

# EatFit

A project report submitted for  
**Android Apps Development Lab (Semester VII)**  
by

Ryan Dsilva (7999)  
Niharika Gogate (8004)  
Katoushka Gracias (8007)

Under the guidance of  
Prof. Dr. Nilesh Patil

(sign with date)



DEPARTMENT OF INFORMATION TECHNOLOGY  
Fr. Conceicao Rodrigues College of Engineering  
Bandra (W), Mumbai - 400050  
University of Mumbai

## **Approval Sheet**

### **Project Report Approval**

This project report entitled EatFit developed by Ryan Dsilva (7999), Niharika Gogate (8004), Katoushka Gracias (8007) is approved as project for Android Apps Development Lab in Fourth Year Engineering (Sem - VII), Information Technology.

Examiners

1. \_\_\_\_\_

2. \_\_\_\_\_

Date:

Place:

## **Abstract**

Smartphones and tablets are slowly but steadily changing the way we look after our health and fitness. Today, many high quality mobile apps are available for users and health professionals and cover the whole health care chain, i.e. information collection, prevention, diagnosis, treatment and monitoring. Our team has developed a mobile health and fitness app called EatFit. Eatfit is developed using the Flutter framework and using a PyTorch Deep Learning Model to identify the food from the images you supply through the device camera or gallery. Using this we can track the daily calorie intake of a person. Our application also suggests daily exercises and shows audio visual instructions for each exercise. There is also a feature for tracking the user's run and all the features are also provided through a chatbot interface that was made using DialogFlow. This application will help users keep track of their eating habits while also keeping them motivated to do regular exercise.

### **Keywords:**

fitness, healthcare, flutter, pytorch, exercise, calories

## Table Of Contents

Sr. No.	Topic	Page No.
1	Introduction	4
2	Widgets	6
3	Layouts	8
4	Intents	10
5	Activity	11
6	Firebase	14
7	Camera API	15
8	Location API	18
9	Generate APK File	19
10	Conclusion	20
11	Source Code	21
12	References	62

# **Chapter 1**

## **Introduction**

### **Introduction to Android:**

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc. It contains a Linux-based Operating System, middleware and key mobile applications. It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. It provides support for messaging services (SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Bluetooth, Wi-Fi etc.), media, handset layout etc.

### **Features of Android:**

- It is open-source.
- Anyone can customize the Android Platform.
- There are a lot of mobile applications that can be chosen by the consumer.
- It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

### **Introduction to Flutter:**

Flutter is an open-source mobile application development SDK primarily developed and sponsored by Google, used for developing applications for Android and iOS—as well as being the primary method of creating applications for the Google Fuchsia operating system. Flutter is written in C, C++, and Dart, and uses the Skia Graphics Engine. It offers a rich set of fully customizable widgets for building native interfaces, including the beautiful Material Design library and Cupertino (iOS-flavored) widgets, rich motion APIs, smooth natural scrolling, platform awareness, and hot reload—which helps to quickly build UIs without losing state on emulators, simulators, and any hardware for iOS and Android. Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms. Flutter's simplicity makes it a good candidate for fast development. Its customization capability and extendibility makes it even more powerful. With Flutter, developers have full control over the widgets and its layout.

Flutter framework offers the following features to developers –

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms.
- High performance application.



## Chapter 2

### Widgets

Widgets are the basic building blocks of a Flutter app's user interface. Each widget is an immutable declaration of part of the user interface. Unlike other frameworks that separate views, view controllers, layouts, and other properties, Flutter has a consistent, unified object model: the widget.

A widget can define:

- a structural element (like a button or menu)
  - a stylistic element (like a font or color scheme)
  - an aspect of layout (like padding)
- and so on.

Widgets form a hierarchy based on composition. Each widget nests inside, and inherits properties from its parent. There is no separate “application” object. Instead, the root widget serves this role. You can respond to events, like user interaction, by telling the framework to replace a widget in the hierarchy with another widget. The framework then compares the new and old widgets and efficiently updates the user interface.

Widgets are themselves often composed of many small, single-purpose widgets that combine to produce powerful effects. For example, Container, a commonly-used widget, is made up of several widgets responsible for layout, painting, positioning, and sizing. Specifically, Container is made up of LimitedBox, ConstrainedBox, Align, Padding, DecoratedBox, and Transform widgets. Rather than subclassing Container to produce a customized effect, you can compose these, and other, simple widgets in novel ways. You can also control the layout of a widget by composing it with other widgets. For example, to center a widget, you wrap it in a Center widget. There are widgets for padding, alignment, row, columns, and grids. These layout widgets do not have a visual representation of their own. Instead, their sole purpose is to control some aspect of another widget's layout. To understand why a widget renders in a certain way, it's often helpful to inspect the neighboring widgets.

In Flutter, widgets can be grouped into multiple categories based on their features, as listed below –

- Platform specific widgets
- Layout widgets
- State maintenance widgets
- Platform independent / Basic widgets

#### Platform Specific Widgets

Flutter has widgets specific to a particular platform - Android or iOS. Android specific widgets are designed in accordance with Material design guideline by Android OS. Android specific widgets are called Material widgets. iOS specific widgets are designed in accordance with Human Interface Guidelines by Apple and they are called as Cupertino widgets.

## Layout Widgets

In Flutter, a widget can be created by composing one or more widgets. To compose multiple widgets into a single widget, Flutter provides large number of widgets with layout feature. For example, the child widget can be centered using Center widget.

Some of the popular layout widgets are as follows –

- Container – A rectangular box decorated using BoxDecoration widgets with background, border and shadow.
- Center – Center its child widget.
- Row – Arrange its children in the horizontal direction.
- Column – Arrange its children in the vertical direction.
- Stack – Arrange one above the other.

## State Maintenance Widgets

In Flutter, all widgets are either derived from StatelessWidget or StatefulWidget. Widget derived from StatelessWidget does not have any state information but it may contain widget derived from StatefulWidget. The dynamic nature of the application is through interactive behavior of the widgets and the state changes during interaction. For example, tapping a counter button will increase / decrease the internal state of the counter by one and reactive nature of the Flutter widget will auto re-render the widget using new state information.

## Platform Independent / Basic Widgets

Flutter provides large number of basic widgets to create simple as well as complex user interface in a platform independent manner.

### Example Widget:

```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(this.title)),
      body: Center( child: Icon(Icons.email)),
    );
  }
}
```



## Chapter 3

### Layouts

The core of Flutter's layout mechanism is widgets. In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets. You create a layout by composing widgets to build more complex widgets.

One of the most common layout patterns is to arrange widgets vertically or horizontally. You can use a Row widget to arrange widgets horizontally, and a Column widget to arrange widgets vertically. To create a row or column in Flutter, you add a list of children widgets to a Row or Column widget. In turn, each child can itself be a row or column, and so on. Row and Column are basic primitive widgets for horizontal and vertical layouts—these low-level widgets allow for maximum customization. Flutter also offers specialized, higher level widgets that might be sufficient for your needs. For example, instead of Row you might prefer ListTile, an easy-to-use widget with properties for leading and trailing icons, and up to 3 lines of text. Instead of Column, you might prefer ListView, a column-like layout that automatically scrolls if its content is too long to fit the available space.

#### Type of Layout Widgets

Layout widgets can be grouped into two distinct categories based on its child –

- Widget supporting a single child
- Widget supporting multiple child

#### Single Child Widgets

These widgets will have only one widget as its child and every widget will have a special layout functionality. For example, Center widget just centers its child widget with respect to its parent widget and Container widget provides complete flexibility to place its child at any given place inside it using different options like padding, decoration, etc. Single child widgets are great options to create high quality widget having single functionality such as button, label, etc.

#### Multiple Child Widgets

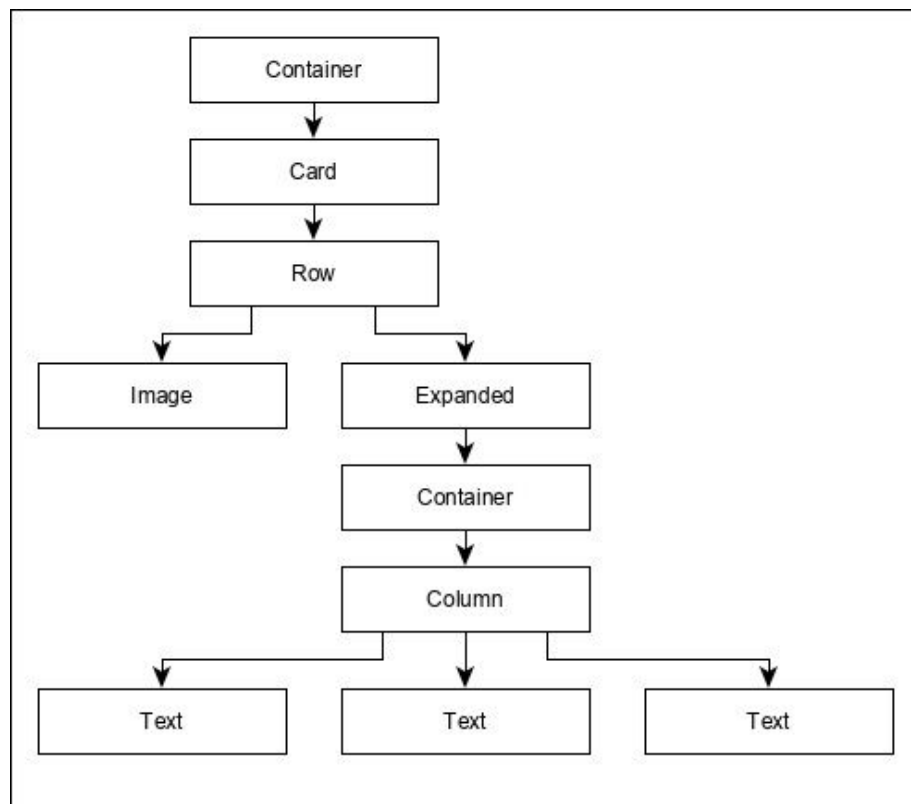
These widgets will have more than one child widgets and the layout of each widget is unique. For example, Row widget allows the laying out of its children in horizontal direction, whereas Column widget allows laying out of its children in vertical direction. By composing Row and Column, widget with any level of complexity can be built.

- Row – Allows to arrange its children in a horizontal manner.
- Column – Allows to arrange its children in a vertical manner.
- ListView – Allows to arrange its children as list.
- GridView – Allows to arrange its children as gallery.
- Expanded – Used to make the children of Row and Column widget to occupy the maximum possible area.
- Table – Table based widget.
- Stack – Stack based widget.

