PROJECT FINAL REPORT

BSc Computer Science

GymInSync: A Mobile Personal Training Application

Author: Amir Mashallah Ali

Supervised by: Christopher Treglohan

TABLE OF CONTENTS

Table of Contents

1 Preface

1.1 Abstract

The aim of this report is to outline and discuss the steps taken in developing a functional fitness application with a user-friendly design to manage the fitness journey of users.
The main goal of the application is to allow users to track their gym progress, as well as facilitate educational empowerment by empowering users with fitness knowledge which would enable them to make informed decisions on a day-to-day basis with their workout routines.
Throughout this report, it will detail the development process taken in order to achieve the initial goals, and to conclude and evaluate whether the application has met the expectations.

1.2 Introduction

A lot of mistakes were made during my gym journey. And as I have been going to the gym for a few years now, I have met and seen many people struggle to progress and keep up with their gym plan.
Especially in recent times. Gym and fitness have seen a big increase in the recent years due to social media. Personal trainers have been gaining a huge following on social media, and it is safe to assume that the overall number of gym memberships has been increased in the past few years due to the fact that gym and fitness is one of the most followed trends due to the promotion of fitness by celebrities (*Statesman News Service (2022)).*
Many believe that just attending the gym would help with to achieve their goals, but they don't realise what exercises to do perform and how to perform them in order to target certain muscles.
This is a big problem in the gym and fitness industry as it may lead to possible injuries and demotivation if progress is not seen. Witnessing the difficulties of others to find effective solutions and convenience to achieve the desired goal motivated me to create an application dedicated to help new gym members, so they can understand what exercises should be implemented in their gym routine, and to help them train, and to track their progress.

A fitness application like mine which is developed with the view of enhancing the user experience through friendly designs and educational empowerment is based on current research and the most perfect practices that technology and fitness education offer. Individuals with struggles and health conditions such as fibromyalgia can really benefit from an application like this because they use mobile apps to manage their health *(J. An (2024)).*
This paper, as a theoretical frame of reference and practical guidelines drawn from the use of technical references, seeks to be helpful in mapping out the challenges of app development and the optimal way an application developed can add value for its users.

1.3 Motivation

There was an interest in app development prior to this project. Different coding languages were performed in previous assignments and private projects such as Python, Java, JavaScript, HTML, CSS etc.
One of the core technical choices in the development of your fitness application was the use of React Native. This really befits the current best practices in cross-platform mobile development, which offer efficiency without necessarily compromising performance. The most prominent aspect of Facebook in React Native documentation is being capable of building native applications that leverage development time and resources from a single JavaScript codebase, thus ensuring a perfect experience of its users on both iOS and Android platforms *(Facebook, 2021).* Moreover, studies such as "Evaluating React Native: From a Mobile Developer's Perspective*" (Hansson, A., et al., 2017)* demonstrate that React Native offers improved developer productivity and performance for them to remain competitive.

Creating a prototype gym application as a project would keep me motivated and interested throughout. This was important to me as through previous experience, that lack of motivation has had a real effect on the quality of work produced due to not being passionate about the end goal.

The learning curve for the project ended up being sufficiently steeper than first imagined, due to the number of problems faced with the coding from start to finish.
In the end it was worth it as new knowledge and skill was learnt on how to use React Native was obtained, as well as how to style an app professionally, storing data in the app. Alongside development skills, time managements skills have also been improved as well as prioritising tasks based on importance.

1.4 Aims

The aims for the project are many but the main aim was to create and develop a prototype gym mobile application which has educational content to help users with their fitness journey. Other than the main aims, additional aims that was visioned were a few that revolves around three key directions, which is:

- User engagement
- Educational value
- Information security

One of which was to create an user-friendly to allow users to navigate through the application easily, track their progress, take pictures, and give fitness and nutrition advice by text and video. The knowledge that the app supplies would enable users to make informed decisions on a day-to-day basis with terms of dietary choices and workout routines. This is achievable by a proper interactive learning program within the app.

In order to achieve information security, the privacy and security of the users will be ensured by implementing robust security measures. This will include the encryption of sensitive personal data such as fitness data.

While working through the prototype, new skills will be learned when progressing through the development such as:

- Learn and understand React Native
- Learn and understand Expo
- Learn and understand security protocols
- Learn and understand testing
- Learn and understand the development of mobile applications such as the front-end, back-end, designing, functions, limitations, databases and more.

1.5 Acknowledgements

I want to express my gratitude to my supervisor Christoper Treglohan who has helped out with the approach and the vision of the GymInSync prototype. Without the assistance, I would not have known on when to start the development and where to exactly start the research of the project.

2 Background

2.1 Literature Review

Personal fitness with integrated technology has become more pronounced with the rise of health consciousness globally and the growth and social media platforms of fitness influencers. People in their day-to-day lifestyles are more exposed to health and fitness content such as on Instagram and TikTok. This becomes a major factor in the trend, accompanied by celebrities and personal fitness trainers commanding very large followings and at a large scale. They have a big role in promoting the use of gym memberships and fitness applications, which have grown enormously *(Statesman News Service, 2022).*
From such a perspective, it becomes quite obvious that the relations between social media, celebrity influence, and personal fitness activity are changing, which therefore implies the necessity to provide a much more detailed analysis of how exactly technology might be used to improve the experience in keeping fit and track of progress.

Regardless of the growing number of fitness interests, a good number of fitness members face a lot of challenges moving forward with their fitness, with reasons ranging from the lack of information on the exercises to the guidelines to apply. This gap shows that the education through fitness applications can enable the users to have access to valid information concerning their workout routines and dietary habits. The growth of user-friendly fitness apps imparting high value in learning and tracking progress could play an important role in users reaching their fitness goals in a safe and effective manner.

In terms of the technical aspects of the prototype, the type of development framework used can decide the functionality of the app and the user experience. fully respects the selection of a technical framework to choose one in the application for deciding on the functionality and user experience of the application. Currently React Native technology is at the top in the cross-platform development of mobile applications and is capable of building from a common JavaScript codebase which is very efficient for applications *(Hansson et al., 2017).* This framework makes it easy to develop such applications that work properly on the platforms of both iOS and Android, which is very important by the fact of considering the diverse users' base for fitness applications. Besides, it matches very well with modern best practices in deployment of technology, since it makes the transition of user experience between devices very smooth.

These are fitness applications that handle very sensitive user data, like health metrics and personal tracking of activities. The security of this data and its privacy is an aspect to be considered. Therefore, it would require a high level of security with regard to data encryption and strong authentication protocols in order to avoid unauthorized access or breaching that might lead to data compromise. On the success of an application and trust from its users, it mainly depends on the protection of the personal information of the users. This puts much emphasis on first guaranteeing security during the application's construction process.

But individuals may face technostress when there is too much technology to adapt to *(A. Aminia. 2024)* But this application should be easy to follow for everyone to avoid some users from suffering from technostress.

The growth in fitness apps is sort of an interesting opportunity for leveraging technology in personal health improvement. It can be quite important for the developers to make software that would contribute and perhaps enhance user fitness experience to the overall well-being and knowledge of the user by focusing on user-friendly design, educational empowerment, and robust security measures. Future research and development work, with technological advancement, can prove very handy in solving the new problems and needs arising in the fitness world.

2.2 Tools

Several tools were used for the prototype, and it will be enlisted in this section with some background information about each and why it was used. This will give the reader a better understanding of the overall vision of the project.

2.2.1 React Native

React Native, first released in 2015 and was developed by Meta, is a framework for building native apps using JavaScript for both iOS and Android. These apps are native which means that they are not web apps that look like mobile app. Android and iOS programming is not required in this case. In React Native, application code can all be written in JavaScript and be shared across IOS and Android. This is one of the main advantages, as well as live updates. Whenever the code is changed, it can be viewed instantly without the need to refresh. This will make the development of the application more efficient.
Without React Native, a new code base for iOS will have to be created, which is written in Swift or Objective-C, and another for Android which is written in Jaca or Kotlin. This would take longer and make it much more complicated than it needs to be.

2.2.2 Expo and Expo Go

Expo is a set of tools and a framework that sits on top of React Native and hides a lot of complexity from the developer. It has everything that is required to build an app without the need to deal with native code. This makes it faster and easier to build React Native apps for me since no prior knowledge of mobile development was attained.
This comes with Expo Go which is a mobile app that can be installed to enable me to view the project on a real device and test it.
A con about Expo is that it limit what Expo gives in terms of native features, but this shouldn't be a big problem because Expo does offer a lot of native features, therefore a real complete app can still be developed.

2.2.3 Node.js

Node.js is an open-source and cross-platform runtime environment for executing JavaScript Code outside the browser. It is used to build back-end services known as API (Application Programming Interface). These services power client apps which in this case is a mobile application. This client sees and interacts with the app, but this app is just the surface. The app needs to communicate with services that sit on the server to store data.
Node.js is suitable for high-scalable, data-intensive and real-time apps.
Other software could have been used instead of Node.js such as ASP.NET. But Node.js is great for prototyping and agile development which would be perfect for me.

2.2.4 Visual Studio

Visual Studio is an IDE (Integrated Development Environment) created by Microsoft, and it can be used to develop several programs such as web apps, web services, and for this instance, mobile apps. Visual Studio can support several programming languages which makes it very versatile, and in this case, it will be used for JavaScript. Visual Studio is used because it provides a lot of extensions and shortcuts which can be useful as app is developed.

2.2.5 Firebase

Firebase is a real-time database that can be used to store data for the app. The authentication system is secure, and it is easy to use which is suitable for a newcomer to mobile application development.

2.3 Application Comparison

Before starting the development of GymInSync, first the analyses of other existing gym apps needs to be completed to help out with the design and functionality of the project. The two existing mobile apps that was looked at were Pump Gym App and FitNotes App. These three apps are unique in their own perspective and can be the key to the success of the GymInSync gym app.

2.3.1 Pump Gym App

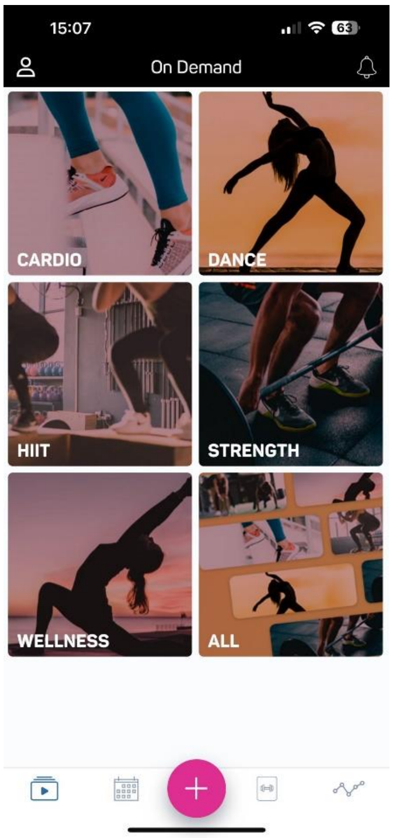


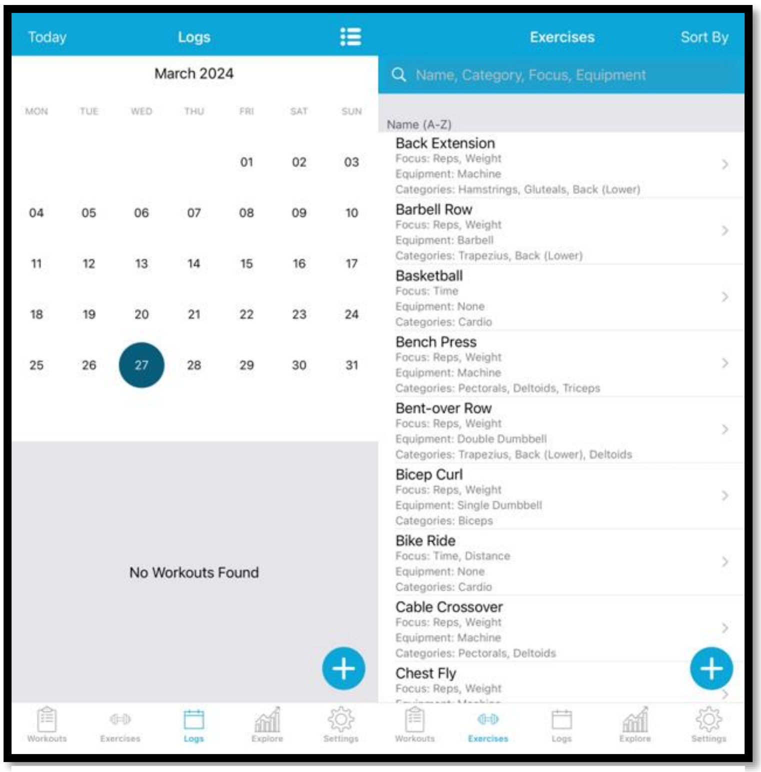
Figure 1 - Pump Gym main screen on mobile application.

The Pump Gym application has a nice and simple main screen with six cards. One of the features of the main screen is that each card has a picture to illustrate the title of the cards. It clearly categorizes the workouts and health. This, perhaps makes it look more visually appealing therefore can engage users.

Also, the navigation bar is very minimalistic and not too complex. T features very beneficial and can be used for GymInSync.

The tab on the top left can allow users to edit their details and perhaps get a premium version of the app. This will not be considered for this project because the prototype will focus more on the educational aspect and tracking progress. But perhaps this feature can be used to have more options.

2.3.2 FitNotes App



Name (A-Z)

Back Extension
Focus: Reps, Weight
Equipment: Machine
Categories: Hamstrings, Gluteals, Back (Lower)

Barbell Row
Focus: Reps, Weight
Equipment: Barbell
Categories: Trapezius, Back (Lower)

Basketball
Focus: Time
Equipment: None
Categories: Cardio

Bench Press
Focus: Reps, Weight
Equipment: Machine
Categories: Pectorals, Deltoids, Triceps

Bent-over Row
Focus: Reps, Weight
Equipment: Double Dumbbell
Categories: Trapezius, Back (Lower), Deltoids

Bicep Curl
Focus: Reps, Weight
Equipment: Single Dumbbell
Categories: Biceps

Bike Ride
Focus: Time, Distance
Equipment: None
Categories: Cardio

Cable Crossover
Focus: Reps, Weight
Equipment: Machine
Categories: Pectorals, Deltoids

Chest Fly
Focus: Reps, Weight

Figure 2 - logs screen and exercises screen from FitNotes App.

The FitNotes application has features that can be very fitting for the gym app prototype. Firstly, it has a calendar. This feature can be implemented within the project so the user may enter statistics in depending on what day they wish. Another remarkable aspect of this application is the feature to search for exercises. The FitNotes app has all the exercises and sports, but this could be used for just the main gym exercises that are commonly performed. Perhaps a favourite button can be added to save the exercise, and maybe a "plus" button to add the exercise to workout day. For example, add the bench press to Monday. This may seem insignificant, but it would be vital to help users plan their workouts and to understand which muscles are trained with different exercises.

3 Pre-Development Setup

Before starting with the development of the gym application prototype, a plan was required otherwise I would not be organised and would not know what to do in the present and future.

