National College of Ireland

BSHC4

Software Development

2022/2023

Conor Odoemene

X19445706

X19445706@student.ncirl.ie

Youtine – A Mobile App

Technical Report

Contents

YOUTINE

by Conor Odoemene

WHAT IS YOUTINE

Youtine is a mobile application designed to improve one's mental health by addressing the key factors that can cause such issues.

FACTORS IMPACTING MENTAL HEALTH

- Lack of a routine
- Low self esteem
- Lack of physical stimulation/excercise
- Poor/unhealthy diet

3 DAILY CHALLENGES

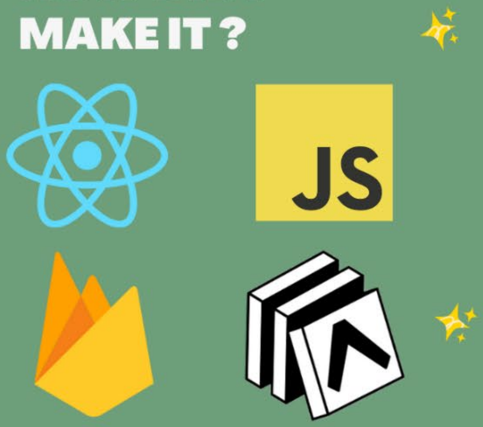
The 3 daily challenges is how Youtine intends to directly address the factors which can cause mental health issues.

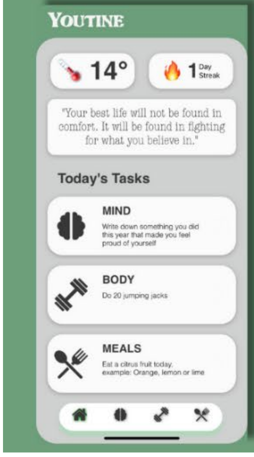
MIND CHALLENGE

BODY CHALLENGE

MEAL CHALLENGE

HOW DID I MAKE IT ?





YOUTINE

14°

1 Day Streak

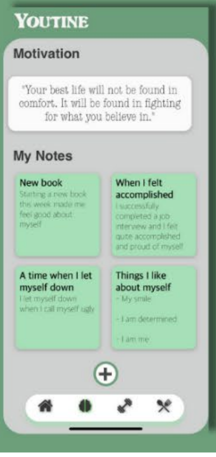
"Your best life will not be found in comfort. It will be found in fighting for what you believe in."

Today's Tasks

MIND
Write down something you did this year that made you feel proud of yourself

BODY
Do 20 jumping jacks

MEALS
Eat a citrus fruit today, example: Orange, lemon or lime



YOUTINE

Motivation

"Your best life will not be found in comfort. It will be found in fighting for what you believe in."

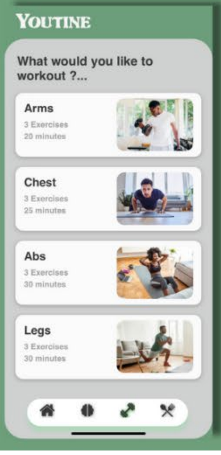
My Notes

New book
Starting a new book this week made me feel good about myself

When I felt accomplished
I successfully completed a job interview and I felt quite accomplished and proud of myself

A time when I let myself down
I let myself down when I call myself ugly

Things I like about myself
- My smile
- I am determined
- I am me



YOUTINE

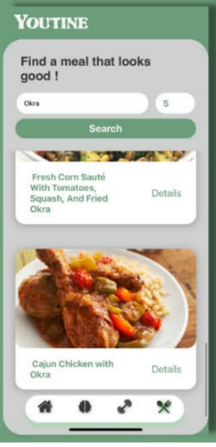
What would you like to workout ?...

Arms
3 Exercises
20 minutes

Chest
3 Exercises
25 minutes

Abs
3 Exercises
30 minutes

Legs
3 Exercises
30 minutes



YOUTINE

Find a meal that looks good !

Okra | 5

Fresh Corn Sauté With Tomatoes, Squash, And Fried Okra — Details

Cajun Chicken with Okra — Details

Executive Summary

According to the World Health Organization (WHO), mental illness affects one in four people globally (World Health Organization, 2001). This in short is the reason for Youtine. Youtine is a mobile application designed to improve the mental and physical well-being of its users through daily challenges that focus on the mind, body, and meals. in Ireland alone, 18.5% of the population experiences a mental health problem in any given year (Mental Health Ireland, 2015), and mental health issues are estimated to cost the economy up to €8.2 billion per year (Roe, 2021). Additionally, 37% of Irish people are considered overweight or obese, with poor nutrition and lack of physical activity contributing to these rates (HSE, 2015). Many studies have found that poor routines, low self-esteem, inadequate physical activity, and an unhealthy diet can all contribute to deteriorating mental health (Fay W. Boozman, 2022). Youtine aims to combat these issues by providing users with a clear and concise platform that encourages healthy habits that are simple and achievable but still effective.

Youtine is also particularly beneficial for young people who may struggle with the negative effects of social media and phone addiction. By using their phones for positive and healthy activities, young people can improve their mental and physical well-being, and develop lifelong habits that promote overall health.

The application was developed for iOS devices using react native, with a login functionality which is facilitated by firebase. Once logged in users can expect to be greeted by todays three challenges, an inspirational quote, the weather, and their daily streak count on the home screen.

Users can take and view notes in the app, a feature designed for helping promote reflection and mindfulness. Journaling and reflection have been proven to have positive effects on mental health, as they allow individuals to process their thoughts and emotions in a safe and constructive way (University of Rochester Medical Center, n.d.).

The app also provides simple workout routines that can be done at home, making it accessible for users of all fitness levels. Regular exercise has been shown to improve mental health by reducing stress and anxiety and increasing endorphin levels (Better Health Channel, 2021).

In addition to this, Youtine also offers a section for healthy meals, allowing users to easily search for and access nutritious recipes. A healthy diet has been linked to better physical and mental health outcomes, making it an important aspect of overall well-being (aetna, 2023).

1.0 Introduction

1.1. Background

In developing this project, my aim was not only to challenge myself by mastering cutting-edge technologies but also to craft a meaningful app that could resonate with a real-world audience. Today, mobile phones and social media have become deeply ingrained in our daily lives, offering countless advantages. However, they also contribute to the cultivation of an underlying problem that affects many individuals in my generation – the decline of mental health.

This issue is particularly prevalent among young men, and even more so among young Black men, who often face unique challenges in today's world. The deeply ingrained patriarchal norms of our society have led to men often feeling hesitant to express their emotions or seek help. This pressure to adhere to traditional gender roles can hinder open communication and perpetuate the harmful belief that vulnerability is a sign of weakness, this can be observed especially in African & Caribbean communities (Motswatswa, 2022). As someone who has been personally affected by this struggle, I feel a strong responsibility to address it using the tools I have at my disposal. In this project, I intend to harness the power of technology to help mitigate the very problem it has inadvertently helped create.

By combining my own experiences with a strong foundation of information, I aim to create an app that is not only innovative and engaging but also provides genuine support and resources for those who need it most. Through this endeavour, I hope to foster a more open and achievable environment for individuals dealing with mental health challenges and ultimately help them lead healthier, more fulfilling lives.

1.2. Aims

This project’s primary aim is to produce a polished iOS app with the purpose of being a part of a user’s daily routine. The four main features which are at the core of this app are:

• Registration & Login – Youtine uses Firebase to allow for users to register an account with our app and login.

• Tasks – The daily tasks are to be generated every 24 hours, with a checkbox that allows the user to mark each completed task as finished.

• Note taking – This feature allows for in-app recording of notes and displaying them back to the user.

• API’s – An API call to display the temperature and an inspirational quote on the home screen. An API call to display meals and recipes when searched for in the app meals screen.

1.3. Technology

These are some key technologies being used in this project, which were chosen after extensive research and consideration on which would best meet the objectives. They are as follows:

- React Native – The open-source UI software framework React Native was developed by Meta Platforms, Inc. It enables me to leverage the React framework combined with native platform features to create an iOS application with very minor code changes can also run-on Android.

- Expo – Expo is a framework that extends React Native, a software library built by Meta to develop native iOS and Android apps using a single codebase written in JavaScript. Expo allows me to test and debug the app on my personal iOS and Android device.

- Firebase – Google Firebase offers an extensive array of features and tools. For my project, I'll be focusing on utilising their Firestore and User Authentication services. Firestore, a flexible and scalable cloud-based database, will be employed for efficient data management, whilst User Authentication will ensure secure and personalised access for app users.

- AWS S3 - Amazon S3 (Simple Storage Service) is a highly scalable cloud-based storage service offered by Amazon Web Services (AWS). S3 allows users to store and retrieve large amounts of data.

- jest testing–Jest is a JavaScript testing framework that is commonly used for unit testing but can also be used for integration testing and class testing. It provides a simple and intuitive way to write tests for JavaScript code.

- Git & Github – In this project, I've embraced the use of Git and GitHub to enhance my personal development process. Git allows me to efficiently manage my codebase, while GitHub serves as an ideal platform for sharing my work. These tools have significantly streamlined my workflow and contributed to the project's organisation and my peace of mind.

1.4. Structure

The layout of this document is as follows:

- The project's motivations and a description of the technology employed.
- Project requirements that are both functional and non-functional, as well as a use case diagram that illustrates how users interact with the system.
- A look at the project's testing procedures.
- The project's conclusion.
- Additional outside documents that support this one.

2.0 System

2.1. Requirements

As Youtine is a self-improvement focused app that functions as a means of aiding one's personal growth in 3 specific categories by doing challenges. Therefore, the application must have a means to facilitate the undertaking of each challenge. The mind challenges are of specific importance as almost all tasks will be expected to be completed in the app.

2.1.1. Functional Requirements

• Platform compatibility: The app is designed specifically for iPhone users and must be compatible with various iPhone models and iOS versions.

• User authentication: The app requires users to register, log in, and log out to ensure secure and personalised access to the app's features.

• Data management: Registration and login data must be securely stored and managed using the Firebase Database to maintain user privacy.

• Mind challenge notes: Users should have the ability to add notes to their mind screen, cataloguing their thoughts and reflections during the challenge.

• Challenge completion tracking: The app must enable users to mark challenges as completed, allowing them to monitor their progress.

• User progress visualisation: A dedicated profile page should display the user's progress through the challenges, offering insights and motivation.

• Daily challenge generation: The app must generate three daily challenges in the specified categories, providing variety and promoting personal growth.

• Meal search functionality: Users should be able to search for meals within the app, with relevant results displayed based on their input.

2.1.2. Non-Functional Requirements

- Responsiveness: The app should be highly responsive, ensuring a smooth user experience during navigation and interaction with the features, such as adding notes, checking off challenges, and searching for meals.

- User Interface: Youtine should have an intuitive and visually appealing user interface, making it easy for users to navigate and engage with the app's features effectively.

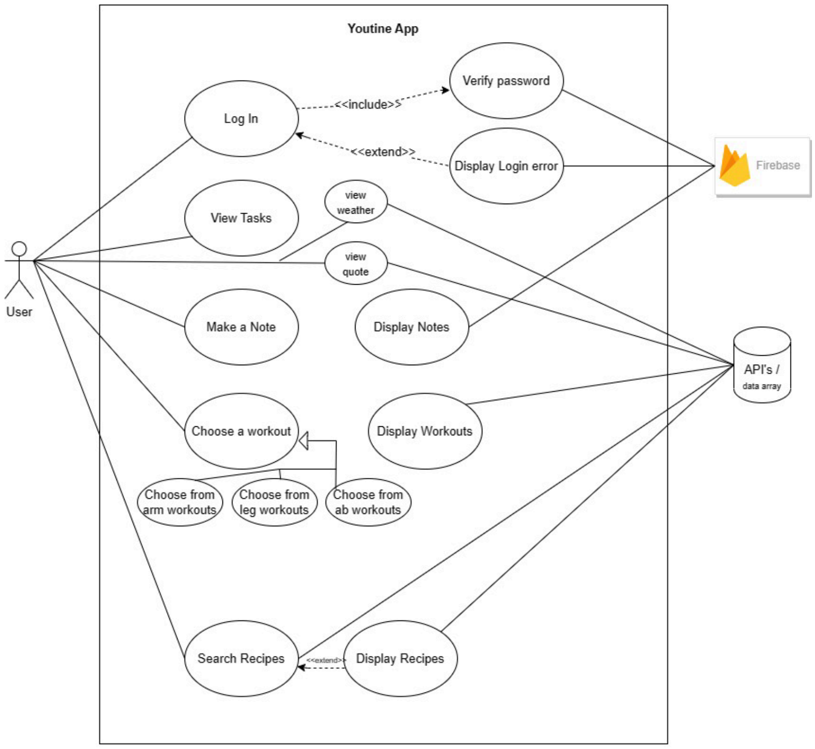
- Scalability: The app must be able to handle an increasing number of users and challenges without compromising performance or functionality.

- Data Security: User data, including personal information and challenge progress, must be securely stored, and protected to maintain user privacy and trust.

- Streamlined and Consistent UI: Youtine should have a cohesive and well-structured user interface that provides a seamless experience across all features and screens. This consistent design approach ensures that users can easily navigate and interact with the app, enhancing overall usability and user satisfaction.

2.1.3. Use Case Diagrams

2.1.3.1. Use Case Diagram – App Overview



2.1.3.2. Requirement 1 : Log in

Users must be logged in to access their profile, create notes, view workouts, and find recipes.

2.1.3.3. Description & Priority

For users to effectively track their progress and engage with Youtine's features, they must be able to access their profile, create notes, view workouts, and find recipes. Logging in ensures personalised and secure access to these essential functionalities. This requirement is of high priority as it is needed to access the app.

2.1.3.4. Use Case

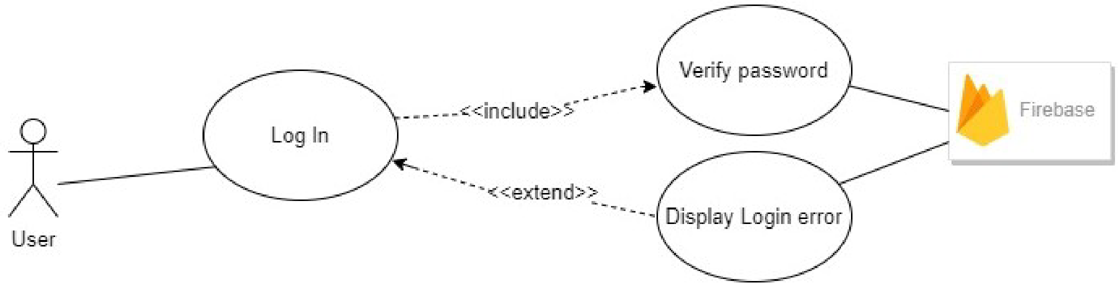
Scope

The scope of this use case is to ensure users can utilise this app freely as intended and for them to be able reap its benefits.

Description

This use case describes the process of logging in.

Use Case Diagram



Flow Description

Precondition

- The user starts up the app.
- The device is connected to the internet.

Activation

This use case begins when a user has started the app.

Main flow

1. The <User> inputs their log in details
2. Firebase verifies the password

Alternate flow

A1 : <Incorrect Password>
1. The <User> inputs their log in details
2. Firebase displays log in error

Exceptional flow

E1 : <No Internet>
1. The device has no internet connection & the system cannot establish a connection to Firebase.
3. The system shows “Connection Error.”

Termination

The app navigates to the Homepage

Post condition

The system goes into a wait state

2.1.3.5. Requirement 2 : Create and View Notes

Users must be logged in to create and view notes related to their challenges and personal growth journey.

2.1.3.6. Description & Priority

For users to effectively benefit from Youtine's self-improvement potential, they must be able to create and view notes to document their thoughts, reflections, and progress. This functionality allows users to better understand their growth journey and monitor their improvement. This requirement is of high priority, as it directly supports the app's goal of facilitating personal growth and self-improvement.

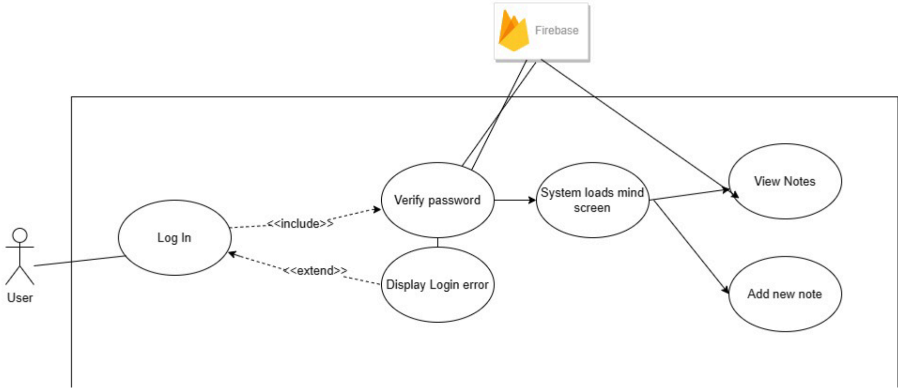
2.1.3.7. Use Case - Create and View Notes

Scope

The scope of this use case is to enable users to create and view notes within the app, allowing them to document their thoughts and reflections related to the challenges.

Description

This use case describes the process of creating and viewing notes. Use Case Diagram.



Flow Description

Precondition

• The user is logged in to the app.

• The device is connected to the internet.

Activation

- This use case begins when a user decides to create or view notes in the app.

Main flow

1. The <User> navigates to the mind section.
2. The <User> presses the '+' button for adding notes and is taken to the add notes screen.
3. The <User> inputs a title and writes their note below then presses 'save' button.
4. The note is displayed on mind screen

Alternate flow

A1 : <Missing Field>

4. The <User> does not input a title in the designated text area.
5. A note will not be saved.

Exceptional flow

A1 E1 : <No Internet>

1. The device has no internet connection & the system cannot sync the notes with the cloud.
2. The system shows a "Connection Error" message.

