# CE301 Capstone Project

**ClickFit – Personal Training App**

**Bradley Gichangi – 1904015**

**Dr. Renato Amorim (Supervisor)**

**Prof. John Gan (2nd Assessor)**

**Computer Science**

Logo

Description automatically generated

## Abstract

ClickFit is an app which connects personal trainers to clients. Personal trainers are able to display their expertise on their page to attract a client base, they can make posts that may include their clients progress to encourage clients to work with them. Clients can search for trainers who are nearby and also choose trainers who specialise in the exact fields they are interested in. There are 2 main aims for ClickFit. First of which is to allow personal trainers to work independently from a third party such as a gym by connecting them directly to clients. Secondly, to allow clients to easily find personal trainers in their area.

To create my app, I am using Flutter which is an open source software development kit that allows for cross-platform application development and includes a geolocator plugin which I will be using . Furthermore, I will be using Firebase which is platform that allows for cloud storage (which I will be using to store the database) and instant messaging.

## Table of Contents

[CE301 Capstone Project 1](#_Toc102121869)

[Abstract 2](#_Toc102121870)

[Table of Contents 3](#_Toc102121871)

[1.0 Introduction 4](#_Toc102121872)

[1.1 What is ClickFit? 4](#_Toc102121873)

[1.2 What is the aim of ClickFit? 4](#_Toc102121874)

[1.3 Why is ClickFit a useful product? 4](#_Toc102121875)

[1.4 What are the competitors? 5](#_Toc102121876)

[1.5 What functionalities will ClickFit have? 5](#_Toc102121877)

[1.6 How will these functionalities be implemented? 5](#_Toc102121878)

[1.7 Why have I chosen this design? 5](#_Toc102121879)

[1.8 Work Breakdown Structure 6](#_Toc102121880)

[1.9 Storyboard 7](#_Toc102121881)

[2.0 Literature Review 8](#_Toc102121882)

[2.1 Flutter 8](#_Toc102121883)

[2.2 Android Studio 9](#_Toc102121884)

[2.3 Microsoft Azure 9](#_Toc102121885)

[2.4 Firebase 10](#_Toc102121886)

[2.5 Geolocator 10](#_Toc102121887)

[2.6 Google Maps Platform 10](#_Toc102121888)

[3.0 Sprint 1 - Login/Signup 12](#_Toc102121889)

[3.1 Introduction 12](#_Toc102121890)

[3.2 Requirements 12](#_Toc102121891)

[3.3 Design 13](#_Toc102121892)

[3.3.1 Login Page Design 13](#_Toc102121893)

[3.3.2 Signup Page Design 13](#_Toc102121894)

[3.3.3 Entity Relationship Diagram 14](#_Toc102121895)

[3.4 Development 14](#_Toc102121896)

[3.5 Testing 16](#_Toc102121897)

[3.6 Conclusion 19](#_Toc102121898)

[4.0 Sprint 2 - Profile Page/Following 19](#_Toc102121899)

[4.1 Introduction 19](#_Toc102121900)

[4.2 Requirements 19](#_Toc102121901)

[4.3 Design 20](#_Toc102121902)

[4.3.1 Profile Page and Search Page Design 20](#_Toc102121903)

[4.4 Development 20](#_Toc102121904)

[4.5 Testing 24](#_Toc102121905)

[4.6 Conclusion 26](#_Toc102121906)

[5.0 Sprint 3 - Feed/Posting 26](#_Toc102121907)

[5.1 Introduction 26](#_Toc102121908)

[5.2 Requirements 27](#_Toc102121909)

[5.3 Design 28](#_Toc102121910)

[5.3.1 Home Page Design 28](#_Toc102121911)

[5.4 Development 28](#_Toc102121912)

[5.5 Testing 30](#_Toc102121913)

[5.6 Conclusion 31](#_Toc102121914)

[6.0 Sprint 4 - Messaging 31](#_Toc102121915)

[6.1 Introduction 31](#_Toc102121916)

[6.2 Requirements 31](#_Toc102121917)

[6.3 Design 32](#_Toc102121918)

[6.3.1 Messaging Page Design 32](#_Toc102121919)

[6.4 Development 32](#_Toc102121920)

[6.5 Testing 34](#_Toc102121921)

[6.6 Conclusion 35](#_Toc102121922)

[7.0 Project Planning 36](#_Toc102121923)

[8.0 Conclusion 37](#_Toc102121924)

[References 38](#_Toc102121925)

# 1.0 Introduction

## 1.1 What is ClickFit?

ClickFit is an app which connects personal trainers to clients. Personal trainers are able to display their expertise on their page to attract a client base, they can make posts that may include their clients progress to encourage clients to work with them. Clients can search for trainers who are nearby and also choose trainers who specialise in the exact fields they are interested in.

## 1.2 What is the aim of ClickFit?

There are 2 main aims for ClickFit. First of which is to allow personal trainers to work independently from a third party such as a gym by connecting them directly to clients. Secondly, to allow clients to easily find personal trainers in their area.

## 1.3 Why is ClickFit a useful product?

By interacting with trainers in the industry, I found that gyms would take advantage of trainers, especially those who lacked in experience. They would attempt to charge extortionate rates for use of their facilities, or they would offer to pay them by the hour in return for a service that was worth much more. This app gives trainers control over their careers and allows them to brand, promote and sell their services to a much larger market, at just a fraction of the cost. Furthermore, the social side of the app allows them to connect with potential clients and market themselves in an app that is specific to the industry. People do not hire personal trainers to learn how to exercise, exercise tutorials are in abundance in this age of technology. The majority of people who hire trainers do so because they lack the motivation to learn on their own to actually do the exercise. Trainers provide these people with an external form of motivation that encourages them to learn the benefits of exercise and help them form positive and adaptive behaviours. Furthermore, unlike in most commercial gyms, clients are able to find trainers who are specialists, For example, a trainer who is qualified in exercise rehabilitation would be a perfect candidate for a person suffering with injuries but still wishes to get fit. This app allows clients to connect with trainers in a way that does not involve a third party, who increases costs and creates scheduling hassle. Instead, clients are able to book at their own convenience at a price that is set by the trainer based on their experience and expertise online.

## 1.4 What are the competitors?

There are many competitors on the space but none of which take this approach. Majority of personal training apps are membership based and provide a user with exercises that the user can do from home. Although there is a market for this, in person training has no substitute.

## 1.5 What functionalities will ClickFit have?

* A login page where the user can choose to make an account as a trainer or client, the trainer and client account will have different functions
* Messaging capabilities for the trainers to conversate with their clients
* Geolocation to measure the distance from the client to the personal trainer or their gym, this allows clients to find trainers in their vicinity
* Home page will consist of a twitter style feed from users that you follow
* In future, advertisement can be integrated into user’s feed and payment capabilities for clients to pay for sessions

## 1.6 How will these functionalities be implemented?

* For app development I will used android studio with flutter.
* Flutter is an open source software development kit.
* Flutter allows for development on both android and iOS platforms.
* To connect the application to a local database, XAMPP will be used for PHPMyAdmin.
* For a global database Microsoft Azure can be used.
* For messaging, the firebase plugin will be used.
* For geolocation, the geolocator plugin will be used.

## 1.7 Why have I chosen this design?

Easy intuitive design.

Black is the primary colour with red accents. Essentially the app will be in dark mode which is an increasingly popular design that is being implemented currently.

The red represents energy and passion which is fitting for an exercise-oriented app.

## 1.8 Work Breakdown Structure

Diagram

Description automatically generated

## 1.9 Storyboard

Graphical user interface, application

Description automatically generated

# 2.0 Literature Review

## 2.1 Flutter

Flutter is an open source software development kit that I am using with android studio. Flutter is a cross platform software, so it allows for development on both android and iOS platforms.

Dart is a programming language that is optimised and tailored for user interface development, there are also no competing standards. It is also a similar language to Java which I have previous experience with [2]. As Dart grows in popularity, the number of developers working with it also grows. This means there is growing support and issues I may come by can likely be solved because of this.

Although my app isn’t reliant on great graphics and high performance, flutter has the ability to support apps with 120fps which is not common for other cross-platform technologies.

I have not used Flutter prior, my first experience with it was in the creation of this app. Flutter has a wide array of clear documentation and a good community of developers that provide working examples and development templates which allow for new developers to learn how to use the platform. Furthermore, those that are learning and coming across issues share them with the community and allow other developers to fix similar issues by viewing their solutions.

Flutter uses a single code base that works for both iOS and Android platforms. This saves time and cost as separate codes aren’t needed to tailor to the different platforms. Platform primitive code is not used, instead widgets such as “ElevatedButton” are translated to fit the platform [9]. So, for iOS it would be “UI button” and “widget.button” for Android.

The most notable alternative to flutter is React Native, which is also an open source platform for cross platform development [6]. JavaScript is the programming language that is used on React which is one of the most popular programming languages so for new developers it is easy to get accustomed to. However, Dart is similar to Java which I am well versed in, so the use of JavaScript is not an advantage in my development.

As both Flutter and React Native are new to me, documentation is important as the more detail there is, the easier it is for me to become acquainted to the software [3]. However, Native is more dependent on developers’ previous use of JavaScript making it more difficult to learn than flutter. Furthermore, React Native enables the communication between JavaScript and the native language via the JavaScript bridge. This is done by the 2 sides communicating by sending JSON messages. Although the user interface does run smoothly, it is still possible for it to lag as the use of a bridge impacts the rendering speed. Flutter doesn’t require a bridge as the components are native and included in framework such as Cupertino and material design meaning it is more stable and predictable.

## 2.2 Android Studio

Android Studio is the official integrated development environment for Android’s operating system. It uses JetBrains’ IntelliJ IDEA software and I am using dart which is a programming language which is similar to Java [1]. In my other modules I program with IntelliJ IDEA using Java, so this software is something I am familiar with which will be advantageous to me in the development of my application. Android Studio provides a wide range of android emulators from Google Pixel to Samsung Galaxy, it also includes Android television operating system and tablet operating system. These emulators are advanced and have numerous features, including the ability to simulate GPS location, motion sensors, network latency and multi-touch input. What I will be using however, is the web emulator as it is a lightweight emulator and doesn’t demand a large amount of system resources, and the machine I am running the application on isn’t high performance. Android Studio also provides templates and sample applications which are helpful for initial development as it can provide a good starting point to build off of [11]. The Flutter SDK is used within Android Studio, and it comes with a Firebase assistant that helps the application connect to Firebase to provide features such as authentication and notifications. All these things work hand in hand to fulfil all the needs of my application.

Previously, I have developed an application using the Unity engine which is a cross-platform engine primarily used for the development of games [8]. Although, it is a software that I am accustomed to and is capable of application development it is not optimised for this purpose. If I wanted to incorporate a game aspect to my application, then Unity would be ideal especially if three dimensional elements needed to be integrated.

## 2.3 Microsoft Azure

Microsoft Azure, formerly known as Windows Azure, is Microsoft's public cloud computing platform. It provides a range of cloud services, including compute, analytics, storage and networking [10]. Users can pick and choose from these services to develop and scale new applications or run existing applications in the public cloud. Azure and Firebase have no similar categories. Azure is categorized as Cloud Platform as a Service (PaaS) and uses relational DBMS compatible with SQL server. In comparison, Firebase is a Baas, supported by a data store. Originally, the plan was to use Microsoft Azure as it is a platform that is commonly used for hosting databases and supports the use of both SQL and NoSQL. However, although firebase only provides NoSQL non-relational databases, it also provides cloud messaging. This means, if I use the firebase all the tools that are required within my project are centralised within one platform. Furthermore, as firebase does not use a relational database format, the retrieval of data is faster.

## 2.4 Firebase

Firebase is Google platform used for creating mobile and web applications. It manages all data real-time in the database. This is ideal as I am implementing an instant messaging functionality in my application. As Firebase is a real-time database, it allows syncing of data across devices without the need of refreshing the screen [12]. Furthermore, I am intending to implement geolocation services in my app so that people are able to be recommended trainers that are in their vicinity. A real time database is useful for GPS so that is another important use for firebase.

Firebase also provides a cloud storage service which can be used to store images which can be used to store profile images and images in user posts [5]. Initially for proof of concept, I have been using a MySQL database which is stored locally using XAMPP. The problem with this is that this means that the data can only be accessed from my computer and is unable to be scaled up. Firebase cloud storage is capable of being used in apps that are widely popular so if the numbers of users greatly increase the method of storage will not need to be substituted. Firebase stores data in JSON format. This data format is available via REST APIs or client libraries readily. The data will be stored in the cloud, and ultimately, I will be able to access it anywhere, anytime, without any problem. During early stage development I am using a MySQL database and I am extracting the data into my application by using JSON, this means the switch over to Firebase will be seamless.

The Parse server allows its users to self-host. It is an open-source service supported by open-source developers. Firebase provides central data management, whereas the Parse server uses different databases to store different kinds of databases. For instance, MongoDB for data storage and Amazon s3 bucket for file storage [7].

## 2.5 Geolocator

One of the main aims of my app is to find trainers in the vicinity of the user. This requires geolocation services in order to measure the distance between the user and trainer. The functionalities of the Geolocator plugin include: Gets last known location of device, gets current location of the device, provides continuous locations updates, checks if location services are enabled on the device, calculates the distance between two geocoordinates and calculates the bearing between two geocoordinates [4]. Geolocator gets the latitude and longitude values which are coordinates to numerically describe the device’s location. These values are useful when performing calculations or finding precise locations, however, it is not user friendly as they are essentially meaningless. To solve this issue, the geolocator and geocoding packages are used in conjunction, to derive the address from latitude and longitude coordinates. Placemark also contains information like locality, post code and country [5]. For example, the coordinates latitude: 37.79, longitude: -122.4 corresponds to a place in San Francisco, USA. When using location services there are a number of things to consider, first of which is permissions. In order to store location data, the user must allow for the system to save this data so the user must be prompted. Moreover, as it is a cross-platform application the settings may have to be catered depending on which platform the application is being ran on. For example, in Android 11 the user has to accept the location permission [13] to always allow to use the background location but the option to always allow is not presented on the location permission dialog prompt. The user has to enable it manually from the app settings. Additionally, in iOS the user has ability to deny location services forever so using the request permissions function will not show the native prompt asking for location permissions.

## 2.6 Google Maps Platform

The Google Maps Platform [14] is a helpful tool that I can use in my project and is commonly used in numerous large commercial apps and websites. The Google Maps Platform is a set of SDKs and APIs that allows developers to embed Google Maps into mobile apps and web pages, or to retrieve data from Google Maps. There are three main uses for the platform: maps, routes and platform. In the maps section they offer Maps SDKs [15] which allow the user to view dynamic maps within the website or application. Embedded and static maps that are interactive and can be used within a website. Street view imagery is also available which allows a 360-degree view of the location from the POV of a person which is helpful as it gives the user a virtual tour of the location. Routes are especially useful for my application as it provides directions to the locations including multiple transportation modes alongside real-time traffic information. This would be helpful for users when they plan to travel to their trainers or possibly if the trainer schedules a home visit. Routes also contains a distance matrix which is used to calculate travel times and distances from multiple origins to multiple destinations. The Google Maps Platform also contains places API and SDKs which contains details on establishments, and notable areas such as restaurants, banks, parks and so on. However, the only places that would be relevant to my application would be places that deal with health and fitness such as gyms or physiotherapy centres. Furthermore, Maps is capable of geocoding which converts longitude and latitude coordinates into addresses and addresses into coordinates. The coordinates from the user can be saved in the database and used to measure distance even when the other user is not actively on the application. Geolocation is also available which gets the approximate location of the user’s device by using nearby cell towers and Wi-Fi nodes. To incorporate the API in my project I would just have to make a project and Google cloud and use the API key provided to link Google cloud to my application. The service allows for free use of up to $200 which would be more than enough for my project at this stage, if I wanted to upscale in future, I would have to take this fee into account.

My application is built to be a cross-platform application that is usable on both iOS and Android platforms. However, running the app on an emulator is very CPU intensive and the system I am running this application on does not have the required power for it to run quickly and smoothly. For this reason, I am emulating as a web application which allows the application to run smoother but in doing so, I am unable to use Google Maps Platform as Flutter does not support a windows version of Google Maps. Therefore, despite Google Maps Platform offering a much greater service, with a wider array of components that could be useful I will be using the Geolocator plugin as this is able to work in a web app. This is something I can improve on in future development when I have the necessary resources.

# 3.0 Sprint 1 - Login/Signup

## 3.1 Introduction

This sprint is focused on the login and signup functions. The purpose of this is to allow the user to create their own profile which they have exclusive access to and can change their profile details within the app. There are also 2 different interfaces for the trainers and clients and depending on the signup option will determine what type of accessibility that specific account will have.

## 3.2 Requirements

**Login function**

**Description:** Page that accepts user’s input to login to app.

**Summary:** User’s input is accepted and compared to records in database to check if there’s a match by using an SQL select query. User is granted access if a match is found.

**Signup function**

**Description:** Page that accepts user’s input to sign up to app.

**Summary:** User inputs information into text fields and information is inserted into the database using an SQL insert query and the user is granted access to the home page.

**Logout function**

**Description:** User can log out of their account

**Summary:** User is able to log out of their account and taken to original login page. They are then able to log in to any other account.

**Incorrect login/signup prompts**

**Description:** User is prompted when incorrect details are inputted into text fields

**Summary:** Prompts are provided when text fields contain incorrect details for the login or incorrect formats for text fields in signup.

**Password censorship**

**Description:** When user is inputting password, it is censored

**Summary:** Password text field is configured to be obscure text so that text in password field isn’t intelligible.

**Password hashed**

**Description:** Password hashed before being stored

**Summary:** Password is hashed using a hashing algorithm so if database is infiltrated, passwords cannot be easily accessed.

