(Time Token Generator)

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

(Shubham Gupta (161322))

Under the supervision of

(Mr. Surjeet Singh)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Token Generator"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from May 2020 to June 2020 under the supervision of **Surjeet Singh** Computer Science Department.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shubham Gupta, 161322.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Mr. Surjeet Singh
Designation: Assistant Professor (Grade-I)
Department name: CSE&IT
Dated:

ACKNOWLEDGEMENT

I would like to take the opportunity to thank and express my deep sense of gratitude to my
mentor and project guide Mr. Surjeet Singh for his immense support and valuable guidance
without which it would not have been possible to reach at this stage of our minor project.

We have grown both academically and personally from this experience and are very grateful
for having had the opportunity to conduct this study

I am also obliged to all my faculty members for their valuable support in their respective fields which helped me in reaching at this stage of my project.

Finally, we like to thank our family and friends for their constant support. Without their contribution it would have been impossible to complete our work.

Date:

Contents

Chapter 1 INTRODUCTION 6
1.1 Introduction 8
1.2 Problem Statement 9
1.3 Objectives 9
1.4 Methodology 10
Chapter 2 LITERATURE SURVEY 10
2.1 Dart 10
2.1.1 dart:core library 10
2.2 Flutter 14
2.2.1 Features 16
2.2.2 Advanteges 17
2.2.3 Widgets 17
2.2.4 Flutter Framework 19
2.2.5 User Interaction 20
2.3 Flutter Layout 21
2.3.1 Create and Display a widget 23
2.3.2 Layout for Multiple Widget 24
2.3.3 Sizing of widget 26
2.3.4 Nesting of Rows and Columns 29
2.3.5 Common Layout widgets 30
2.4 Packages in Flutter 31
2.4.1 http Package 32
2.4.2 Location Package 33
2.5 API (Application Programming Interface) 34
2.5.1 Google Maps API 35
2.5.2 Places API 36
2.5.3 Directions API 37
2.6 Firebase 37
2.6.1 Cloud Firestore: 37
Chapter 3 SYSTEM DEVELOPMENT 39
3.1 Silvers 39

3.2 OnPressed 40
3.3 Location 42
3.4 Navigate to next screen 43
3.4 Algorithm 48
3.8 Hardware Requirements 50
3.9 Software Requirements 50
Chapter 4 PERFORMANCE ANALYSIS 51
4.1 Home Page of the Application 51
4.2 Add Time 53
4.3 Result Page 54
Chapter 5 CONCLUSIONS 57
5.1 Conclusions 57
5.2 Future Work 57
5.2.1 Application 57
References 58

List of Figures

1. Construction of variables from literals [3]
2. Assigning a sequence of characters to String variable[3]
3. Regular expression that match up with a string having 1 or more digits[3]
4. Example of List[3]
5. Example of Sets[3]
6. Example of Maps[3]
7. Example to create DateTime object[3]
8. Example to create duration object specifying the individual time units. [3]
9. Example of uri object[3]
10. Flutter Application Example (Android)[6]
11. Flutter Application Example (iOS)[6]
12. Broad class hierarchy of Widgets[6]
13. Flutter System overview[7]
14. Example of Stateful widget for user interaction[15]
15. Screeen Grab of a simple flutter UI
16. Widget Tree for above UI [7]
17. Text visible widget is incorporated in center layout widget[7]
18. Layout widget in body of Scaffold[7]
19. Layout widget using Column and row[9]
20. Alignment of widgets[9]
21. Overflowing row in a widget resulting a yellow and black striped pattern[9]
22. Widget Wrapped in Expanded to prevent overflowing
23. Widget Wrapped in Expanded with Flex: 2 to double its size[9]
24. . Nesting of widgets marked with box with red outline[9]
25. The widget tree for nesting of rows and columns[9]
26. Example of Usage of http Package[17]
27. Example on usage of Location package in flutter [16]

28. Example of API[12]
29. Example of Cloud Firestore[19]
30. Home Page of the application (home.dart {1})
31. Home Page of the application (home.dart {2})
32. Home Page of the application (home.dart {3})
33. Code to retrieve location from device (location.dart)
34. code to ask user to add time he will spend addTime.dart{1}
35. code to ask user to add time he will spend addTime.dart{2}
36. code to ask user to add time he will spend addTime.dart{3}
37. Algorithm to find Time user reaches and leaves the store{1}
38. Algorithm to find Time user reaches and leaves the store{2}
39. Snapshots of application with home page(when location is not available)
40. Snapshots of application with home page(when location is available)
41. Snapshots of application with page to add max time user will spend at the store
42. Snapshots of application with result page giving details about your journey
43. Snapshots of application with result page giving details about your journey (With TIme)