3.1 Pre-Development

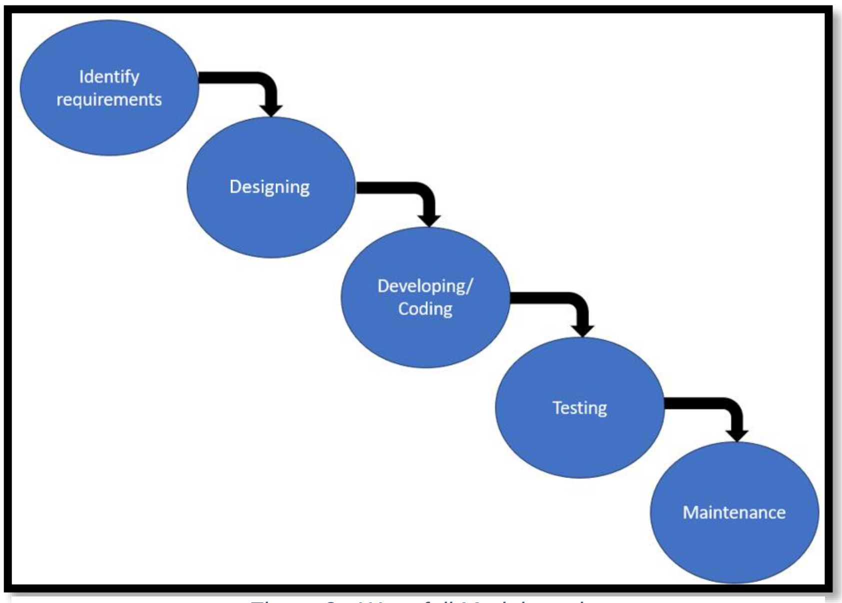


Figure 3 - Waterfall Model used.

The Waterfall Model is one of the oldest software development approaches. This approach progresses in a series of stages in such a way that each stage provides an input to the next stage. This can help me to be more organised and can save a lot of time when developing the prototype as it gives me a plan to follow. But this does also have issues as it will not be flexible with the coding and designing.

First step is to "Identify requirements"; this is the base phase in which the goals of the project are set, regarding functionality and expectations from the user, as well as software needed to develop the application prototype.
The next step is “Designing” where the application will be designed before any development.

After “Designing”, the "Developing/Coding," stage will start. This means that the actual developing and coding activities carried out based on the defined requirements and design documentation. The next phase is "Testing," in which the intent is to make sure that the code is compliant with the requirements, and it finds any bugs or possible pitfalls and fix them.
The final stage of "Maintenance" involves continued support and bug fixes, along with any updates provided to the software that come after the deployment phase.

Even though this model is suitable for the project that was planned, it did not serve as expected. It did not achieve everything that was planned with this project so half-way through the development of the prototype, the focus was solely on the development of the application rather than anything else.

So instead of the waterfall model, a new model was followed which looked like Figure 4. It is not like the agile methodology, but it does incorporate some elements of the agile methodology as it emphasises on the iteration and continues improvement.

This allows a style to me set which is more flexible with the development of the prototype, and to quickly adapt the code to change. But at the same time, following this model would cause issues such as unorganised tasks and I would focus more on the quality on singular function rather than the whole general prototype.

If this model was followed instead of Figure 3, more would have been achieved terms of progress and quality with the prototype.

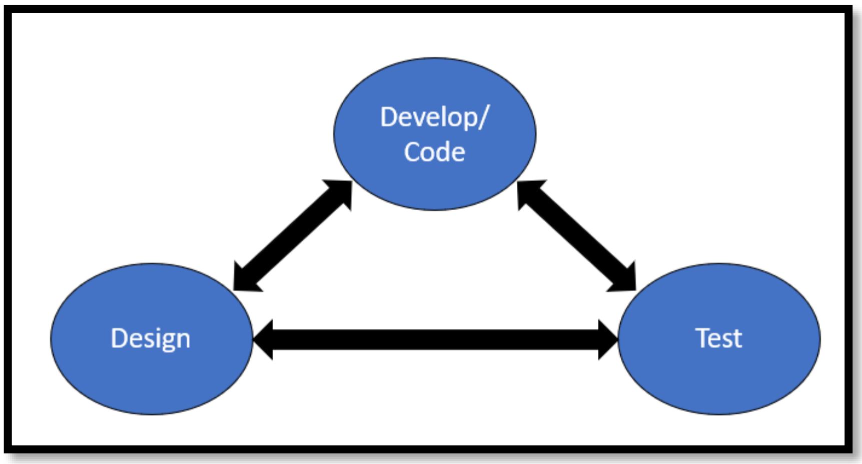


Figure 4 - Model used.

3.2 Required Functions

In order to develop and design the gym prototype application, first the functions had to be identified to make the app work to achieve the aims. If this was not done at the start, it would make the whole progression of the development more difficult as there will be no vision.

- The user of the application should be able to view the main screen. This screen should have tabs to allow the user to navigate between screens. Each screen has a different function.
- One of the screens should have a list of the exercises with pictures. The user should be able to favourite the exercises that they like. These favourites can be viewed in the favourites screen. And users should also be able to add exercises to the desired dates of the week. For example, the user should be able to add the bench-press exercise to their Monday workout plan.
- Users should be able to add statistics to a diary such as personal bests in terms of strength and weight. This would allow users to track their progress over time.
- When statistics are added, they can view it in their history screen. This screen will display a visual graph so the user can see if they are doing better or worse than before. It will have different graphs, one for each statistic such as weight, bench-press personal best, body fat percentage etc.
- Another feature that the gym prototype will feature is a round-timer. This round-timer can assist users with certain exercises such as HIT (High Intensity Training) sessions, boxing, circuits etc.
- Users should be able to add their goal(s) in a section. This allows users to have an aim on what to train on. This could be a certain body weight, body fat percentage, or a personal best at an exercise perhaps.

3.3 Colour Scheme

After understanding what needed to be done with the application in terms of functionality, the gym app structure needed to be designed. Before this was done, different gym interiors were analysed and compared in order to get inspiration.

3.3.1 Buzz Gym

The first gym that was analysed was the gym that I used to attend. This game was called Buzz Gym, and their main colour was green. Firstly, the colour is a bit off as it looks more of a colour that fits an app about the environment which therefore might not correlate fully with a gym atmosphere. Their interior had a blend of other colours such as white, black, and grey which gives a sleek feeling when you enter the gym.
Overall, the gym gives off a traditional feeling that is usually associated with gym environments, but it just deviates a bit because of the green colour.

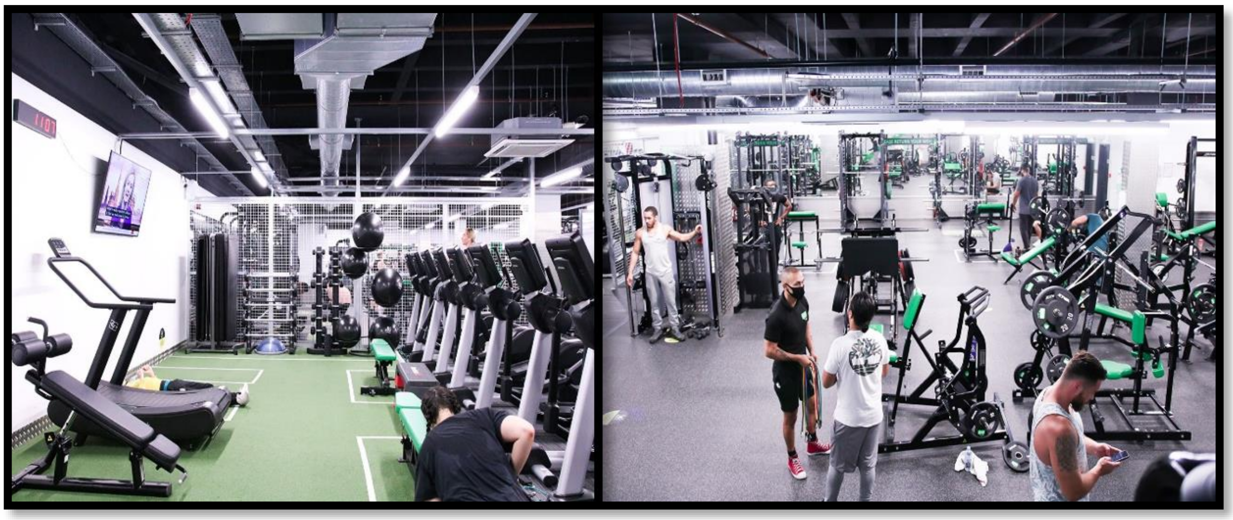


Figure 5 - Buzz Gym interior.

3.3.2 The Gym Group

The colour scheme in this gym gives off a cool atmosphere due to the vibrant colours. The main colour observed is blue which creates a relaxed feeling, a sense of calmness. The other colours observed were green, orange, white, grey, and black.

It looks like orange was used to inject energy. Even though the colours may not be suitable for the prototype, the colour could perhaps be used for certain buttons or features such as the back/close button.

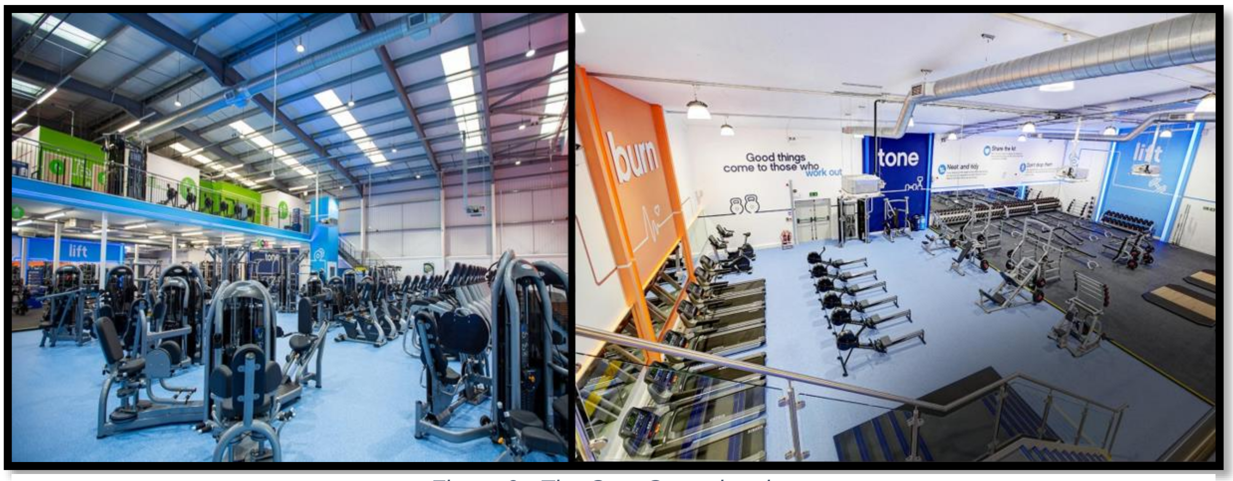


Figure 6 - The Gym Group interior.

3.3.4 Kings Gym

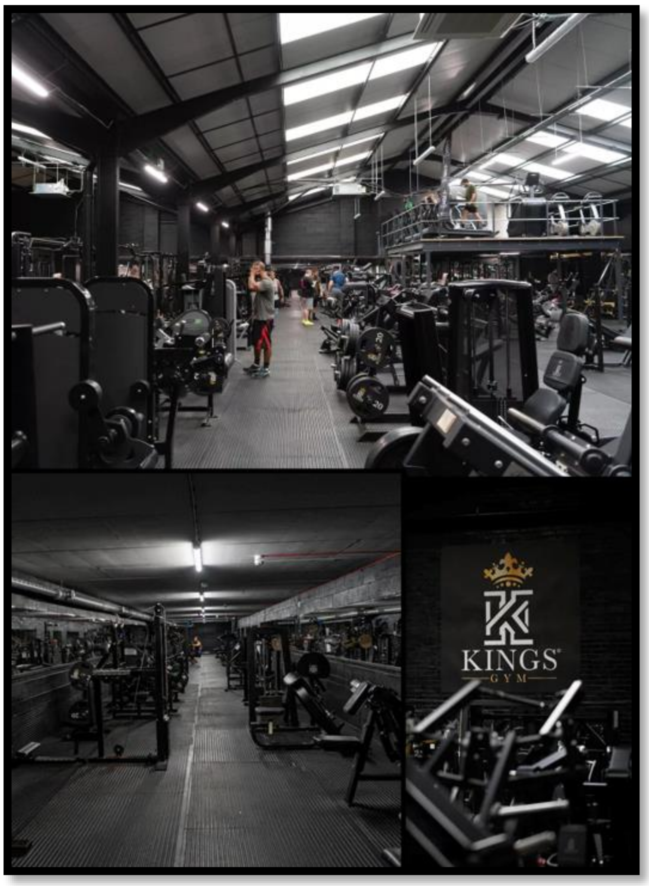


Figure 7 - Kings Gym interior.

Kings' Gym is the one that feels more of a proper gym compared to the others due to the choice of colours used. This gym embraces more of a monochromatic colours palette which gives off a sense of strength, power, seriousness, and professionalism. That serious feeling is what is needed for the prototype.

This is not the only reason for picking this colour scheme. It is more aesthetically pleasing, visually appealing and more of a gym vibe/sense which gives a deeper connection with the fitness journey.

The minimal use of vibrant colours allows attention to be draw to the industrial steel and black gym equipment.

3.4 Logo

The King’s Gym logo was the main inspiration behind the logo used for the gym application prototype. Both logos are modern is quite a minimalistic design, and they are only made are black and white colours to present aesthetic vibes. These logos were created on Canva.

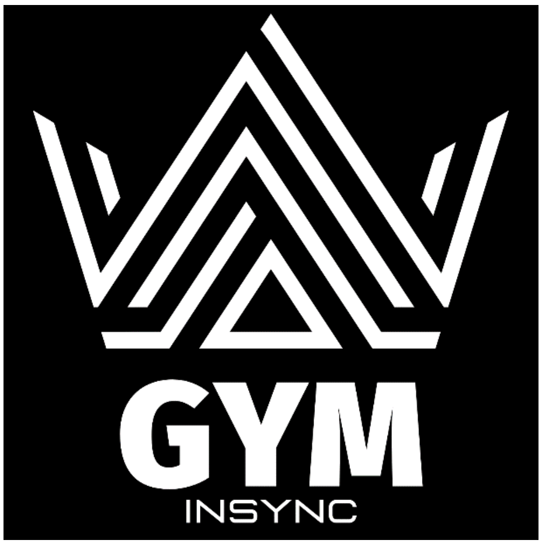


Figure 8 - Logo 1

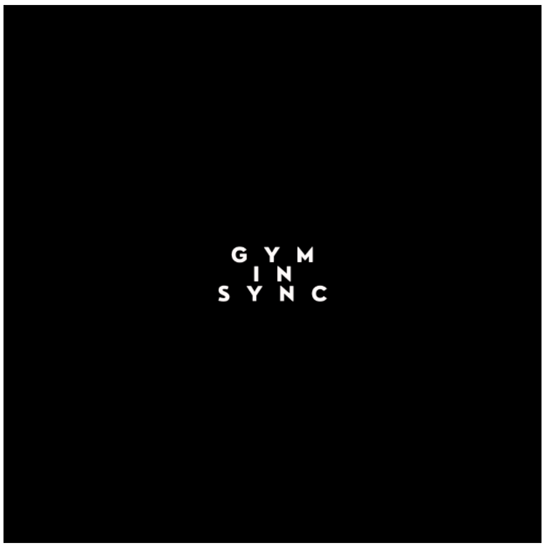


Figure 9 - Logo 2

Logo 1 has a crown-like symbol to convey a sense of excellence. And the dynamic design that reflexes the progressiveness and structure of fitness routines.