## Graphical user interface, application Description automatically generated**3.3 Design**

### **3.3.1 Login Page Design**

### Graphical user interface, application Description automatically generated3.3.2 Signup Page Design

### 3.3.3 Entity Relationship Diagram

**Diagram

Description automatically generated**

The entity relationship diagram is a representation of the relational database I am using in my project. The tables are created to suit the main functionalities of my program. Firstly, the login and signup page use the user’s information in order to verify that the inputs are valid, and this uses the users table which is the primary table in my database. The profile page uses the followers table to keep track of which people each user follows and displaying a following and follower count. For the profile pages of the trainers, the user expertise table is used to allow the trainers to display their expertise so they can find client who are seeking training in their expertise. Lastly, the posts table is used for the home page where posts made by users and the people, they’re following will appear on the home page in a twitter-style feed. These posts will also appear on user’s profiles. The database is also in third normal form, I use unique primary keys and foreign keys to link related tables. There are also no transitive functional dependencies. The primary key for the users’ table is the email as it is the only unique identifier out of the data we ask from the user. A username could be used but using an email saves you from remembering different usernames as they could possibly be taken on different sites/apps. The followers table contains emails as foreign keys from the user table as they’re also users and the primary key is an ID as a single user can follow multiple users and a unique identifier is needed. The user-expertise table consists of two foreign keys as its purpose is to be an intermediary connection between the users and expertise tables. The expertise table has an ID as a primary key the reason for this is that if need be, changes can be made to the table and they will be reflected in every use of the record rather than having to change every individual record.

## 3.4 Development

To incorporate the login and signup functions, I originally used XAMPP which provides a local MySQL relational database. To bridge my app with the database I used an SQL query in PHP and converted the results into JSON. The JSON data is then parsed into flutter. For the login the input is compared to the JSON data that is inputted. For the signup function, a POST method is used to parse the user input to PHP and an insert query is used to save the data into the database.

A condition statement is used in order to verify all inputs are valid before being saved into the database (Fig.1). Firstly, I check that all the inputs are not left empty otherwise the user is notified by an error that is raised that displays “text field cannot be left blank”. FlutterToast is a useful plugin for this purpose. Secondly, the email input is verified to check its validity, and this is done by using the email validator plugin. If invalid, an error is raised explaining the case.

Once all checks are undergone, the data is parsed to a PHP file.

Graphical user interface, text, application, email

Description automatically generated

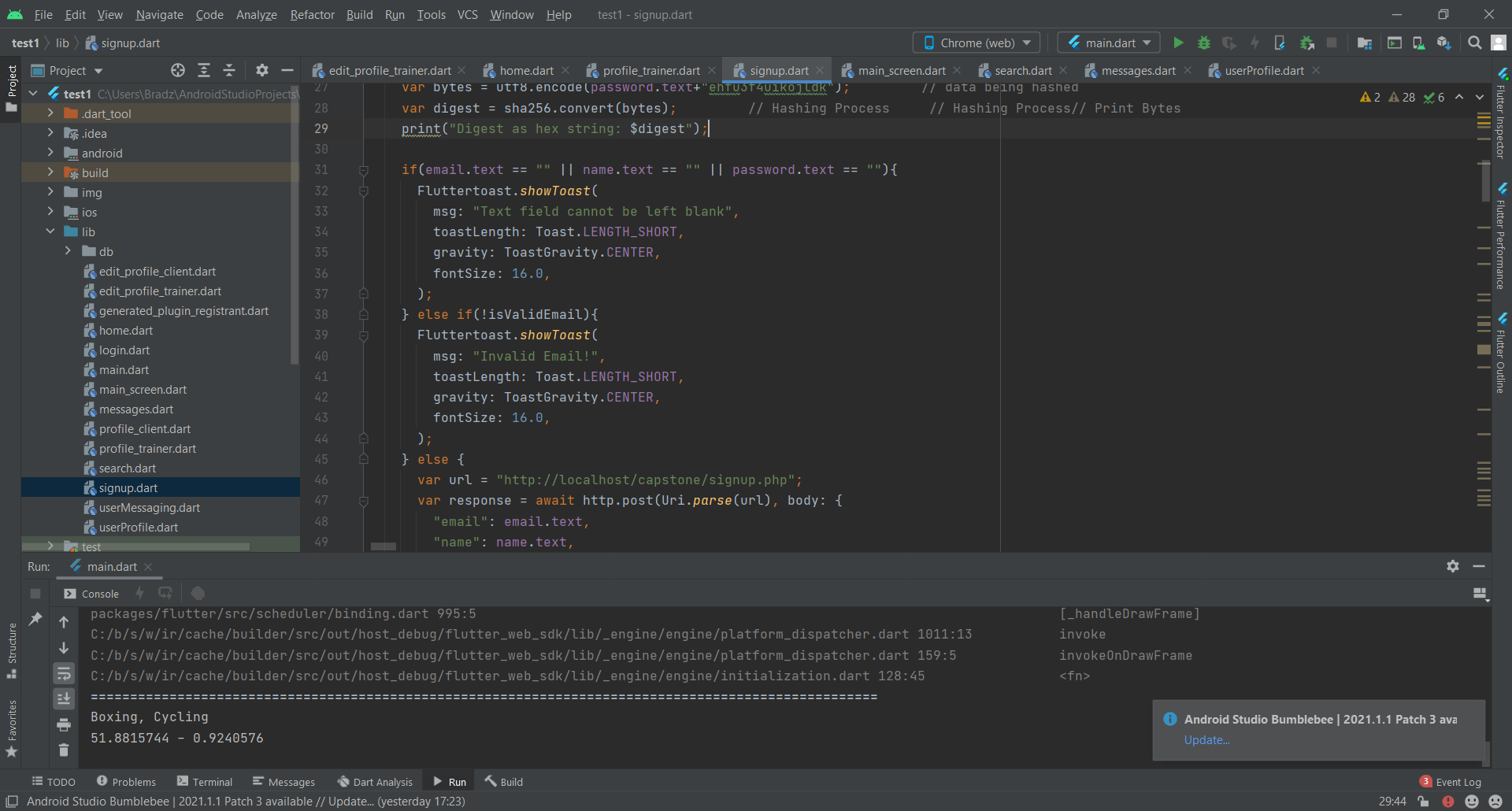


Figure 1

Figure 2

The data is received via post requests (Fig.2) and then inserted into the users table using an SQL insert query.

Similar to the signup function, the inputs are checked to see if they are empty and if the email is valid. The relevant errors are displayed (Fig.3) if these conditions are met or else the password is hashed and parsed alongside the email to the PHP file. The password has a salt concatenated then SHA256 hashing algorithm is applied in order to ensure security.

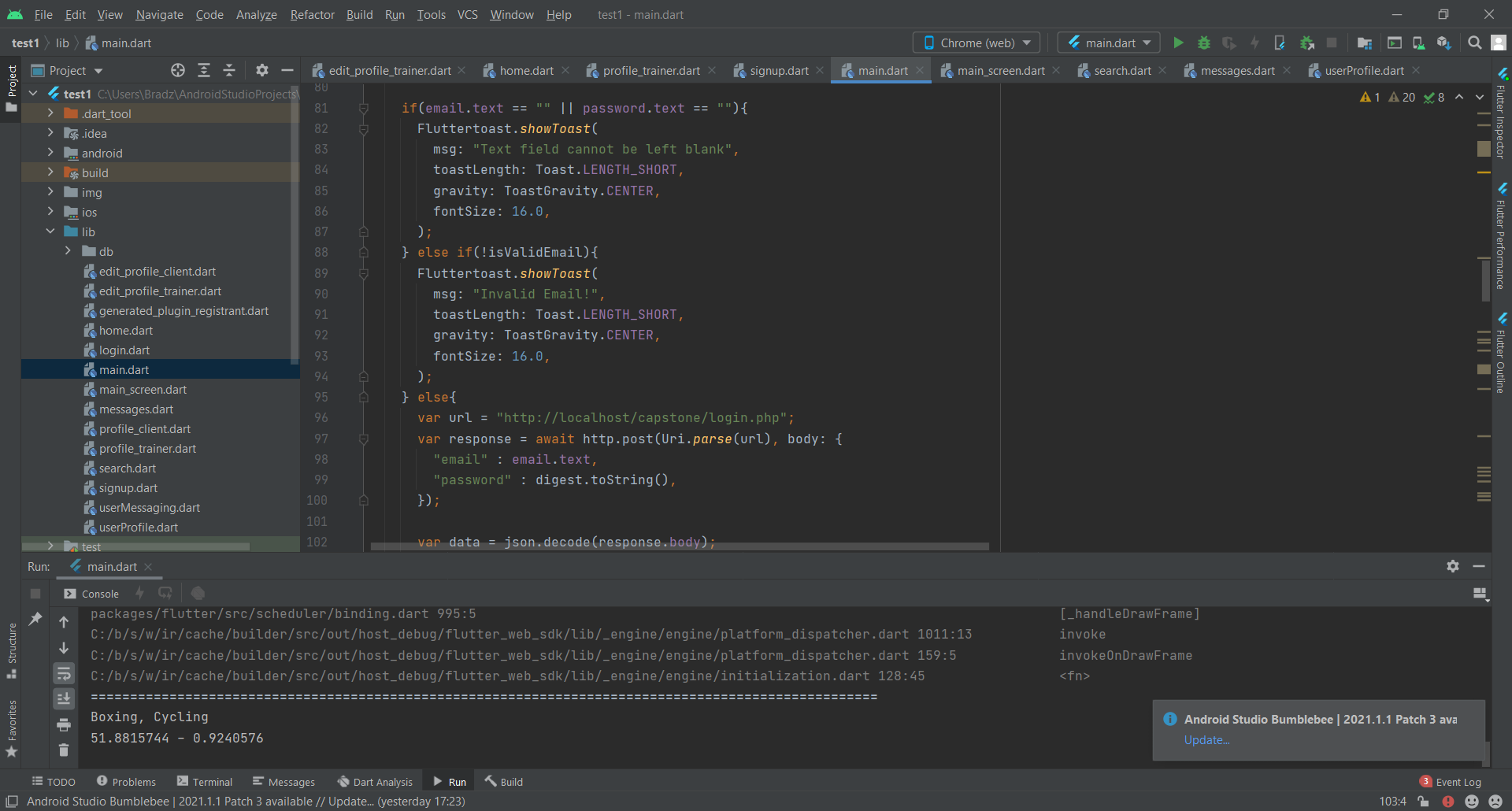


Figure 3

In the PHP file, the email and password are received, and a select statement is used to check if the user exists in the user statement (Fig.4). If they do, the user is granted access to the home page, else an error is raised explaining the user doesn’t exist.

Graphical user interface, text, application, email

Description automatically generated

Figure 4

Later into development I found a better way to incorporate these functions. This included using firebase which is cloud storage. Firebase has a feature called authentication which is used for the creation and verification of accounts.

Using firebase authentication (Fig.5), the whole process is streamlined, and signup can be done with one line which is shown on the top, the password is hashed automatically in the authentication database. However, the non-sensitive data is also saved in the Firestore database in the “userinfo” collection. The reason for the data to be saved twice is because the authentication database cannot be accessed by other users so if I wanted to search for other users it would not be possible. As Firestore is cloud storage, it doesn’t occupy local storage while also requiring less code. Therefore, this is an improvement if my initial code.

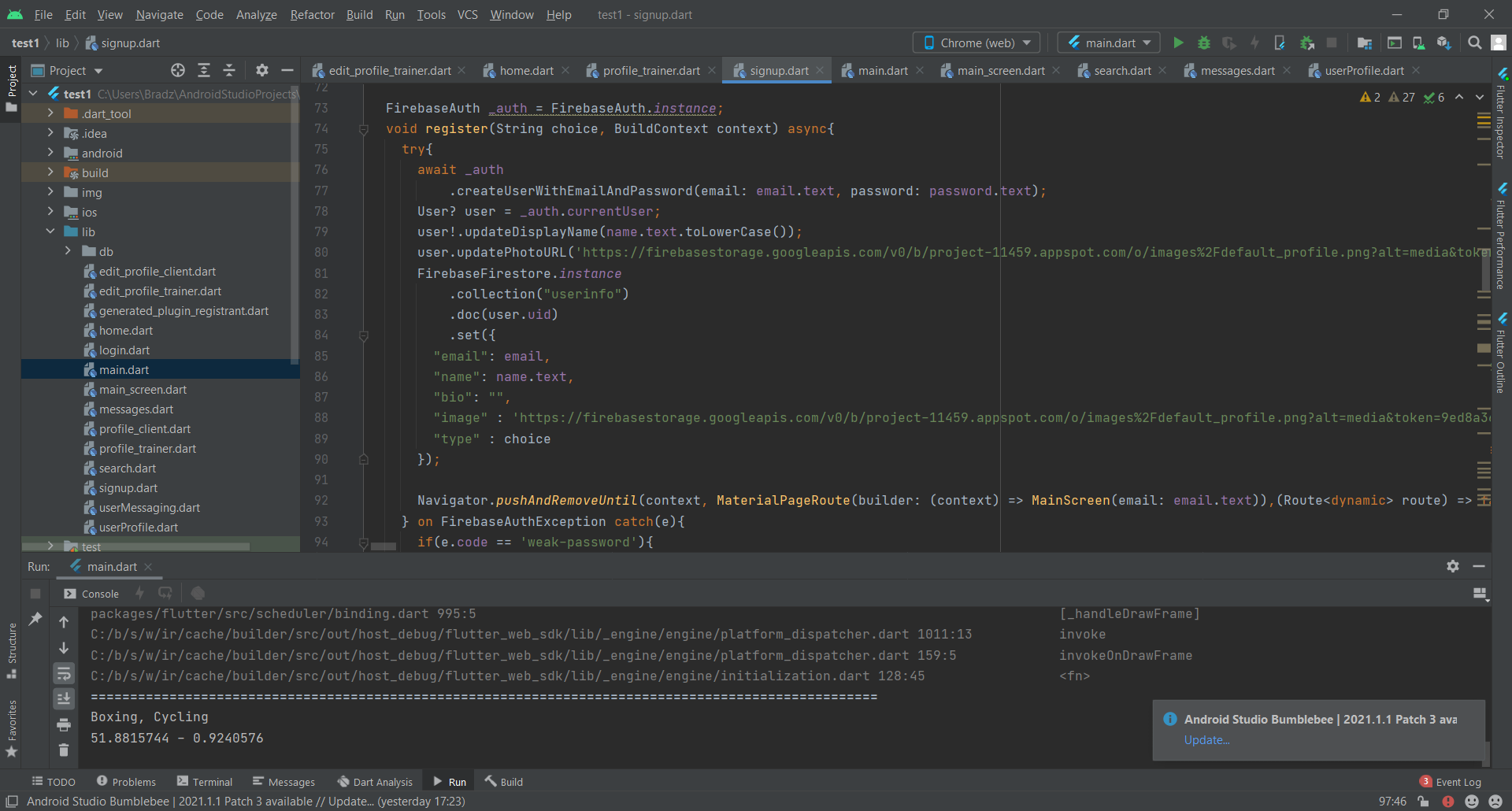


Figure 5

The login also only requires a single line of code within a try/catch statement (Fig.6) in case the inputs are incorrect, and the error is displayed. If the inputs are valid and exist in the authentication database, the user is granted access to the home page.

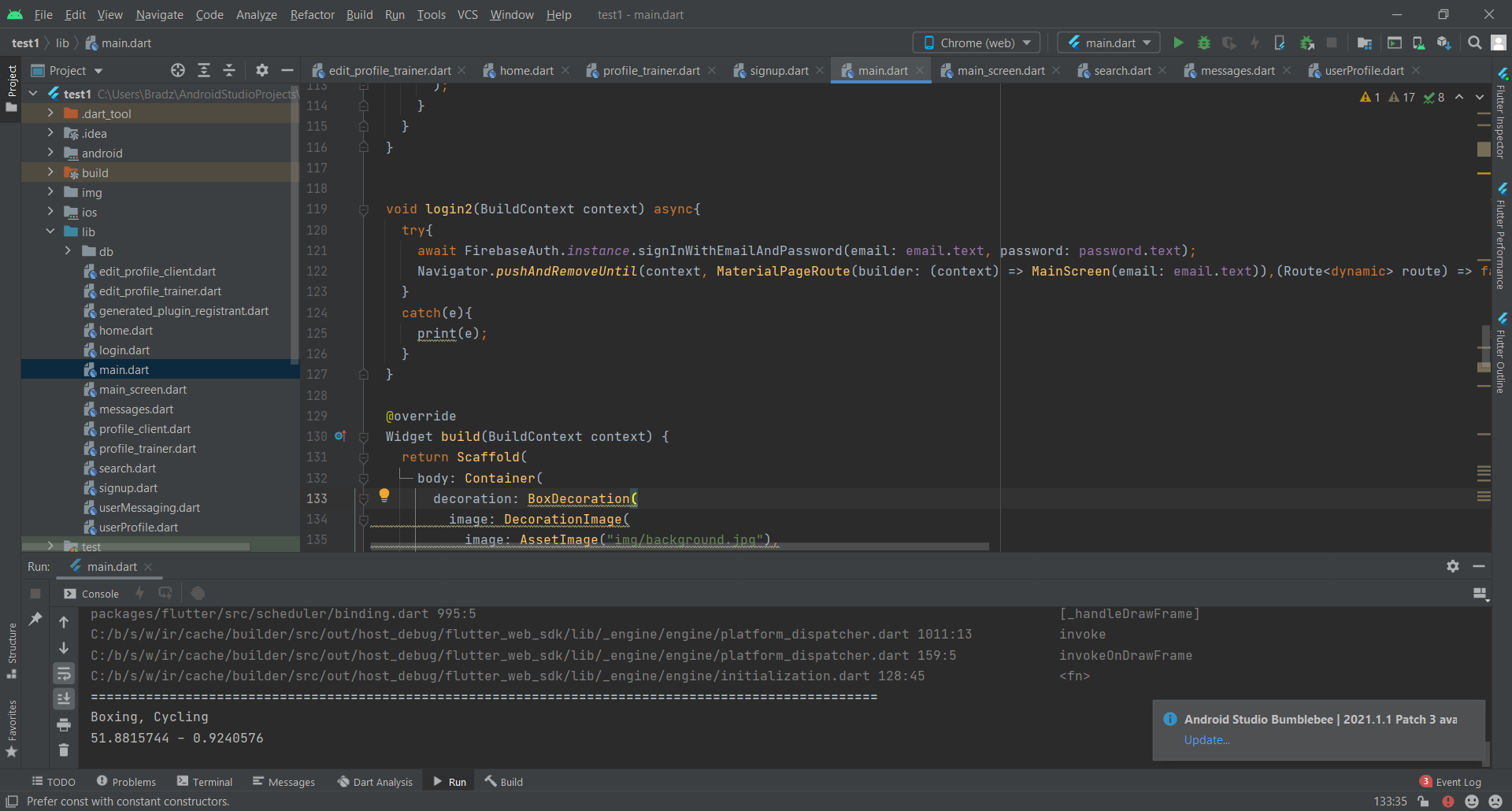
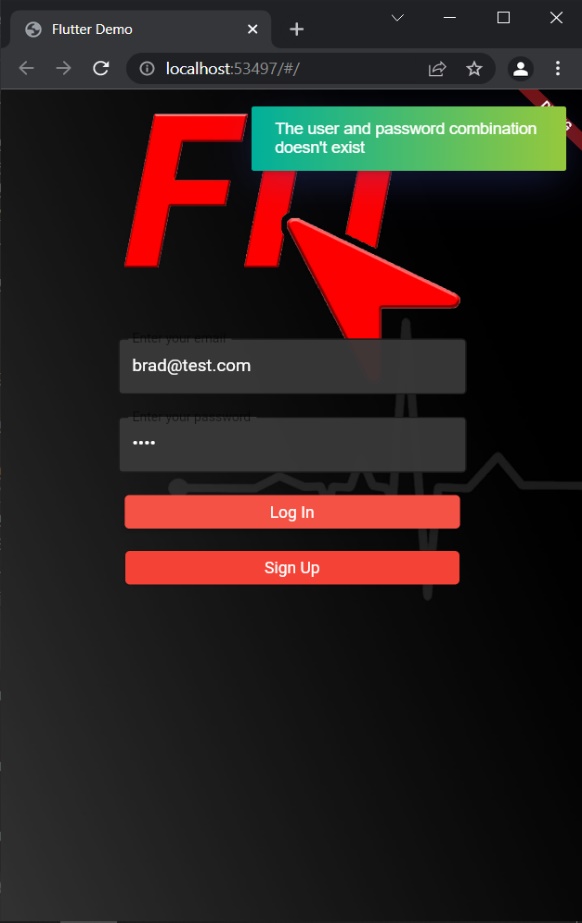