### Example Layout:

```
class ProductBox extends StatelessWidget {
  ProductBox({Key key, this.name, this.description, this.price,
    this.image})
    : super(key: key);

  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(2),
      height: 120, child: Card(
        child: Row(
          children: <Widget>[
            Image.asset("assets/appimages/" +image), Expanded(
              child: Container(
                padding: EdgeInsets.all(5),
                child: Column(
                  children: <Widget>[
                    Text(this.name, style: TextStyle(fontWeight:
                      FontWeight.bold)), Text(this.description),
                    Text("Price: " + this.price.toString()),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



## Chapter 4

### Intents

In Android, there are two main use cases for Intents: navigating between Activities, and communicating with components. Flutter, on the other hand, does not have the concept of intents, although you can still start intents through native integrations (using a plugin). Flutter doesn't really have a direct equivalent to activities and fragments; rather, in Flutter you navigate between screens, using a Navigator and Routes, all within the same Activity.

A Route is an abstraction for a “screen” or “page” of an app, and a Navigator is a widget that manages routes. A route roughly maps to an Activity, but it does not carry the same meaning. A navigator can push and pop routes to move from screen to screen. Navigators work like a stack on which you can push() new routes you want to navigate to, and from which you can pop() routes when you want to “go back”.

In Android, you declare your activities inside the app's AndroidManifest.xml. In Flutter, you have a couple options to navigate between pages:

- Specify a Map of route names. (MaterialApp)
- Directly navigate to a route. (WidgetApp)

Flutter can handle incoming intents from Android by talking directly to the Android layer and requesting the data that was shared. The basic flow implies that we first handle the shared text data on the Android native side (in our Activity), and then wait until Flutter requests for the data to provide it using a MethodChannel.

#### Example Navigation:

```
void main() {
  runApp(MaterialApp(
    home: MyAppHome(), // becomes the route named '/'
    routes: <String, WidgetBuilder> {
      '/a': (BuildContext context) => MyPage(title: 'page A'),
      '/b': (BuildContext context) => MyPage(title: 'page B'),
      '/c': (BuildContext context) => MyPage(title: 'page C'),
    },
  ));
}

// Then using the Navigator Object

Navigator.of(context).pushNamed('/b');
```

## Chapter 5

### Activity

In Android, the Activity is the foundation of everything that shows up on the screen. Buttons, toolbars, and inputs, everything is in an Activity. In Flutter, the rough equivalent to an Activity is a Widget. Widgets don't map exactly to Android Activities, but while you're getting acquainted with how Flutter works you can think of them as "the way you declare and construct UI". However, these have a few differences to an Activity. To start, widgets have a different lifespan: they are immutable and only exist until they need to be changed. Whenever widgets or their state change, Flutter's framework creates a new tree of widget instances. In comparison, an Android Activity is drawn once and does not redraw until `invalidate` is called.

Flutter's widgets are lightweight, in part due to their immutability. Because they aren't views themselves, and aren't directly drawing anything, but rather are a description of the UI and its semantics that get "inflated" into actual view objects under the hood. Flutter includes the Material Components library. These are widgets that implement the Material Design guidelines. Material Design is a flexible design system optimized for all platforms, including iOS. But Flutter is flexible and expressive enough to implement any design language. For example, on iOS, you can use the Cupertino widgets to produce an interface that looks like Apple's iOS design language.

In Android, you update your views by directly mutating them. However, in Flutter, Widgets are immutable and are not updated directly, instead you have to work with the widget's state. This is where the concept of Stateful and Stateless widgets comes from. A `StatelessWidget` is just what it sounds like—a widget with no state information. `StatelessWidgets` are useful when the part of the user interface you are describing does not depend on anything other than the configuration information in the object. For example, in Android, this is similar to placing an `ImageView` with your logo. The logo is not going to change during runtime, so use a `StatelessWidget` in Flutter. If you want to dynamically change the UI based on data received after making an HTTP call or user interaction then you have to work with `StatefulWidget` and tell the Flutter framework that the widget's State has been updated so it can update that widget.

The important thing to note here is at the core of both stateless and stateful widgets behave the same. They rebuild every frame, the difference is the `StatefulWidget` has a State object that stores state data across frames and restores it.

#### Stateful Widget Lifecycle:

##### 1. `createState()`

When Flutter is instructed to build a `StatefulWidget`, it immediately calls `createState()`. This method must exist. A `StatefulWidget` rarely needs to be more complicated than this.

```
class MyHomePage extends StatefulWidget {  
  @override  
  _MyHomePageState createState() => new _MyHomePageState();  
}
```

```
}
```

## 2. mounted is true

When createState creates the state class, a buildContext is assigned to that state.

A BuildContext is, overly simplified, the place in the widget tree in which this widget is placed. All widgets have a bool this.mounted property. It turns true when the buildContext is assigned. It is an error to call setState when a widget is unmounted. This property is useful when a method on your state calls setState() but it isn't clear when or how often that method will be called. Perhaps its being called in response to a stream updating. You can use if (mounted) {... to make sure the State exists before calling setState().

## 3. initState()

This is the first method called when the widget is created. initState is called once and only once. It must also call super.initState().

Initialize data that relies on the specific BuildContext for the created instance of the widget.

Initialize properties that rely on this widget's 'parent' in the tree.

Subscribe to Streams, ChangeNotifiers, or any other object that could change the data on this widget.

```
@override
initState() {
  super.initState();
  // Add listeners to this class
  cartItemStream.listen((data) {
    _updateWidget(data);
  });
}
```

## 4. didChangeDependencies()

The didChangeDependencies method is called immediately after initState on the first time the widget is built. It will also be called whenever an object that this widget depends on data from is called. For example, if it relies on an InheritedWidget, which updates. build() is always called after didChangeDependencies is called, so this is rarely needed.

## 5. build()

This method is often called (think fps + render). It is a required, @override and must return a Widget. Remember that in Flutter all GUI is a widget with a child or children, even 'Padding', 'Center'.

## 6. didUpdateWidget(Widget oldWidget)

didUpdateWidget() is called if the parent widget changes and has to rebuild this widget (because it needs to give it different data), but it's being rebuilt with the same runtimeType, then this method is called. If the state's build() method relies on a Stream or other object that can change, unsubscribe from the old object and re-subscribe to the new instance in didUpdateWidget().

```
@override
void didUpdateWidget(Widget oldWidget) {
  if (oldWidget.importantProperty != widget.importantProperty) {
    _init();
  }
}
```

## **7. setState()**

The 'setState()' method is called often from the Flutter framework itself and from the developer. It is used to notify the framework that "data has changed", and the widget at this build context should be rebuilt. setState() takes a callback which cannot be async. It is for this reason it can be called often as required, because repainting is cheap.

```
void updateProfile(String name) {  
  setState(() => this.name = name);  
}
```

## **8. deactivate()**

This is rarely used. 'deactivate()' is called when State is removed from the tree, but it might be reinserted before the current frame change is finished. This method exists basically because State objects can be moved from one point in a tree to another.

## **9. dispose()**

'dispose()' is called when the State object is removed, which is permanent. This method is where to unsubscribe and cancel all animations, streams, etc.

## **10. mounted is false**

The state object can never remounted, and an error is thrown if setState() is called.

## Chapter 6

### Firestore

Firestore is a backend platform for building Web, Android and IOS applications. It offers real time database, different APIs, multiple authentication types and hosting platform. This is an introductory tutorial, which covers the basics of the Firestore platform and explains how to deal with its various components and sub-components.

#### Firestore Features:

- Real-time Database – Firestore supports JSON data and all users connected to it receive live updates after every change.
- Authentication – We can use anonymous, password or different social authentications.
- Hosting – The applications can be deployed over secured connection to Firestore servers.

#### Firestore Advantages:

- It is simple and user friendly. No need for complicated configuration.
- The data is real-time, which means that every change will automatically update connected clients.
- Firestore offers simple control dashboard.

The services used in our application:

#### Cloud Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firestore and Google Cloud Platform. Like Firestore Realtime Database, it keeps your data in sync across client apps through real time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firestore and Google Cloud Platform products, including Cloud Functions.

#### Firestore Authentication

Firestore Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. Firestore Authentication integrates tightly with other Firestore services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.

#### Example Code:

##### Firestore:

```
Firestore.instance.collection('books').document().setData({ 'title':  
'title', 'author': 'author' });
```

##### Authentication:

```
final FirebaseAuth user = (await auth.createUserWithEmailAndPassword(  
email: 'an email',password: 'a password' )).user;
```

## Chapter 7

### Camera

Camera is mainly used to capture pictures and videos. We can control the camera by using methods of camera API. Android provides full access to the device camera hardware so you can build a wide range of camera or vision-based apps. Or if you just need a way for the user to capture a photo, you can simply request an existing camera app to capture a photo and return it to you.

Flutter provides the camera plugin for this purpose. The camera plugin provides tools to get a list of the available cameras, display a preview coming from a specific camera, and take photos or videos. This following demonstrates how to use the camera plugin to display a preview, take a photo, and display it:

- Add the required dependencies.
- Get a list of the available cameras.
- Create and initialize the CameraController.
- Use a CameraPreview to display the camera's feed.
- Take a picture with the CameraController.
- Display the picture with an Image widget.

#### Example:

```
import 'dart:async';
import 'dart:io';
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:path/path.dart' show join;
import 'package:path_provider/path_provider.dart';

Future<void> main() async {
  final cameras = await availableCameras();
  final firstCamera = cameras.first;

  runApp(
    MaterialApp(
      theme: ThemeData.dark(),
      home: TakePictureScreen(
        camera: firstCamera,
      ),
    ),
  );
}

class TakePictureScreen extends StatefulWidget {
  final CameraDescription camera;
  const TakePictureScreen({
    Key key,
    @required this.camera,
  }) : super(key: key);
```



```

    @override
    TakePictureScreenState createState() => TakePictureScreenState();
}

class TakePictureScreenState extends State<TakePictureScreen> {
  CameraController _controller;
  Future<void> _initializeControllerFuture;

  @override
  void initState() {
    super.initState();
    _controller = CameraController(
      widget.camera,
      ResolutionPreset.medium,
    );
    _initializeControllerFuture = _controller.initialize();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Take a picture')),
      body: FutureBuilder<void>(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            return CameraPreview(_controller);
          } else {
            return Center(child: CircularProgressIndicator());
          }
        },
      ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.camera_alt),
        onPressed: () async {
          try {
            await _initializeControllerFuture;

            final path = join(
              (await getTemporaryDirectory()).path,
              '${DateTime.now()}.png',
            );
            await _controller.takePicture(path);
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => DisplayPictureScreen(imagePath:path),
              ),
            );
          } catch (e) {
            print(e);
          }
        },
      ),
    );
  }
}

```

```

        },
      ),
    );
  }
}

class DisplayPictureScreen extends StatelessWidget {
  final String imagePath;
  const DisplayPictureScreen({Key key, this.imagePath}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Display the Picture')),
      body: Image.file(File(imagePath)),
    );
  }
}

```

## Chapter 8

### Location API

Android location APIs make it easy for you to build location-aware applications, without needing to focus on the details of the underlying location technology. This becomes possible with the help of Google Play services, which facilitates adding location awareness to your app with automated location tracking, geofencing, and activity recognition. The Location object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude and velocity.

In Flutter, we use the location package to achieve this. The location package handles getting location on Android and iOS. It also provides callbacks when location is changed.

#### Example:

```
var currentLocation = LocationData;
var location = new Location();
try {
  currentLocation = await location.getLocation();
} on PlatformException catch (e) {
  if (e.code == 'PERMISSION_DENIED') {
    error = 'Permission denied';
  }
  currentLocation = null;
}
```

#### OR