Logo 2 is much simpler to logo 1, with only white text on a black background. But regardless of it being very simple, it looks very clean and has a sophisticated look.

The first logo catches the eye immediately so it will be used in the log in screen, and the second logo can be used within the app.

3.5 Use Case Diagram

Designing a use case diagram for the gym app has helped to vision the application and its functionality better. It helped with understanding the services of the app that has to offer as one of the aims was to make it impactful for users to keep them on track of their progressive gym journey.

Figure 10 shows the use case and the availability of the users when they enter the app. Every function of other screens depends on the main menu screen. From the main menu screen, the user can freely navigate to login or sign up, use the round timer, view the exercise etc.

The difference between a logged in user and a user that is not logged in is the fact that the logged in user can track their progress and save their data to a secure database, whereas a user that is not logged in cannot track their progress because when the app is closed, the user will not view their previous statistics.

The reason why users without an account are allowed access to the app was because the app is for every user regardless of if an account is made not, but only certain features was available for everyone, such as the exercises with all the information and the round timer can be used by users with and without an account.

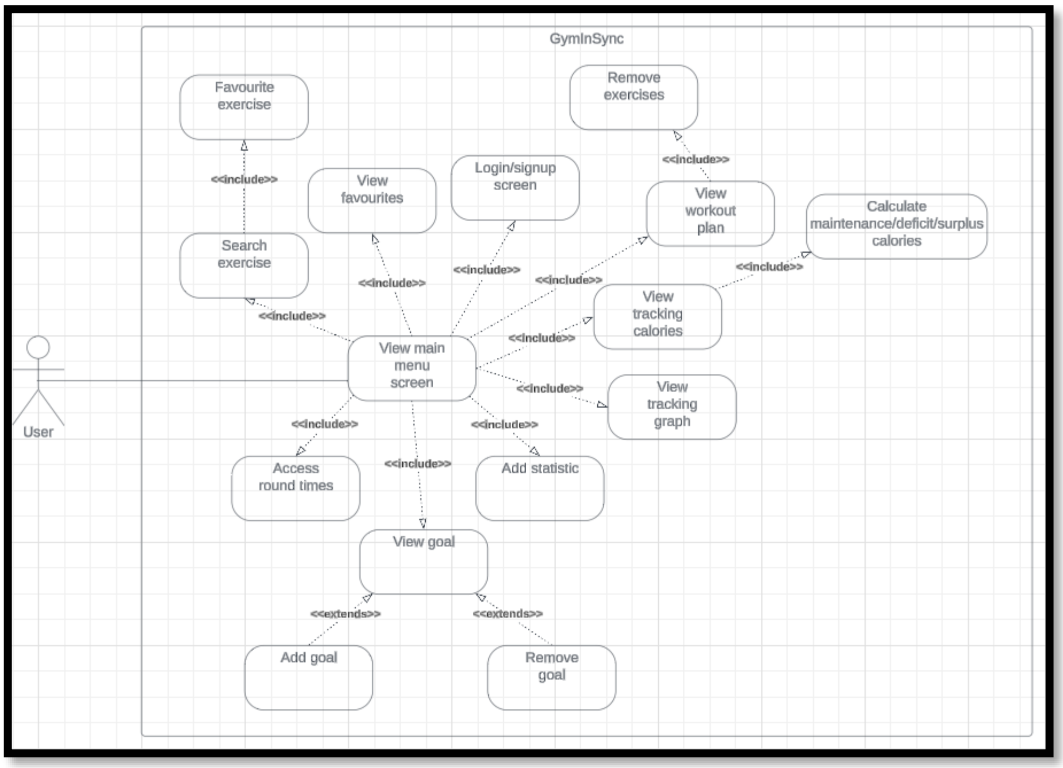


Figure 10 - Use Case Diagram.

3.6 Application Development Setup

The proper working environment needed to be setup first in order to start the development of the gym application prototype on visual studios. First of all, Visual Studios was installed.
After that, Node.js was required to be downloaded on the device. Once downloaded, the command prompt was opened.
It was necessary to ensure that the version of node.js is v16 or later, so to do this, the following line was entered:

“node -v”

This confirms that node.js version 21.5.0 is running on the device.

After this was confirmed, now the Expo CLI needed to be installed globally, and to do this, the following line was entered in the command prompt:

“npm I -g expo-cli”

Since the permissions did not have to be configured, the command line “sudo” did not have to be prefixed. And now with Expo CLI, projects can be easily created and run by React Native.

The next stage was to downloaded Expo Go on the personal phone. With this app, it can easily run the project on the personal phone device to view live updates. Expo Go required a log in, therefore one was created.

Now that the node.js, Expo CLI, and Expo Go is ready, it was time to set up Visual Studios.
After opening Visual Studios, a few extensions were installed. These extensions on Visual Studios will make coding a bit easier. Each extension is explained:

- **Material Icon Theme:** This allows icons to be used in the project.
- **Prettier - Code Formatter:** This allows the code to be formatted so it is more pleasing to the eye.
- **React Native Tools:** This allows react native applications to debugged inside Visual Studio code.
- **React-Native/React/Redux snippets:** This provides code snippets, so shortcuts can be typed up to generate code very quick.

Also, the settings were also tweaked, so whenever the code is saved, it will be reformatted using the Prettier - Code Formatter extension.

Now that it is all ready, the Expo project is created.

Using the Command Prompt window, the following line was entered:

“expo init GymInSync”

This initialises the development of the application.

Then the workflow is picked for building the prototype.

There were two choices:

- **The managed workflow** – This will take care of all the complexities behind the scenes such as the IOS and Android projects, therefore focus can be drawn purely on just the JavaScript project.
- **The bare workflow** – This provides bare bone React Native project with the iOS and Android projects seperately

For the sake of convenience and time, Managed Workflow with the blank template was used.

Now the folder called "GymInSync" needs to be accessed, and to do this, the following command lien was used to enter the folder:

"cd GymInSync"

Then the following command lien was used to open the folder in Visual Studios:

"."

This created the following three folders:

- **vscode**
- **assets** – this is where all the images, videos files and other kind of assets that need to be bundled with the app.
- **node_modules**

Now finally, to start an Expo server to serve application, the Visual Studios terminal window needs to be open. This terminal will be open at all times.
The following command line was entered:

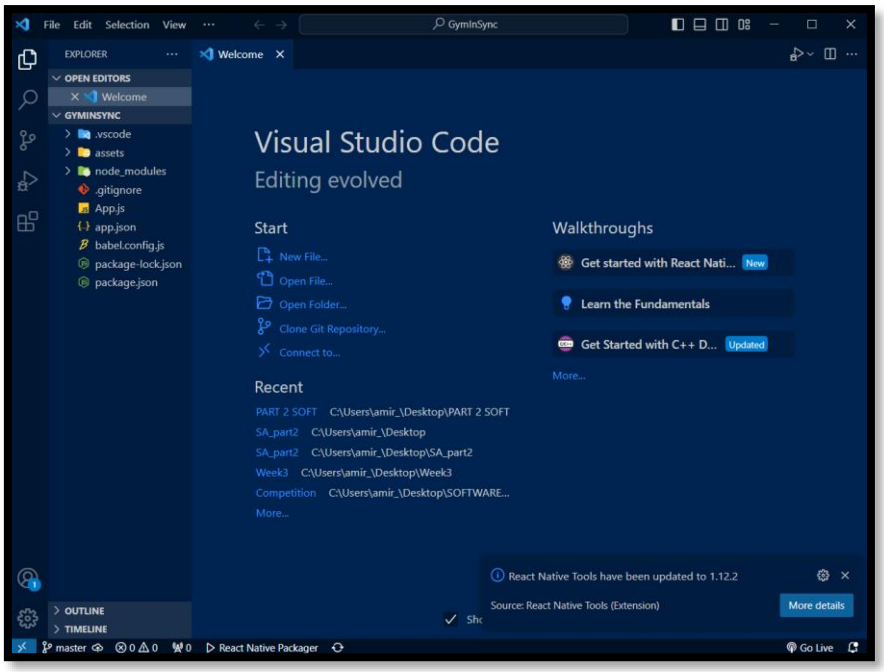
"npm start"

After entering "npm start", Visual Studios replied with a warning and advising to use the following command prompt because the installation of the webpack configuration was forgotten:

"npx expo install react-native-web@~0.19.6 react-dom@18.2.0 @expo/webpack-config@^19.0.0"

This now allows the web browser to be opened whenever the key W is pressed to open web support and view it on Metro Bundler.
And in order to debug the code, a launch.json file was created and ReactNative Environment was used, and Packager was Attached.

After all that is completed, it should look a bit like figure 11 with the folders on the left side along with App.js which is a basic React Native component.



React Native Tools have been updated to 1.12.2

Source: React Native Tools (Extension)

Figure 11 - Visual Studios

Now the pre-development stage is completed, and the development stage can start.

4. Application Development

To start the development of the gym application, different screens that was planned to be implemented needed to be created.
A new screens folder was created within the GymInSync folder.
Also, a new components folder was created for any future components that was planned to be created to make the functionality and display of the code a lot cleaner.
All these folders as well as the assets folder were placed in a new folder called App. This is now neatly sorted in order to navigate between the codes more efficiently.

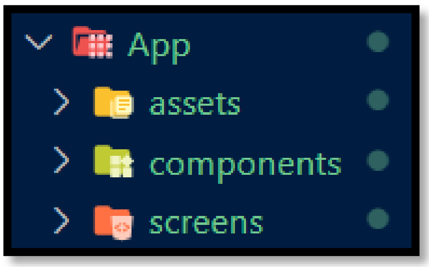


Figure 12 - App folder

4.1 Components

A component is a building block that can be used to manage the mobile application depending on the sought functionality. Each component is different and can be used and combined to achieve a certain function. These are all provided by React Native and can be simply imported into the code so no extra instalments need to be made.

The following are the components used in this mobile gym application.

1) **View** is similar to 'div' in web developments. It is the main fundamental container component that supports accessibility controls, style, and touch handling.

2) **SafeAreaView** is similar to the View component, but when the code is wrapped in the safeareaview container, it displays and adjusts the screen to ensure that the content does not obscure feature from the device such as the navigation bar of Android devices and the top notch on the latest iPhone models.

3) **StyleSheet** is similar to the stylesheets found in CSS. It allows containers to be styled and can be used multiple times in the application.

4) **Text** is a basic component that allows text to be displayed.

5) **Image** is another basic component that allows images to be displayed.

6) **ScrollView** is a container that allows the user to scroll horizontally or vertically.

7) **TouchableOpacity** is a container that can be used to wrap views to make the respond to touches.

Other components need to be imported and installed from external libraries known as dependencies.

8) **Alert** is used to display an alert to the user during confirmation or acknowledging a message.

9) **Modal** is another screen that pops up or slides up and overlays the current screen, this helps with continuing the functionality of the current screen and temporarily allowing the user to commit another action.

10) **StatusBar** is used for design purposes. It controls the appearance of the status bar on top of the screen.

11) **TextInput** allows users to enter text into text input fields.

12) **Keyboard** allows the keyboard to appear to grant the user to interact with it and type text into the text input fields.

13) **Dimensions** retrieves information about the dimensions of the device. It is used to create a responsive design and layout so it adapts to the screen size.

14) **FlatList** renders a big list of data.

4.1.1 Created Components

Components can also be created to allow function to be reused throughout the code. Otherwise, the code needs to be written every time. In the case of this project, nine functional components were created.
The created components serve and build gym application in such a manner to help in achieving the main accent on elements of User Interface (UI), navigation, and state management of exercises.
Each component and its function have been broken down as the following:

1) **Buttons.js**: This component file is used to define a customizable button for gym application. This button can be custom configured according to the needs with customizations like title, background colour, text colour, and image with further customization for styling. It also has the implementation of TouchableOpacity which wraps views to make them respond correctly to touches. When the button gets pressed, the function that is passed through the 'onPress' prop will be run. The button can display text or an image.
2) **PRexercises.js**: This exports an array of objects for Personal Record(s) (PR) of the user. Each has two properties: id and title. The size of the array is 12 in which the user can pick from to add a PR.
3) **TopLogo.js**: This component supports title, onPress action, backgroundColour, colour, image, and extra styling customization. The primary use of this is to display the logo for branding.
4) **TextInputBox.js**: This is a text input box with optional icon support under the React Native apps environment, inheriting from 'MaterialCommunityIcons' for icons. The text input box can be styled and includes features of a background colour, text colour, and icon display. This component is helpful in forms or search bars for any input field requirement.
5) **BottomTab.js**: This defines the bottom tab navigator and allows the tab title and icon to be modified. Each button can navigate depending on the onPress action. It's useful for navigation in mobile applications.
6) **Cards.js**: This component displays the cards with information. It is designed in such a way that it contains the code for the image, title, and subtitle for each card. The cards can be pressed and execute a function upon press, which in this case is navigation. This component is useful for displaying lists of items where each item can be interacted with.
7) **Details.js**: This implements a modal dialog to show item details on press. It can be visible and can be closed because it has a close function. This component is useful in showing more details about the items from within a popup modal without any navigation.
8) **ExerciseContext.js**: This defines a React context for the exercises of the app, where one can add and toggle the favourite status of exercises retrieve favourited exercises. The context provider makes it easier to share state management related to exercises across different screens of the gym app.
9) **Exercises.js**: This has an array of exercise objects and can be exported. Each exercise contains an 'id', 'title', 'subtitle', and 'image'. This data structure is part of the values provided by ExerciseContext in order that the list of exercises can be managed and displayed on this app.

4.2 Dependencies

Dependencies are external libraries, and these libraries are sometimes required since not every library is provided by the React Native framework. These dependencies offer new functionalities that can allows the mobile app to handle specific tasks. The dependencies were found after a quick Google search for the functions that were required for GymInSync.

The libraries can be installed using the terminal in Visual Studio and can then be implemented within the code by importing the component from the external dependency.

The following command line is required to install an external dependency:

```
npm install <enter package name here>
```

The following dependencies (in no specific order) along with the components were required during the development of the mobile gym application:

1) **'react-native-calendars'** – 'Calender' allows a live calendar to be integrated into the app.
2) **'@react-native-picker/picker'** – 'Picker' creates menus that have the ability to drop down and allows the user to select an item from a list of options.
3) **'react-native-vector-icons/FontAwesome'** – 'Icon' imports icons created by 'FontAwesome' and can be used for design purposes.
4) **'react-native-chart-kit'** – 'LineChart' allows line charts to be create from data.
5) '**@react-native-async-storage/async-storage'** – 'AsyncStorage' stores small amounts of data.
6) **'expo-constants'** – 'Constants' provides constants and data about the app and the device it is running on.

4.3 Screens

The screens are the UIs, it is what the user will see and interact with. A total of nine screens are connecting to each other through App.js to make the gym application run.
All the screens have a few things in common such as the style and a back button. Every screen have a back button to allow the user to navigate back to the previous screens, and every screen is inspired by the colour scheme inspired by Kings Gym. They all shows the monochromatic colours palette as well a minimum colours were used, the same as Kings gym after analysing their interiors. This gives a professional and serious energy to the user and the colour scheme is suitable for a gym app.
It was ensured that the styles in the screens were consistent.

