative AI

#### A.1. Third-Party Code

#### A.2. Libraries

Plugin Name	Source/Version	Usage
<b>Android core</b>	androidx.core:core-ktx:1.13.0	Android Core functionality
<b>Android Lifecycle</b>	androidx.lifecycle:lifecycle-runtime-ktx:2.7.0	Android Lifecycle functionality
<b>Android Activity</b>	androidx.activity:activity-compose:1.9.0	Android Activity functionality for Jetpack Compose
<b>Android Navigation</b>	androidx.navigation:navigation-compose:2.7.7	Navigation functionality for Jetpack Compose, used in the app e.g. NavHostController, NavController
<b>Android Compose Elements</b>	androidx.compose:compose-bom:2024.04.01	BOM* for managing the versions of the other dependencies as part of Compose
	androidx.compose.ui:ui	Compose UI element, provides various UI elements such as Alignment, Alignment.Horizontal, Alignment.Vertical and Modifier, elements used in all parts of the app.
	androidx.compose.material3:material3	Provides a wide range of Material3 elements providing most of the app's functionality and appearance, such as ModalNavigationDrawer, Surface, HorizontalDivider and Card
<b>Firebase Elements</b>	com.google.firebase:firebase-bom:32.8.0	BOM* for managing the versions of the other dependencies as part of Firebase
	com.google.firebase:firebase-firestore	Firestore dependency that provides all the functionality needed to connect and interact with the cloud storage service

	com.google.firebase:firebase-auth	Authentication dependency that provides the methods required to log users in and out, and gives access to their unique user IDs that can be applied to the Firebase Firestore
<b>Firestore Auth UI</b>	com.firebaseui:firebase-ui-auth:7.2.0	Pre-built Firebase authentication interface used in the app
<b>Google Play Authentication</b>	com.google.android.gms:play-services-auth:20.7.0	Dependency required by the Firestore Auth UI
<b>Facebook SDK</b>	com.facebook.android:facebook-android-sdk:8.2.0	Facebook SDK dependency required by the Firestore Auth UI – Facebook authentication not actually used
<b>JUnit</b>	junit:junit:4.13.2	Framework used for automated testing
<b>Android JUnit plugin</b>	androidx.test.ext:junit:1.1.5	Allows JUnit tests to operate on Android functions
<b>Android Espresso plugin</b>	androidx.test.espresso:espresso-core:3.5.1	A user-friendly Android UI automated testing framework similar to Cucumber.
<b>Android Compose Test Elements</b>	androidx.compose:compose-bom:2024.03.00	BOM* dependency that controls the versions
	androidx.compose.ui:ui-test-junit4	JUnit dependency that supports testing on jetpack compose elements of the app
	androidx.compose.ui:ui-tooling	Other testing dependency for jetpack compose
	androidx.compose.ui:ui-test-manifest	Other testing dependency for jetpack compose

\* BOM (Bill of Materials) – A special kind of POM (Project Object Model) that is used to define and maintain the versions of a project's dependencies.

## B. Cloud Storage Solutions Analysis

### B1 Requirements

- **MUST be free**  
As this is a project not initially intended to be marketed publicly, and as there is no initial source of finance for this project, the demand for storage and API solutions will be very small, and as such the solution must be free of any charges, up front or otherwise.
- **MUST provide suitable storage capacity**  
The capacity required for this application is yet to be determined, so having a large enough capacity for data will be beneficial in the development and testing of the project.
- **MUST be fast and responsive**  
A storage solution that is slow and inconsistent with its responses may cause major problems with the app performance and may lead to issues with development.
- **SHOULD be backed up**  
If the solution backs up its data, it is far more secure and the risks of clients losing data due to a corruption or other error are far slimmer.
- **CAN provide authentication services**  
Being able to integrate authentication with data access could simplify development exponentially, allowing development time to be focused on other areas.

### B2 Candidates

- Azure (Microsoft) <sup>[1]</sup>
- Amazon Web Services/AWS (Amazon) <sup>[2]</sup>
- Cloudflare <sup>[3]</sup>
- Firebase (Google) <sup>[4]</sup>
- Wasabi <sup>[5]</sup>

## B3 Analysis Table

	<i>Azure</i>	<i>AWS</i>	<i>Cloudflare</i>	<i>Firebase</i>	<i>Wasabi</i>
<i>Storage Option</i>	Cosmos DB <sup>[6]</sup>	SimpleDB <sup>[7]</sup>	R2 Object Storage <sup>[8]</sup>	Cloud Firestore <sup>[9]</sup>	Hot Cloud Storage <sup>[10]</sup>
<i>Storage Type</i>	NoSQL or PostgreSQL	NoSQL	Object Buckets	NoSQL	Object Buckets
<i>Price</i>	Free <sup>[11]</sup>	Free <sup>[12]</sup>	Free <sup>[13]</sup>	Free <sup>[14]</sup>	Free (30 day trial) <sup>[15]</sup>
<i>Storage Capacity</i>	25GB	1GB/month	10GB/Month	1GiB	1TB
<i>Rate Limits</i>	1,000 calls per second	25 hours of usage per month	1 million calls/month (modifying) 10 million calls/month (reading)	50,000 reads, 20,000 writes, 20,000 deletes per day	1,000 GET 100 PUT 100 POST 10 DELETE Per minute
<i>Backed Up</i>	Yes	Yes	No	Paid (charged per GiB/month)	Yes
<i>Account Integration</i>	-	-	-	Google	-
<i>App Development tools</i>	Few	Yes	No	Yes	No
<i>Have I used it before?</i>	No	Yes – Year in Industry	No	No	No

## B4 Conclusion

In conclusion, Firebase will be used as it provides the most support with Google integration. With this, I can incorporate it with Firebase authentication for the most efficient and effective solutions to both problems, without having to spend any funds. The fact that the backups are a paid feature is not detrimental to the decision as there will only be limited amounts of data stored and very small amounts of it, if any, will be critical to the app's operation.