```
var location = new Location();

location.onLocationChanged().listen((LocationData currentLocation) {
  print(currentLocation.latitude);
  print(currentLocation.longitude);
});
```

## Chapter 9

### Generate APK

Android Package (APK) is the package file format used by the Android operating system for distribution and installation of mobile apps and middleware. To make an APK file, a program for Android is first compiled, and then all of its parts are packaged into one container file. An APK file contains all of a program's code (such as dex files), resources, assets, certificates, and manifest file. As is the case with many file formats, APK files can have any name needed, provided that the file name ends in the file extension ".apk". APK files are a type of archive file, specifically in zip format-type packages, based on the JAR file format, with .apk as the filename extension.

Although app bundles are preferred over APKs, there are stores that don't yet support app bundles. In this case, build a release APK for each target ABI (Application Binary Interface).

#### From the Command Line:

```
Enter cd <app dir>
(Replace <app dir> with your application's directory.)
Run flutter build apk --split-per-abi
(The flutter build command defaults to --release.)
```

This command results in two APK files:

```
<app dir>/build/app/outputs/apk/release/app-armeabi-v7a-release.apk
<app dir>/build/app/outputs/apk/release/app-arm64-v8a-release.apk
```

Removing the --split-per-abi flag results in a fat APK that contains your code compiled for all the target ABIs. Such APKs are larger in size than their split counterparts, causing the user to download native binaries that are not applicable to their device's architecture.

## **Conclusion**

Healthy living is a combination of many things, including good nutrition, regular exercise and a positive attitude. Taking care of your body and feeling pride in your accomplishments can improve both your physical and mental health. There are many things you can do to improve your quality of life - improving your diet and exercising regularly are two of the easiest steps. From tracking our eating habits and how much activity we partake in we can get an idea of how active we are and if we are eating healthy. When comparing the two week period we were able to see some improvements from the first week and the second logs. Our deep learning model achieved an accuracy of 80% - 85% classifying food images. The application was used for a week by one of our teammates and the application helped her to maintain her health conscious routine more effectively.

## Source Code

### <pubspec.yaml>

```
name: eatfit
description: Fitness Application to help people track eating habits.
version: 1.0.0+1
```

```
environment:
  sdk: ">=2.1.0 <3.0.0"
```

```
dependencies:
  flutter:
    sdk: flutter
  provider: ^3.1.0
  dio: ^2.1.16
  fluro: ^1.5.1
  firebase_core: ^0.4.0+9
  firebase_auth: ^0.14.0+5
  cloud_firestore: ^0.12.9+3
  image_picker: ^0.6.1+4
  circle_wave_progress: ^0.0.4
  flutter_dialogflow: ^0.1.2
  location: ^2.3.5
  audioplayers: ^0.7.8
  intl: ^0.16.0
  google_maps_flutter: ^0.5.21+7
```

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_launcher_icons: ^0.7.3
```

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

```
flutter:
  uses-material-design: true
  assets:
    - assets/icon.png
    - assets/credentials.json
    - assets/gifs/SDM_9.gif
    - assets/gifs/SDM_12.gif
    - assets/gifs/SDM_23.gif
    - assets/gifs/SDM_18.gif
    - assets/gifs/SDM_58.gif
    - assets/gifs/SDM_65.gif
    - assets/audio/SDM_9.mp3
    - assets/audio/SDM_12.mp3
    - assets/audio/SDM_23.mp3
    - assets/audio/SDM_18.mp3
    - assets/audio/SDM_58.mp3
    - assets/audio/SDM_65.mp3
  fonts:
    - family: Montserrat
      fonts:
        - asset: assets/fonts/Montserrat-Regular.ttf
          weight: 400
        - asset: assets/fonts/Montserrat-Thin.ttf
          weight: 100
        - asset: assets/fonts/Montserrat-Bold.ttf
          weight: 700
```

### <main.dart>

```

import 'package:eatfit/router.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

void main() {
  FluroRouter.setupRouter();
  runApp(EatFit());
}

class EatFit extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        StreamProvider<FirebaseUser>.value(
          value: FirebaseAuth.instance.onAuthStateChanged,
        ),
      ],
      child: MaterialApp(
        title: 'EatFit',
        theme: ThemeData(
          primaryColor: Color(0xFF01161E),
          accentColor: Color(0xFF92DCE5),
          errorColor: Color(0xFFD64933),
          //canvasColor: Colors.transparent,
          fontFamily: 'Montserrat',
        ),
        initialRoute: 'home',
        onGenerateRoute: FluroRouter.router.generator,
        debugShowCheckedModeBanner: false,
      ),
    );
  }
}

```

#### **<router.dart>**

```

import 'package:eatfit/components/root.dart';
import 'package:eatfit/views/auth.dart';
import 'package:eatfit/views/chat.dart';
import 'package:eatfit/views/exercise/doExercise.dart';
import 'package:eatfit/views/exercise/exerciseHome.dart';
import 'package:eatfit/views/food.dart';
import 'package:eatfit/views/home.dart';
import 'package:eatfit/views/landing.dart';
import 'package:eatfit/views/maps.dart';
import 'package:eatfit/views/snap.dart';
import 'package:flutter/material.dart';
import 'package:fluro/fluro.dart';

class FluroRouter {
  static Router router = Router();

  static Handler _rootHandler = Handler(
    handlerFunc: (BuildContext context, Map<String, dynamic> params) {
      return Landing();
    },
  );

  static Handler _authHandler = Handler(
    handlerFunc: (BuildContext context, Map<String, dynamic> params) {
      return Auth();
    },
  );

  static Handler _homeHandler = Handler(

```

```

        handlerFunc: (BuildContext context, Map<String, dynamic> params) {
          return Root(
            child: Home(
              id: params["id"][0],
            ),
          );
        },
      );
    );

    static Handler _exerciseHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Root(
          child: ExerciseHome(),
        );
      },
    );

    static Handler _snapHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Root(
          child: Snap(
            value: int.parse(params["value"][0]),
          ),
        );
      },
    );

    static Handler _foodHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Root(
          child: Food(),
        );
      },
    );

    static Handler _chatHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Root(
          child: Chat(),
        );
      },
    );

    static Handler _doExerciseHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Root(
          child: DoExercise(
            id: params["id"][0],
          ),
        );
      },
    );

    static Handler _mapsHandler = Handler(
      handlerFunc: (BuildContext context, Map<String, dynamic> params) {
        return Maps();
      },
    );

    static void setupRouter() {
      router.define(
        'home',
        handler: _rootHandler,
        transitionType: TransitionType.cupertino,
      );
      router.define(

```



```

        'auth',
        handler: _authHandler,
        transitionType: TransitionType.cupertino,
    );
    router.define(
      'dashboard/:id',
      handler: _homeHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'exercise',
      handler: _exerciseHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'snap/:value',
      handler: _snapHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'food',
      handler: _foodHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'chat',
      handler: _chatHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'exercise/:id',
      handler: _doExerciseHandler,
      transitionType: TransitionType.cupertino,
    );
    router.define(
      'maps',
      handler: _mapsHandler,
      transitionType: TransitionType.cupertino,
    );
  }
}

```

#### **<models/user.dart>**

```

import 'package:eatfit/util/calorie.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';

```

```

import 'meal.dart';

```

```

class User with ChangeNotifier {

```

```

  String id;
  String name;
  String email;
  List<Meal> meals;
  int currentCalories;
  int lifestyleChoice;
  String gender;

```

```

  User(
    {this.id,
     this.name,
     this.meals,
     this.email,
     this.currentCalories,
     this.lifestyleChoice,
     this.gender});

```

```

factory User.fromFirestore(DocumentSnapshot doc) {
  Map data = doc.data;
  return User(
    id: doc.documentID,
    name: data['name'] ?? '',
    email: data['email'] ?? '',
    currentCalories: data['currentCalories'] ?? 0,
    meals: Meal.fromData(data['meals']) ?? [],
    lifestyleChoice: data['lifestyleChoice'] ?? Calorie.MALE_MAINTAIN,
    gender: data['gender'] ?? "Male",
  );
}

String getName() => this.name;
String getID() => this.id;
String getEmail() => this.email;
List<Meal> getMeals() => this.meals;
int getCurrentCalories() => this.currentCalories;
int getLifestyleChoice() => this.lifestyleChoice;
String getGender() => this.gender;

setName(String name) => this.name = name;
setLifestyleChoice(int value) => this.lifestyleChoice = value;
setGender(String gender) => this.gender = gender;

void addMeal(Meal meal) {
  this.meals.add(meal);
  notifyListeners();
}

void updateCalorieCount(int value) {
  this.currentCalories += value;
  notifyListeners();
}
}

<models/meal.dart>
import 'package:cloud_firestore/cloud_firestore.dart';

class Meal {
  String name;
  int calories;
  DateTime time;

  Meal({this.name, this.calories, this.time});

  static List<Meal> fromData(List meals) {
    List<Meal> finalMeals = [];
    for (var data in meals) {
      Meal m = Meal(
        name: data['name'] ?? '',
        calories: data['calories'] ?? 0,
        time:
          DateTime.fromMillisecondsSinceEpoch(data['time'].seconds * 1000) ??
            '',
      );
      finalMeals.add(m);
    }
    return finalMeals;
  }
}

factory Meal.fromFirestore(DocumentSnapshot doc) {
  Map data = doc.data ?? {};
  return Meal(
    name: data['name'] ?? '',

```

```

        calories: data['calories'] ?? 0,
        time: DateTime.fromMillisecondsSinceEpoch(data['time'].seconds * 1000) ??
        '',
    );
}

String getMeal() => this.name;
int getCalorieValue() => this.calories;
DateTime getTimeOfConsumption() => this.time;

setMeal(String name) => this.name = name;
setCalorieValue(int calories) => this.calories = calories;
setTimeOfConsumption(DateTime date) => this.time = date;
}

```

#### <util/api.dart>

```

import 'package:dio/dio.dart';
import 'dart:io';
import 'dart:async';

class API {
  static final String url = 'https://cc6a3720.ngrok.io';
  static BaseOptions opts = BaseOptions(
    baseUrl: url,
    responseType: ResponseType.json,
    connectTimeout: 30000,
    receiveTimeout: 30000,
  );
  static final service = Dio(opts);

  const API();

  Future<String> predictFood(File image) async {
    FormData data =
      FormData.fromMap({"file": await MultipartFile.fromFile(image.path)});
    Response res = await service.post(
      "/classify_food",
      data: data,
      options: Options(
        method: 'POST',
      ),
    );
    String prediction = res.data['result'].toString();
    return prediction;
  }
}

```

#### <util/db.dart>

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:eatfit/models/user.dart';

class DatabaseService {
  final firestoreDB = Firestore.instance;
  Future<User> getUser(String id) {
    return firestoreDB
      .collection('users')
      .document(id)
      .get()
      .then((doc) => User.fromFirestore(doc));
  }

  Future<void> addMeal(List data, String id) {
    return firestoreDB
      .collection('users')
      .document(id)

```

```

        .updateData({"meals": FieldValue.arrayUnion(data)}));
    }
    Future<void> updateUser(Map data, String id) {
        return firestoreDB.collection('users').document(id).updateData(data);
    }
}

<components/root.dart>
import 'package:eatfit/components/bottomAppBar.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class Root extends StatefulWidget {
    final Widget child;

    const Root({Key key, this.child}) : super(key: key);