4.3.1 Main Screen

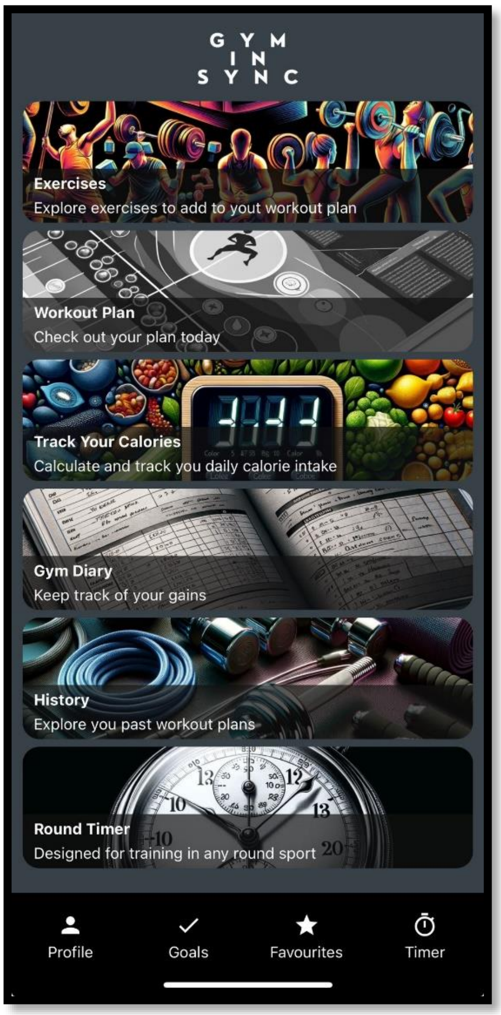

GYM IN SYNC
Exercises
Explore exercises to add to yout workout plan
Workout Plan
Check out your plan today
Track Your Calories
Calculate and track you daily calorie intake
Gym Diary
Keep track of your gains
History
Explore you past workout plans
Round Timer
Designed for training in any round sport
Profile
Goals
Favourites
Timer

Figure 13 - Main Screen

The first screen created was the main screen. This was chosen to be done first because, just as the use case diagram shows, users will be able to navigate from the main screen to other screens.
This screen required different components which are reusable pieced of User Interface (UI):

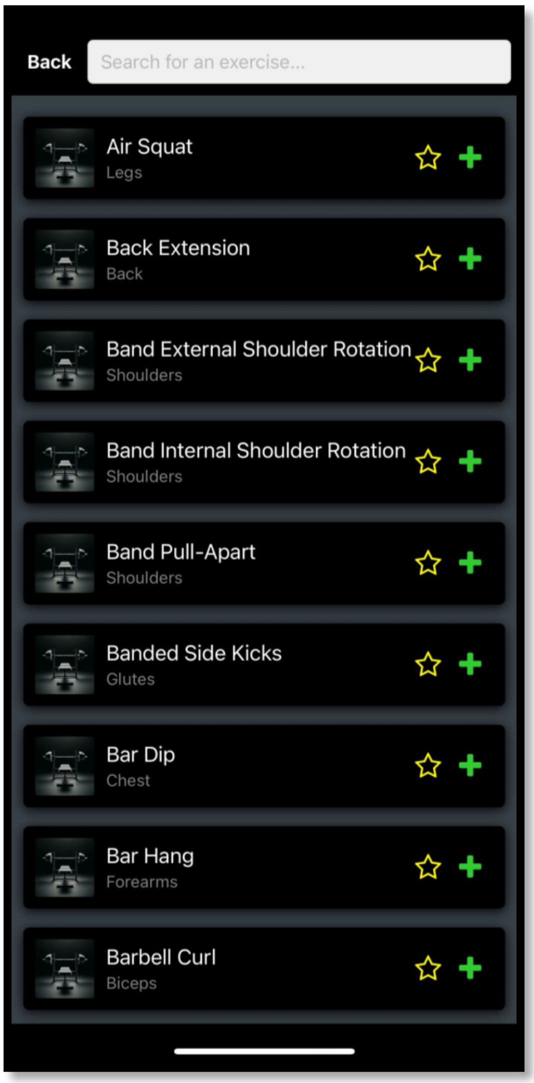
- Cards.js – Displays the cards with an image, title and subtitle.
- BottomTab.js – Displays the tab at the bottom with icons and tables.
- TopLogo.js – Displays the logo on the top of the screen.

These components are all put together by MainScreen.js (figure 13).

Cards.js and **TopLogo.js** have the props View, Stylesheet, Image, Text, Touchability.
BottomTap.js has the props View, StyleSheet, Text, Touchability.
MainScreen.js has the props View, StyleSheet, ScrollView as well as the other components (Cards.js, TopLogo.js, BottomTap.js).

The pictures were created by using a photopea which is an online photoshop tool as well as canva which is an online graphic design tool.
Initially the title and description did not have a see-through background, it was just black. By setting the opacity to 0.7, it makes the cards more visually pleasing.

4.3.2 Exercises Screen & Favourites Screen



Air Squat	Legs
Back Extension	Back
Band External Shoulder Rotation	Shoulders
Band Internal Shoulder Rotation	Shoulders
Band Pull-Apart	Shoulders
Banded Side Kicks	Glutes
Bar Dip	Chest
Bar Hang	Forearms
Barbell Curl	Biceps

Figure 14 - Exercise Screen

When the user presses on the “Exercises” card, the app will navigate to the “ExerciseScreen”.
Initially, the flatlist of exercises were all in the same code as ExerciseScreen, but it was seperated to make it more organised and easily accessible when editing the code, otherwise there would have been more than 500 lines of code within ExerciseScreen. Therefore the exercises are imported as an array and rendered from another component called Exercises.js.
The cards displayed on this screen all have an image, title, subtitle, a star icon and a plus icon.
The title is the name of the exercise, and the subtitle is the muscle which the exercise hits.
This was done in such a manner so if the user searches for an exercise in the search bar on top of the screen, the screen will live update to display the exercises related to the title. But if the user wants to find exercises based on the muscle to train, it would also work since the subtitle is the muscle.
So essentially, in order to search for an exercise, the user will have to type in the name of the exercise or the muscle to train (as shown in figure 15 and 16).

The ExerciseScreen is the interface for users to interact with, and it uses seval components to provide the functionality.
It uses the Details.js component when the user presses on the exerise, a modal with the exercise details will appear on the screen. But unfortunately, no exercise details were added due to focusing more on the higher priority functions and design.

The user can press the plus icon. When pressed, a modal will appear with weekdays. The user can add an exercise to a day to create a workout plan for that specific day (figure 18).

This screen is connected to another screen which is called the FavouritesScreen through a component called ExerciseContext.js. When the user favourites on the exercises screen by pressing the star, the star change from outline to full star. This indicates tha the exercise has been favourited (Figure 17). When this is done, it will toggle the function toggleFavorite in ExerciseContext.js and the favourited exercise will appear in the favourited screen. An example is shown in figure 19 to figure 20.

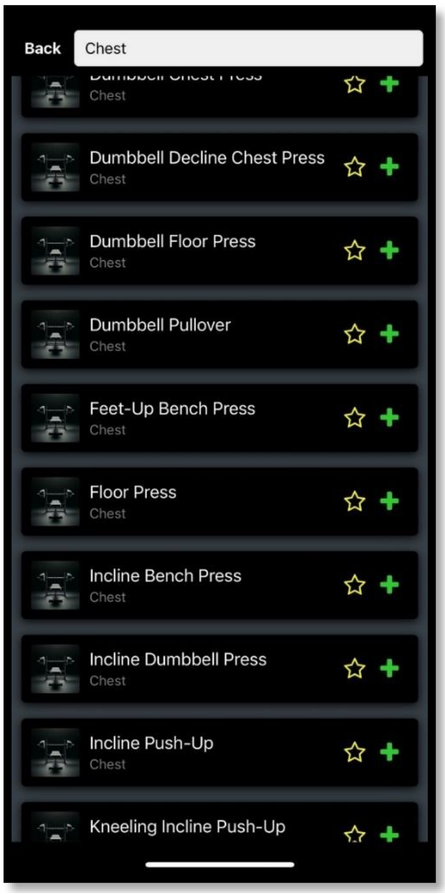


Figure 15 - searching exercises by muscle...

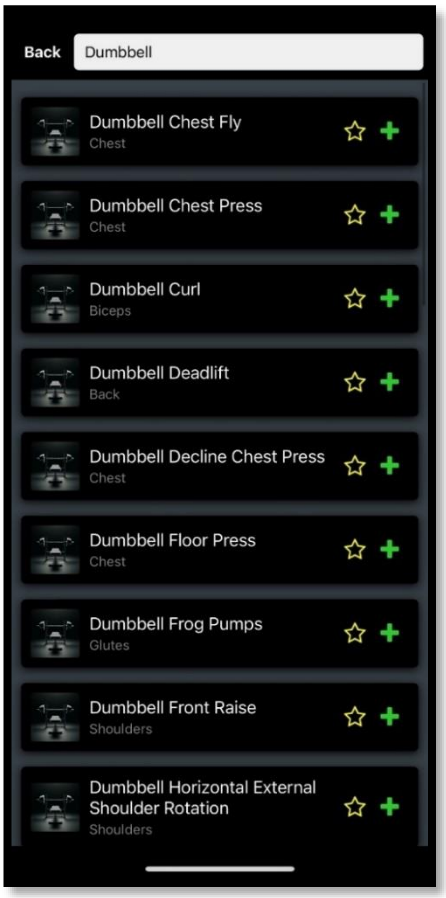


Figure 16 - searching exercises by exercise.

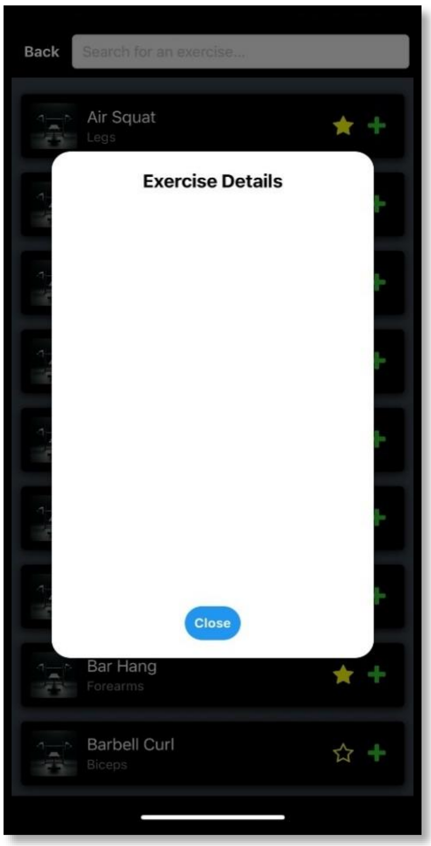


Figure 17- modal when an exercise is pressed.

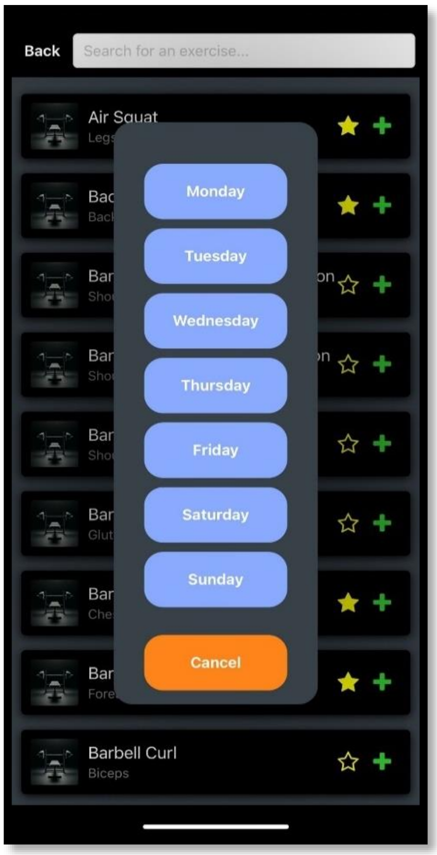


Figure 111 - modal to add exercise to the desired day.

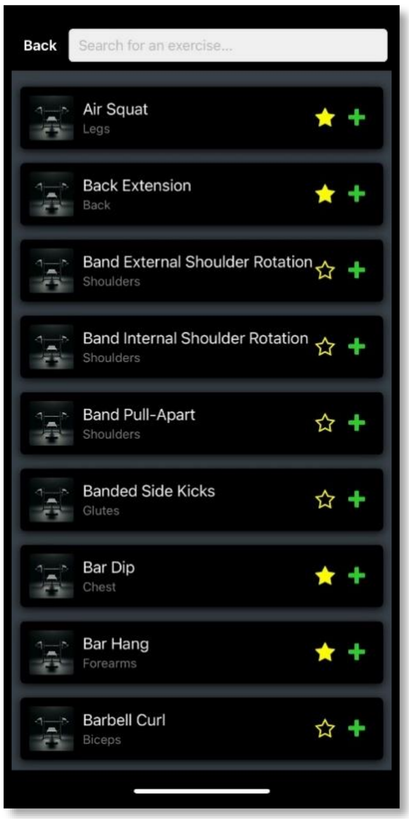


Figure 19 - favourited exercises in ExerciseScreen.js.

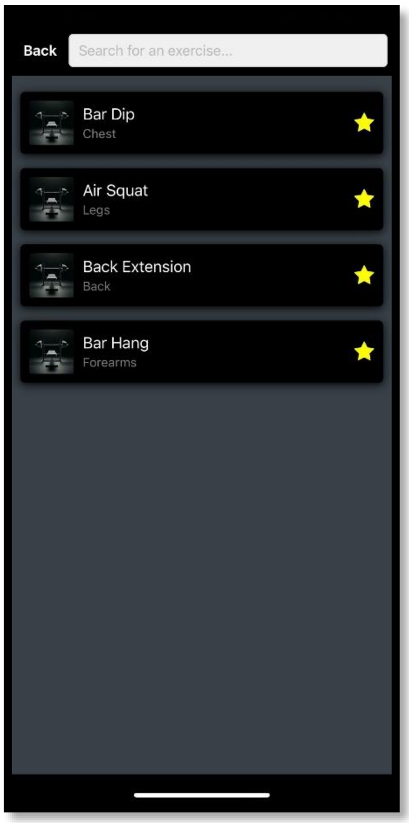


Figure 20 - favourited exercises in FavouritesScreen.js.

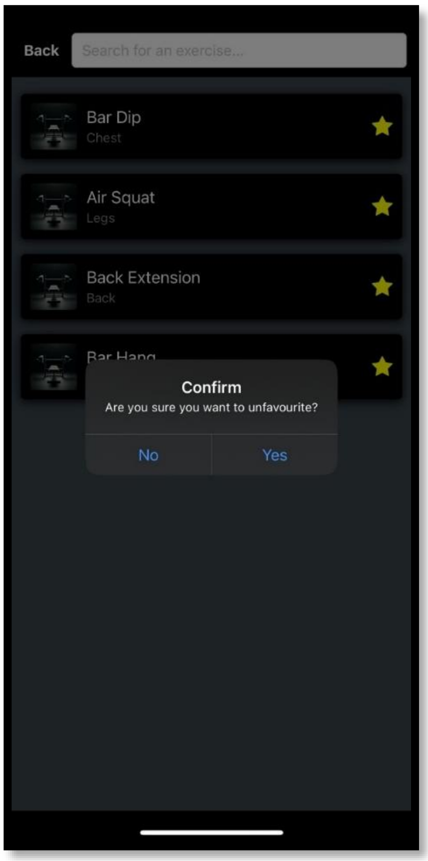


Figure 21 - confirmation before unfavouriting.