Chapter 1 INTRODUCTION

1.1 Introduction

Dart is a client-based programming language for applications development on multiple platforms. It is created by Google and is utilized to make mobile applications, desktop applications, web applications and servers.[1]

Dart is an class based and object-oriented language with garbage collection language with C like syntax. It supports abstracts classes, interfaces, reified generics, type inference and mixins.[1]

Dart SDK comes with the Dart VM as its compiler. The Dart-SDK also includes a utility tool - dart2js, which is a transpiler that is used to convert dart code to JavaScript code by generating JavaScript equivalent of a Dart Script.[2]

Dart has a rich set of core libraries that gives us essential code for lots of daily programming tasks like (dart:collection) helps in working on collections of objects, (dart:math) is used for making calculations and (dart:convert) is for encoding and decoding data.[3]

Dart has following core libraries:

dart:core: They are theBuilt-in types, assortments, and other center usefulness. This library is consequently brought into each Dart program.

dart:io: I/O for programs that can use the Dart VM, including servers, Shudder applications, and request line scripts.

dart:html: DOM and distinctive APIs for program based applications. HTML parts and various resources for online applications that need to team up with the program and the DOM (Archive Item Model)

dart:math: Numerical constants and limits, notwithstanding a self-assertive number generator.

dart:convert: Encoders and decoders for changing over between different data depictions, including JSON and UTF-8.

dart:async: Includes support for offbeat programming it incorporates classes like Stream and Future.
[4]

1.2 Problem Statement

In this project, i would like to make an application that works on multiple platforms and explore various ways to reduce the number of people in a local shop. In this application we take input from the user on the nearby local shop user want to visit and approximate time user will take to shop the desired item. App will respond to user with the time he needs to leave his house which will include the travel time for the user to reach the shop such that crowd inside the shop could be controlled.

1.3 Objectives

Our objective is to achieve an effective way to reduce number of people at a particular shop with least impact on owner’s business. We also explore newer and better ways of crowd reducing where people can gather.

1.4 Methodology

The idea behind this project is to use Flutter toolkit to create an application that works on multiple platform and can have native like performance in real world scenario. This Application takes time user could spend in a local store and find the time he should leave his residence and time he would be done from shopping using this data, we can calculate number of people in a store and reduce the crowd in store by assigning time for user to leave the store and we can later in future implement ML algorithms to automatically guess the time taken by user to shop a particular item.

Chapter 2 LITERATURE SURVEY

2.1 Dart

Dart is an open-source widely accommodating programming language. It is from the start progressed by Google. Dart is another coding language proposed for the server correspondingly as the program. Presented by Google, the Dart SDK ships with its compiler which is called DartVM The SDK in like way combines an utility - dart to js, a transpiler that makes js similitude a Dart Content. This instructional exercise gives a fundamental level comprehension of the Dart programming language.[1]

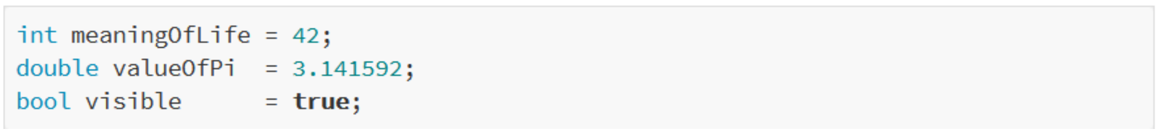
2.1.1 dart:core library

This library includes built in types, collection, and other center usefulness for each Dart program. This library is naturally imported in every dart program.

A few classes in this library, for example, String and num, and support Dart's built in information types. Different classes, for example, List and Map, give information structures to overseeing assortments of items. What's more, still different classes speak to generally utilized sorts of information, for example, URIs, dates and times, and blunders.[3]

2.1.1.1 Numbers and booleans

int and double offer help for Dart's worked in numerical data types: whole numbers and double precision floating point numbers, separately. An object of type bool is either obvious or bogus. variables of these sorts can be developed from literals.[3]



```
int meaningOfLife = 42;
double valueOfPi   = 3.141592;
bool visible       = true;
```

Figure 1. Construction of variables from literals[3]