    @override
    _RootState createState() => _RootState();
}

class _RootState extends State<Root> {
    FirebaseAuth fbUser;

    @override
    void initState() {
        super.initState();
        FirebaseAuth.instance.currentUser().then((user) {
            setState(() {
                this.fbUser = user;
            });
        });
    }

    @override
    Widget build(BuildContext context) {
        var _items = [
            {'icon': Icon(Icons.photo_camera), 'text': Text("Camera")},
            {'icon': Icon(Icons.photo), 'text': Text("Gallery")}
        ];
        return Scaffold(
            appBar: AppBar(
                title: Text(
                    "EatFit",
                    style: TextStyle(
                        color: Theme.of(context).accentColor,
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        letterSpacing: 1.5,
                    ),
                ),
            centerTitle: true,
            elevation: 0,
            backgroundColor: Theme.of(context).primaryColor,
        ),
        body: this.widget.child,
        floatingActionButton: FloatingActionButton(
            focusColor: Color(0xFF7C7C7C),
            onPressed: () => showModalBottomSheet(
                context: context,
                builder: (BuildContext context) {
                    return ListView.separated(
                        shrinkWrap: true,
                        separatorBuilder: (context, index) => Divider(
                            color: Colors.black26,
                        ),

```

```

        itemCount: 2,
        itemBuilder: (context, index) => Padding(
          padding: EdgeInsets.all(10),
          child: ListTile(
            title: _items[index]['text'],
            leading: _items[index]['icon'],
            onTap: () => Navigator.pushNamed(context, "snap/$index"),
          ),
        ),
      ),
    ),
    tooltip: 'Action',
    child: Icon(
      Icons.add,
      color: Colors.black,
      size: 30,
    ),
    elevation: 3,
  ),
  floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
  bottomNavigationBar: FABBottomAppBar(
    notchShape: CircularNotchedRectangle(),
    color: Theme.of(context).accentColor,
    selectedColor: Theme.of(context).accentColor,
    backgroundColor: Theme.of(context).primaryColor,
    centerItemText: "Snap!",
    iconSize: 30,
    onTabSelected: _selectedTab,
    items: [
      FABBottomAppBarItem(iconData: Icons.home, text: 'Home'),
      FABBottomAppBarItem(iconData: Icons.local_dining, text: 'Food'),
      FABBottomAppBarItem(iconData: Icons.fitness_center, text: 'Exercise'),
      FABBottomAppBarItem(iconData: Icons.exit_to_app, text: 'Logout'),
    ],
  ),
);
}

void _selectedTab(int index) {
  print("Tab: $index");
  switch (index) {
    case 0:
      Navigator.pushReplacementNamed(context, 'dashboard/${this.fbUser.uid}');
      break;
    case 1:
      Navigator.pushNamed(context, 'food');
      break;
    case 2:
      Navigator.pushNamed(context, 'exercise');
      break;
    case 3:
      FirebaseAuth.instance.signOut();
      Navigator.pushReplacementNamed(context, 'home');
      break;
    default:
      print(index);
  }
}
}
}

```

#### <components/customCard.dart>

```

import 'package:flutter/material.dart';

class CustomCard extends StatelessWidget {
  final Widget content;
  final Color bgColor;

```

```

CustomCard({Key key, this.content, this.bgColor}) : super(key: key);

@override
Widget build(BuildContext context) {
  return Container(
    margin: const EdgeInsets.symmetric(vertical: 16, horizontal: 24),
    child: new Stack(
      children: <Widget>[
        new Container(
          decoration: new BoxDecoration(
            color: bgColor,
            shape: BoxShape.rectangle,
            borderRadius: new BorderRadius.circular(8.0),
            boxShadow: <BoxShadow>[
              new BoxShadow(
                color: Colors.black12,
                blurRadius: 15,
                offset: new Offset(2.0, 7.0),
              )
            ],
          ),
          child: content,
        ),
      ],
    ),
  );
}

<views/landing.dart>
import 'package:eatfit/components/customLoader.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class Landing extends StatefulWidget {
  Landing({Key key}) : super(key: key);

  _LandingState createState() => _LandingState();
}

class _LandingState extends State<Landing> {
  @override
  void initState() {
    FirebaseAuth.instance.currentUser().then((currentUser) {
      if (currentUser == null) {
        Navigator.pushReplacementNamed(context, 'auth');
      } else {
        Navigator.pushReplacementNamed(context, 'dashboard/${currentUser.uid}');
      }
    });
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        color: Colors.white,
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              Text(
                "EatFit",

```

```

        style: TextStyle(
          fontSize: 20,
          color: Theme.of(context).primaryColor,
        ),
      ),
      Padding(
        padding: EdgeInsets.all(10),
        child: Image.asset(
          'assets/icon.png',
          width: 100,
          height: 100,
        ),
      ),
    ),
    CustomLoader(),
  ],
),
),
),
);
}
}

```

#### <views/auth.dart>

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:eatfit/util/bottomWaveClipper.dart';

class Auth extends StatefulWidget {
  @override
  _AuthState createState() => _AuthState();
}

class _AuthState extends State<Auth> {
  final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();
  TextEditingController _emailController = new TextEditingController();
  TextEditingController _passwordController = new TextEditingController();
  TextEditingController _nameController = new TextEditingController();
  String _email;
  String _password;
  String _displayName;
  bool isLoading = false;
  final FirebaseAuth firebaseAuth = FirebaseAuth.instance;
  final Firestore db = Firestore.instance;

  @override
  void initState() {
    super.initState();
  }

  void showInSnackBar(String value) {
    FocusScope.of(context).requestFocus(new FocusNode());
    _scaffoldKey.currentState?.removeCurrentSnackBar();
    _scaffoldKey.currentState.showSnackBar(new SnackBar(
      content: new Text(
        value,
        textAlign: TextAlign.center,
        style: TextStyle(
          color: Colors.white,
          fontSize: 16.0,
        ),
      ),
      backgroundColor: Colors.blue,
      duration: Duration(seconds: 3),
    ));
  }
}

```

```

@override
Widget build(BuildContext context) {
  Color primary = Theme.of(context).primaryColor;

  Widget logo() {
    return Padding(
      padding:
        EdgeInsets.only(top: MediaQuery.of(context).size.height * 0.15),
      child: Container(
        width: MediaQuery.of(context).size.width,
        height: 220,
        child: Stack(
          children: <Widget>[
            Positioned(
              child: Container(
                child: Align(
                  child: Container(
                    decoration: BoxDecoration(
                      shape: BoxShape.circle,
                      color: Colors.white,
                    ),
                    width: 175,
                    height: 175,
                  ),
                ),
              ),
              height: 154,
            ),
            Positioned(
              child: Container(
                height: 154,
                child: Align(
                  child: Text(
                    "EatFit",
                    style: TextStyle(
                      fontSize: 48,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ),
                  ),
                ),
              ),
            ),
            Positioned(
              width: MediaQuery.of(context).size.width * 0.15,
              height: MediaQuery.of(context).size.width * 0.15,
              bottom: MediaQuery.of(context).size.height * 0.046,
              right: MediaQuery.of(context).size.width * 0.22,
              child: Container(
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: Colors.white,
                ),
              ),
            ),
            Positioned(
              width: MediaQuery.of(context).size.width * 0.08,
              height: MediaQuery.of(context).size.width * 0.08,
              bottom: 0,
              right: MediaQuery.of(context).size.width * 0.32,
              child: Container(
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: Colors.white,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```



```

    ),
  ),
],
),
),
);
}

//input widget
Widget _input(Icon icon, String hint, TextEditingController controller,
  bool obscure) {
  return Container(
    padding: EdgeInsets.only(left: 20, right: 20),
    child: TextField(
      controller: controller,
      obscureText: obscure,
      style: TextStyle(
        fontSize: 20,
      ),
      decoration: InputDecoration(
        hintStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 20),
        hintText: hint,
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(30),
          borderSide: BorderSide(
            color: Theme.of(context).primaryColor,
            width: 2,
          ),
        ),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(30),
          borderSide: BorderSide(
            color: Theme.of(context).primaryColor,
            width: 3,
          ),
        ),
        prefixIcon: Padding(
          child: IconTheme(
            data: IconThemeData(color: Theme.of(context).accentColor),
            child: icon,
          ),
          padding: EdgeInsets.only(left: 30, right: 10),
        ),
      ),
    ),
  );
}

//button widget
Widget _button(String text, Color splashColor, Color highlightColor,
  Color fillColor, Color textColor, void function()) {
  return RaisedButton(
    highlightElevation: 0.0,
    splashColor: splashColor,
    highlightColor: highlightColor,
    elevation: 0.0,
    color: fillColor,
    shape: RoundedRectangleBorder(
      borderRadius: new BorderRadius.circular(30.0),
    ),
    child: Text(
      text,
      style: TextStyle(
        fontWeight: FontWeight.bold,
        color: textColor,
        fontSize: 20,
      ),
    ),
  );
}

```

```

    ),
  ),
  onPressed: () {
    function();
  },
);
}

void _loginUser() async {
  showInSnackBar('Logging You In ...');
  _email = _emailController.text;
  _password = _passwordController.text;
  _emailController.clear();
  _passwordController.clear();
  final FirebaseUser user = (await firebaseAuth.signInWithEmailAndPassword(
    email: _email, password: _password))
    .user;
  Navigator.pushReplacementNamed(context, "dashboard/${user.uid}");
}

void _registerUser() async {
  showInSnackBar('Signing You Up ...');
  _email = _emailController.text;
  _password = _passwordController.text;
  _displayName = _nameController.text;
  _emailController.clear();
  _passwordController.clear();
  _nameController.clear();
  final FirebaseUser _user =
    (await firebaseAuth.createUserWithEmailAndPassword(
      email: _email, password: _password))
      .user;
  DocumentReference ref = db.collection("users").document(_user.uid);
  ref.setData({
    'id': _user.uid,
    'email': _user.email,
    'name': _displayName,
    'gender': 'Male',
  }, merge: true);
  Navigator.pushReplacementNamed(context, "dashboard/${_user.uid}");
}

void _loginSheet() {
  _scaffoldKey.currentState.showBottomSheet<void>((BuildContext context) {
    return DecoratedBox(
      decoration: BoxDecoration(color: Colors.transparent),
      child: ClipRRect(
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(40.0),
          topRight: Radius.circular(40.0),
        ),
        child: Container(
          child: ListView(
            children: <Widget>[
              Container(
                child: Stack(
                  children: <Widget>[
                    Positioned(
                      left: 10,
                      top: 10,
                      child: IconButton(
                        onPressed: () {
                          Navigator.of(context).pop();
                          _emailController.clear();
                          _passwordController.clear();
                        },

```