Termination

The note is created and displayed on the mind screen.

Post condition

The system goes into a wait state, ready for the user's next interaction.

2.1.3.8. Requirement 3 : Find a Meal

Users must be logged in and on the meals screen to search for meals, access meal details such as recipes, ingredients, and calorie information.

2.1.3.9. Description & Priority

For users maintain a balanced diet and effectively execute the meals daily challenge, they must be able to search for meals and access relevant details like recipes, ingredients, and calorie information. This functionality supports healthy eating habits and contributes to users' overall well-being. This requirement is of high priority.

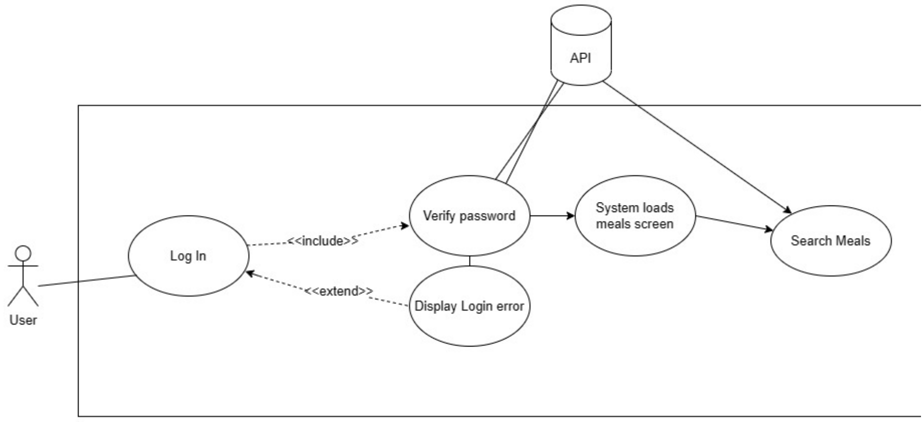
2.1.3.10. Use Case – Meal Search

Scope

The scope of this use case is to enable users to search for meals and access detailed information about them within the app.

Description

This use case describes the process of finding a meal using the Edamam API. Use Case Diagram.



Flow Description

Precondition

• The user is logged in to the app.

• The device is connected to the internet.

Activation

• This use case begins when a user decides to search for a meal in the app.

Main flow

1. The <User> navigates to the meals screen.
2. The <User> inputs a search prompt and submits the query.
3. The app returns a list of meals based on the search prompt using the Edamam API.
4. The <User> selects a meal to access further details, such as recipe, ingredients, and calorie information.

Alternate flow

A1 : <Missing Field>

1. The <User> does not input a search prompt in the designated text area.
2. Results will not appear

A2 : <No Results>

1. The <User> inputs a search prompt and submits the query.
2. No results are returned.

Exceptional flow

3. E1 : <No Internet>
    1. The device has no internet connection & the system cannot access the Edamam API.
    2. The system shows a "Connection Error" message.

Termination

The user views the meal details and returns to the meals screen or continues using other app features.

Post condition

The system goes into a wait state, ready for the user's next interaction.

List further functional requirements here, using the same structure as for Requirement1.

2.1.4. Data Requirements

- Weather API - Weather API - OpenWeatherMap
- Inspiration Quotes/Affirmation API - https://zenquotes.io/
- Recipe search API - Edamam - Food Database API, Nutrition API and Recipe API
- The user's personal information is required for the Registration & Login operations of this app.
- Firebase firestore is being used to securely store created users. Notes and the daily login streak will be linked to each user using an auto generated user ID.

2.1.5. User Requirements

- Users are required to have an email address.
- Users are required to have an iOS device.
- Internet connection is necessary.
- When registering for the app, users must enter their email address, and password.
- For the app's login procedure, users must enter an email address and password that match their login information.

2.1.6. Environmental Requirements

- Users need to be comfortable with iOS mobile devices.

- Regardless of whether they are using Wi-Fi or cellular data, users must use the app when online.

2.1.7. Usability Requirements

- The app is easy to navigate and user-friendly.
- All buttons must be clearly labelled or have obvious pictorial indication of the purpose they serve.
- Buttons and touchables must provide feedback to the user when actions are taken.
- The app is responsive and runs smoothly on all supported devices.

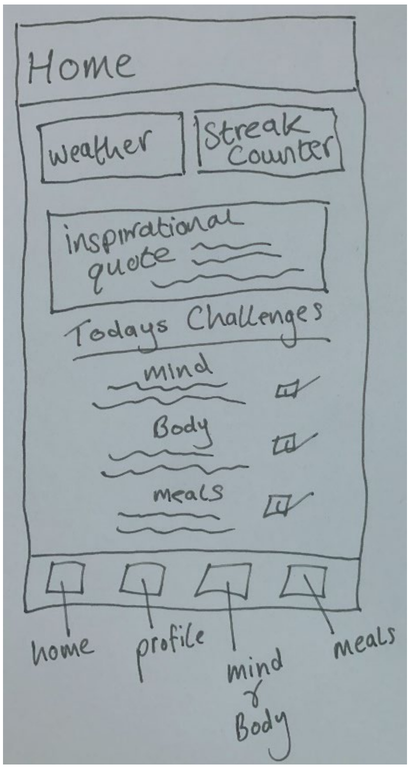
2.2. Design & Architecture

The mobile application, Youtine, was developed using React Native and written in JavaScript. The application utilises independent and reusable code snippets known as React components, which are efficiently laid out on each screen to avoid redundancy. To enhance user experience, the project design and architecture was chosen to be minimalist with rounded features and earthy tones. Visual Studio Code was employed for the app's development, and testing was carried out on an iPhone XR running the latest iOS version. Firebase was incorporated to provide user authentication and firestore storage capabilities, enabling users to create accounts and store notes securely within the application. All user data is encrypted and stored in Firebase, with the developer having full access to it.

2.2.1 Graphical User Interface (GUI)

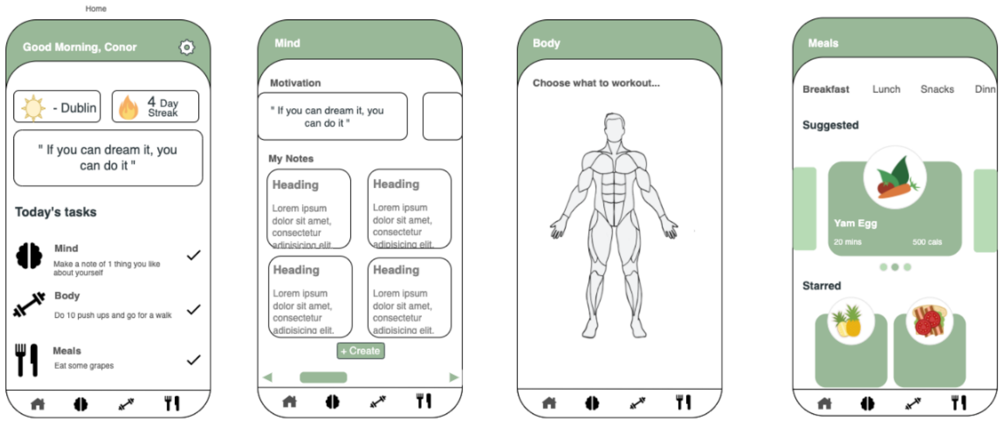
Low Fidelity - Earliest mock up

Drafts of the user interface were drawn on paper by me. This was intended to be an early design guideline for myself as I improved the fidelity of the UI .



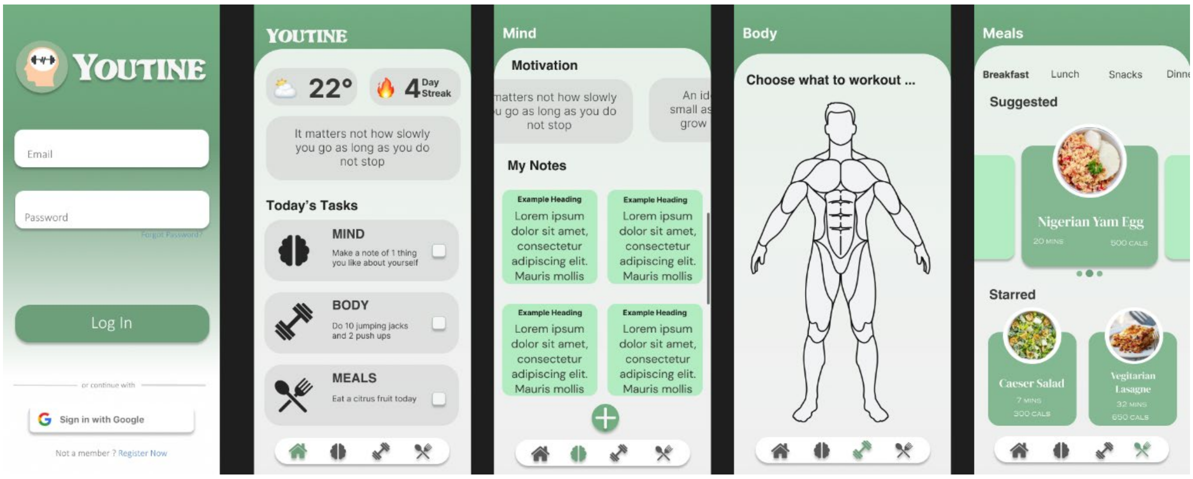
Pictured above is my initial concept for my home screen. It features a view (react native div equivalent) for the weather and daily login streak counter, followed by an inspirational quote and The 3 challenges for the day. There is a bottom navigation bar for navigating to the profile, mind & body and meals screens respectively. This layout would lay the foundations for future iterations of the UI.

Wireframes

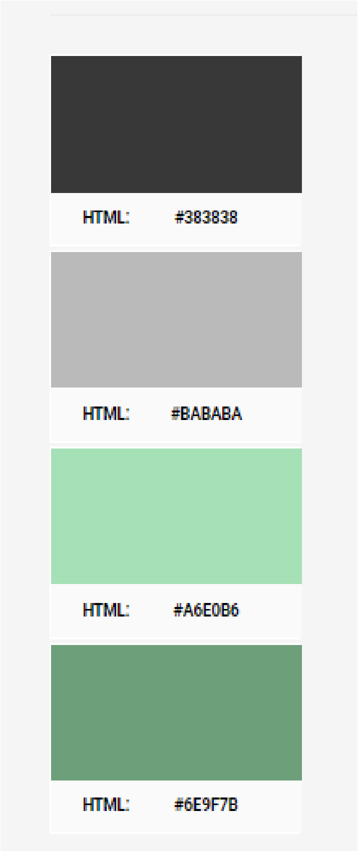

Home
Good Morning, Conor
- Dublin
4 Day Streak
" If you can dream it, you can do it "
Today's tasks
Mind
Make a note of 1 thing you like about yourself
Body
Do 10 push ups and go for a walk
Meals
Eat some grapes
Mind
Motivation
" If you can dream it, you can do it "
My Notes
Heading
Lorem ipsum dolor sit amet, consectetur adipisicing elit.
+ Create
Body
Choose what to workout...
Meals
Breakfast
Lunch
Snacks
Dinn
Suggested
Yam Egg
20 mins
500 cals
Starred

Pictured above are the wireframes I designed to visually describe the foundational appearance of the core pages for this application. The home page bears a striking resemblance to my initial drawing but there are some notable differences. From the top the 'Home' text has been replaced with some greeting text for the user. On the right of that you will see a cog icon, this was where a hypothetical settings screen was to be accessed from. Below the 3 views (weather, streaks & quote) remain relatively unchanged apart from their border radius which has been rounded. This radius change set a precedence for future designs. Moving on the 'Todays Challenges' text has been left aligned and altered to read 'Today's tasks' instead. This change was made because the alliteration was satisfying to readers and the shorter text looked more appropriated when aligned left. When looking at the tasks, we now find there is an associated image beside each one and the text as also now been aligned to the left. A button for the profile screen no longer resides on the bottom navigation bar and the mind & body screen has been separated into 2.

Stylised Mock up



I utilised Figma design tool to bring the wireframes to life and better visualise what they might look like on an actual device. The largest and most obvious change is the hovering bottom navigation bar concept. I added this as I felt rounded off corners was to be a key part of the application's design philosophy. The tasks on the home screen have also been separated onto views and a colour palette has been chosen for when the app goes into construction.



HTML: #383838

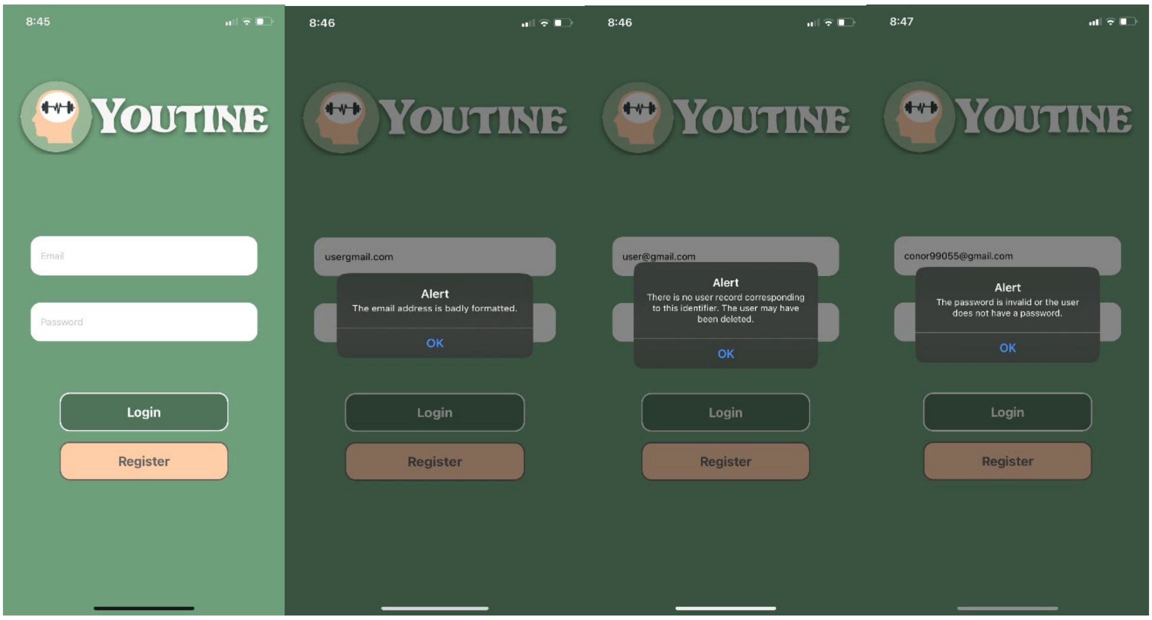
HTML: #BABABA

HTML: #A6E0B6

HTML: #6E9F7B

Pictured above is the colour palette .

High Fidelity Application Screenshots

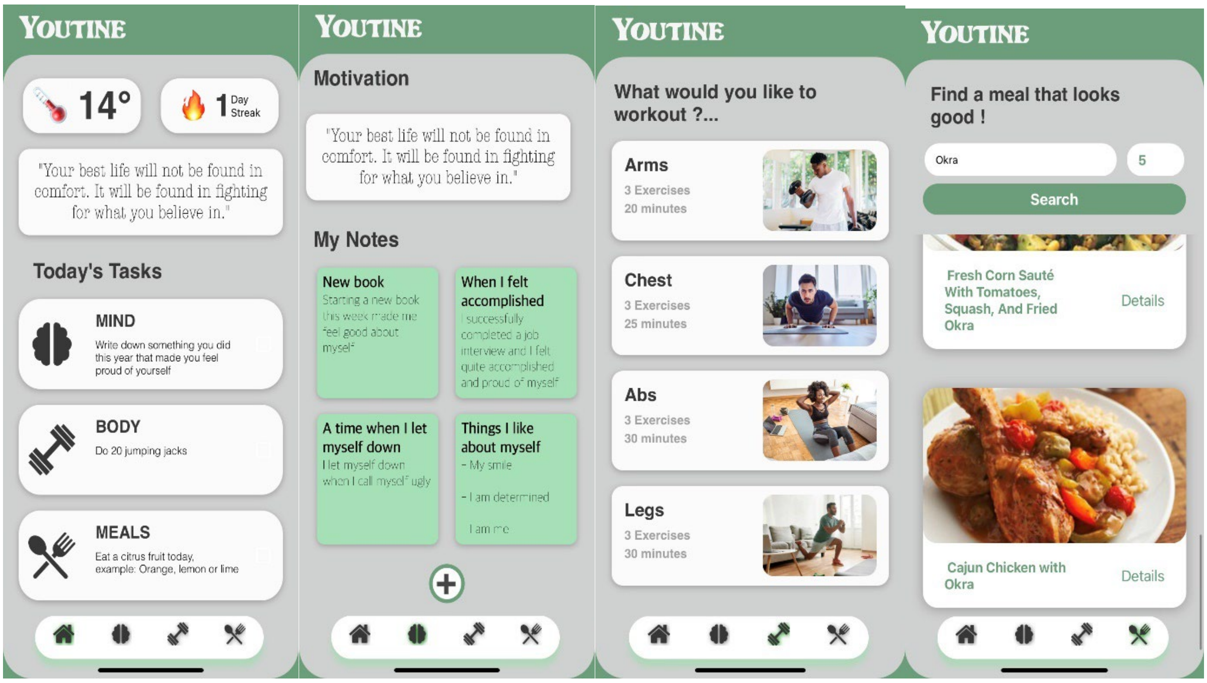


Alert
The email address is badly formatted.