Figure 6

## 3.5 Testing

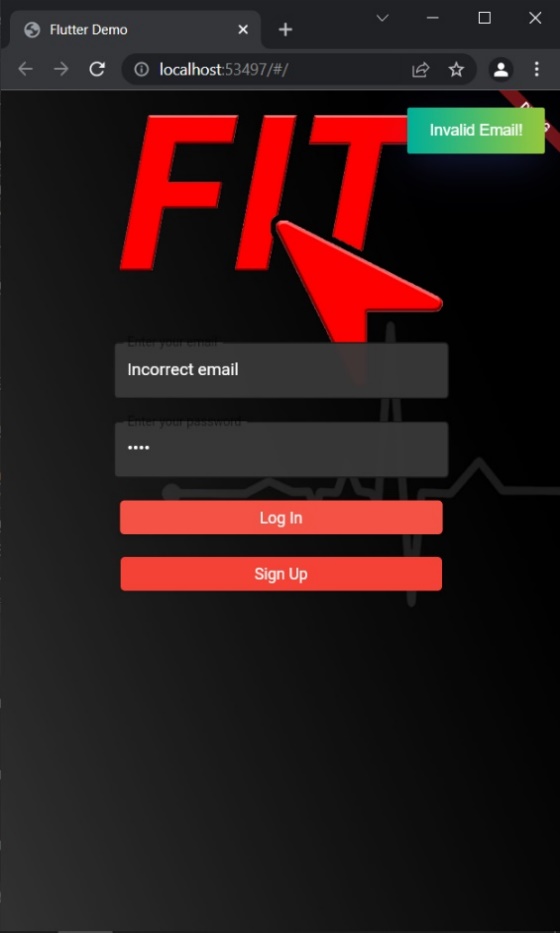
Table 1

|  |  |
| --- | --- |
| **Test** | **Outcome** |
| Log in with correct details | Pass. Login details that have been stored in the database were used to login and the user gained access to the home page. |
| Log in with wrong email | Pass. See Fig. 1 Login details that included an email that did not exist in the database were used and an error was shown explaining the email is invalid. |
| Log in with wrong password | Pass. See Fig. 2 Login details that included a valid email, but incorrect password was used, and an error was shown explaining that the combination of inputs doesn’t exist. |
| Log in with no inputs | Pass. See Fig. 3 Login page input fields were left empty and login was attempted. Error explaining text fields cannot be left blank is shown. |
| Signup with valid inputs | Pass. Valid signup details that don’t exist in the database are used and the user gains access to the home page. |
| Signup with no inputs | Pass. See Fig. 4. Signup page input fields were left empty and signup was attempted. Error explaining text fields cannot be left blank is shown. |
| Signup with used email | Pass. See Fig. 5. Signup is attempted with an already existing email in the database. An error is raised explaining that the email has already been used. |
| Log out of account | Pass. When log out button is used, user is logged out and taken back to the login page. |



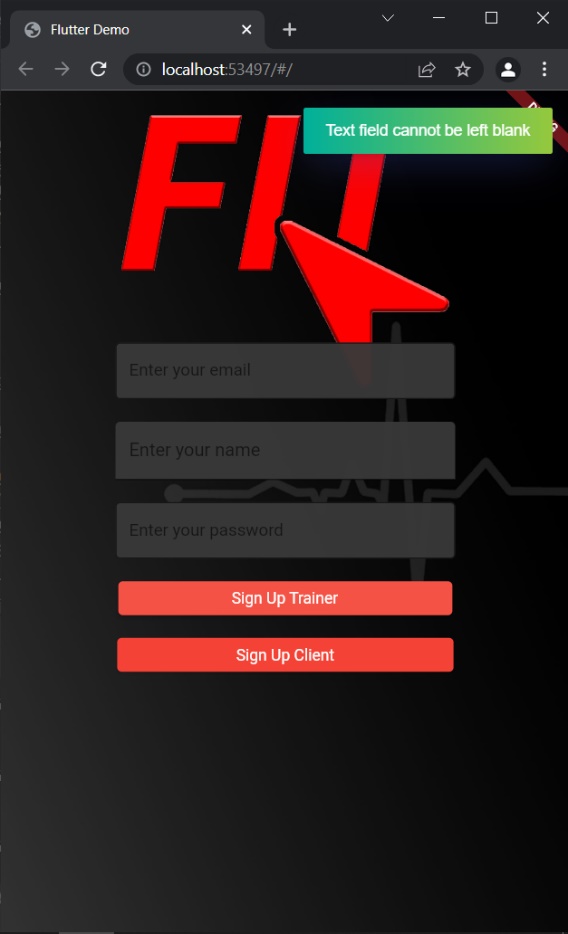
**Login with wrong password test**

Figure 7



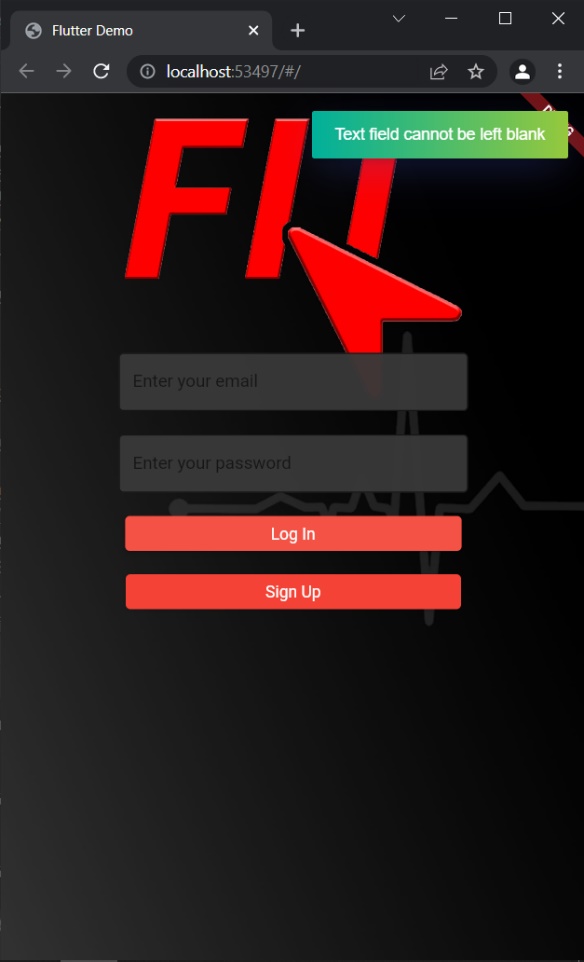
**Login with wrong email test**

Figure 8



**Signup with no inputs test**

Figure 9



**Login with no inputs test**

Figure 10



**Signup with existing email test**

Figure 11

## 3.6 Conclusion

All tasks that were set out for this sprint were a success. I was not able to replicate my login and signup designs exactly, as the original designs included radio buttons to choose between signing up as a trainer or a client. I had trouble with configuring radio buttons within Flutter nonetheless I was able to create an alternative solution and use 2 separate signup buttons instead. Another issue I was having was raising errors when invalid inputs were used however, I discovered a plugin called Fluttertoast which worked for this specific purpose. This section is important as the app is useless without the ability for the user to create an account and builds the basis for my next sprint which focuses on the profile pages of the users.

# 4.0 Sprint 2 - Profile Page/Following

## 4.1 Introduction

This sprint is focused on creating the interfaces for the profile page of the user and other users’ profile that they are visiting. The profile pages for clients and trainers are distinct and will have different capabilities as the trainers’ must be able to choose expertise that can be displayed to people when their profile is searched. The search page will also be created to be able to find other users within the app. With this functionality they will be able to be follow other users and be able to view their posts (which will be worked on in the next sprint).

## 4.2 Requirements

**Add image to profile**

**Description:** User is able to change profile picture

**Summary:** User can edit their profile and change their profile image by uploading an image, this is then stored in the database by using an SQL update query to the users’ table.

**Change profile name**

**Description:** User is able to change their name

**Summary:** User can edit their profile and change their name by changing the contents of the name field in the profile editor. The changes are then reflected in the database by using an SQL update query.

**Change profile password**

**Description:** User is able to change their password

**Summary:** User can edit their profile and change their password by using the change password function. The current password must be inputted in order to verify that it is the correct user, then the new password is inputted and saved. This password is then hashed and inputted into the database by using an update SQL query to apply changes to the users’ table.

**Change trainer expertise**

**Description:** Trainer is able to change their expertise

**Summary:** User can edit their profile and change their expertise by choosing from a selection provided. They can also add to the selection if their expertise isn’t available. An SQL insert query is used to add to the expertise table if it doesn’t exist. If an existing option is chosen an SQL insert query is used to add to the user-expertise table to link the trainer with the expertise.

**Add expertise**

**Description:** Trainer is able to add expertise

**Summary:** Using edit profile interface, the trainer is able to add expertise that are not already existing in the database.

**Follow users**

**Description:** User is able to follow other users

**Summary:** User is able to access other users’ profiles via the search page. Pressing the following button, a connection between the user and the person they are trying to follow is inputted into the database using an SQL insert query into the followers table.

**User search**

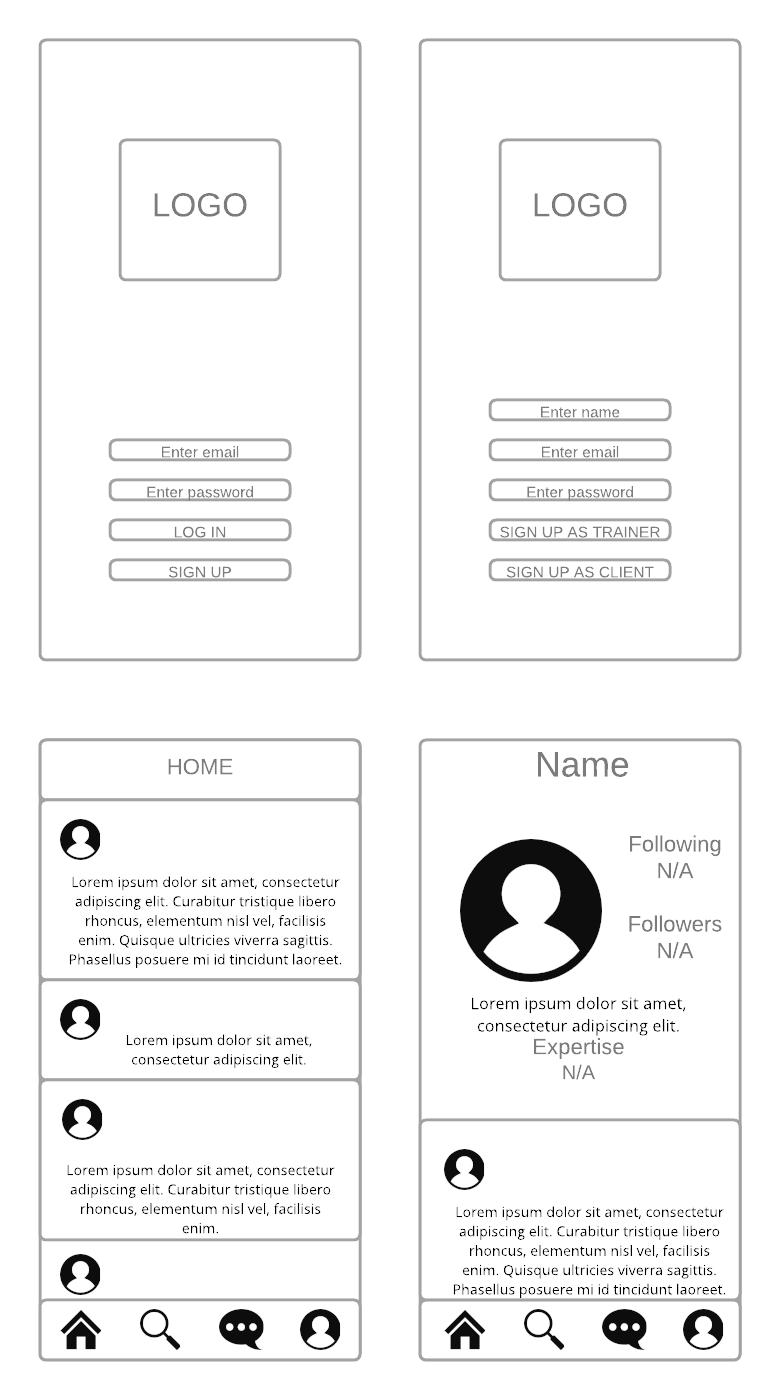
**Description:** User can search for other users using search bar.

**Summary:** User inputs text into the search bar, this then filters the users from the database using SQL queries. When users are displayed, they are able to be clicked to view the profile of the user and can view information such as: following/followers, posts made by the user and other relevant information.

## 4.3 Design

### 4.3.1 Profile Page and Search Page Design

**Graphical user interface, text, application

Description automatically generated**

## 4.4 Development

The profile page has three separate screens, first of which is the profile page of the user themselves. This is accessed when the user visits their own profile page, their name, profile image, follower and following count, bio, expertise (if they’re a trainer) and previous posts made by the user. The other page is the edit profile page where the user can edit the information on their own account, this is accessible via the profile page. In the edit profile page, the user is able to choose a new image, change their, name, bio and password. They’re also able to choose expertise and add a new option if the option doesn’t exist. In order for the user to change their password however, they have to also input their current password. This is a security measure in order to make sure that this change is made by the account holder. The last instance of the profile page is when a user visits the page of another use, this page is similar to when the user visits their own page however, there is no option to edit the profile and there will be an option to follow the user or to message them.

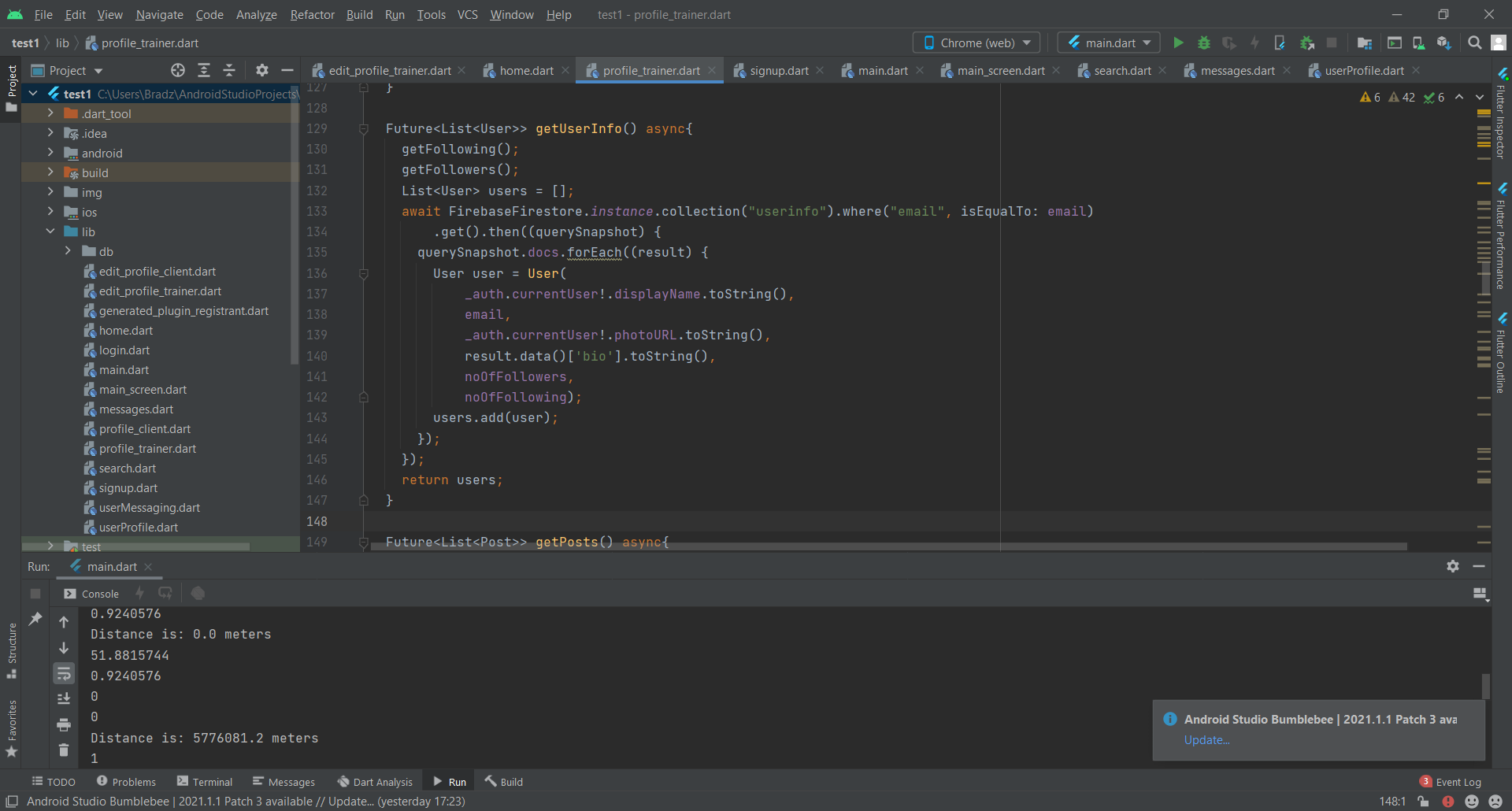


Figure 12

To display the user’s information on their profile it must first be read from the Firestore database as shown in Fig.12. The “userinfo” collection is searched and the user’s email is used to filter the documents to find the relevant record. A new user object is then created which stores the user’s name, email, profile image, bio, number of followers and number of users they’re following.