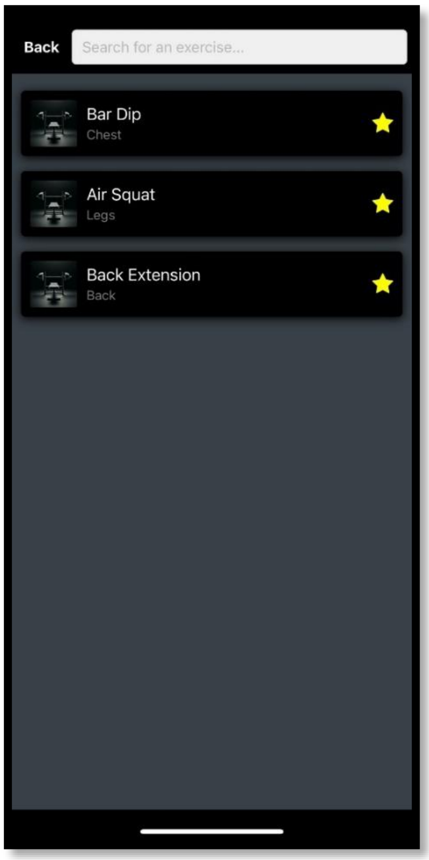
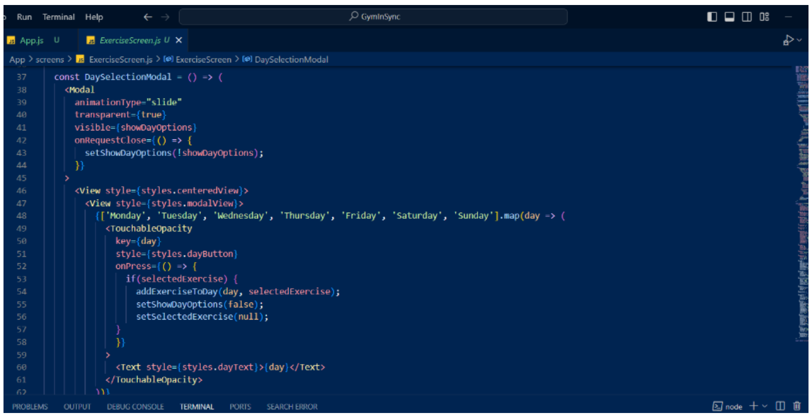


Figure 21 - result after unfavouriting exercise.

The user can also unfavourite the exercise. When the user presses the full star in the exercises screen, it will just unfavourite the exercise.

But when the user presses the full star in the favourites screen, an alert will pop up to confirm the choice of the user and there is a choice to confirm or cancel the choice. This is because the user may accidentally unfavourite an exercise, and the alert will prevent this.

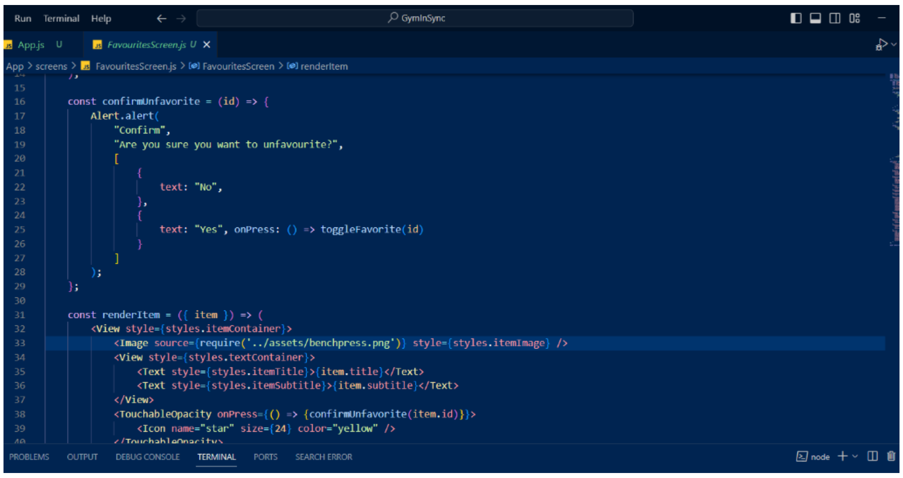
The combination of the components creates an interactive experience which allows users to favourite exercises and personalise work out plans. And the overall design prioritises convenience and accessibility which is crucial in order to make the UI engaging.



Run Terminal Help
GymInSync
App.js U
ExerciseScreen.js U
App > screens > ExerciseScreen.js > ExerciseScreen > DaySelectionModal
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
node

```
const DaySelectionModal = () => (
  <Modal
    animationType="slide"
    transparent={true}
    visible={showDayOptions}
    onRequestClose={() => {
      setShowDayOptions(!showDayOptions);
    }}
  >
    <View style={styles.centeredView}>
      <View style={styles.modalView}>
        {['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'].map(day => (
          <TouchableOpacity
            key={day}
            style={styles.dayButton}
            onPress={() => {
              if(selectedExercise) {
                addExerciseToDay(day, selectedExercise);
                setShowDayOptions(false);
                setSelectedExercise(null);
              }
            }}
          >
            <Text style={styles.dayText}>{day}</Text>
          </TouchableOpacity>
        ))}
```

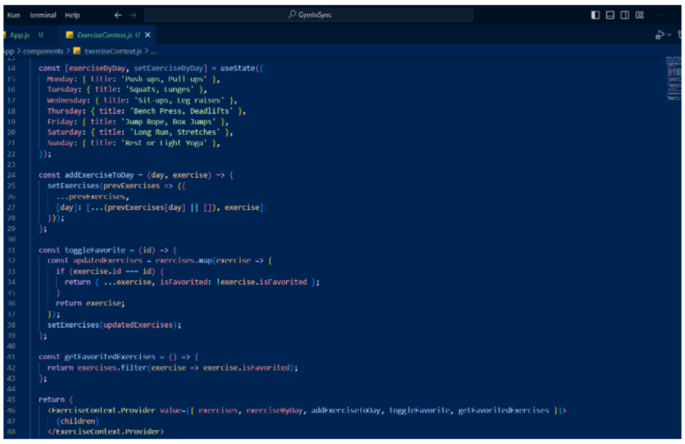
Figure 14a - ExerciseScreen.js code snippet.



Run Terminal Help
GymInSync
App.js U
FavouritesScreen.js U
App > screens > FavouritesScreen.js > FavouritesScreen > renderItem
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
node

```
  const confirmUnfavorite = (id) => {
    Alert.alert(
      "Confirm",
      "Are you sure you want to unfavourite?",
      [
        {
          text: "No",
        },
        {
          text: "Yes", onPress: () => toggleFavorite(id)
        }
      ]
    );
  };

  const renderItem = ({ item }) => (
    <View style={styles.itemContainer}>
      <Image source={require('../assets/benchpress.png')} style={styles.itemImage} />
      <View style={styles.textContainer}>
        <Text style={styles.itemTitle}>{item.title}</Text>
        <Text style={styles.itemSubtitle}>{item.subtitle}</Text>
      </View>
      <TouchableOpacity onPress={() => {confirmUnfavorite(item.id)}}>
        <Icon name="star" size={24} color="yellow" />
      </TouchableOpacity>
```

Figure 14b - FavouriteScreen.js code snippet.



```
  const [exerciseByDay, setExerciseByDay] = useState({
    Monday: { title: 'Push ups, Pull ups' },
    Tuesday: { title: 'Squats, Lunges' },
    Wednesday: { title: 'Sit-ups, Leg raises' },
    Thursday: { title: 'Bench Press, Deadlifts' },
    Friday: { title: 'Jump Rope, Box Jumps' },
    Saturday: { title: 'Long Run, Stretches' },
    Sunday: { title: 'Rest or Light Yoga' },
  });

  const addExerciseToDay = (day, exercise) => {
    setExercises(prevExercises => ({
      ...prevExercises,
      [day]: [...(prevExercises[day] || []), exercise]
    }));
  };

  const toggleFavorite = (id) => {
    const updatedExercises = exercises.map(exercise => {
      if (exercise.id === id) {
        return { ...exercise, isFavorited: !exercise.isFavorited };
      }
      return exercise;
    });
    setExercises(updatedExercises);
  };

  const getFavoritedExercises = () => {
    return exercises.filter(exercise => exercise.isFavorited);
  };

  return (
    <ExerciseContext.Provider value={{ exercises, exerciseByDay, addExerciseToDay, toggleFavorite, getFavoritedExercises }}>
      {children}
    </ExerciseContext.Provider>
```

Figure 14c - ExerciseContext.js code snippet.

4.3.3 Workout Screen

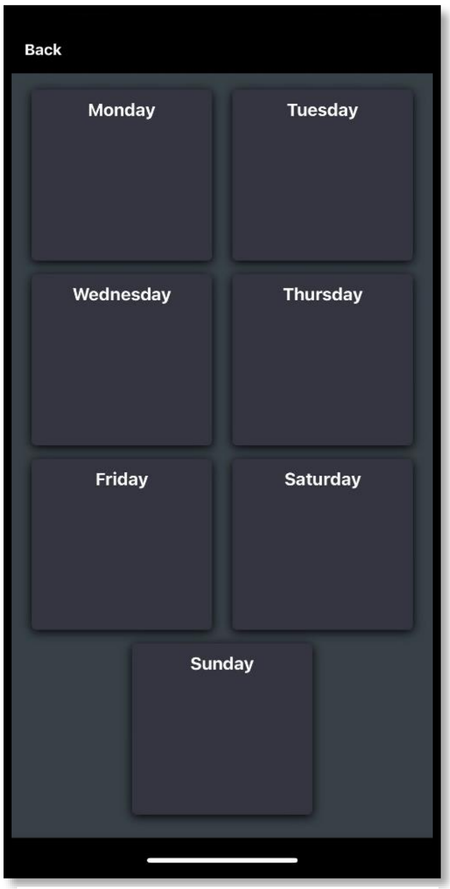


Figure 22 - workout screen.

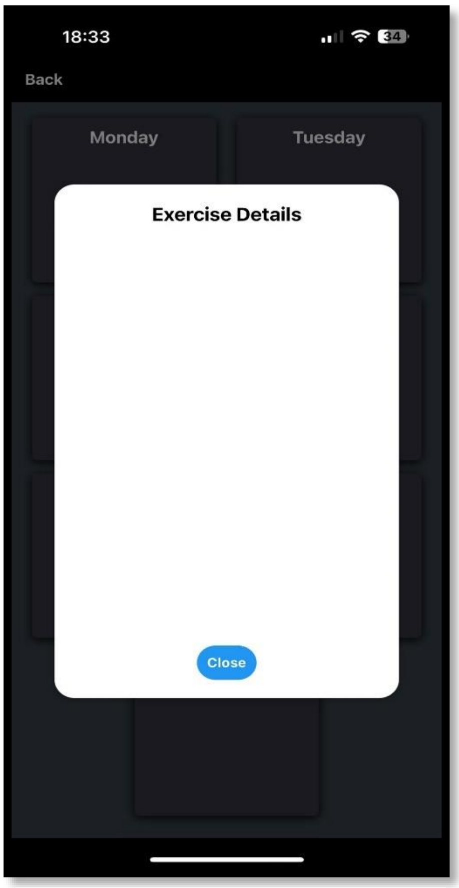
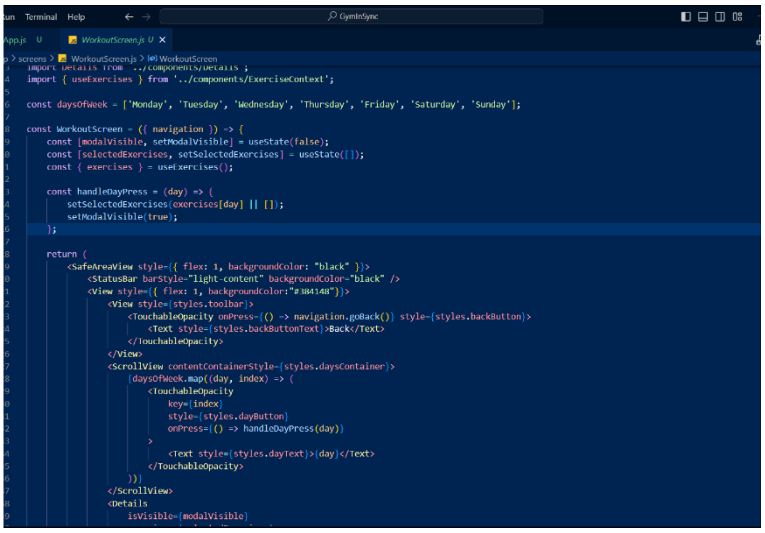


Figure 23 - modal when card is pressed.

The workout screen consists of seven cards, each card presents a day of the week. The user can pick days for rest days, and days for training certain muscle groups. The exercises are added from the exercises screen when the plus icon is pressed, and the user has a choice to add an exercise to a day (figure 18).

When pressing on the card, the Details.js component will be displayed as a modal with the details of the exercise as shown in figure 18 but no details have been added yet.

It was planned to display the exercises added on the cards, but this was not achieved successfully.



```
import Details from '../components/Details';
import { useExercises } from '../components/ExerciseContext';

const daysOfWeek = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];

const WorkoutScreen = ({ navigation }) => {
    const [modalVisible, setModalVisible] = useState(false);
    const [selectedExercises, setSelectedExercises] = useState([]);
    const { exercises } = useExercises();

    const handleDayPress = (day) => {
        setSelectedExercises(exercises[day] || []);
        setModalVisible(true);
    };

    return (
        <SafeAreaView style={{ flex: 1, backgroundColor: "black" }}>
            <StatusBar barStyle="light-content" backgroundColor="black" />
            <View style={{ flex: 1, backgroundColor:"#384148"}}>
                <View style={styles.toolbar}>
                    <TouchableOpacity onPress={() => navigation.goBack()} style={styles.backButton}>
                        <Text style={styles.backButtonText}>Back</Text>
                    </TouchableOpacity>
                </View>
                <ScrollView contentContainerStyle={styles.daysContainer}>
                    {daysOfWeek.map((day, index) => (
                        <TouchableOpacity
                            key={index}
                            style={styles.dayButton}
                            onPress={() => handleDayPress(day)}
                        >
                            <Text style={styles.dayText}>{day}</Text>
                        </TouchableOpacity>
                    ))}
                </ScrollView>
                <Details
                    isVisible={modalVisible}
```

Figure 12a - WorkoutScreen.js code snippet.