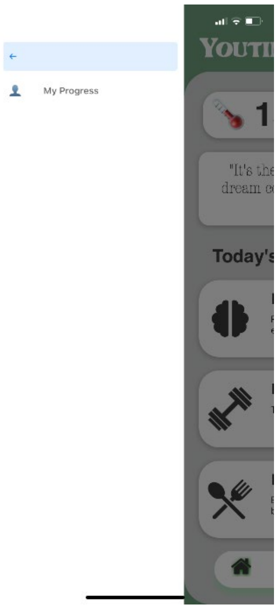
Alert
There is no user record corresponding to this identifier. The user may have been deleted.

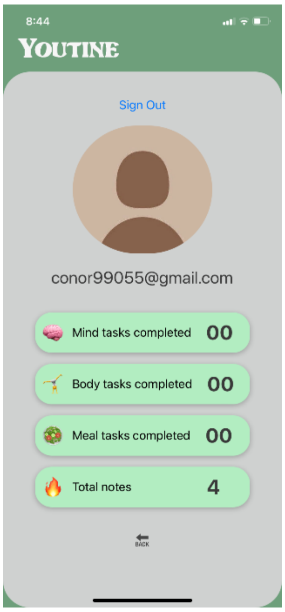
Alert
The password is invalid or the user does not have a password.

Pictured above is the login screen users are greeted with when they start the app, it is simple and clear. There are 2 inputs for the email and password respectively and 2 buttons, login and register. The user presses whichever button is applicable to them.

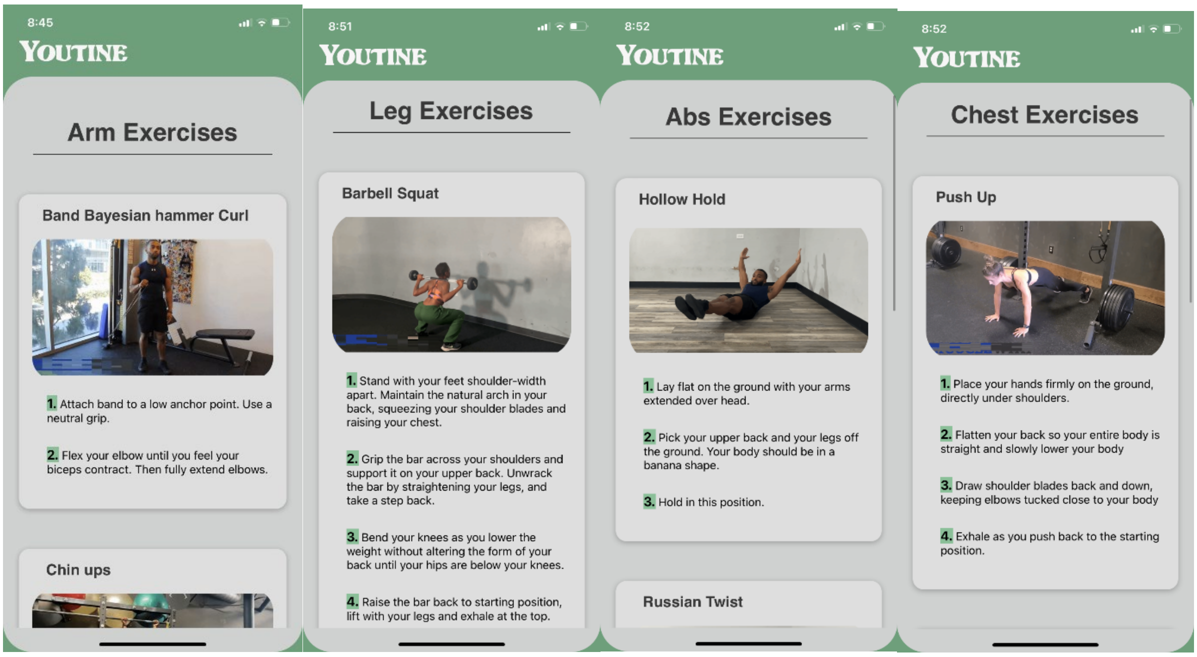

YOUTINE
14°
1 Day Streak
"Your best life will not be found in comfort. It will be found in fighting for what you believe in."
Today's Tasks
MIND
Write down something you did this year that made you feel proud of yourself
BODY
Do 20 jumping jacks
MEALS
Eat a citrus fruit today, example: Orange, lemon or lime
YOUTINE
Motivation
"Your best life will not be found in comfort. It will be found in fighting for what you believe in."
My Notes
New book
Starting a new book this week made me feel good about myself
When I felt accomplished
I successfully completed a job interview and I felt quite accomplished and proud of myself
A time when I let myself down
I let myself down when I call myself ugly
Things I like about myself
- My smile
- I am determined
I am me
YOUTINE
What would you like to workout ?...
Arms
3 Exercises
20 minutes
Chest
3 Exercises
25 minutes
Abs
3 Exercises
30 minutes
Legs
3 Exercises
30 minutes
YOUTINE
Find a meal that looks good !
Okra
5
Search
Fresh Corn Sauté With Tomatoes, Squash, And Fried Okra
Details
Cajun Chicken with Okra
Details

Once logged in users will be presented to the home screen where they can then use the bottom navigation to move between screens. The body and meals page saw quite an overhaul during the app implementation, both now having a cleaner look which takes less explanation for when the user interacts with them. Users must log in to access these sections of the application.





From the home screen users can now access the side drawer with a swipe right from the edge of the screen. The drawer houses a back button and a button labelled ‘My Progress’ which takes us to our profile, this button was previously found on the bottom tab navigator in my initial drawing. On the profile screen we will find a sign out button and a summary of the progress each user has made.



8:45

YOUTINE

Arm Exercises

Band Bayesian hammer Curl

1. Attach band to a low anchor point. Use a neutral grip.

2. Flex your elbow until you feel your biceps contract. Then fully extend elbows.

Chin ups

8:51

YOUTINE

Leg Exercises

Barbell Squat

1. Stand with your feet shoulder-width apart. Maintain the natural arch in your back, squeezing your shoulder blades and raising your chest.

2. Grip the bar across your shoulders and support it on your upper back. Unwrack the bar by straightening your legs, and take a step back.

3. Bend your knees as you lower the weight without altering the form of your back until your hips are below your knees.

4. Raise the bar back to starting position, lift with your legs and exhale at the top.

8:52

YOUTINE

Abs Exercises

Hollow Hold

1. Lay flat on the ground with your arms extended over head.

2. Pick your upper back and your legs off the ground. Your body should be in a banana shape.

3. Hold in this position.

Russian Twist

8:52

YOUTINE

Chest Exercises

Push Up

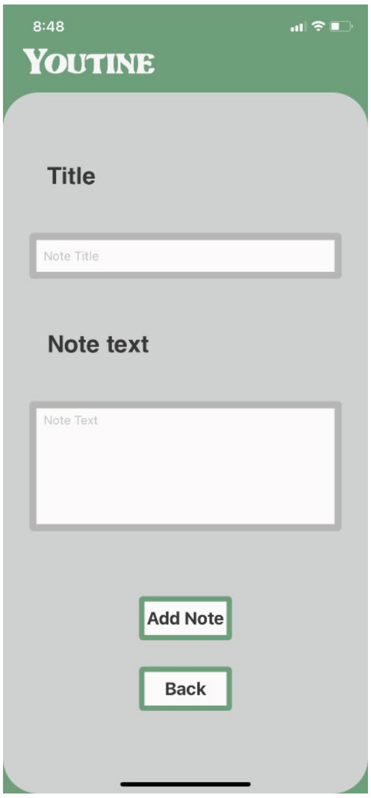
1. Place your hands firmly on the ground, directly under shoulders.

2. Flatten your back so your entire body is straight and slowly lower your body

3. Draw shoulder blades back and down, keeping elbows tucked close to your body

4. Exhale as you push back to the starting position.

The body page features options to help the users on a workout, these workouts are intended to help facilitate the completion of the body task. Each workout has a title, a video demonstration, and instructions.



8:48

YOUTINE

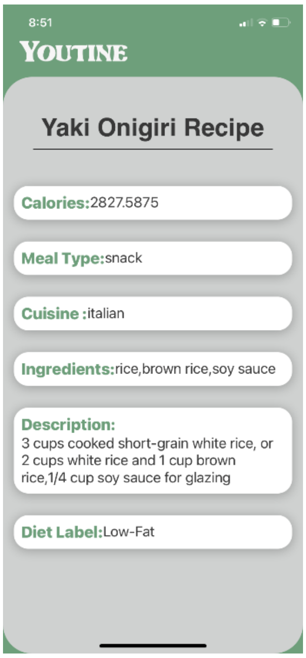
Title

Note Title

Note text

Note Text

Adding a note is simple. Users need only input a title and then the body of the note and press ‘Add Note’. The Back button will take users back to the mind page.



8:51

YOUTINE

Yaki Onigiri Recipe

Calories:2827.5875

Meal Type:snack

Cuisine :italian

Ingredients:rice,brown rice,soy sauce

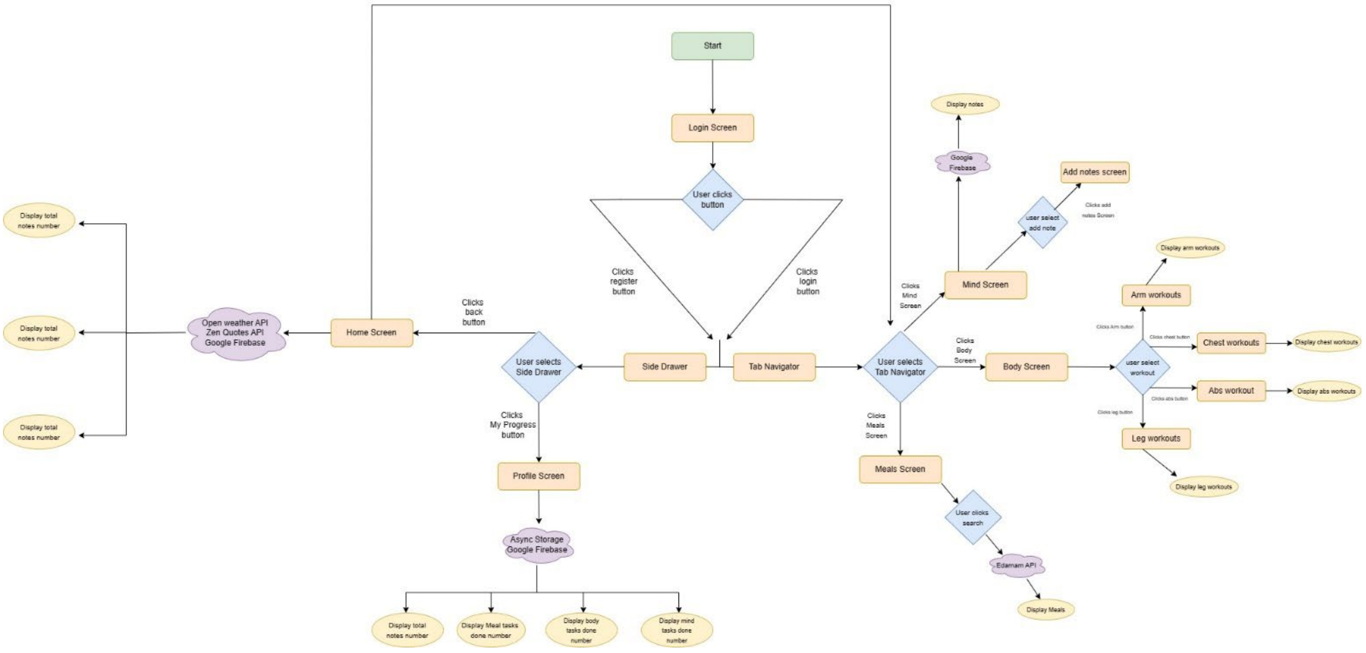
Description:
3 cups cooked short-grain white rice, or 2 cups white rice and 1 cup brown rice,1/4 cup soy sauce for glazing

Diet Label: Low-Fat

When a user searches for a meal on the meals page, each meal has a details button which presents information on the meal in question.

scroll vertically.

2.2.2 Activity Flow Diagram

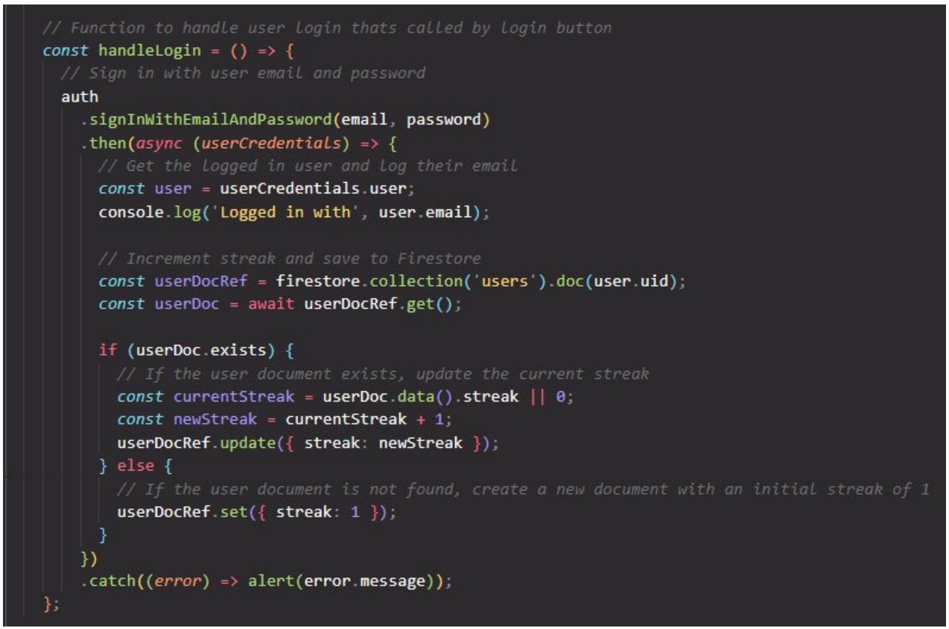


3.0 Implementation

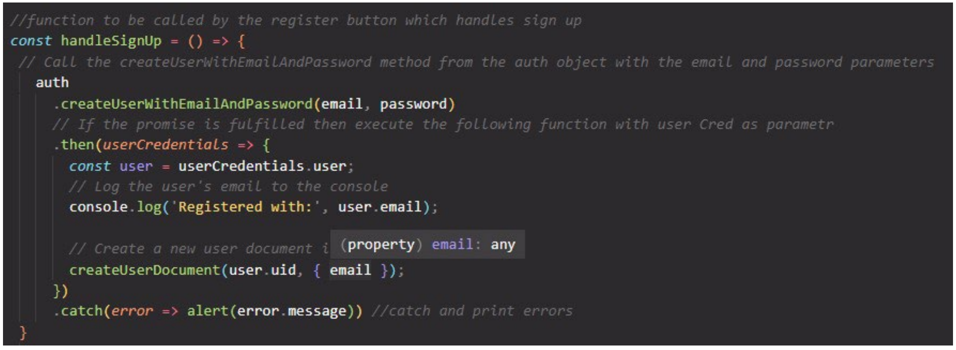
3.0.1 Login and Registration

In the LoginScreen.js component, the handleSignUp and handleLogin functions were implemented to manage user authentication and Firestore database updates. The handleSignUp function is invoked when a user enters their email and password to create a new account. The auth module's createUserWithEmailAndPassword method is called to create a new user with the provided credentials. Once the user is successfully created, the user's information is logged to the console and a new user document is created in Firestore using the createUserDocument function.

The handleLogin function is invoked when a user logs in with their existing email and password. The auth module's signInWithEmailAndPassword method is called to authenticate the user. Once the user is successfully authenticated, their information is logged to the console, and their streak count is updated in Firestore. The user's streak count is retrieved from the Firestore database using the user's unique ID. If the user already has a streak count in their document, the current streak count is incremented, and the updated count is saved to Firestore. If the user is logging in for the first time and a document with their ID is not found, a new document is created with an initial streak count of 1.



```
// Function to handle user login thats called by login button
const handleLogin = () => {
  // Sign in with user email and password
  auth
    .signInWithEmailAndPassword(email, password)
    .then(async (userCredentials) => {
      // Get the logged in user and log their email
      const user = userCredentials.user;
      console.log('Logged in with', user.email);

      // Increment streak and save to Firestore
      const userDocRef = firestore.collection('users').doc(user.uid);
      const userDoc = await userDocRef.get();

      if (userDoc.exists) {
        // If the user document exists, update the current streak
        const currentStreak = userDoc.data().streak || 0;
        const newStreak = currentStreak + 1;
        userDocRef.update({ streak: newStreak });
      } else {
        // If the user document is not found, create a new document with an initial streak of 1
        userDocRef.set({ streak: 1 });
      }
    })
    .catch((error) => alert(error.message));
};
```



```
//function to be called by the register button which handles sign up
const handleSignUp = () => {
 // Call the createUserWithEmailAndPassword method from the auth object with the email and password parameters
   auth
     .createUserWithEmailAndPassword(email, password)
     // If the promise is fulfilled then execute the following function with user Cred as parametr
     .then(userCredentials => {
       const user = userCredentials.user;
       // Log the user's email to the console
       console.log('Registered with:', user.email);

       // Create a new user document i (property) email: any
       createUserDocument(user.uid, { email });
     })
     .catch(error => alert(error.message)) //catch and print errors
 }
```

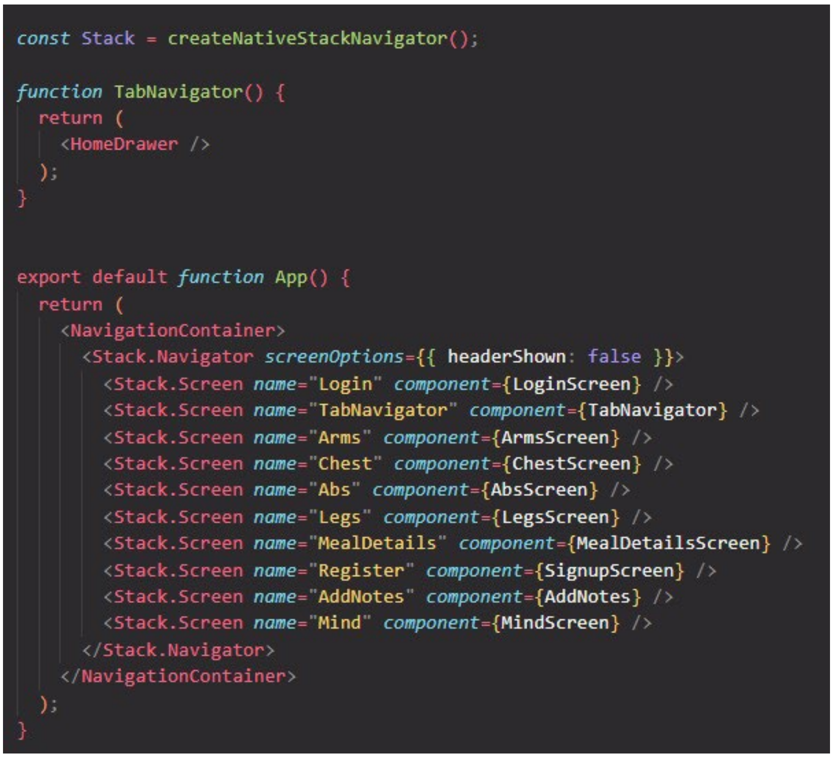
Overall, the implementation of these functions in the LoginScreen.js component ensures that user authentication is properly managed and Firestore database updates are correctly handled. These functions allow for the creation and management of user accounts, as well as the tracking of their daily streaks, contributing to a personalised and motivating user experience.