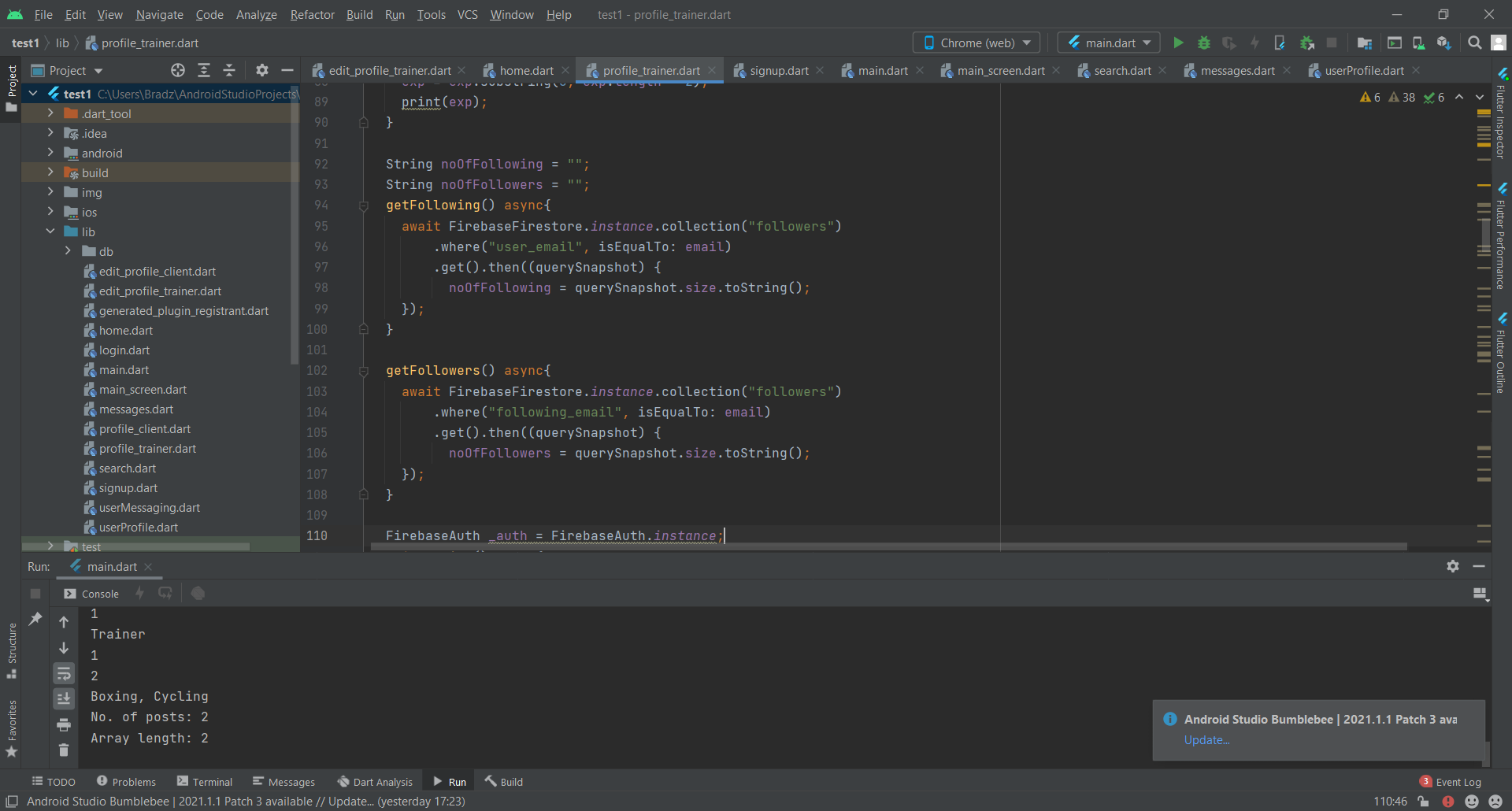


Figure 13

To get the number of following and followers the user has, getter methods are called. In get following the followers collection is queried and filtered for records where the “user\_email” matches the user’s email (Fig.13). The number of records that match this query corresponds to the number of people that user follows, and it is then saved to the global string “noOfFollowing” which is used to add to the user object. Similar to the previous method, get followers queries the followers collection as well however, the “following\_email” field is matched to the email instead. The number of records that are returned corresponds to the number of followers that user has, and this value is then saved to the global string “noOfFollowers”.

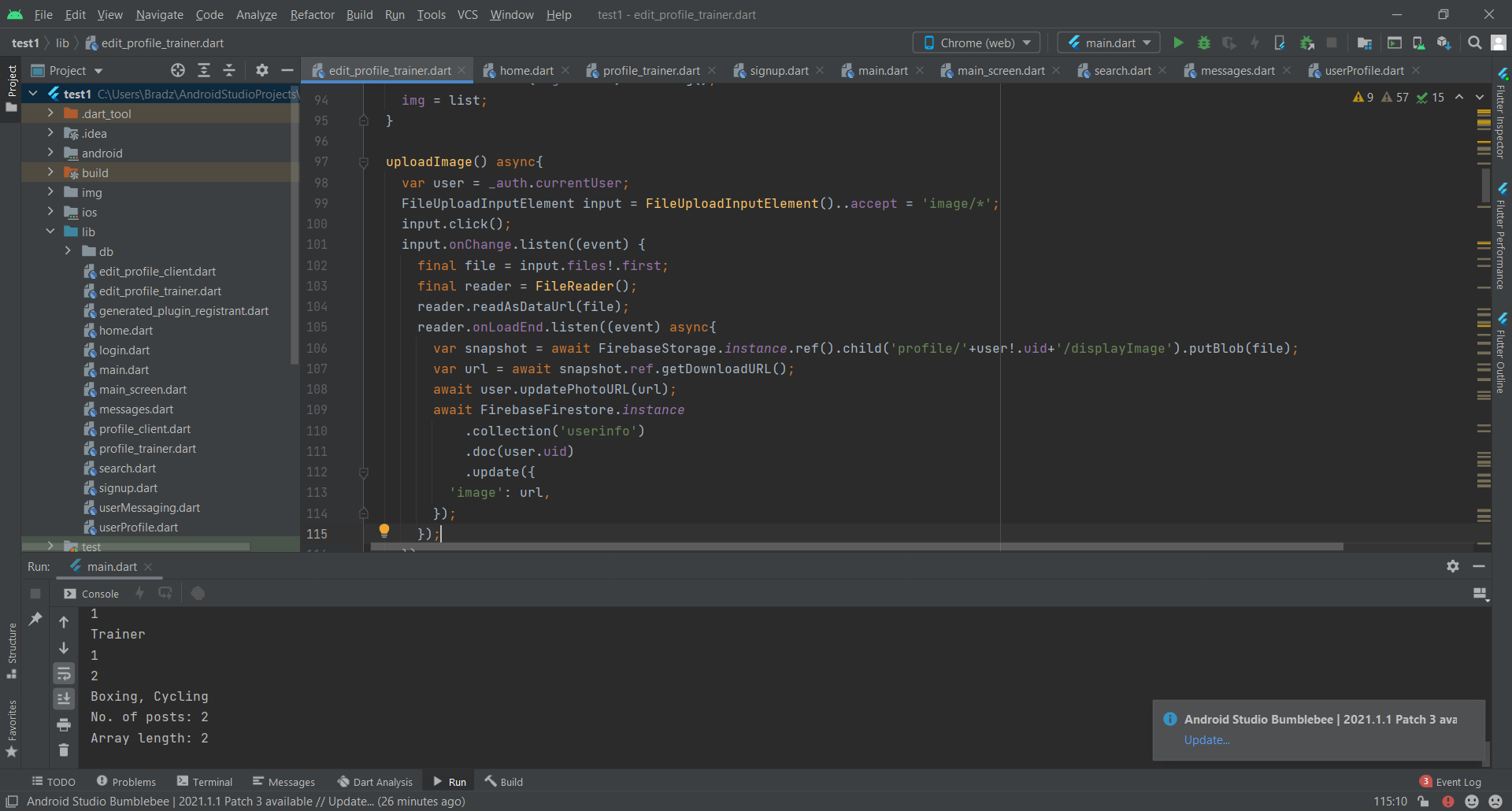


Figure 14

In the edit profile page, the user is able to change their profile image by uploading an image (Fig.14) from their device. File upload input element requests the user to input an image from the gallery, the file is then received by a file reader object which is stored into Firebase cloud storage. The URL for this image is then stored in both the authentication and Firestore database.

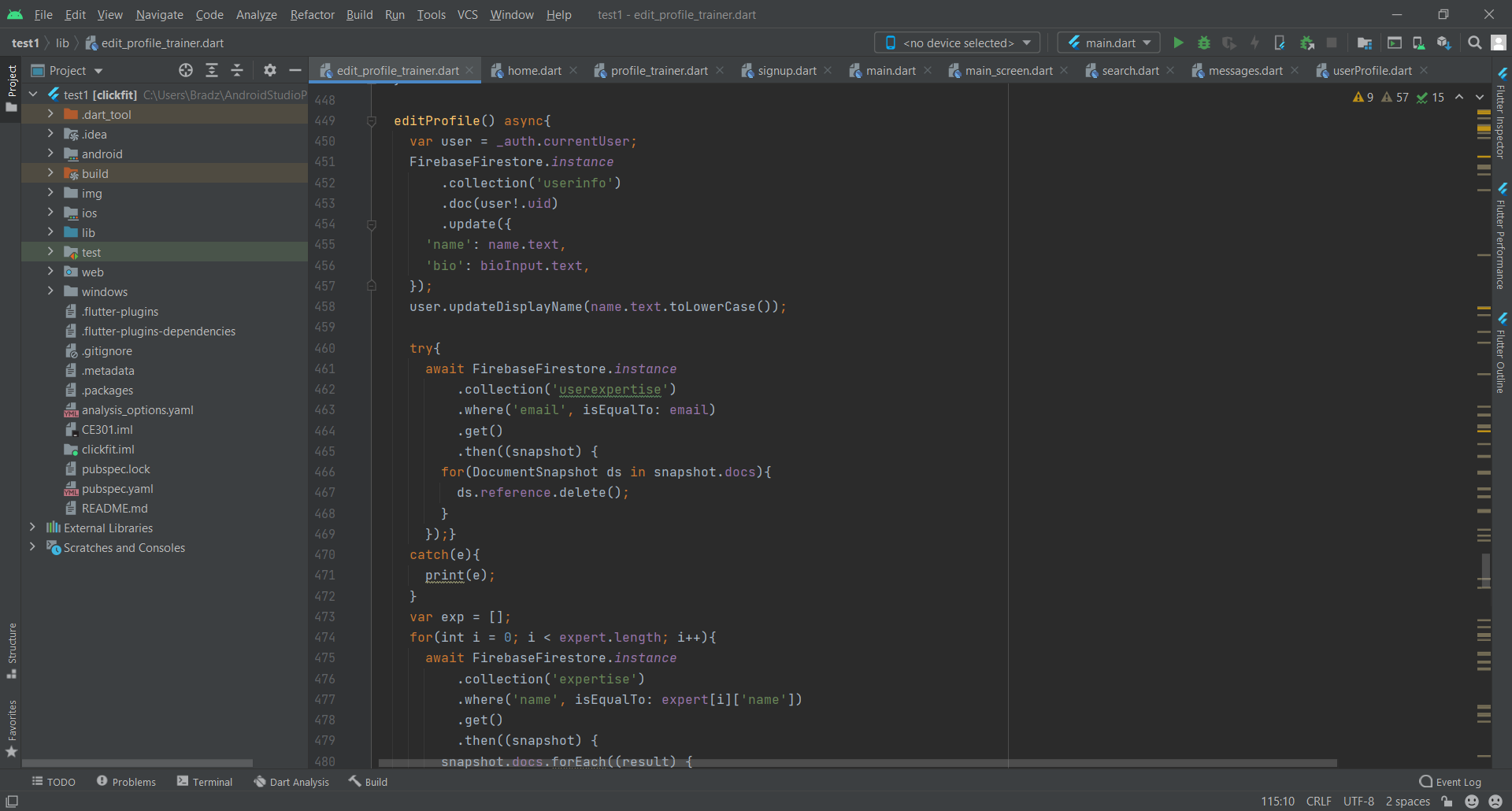


Figure 15

To edit the user’s name and bio, the text field inputs are taken and used to update the “userinfo” collection (Fig.15). The name field is stored as lowercase so that the names are stored in a uniform standard which would be useful for future development, more specifically in user search.

When changing the password (Fig.16), there has to be a verification process as a measure of security. This is to make sure only the real account holder can change the password. To do this the user must input their old password correctly before being able to change it. If it isn’t correct an error is displayed and if the password is too weak an error is also displayed to ensure that further security of the accounts. By using Firebase authentication, the passwords are automatically hashed with a salt in the backend.