4.3.4 Diary Screen and History Screen

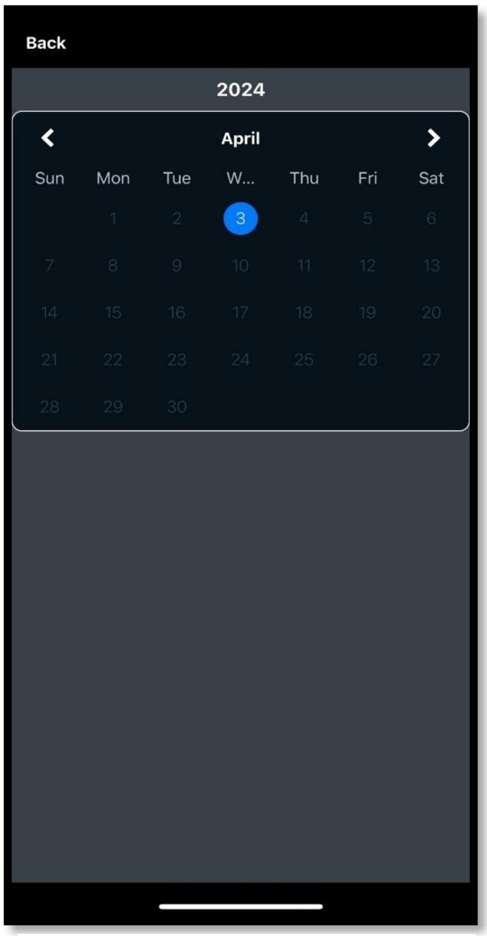


Figure 24 - Diary Screen

The DiaryScreen.js and HistoryScreen.js was one of the hardest challenges faced during the whole project. The React Native understanding needed to be developed a bit more before starting these screens which is why the development of the screens was later compared to the other screens.

The diary screen displays a calendar which allows users to interact with by going back a month in the past or into the future, and the calendar shows the days of the week. This calendar was imported from an external dependency as it is not supplied by React Native directly.

The user can press the date, and a modal will slide up giving the user a choice to add a statistic or to cancel.

When the user chooses to add a statistic, the modal will close, and another modal will slide up and fill up the screen. In this modal, the user has a choice to pick a type of statistic and enter it.

In this modal, two ways have been presented on how to take in statistics.

One way is displayed in ‘weight’. When the user chooses weight, the user can either enter the weight in kilograms or pounds. Whichever is chosen, it will be automatically converted into the other.

The other way is displayed in ‘height’. When the user chooses to enter a statistic for height, the user can choose which measurement is desired, the metric or imperial. And whichever is chosen is what the user will input.

This is just a thoughtful touch which was observed by the pump gym application as this allows the app to be accessible and used by users who use kilograms or pounds therefore being more convenient.

When data is entered into the statistic, an alert will pop up and let the user know that the statistic has been taken into account using AsyncStorage in order to display it on the history screen.

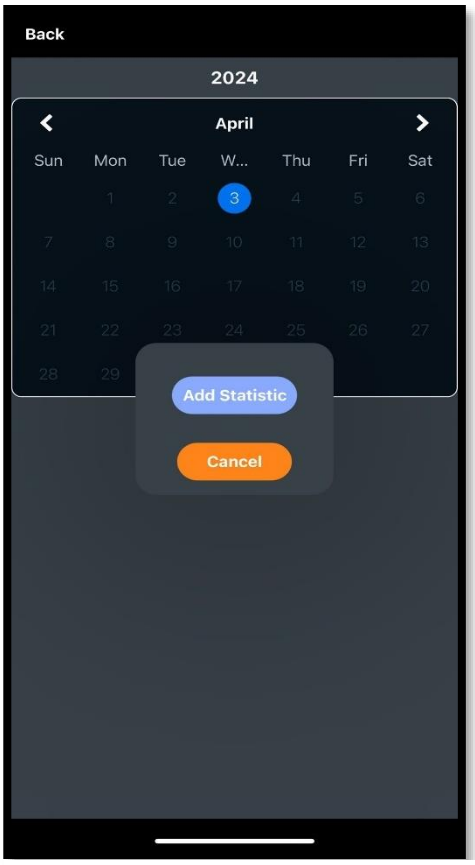


Figure 25 - modal to enter or cancel statistic.

The user can exit the diary screen by pressing the back screen to go back to the main menu, and then enter the history screen by pressing the history card.

The history screen displays graphs. It uses the data to stored from the diary screen to generate a line graph.
The graph for weight shows two lines, one for pounds (blue) and one for kilograms (red). The user has the ability to swipe right to view another graph for another statistic. As displayed in figure 28, the weight and body fat percentage is shown.
The user can also reset the data at the bottom of the screen to restart the data.
The graph has the date in the x-axis and measurement in the y-axis which helps the graph to represent the history of statistics and is easily understood by users.

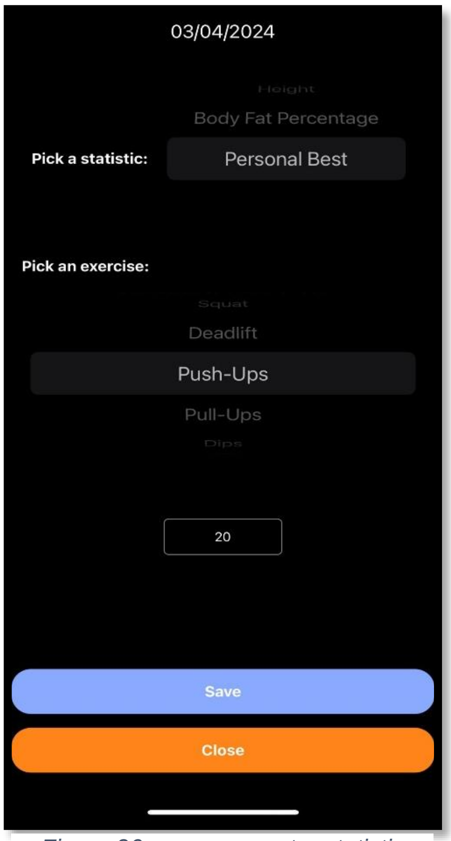


Figure 26 - user can enter statistic (push up reps in example)

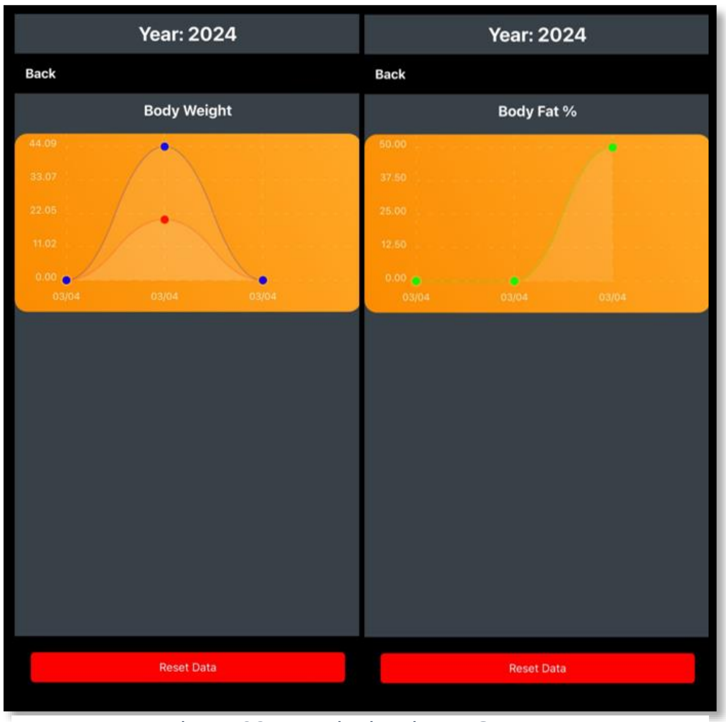


Figure 28 - graphs in History Screen.

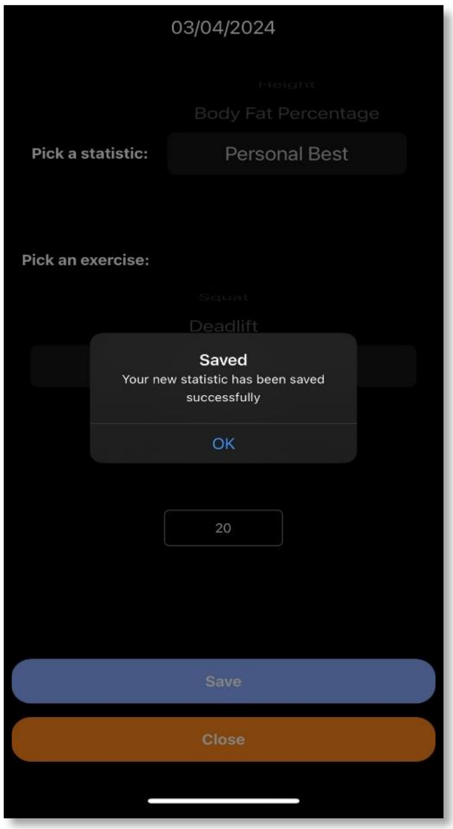


Figure 27 - confirmation of statistic saved.

4.3.5 Timer Screen

The function of the timer screen is designed in such a way to serve as a suitable tool for users to do exercises that are made up of rounds such as sparring or circuits.
The pickers in the timer screen are three: rest time, round time, rounds.
The rest time can be selected in intervals of ten seconds whereas the round time can be selected by intervals of five seconds. This is because rest time is usually by rounded to the nearest ten seconds whereas a round time can sometimes be 45 seconds which requires intervals of five.
When the timer is started, the timer will start counting down from the selected round time.
The timer can be paused or stopped. When stopped it will return to the initial screen of TimerScreen.js.
When paused, the timer will be paused, and the user can either resume or completely stop the timer.
After the round time is over, the timer will enter the rest time. This can be indicated as it will say "rest time" in red.
When the timer finishes, an alert will pop up to indicate that the timer has ended.

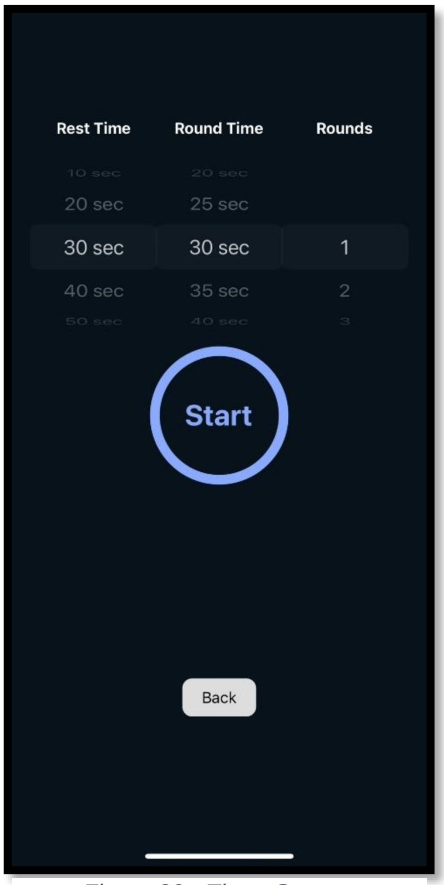


Figure 29 - Timer Screen.

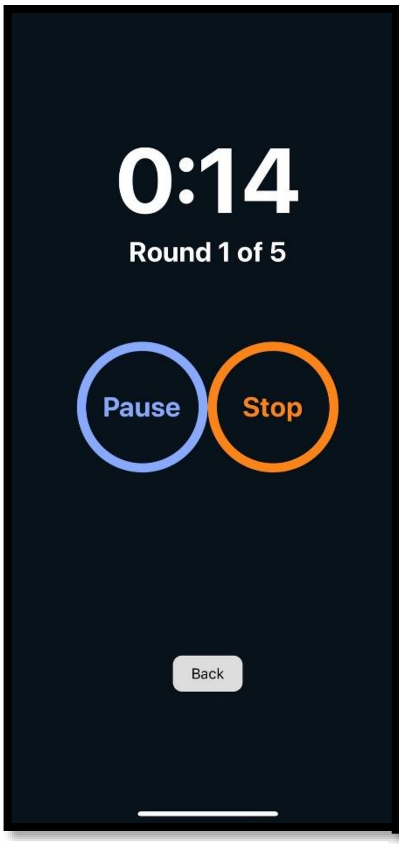


Figure 30 - Timer started.

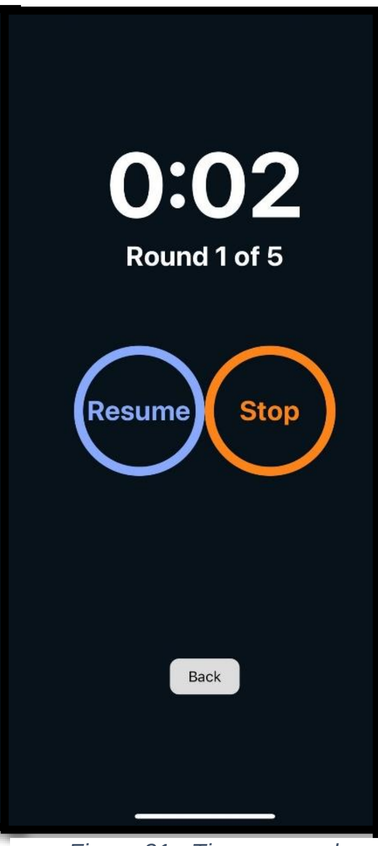


Figure 31 - Timer paused.

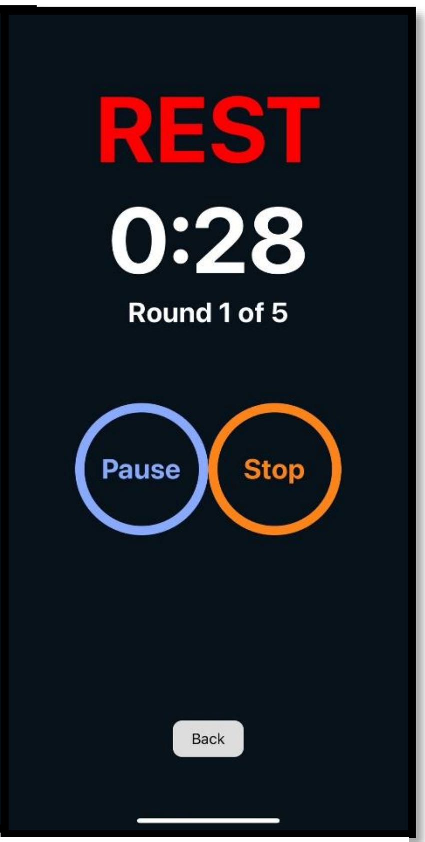
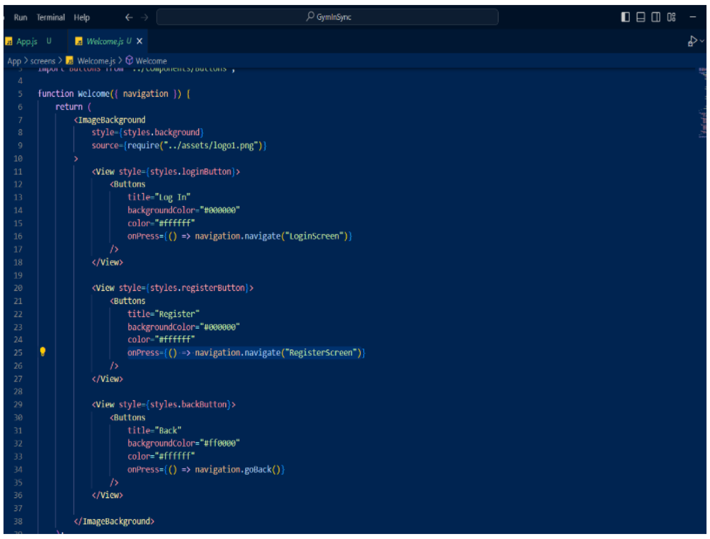


Figure 32 - Timer in rest state.