3.0.2 Application Navigation

The ‘single stack navigator restraint’ refers to the limitation of the StackNavigator in React Navigation to manage only one stack of screens at a time. That is, when you want to navigate from one stack of screens to another, you need to replace the entire StackNavigator and lose the ability to go back to previous screens in the original stack.

To overcome this limitation, the code below uses multiple navigators. Specifically, it uses a DrawerNavigator and a TabNavigator in addition to the StackNavigator. The DrawerNavigator is used as the root navigator and contains the TabNavigator, which has its own stack of screens. By structuring the navigation in this way, the StackNavigator is only responsible for navigating between screens within a specific stack, while the TabNavigator can handle navigation between different stacks.

This structure allows for more flexible navigation within the app, with the ability to go back to previous screens in a specific stack while also navigating between different stacks of screens.

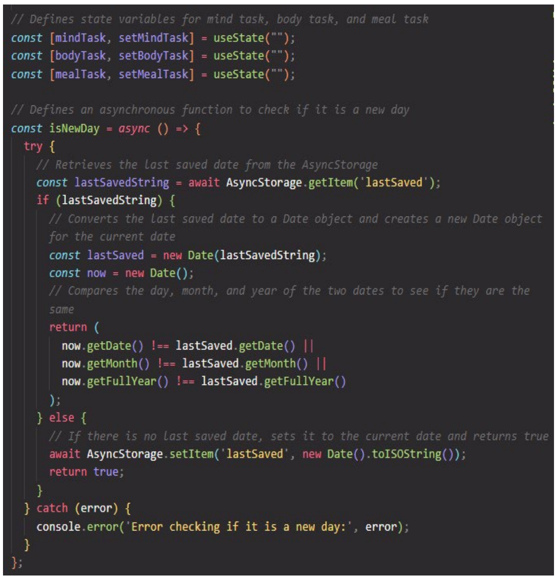


```
const Stack = createNativeStackNavigator();

function TabNavigator() {
  return (
    <HomeDrawer />
  );
}


export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="TabNavigator" component={TabNavigator} />
        <Stack.Screen name="Arms" component={ArmsScreen} />
        <Stack.Screen name="Chest" component={ChestScreen} />
        <Stack.Screen name="Abs" component={AbsScreen} />
        <Stack.Screen name="Legs" component={LegsScreen} />
        <Stack.Screen name="MealDetails" component={MealDetailsScreen} />
        <Stack.Screen name="Register" component={SignupScreen} />
        <Stack.Screen name="AddNotes" component={AddNotes} />
        <Stack.Screen name="Mind" component={MindScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

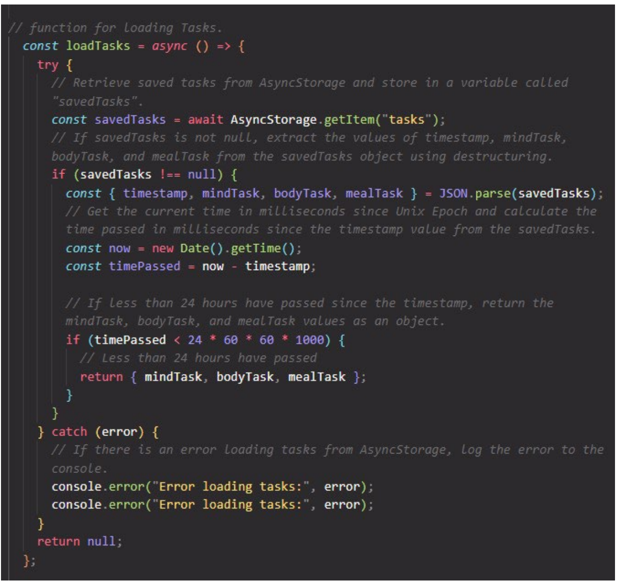
1.0.1 Adding and Retrieving Note

Youtine provides users with daily tasks . To retrieve these tasks, the application uses two main functions, loadTasks and isNewDay. The loadTasks function retrieves previously saved tasks from the local storage using the AsyncStorage API. If there are no saved tasks or if the saved tasks were completed more than 24 hours ago, the function generates new daily tasks and saves them to the local storage. Otherwise, the function returns the previously saved tasks. This process is achieved by first checking if the saved tasks exist in the storage using the getItem method of the AsyncStorage API. If the saved tasks exist, the function extracts the tasks' properties and checks the time passed since they were saved using the getTime method. If less than 24 hours have passed, the function returns the saved tasks. Otherwise, the function generates new tasks and saves them to the storage using the setItem method of the AsyncStorage API.

The isNewDay function checks if a new day has started since the last saved task. It does this by retrieving the last saved date from the local storage and comparing it with the current date using the getDate, getMonth, and getFullYear methods of the Date API. If the saved date is not available, the function sets the last saved date to the current date and returns true, indicating that a new day has started. If there is an error in retrieving the last saved date, the function returns false. This function is used to determine whether to generate new daily tasks or return previously saved tasks. If a new day has started, the function generates new tasks and saves them to the local storage. Otherwise, the function returns the saved tasks. The isNewDay function works in tandem with the loadTasks function to ensure that users receive new daily tasks every 24 hours, providing a fresh and personalized experience for each user.



```
// Defines state variables for mind task, body task, and meal task
const [mindTask, setMindTask] = useState("");
const [bodyTask, setBodyTask] = useState("");
const [mealTask, setMealTask] = useState("");

// Defines an asynchronous function to check if it is a new day
const isNewDay = async () => {
  try {
    // Retrieves the last saved date from the AsyncStorage
    const lastSavedString = await AsyncStorage.getItem('lastSaved');
    if (lastSavedString) {
      // Converts the last saved date to a Date object and creates a new Date object
      for the current date
      const lastSaved = new Date(lastSavedString);
      const now = new Date();
      // Compares the day, month, and year of the two dates to see if they are the
      same
      return (
        now.getDate() !== lastSaved.getDate() ||
        now.getMonth() !== lastSaved.getMonth() ||
        now.getFullYear() !== lastSaved.getFullYear()
      );
    } else {
      // If there is no last saved date, sets it to the current date and returns true
      await AsyncStorage.setItem('lastSaved', new Date().toISOString());
      return true;
    }
  } catch (error) {
    console.error('Error checking if it is a new day:', error);
  }
};
```

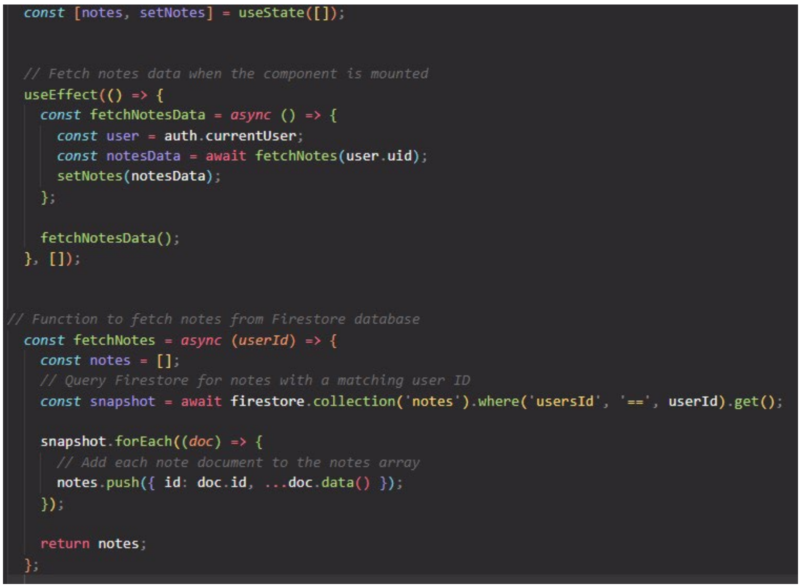


```
// function for loading Tasks.
  const loadTasks = async () => {
    try {
      // Retrieve saved tasks from AsyncStorage and store in a variable called
      "savedTasks".
      const savedTasks = await AsyncStorage.getItem("tasks");
      // If savedTasks is not null, extract the values of timestamp, mindTask,
      bodyTask, and mealTask from the savedTasks object using destructuring.
      if (savedTasks !== null) {
        const { timestamp, mindTask, bodyTask, mealTask } = JSON.parse(savedTasks);
        // Get the current time in milliseconds since Unix Epoch and calculate the
        time passed in milliseconds since the timestamp value from the savedTasks.
        const now = new Date().getTime();
        const timePassed = now - timestamp;

        // If less than 24 hours have passed since the timestamp, return the
        mindTask, bodyTask, and mealTask values as an object.
        if (timePassed < 24 * 60 * 60 * 1000) {
          // Less than 24 hours have passed
          return { mindTask, bodyTask, mealTask };
        }
      }
    } catch (error) {
      // If there is an error loading tasks from AsyncStorage, log the error to the
      console.
      console.error("Error loading tasks:", error);
      console.error("Error loading tasks:", error);
    }
    return null;
  };
```

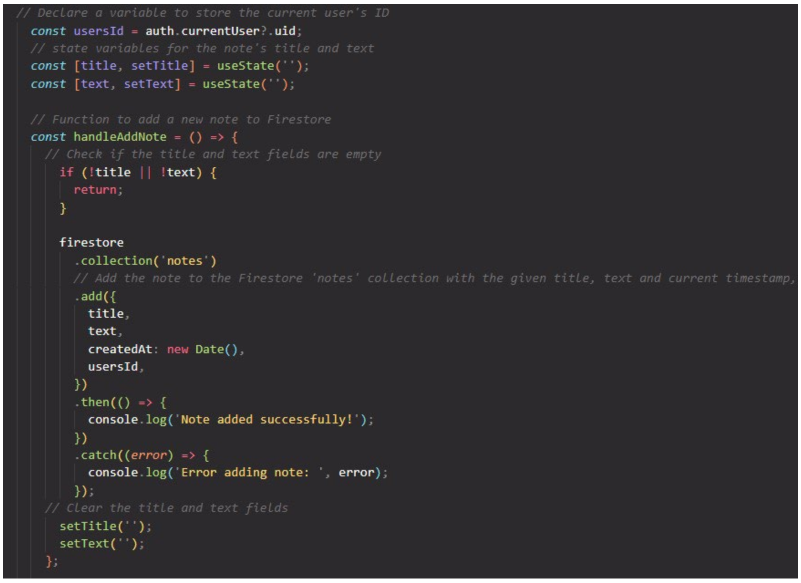
1.0.2 Adding and Retrieving Notes

The MindScreen.js component in Youtine utilises the useEffect hook to fetch the user's notes data from Firestore. This is achieved by calling the fetchNotesData function once when the component mounts, as specified in the empty dependency array argument. The fetchNotesData function is an asynchronous function that calls the fetchNotes function, which retrieves all the notes created by the logged-in user from Firestore. The notes data is stored in the state using the setNotes function.

The AddNotes.js component is responsible for adding notes to the Firestore database. The handleAddNote function is called when the user clicks the "Add Note" button. The function first checks if both the title and text fields are not empty before adding the note. Once validated, the function creates a new note object containing the title, text, the user's ID, and the current timestamp. This note object is then added to the Firestore notes collection using the Firestore add() method. If the note is successfully added to the collection, the state is updated by calling the setTitle and setText functions to clear the input fields.



```
  const [notes, setNotes] = useState([]);


  // Fetch notes data when the component is mounted
  useEffect(() => {
    const fetchNotesData = async () => {
      const user = auth.currentUser;
      const notesData = await fetchNotes(user.uid);
      setNotes(notesData);
    };

    fetchNotesData();
  }, []);


// Function to fetch notes from Firestore database
  const fetchNotes = async (userId) => {
    const notes = [];
    // Query Firestore for notes with a matching user ID
    const snapshot = await firestore.collection('notes').where('usersId', '==', userId).get();

    snapshot.forEach((doc) => {
      // Add each note document to the notes array
      notes.push({ id: doc.id, ...doc.data() });
    });

    return notes;
  };
```



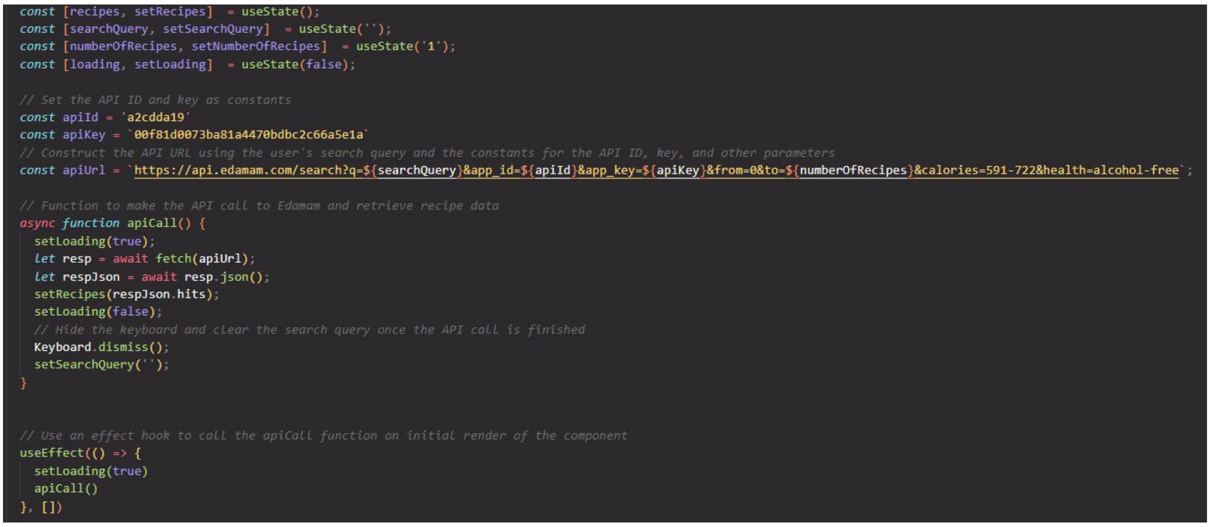
```
 // Declare a variable to store the current user's ID
  const usersId = auth.currentUser?.uid;
  // state variables for the note's title and text
  const [title, setTitle] = useState('');
  const [text, setText] = useState('');

  // Function to add a new note to Firestore
  const handleAddNote = () => {
    // Check if the title and text fields are empty
      if (!title || !text) {
        return;
      }

      firestore
        .collection('notes')
        // Add the note to the Firestore 'notes' collection with the given title, text and current timestamp,
        .add({
          title,
          text,
          createdAt: new Date(),
          usersId,
        })
        .then(() => {
          console.log('Note added successfully!');
        })
        .catch((error) => {
          console.log('Error adding note: ', error);
        });
    // Clear the title and text fields
      setTitle('');
      setText('');
    };
```

The MindScreen and AddNotes components effectively interact with Firestore, which enables users to store and retrieve their notes in real-time, providing a smooth user experience.

1.0.3 Search Meals

In the MealsScreen component, the state is initialised using the useState() hook. The 'recipes' state variable is used to store the data returned from the API call, while the 'searchQuery' state variable is used to store the user's search query input. The 'numberOfRecipes' state variable is used to determine the number of recipes to be displayed. The 'loading' state variable is used to track if the API call is in progress.

The API call is made using the fetch() method, which sends a GET request to the specified API endpoint. The 'apiUrl' variable is used to store the API endpoint URL constructed using the 'searchQuery', 'apiId', and 'apiKey' variables. The 'apiCall()' function is declared using the async keyword to ensure that the function runs asynchronously. The function first sets the 'loading' state variable to true before fetching data from the API endpoint using the fetch() method. The returned data is then converted to JSON format using the json() method. The 'recipes' state variable is then updated with the data using the setRecipes() method, and 'loading' is set back to false. Finally, the 'Keyboard.dismiss()' method is called to dismiss the keyboard and the 'searchQuery' state variable is reset to an empty string.

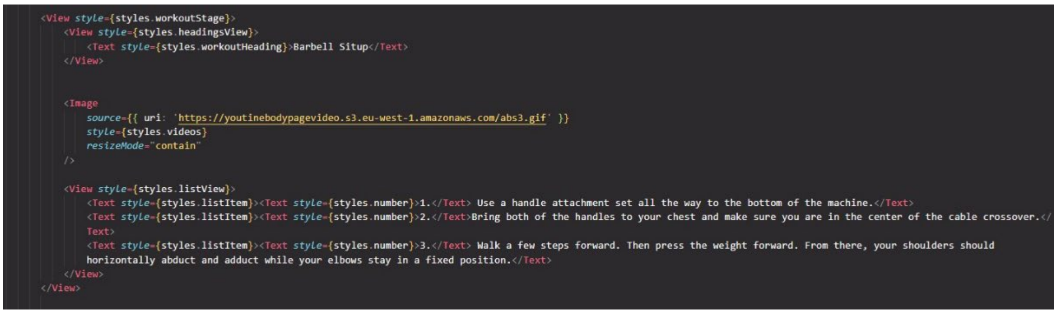