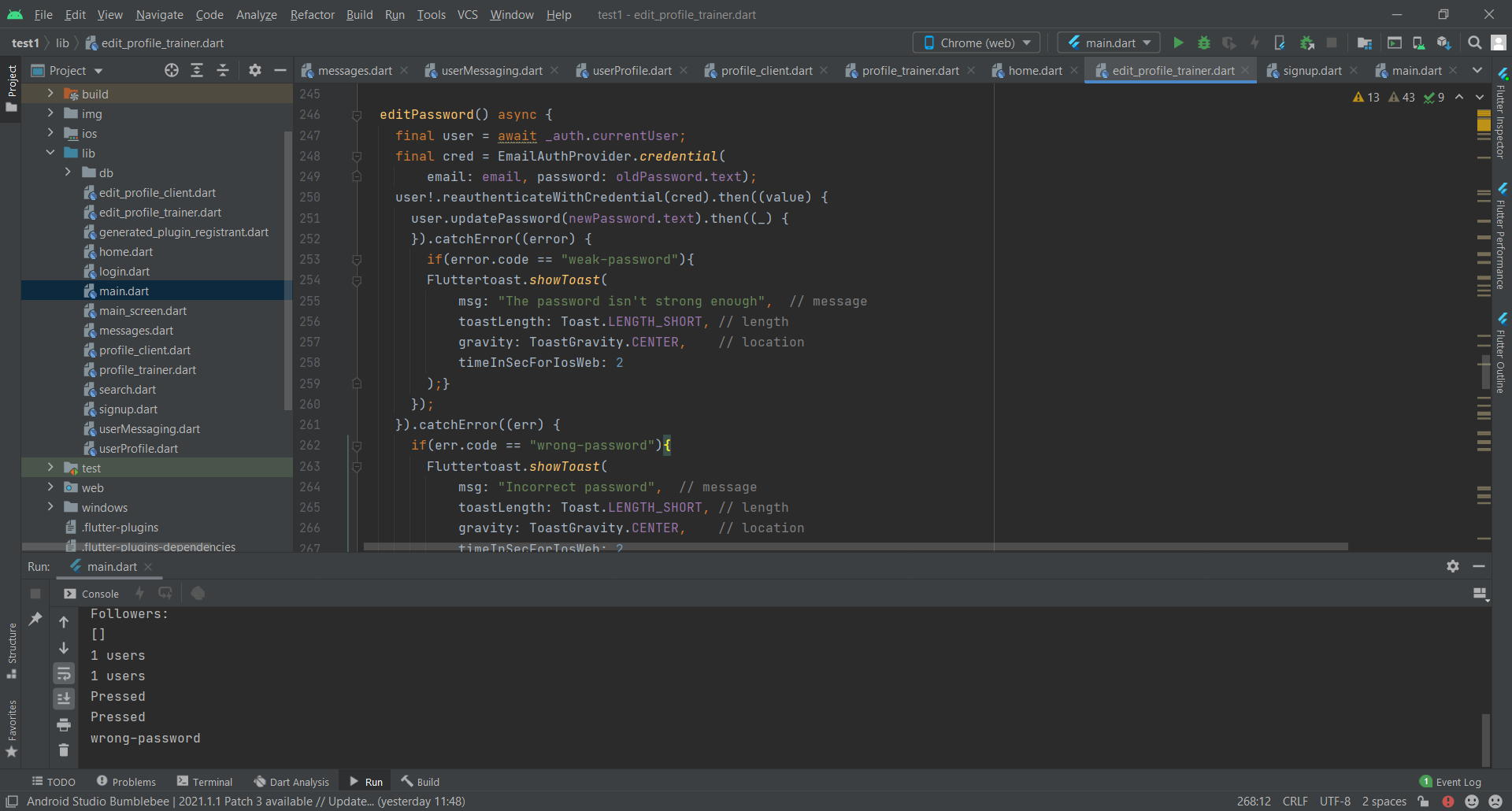


Figure 16

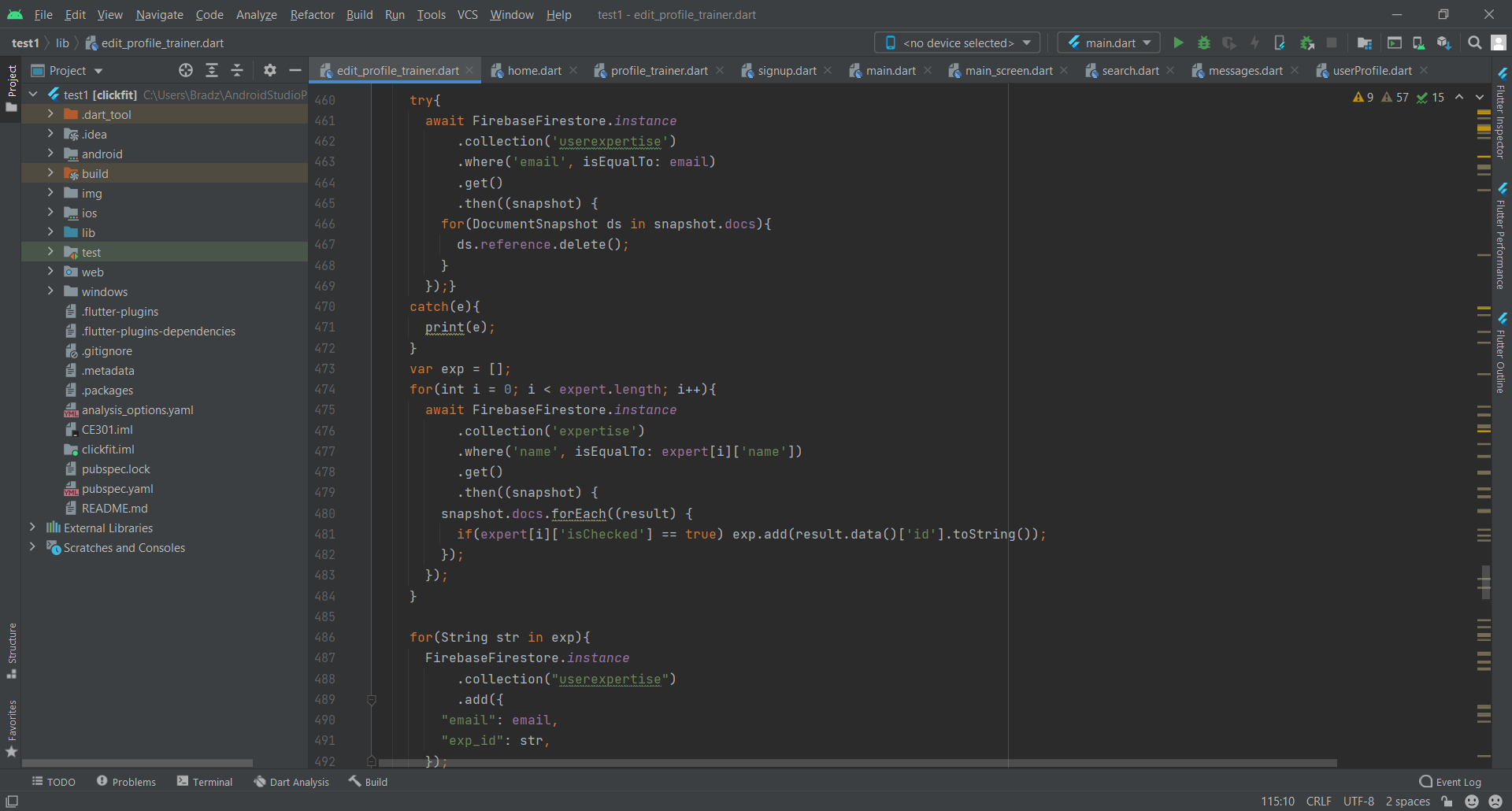


Figure 17

In the Firebase database there are two databases, first of which is the database holding the expertise name and an ID and the other named “userexpertise” which is a link between the users and expertise collections. Essentially, this emulates a realtional database although foreign keys cannot be set in Firebase. Due to the way I have configured the collections, in order to update the user’s expertise I remove all records of the current user in the “userexpertise” collection. This saves the time in checking it the user already has the expertise already saved. There is a map that contains all the expertise with the name of the expertise as a key and a boolean value describing if the checkbox has been checked or not. A for loop is used to go through the map (Fig.17) and if the checkbox is checked the ID of the expertise is added to a list and then this list is added to the “userexpertise” collection alongisde the email of the user.

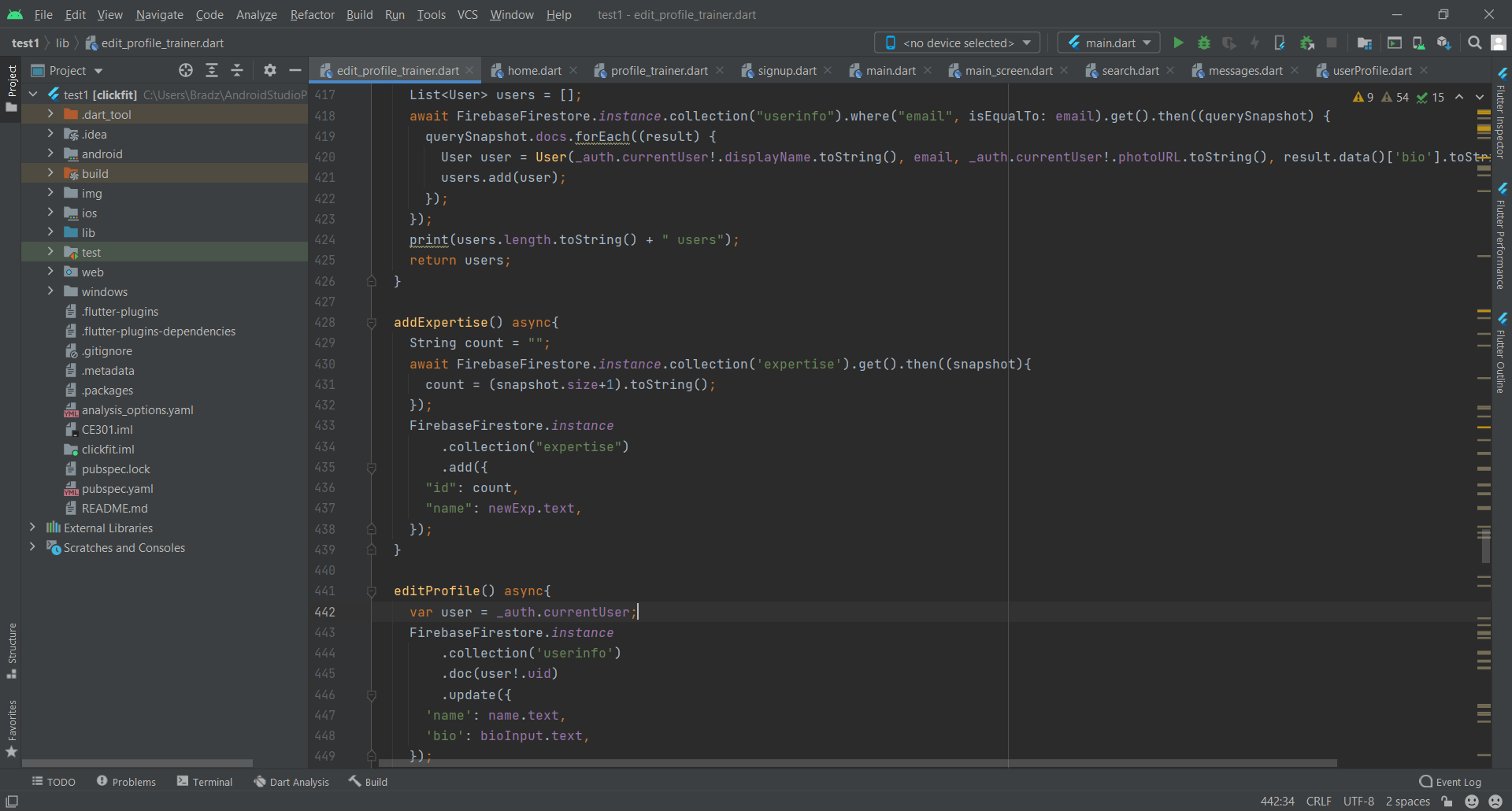


Figure 18

Trainers choose from a selection of expertise that are provided however, this trainer may be in a certain field that is not displayed in the app. For this reason, there is a functionality to allow the user to add their own choice of expertise to the collection (Fig.18). This works by taking the text input and adding that alongside the ID to the collection. The ID is a numerical value that corresponds to each record and cannot be a duplicate, to ensure that there are no duplicates the IDs correspond to the order in which the record appears in the collection. The number of documents in the collection are counted and then added by one to be the ID of the new record.

The log out button is also situated in the profile page. When this is pressed, the user is logged out of the Firebase authentication database then transferred back to the login page (Fig.19). The reason why the user is logged out of authentication first is because the object will store the user’s data until it is instructed otherwise.

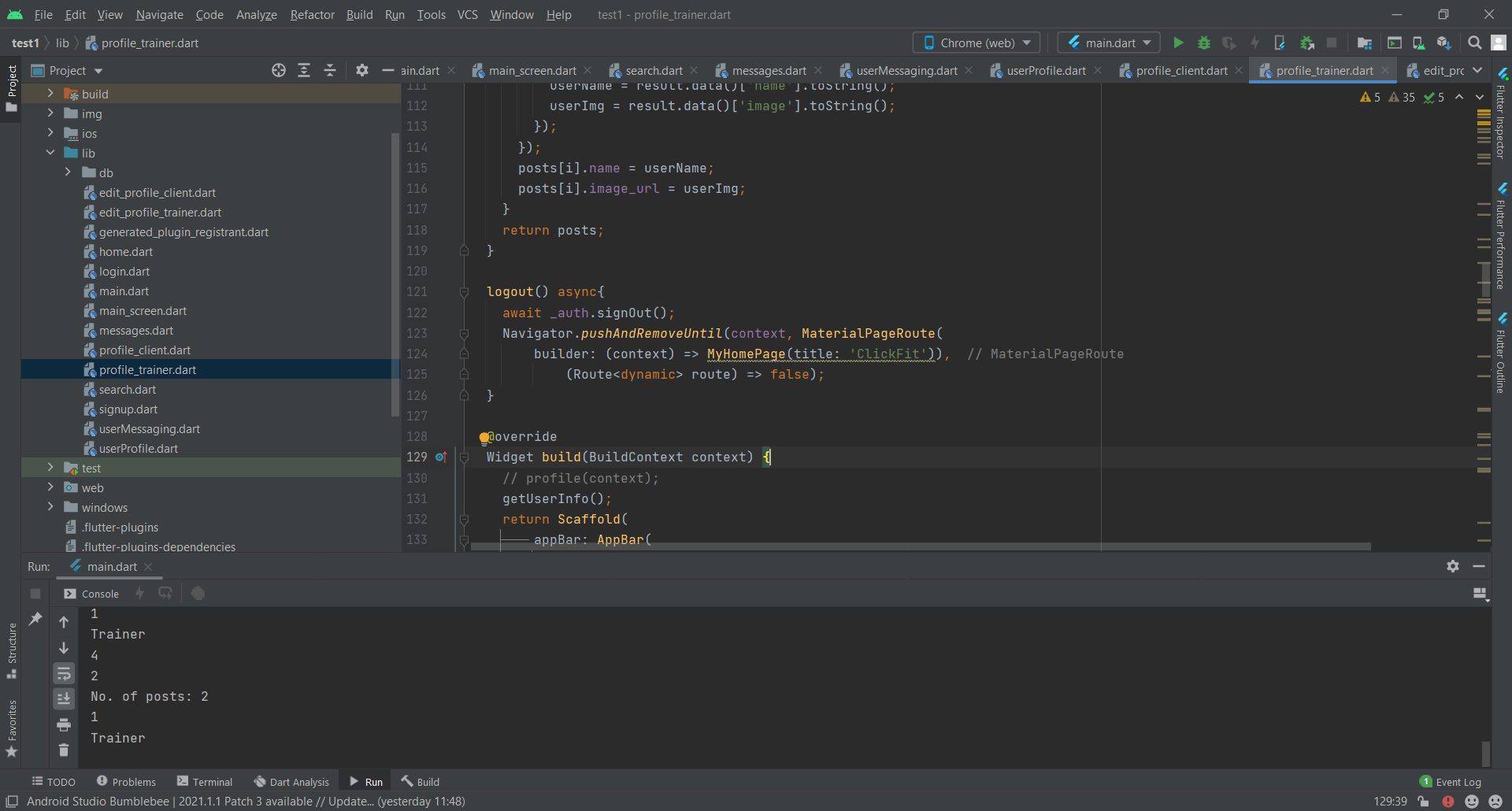
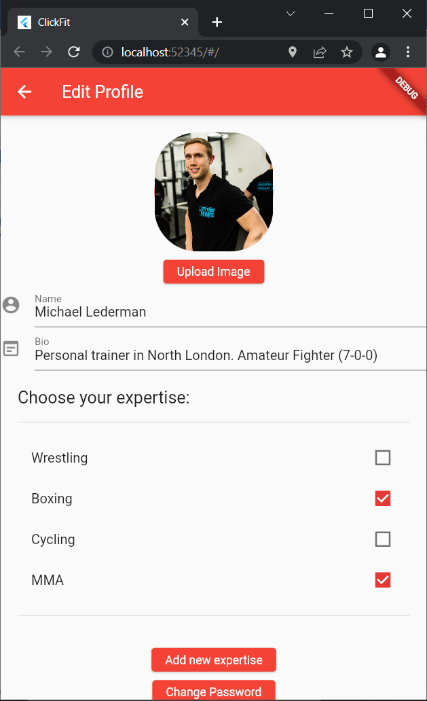


Figure 19

## 4.5 Testing

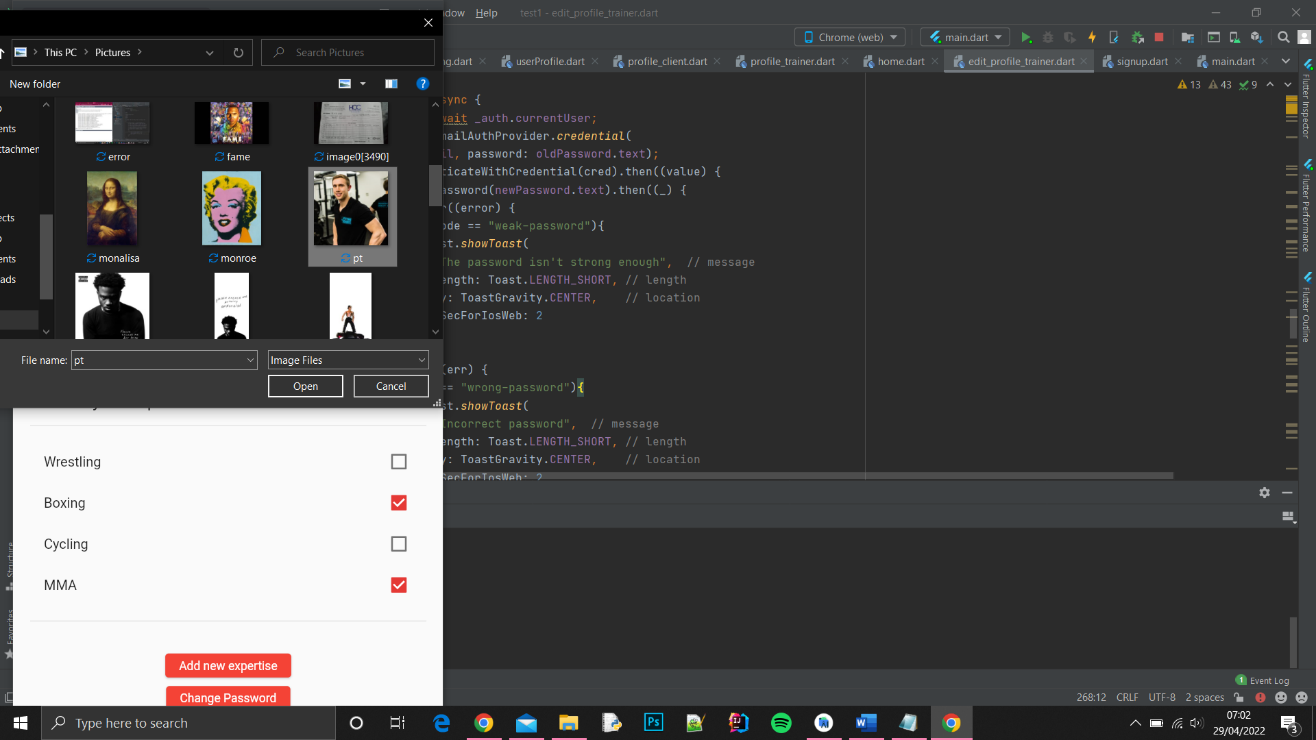
Table 2

|  |  |
| --- | --- |
| **Test** | **Outcome** |
| Add image to profile | Pass. See Fig. 20 & 21. Choose image button is pressed and an image is chosen from local storage, the profile is then saved, and the image is uploaded to the database. |
| Change profile name | Pass. See Fig. 22 & 23. The name field is changed in the profile editor and the profile is saved. Name field is then changed in Firestore database. |
| Change profile password | Pass. See Fig. 24. Change password popup is initiated, and the current password is inputted as well as the desired new password and the changes are saved. Password change is reflected in database. |
| Change trainer’s expertise | Pass. See Fig. 25 & 26. Tick boxes for trainer expertise are shown in profile editor and when ticked, changes are reflected in profile page. |