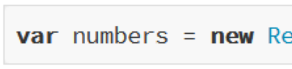
2.1.1.2 Strings and regular expressions

String is immutable and characterizes an arrangement of characters. For example[3]

```
String s
```

Figure 2. Assigning a sequence of characters to String variable

RegExp executes Dart normal articulations, which give a syntax to coordinating examples inside content. For instance, here's a customary articulation that coordinates a string of at least one digits: [3]



```
var numbers = new Re
```

Figure 3. Regular expression that match up with a string having 1 or more digits[3]

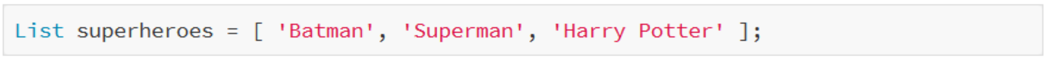
Dart regular expressions have a similar language structure and semantics as JavaScript normal articulations.

2.1.1.3 Collections

The dart:core library gives fundamental collections, for example, List, Map, and Set.

List:

A List is an arranged collection of items, with a length. Records are here and there called clusters. Utilize a List when you have to get to objects by file.

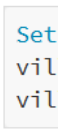


```
List superheroes = [ 'Batman', 'Superman', 'Harry Potter' ];
```

Figure 4. Example of List[3]

Set:

A Set is an unordered collection of various variables or objects. You can't get item from set by using index (position). Including a duplicating thing has no impact.



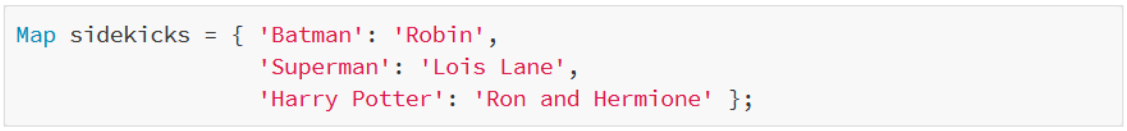
```
Set
vil
vil
```

Figure 4. Example of Sets[3]

Map:

A Map is an unordered collection of key-esteem sets. Maps are some of the time called cooperative exhibits since maps partner a key to some an incentive for simple recovery.

Keys are one of a kind. Utilize a Map when you have to get to objects by a one of a kind identifier.[3]



```
Map sidekicks = { 'Batman': 'Robin',
                  'Superman': 'Lois Lane',
                  'Harry Potter': 'Ron and Hermione' };
```

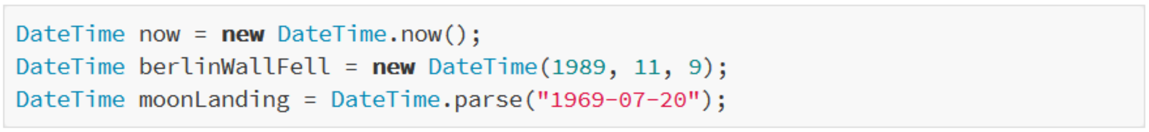
Figure 6. Example of Maps[3]

Notwithstanding these classes, dart:core contains Iterable, an interface that characterizes usefulness regular in collections of articles. Models remember the capacity to run a capacity for every component in the collection, to apply a test to every component, to recover an article, and to decide length.[2]

2.1.1.4 Date and time

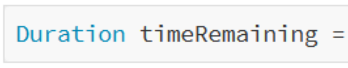
Use DateTime to speak to a point in time and Duration to speak to a range of time.

You can make DateTime objects with constructors or by parsing an effectively designed string.[3]



```
DateTime now = new DateTime.now();
DateTime berlinWallFell = new DateTime(1989, 11, 9);
DateTime moonLanding = DateTime.parse("1969-07-20");
```

Figure 7. Example to create DateTime object[3]

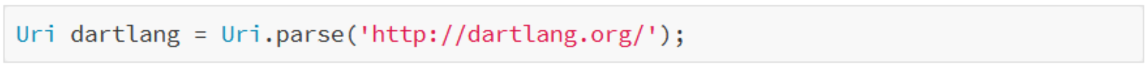


```
Duration timeRemaining =
```

Figure 8. Example to create duration object specifying the individual time units.[3]

2.1.1.5 Uri

A Uri object speaks to a uniform Resourse identifier, which recognizes an asset on the web.[3]



```
Uri dartlang = Uri.parse('http://dartlang.org/');
```

Figure 9. Example of uri object.[3]

2.1.1.6 Errors

The Error class speaks to the event of a error during runtime. Subclasses of this class speak to explicit sorts of errors.[4]