```

        icon: Icon(
          Icons.close,
          size: 30.0,
          color: Theme.of(context).primaryColor,
        ),
      ),
    ],
  ),
  height: 50,
  width: 50,
),
SingleChildScrollView(
  child: Column(
    children: <Widget>[
      Container(
        width: MediaQuery.of(context).size.width,
        height: 140,
        child: Stack(
          children: <Widget>[
            Positioned(
              child: Align(
                child: Container(
                  width: 150,
                  height: 150,
                  decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: Theme.of(context).primaryColor,
                  ),
                ),
              alignment: Alignment.center,
            ),
            Positioned(
              child: Container(
                child: Text(
                  "LOGIN",
                  style: TextStyle(
                    fontSize: 26,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ),
                ),
              alignment: Alignment.center,
            ),
          ],
        ),
      ),
      Padding(
        padding: EdgeInsets.only(bottom: 20, top: 60),
        child: _input(
          Icon(Icons.email),
          "EMAIL",
          _emailController,
          false,
        ),
      ),
      Padding(
        padding: EdgeInsets.only(bottom: 20),
        child: _input(
          Icon(Icons.lock),
          "PASSWORD",
          _passwordController,
          true,
        ),
      ),
    ],
  ),
),

```

```

    ),
    SizedBox(
      height: 20,
    ),
    Padding(
      padding: EdgeInsets.only(
        left: 20,
        right: 20,
        bottom: MediaQuery.of(context).viewInsets.bottom,
      ),
      child: Container(
        child: _button(
          "LOGIN",
          Colors.white,
          primary,
          primary,
          Colors.white,
          _loginUser,
        ),
        height: 50,
        width: MediaQuery.of(context).size.width,
      ),
    ),
    SizedBox(
      height: 20,
    ),
  ],
),
],
),
height: MediaQuery.of(context).size.height / 1.1,
width: MediaQuery.of(context).size.width,
color: Colors.white,
),
),
);
});
}

void _registerSheet() {
  _scaffoldKey.currentState.showBottomSheet<void>((BuildContext context) {
    return DecoratedBox(
      decoration: BoxDecoration(color: Colors.transparent),
      child: ClipRRect(
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(40.0),
          topRight: Radius.circular(40.0),
        ),
        child: Container(
          child: ListView(
            children: <Widget>[
              Container(
                child: Stack(
                  children: <Widget>[
                    Positioned(
                      left: 10,
                      top: 10,
                      child: IconButton(
                        onPressed: () {
                          Navigator.of(context).pop();
                          _emailController.clear();
                          _passwordController.clear();
                          _nameController.clear();
                        },
                        icon: Icon(

```

```
Icons.close,  
size: 30.0,  
color: Theme.of(context).primaryColor,  
),  
),  
),  
],  
),  
height: 50,  
width: 50,  
),  
SingleChildScrollView(  
child: Column(children: <Widget>[  
Container(  
width: MediaQuery.of(context).size.width,  
height: 140,  
child: Stack(  
children: <Widget>[  
Positioned(  
child: Align(  
child: Container(  
width: 150,  
height: 150,  
decoration: BoxDecoration(  
shape: BoxShape.circle,  
color: Theme.of(context).primaryColor,  
),  
),  
alignment: Alignment.center,  
),  
),  
Positioned(  
child: Container(  
child: Text(  
"REGISTER",  
style: TextStyle(  
fontSize: 26,  
fontWeight: FontWeight.bold,  
color: Colors.white,  
),  
),  
alignment: Alignment.center,  
),  
),  
],  
),  
),  
Padding(  
padding: EdgeInsets.only(  
bottom: 20,  
top: 60,  
),  
child: _input(  
Icon(Icons.account_circle),  
"NAME",  
_nameController,  
false,  
),  
),  
Padding(  
padding: EdgeInsets.only(  
bottom: 20,  
),  
child: _input(  
Icon(Icons.email),  
"EMAIL",
```



```

    ),
    padding: EdgeInsets.only(
      top: 80,
      left: 20,
      right: 20,
    ),
  ),
  Padding(
    child: Container(
      child: OutlineButton(
        highlightedBorderColor: Colors.white,
        borderSide: BorderSide(color: Colors.white, width: 2.0),
        highlightElevation: 0.0,
        splashColor: Colors.white,
        highlightColor: Theme.of(context).primaryColor,
        color: Theme.of(context).primaryColor,
        shape: RoundedRectangleBorder(
          borderRadius: new BorderRadius.circular(30.0),
        ),
        child: Text(
          "REGISTER",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
            fontSize: 20,
          ),
        ),
        onPressed: () {
          _registerSheet();
        },
      ),
      height: 50,
    ),
    padding: EdgeInsets.only(top: 10, left: 20, right: 20),
  ),
  Expanded(
    child: Align(
      child: ClipPath(
        child: Container(
          color: Colors.white,
          height: 300,
        ),
        clipper: BottomWaveClipper(),
      ),
      alignment: Alignment.bottomCenter,
    ),
  ),
],
crossAxisAlignment: CrossAxisAlignment.stretch,
),
);
}
}

```

#### <views/home.dart>

```

import 'package:circle_wave_progress/circle_wave_progress.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:eatfit/components/customLoader.dart';
import 'package:eatfit/models/user.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

```

```

class Home extends StatefulWidget {
  final String id;

  const Home({Key key, this.id}) : super(key: key);

```

```

    @override
    _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  final Firestore db = Firestore.instance;
  bool _isLoading = true;
  User user;

  Future<User> _getCurrentUserFromFirestore(String id) async {
    return User.fromFirestore(await db.collection('users').document(id).get());
  }

  @override
  void initState() {
    super.initState();
    this._getCurrentUserFromFirestore(this.widget.id).then((_user) {
      setState(() {
        this.user = _user;
        this._isLoading = false;
      });
    });
  }

  double getPercentage() {
    return (this.user.currentCalories / this.user.lifestyleChoice) * 100;
  }

  @override
  Widget build(BuildContext context) {
    return !this._isLoading
      ? Container(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              Container(
                child: Row(
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.fromLTRB(10, 0, 6, 20),
                      child: Text(
                        "Welcome",
                        style: TextStyle(
                          color: Colors.white,
                          fontWeight: FontWeight.w100,
                          fontSize: 30,
                        ),
                      ),
                    ),
                  ],
                ),
                Padding(
                  padding: const EdgeInsets.fromLTRB(0, 0, 10, 20),
                  child: Text(
                    this.user.name,
                    style: TextStyle(
                      color: Theme.of(context).accentColor,
                      fontWeight: FontWeight.bold,
                      fontSize: 30,
                    ),
                  ),
                ),
              ],
            ),
            padding: EdgeInsets.all(10),

```



```

width: double.infinity,
decoration: new BoxDecoration(
  color: Theme.of(context).primaryColor,
  borderRadius: new BorderRadius.only(
    bottomLeft: const Radius.circular(50.0),
    bottomRight: const Radius.circular(50.0),
  ),
),
),
),
Padding(
  padding: const EdgeInsets.all(5.0),
  child: Text(
    "Today's Calorie Intake\n" +
      this.user.currentCalories.toString() +
      " / " +
      this.user.lifestyleChoice.toString(),
    style: TextStyle(
      letterSpacing: 0.5,
      //fontWeight: FontWeight.w100,
      color: Theme.of(context).primaryColor,
      fontSize: 25,
    ),
    textAlign: TextAlign.center,
  ),
),
),
Padding(
  padding: const EdgeInsets.all(10.0),
  child: CircleWaveProgress(
    backgroundColor: Colors.transparent,
    borderColor: Theme.of(context).primaryColor,
    borderWidth: 7.0,
    waveColor: Theme.of(context).accentColor,
    progress: getPercentage(),
    size: 275.0,
  ),
),
),
Padding(
  padding: const EdgeInsets.all(10),
  child: RaisedButton(
    splashColor: Color(0xFF7C7C7C),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Icon(
            Icons.chat,
            color: Theme.of(context).accentColor,
          ),
        ),
        Text(
          "Talk to Mr. X",
          style: TextStyle(
            color: Theme.of(context).accentColor,
            fontSize: 20,
          ),
        ),
      ],
    ),
  ),
  onPressed: () => Navigator.of(context).pushNamed("chat"),
  padding: EdgeInsets.all(10),
  color: Theme.of(context).primaryColor,
  elevation: 3,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(20),
  ),
),

```

```

        ),
      ),
      Padding(
        padding: EdgeInsets.fromLTRB(10, 10, 10, 30),
        child: Text(
          "@ Ryan Dsilva 2019-20",
          style: TextStyle(
            fontWeight: FontWeight.w100,
          ),
        ),
      ),
    ],
  ),
),
): Center(child: CustomLoader());
}
}

```

#### <views/chat.dart>

```

import 'package:eatfit/components/customLoader.dart';
import 'package:eatfit/models/user.dart';
import 'package:eatfit/util/db.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_dialogflow/dialogflow_v2.dart';

class Chat extends StatefulWidget {
  Chat({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _Chat createState() => new _Chat();
}

class _Chat extends State<Chat> {
  final List<ChatMessage> _messages = <ChatMessage>[];
  final TextEditingController _textController = new TextEditingController();
  User user;
  bool isLoading = true;
  DatabaseService db = DatabaseService();
  @override
  void initState() {
    super.initState();
    FirebaseAuth.instance.currentUser().then((user) {
      db.getUser(user.uid).then((newUser) {
        setState(() {
          this.user = newUser;
          this.isLoading = false;
        });
      });
    });
  }

  Widget _buildTextComposer() {
    return new IconTheme(
      data: new IconThemeData(color: Theme.of(context).accentColor),
      child: new Container(
        margin: const EdgeInsets.symmetric(horizontal: 8.0),
        child: new Row(
          children: <Widget>[
            new Flexible(
              child: new TextField(
                controller: _textController,
                onSubmitted: _handleSubmitted,

```

```

        decoration:
            new InputDecoration.collapsed(hintText: "Send Message"),
    ),
),
new Container(
    margin: new EdgeInsets.symmetric(horizontal: 4.0),
    child: new IconButton(
        icon: new Icon(Icons.send),
        onPressed: () => _handleSubmitted(_textController.text),
    ),
),
],
),
),
);
}

void response(query) async {
    _textController.clear();
    AuthGoogle authGoogle =
        await AuthGoogle(fileJson: "assets/credentials.json").build();
    Dialogflow dialogflow =
        Dialogflow(authGoogle: authGoogle, language: Language.english);
    AIResponse response = await dialogflow.detectIntent(query);
    ChatMessage message = new ChatMessage(
        text: response.getMessage() ??
            new CardDialogflow(response.getListMessage()[0]).title,
        name: "Mr. X",
        type: false,
    );
    setState(() {
        _messages.insert(0, message);
    });
}

void _handleSubmitted(String text) {
    _textController.clear();
    ChatMessage message = new ChatMessage(
        text: text,
        name: this.user.name,
        type: true,
    );
    setState(() {
        _messages.insert(0, message);
    });
    response(text);
}

@override
Widget build(BuildContext context) {
    return new Column(
        children: <Widget>[
            new Flexible(
                child: new ListView.builder(
                    padding: new EdgeInsets.all(8.0),
                    reverse: true,
                    itemBuilder: (_, int index) => _messages[index],
                    itemCount: _messages.length,
                ),
            ),
            new Divider(height: 1.0),
            !this.isLoading
                ? CustomLoader()
                : Container(
                    decoration:
                        new BoxDecoration(color: Theme.of(context).cardColor),

```

```

        child: _buildTextComposer(),
      ),
    ],
  );
}

class ChatMessage extends StatelessWidget {
  ChatMessage({this.text, this.name, this.type});

  final String text;
  final String name;
  final bool type;