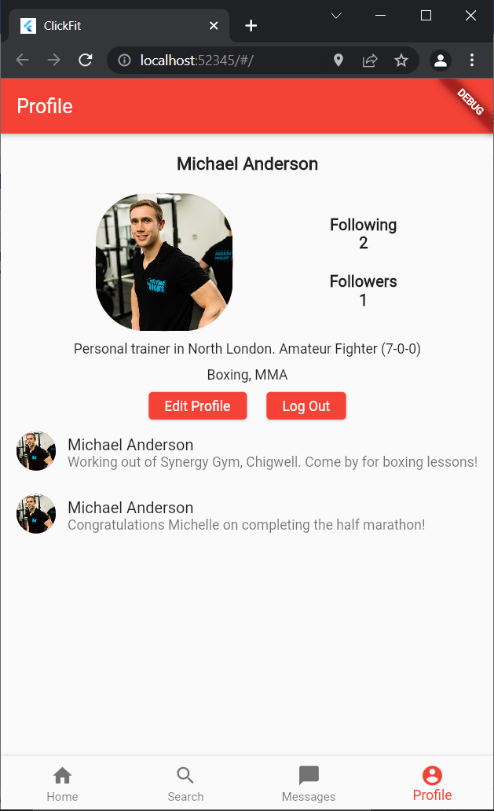
**Add image to profile part 2**

Figure 21



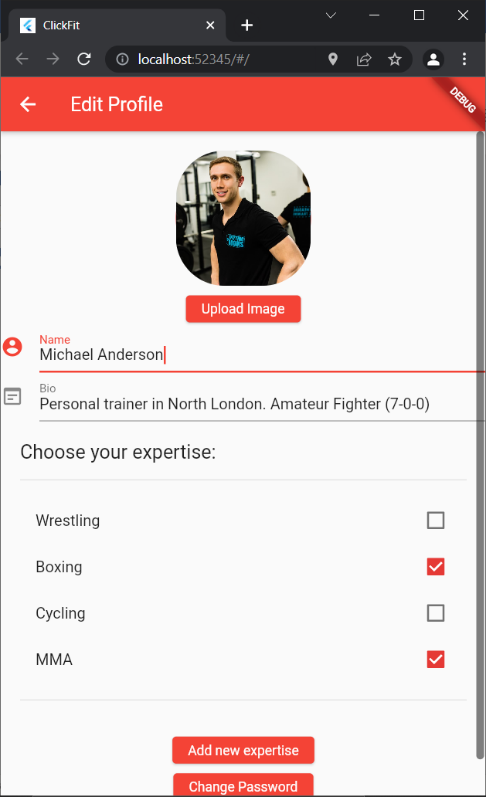
**Add image to profile part 1**

Figure 20



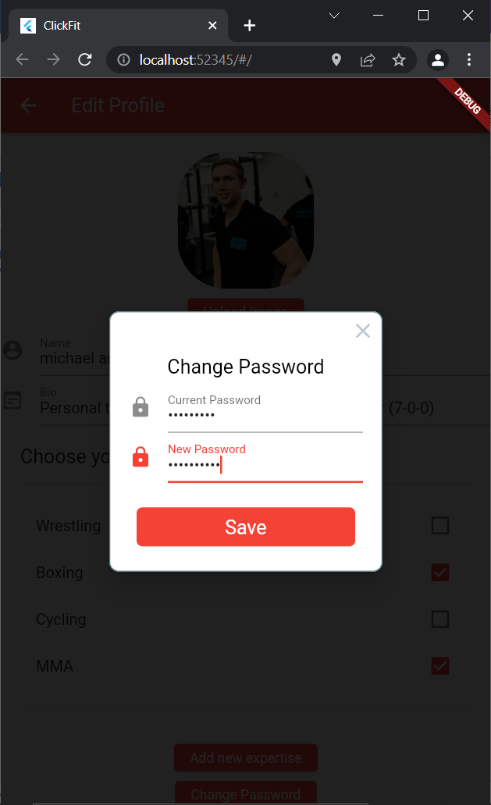
**Change profile name part 2**

Figure 23



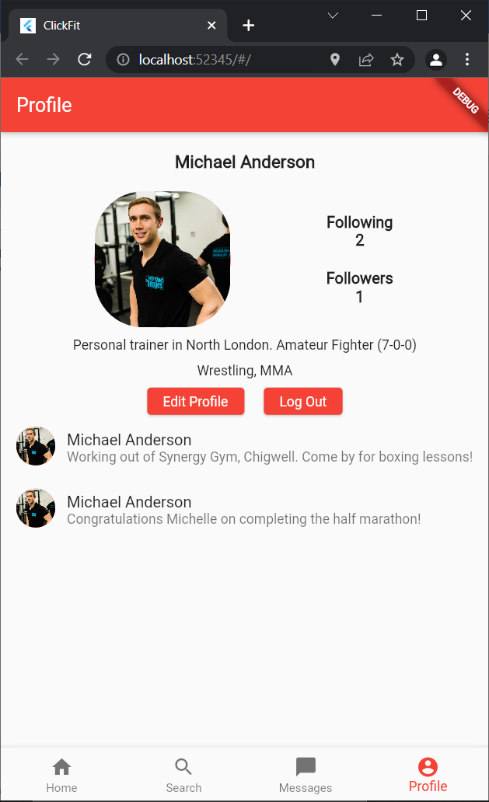
**Change profile name part 1**

Figure 22



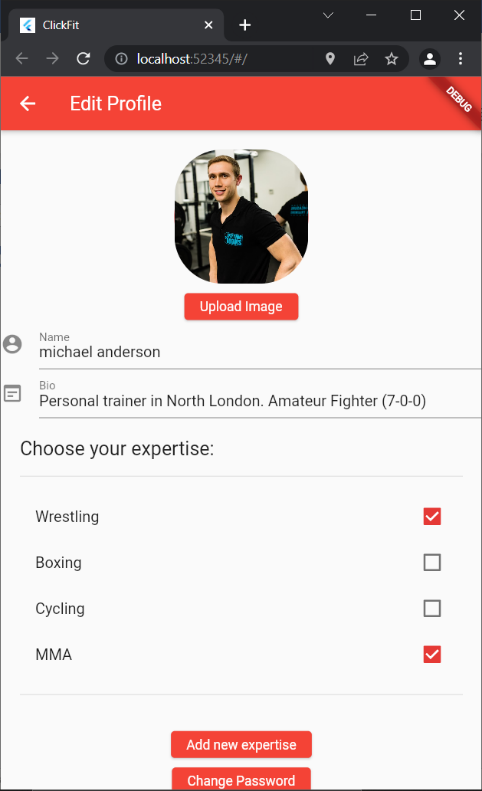
**Change profile password**

Figure 24



**Change trainer’s expertise part 2**

Figure 26



**Change trainer’s expertise part 1**

Figure 25

## 4.6 Conclusion

All tasks that were set out for this sprint were successfully executed. Originally, the plan was to save the image as a URL in the database but to do so the images would have to be saved online prior but in the initial stage of development of my app, I was running the app locally. To overcome this issue, I converted the uploaded images into bytes and saved the bytes into the database. To display the image, I would then load the bytes back into the app and then convert them back to an image. The problem with this, however, is that I ran into issues with images that were high definition as they would create a very large number of bytes which takes a considerable amount of storage. Using firebase storage, I was able to upload the image file to the database and save the URL reference to the user in the collection. This remedies the issue with the high definition images which take a larger amount of space. Another functionality I had to reconsider was the expertise for the trainer. Initially, I intended to implement this with a text field input however, with this the user would be able to input anything in and there would be no way to verify if it is a valid area of expertise. To combat this, I implemented checkboxes which include pre-existing common options for the user to choose with the ability to add new expertise which are not included. The benefit of this is that the expertise is located in a single collection and the goal is to in future have moderation of the expertise and they will be verified prior to them being visible to all users. With the profiles fully functioning, it naturally leads to the next stage of development which focuses on user posting. This will be implemented in sprint 3.

# 5.0 Sprint 3 - Feed/Posting

## 5.1 Introduction

This sprint is focused on allowing the user to create posts that are able to be viewed on the user’s home page. Both the user’s posts and the other people they’re following will appear on the home page in a twitter style feed in chronological order, allowing the user to scroll through relevant posts on their timeline.

## 5.2 Requirements

**Post to feed**

**Description:** User is able to post to home feed

**Summary:** User is able to write posts which are uploaded to database that appear on home feed of users and people they are following in chronological order.

**Traverse pages using navigation bar**

**Description:** User is able to access different pages using navigation bar found at the bottom of the screen

**Summary:** Navigation bar is found at the bottom of the screen containing interactive icons for the home page, user search page, messaging page and profile page. Clicking these icons allows quick access to these pages.

**View followers’ posts**

**Description:** User is able to view posts made by people they have followed

**Summary:** Posts made by people that the user follows appear on the user’s feed in a twitter style timeline in chronological order.

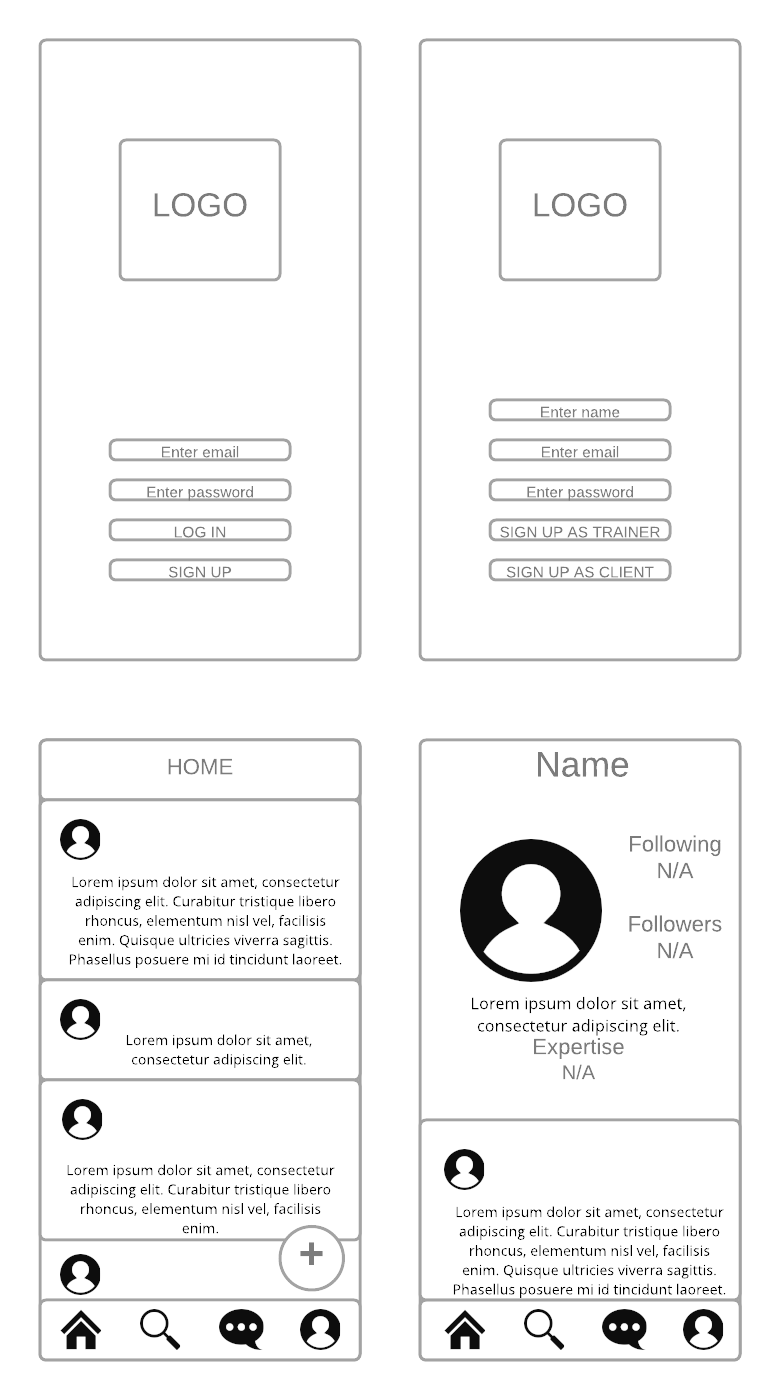
**Get user’s location**

**Description:** User is able to view the location of their trainer’s

**Summary:** Posts made by people that the user follows appear on the user’s feed in a twitter style timeline in chronological order.

## 5.3 Design

### 5.3.1 Home Page Design



## 5.4 Development

The home page displays posts belonging to themselves and the users they follow. This includes the user’s name, profile image and the text of their post. There is also a button which allows the user to create their own posts which will appear on the feed which is displayed in chronological order. The posts made by the user will also be visible on their own profile.

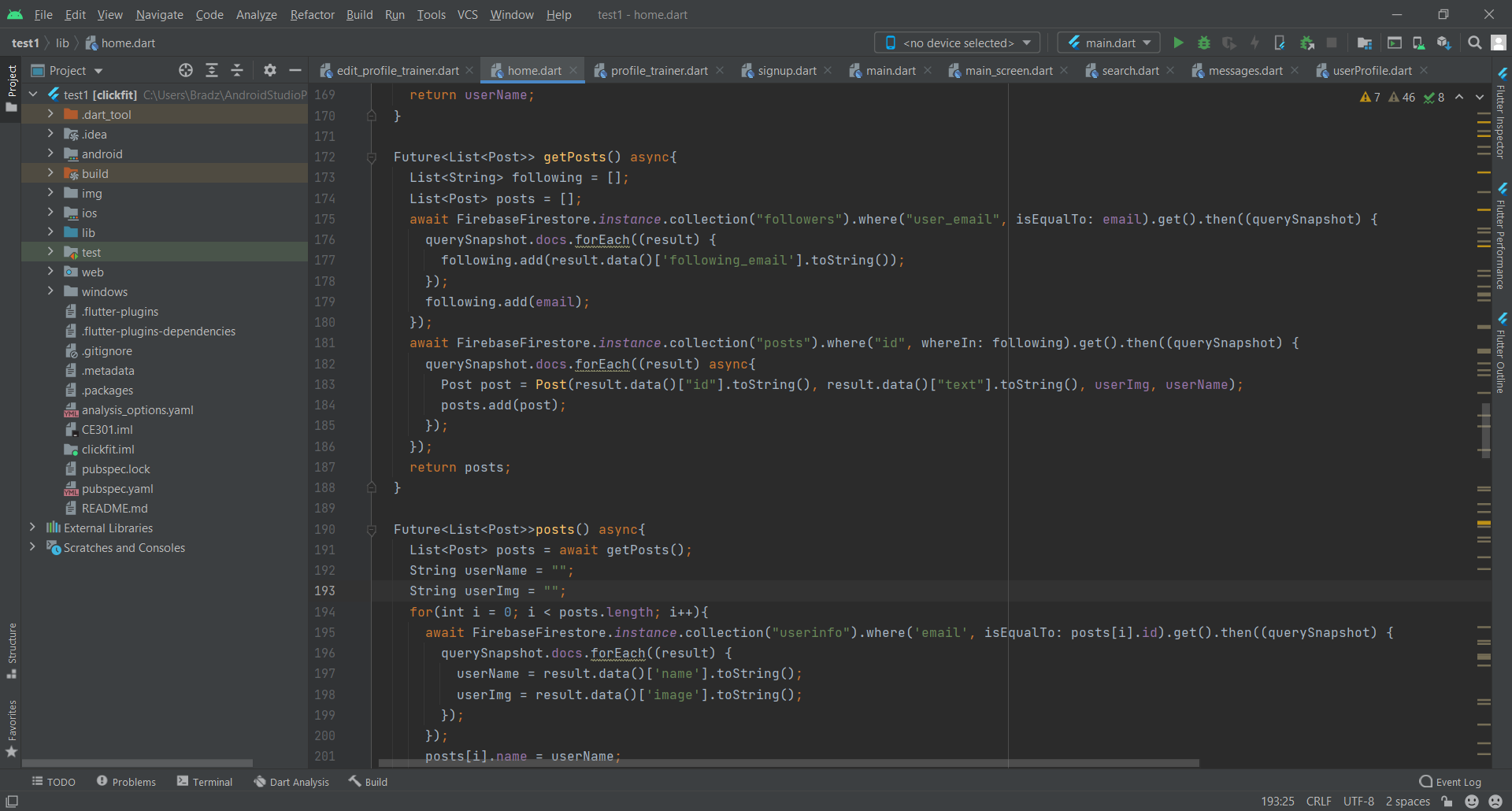


Figure 27

To get the posts from people that the user follows, the “followers” collection is first queried (Fig.27), finding all records where the “user\_email” field is the same as the current user’s email. The “following\_emails” are then added to a list and then the current user is also added to this list. This list is then used to search the posts collection, returning the documents that are made by the user or the users they are following. The post object that is created holds the email, post text, profile image of the user and profile name of the user.

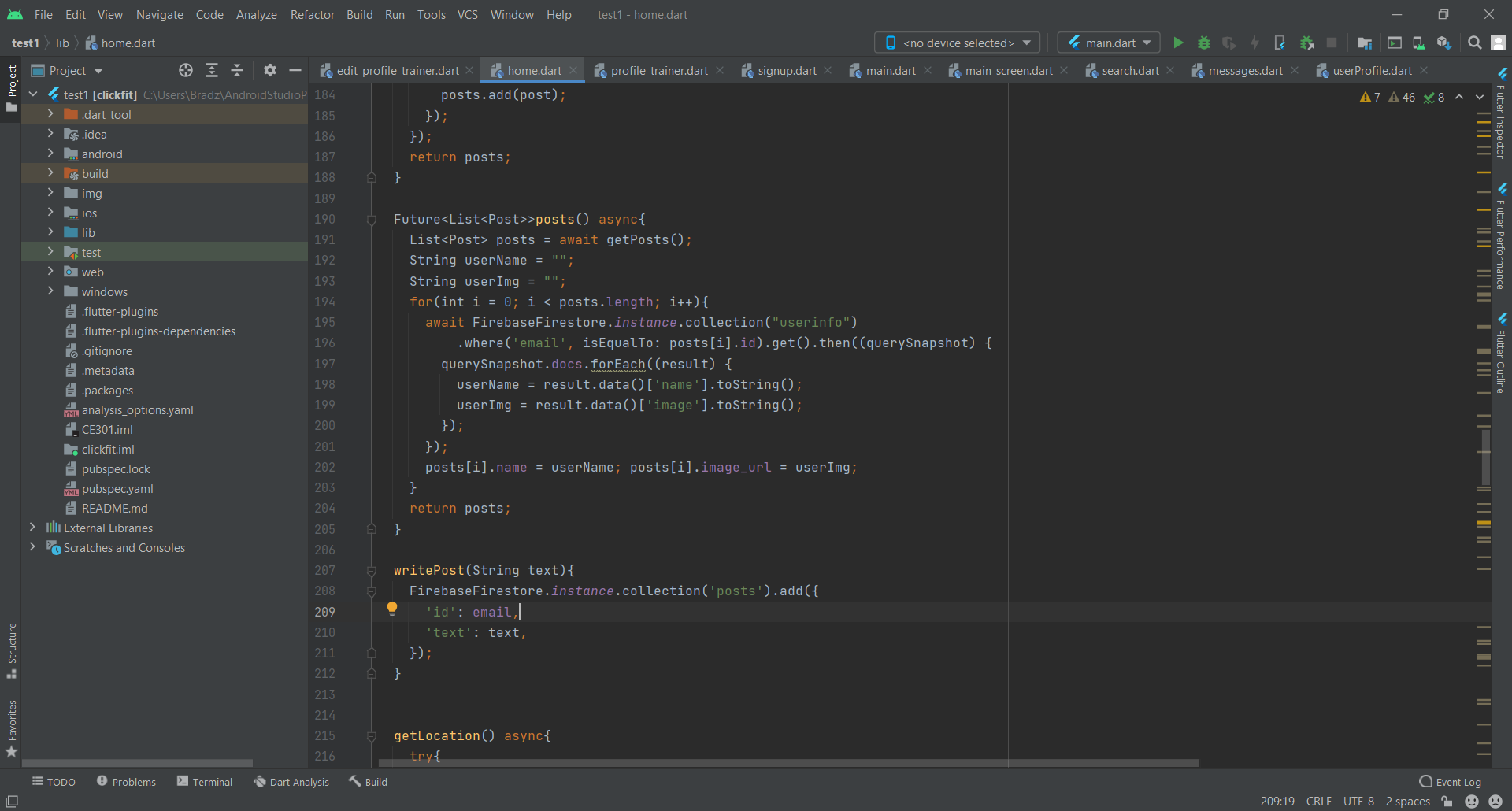


Figure 28

To get the user’s profile image and name, the userinfo has to be searched. The get posts method returns a list of post objects, I loop through these objects and search the “userinfo” collection with the email. The names and profile images are then extracted and the post objects are updated (Fig.28). The list of posts is then returned to a future builder which creates the widgets for the posts and then they’re displayed on the home page.