2.2 Flutter

Flutter is made by Google as a user interface toolkit for helping developers making lovely, natively compiled applications for versatile, web, and work area from a one single code for multiple platforms. [6] The objective is to empower developers to make applications with beautiful user interface that works as fast as native applications. We grasp contrasts in looking over practices, typography, symbols, and then some.[8] Flutter incorporates a cutting edge react- style framework, which is a 2D rendering engine, instant widgets, and development tools. These parts cooperate to enable you to configuration, fabricate, test, and troubleshoot applications. Everything is composed around a couple of center standards.[6]

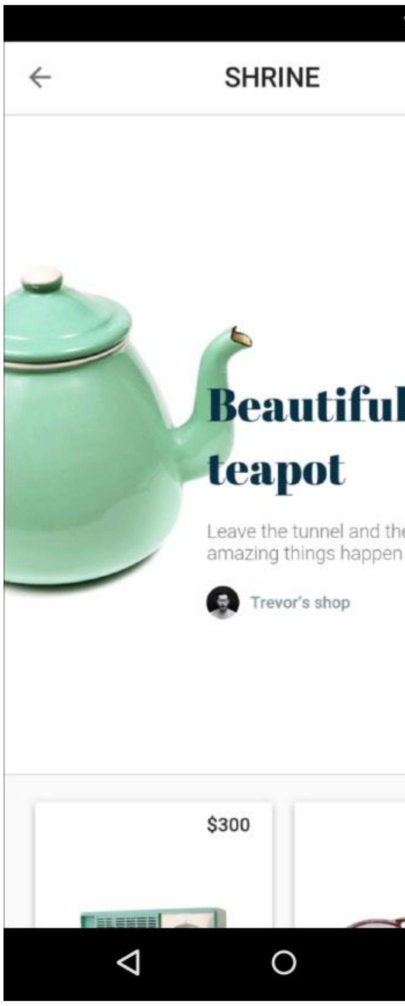


Figure 10. Flutter Application Example (Android).[6]

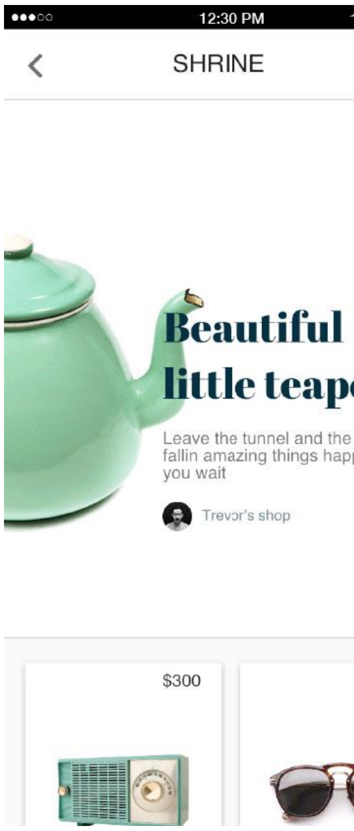


Figure 11. Flutter Application Example (iOS).[6]

2.2.1 Features

- Expressive, beautiful UIs

  Flutter's includes built in wonderful Material Design and Cupertino (iOS like design style) widgets, smooth common looking over, rich motion APIs and stage mindfulness.

- Fast development :

  Flutter has hot reload feature which helps you quickly and easily conduct tests with your app and build UIs, add features, and fix bugs faster without restarting app every time you change something in code.

- Native Performance

  Flutter's widgets includes all basic stage features like for example, scrolling,looking over, navigation, symbols and fonts to provide high performance on the Web, iOS, Android.[8]

2.2.2 Advanteges

- Be highly productive by creating iOS and Android application from single codebase
- Accomplish more with less code, even on a native OS, with an advanced, expressive language and a revelatory methodology
- Model and emphasize without any problem. Make changes to code and reloading your application as it runs on your device (with hot reload)
- Acknowledge custom, excellent, brand-driven plans, without the restrictions of OEM gadget sets
- Advantage from a rich arrangement of Material Design and Cupertino (iOS-flavor) gadgets manufactured utilizing Flutter's own system[8]

2.2.3 Widgets

In Flutter Everything is made up of a widget including flutter itself. Widgets are the essential structure squares of a Flutter application's USER INTERFACE. Every widget is an unchanging statement of part of the USER INTERFACE. Not at all like different frameworks that different perspectives, see controllers, formats, and different properties, Flutter has a reliable, bound together object model: the widget. Widgets structure a chain of importance dependent on arrangement. Every widget settles inside, and acquires properties from, its parent. There is no different "application" object. Rather, the root widget serves this job.