```
const [recipes, setRecipes]  = useState();
const [searchQuery, setSearchQuery]  = useState('');
const [numberOfRecipes, setNumberOfRecipes]  = useState('1');
const [loading, setLoading]  = useState(false);

// Set the API ID and key as constants
const apiId = 'a2cdda19'
const apiKey = `00f81d0073ba81a4470bdbc2c66a5e1a`
// Construct the API URL using the user's search query and the constants for the API ID, key, and other parameters
const apiUrl = `https://api.edamam.com/search?q=${searchQuery}&app_id=${apiId}&app_key=${apiKey}&from=0&to=${numberOfRecipes}&calories=591-722&health=alcohol-free`;

// Function to make the API call to Edamam and retrieve recipe data
async function apiCall() {
  setLoading(true);
  let resp = await fetch(apiUrl);
  let respJson = await resp.json();
  setRecipes(respJson.hits);
  setLoading(false);
  // Hide the keyboard and clear the search query once the API call is finished
  Keyboard.dismiss();
  setSearchQuery('');
}


// Use an effect hook to call the apiCall function on initial render of the component
useEffect(() => {
  setLoading(true)
  apiCall()
}, [])
```

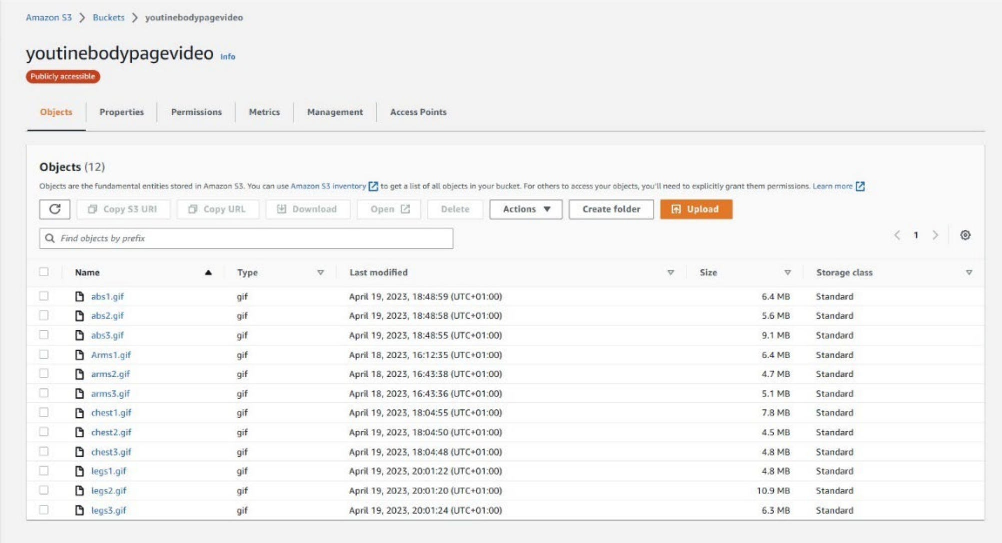
The useEffect() hook is used to automatically run the 'apiCall()' function when the component mounts. This is achieved by setting the dependency array to an empty array, which indicates that the effect should only be executed once, on component mount. All in all, This implementation allows users to search for recipes by entering keywords in the search field, with the results displayed in the UI. The loading state variable also provides a visual indication to the user that data is being fetched.

1.0.4 Displaying Workouts

This code is a snippet is an example of how each workout is rendered onto the screen of users.



```
<View style={styles.workoutStage}>
    <View style={styles.headingsView}>
        <Text style={styles.workoutHeading}>Barbell Situp</Text>
    </View>


    <Image
        source={{ uri: 'https://youtinebodypagevideo.s3.eu-west-1.amazonaws.com/abs3.gif' }}
        style={styles.videos}
        resizeMode="contain"
    />

    <View style={styles.listView}>
        <Text style={styles.listItem}><Text style={styles.number}>1.</Text> Use a handle attachment set all the way to the bottom of the machine.</Text>
        <Text style={styles.listItem}><Text style={styles.number}>2.</Text>Bring both of the handles to your chest and make sure you are in the center of the cable crossover.</
        Text>
        <Text style={styles.listItem}><Text style={styles.number}>3.</Text> Walk a few steps forward. Then press the weight forward. From there, your shoulders should
        horizontally abduct and adduct while your elbows stay in a fixed position.</Text>
    </View>
</View>
```



youtinebodypagevideo

Objects (12)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	abs1.gif	gif	April 19, 2023, 18:48:59 (UTC+01:00)	6.4 MB	Standard
□	abs2.gif	gif	April 19, 2023, 18:48:58 (UTC+01:00)	5.6 MB	Standard
□	abs3.gif	gif	April 19, 2023, 18:48:55 (UTC+01:00)	9.1 MB	Standard
□	Arms1.gif	gif	April 18, 2023, 16:12:35 (UTC+01:00)	6.4 MB	Standard
□	arms2.gif	gif	April 18, 2023, 16:43:38 (UTC+01:00)	4.7 MB	Standard
□	arms3.gif	gif	April 18, 2023, 16:43:36 (UTC+01:00)	5.1 MB	Standard
□	chest1.gif	gif	April 19, 2023, 18:04:55 (UTC+01:00)	7.8 MB	Standard
□	chest2.gif	gif	April 19, 2023, 18:04:50 (UTC+01:00)	4.5 MB	Standard
□	chest3.gif	gif	April 19, 2023, 18:04:48 (UTC+01:00)	4.8 MB	Standard
□	legs1.gif	gif	April 19, 2023, 20:01:22 (UTC+01:00)	4.8 MB	Standard
□	legs2.gif	gif	April 19, 2023, 20:01:20 (UTC+01:00)	10.9 MB	Standard
□	legs3.gif	gif	April 19, 2023, 20:01:24 (UTC+01:00)	6.3 MB	Standard

The video for this workout is deployed using Amazon Web Services (AWS) S3, with the URL specified in the source attribute of the Image component. The resizeMode property is set to "contain" to ensure that the entire video is visible within the component.

In addition to the video, there is a set of instructions listed below it, detailing the steps for performing the workout. The instructions are displayed in a View component with styling to create a list-like appearance, and each step is numbered using a custom Text component with its own styling. Overall, this code contributes to a larger project aimed at providing users with a variety of workout videos and instructions to help them achieve their fitness goals.

4.0 Testing

Testing is a crucial aspect of software development that ensures the quality and reliability of the product. In my project, I utilised React Native testing with Jest for unit, integration, and class testing. Jest is a popular testing framework that allows developers to write tests for JavaScript applications. With Jest, I was able to write tests that could simulate various scenarios and ensure that the application works as intended.

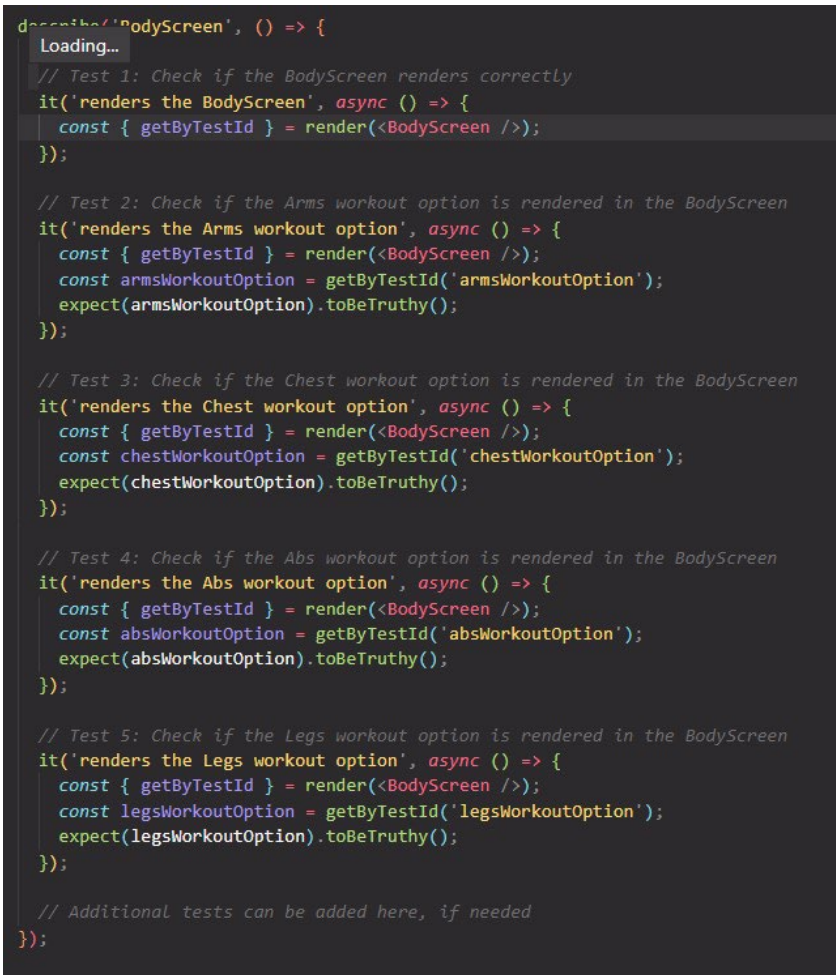
For unit testing, I wrote tests that check individual components of the application to ensure that they perform as expected. Integration tests, on the other hand, test the interactions between various components of the application. Lastly, class testing helped me to verify the functionality of individual classes and their methods. By using these testing methods, I was able to identify and fix issues early in the development process, resulting in a more stable and reliable application.

React Native Testing Library uses Jest as its test runner and assertion library. Jest provides a platform for defining and running unit, integration, and end-to-end tests, while also providing helpful utilities for mocking and stubbing dependencies.

Below I will explain the tests I ran followed by some code snippets .

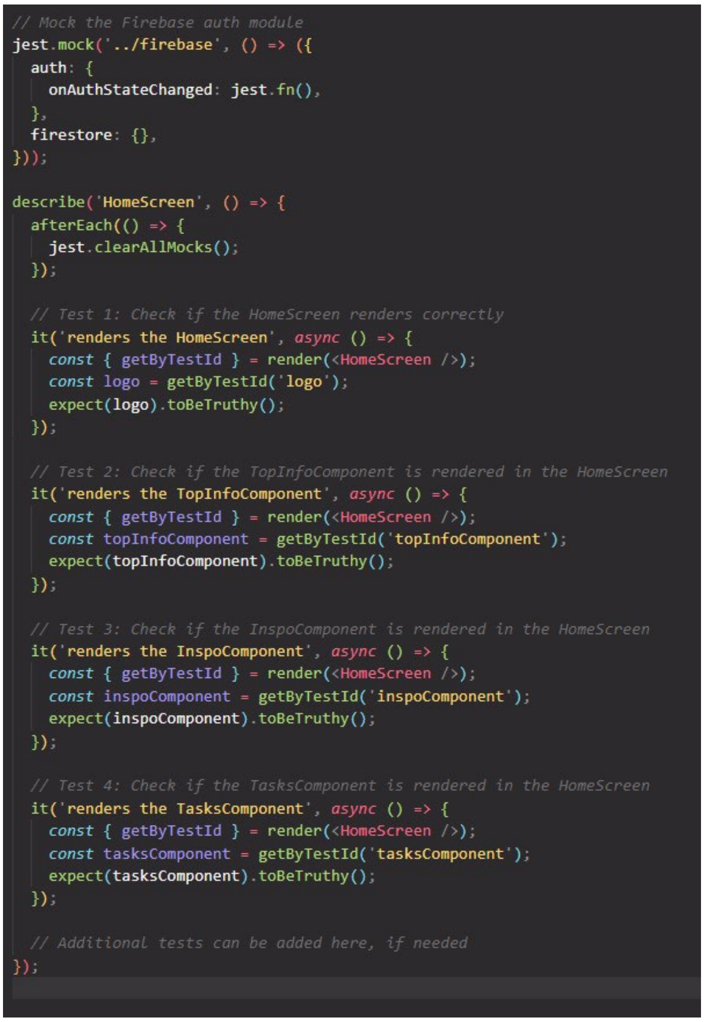
4.0.1 Unit Testing

This test file uses the Jest framework to test the BodyScreen component in Youtine. The first test simply checks if the BodyScreen component renders without any errors. The next four tests check if each workout option (Arms, Chest, Abs, and Legs) is correctly rendered in the BodyScreen component by using the getByTestId function to locate the corresponding testIDs. The tests expect the testIDs to be truthy if they exist in the rendered component. These tests ensure that the BodyScreen component is displaying all the workout options correctly and will help catch any potential rendering issues.



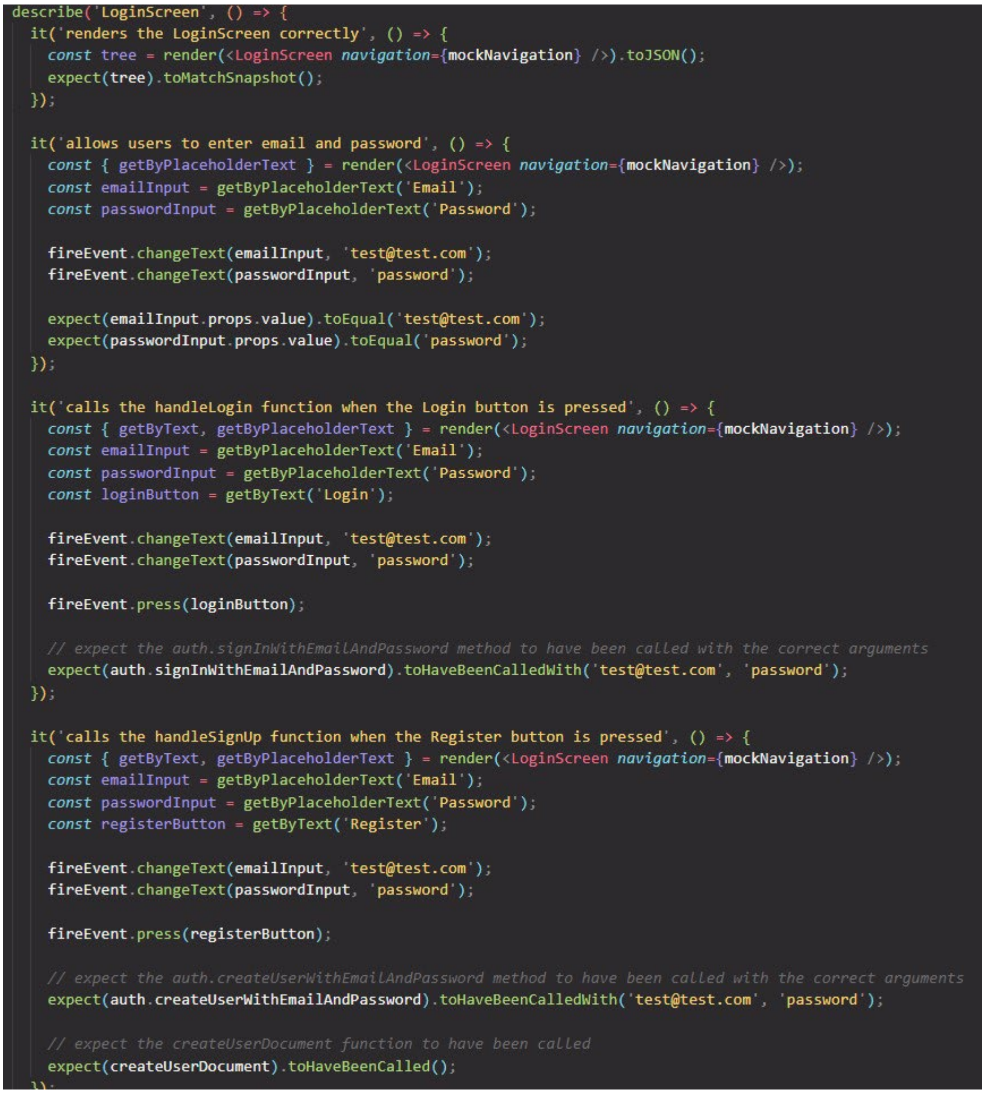
```
describe('BodyScreen', () => {

  // Test 1: Check if the BodyScreen renders correctly
  it('renders the BodyScreen', async () => {
    const { getByTestId } = render(<BodyScreen />);
  });

  // Test 2: Check if the Arms workout option is rendered in the BodyScreen
  it('renders the Arms workout option', async () => {
    const { getByTestId } = render(<BodyScreen />);
    const armsWorkoutOption = getByTestId('armsWorkoutOption');
    expect(armsWorkoutOption).toBeTruthy();
  });

  // Test 3: Check if the Chest workout option is rendered in the BodyScreen
  it('renders the Chest workout option', async () => {
    const { getByTestId } = render(<BodyScreen />);
    const chestWorkoutOption = getByTestId('chestWorkoutOption');
    expect(chestWorkoutOption).toBeTruthy();
  });

  // Test 4: Check if the Abs workout option is rendered in the BodyScreen
  it('renders the Abs workout option', async () => {
    const { getByTestId } = render(<BodyScreen />);
    const absWorkoutOption = getByTestId('absWorkoutOption');
    expect(absWorkoutOption).toBeTruthy();
  });