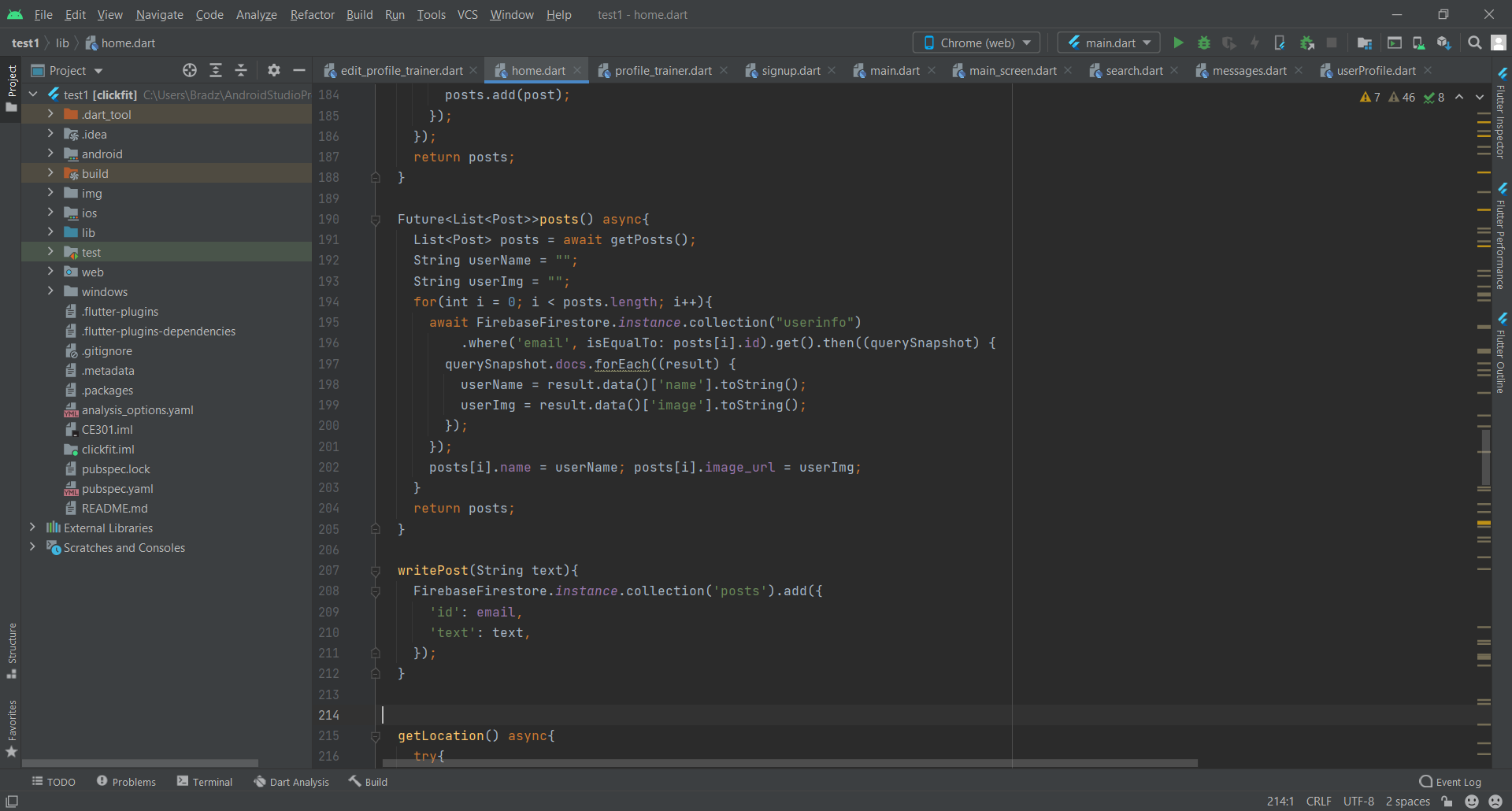


Figure 29

The home page also gives an option for the user to write their own post (Fig.29), when the button in the bottom right of the screen is pressed a popup is triggered. It provides a text field input where the user can write their post which is added to the database to then appear on the feed and on their profile. The post is added to the “posts” collection with the user’s email as the ID field and the text input as the text field.

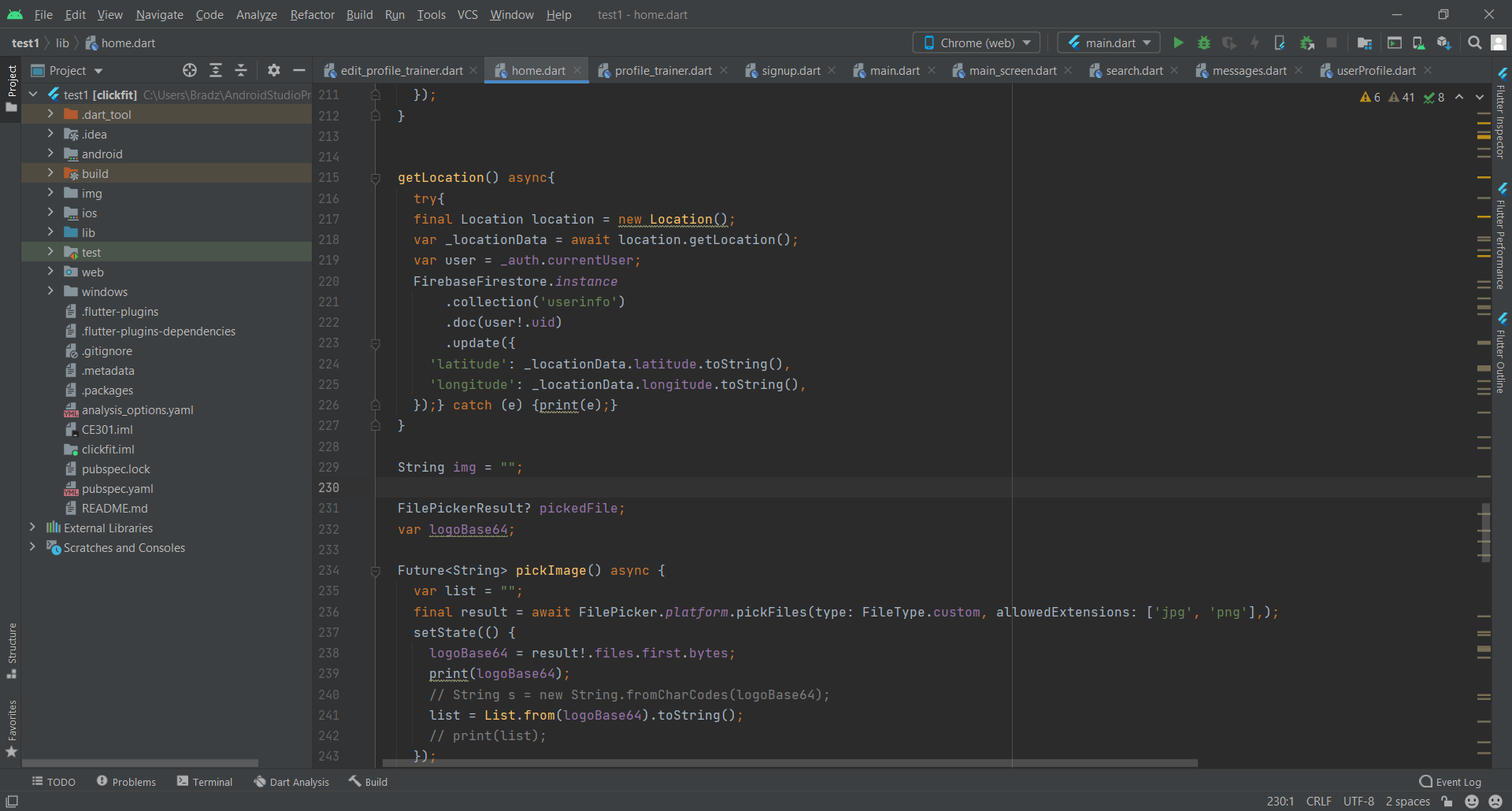


Figure 30

An important function in the app is the location services, this gets the current location of the user (Fig.30)so that it can be compared with the location of other users to calculate the distance between them. A new location object is created and then the get location method is used which gets the latitude and longitude coordinates that correspond to the current location, this also requests permission from the user to use location services. Given that the access is granted, the latitude and longitude is converted to string and is used to update the latitude and longitude values of the user’s record in the “userinfo” collection.

## 5.5 Testing

Table 3

|  |  |
| --- | --- |
| **Test** | **Outcome** |
| Search for users | Pass. See Fig.31. User is able to input text and filters user database to find other users. When user is clicked, the user’s profile is displayed. |
| View posts in timeline | Pass. See Fig.32. User is able to view posts from themselves and users they follow on the home page. |
| View users’ posts on their page | Pass See Fig.34. User’s posts are displayed on their own page and other users’ posts are visible when visiting their profiles. |
| Make posts | Pass. See Fig.33. Create post popup is displayed when button is pressed. User is able to input text and upload text to post on the feed. |
| Traverse pages with bottom navigation | Pass. See Fig.31 & 32. Bottom navigation bar is displayed on the bottom of the screen with 4 interactive icons. When the buttons are pressed the respective pages are displayed. |
| Display distance from user | Pass. See Fig.31. In the search page, if location services are enabled the distance between the user and other users appears in kilometres beneath their name. |
| Order by distance | Pass. See Fig.31. Distance is calculated using geocoordinates and then sorted in increasing order. |

**Feed / Timeline**

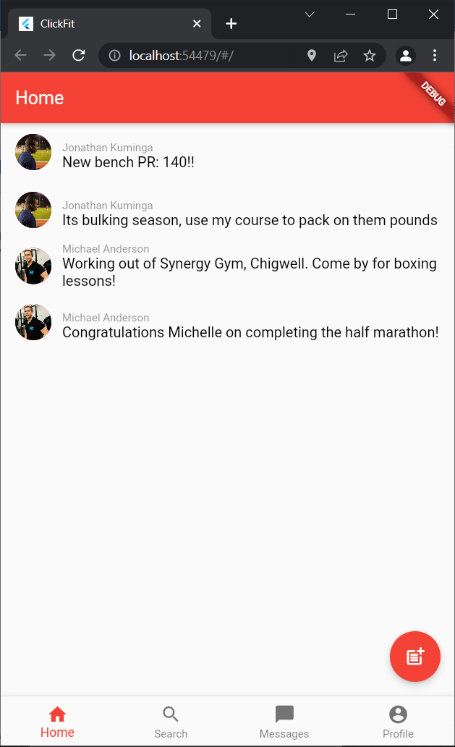


Figure 32

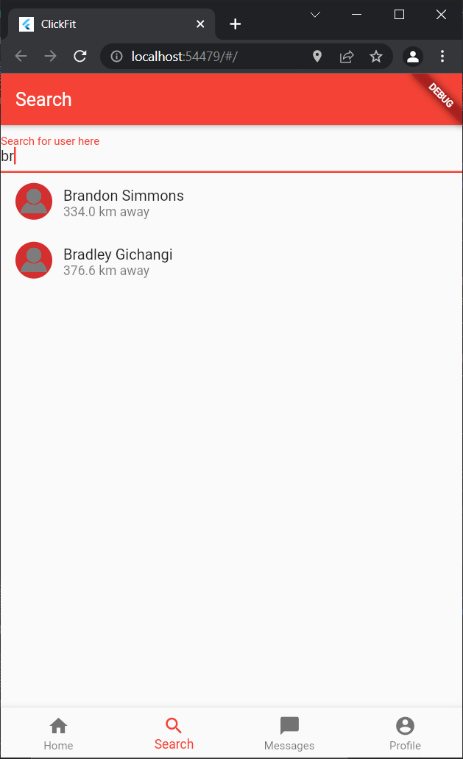
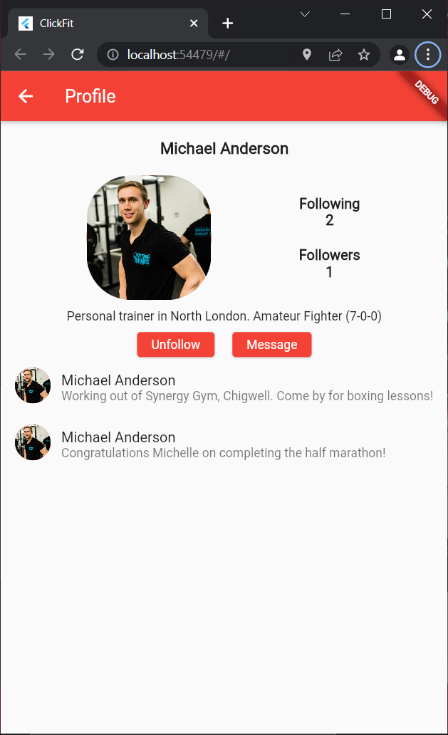


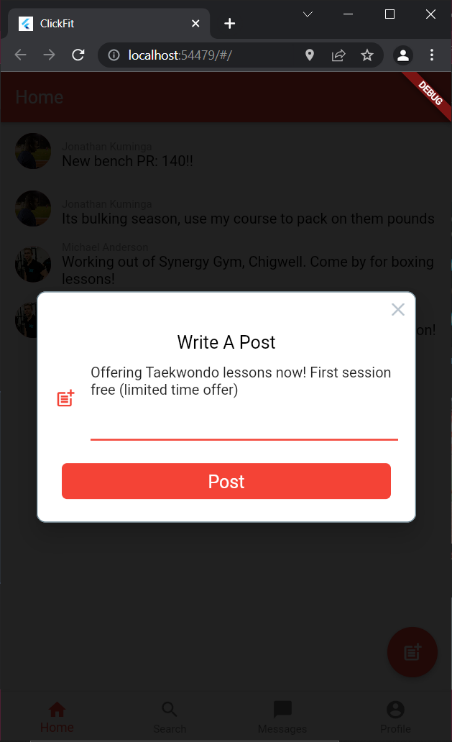
Figure 31

**User search**



**View posts on user’s page**

Figure 34



**Create post**

Figure 33

## 5.6 Conclusion

By comparing the finished application to the functional requirements, I can conclude that this sprint was an overall success. Although it was a success, there are things that could have been implemented to increase the functionality and better the user experience. Most notably by having the ability for the user to upload images to the feed. This is because images are more eye catching the users and would help with user interaction as trainers can post client progress images as promotion, flyers for sessions or other useful information.

# 6.0 Sprint 4 - Messaging

## 6.1 Introduction

This sprint is focused on creating an interface to allow instant messaging between users within the app. The messages are to be easily accessed from the messaging page that can be found using the bottom navigation bar and conversations can be started from visiting the profile pages of users using the search function. This is essential as it allows the trainers to communicate with clients so that they can liaise, organise sessions and etc.

## 6.2 Requirements

**Message notifications**

**Description:** User is notified when a message is received

**Summary:** Notifications are provided whenever the user receives a message with the sender’s name and brief text.

**Messaging**

**Description:** User is able to message another user

**Summary:** Using the messaging service in the app, a user is able to have instant message with another user.

**Permanent storage**

**Description:** Messages made by users are permanently stored in the database

**Summary:** Messages are stored permanently in the database so that when the user logs out, they can still view messages made in their history.