Widgets are themselves regularly made out of some other widgets that consolidate to deliver incredible impacts. For instance, Container, a regularly utilized widget, is comprised of a few widgets liable for design, painting, situating, and measuring. In particular, Container is comprised of LimitedBox, ConstrainedBox, Align, Padding, DecoratedBox, and Transform widgets.[6]

How Widgets are build:

You characterize the one of a kind qualities of a widget by executing a form() work that profits a tree (or progression) of widgets. This tree speaks to the widget's a piece of the USER INTERFACE in progressively solid terms. For instance, a toolbar widget may have a form work that profits an even design of some content and different catches. The framework at that point recursively solicits each from these widgets to work until the procedure bottoms out in completely solid widgets, which the framework at that point fastens together into a tree.

A widget's assemble capacity ought to be liberated from reactions. At whatever point it is approached to manufacture, the widget should restore another tree of widgets paying little mind to what the widget recently returned. The framework does the truly difficult work of contrasting the past form and the current form and figuring out what adjustments should be made to the USER INTERFACE.

This mechanized correlation is very powerful, empowering superior, intelligent applications. What's more, the plan of the assemble work rearranges your code by concentrating on announcing what a widget is made of, as opposed to the complexities of refreshing the USER INTERFACE starting with one state then onto the next. [6]

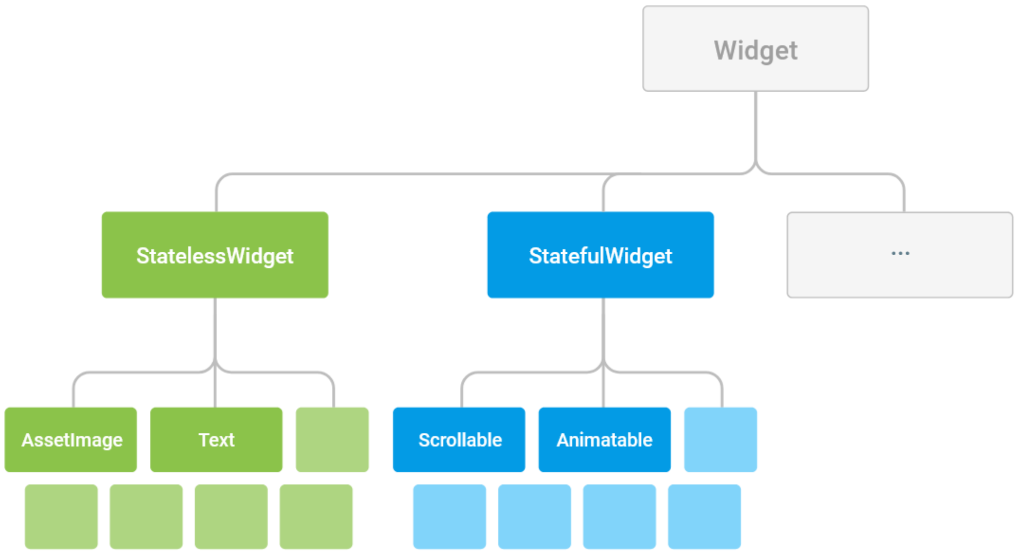


Figure 12 Broad class hierarchy of Widgets [6]

2.2.4 Flutter Framework

The Flutter framework is sorted out into a progression of layers, with each layer expanding upon the past layer.

The upper layers of the framework are utilized more every now and again than the lower layers. For the total arrangement of libraries that make up the Flutter's layered framework, see our API documentation.

The objective of this structure is to assist you with accomplishing more with less code. For instance, the Material layer is worked by making fundamental widgets from the widgets layer, and the widgets layer itself is worked by organizing lower-level objects from the rendering layer.