4.3.6 Profile Screen

When the profile icon is pressed on the tab in the main screen, it will navigate the user to a welcome screen. A big logo of the gym is displayed, and the user can either login, register or go back to the main screen.



```
import Buttons from "../components/Buttons";

function Welcome({ navigation }) {
    return (
        <ImageBackground
            style={styles.background}
            source={require("../assets/logo1.png")}
        >
            <View style={styles.loginButton}>
                <Buttons
                    title="Log In"
                    backgroundColor="#000000"
                    color="#ffffff"
                    onPress={() => navigation.navigate("LoginScreen")}
                />
            </View>

            <View style={styles.registerButton}>
                <Buttons
                    title="Register"
                    backgroundColor="#000000"
                    color="#ffffff"
                    onPress={() => navigation.navigate("RegisterScreen")}
                />
            </View>

            <View style={styles.backButton}>
                <Buttons
                    title="Back"
                    backgroundColor="#ff0000"
                    color="#ffffff"
                    onPress={() => navigation.goBack()}
                />
            </View>

        </ImageBackground>
```

Figure 33a - Welcome.js code snippet.

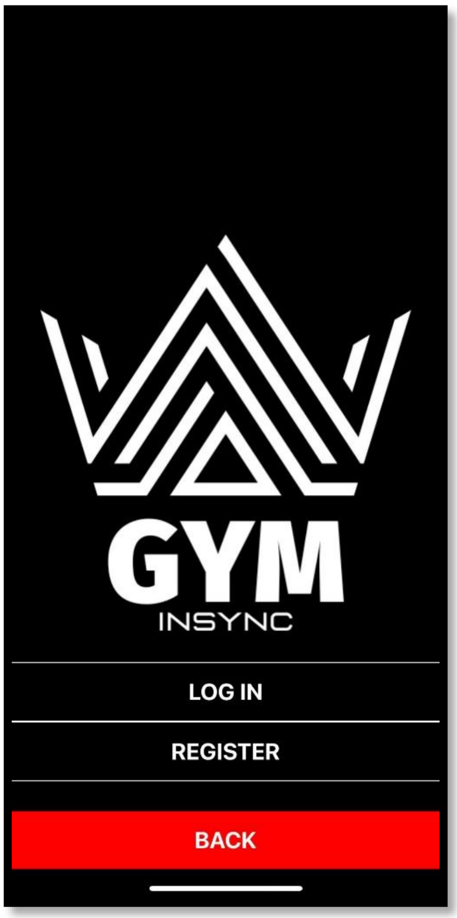


Figure 33 - Welcome Screen

4.3.6.1 Login Screen

The login screen requires the user to enter their email address and password to be able to login and store the data into their account. The password is not visible because in the code, secureTextEntry has been set to true which makes the password look like black dots.
The textboxes have placeholders to indicate where to type in the email and password, as well as icons for designing purposes.
When logged in, it will lead back to the main screen and allow users to store data.

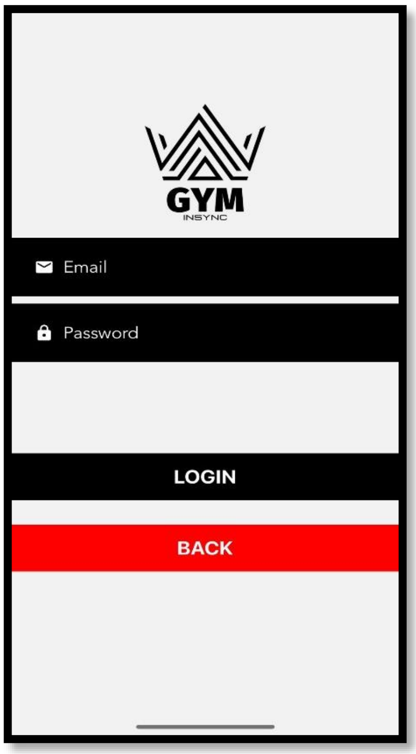
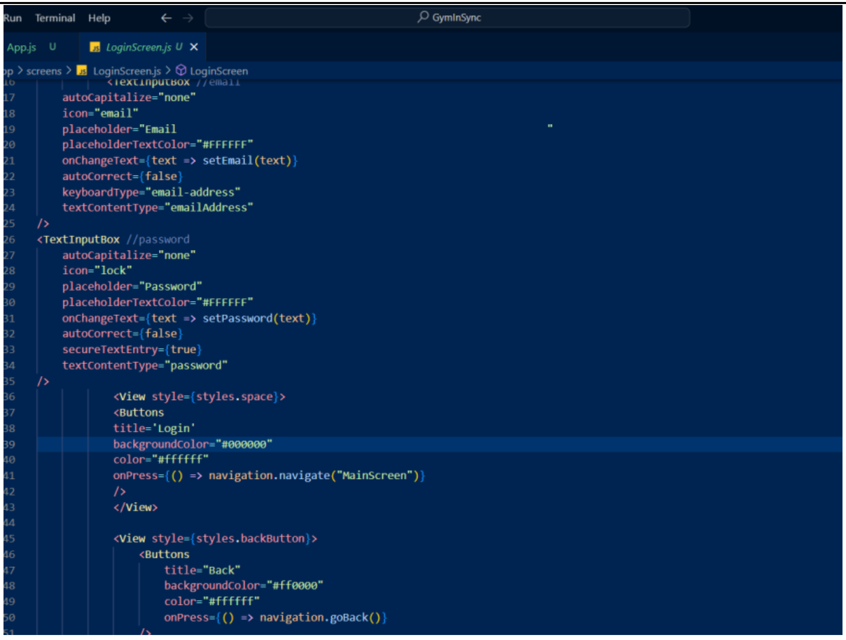


Figure 34 - Login Screen



```
Run  Terminal  Help      GymInSync
App.js  U     LoginScreen.js U ×
pp > screens > LoginScreen.js > LoginScreen
                <TextInputBox //email
            autoCapitalize="none"
            icon="email"
            placeholder="Email                                    "
            placeholderTextColor="#FFFFFF"
            onChangeText={text => setEmail(text)}
            autoCorrect={false}
            keyboardType="email-address"
            textContentType="emailAddress"
        />
        <TextInputBox //password
            autoCapitalize="none"
            icon="lock"
            placeholder="Password"
            placeholderTextColor="#FFFFFF"
            onChangeText={text => setPassword(text)}
            autoCorrect={false}
            secureTextEntry={true}
            textContentType="password"
        />
                    <View style={styles.space}>
                    <Buttons
                    title='Login'
                    backgroundColor="#000000"
                    color="#ffffff"
                    onPress={() => navigation.navigate("MainScreen")}
                    />
                    </View>

                    <View style={styles.backButton}>
                        <Buttons
                            title="Back"
                            backgroundColor="#ff0000"
                            color="#ffffff"
                            onPress={() => navigation.goBack()}
```

Figure 34a - LoginScreen.js code snippet.

4.3.6.2 Register Screen

The register screen requires the user to enter their first name, surname, date of birth, email address and password to be able to create an account in order to start to store the data into their account.

The textboxes have placeholders to indicate where to type in the email and password, as well as icons for designing purposes.

When logged in, it will lead back to the main screen and allow users to store data.

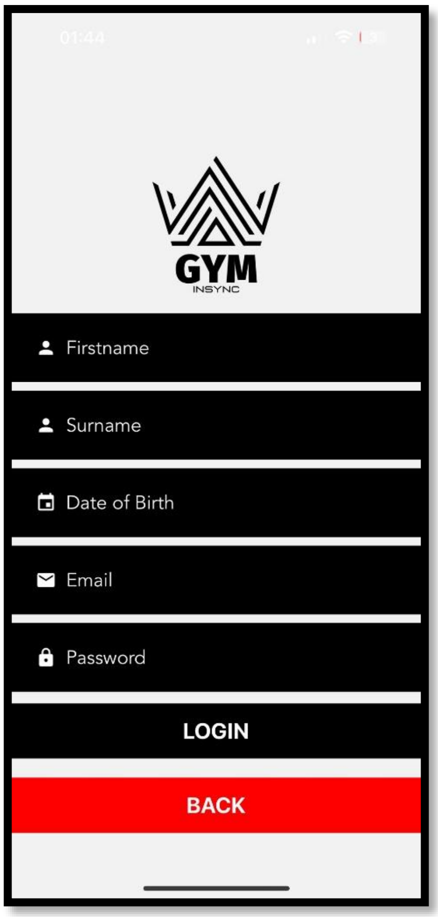
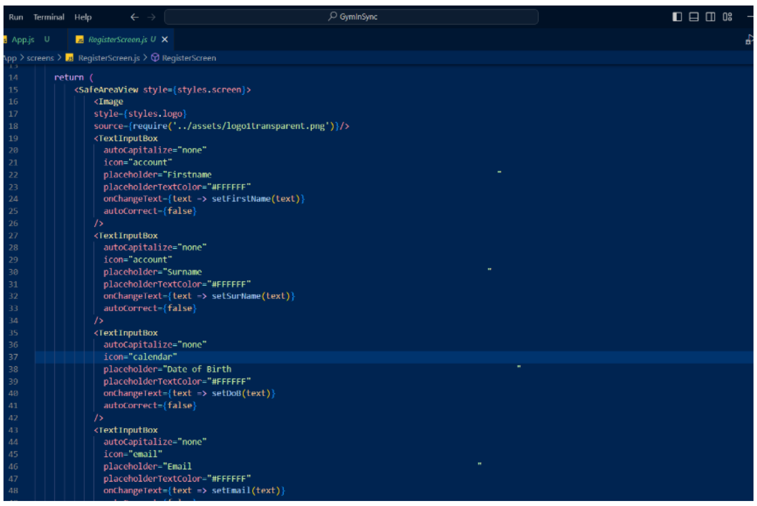


Figure 35 - Register Screen



```
Run  Terminal  Help      GymInSync
App.js  U     RegisterScreen.js U ×
pp > screens > RegisterScreen.js > RegisterScreen
    return (
        <SafeAreaView style={styles.screen}>
            <Image
            style={styles.logo}
            source={require('../assets/logo1transparent.png')}/>
            <TextInputBox
              autoCapitalize="none"
              icon="account"
              placeholder="Firstname                                    "
              placeholderTextColor="#FFFFFF"
              onChangeText={text => setFirstName(text)}
              autoCorrect={false}
            />
            <TextInputBox
              autoCapitalize="none"
              icon="account"
              placeholder="Surname                                     "
              placeholderTextColor="#FFFFFF"
              onChangeText={text => setSurName(text)}
              autoCorrect={false}
            />
            <TextInputBox
              autoCapitalize="none"
              icon="calendar"
              placeholder="Date of Birth                                  "
              placeholderTextColor="#FFFFFF"
              onChangeText={text => setDoB(text)}
              autoCorrect={false}
            />
            <TextInputBox
              autoCapitalize="none"
              icon="email"
              placeholder="Email                                     "
              placeholderTextColor="#FFFFFF"
              onChangeText={text => setEmail(text)}
```

Figure 35a - RegisterScreen.js code snippet.

5. Testing

The gym application is tested in five strategies, and each strategy can be applied to ensure that the application meets the design and function that was expected as a goal.

5.1 Functional Testing

Functional testing is done to ensure that the features of the app work smoothly and as they were intended such as the functions that were implemented.

Testing the functionality of the app resulted to a few pros and cons.

The pro is that about 60% of the application works as intended. Exercises can be favourited and shown in favourites, PRs can be stored as data and shown on the graph, the timer works perfectly fine without any errors, the navigation runs smoothly, and lastly the design of the screens meets the expectations.

But the cons were a few. When the calories card is pressed on the main screen, it does not navigate as it does not have a screen. This means that the functions for calculating deficit, surplus and maintenance calories do not function as they do not exist.

Another error displayed when testing the functionality of the app is a pop up on the history screen when navigating to it from the main screen. This might be due to an invalid number or other improper component. Also, there are only graphs for body weight and bodyfat percentage. This means other statistics cannot be displayed as a graph. Also, the 'reset data' button on the history screen does not fully reset the data unless the application is refreshed.

The last con about the mobile gym application is the fact that it does not store the data in the profiles. When a profile is created, the names, email addresses, and password are not stored because the firebase database is not implemented.

5.2 Usability Testing

Besides the functionality of the application, the interface looks visually pleasing and easy to follow. The navigation flows smoothly, and the content layouts are consistent through each screen. This makes it easy to navigate through the app and could be educational if only the details of the exercises were provided.

5.3 Performance Testing

The responsiveness of the applications is quite good. But in one case, it is quite slow.

When the exercises screen is opened up, it takes a few seconds for all the exercises to be rendered in. This may be due to the fact that there are too many items. This can be improved by using another function to render the items quicker. The procession of large amounts of data cannot be tested as the database is not setup.

5.4 Security Testing

Security testing cannot take place because no encryption or authentication methods are present in the application. This is because a database has not been implemented. This automatically fails the gym application in all security aspects.

5.5 Cross-Platform Compatibility Testing

Since the application is developed on React Native, the app should be run on both Android or iOS devices to ensure that the app appears and functions the same as expected. The application has been tested on an iOS device (iPhone 13 Pro Max) but has not been tested on other iOS devices. This could be fatal because different devices have different sizes of user interfaces.

The application has also not been tested on any Android device because no Android device was obtained. An Android emulator was installed on the device but did not run due to some complications, therefore this may fail the application in terms of offering Android devices the same service as iOS devices.

6. Evaluation

The GymInSync mobile application can enhance gym experience for users by offering a platform to gain educational insights as well as create workout routines and track fitness progress. However, the development of the mobile application was not as easy as expected due to facing many obstacles.

Prior to starting the prototype, the aim was to develop a gym application that is functional as well as user-friendly. The vision for the prototype was to achieve the initial objective, which is to provide educational information to enhance the knowledge if users and allow to track the gym progress in real-time by recording statistics.
It is evident that my personal journey and experience has allowed me to shape the application to address the exact needs such as the lack of knowledge of exercises.
Although the comparisons were made with other existing gym applications such as Pump Gym and FitNotes, a deeper analysis should have been made. A deeper analysis of the market could have potentially provided me greater strength in terms of relevance by giving a distinct value proposition for got GymInSync. Perhaps this would have allowed GymInSync to be more successful.

During the development, the software used were appropriate for mobile application development as it allowed me to develop a mobile application for both Android and iOS conveniently. However, one of the software was neglected which is the database behind the application. During the development of a database using firebase, I failed to implement it into GymInSync due to facing man errors and obstacles. So due to time constrict, the installation of a database was cancelled, and I continued the development of GymInSync. Perhaps in the future, I could be more prepared by attaining knowledge about setting up a database using firebase prior to starting the development of GymInSync which would have saved a lot of time. Due to errors and obstacles faced in firebase, a lot of valuable time was lost.
Also, a few functions were missing such as the calorie counter and setting a goal. This was neglected due to time constrict and focusing on the higher priority functions. Perhaps scaling down on a few features would have allowed me to ensure that the overall functionality of GymInSync is completed.

The design elements of GymInSync contributes to an engaging user experience as it was inspired by the interior of King's Gym. The monochromic colour scheme and logo made the visual appeal of GymInSync enhanced and made it look professional.
A story board should have been created before starting the development of GymInSync. This would have given me a much clearer vision of GymInSync and its overall design and functionality.