  List<Widget> otherMessage(context) {
    return <Widget>[
      new Container(
        margin: const EdgeInsets.only(right: 16.0),
        child: new CircleAvatar(child: new Text('B')),
      ),
      new Expanded(
        child: new Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            new Text(this.name,
              style: new TextStyle(fontWeight: FontWeight.bold)),
            new Container(
              margin: const EdgeInsets.only(top: 5.0),
              child: new Text(text),
            ),
          ],
        ),
      ),
    ];
  }

  List<Widget> myMessage(context) {
    return <Widget>[
      new Expanded(
        child: new Column(
          crossAxisAlignment: CrossAxisAlignment.end,
          children: <Widget>[
            new Text(this.name, style: Theme.of(context).textTheme.subhead),
            new Container(
              margin: const EdgeInsets.only(top: 5.0),
              child: new Text(text),
            ),
          ],
        ),
      ),
      new Container(
        margin: const EdgeInsets.only(left: 16.0),
        child: new CircleAvatar(
          child: new Text(
            this.name[0],
            style: new TextStyle(fontWeight: FontWeight.bold),
          ),
        ),
      ),
    ];
  }

  @override
  Widget build(BuildContext context) {
    return new Container(
      margin: const EdgeInsets.symmetric(vertical: 10.0),
      child: new Row(

```

```

        crossAxisAlignment: CrossAxisAlignment.start,
        children: this.type ? myMessage(context) : otherMessage(context),
      ),
    );
  }
}

```

#### <views/food.dart>

```

import 'package:eatfit/components/customCard.dart';
import 'package:eatfit/components/customLoader.dart';
import 'package:eatfit/models/user.dart';
import 'package:eatfit/util/db.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class Food extends StatefulWidget {
  const Food({Key key}) : super(key: key);

  @override
  _FoodState createState() => _FoodState();
}

class _FoodState extends State<Food> {
  bool isLoading = true;
  DatabaseService db = DatabaseService();
  User user;

  @override
  void initState() {
    super.initState();
    FirebaseAuth.instance.currentUser().then((fbuser) {
      db.getUser(fbuser.uid).then((user) {
        setState(() {
          this.user = user;
          this.isLoading = false;
        });
      });
    });
  }

  final recMeals = [
    'Eggs and Brown Bread',
    'Salad & Vegetable Stir Fry',
    'Soup and Fruits',
  ];

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: this.isLoading
        ? Center(
            child: CustomLoader(),
          )
        : Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              CustomCard(
                bgColor: Theme.of(context).primaryColor,
                content: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.center,
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.all(8.0),

```

```

        child: Text(
          "Meal Log",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            fontSize: 20,
            color: Theme.of(context).accentColor,
          ),
        ),
      ),
    ),
    Divider(
      color: Theme.of(context).accentColor,
      thickness: 2,
    ),
    ListView.separated(
      itemBuilder: (context, index) {
        return ListTile(
          title: Text(
            this.user.getMeals()[index].getMeal(),
            style: TextStyle(
              fontSize: 16,
              color: Theme.of(context).accentColor,
            ),
          ),
          subtitle: Text(
            new DateFormat.yMMMEd('en_US')
              .add_jms()
              .format(this
                .user
                .getMeals()[index]
                .getTimeOfConsumption())
              .toString(),
            style: TextStyle(
              color: Colors.grey.shade400,
            ),
          ),
          trailing: Text(
            this
              .user
              .getMeals()[index]
              .getCalorieValue()
              .toString(),
            style: TextStyle(
              color: Theme.of(context).errorColor,
              fontSize: 20,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
      ),
    );
  },
  itemCount: this.user.getMeals().length,
  shrinkWrap: true,
  separatorBuilder: (context, index) {
    return Divider(
      color: Theme.of(context).accentColor,
    );
  },
),
],
),
),
CustomCard(
  bgColor: Theme.of(context).primaryColor,
  content: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.center,
    children: <Widget>[

```

```

        Padding(
          padding: const EdgeInsets.all(12.0),
          child: Text(
            "Recommended Meals",
            style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 20,
              color: Theme.of(context).accentColor,
            ),
          ),
        ),
        Divider(
          color: Theme.of(context).accentColor,
          thickness: 2,
        ),
        ListView.separated(
          itemBuilder: (context, index) {
            return ListTile(
              title: Text(
                this.recMeals[index],
                style: TextStyle(
                  fontSize: 16,
                  color: Theme.of(context).accentColor,
                ),
              ),
            ),
          ),
          itemCount: 3,
          shrinkWrap: true,
          separatorBuilder: (context, index) {
            return Divider(
              color: Theme.of(context).accentColor,
            );
          },
        ),
      ],
    ),
    SizedBox(
      height: 30,
    ),
  ],
),
);
}
}

```

#### **<views/maps.dart>**

```

import 'dart:async';
import 'package:eatfit/components/customLoader.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:location/location.dart';

class Maps extends StatefulWidget {
  const Maps({Key key}) : super(key: key);

  @override
  _MapsState createState() => _MapsState();
}

class _MapsState extends State<Maps> {
  Completer<GoogleMapController> _controller = Completer();
  List<LatLng> _latlng = List();
  Set<Marker> _markers = Set();
  Set<Polyline> _polylines = Set();
}

```

```

LocationData _startLocation;
LocationData finishLocation;
Location location = new Location();
Timer timer;
bool isLoading = true;
bool isRunning = true;

@override
void initState() {
  super.initState();
  this.getStartPos();
  timer = Timer.periodic(
    Duration(seconds: 3),
    (Timer t) => this.getCurrentPos(),
  );
}

void getStartPos() async {
  LocationData start = await location.getLocation();
  setState(() {
    this._startLocation = start;
    this._markers.add(
      Marker(
        markerId: MarkerId(start.toString()),
        position: LatLng(start.latitude, start.longitude),
      ),
    );
    this.isLoading = false;
  });
}

void getCurrentPos() async {
  LocationData curr = await location.getLocation();
  setState(() {
    this._latlng.add(LatLng(curr.latitude, curr.longitude));
    if (this._polylines.isEmpty) {
      this._polylines.add(
        Polyline(
          color: Theme.of(context).primaryColor,
          polylineId: PolylineId('runroute'),
          points: this._latlng,
        ),
      );
    } else {
      this._polylines.clear();
      this._polylines.add(
        Polyline(
          color: Theme.of(context).primaryColor,
          polylineId: PolylineId('runroute'),
          points: this._latlng,
        ),
      );
    }
  });
}

void completeRun() async {
  LocationData end = await location.getLocation();
  setState(() {
    this.finishLocation = end;
    this._markers.add(
      Marker(
        markerId: MarkerId(end.toString()),
        position: LatLng(end.latitude, end.longitude),
      ),
    );
    this.isRunning = false;
  });
}

```



```

    });
    timer?.cancel();
  }

  @override
  void dispose() {
    timer?.cancel();
    super.dispose();
  }

  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: !isLoading
          ? Stack(
              alignment: Alignment.bottomCenter,
              children: [
                GoogleMap(
                  mapToolbarEnabled: true,
                  zoomGesturesEnabled: true,
                  polylines: _polylines,
                  markers: _markers,
                  onMapCreated: (GoogleMapController controller) {
                    _controller.complete(controller);
                  },
                  myLocationEnabled: true,
                  initialCameraPosition: CameraPosition(
                    target: LatLng(this._startLocation.latitude,
                      this._startLocation.longitude),
                    zoom: 16.0,
                  ),
                  mapType: MapType.normal,
                ),
                isRunning
                  ? Padding(
                      padding: const EdgeInsets.all(20),
                      child: FloatingActionButton(
                        onPressed: () => this.completeRun(),
                        child: Icon(
                          Icons.cancel,
                          color: Theme.of(context).primaryColor,
                        ),
                      ),
                )
              ],
            )
          : Padding(
              padding: const EdgeInsets.all(20),
              child: FloatingActionButton(
                onPressed: () => print('Start New!'),
                child: Icon(
                  Icons.directions_run,
                  color: Theme.of(context).primaryColor,
                ),
            ),
          ),
      ),
    ),
  );
}

```

```

<views/snap.dart>
import 'dart:async';
import 'dart:io';
import 'package:eatfit/components/customLoader.dart';

```

```

import 'package:eatfit/models/user.dart';
import 'package:eatfit/util/api.dart';
import 'package:eatfit/util/db.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:eatfit/components/customCard.dart';

class Snap extends StatefulWidget {
  final int value;

  const Snap({Key key, this.value}) : super(key: key);

  @override
  _SnapState createState() => _SnapState();
}

class _SnapState extends State<Snap> {
  File _image;
  API server = API();
  DatabaseService db = DatabaseService();
  bool isLoading = true;
  String prediction = '';
  User user;
  TextEditingController _calorieController = new TextEditingController();
  String calories;

  @override
  initState() {
    super.initState();
    FirebaseAuth.instance.currentUser().then((user) {
      db.getUser(user.uid).then((currUser) {
        setState(() {
          this.user = currUser;
        });
      });
    });
    getImage(this.widget.value);
  }

  void predict() async {
    String res = await server.predictFood(this._image);
    setState(() {
      this.prediction = res;
      this.isLoading = false;
    });
  }

  void updateMeal() {
    calories = _calorieController.text;
    _calorieController.clear();
    Map m = {
      "name": this.prediction,
      "calories": int.parse(calories),
      "time": DateTime.now()
    };
    List l = [m];
    db.addMeal(l, this.user.id);
    int newCal = this.user.currentCalories + int.parse(calories);
    db.updateUser({"currentCalories": newCal}, this.user.id);
    Navigator.popAndPushNamed(context, 'food');
  }

  Future getImage(int value) async {
    switch (value) {
      case 0:
    }
  }

```

```

        var image = await ImagePicker.pickImage(source: ImageSource.camera);
        setState(() {
          _image = image;
        });
        break;
      case 1:
        var image = await ImagePicker.pickImage(source: ImageSource.gallery);
        setState(() {
          _image = image;
        });
        break;
      default:
        print("Wrong Choice");
    }
  }
}

Widget _input(
  Icon icon, String hint, TextEditingController controller, bool obscure) {
  return Container(
    padding: EdgeInsets.only(left: 20, right: 20),
    child: TextField(
      controller: controller,
      obscureText: obscure,
      style: TextStyle(
        fontSize: 20,
      ),
      decoration: InputDecoration(
        hintStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 20),
        hintText: hint,
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(30),
          borderSide: BorderSide(
            color: Theme.of(context).primaryColor,
            width: 2,
          ),
        ),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(30),
          borderSide: BorderSide(
            color: Theme.of(context).primaryColor,
            width: 3,
          ),
        ),
        prefixIcon: Padding(
          child: IconTheme(
            data: IconThemeData(color: Theme.of(context).accentColor),
            child: icon,
          ),
          padding: EdgeInsets.only(left: 30, right: 10),
        ),
      ),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    child: new CustomCard(
      bgColor: Colors.grey.shade100,
      content: new Container(
        margin: EdgeInsets.all(10),
        padding: EdgeInsets.fromLTRB(5, 5, 5, 5),
        child: new Column(
          mainAxisAlignment: MainAxisAlignment.start,
          children: <Widget>[

```