  // Test 5: Check if the Legs workout option is rendered in the BodyScreen
  it('renders the Legs workout option', async () => {
    const { getByTestId } = render(<BodyScreen />);
    const legsWorkoutOption = getByTestId('legsWorkoutOption');
    expect(legsWorkoutOption).toBeTruthy();
  });

  // Additional tests can be added here, if needed
});
```

The code below tests the HomeScreen component in Youtine. It uses the render method from the @testing-library/react-native library to create an instance of the HomeScreen component and perform assertions on the rendered output. The tests check if the logo, top info component, inspo component, and tasks component are all rendered correctly. The auth module from the Firebase SDK is also mocked using the jest.mock method to test the component in isolation. The afterEach method is used to clear all mocked functions after each test, to ensure a clean slate for the next test.



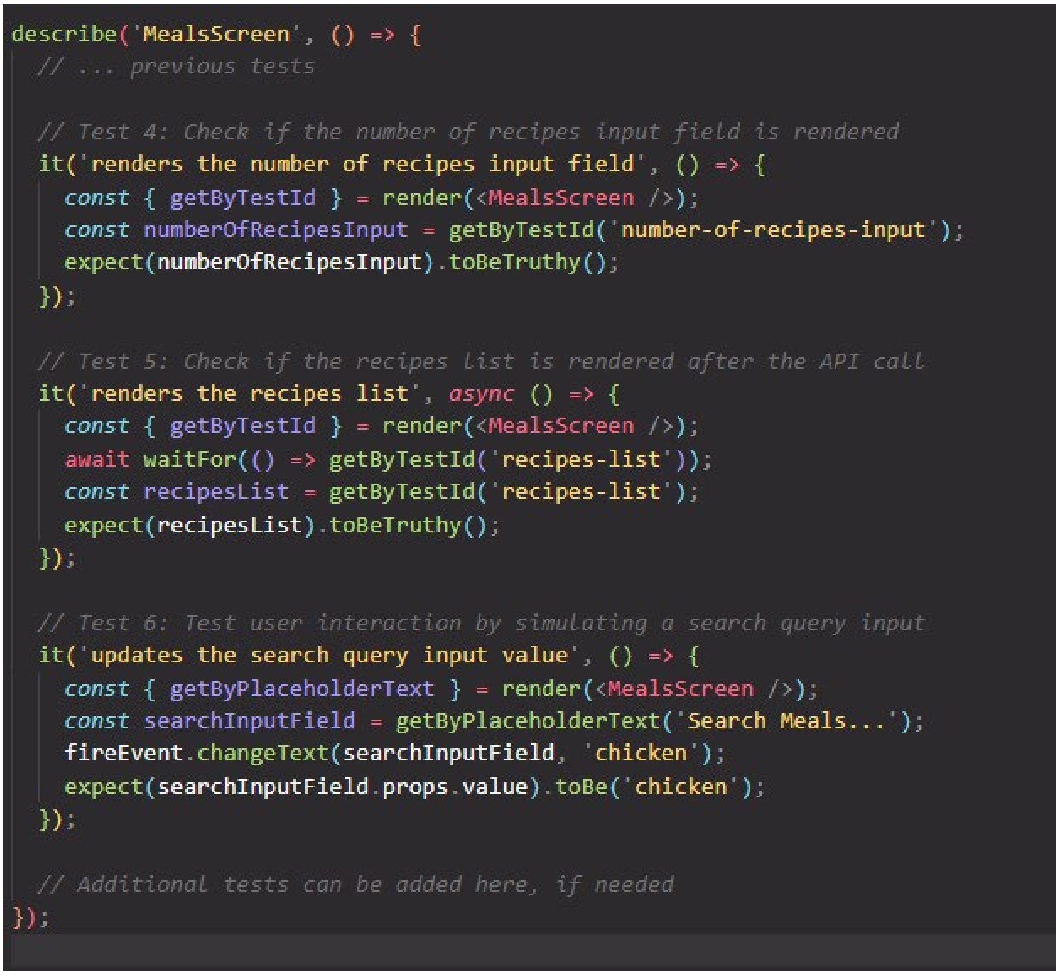
```
// Mock the Firebase auth module
jest.mock('../firebase', () => ({
  auth: {
    onAuthStateChanged: jest.fn(),
  },
  firestore: {},
}));

describe('HomeScreen', () => {
  afterEach(() => {
    jest.clearAllMocks();
  });

  // Test 1: Check if the HomeScreen renders correctly
  it('renders the HomeScreen', async () => {
    const { getByTestId } = render(<HomeScreen />);
    const logo = getByTestId('logo');
    expect(logo).toBeTruthy();
  });

  // Test 2: Check if the TopInfoComponent is rendered in the HomeScreen
  it('renders the TopInfoComponent', async () => {
    const { getByTestId } = render(<HomeScreen />);
    const topInfoComponent = getByTestId('topInfoComponent');
    expect(topInfoComponent).toBeTruthy();
  });

  // Test 3: Check if the InspoComponent is rendered in the HomeScreen
  it('renders the InspoComponent', async () => {
    const { getByTestId } = render(<HomeScreen />);
    const inspoComponent = getByTestId('inspoComponent');
    expect(inspoComponent).toBeTruthy();
  });

  // Test 4: Check if the TasksComponent is rendered in the HomeScreen
  it('renders the TasksComponent', async () => {
    const { getByTestId } = render(<HomeScreen />);
    const tasksComponent = getByTestId('tasksComponent');
    expect(tasksComponent).toBeTruthy();
  });

  // Additional tests can be added here, if needed
});
```

The following tests check whether the LoginScreen component renders correctly, allows users to enter email and password, and triggers the handleLogin and handleSignUp functions when the Login and Register buttons are pressed respectively. The mockNavigation object is used to simulate navigation to other screens. Additionally, the mock Firebase auth methods are used to simulate user authentication. Overall, the tests ensure the LoginScreen component functions as expected, and user input is processed correctly.



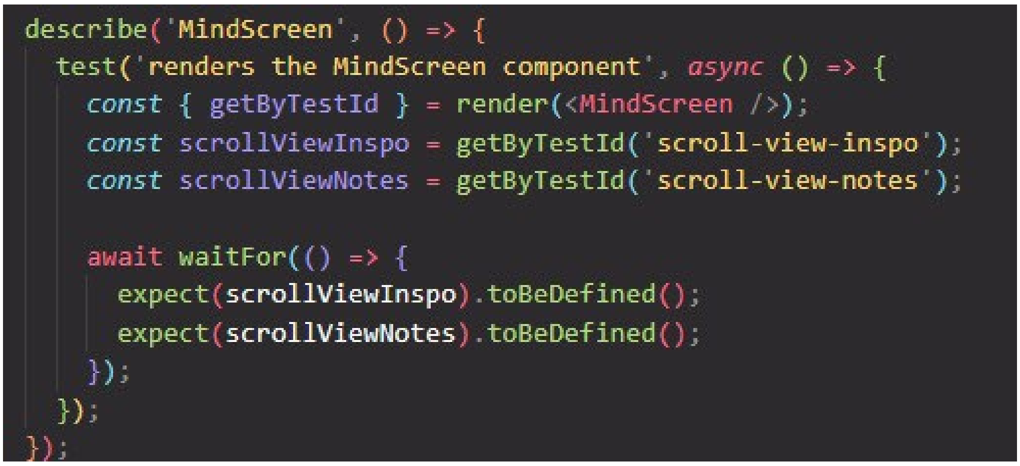
```
describe('LoginScreen', () => {
  it('renders the LoginScreen correctly', () => {
    const tree = render(<LoginScreen navigation={mockNavigation} />).toJSON();
    expect(tree).toMatchSnapshot();
  });

  it('allows users to enter email and password', () => {
    const { getByPlaceholderText } = render(<LoginScreen navigation={mockNavigation} />);
    const emailInput = getByPlaceholderText('Email');
    const passwordInput = getByPlaceholderText('Password');

    fireEvent.changeText(emailInput, 'test@test.com');
    fireEvent.changeText(passwordInput, 'password');

    expect(emailInput.props.value).toEqual('test@test.com');
    expect(passwordInput.props.value).toEqual('password');
  });

  it('calls the handleLogin function when the Login button is pressed', () => {
    const { getByText, getByPlaceholderText } = render(<LoginScreen navigation={mockNavigation} />);
    const emailInput = getByPlaceholderText('Email');
    const passwordInput = getByPlaceholderText('Password');
    const loginButton = getByText('Login');

    fireEvent.changeText(emailInput, 'test@test.com');
    fireEvent.changeText(passwordInput, 'password');

    fireEvent.press(loginButton);

    // expect the auth.signInWithEmailAndPassword method to have been called with the correct arguments
    expect(auth.signInWithEmailAndPassword).toHaveBeenCalledWith('test@test.com', 'password');
  });

  it('calls the handleSignUp function when the Register button is pressed', () => {
    const { getByText, getByPlaceholderText } = render(<LoginScreen navigation={mockNavigation} />);
    const emailInput = getByPlaceholderText('Email');
    const passwordInput = getByPlaceholderText('Password');
    const registerButton = getByText('Register');

    fireEvent.changeText(emailInput, 'test@test.com');
    fireEvent.changeText(passwordInput, 'password');

    fireEvent.press(registerButton);

    // expect the auth.createUserWithEmailAndPassword method to have been called with the correct arguments
    expect(auth.createUserWithEmailAndPassword).toHaveBeenCalledWith('test@test.com', 'password');

    // expect the createUserDocument function to have been called
    expect(createUserDocument).toHaveBeenCalled();
  });
```

In this Jest test file, the MealsScreen component is being tested using various tests. The tests ensure that the component renders correctly, that the filter feature updates the search query input value, that the number of recipes input field is rendered, and that the recipes list is rendered after an API call. The render method from @testing-library/react-native is used to render the MealsScreen component, and the fireEvent method is used to simulate user interaction with the search input field. Additionally, the waitFor method is used to wait for the recipes list to render after an API call. Overall, these tests ensure that the MealsScreen component functions correctly and provides a good user experience.



```
describe('MealsScreen', () => {
  // ... previous tests

  // Test 4: Check if the number of recipes input field is rendered
  it('renders the number of recipes input field', () => {
    const { getByTestId } = render(<MealsScreen />);
    const numberOfRecipesInput = getByTestId('number-of-recipes-input');
    expect(numberOfRecipesInput).toBeTruthy();
  });

  // Test 5: Check if the recipes list is rendered after the API call
  it('renders the recipes list', async () => {
    const { getByTestId } = render(<MealsScreen />);
    await waitFor(() => getByTestId('recipes-list'));
    const recipesList = getByTestId('recipes-list');
    expect(recipesList).toBeTruthy();
  });

  // Test 6: Test user interaction by simulating a search query input
  it('updates the search query input value', () => {
    const { getByPlaceholderText } = render(<MealsScreen />);
    const searchInputField = getByPlaceholderText('Search Meals...');
    fireEvent.changeText(searchInputField, 'chicken');
    expect(searchInputField.props.value).toBe('chicken');
  });

  // Additional tests can be added here, if needed
});
```

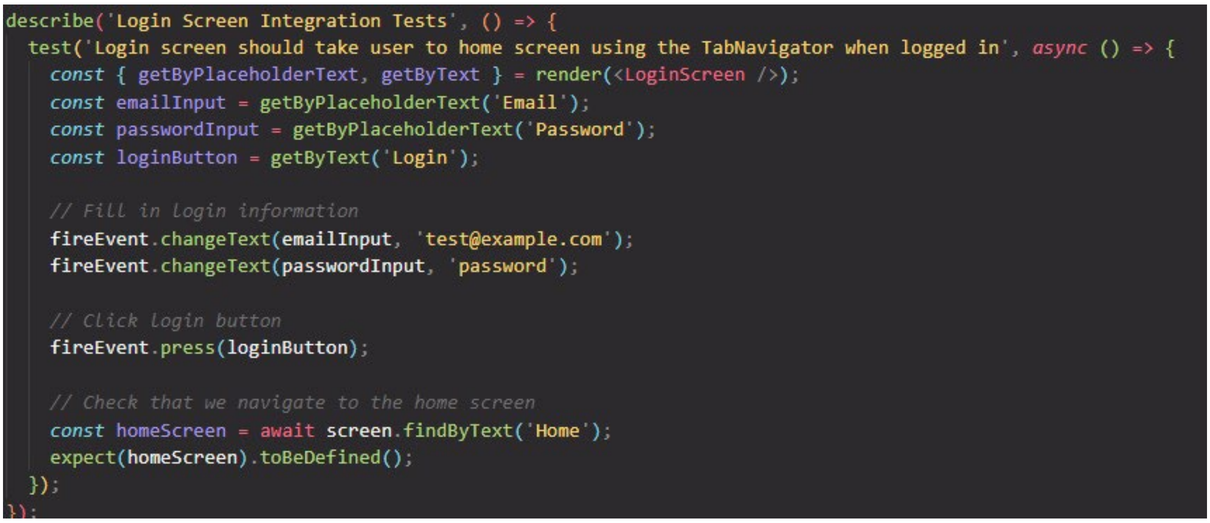
In this test, the Jest testing library is used to check whether the MindScreen component is rendered correctly. The render function is called to render the component and the getByTestId function is used to get the scrollViewInspo and scrollViewNotes elements by their test IDs. The waitFor function is used to wait for the elements to be defined before making assertions. Finally, expect statements are used to check if the elements are defined, which confirms that the component is rendered correctly.



```
describe('MindScreen', () => {
  test('renders the MindScreen component', async () => {
    const { getByTestId } = render(<MindScreen />);
    const scrollViewInspo = getByTestId('scroll-view-inspo');
    const scrollViewNotes = getByTestId('scroll-view-notes');

    await waitFor(() => {
      expect(scrollViewInspo).toBeDefined();
      expect(scrollViewNotes).toBeDefined();
    });
  });
});
```

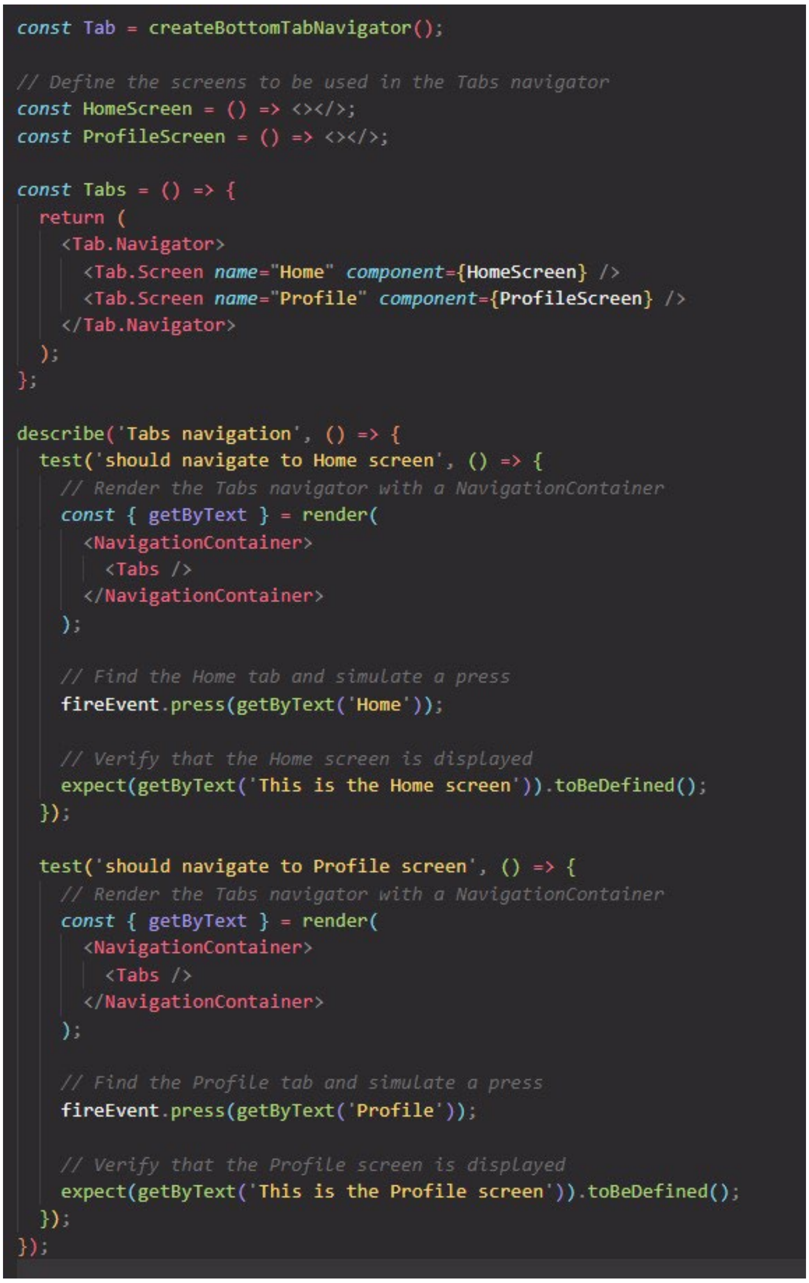
4.0.2 Integration Testing

In this test, an integration test is being performed on the LoginScreen component to verify that the user is redirected to the home screen through the TabNavigator when they log in. The render method from @testing-library/react-native library is used to render the LoginScreen component. The test then proceeds to simulate user interaction by filling in login information with the fireEvent.changeText method and then clicking on the login button with fireEvent.press. Finally, the screen.findByText method is used to find the 'Home' text on the home screen to confirm that the user was redirected successfully.