## 6.3 Design

### Graphical user interface, text, application Description automatically generated6.3.1 Messaging Page Design

## 6.4 Development

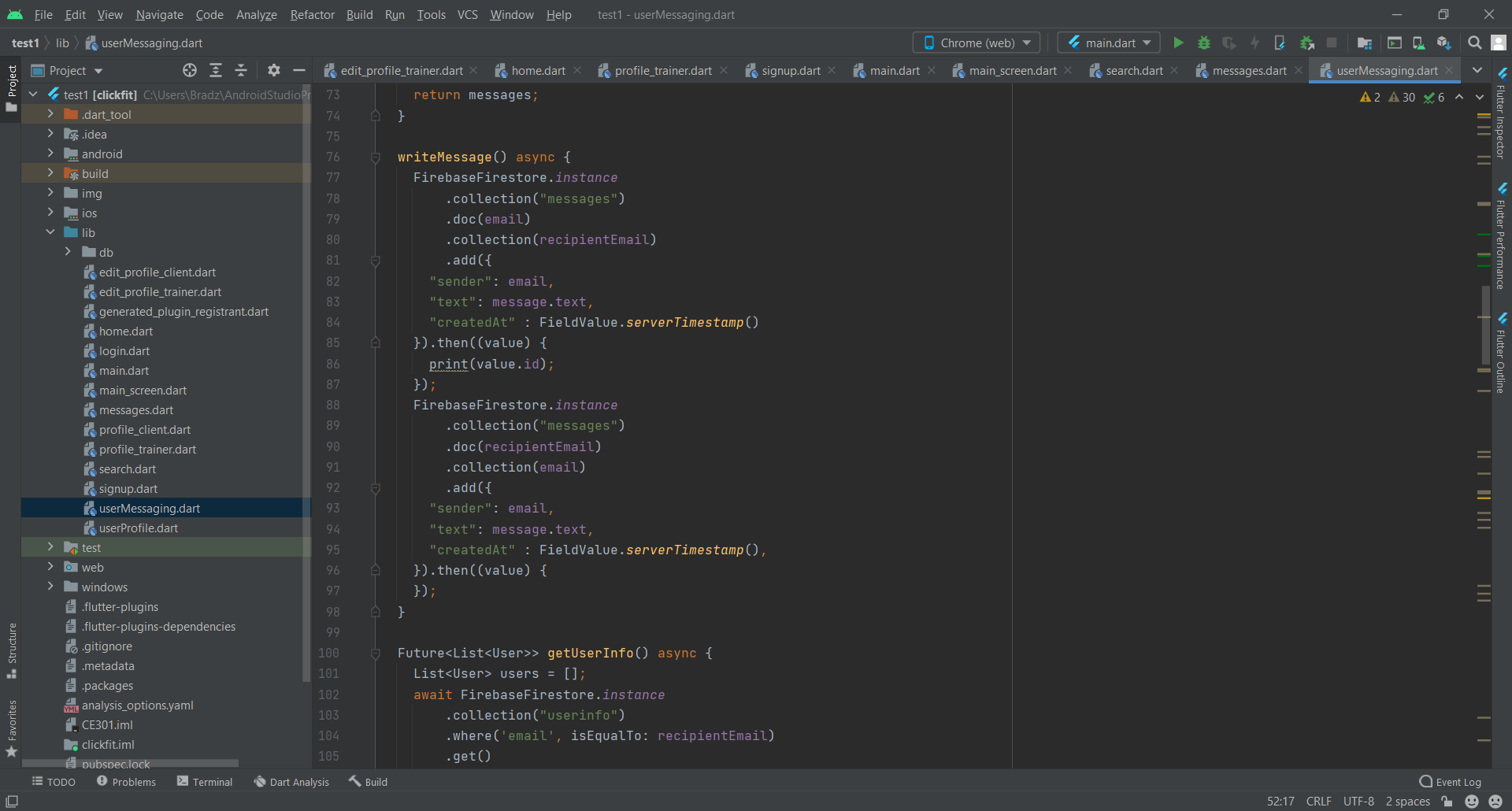
Messaging is an important aspect of the app as it is what provides the social aspect and therefore making the app a type of social media. Initially, the plan was to use the in-app messaging Firebase tool however, this tool is only available for mobile platforms that run Android or iOS. As I am running my app via the web, I devised an alternate method using the Firestore database.

Figure 34

The solution I created for this was have the documents in the collection correspond to the sender’s email and a subcollection that corresponds to the recipients email (Fig.34). All the messages between the users would then be held within this subcollection. However, this only allows the messages to be viewed by the sender. In order to have the data visible by both parties, the same process had to be executed with the intended recipient as the document and the sender as the subcollection. The messages contain three fields: sender, text and created at. Sender to know which user is sending the message in order to know where to position the text in the interface. Text in order to store the message sent by the user and finally, created at which stores the date and time of the creation of the message. This allows for the messages to be ordered chronologically.

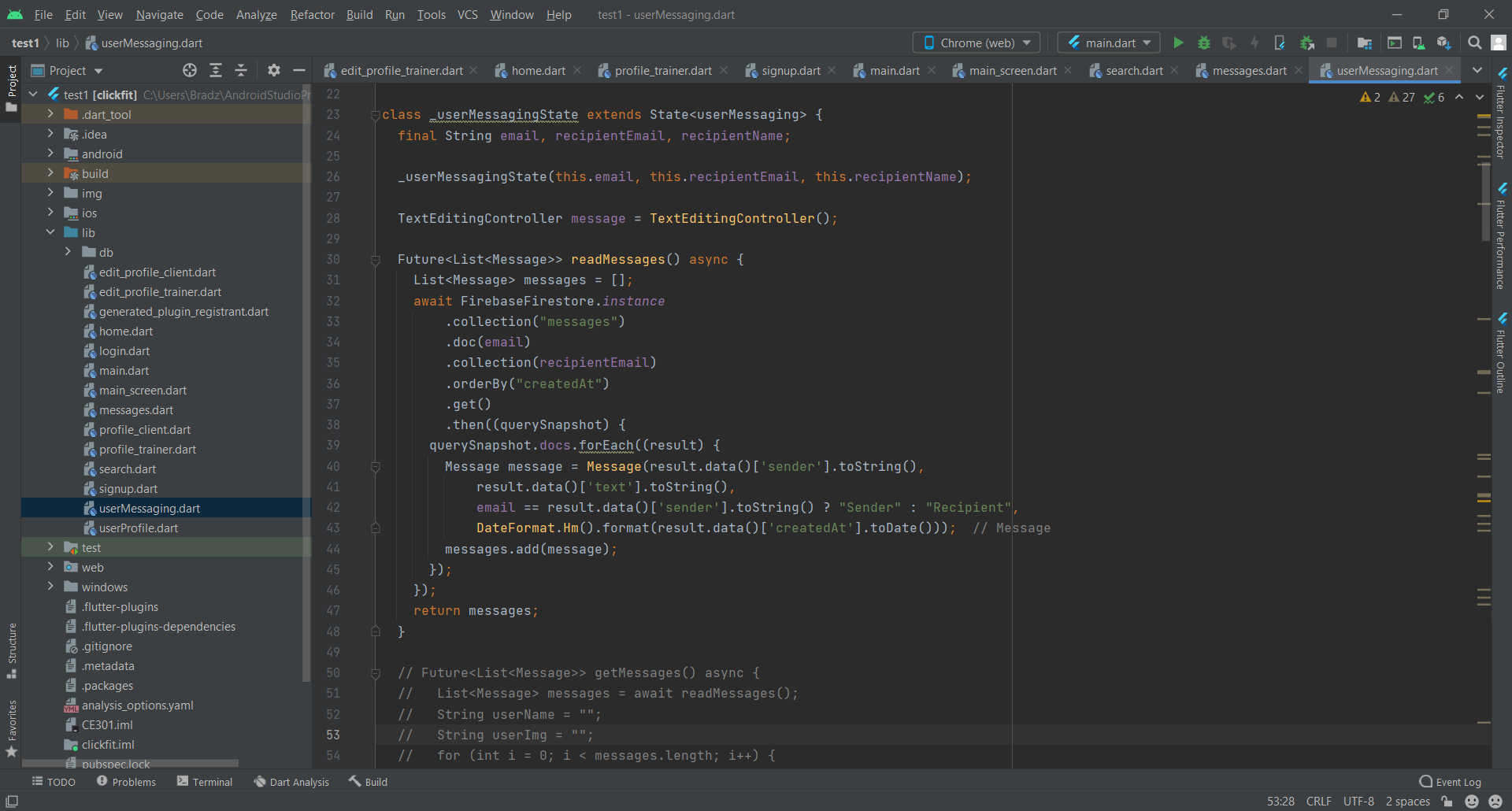


Figure 35

To read the messages the read messages method is called (Fig.35). This is down by getting the data from the messages collection, in the document that matches the user email with the subcollection that matches the recipients email. A new message object is instantiated for each instance, the sender, text and date time attributes are stored in the object. The current user’s email is compared to the sender attribute and if it is true the “userType” attribute is set to sender otherwise it is recipient. This is necessary as the sender and recipient text boxes are styled different in order to clearly distinguish between the two parties. Shown below in Fig.36.

Graphical user interface, application

Description automatically generated

Figure 36

## 6.5 Testing

Table 4

|  |  |
| --- | --- |
| **Test** | **Outcome** |
| Message other users | Pass. See Fig.37. Users are able to conversate with other users instantly using the messaging interface. |
| Message notifications displayed | Fail. Unable to display notification for new messages. |
| Access messaging through user profile | Pass. See Fig.38. By visiting a user’s profile and using the “message” button a pop up is triggered to initiate a conversation which takes the user to the messaging interface. |
| Preview messages in messages page | Pass. See Fig.39. In the messages page, the preview of the conversation is displayed under the recipient’s name. |

**Messaging from user profile**

**Messaging between users**

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Figure 38

Figure 37

Graphical user interface, text, application

Description automatically generated

**Previewing messages in message page**

Figure 39

## 6.6 Conclusion

The primary objective for this sprint has been successfully completed but not to the standard I had initially anticipated as I was unable to display notifications for new messages. Firebase offers a service named in-app messaging however, this is only usable in android or iOS systems and my app is currently a web application which limits the tools that can be used. By creating a collection for messages in a real time database the messages between users can be saved and the changes will be reflected in the app. However, you are unable to create an empty collection in Firestore which would be ideal for when someone is trying to start a conversation with another user by visiting their page. This way they can access the messaging interface that has no prior messages, to combat this I created a popup that appears when the message button is pressed and the user is prompted to input their first message which allows for the collection to be created and the user can access the messaging interface.

# 7.0 Project Planning

Initially, to plan my development I had created a work breakdown structure. This described how I was going to develop my app from initiation to closeout, starting with detailing the requirements and ending with the report. The work breakdown structure is a vaguer version of the development plan and describes it at a macro level. I am using agile development and created several sprints detailing major components of my app, these components are login/signup, profile page/following, feed/post and messaging. For each sprint there are is an introduction, a list of functional requirements, abstract interface designs, development, testing and a conclusion. The order in which I developed these sections was precise and deliberate in order to have a smooth workflow throughout my project. Each sprint requires the successful completion of the sprints prior to it as it is being built from the bottom up. This method proved to be very useful as it made it very easy to know what I had to work on next, by referring to the functional requirements, I could identify modules that needed to be implemented in my app. Furthermore, by setting out requirements prior to development I could identify possible issues that I could come across in development. For example, when creating the login and signup page the data must be saved in the database. This raises a security issue if the database is infiltrated and so the sensitive data must be encrypted, and this issue can be solved before development starts.

Jira is a software application [16] that is widely used in professional development. It used for the tracking and management of projects and provides several useful tools for this purpose. It is specifically used for agile development which is ideal for me as this is the type of development I am utilizing in my project.

Kanban board is a tool that Jira provides which optimizes workflow [17], allows for the easy identification of bottlenecks and promotes continuous improvement. By having tasks as blocks in different stages, you can easily keep track of tasks and their progress during development. There are also visual metrics such as the cumulative flow diagram which displays a graph which can be used to identify any drops in productivity and possible points of improvement.

To keep track of development, I used a Kanban board and added the sprints as epics and each task within the sprint that needed to be executed would be stories within this epic. If I came across any issues during the development, they would be inputted into the Kanban board as bugs to be fixed. There are 3 states in which these tasks could be in which are the backlog, in progress and done and depending on which stage of development each task was in, they would be placed in their respective state.

From my experience, I believe there are somethings I could have done differently in my planning which would have improved my performance and therefore increase the quality of the product. The main issue was time scale. Although I meticulously planned out each stage of my project, detailing what requirements needed to be met and methods I could use to accomplish these tasks, I did not set a strict time scale for these tasks to be completed. By doing this, I would have been able to work more efficiently, therefore allowing more time to find and fix bugs and implement possible additional features within the app.

There was also issues with the horizon virtual machine which meant that I was unable to update Jira remotely. This caused a decrease in productivity, so the cumulative flow diagram is stagnant through a section in April.

# 8.0 Conclusion

In my project there were 4 primary objectives that I intended to implement in order to deem it a success. These were the login and signup page, profile pages and a following aspect, a feed on the home page and the ability to instant message other users within the app. Each of these were separate sprints within my development.

The first objective is important as this allows the user to create an account which is the basis for my app. By creating the account, it allows for the user to be searched for, to accept and send messages and to edit their own profile information. The login page was identical to the original design I had created initially however, the signup page design was almost identical apart from the radio buttons. These buttons were intended to be used to allow the user to decide whether they want to create a trainer or client account, to remedy this I created two separate sign-up buttons for the trainer and client and both pages worked as intended. The buttons occupy the same amount of real estate on the page as the radio buttons would have so I did not have to stray far from the original design. This section was completed to the quality I intended initially.

The second objective is to create profile pages, allowing the user to follow other users on the app. This also includes the user search in order to find other users. The profile page was created with close attention to the original design; therefore, it is identical to the initial designs. All other designs apart from the first sprint are abstract, including example text on where information is intended to be positioned on the screen. The search page is similar to the design however, in the original design the user search occurs in a drop down list from the text field, however in the finished product there is a list of users and the users are filtered depending on the input as originally envisioned. To improve this section, I could have implemented more features in the original design as I was using this as my guide, making a detailed design of the interface would possibly allow for a sleeker professional look. The downside of this is that the process would be time consuming and possibly reduce time of development.

The third objective is twitter style feed on the home page, displaying posts made by the user and other users that they follow. These posts also appear on the profile page of the user beneath the user information. The design was replicated in the finished product with a list of posts appearing on the home page and a button in the bottom right to allow the user to add a post of their own. The only difference is the icon on this button, in the design the icon is a “+” sign however, in the app the icon is an icon that more accurately illustrates the purpose of the button. To improve this section, I would integrate the ability for users to interact with posts on their feed such as adding likes and comments. This would allow for a greater social aspect in the app. I would also include the ability to upload images with the post, images are more eye catching which would allow for users to draw other users to their profile easier.

The final objective is the messaging aspect, this allows the users to conversate between each other. I had designed the page to view the different conversations the user currently has and was able to replicate the design in the app. However, I did not create a design for the messaging interface so there wasn’t a guide on how the interface was supposed to look during development. The interface is inspired by these WhatsApp messenger and this was achieved by a flutter plugin, using this allowed the interface to have a more polished look after attempting multiple designs.

A large reason to why I chose the software that I used for the development of my project was that it allowed for cross-platform usage while only using a single code base. This is something I unfortunately was unable to take advantage of due to hardware restrictions, the system I developed the program on is not very powerful and therefore, could not support running emulators for the different mobile operating systems.

One of the main things I would improve in future given that I had the hardware capable of handling it would be to implement Google Maps instead of the basic location services which are currently implemented as it provides a number of useful tools, most notably GPS directions to a destination and the ability to visually see the location on the map rather than just the distance from the user.

Using biometric data within the app in order to track the user’s fitness and health would also elevate the apps capabilities and could be integrated in future. Storing this biometric and genetic data is also covered by GDPR [18] in the UK so there would have to be a terms and conditions clause in order to be within the regulations.

Another functionality I would implement in future is a calendar for clients and trainers to organize and schedule sessions. By implementing these functionalities within the app, it incentivizes the user to stay within the app rather than using applications outside of the app. Additionally, having a type of discovery page whereby trainers that the user may not follow can appear to allow the user to branch out and try activities they possibly would not have considered while also providing trainers with more exposure.

# References

[1]"Download Android Studio and SDK tools  |  Android Developers", *Android Developers*, 2022. [Online]. Available: https://developer.android.com/studio. [Accessed: 11- Feb- 2022].

[2]"Flutter development: advantages and disadvantages | Sannacode", *Sannacode*, 2022. [Online]. Available: https://sannacode.com/blog/advantages-and-disadvantages-using-flutter. [Accessed: 28- Jan- 2022].

[3]"Flutter vs React Native: Which is Best for Your App? [2022]", *DOIT Software*, 2022. [Online]. Available: https://doit.software/blog/flutter-vs-react-native#screen14. [Accessed: 28- Jan- 2022].

[4]"geolocator | Flutter Package", *Dart packages*, 2022. [Online]. Available: https://pub.dev/packages/geolocator. [Accessed: 04- Feb- 2022].

[5]"How To Get a User's Location with the Geolocator Plugin in Flutter | DigitalOcean", *Digitalocean.com*, 2022. [Online]. Available: https://www.digitalocean.com/community/tutorials/flutter-geolocator-plugin. [Accessed: 04- Feb- 2022].

[6]"React Native vs. Flutter: Which to Choose?", *Relevant Software*, 2022. [Online]. Available: https://relevant.software/blog/react-native-vs-flutter-which-to-choose-for-cross-platform-development/. [Accessed: 28- Jan- 2022].

[7]"Why Firebase is the Best as a Mobile App Development Backend?", *Tristatetechnology.com*, 2022. [Online]. Available: https://www.tristatetechnology.com/blog/firebase-backend-mobile-app/. [Accessed: 28- Jan- 2022].

[8]U. Technologies, "Unity", *Unity*, 2022. [Online]. Available: https://unity.com/. [Accessed: 11- Feb- 2022].

[9]"What gives Flutter the edge over its other alternatives?", *DEV Community*, 2022. [Online]. Available: https://dev.to/geekyprawins/what-gives-flutter-the-edge-over-its-other-alternatives-22h7. [Accessed: 04- Feb- 2022].

[10]"What is Microsoft Azure and How Does It Work?", *SearchCloudComputing*, 2022. [Online]. Available: https://www.techtarget.com/searchcloudcomputing/definition/Windows-Azure. [Accessed: 11- Feb- 2022].

[11]"Why Android Studio is Awesome ?", *Medium*, 2022. [Online]. Available: https://medium.com/@agicent/why-android-studio-is-awesome-c606c94366e6. [Accessed: 11- Feb- 2022].

[12]"The Ultimate Comparison: Firebase vs Other Platforms", *OS-System*, 2022. [Online]. Available: https://os-system.com/blog/comparison-firebase-with-other-platforms/#:~:text=Parse%20vs%20Firebase,its%20users%20to%20self%2Dhost.&text=Firebase%20provides%20central%20data%20management,s3%20bucket%20for%20file%20storage. [Accessed: 11- Feb- 2022].

[13]2022. [Online]. Available: https://blog.logrocket.com/geolocation-geocoding-flutter/. [Accessed: 04- Feb- 2022].

[14]"Google Maps Platform FAQ  |  Google Developers", *Google Developers*, 2022. [Online]. Available: https://developers.google.com/maps/faq#:~:text=The%20Google%20Maps%20Platform%20is,retrieve%20data%20from%20Google%20Maps. [Accessed: 11- Mar- 2022].

[15]"Google Maps Platform Documentation  |  Google Developers", *Google Developers*, 2022. [Online]. Available: https://developers.google.com/maps/documentation. [Accessed: 11- Mar- 2022].

[16]"Jira", *Productplan.com*, 2022. [Online]. Available: https://www.productplan.com/glossary/jira/. [Accessed: 23- Apr- 2022].

[17]"Jira Kanban boards | Atlassian | Atlassian", *Atlassian*, 2022. [Online]. Available: https://www.atlassian.com/software/jira/features/kanban-boards. [Accessed: 23- Apr- 2022].

[18]"Biometric data and privacy laws (GDPR, CCPA/CPRA)", *Thales Group*, 2022. [Online]. Available: https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/biometric-data. [Accessed: 28- Apr- 2022].