```

Center(
  child: _image == null
    ? Text(
        'No image selected!\nPlease snap your food again!',
        style: TextStyle(
          color: Theme.of(context).errorColor,
          fontWeight: FontWeight.bold,
        ),
        textAlign: TextAlign.center,
      )
    : Image.file(_image),
),
Padding(
  child: RaisedButton(
    onPressed: () => this.predict(),
    child: Text(
      "Predict",
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
    color: Theme.of(context).primaryColor,
  ),
  padding: EdgeInsets.all(5),
),
Padding(
  child: this.isLoading
    ? CustomLoader()
    : Text(
        this.prediction,
        style: TextStyle(
          fontSize: 20,
        ),
      ),
  padding: EdgeInsets.all(10),
),
_input(Icon(Icons.fastfood), "Calories", this._calorieController,
  false),
Padding(
  child: RaisedButton(
    onPressed: () => this.updateMeal(),
    child: Text(
      "Add To Log",
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
    color: Theme.of(context).primaryColor,
  ),
  padding: EdgeInsets.all(5),
),
],
),
),
),
);
}
}

```

**<views/exerciseHome.dart>**

```

import 'package:eatfit/models/exercise.dart';
import 'package:eatfit/util/exerciseList.dart';
import 'package:flutter/material.dart';

```

```

class ExerciseHome extends StatelessWidget {
  const ExerciseHome({Key key}) : super(key: key);

  Widget cards(context, Exercise exercise) {
    return GestureDetector(
      onTap: () {
        Navigator.pushNamed(context, 'exercise/' + exercise.getID());
      },
      child: Container(
        height: 200,
        width: 200,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(30),
          boxShadow: [
            BoxShadow(
              color: Colors.grey,
              blurRadius: 6.0,
            ),
          ],
          color: Colors.grey.shade100,
        ),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Image.asset(
                exercise.getImage(),
                height: 80,
              ),
              SizedBox(
                height: 5,
              ),
              Text(
                exercise.getName(),
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 18),
              ),
              Container(
                padding: EdgeInsets.all(5),
                margin: EdgeInsets.only(top: 4),
                color: Color(0xFF92DCE5),
                child: Text(
                  "x " + exercise.getReps(),
                  style: TextStyle(
                    color: Colors.black,
                    fontWeight: FontWeight.bold,
                    fontSize: 12,
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    List<Exercise> exercises = exerciseList;
    return SafeArea(
      child: Stack(
        children: <Widget>[
          Container(
            height: 300,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.only(

```

```

        bottomLeft: Radius.circular(30),
        bottomRight: Radius.circular(30),
      ),
      color: Theme.of(context).primaryColor,
    ),
    width: double.infinity,
  ),
  Container(
    margin: EdgeInsets.only(left: 90, bottom: 20),
    width: 299,
    height: 279,
    decoration: BoxDecoration(
      color: Colors.blueAccent.shade200,
      // color: Theme.of(context).accentColor,
      borderRadius: BorderRadius.only(
        topLeft: Radius.circular(160),
        bottomLeft: Radius.circular(290),
        bottomRight: Radius.circular(160),
        topRight: Radius.circular(10),
      ),
    ),
  ),
),
CustomScrollView(
  slivers: <Widget>[
    SliverToBoxAdapter(
      child: Padding(
        padding: const EdgeInsets.all(26.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.min,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Text(
              "Today's Exercise",
              style: TextStyle(
                fontSize: 32,
                fontWeight: FontWeight.w800,
                color: Colors.white,
              ),
            ),
            Text(
              "Routine",
              style: TextStyle(
                fontSize: 32,
                fontWeight: FontWeight.w800,
                color: Colors.white,
              ),
            ),
            SizedBox(
              height: 30,
            ),
          ],
        ),
      ),
    ),
  ),
  SliverPadding(
    padding: const EdgeInsets.all(26.0),
    sliver: SliverGrid.count(
      crossAxisCount: 2,
      mainAxisSpacing: 10,
      crossAxisSpacing: 10,
      children: <Widget>[
        cards(
          context,
          exercises[0],
        ),
        cards(

```

```

        context,
        exercises[1],
      ),
      cards(
        context,
        exercises[2],
      ),
      cards(
        context,
        exercises[3],
      ),
      cards(
        context,
        exercises[4],
      ),
      cards(
        context,
        exercises[5],
      ),
    ],
  ),
),
SliverToBoxAdapter(
  child: Padding(
    padding: const EdgeInsets.fromLTRB(30, 20, 30, 30),
    child: RaisedButton(
      onPressed: () => Navigator.of(context).pushNamed('maps'),
      color: Theme.of(context).primaryColor,
      child: Text(
        'Go For A Run',
        style: TextStyle(
          color: Theme.of(context).accentColor,
        ),
      ),
    ),
  ),
),
),
],
),
],
),
);
}
}

```

#### <views/doExercise.dart>

```

import 'package:eatfit/components/customCard.dart';
import 'package:eatfit/models/exercise.dart';
import 'package:eatfit/util/exerciseList.dart';
import 'package:flutter/material.dart';
import 'package:audioplayers/audio_cache.dart';
import 'package:audioplayers/audioplayers.dart';

class DoExercise extends StatefulWidget {
  final String id;
  const DoExercise({Key key, this.id}) : super(key: key);

  @override
  _DoExerciseState createState() => _DoExerciseState();
}

class _DoExerciseState extends State<DoExercise> {
  Duration _duration = new Duration();
  Duration _position = new Duration();
  AudioPlayer advancedPlayer;
  AudioCache audioCache;

```

```

@override
void initState() {
  super.initState();
  initPlayer();
}

void initPlayer() {
  advancedPlayer = new AudioPlayer();
  audioCache = new AudioCache(fixedPlayer: advancedPlayer);

  advancedPlayer.durationHandler = (d) => setState(() {
    _duration = d;
  });

  advancedPlayer.positionHandler = (p) => setState(() {
    _position = p;
  });
}

void seekToSecond(int second) {
  Duration newDuration = Duration(seconds: second);

  advancedPlayer.seek(newDuration);
}

Widget _tab(List<Widget> children) {
  return Center(
    child: Container(
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          slider(),
          Row(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: children
              .map((w) => Container(child: w, padding: EdgeInsets.all(6.0)))
              .toList(),
          ),
        ],
      ),
    );
}

Widget _btn(String txt, IconData icon, VoidCallback onPressed) {
  return RaisedButton.icon(
    onPressed: onPressed,
    color: Theme.of(context).primaryColor,
    icon: Icon(
      icon,
      color: Theme.of(context).accentColor,
    ),
    label: Text(
      txt,
      style: TextStyle(
        fontSize: 16,
        color: Theme.of(context).accentColor,
      ),
    ),
  );
}

```

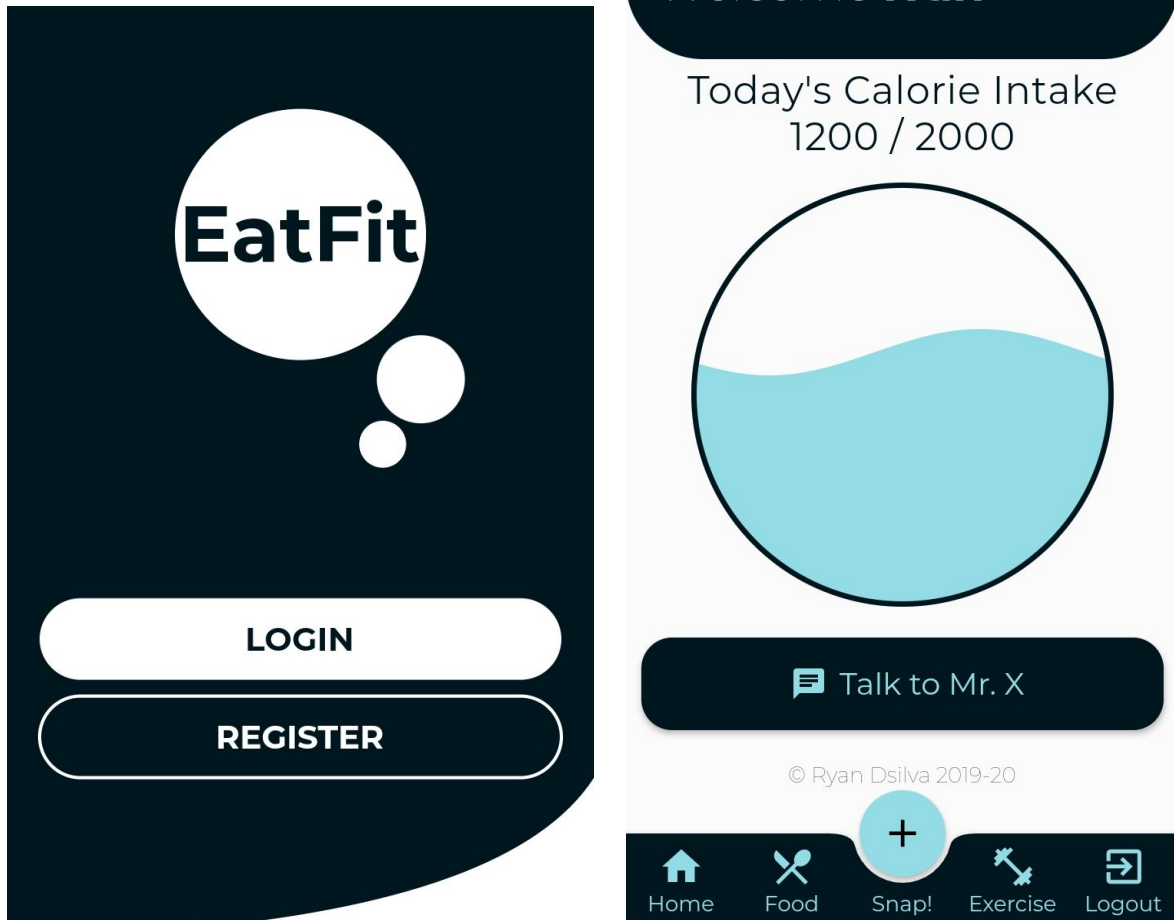