```
describe('Login Screen Integration Tests', () => {
  test('Login screen should take user to home screen using the TabNavigator when logged in', async () => {
    const { getByPlaceholderText, getByText } = render(<LoginScreen />);
    const emailInput = getByPlaceholderText('Email');
    const passwordInput = getByPlaceholderText('Password');
    const loginButton = getByText('Login');

    // Fill in login information
    fireEvent.changeText(emailInput, 'test@example.com');
    fireEvent.changeText(passwordInput, 'password');

    // Click login button
    fireEvent.press(loginButton);

    // Check that we navigate to the home screen
    const homeScreen = await screen.findByText('Home');
    expect(homeScreen).toBeDefined();
  });
});
```

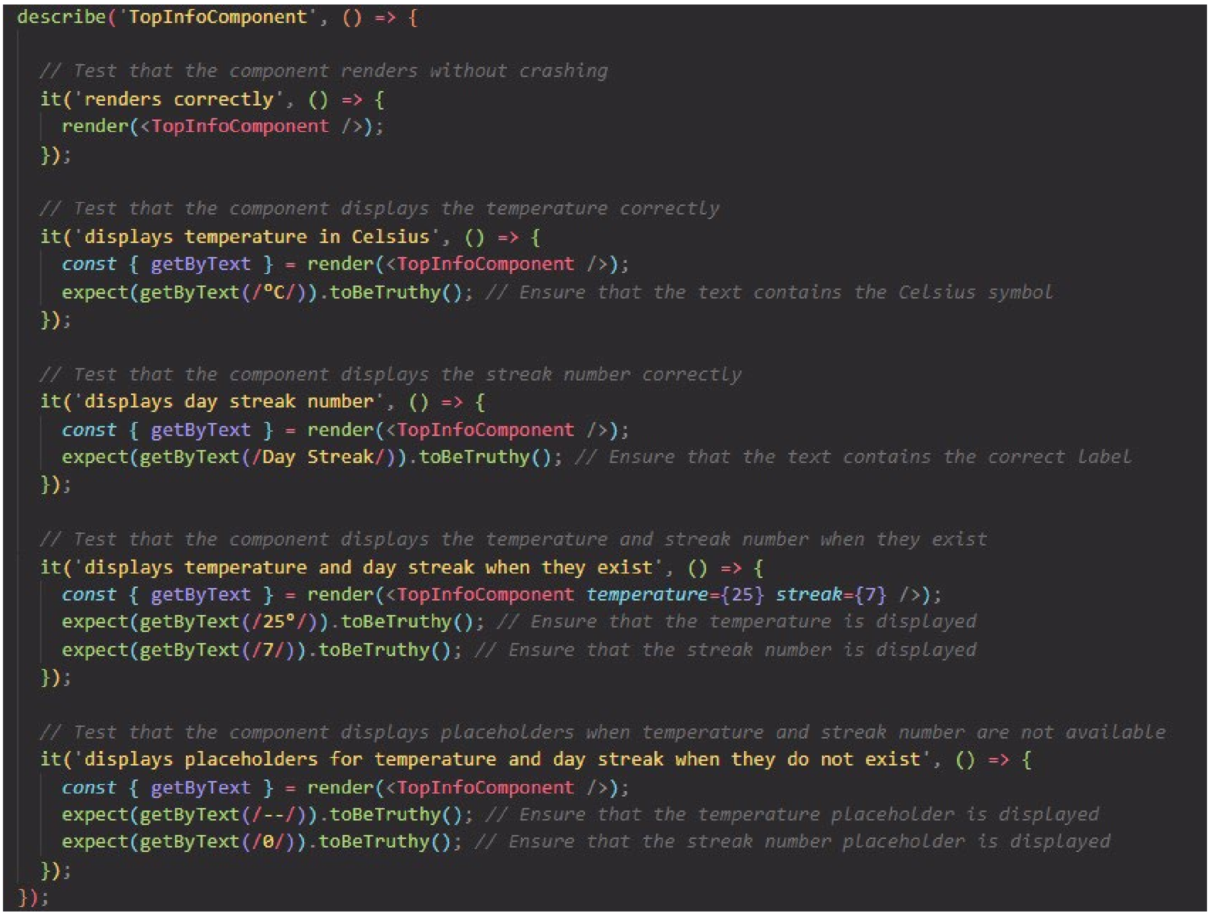
In this code snippet, an integration test suite is defined to test the functionality of the Tabs navigator in the app using the @react-navigation library. The createBottomTabNavigator function is used to create the Tabs navigator, and two screens, HomeScreen and ProfileScreen, are defined as components to be used in the navigator. The two tests in the suite simulate pressing the Home and Profile tabs and then verify that the corresponding screens are displayed. The tests use the render and fireEvent functions from the @testing-library/react-native library to interact with the UI components and check their state. The NavigationContainer component from @react-navigation/native is used to wrap the Tabs navigator, which provides a context for the navigator to function correctly.



```
const Tab = createBottomTabNavigator();

// Define the screens to be used in the Tabs navigator
const HomeScreen = () => <></>;
const ProfileScreen = () => <></>;

const Tabs = () => {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Profile" component={ProfileScreen} />
    </Tab.Navigator>
  );
};

describe('Tabs navigation', () => {
  test('should navigate to Home screen', () => {
    // Render the Tabs navigator with a NavigationContainer
    const { getByText } = render(
      <NavigationContainer>
        <Tabs />
      </NavigationContainer>
    );

    // Find the Home tab and simulate a press
    fireEvent.press(getByText('Home'));

    // Verify that the Home screen is displayed
    expect(getByText('This is the Home screen')).toBeDefined();
  });

  test('should navigate to Profile screen', () => {
    // Render the Tabs navigator with a NavigationContainer
    const { getByText } = render(
      <NavigationContainer>
        <Tabs />
      </NavigationContainer>
    );

    // Find the Profile tab and simulate a press
    fireEvent.press(getByText('Profile'));

    // Verify that the Profile screen is displayed
    expect(getByText('This is the Profile screen')).toBeDefined();
  });
});
```

4.0.3 Class Testing

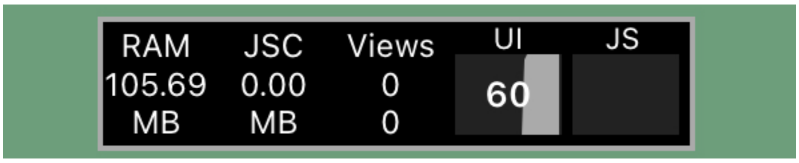
The TopInfoComponent is a React component that displays information related to temperature and streak number. The test suite for this component includes four tests to ensure that the component is rendering correctly and displaying the correct information. The first test checks that the component renders without crashing, the second test ensures that the component displays the temperature in Celsius, the third test ensures that the component displays the day streak number, and the fourth test checks that the component displays the temperature and streak number when they exist and displays placeholders when they do not exist. The tests use the render function from the @testing-library/react-native library to render the component and then use the getByText function to check that the correct information is displayed.



```
describe('TopInfoComponent', () => {

  // Test that the component renders without crashing
  it('renders correctly', () => {
    render(<TopInfoComponent />);
  });

  // Test that the component displays the temperature correctly
  it('displays temperature in Celsius', () => {
    const { getByText } = render(<TopInfoComponent />);
    expect(getByText(/°C/)).toBeTruthy(); // Ensure that the text contains the Celsius symbol
  });

  // Test that the component displays the streak number correctly
  it('displays day streak number', () => {
    const { getByText } = render(<TopInfoComponent />);
    expect(getByText(/Day Streak/)).toBeTruthy(); // Ensure that the text contains the correct label
  });

  // Test that the component displays the temperature and streak number when they exist
  it('displays temperature and day streak when they exist', () => {
    const { getByText } = render(<TopInfoComponent temperature={25} streak={7} />);
    expect(getByText(/25°/)).toBeTruthy(); // Ensure that the temperature is displayed
    expect(getByText(/7/)).toBeTruthy(); // Ensure that the streak number is displayed
  });

  // Test that the component displays placeholders when temperature and streak number are not available
  it('displays placeholders for temperature and day streak when they do not exist', () => {
    const { getByText } = render(<TopInfoComponent />);
    expect(getByText(/--/)).toBeTruthy(); // Ensure that the temperature placeholder is displayed
    expect(getByText(/0/)).toBeTruthy(); // Ensure that the streak number placeholder is displayed
  });
});
```

4.0.4 Performance

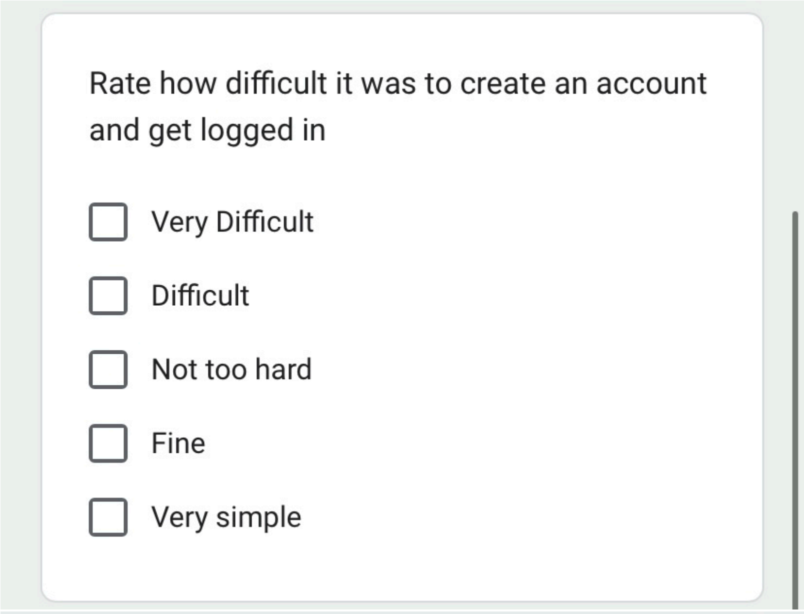
Using Expo's performance monitor I was able to observe my applications' RAM usage at app launch was 105.69. A RAM usage of 105.69 at app launch is considered good compared to other mobile apps. According to a report by PerfTestPlus, the average RAM usage of popular Android apps is around 200 MB, with some apps using up to 500 MB or more. On iOS, the average memory usage for popular apps is around 400 MB, with some apps using up to 1 GB or more (Performance Testing Software Systems, 2012). Therefore, with a RAM usage of 105.69 at app launch, the app is using significantly less memory than the average app and is performing well in terms of memory management.