The list of exercises was big which led me to using the same image for each exercise. This could have been prevented if a list of only ten exercises was created since this is only a prototype. Creating a shorter list would have given me enough time to not only upload the appropriate images for each exercise, but to also add appropriate descriptions to each and perhaps a video tutorial to help users with the proper form to prevent injury. Incorporating these little features would have improved the GymInSync's educational value significantly.

Little features which would have enhanced the functionality of GymInSync were missed due to struggles. Some examples are adding audio to the timer to indicate that the timer has started, entered rest time, resumed time, and ended. Another example is the graphs for other statistics and adding exercises to workout plans. These were all missed due to facing difficulties in implementing them into GymInSync so it had to be skipped in order to focus on other aspects of GymInSync.

The testing of GymInSync's main functionalities such as navigating, favouriting, progress tracking and usability throughout the existing screens performed well. However, this does not justify missing other functionalities that were not implemented. The testing could have been done by beta testers. This would have allowed me to gather real feedback to find areas of improvement from people that attend the gym.

The security test failed which limits the functionality of GymInSync significantly. Due to this, data stored resets whenever GymInSync resets, so no data is stored permanently. It is all temporary. This could have been prevented if firebase was implemented successfully as it would have massively improved the security of the users by using authentication features. In the future, a robust security system as well as personalisation features need to be implemented. This can improve user engagement.

The cross-platform testing of GymInSync was not as extensive as expected. From all that I know, GymInSync might not even function properly on Android, or perhaps, it might not function on Android at all. I should have gotten hold of an Android device or downloaded an Android emulator on another device in order to test GymInSync and its functionality.
This is crucial because GymInSync cannot ensure a consistent user experience for both Android and iOS device users.

Overall, GymInSync has a good foundation in terms of addressing the needs of users that attend gym and fitness through technology. The development of GymInSync as well as its design have displayed my commitment in developing a useful mobile gym application for users that struggle to progress in gym and for fitness enthusiasts.
GymInSync is only a prototype, but the vision is clear and has great potential when developed fully. With constant improvements and developments over time, it could perhaps become a great mobile application to turn by users that start, or have already started, their gym journey.

9. Conclusion

The GymInSync mobile application was an attempted leap towards the merge of technology with personal fitness. The problem that is presently seen in the fitness industry and observed by the fitness centre is this:

- Through the selection of exercises
- Techniques
- The ability to monitor the progression

These observations all have likely resulted in injury to a member or two, or perhaps decreased motivations of a few members in the past.
It was intended to develop a prototype mobile application which could help overcome these issues with educational and friendly design, a guideline not for the usage of the aid only, but also for developing new assistive technologies.

For such developments, different tools and technologies were researched, among which the case includes cross-platform development technology with React Native and its related Expo tools, and Expo Go for easier mobile development, among others.
Node.js was used for the development of backend services, Firebase should have been used for the implementation of the real-time functionality of the database, and Visual Studio for developing the integrated development environment.
These tools were instruments utilised in crafting an application that is not only functional but also scalable and should have been secure.

The developed features of the application include an exercise screen, diary and history screens to track on the go, a timer for the round-based exercises which the user is following, and management of the user profile, all tailored through keeping in mind the journey of the user.
The goal was an exciting, interesting user flow—leading to the achievement of their fitness goal while keeping the application with strong points in usability and performance.

However, like any prototype, GymInSync has areas that require further development and refinement. Performance optimization and security testing were areas some of the challenges were faced, indicating that the research and development need to be continuous. These might include a larger exercise database, integration of health information from wearable devices, and even better security hardening to secure end-user data. Furthermore, testing the prototype on different devices will further be considered for a pleasant user experience in every device platform.

In conclusion, GymInSync embodies the potential of technology to enhance personal fitness. Its existence, in a way, substantiates the importance of user-centred design, education empowerment, and leveraging modern development tools in building applications that actually meet the requirements of gym members in the world at the moment. All the evaluations and recommendations will be considered in order to enhance the functionality and design of the mobile application, which would mold the growth of GymInSync into something more useful in order to achieve fitness and health.

8. Appendix

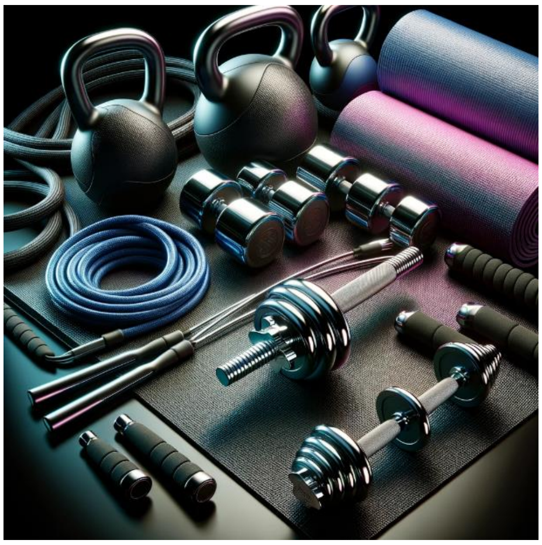


Figure 36 - Workout Card image.

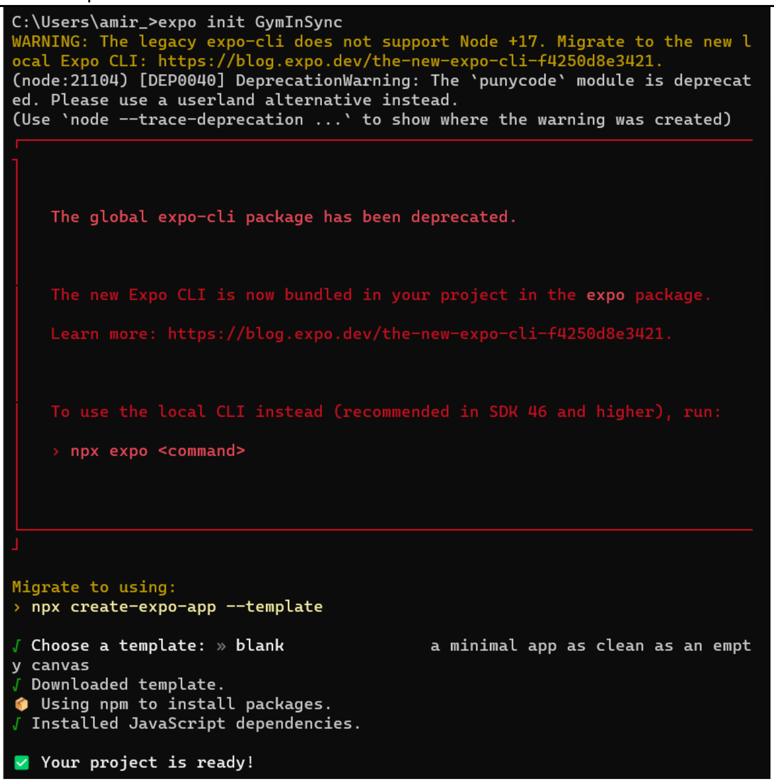
Figure 37 - Timer Card image.

Figure 39 - Diary Card image

Figure 38 - Calories Card image.

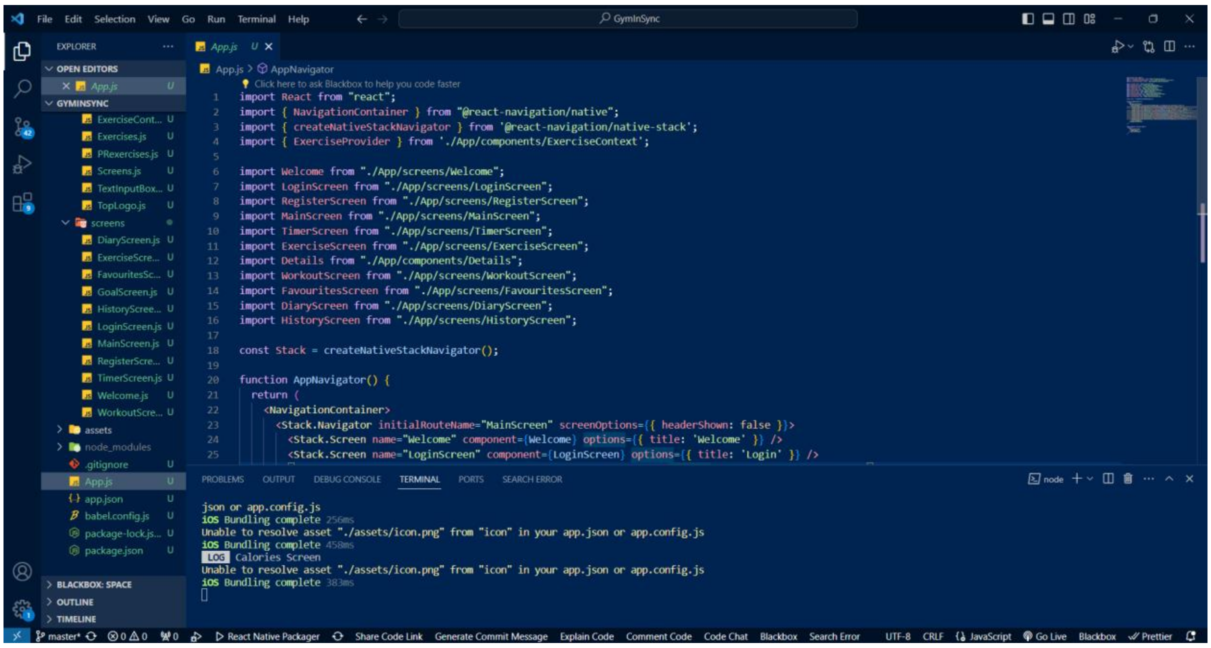
Figure 41 - Workout Card image

Figure 40 - Exercises Card image.



```
C:\Users\amir_>expo init GymInSync
WARNING: The legacy expo-cli does not support Node +17. Migrate to the new l
ocal Expo CLI: https://blog.expo.dev/the-new-expo-cli-f4250d8e3421.
(node:21104) [DEP0040] DeprecationWarning: The `punycode` module is deprecat
ed. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)


   The global expo-cli package has been deprecated.


   The new Expo CLI is now bundled in your project in the expo package.

   Learn more: https://blog.expo.dev/the-new-expo-cli-f4250d8e3421.


   To use the local CLI instead (recommended in SDK 46 and higher), run:

   › npx expo <command>



Migrate to using:
› npx create-expo-app --template

√ Choose a template: » blank               a minimal app as clean as an empt
y canvas
√ Downloaded template.
📦 Using npm to install packages.
√ Installed JavaScript dependencies.

✅ Your project is ready!
```

Figure 42 - Node.js complete installation to get project ready.



```
import React from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { ExerciseProvider } from './App/components/ExerciseContext';

import Welcome from "./App/screens/Welcome";
import LoginScreen from "./App/screens/LoginScreen";
import RegisterScreen from "./App/screens/RegisterScreen";
import MainScreen from "./App/screens/MainScreen";
import TimerScreen from "./App/screens/TimerScreen";
import ExerciseScreen from "./App/screens/ExerciseScreen";
import Details from "./App/components/Details";
import WorkoutScreen from "./App/screens/WorkoutScreen";
import FavouritesScreen from "./App/screens/FavouritesScreen";
import DiaryScreen from "./App/screens/DiaryScreen";
import HistoryScreen from "./App/screens/HistoryScreen";

const Stack = createNativeStackNavigator();

function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="MainScreen" screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Welcome" component={Welcome} options={{ title: 'Welcome' }} />
        <Stack.Screen name="LoginScreen" component={LoginScreen} options={{ title: 'Login' }} />
```

```
json or app.config.js
iOS Bundling complete 256ms
Unable to resolve asset "./assets/icon.png" from "icon" in your app.json or app.config.js
iOS Bundling complete 458ms
 LOG  Calories Screen
Unable to resolve asset "./assets/icon.png" from "icon" in your app.json or app.config.js
iOS Bundling complete 383ms
```

Figure 43 - Final code of App.js

9. References

Statesman News Service (2022) Gym Lifestyle, how the definition has changed in the recent times. https://www.thestatesman.com/lifestyle/gym-lifestyle-definition-1503103352.html

Tavares, B.F. et al. (2020) 'Mobile Applications for Training plan using Android Devices: A Systematic Review and a Taxonomy Proposal,' Information, 11(7), p. 343. https://doi.org/10.3390/info11070343

Android Application Development: A brief overview of Android platforms and evolution of security systems (2019b). https://ieeexplore.ieee.org/abstract/document/9032440

Domes, S., 2017. Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase. Packt Publishing Ltd.

J. An, W. Fan, A. Mittal, Y. Zhang, & A.T. Chen (2024).
Mobile App Use among Persons with Fibromyalgia: A Cross-Sectional Survey.
https://www.sciencedirect.com/science/article/abs/pii/S1526590024004358

M. Sideridou, E. Kouidi, V. Hatzitaki, & I. Chouvarda (2024).
Towards Automating Personal Exercise Assessment and Guidance with Affordable Mobile Technology.
https://www.mdpi.com/1424-8220/24/7/2037

A. Aminia, L. Hatamia, & M.T.A. Shavazib (2024). The Effect of Social Media Use on Job Performance: Exploring the Role of Technostress, Social Capital, and Job Satisfaction as Mediators.
https://jm.um.ac.ir/article_44714_a114dca985b785004e64f2489350b00f.pdf

O.M. Gushchina, E.V. Zhelnina, & E.A. Erofeeva. Methods of Educational and Digital Footprint Data Mining in the Formation of Multicomponent Model of Specialist Competencies.
https://www.elibrary.ru/item.asp?id=60949438

C.A. Franco & C.S. de Blas (2024). USING NONLINEAR LEARNING PATHS IN SELF-DETERMINED LEARNING.
https://library.iated.org/view/AGUADOFRANCO2024USI

S. Chan, J.S. Cho, C. Andrew, & D. Hao (2024). The modern anesthesiologist's manual: the development and maintenance of an anesthesia case reference application.
https://link.springer.com/article/10.1007/s10877-024-01153-2

R. Agrawal & N. Pandey. Strategies for Developing and Deploying Enterprise-Level Mobile Applications on a Large Scale: A Comprehensive Analysis.
https://www.researchgate.net/publication/379118002_Strategies_for_Developing_and_Deploying_Enterprise-Level_Mobile_Applications_on_a_Large_Scale_A_Comprehensive_Analysis

R. Sharma (2024). Innovative mobile application, Weather Wear: Where wardrobe meets the weather conditions.
https://esource.dbs.ie/items/a274e91d-cfdc-4a12-a6bf-4bba1471a7f6

Y. Zeier (2024). Identifying Frameworks in Android Applications using Binary Code Function

Similarity.
https://repositum.tuwien.at/handle/20.500.12708/195747

META (2024)
https://reactnative.dev/

Expo (2024)
https://expo.dev/

Images used from:

1) https://www.healthclubmanagement.co.uk/health-club-management-news/The-Gym-Group-plans-31m-warchest-to-fund-40-new-sites/348052
2) https://theoxfordmagazine.com/news/the-gym-group-has-opened-second-24-7-training-space-in-oxford/
3) https://www.buzzgym.co.uk/london-harrow/gallery/
4) https://www.puregym.com/gyms/hatfield/
5) https://www.puregym.com/gyms/south-ruislip
6) https://www.puregym.com/gyms/bracknell
7) https://www.photopea.com
8) https://www.canva.com