```

@override
Widget build(BuildContext context) {
  final Exercise exercise = exerciseList[int.parse(this.widget.id)];
  return SingleChildScrollView(
    child: Center(
      child: CustomCard(
        bgColor: Colors.grey.shade100,
        content: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Text(
                exercise.getName(),
                style: TextStyle(
                  color: Theme.of(context).primaryColor,
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
            Divider(),
            Image.asset(
              exercise.getImage(),
              width: 300,
              height: 300,
            ),
            SizedBox(
              height: 15,
            ),
            Divider(),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Text(
                "Instructions",
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
            Divider(),
            localAsset(exercise.getAudio()),
            Divider(),
            Chip(
              avatar: Icon(
                Icons.offline_bolt,
                color: Colors.white,
              ),
              label: Padding(
                padding: const EdgeInsets.fromLTRB(0, 4, 4, 4),
                child: Text(
                  "Repetitions: " + exercise.getReps(),
                  style: TextStyle(
                    fontSize: 20,
                    color: Colors.white,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}


```


## App Screenshots




×

REGISTER

 NAME


 EMAIL


 PASSWORD

REGISTER

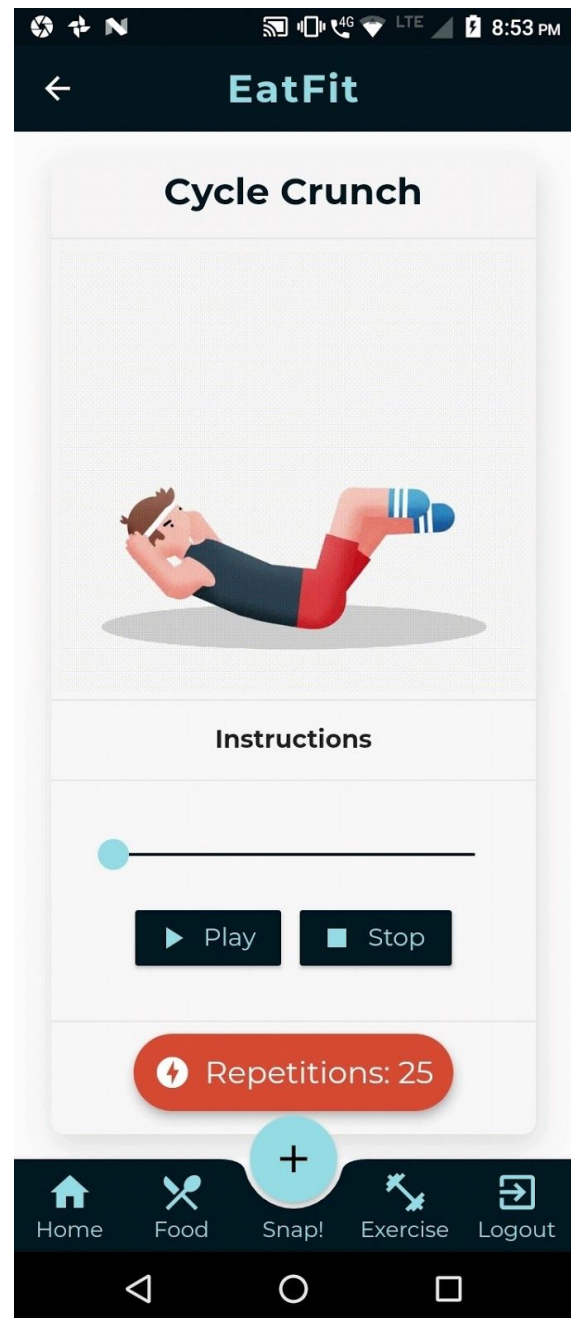
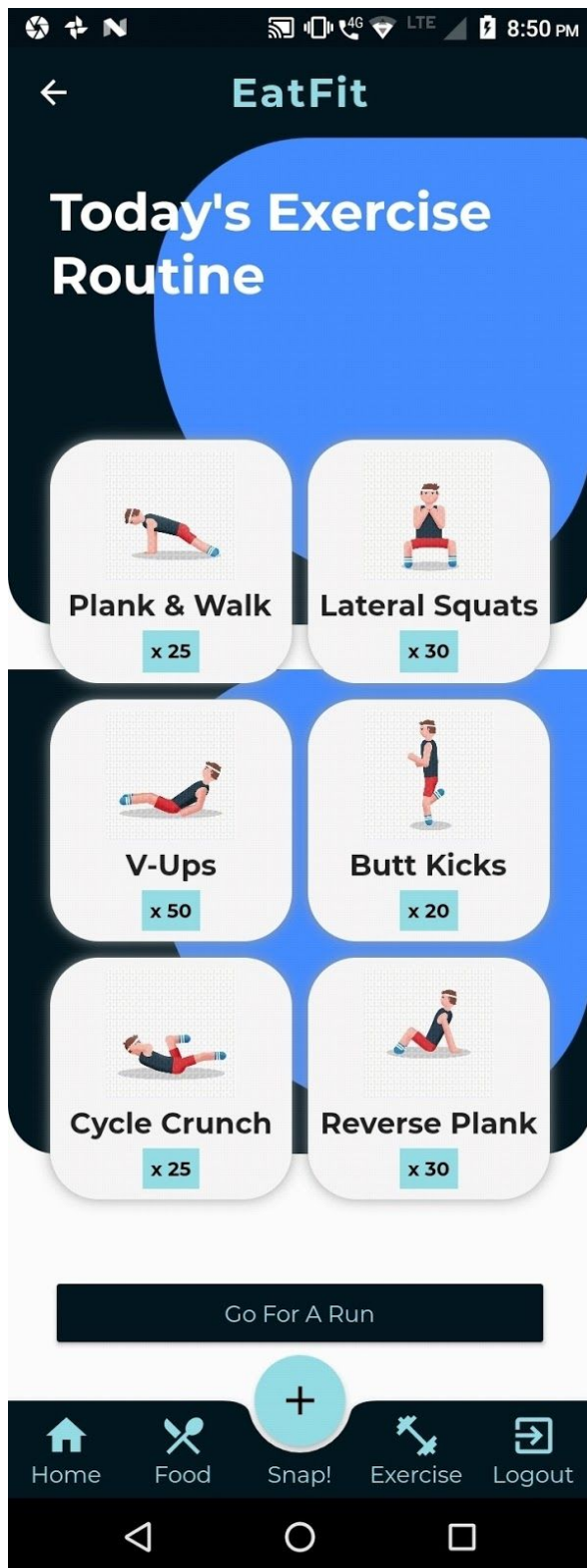
×

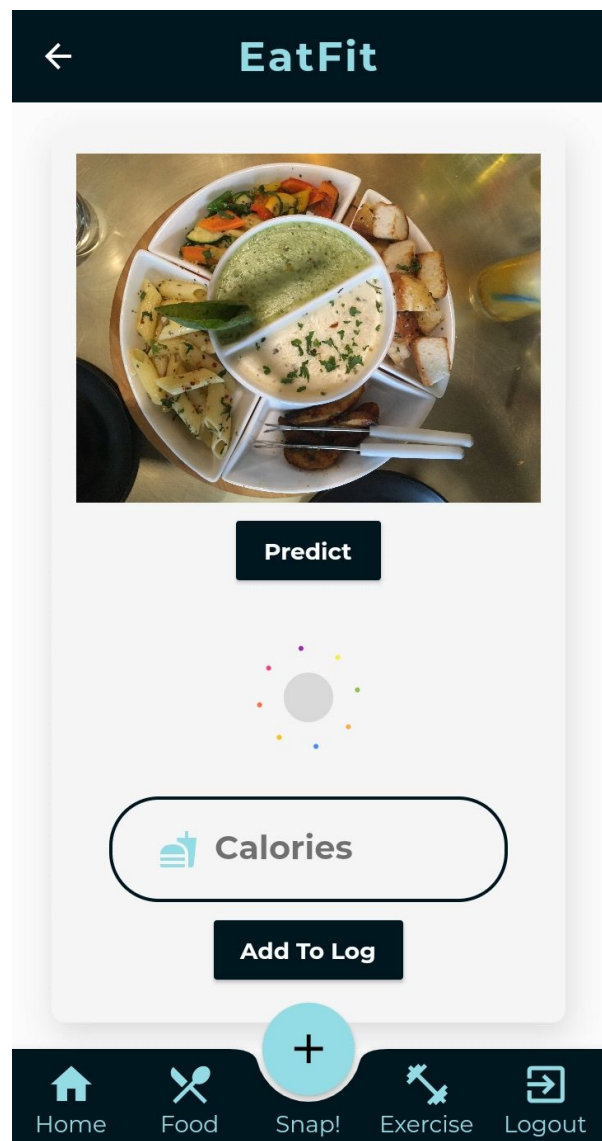
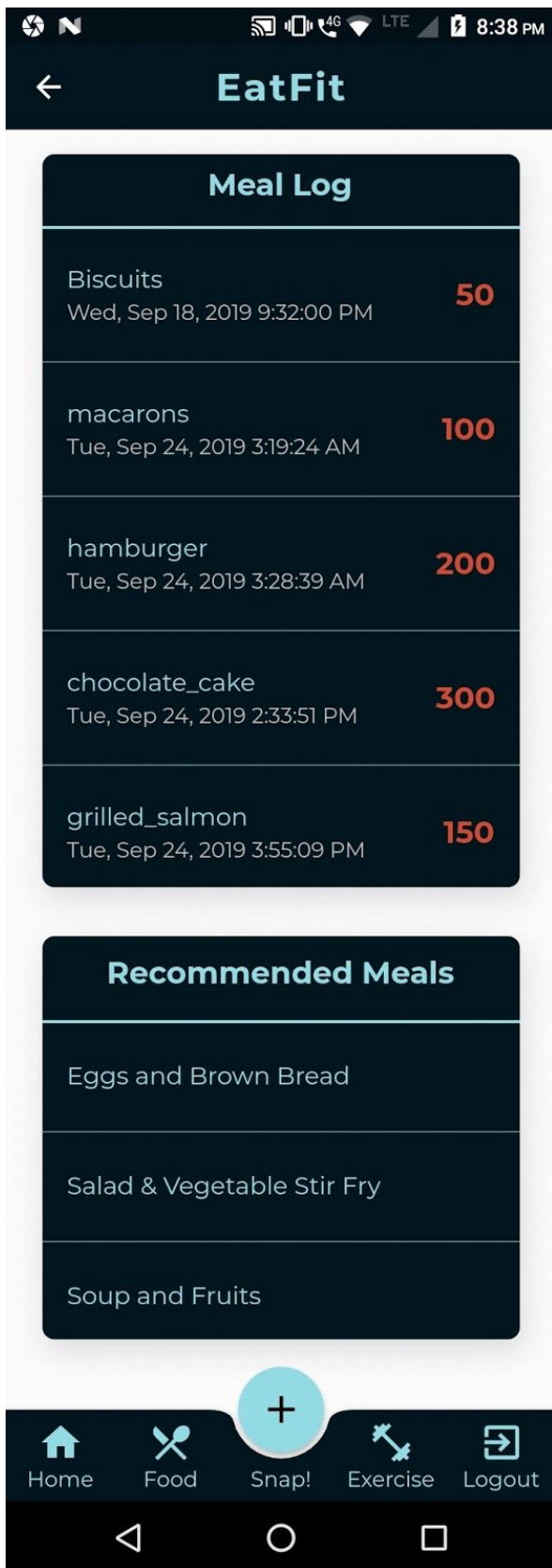
LOGIN

 EMAIL

 PASSWORD

LOGIN







## References

- Flutter Official Documentation, Google - [<https://flutter.dev/docs>]
- Flutter Cookbook, Google - [<https://flutter.dev/docs/cookbook>]
- Flutter For Android Devs, Google - [<https://flutter.dev/docs/get-started/flutter-for/android-devs>]
- Flutter Succinctly, Ed Freitas, Syncfusion Publications - [<https://www.syncfusion.com/ebooks/flutter-succinctly>]
- Flutter Tutorial, TutorialsPoint - [<https://www.tutorialspoint.com/flutter/>]
- Flutter Technical Overview, Google - [<https://flutter.dev/docs/resources/technical-overview>]
- Firebase Documentation, Google - [<https://firebase.google.com/docs/flutter/>]
- Firebase Tutorial, TutorialsPoint - [<https://www.tutorialspoint.com/firebase/>]
- EatFit, Ryan Dsilva - GitHub - [<https://github.com/RyanDsilva/eatfit>]