```
  RAM      JSC    Views    UI      JS
105.69    0.00      0      60
  MB       MB       0
```

4.0.5 Usability Testing

I had members of my family take part in a usability questionnaire where they were told to rate some aspects of the user experience. The caveat was they had no prior experience using the app but were prompted to navigate to 'create an account and get logged in', 'find a workout' and 'navigate to the meals screen and search for a recipe'. After having done so the questions were answered. The choices they had to choose from ranged from 'Very Difficult', 'Difficult', 'Not too hard', 'Fine' and 'Very simple'.



Rate how difficult it was to create an account and get logged in

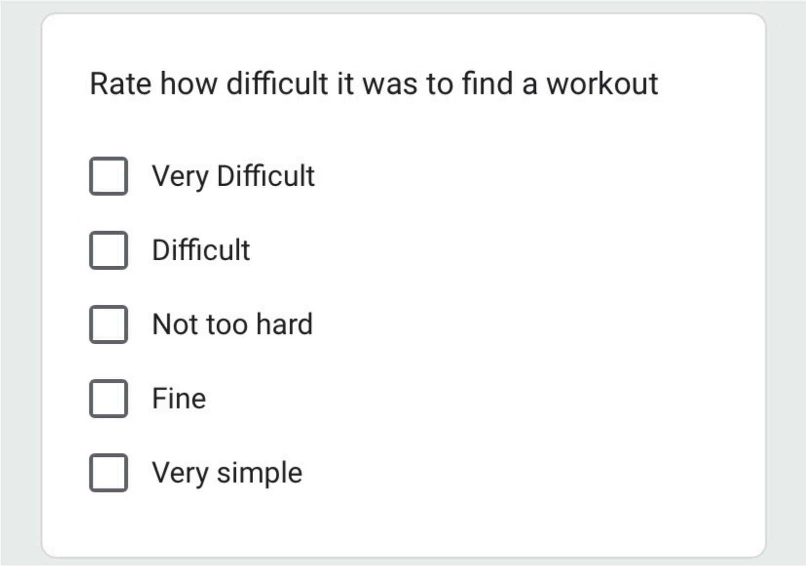
□ Very Difficult

□ Difficult

□ Not too hard

□ Fine

□ Very simple



Rate how difficult it was to find a workout

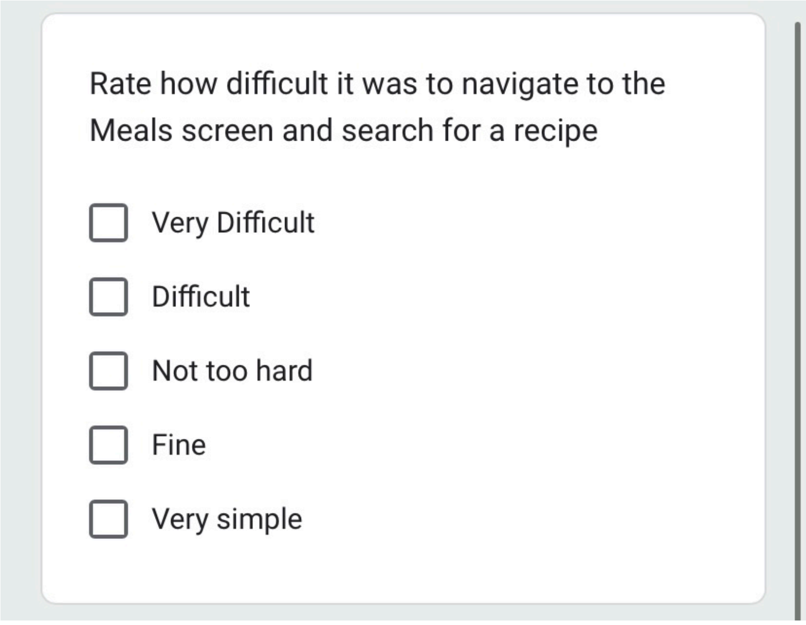
□ Very Difficult

□ Difficult

□ Not too hard

□ Fine

□ Very simple



Rate how difficult it was to navigate to the Meals screen and search for a recipe

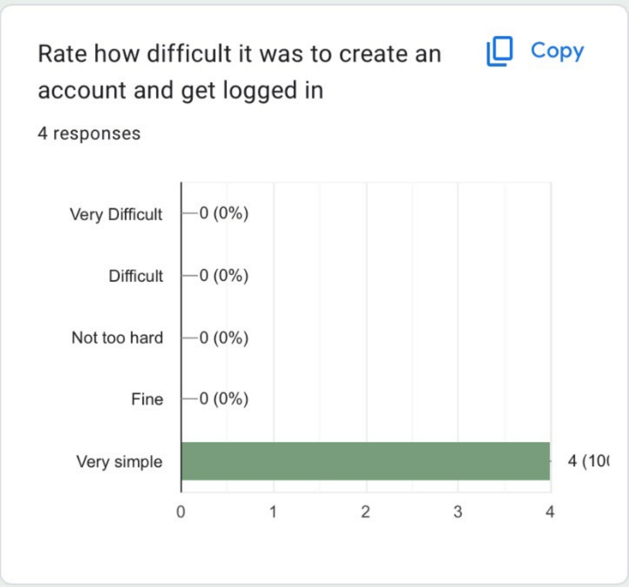
□ Very Difficult

□ Difficult

□ Not too hard

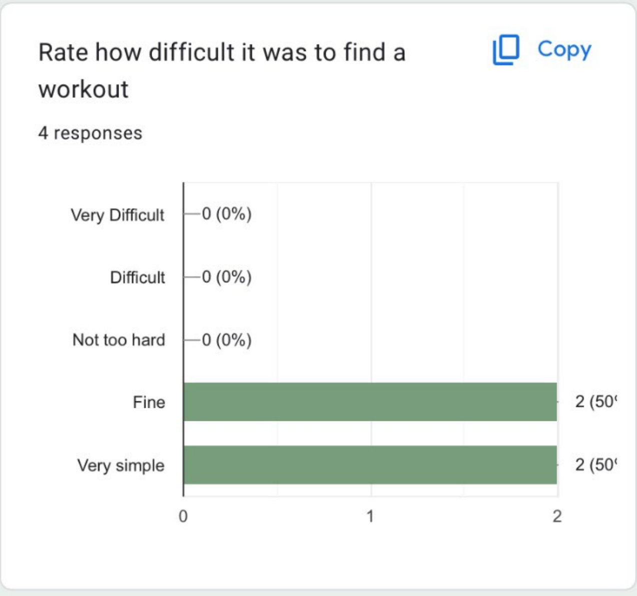
□ Fine

□ Very simple



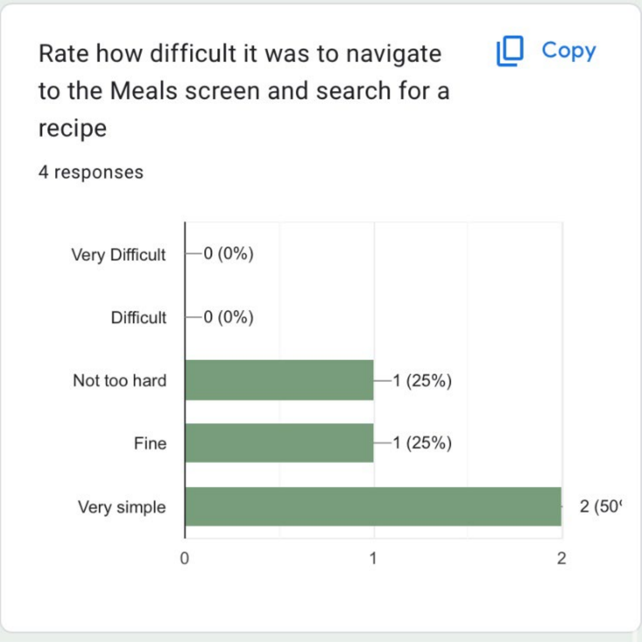
Rate how difficult it was to create an account and get logged in

4 responses



Rate how difficult it was to find a workout

4 responses



Rate how difficult it was to navigate to the Meals screen and search for a recipe

4 responses

5.0 Conclusions

After months of dedication and hard work, it is evident that Youtine has the potential to make a significant impact in the self-improvement app market. Utilising the power of React Native has enabled the creation of a user-friendly and aesthetically pleasing interface, while Firebase has contributed to the implementation of a robust and secure backend. Along the development journey, I have faced multiple challenges that have tested my problem-solving skills and forced me to further refine my programming expertise.

One of the most significant challenges has been keeping up with the ever-changing world of React Native modules. Despite the benefits of these modules, they often require updates and can cause conflicts with other dependencies. However, with persistence and dedication, I have successfully integrated various modules and components to create the experience that is Youtine.

Nonetheless, what truly sets Youtine apart is its potential to positively impact the lives of its users. As someone with a passion for healthy wellbeing and mental health, it has been a fulfilling experience to develop an app that can help individuals attain their desired goals.

In conclusion, developing Youtine has been an arduous yet rewarding journey. A journey that is far from over, as there is always room for refinement and further development. Nevertheless, I am enthusiastic about the future of Youtine and the positive impact it will have on its users.

6.0 Further Development or Research

After working on the Youtine project, I believe that it has the potential to make a positive impact in the fitness app market. However, like any software project, there is always room for improvement and further development.

If given more resources and time, I would focus on enhancing the app's user experience and incorporating more personalised features. This could include features such as a more robust meal planner or customised workout plans based on the user's fitness level and goals. Additionally, I would explore ways to incorporate community-building features, such as discussion forums or support groups, to foster a sense of community among Youtine users. Expanding support to android devices is also an aspiration as I believe as many people as possible should have access to this app.

7.0 References

aetna, 2023. *Food for your mood: How what you eat affects your mental health.* [Online]
Available at: https://www.aetna.com/health-guide/food-affects-mental-health.html#:~:text=Studies%20have%20even%20found%20that,risk%20of%20dementia%20or%20stroke.
[Accessed 4 May 2023].

Better Health Channel, 2021. *Exercise and mental health.* [Online]
Available at: https://www.betterhealth.vic.gov.au/health/healthyliving/exercise-and-mental-health#:~:text=The%20levels%20of%20chemicals%20in,coping%20ability%20and%20self%2Desteem.
[Accessed 22 April 2023].

Fay W. Boozman, P. A. M., 2022. *Associations among Self-Reported Mental Health, Physical Activity, and Diet during the COVID-19 Pandemic.* [Online]
Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8891903/
[Accessed 4 May 2023].

HSE, 2015. *Key Facts.* [Online]
Available at: https://www.hse.ie/eng/about/who/healthwellbeing/our-priority-programmes/heal/key-facts/
[Accessed 5 March 2023].

Mental Health Ireland, 2015. *Research.* [Online]
Available at: https://www.mentalhealthireland.ie/research/
[Accessed 4 December 2022].

Motswatswa, K., 2022. *Rigid concepts of masculinity are pushing Black men to suicide.* [Online]
Available at: https://newafricanmagazine.com/28162/
[Accessed 28 September 2022].

Performance Testing Software Systems, 2012. *Core Performance Testing Principle.* [Online]
Available at: http://www.perftestplus.com/core_criteria.htm
[Accessed 4 May 2023].

Roe, C., 2021. *MENTAL HEALTH IN IRELAND DURING COVID-19.* [Online]
Available at: https://borgenproject.org/mental-health-in-ireland/#:~:text=The%20paper%20reported%20that%20mental,billion%20in%20the%20United%20States.
[Accessed 9 April 2023].

University of Rochester Medical Center, n.d. *Journaling for Mental Health.* [Online]
Available at:
https://www.urmc.rochester.edu/encyclopedia/content.aspx?ContentID=4552&ContentTypeID=1#:~:text=It's%20simply%20writing%20down%20your,and%20improve%20your%20mental%20health.
[Accessed 12 April 2023].

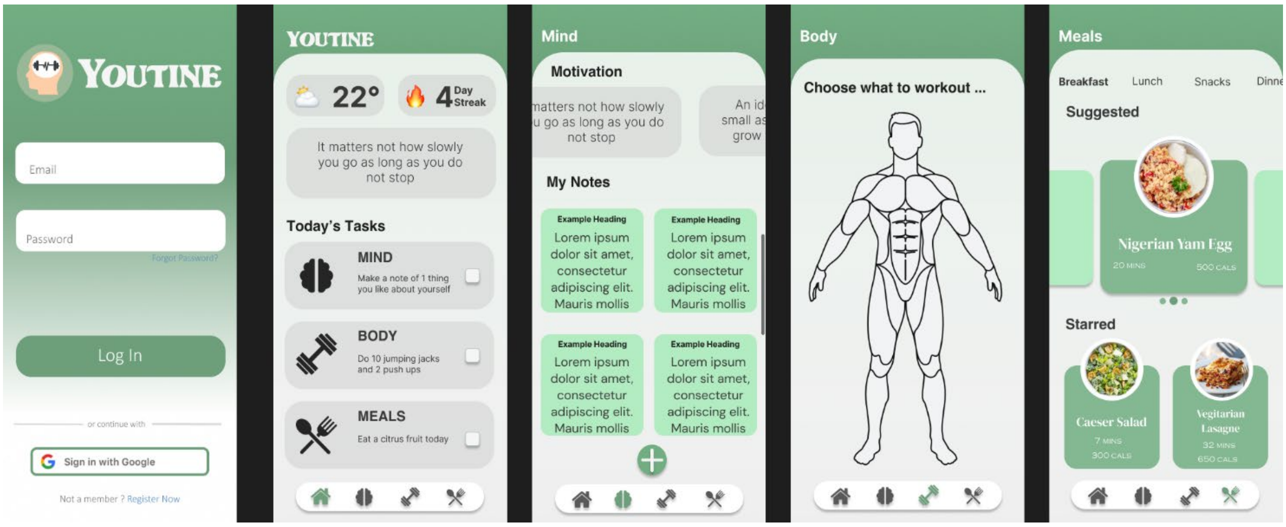
World Health Organization, 2001. *The World Health Report 2001: Mental Disorders affect one in four people.* [Online]

Available at: https://www.who.int/news/item/28-09-2001-the-world-health-report-2001-mental-disorders-affect-one-in-four-people#:~:text=One%20in%20four%20people%20in,ill%2Dhealth%20and%20disability%20worldwide.
[Accessed 28 January 2023].

7.1. Project Proposal

8.0 Objectives

The target audience of this project is primarily individuals who are struggling with their mental health or individuals who have found their mental health to be depleting. The app is also useful for all people who value self-improvement and discipline. Youtine intends on improving the mental health of its users by having them login to the app daily to complete 3 challenges. One challenge for the mind, another for the body and the last is a meal challenge. This app is intended to be supported by iOS.

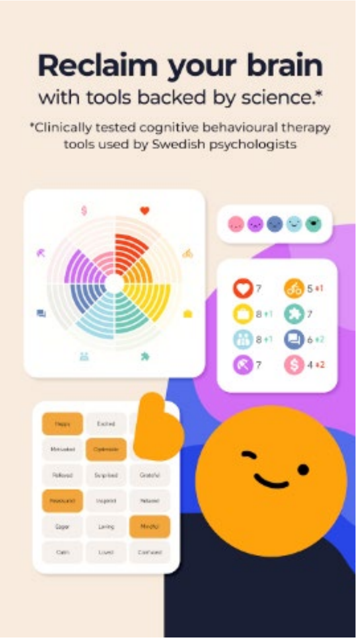


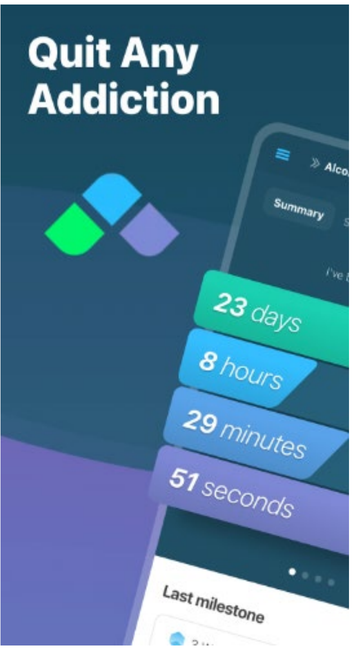
9.0 Background

I chose to undertake this project as recently I noticed many individuals my age are currently suffering with mental disorders such as anxiety and depression. I found it quite unnerving how normalised and common it had become with my generation, so I've decided to do my part in tackling the issue. I began by looking into the factors which can cause or exacerbate a depleting mental state and I found there were some overlapping commonalities. These factors are as follows : Poor routine, a lack of self-esteem, a lack of exercise/adequate physical stimulation & a poor diet. I investigated how I could tackle these 4 issues with just the 1 solution and that is how I came up with Youtine. I made sure to take into consideration the necessity for the daily tasks to be simple enough to do but effective enough to make a difference in one's life. The app will have 4 primary pages, the first being the home page and the remaining 3 being the mind, body & meals page which exist to facilitate the completion of the challenges.

10.0 State of the Art

Mobile applications I found that have a similar ethos would be the likes of ‘Remente’ and ‘I Am Sober’. Remente is a self-care app that prioritises personal growth by scientifically backed goals. I Am Sober is a sobriety counter app which allows users to input whichever addiction they have, and the app will count how long they have gone whilst facilitating a daily check in to confirm the user has not relapsed.

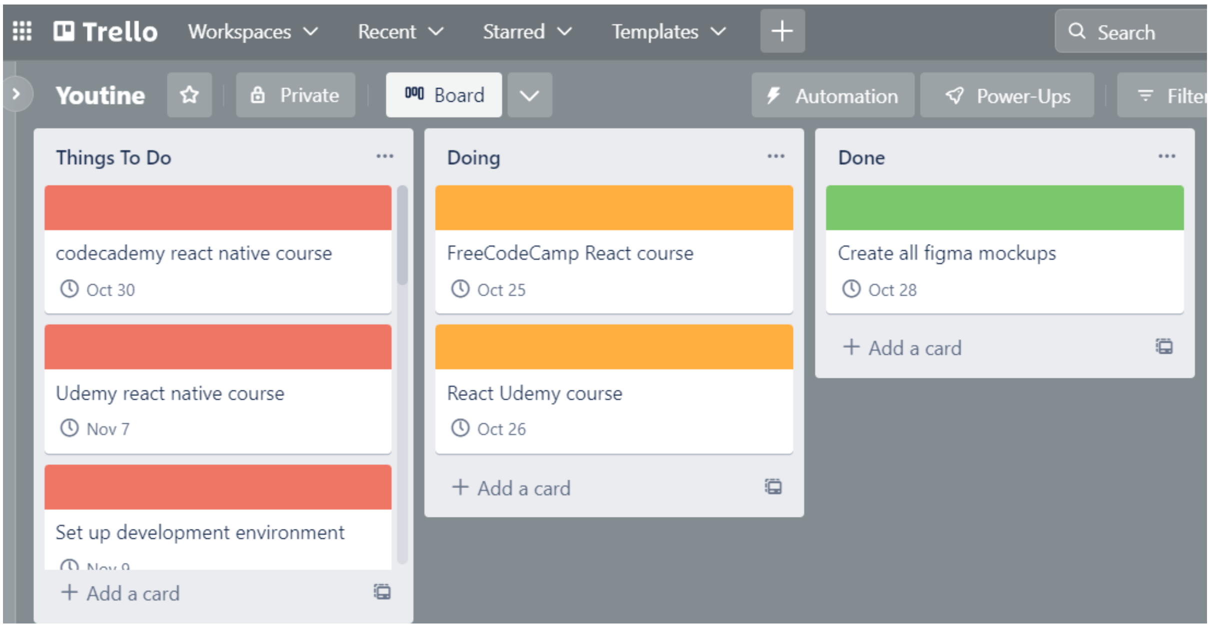

Reclaim your brain
with tools backed by science.*
*Clinically tested cognitive behavioural therapy tools used by Swedish psychologists



My application differs largely from Remente in our approaches to how we want to achieve our goals. Remente goes down the root of providing many tools and trackers for the user to play with. My criticism with this approach would be that users are overwhelmed by all the options and eventually find it easier to ignore using the app at all. Therefore, my own is concise and simple to operate. I Am Sober brings the streaks feature present in my own app to the fore front as their main feature, prioritising the days hours and minutes more than the individual themselves in my opinion. My project stands out by prioritising the users physical and mental health through exercise, diet, motivation, routine, and affirmational note taking all in the one application.

11.0 Technical Approach

I intend to take the agile approach to developing this application. I find this approach results in a smoother development experience with less stress. I will be breaking down each aspect of my project and working towards building them piece by piece. I will identify requirements by designing wireframes of how I desire the app to be structured, then based off those wireframes I will have a better understanding of the space I have and how much each feature needs. After having done this, I will design mock ups which are true to how I want the actual app to look, this way I can see each aspect of the app and I will break down and note each feature in my notes. Based on the feature notes I have made, I then turn them into individual tasks using Trello. Trello allows me to manage what I must do and when I want to have it done.



Things To Do	Doing	Done
codecademy react native course Oct 30	FreeCodeCamp React course Oct 25	Create all figma mockups Oct 28
Udemy react native course Nov 7	React Udemy course Oct 26	
Set up development environment Nov 9		

12.0 Technical Details

I intend to build this application using react native, a Javascript framework which allows developers to creative native Android & iOS mobile applications. I decided on using react native for this reason. Within React native I will write structural code using JSX, a syntax extension to Javascript that largely resembles HTML. It is also possible to use CSS modules to style applications using react native but the primary method of styling I intend to use is the Javascript Style prop. I intend to employ the use of Firebase to service the backend functionality of my application. Firebase authentication will allow me to develop a system which allows users to create user profiles and to log in using a third-party application such as Gmail. The use of the stack data structure will be used for navigation throughout aspects of the app. My integrated development environment (IDE) of choice is visual studio code as I am familiar with the layout and like the extensions it provides.

13.0 Special Resources Required

For this project to work as intended, I will need to employ the use of APIs. A weather API will be needed for the weather widget I have planned to be on the home screen. I will also need an API that will provide me with a database of motivational quotes and affirmation for a widget on the home screen. The meals screen will also draw much of its data from an API.

14.0 Project Plan

I plan to begin my project by taking out time to teach myself the technologies I will be using for this project. I will be using October to go through the FreeCodeCamp & Udemy React.js course and codecademy react native course. I intend to have a fundamental knowledge to confidently allow me to at least begin work by October 30th.

After having finished these courses, I will get started on converting the wireframes I drew up in September into realistic mock-ups using Figma. I will make a mock up of every aspect of the app before building the page. I find having a mock up to copy makes building the pages quite a bit faster and smoother. This should all be done by November 7th.

To start work on my project I will need to set up my development environment. I intend to use Expo via VS code. Expo is an open-source platform intended for making native web, android & iOS apps with React. Expo allows me to use my personal phone to test and see my app as it is being built instead of just an emulator. This setup process can be long, but I should have it ready by November 9th .

The first thing to do for me will be to create the four main pages. The home, mind, body, and Meals pages. Following this I will create a bottom navigation bar and place it on each page. The home page will be designed first, I will then add functionality to the bottom navigation bar . The login screen will be next to be constructed with functionality being implemented using Firebase.

15.0 Testing

The default template of React Native ships with Jest testing framework. Jest allows me to write all sorts of tests . Structuring tests test code by being given a precondition, analysing the action executed by the function being tested then comparing the outcome with the expected outcome. Unit testing, mocking testing , integration testing and component testing is all also available.

16.0 Reflective Journals

15.1 October Reflection

This month I managed to confirm my project idea. I then drew up wireframes which I later converted into proper mock ups using Figma. Now that the mock-ups were finished, I began setting up my laptop to be able to develop using react native. Once the setting up of my development environment was completed, I began designing the frontend aspects of my applications home page , using the mock-up as a guide. This is where I am at now.

15.2 November Reflection

In November, I focused on developing the login and registration functionality using Firebase, along with implementing the daily tasks feature. These were both integral parts of the app and required significant attention to detail. I also spent time improving the app's UI by incorporating icons and refining the overall layout. One of the most challenging aspects of this month was debugging and resolving issues related to state management. Despite the difficulties, it was a fulfilling experience to see the app come to life and start to take shape.

15.3 December Reflection

In December, I continued to fine-tune the app's UI, making it more visually appealing and user-friendly. I also incorporated a feature that allows users to take and view notes in the app, promoting mindfulness and reflection. Testing and debugging were significant focuses this month, and I spent a lot of time fixing bugs and ensuring that the app was running smoothly. It was an excellent opportunity to hone my problem-solving skills and learn more about the development process.

15.4 January Reflection

In January, I worked on implementing the API calls to display weather and inspirational quotes on the home screen, as well as integrating the AWS S3 service for storing videos. Another major focus was improving the app's overall performance and user experience, such as reducing loading times and improving the smoothness of transitions between screens. I felt a sense of accomplishment as I good now see the app coming together as I first Imagined.

15.5 February Reflection

In February, I concentrated on refining the app's workout feature, making it so it is arranged into different body sections and customised so that the workouts are beginner friendly and achievable at home. I also spent time testing and debugging the app, ensuring that all features were working correctly and efficiently. Additionally, I started taking a hard look at the technical report requirements, which helped me to reflect on my progress and evaluate the app's potential impact.

15.6 March Reflection

In March, I shifted my focus to polishing and fine-tuning the app's features, such as improving the search functionality for the meals section and refining the note-taking feature. I worked quite a bit on the profile screen also. As the deadline for submission approached, I felt a mix of nervousness and excitement for what was to come.

15.7 April Reflection

In April it was full steam ahead working on every aspect of the app. It was a bittersweet feeling, as I was proud of what I had accomplished but also sad that I could see the project was coming to an end because despite the challenges the experience of developing Youtine was rewarding.