The layers offer numerous alternatives for building applications. Pick an altered way to deal with open the full expressive intensity of the framework, or use building hinders from the widgets layer, or blend and match. You can form the instant widgets Flutter gives, or make your own custom widgets utilizing similar tools and procedures that the Flutter group used to assemble the framework.[7]

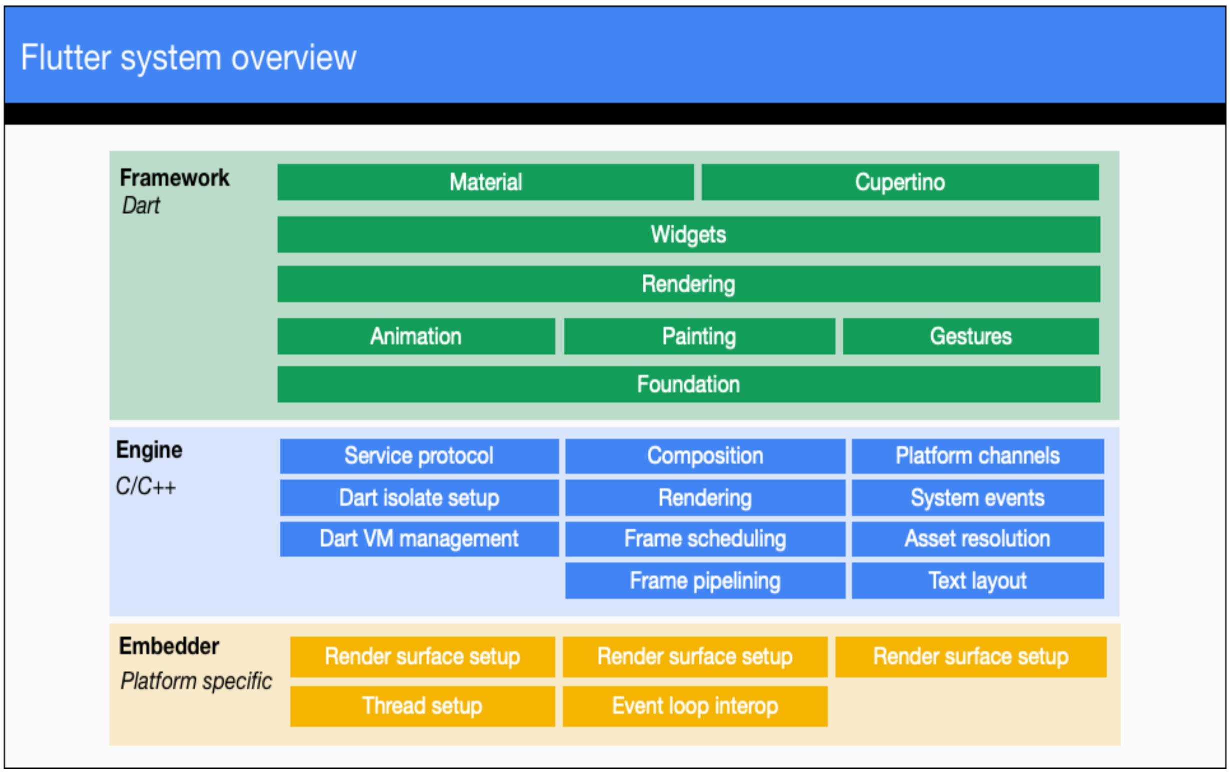


Figure 7 Flutter System overview [7]

2.2.5 User Interaction

On the off chance that the one of a kind attributes of a widget need to change dependent on client communication or different components, that widget is stateful. For instance, if a widget has a counter that increases at whatever point the client taps a catch, the estimation of the counter is the state for that widget. At the point when that worth changes, the widget should be reconstructed to refresh the USER INTERFACE.

These widgets subclass StatefulWidget (as opposed to StatelessWidget) and store their variable state in a subclass of State. [6]

At whatever point you mutate a State object (for instance, by augmenting the counter), you should call setState() to flag the framework to refresh the USER INTERFACE by calling the State's fabricate technique once more. For a case of overseeing state, see the MyApp format which is made with each new Flutter project.

Figure 14 Example of Stateful widget for user interaction [15]

2.3 Flutter Layout

The main part of anything in flutter is a widgets. Everything you see, every design pattern every layout every margin every padding, is a widget. The pictures, icons, and text that you find in a Flutter apps are also most part widgets. In Flutter, nearly everything is a widget-even design style/pattern are widgets.

You make a design by making widgets to amass progressively complex widgets. For example, the principal screen catch underneath shows 3 images with an imprint under each one portraying it's capacity:[7]

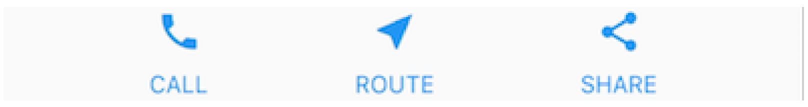


Figure 15 Screeen Grab of a simple flutter UI [7]

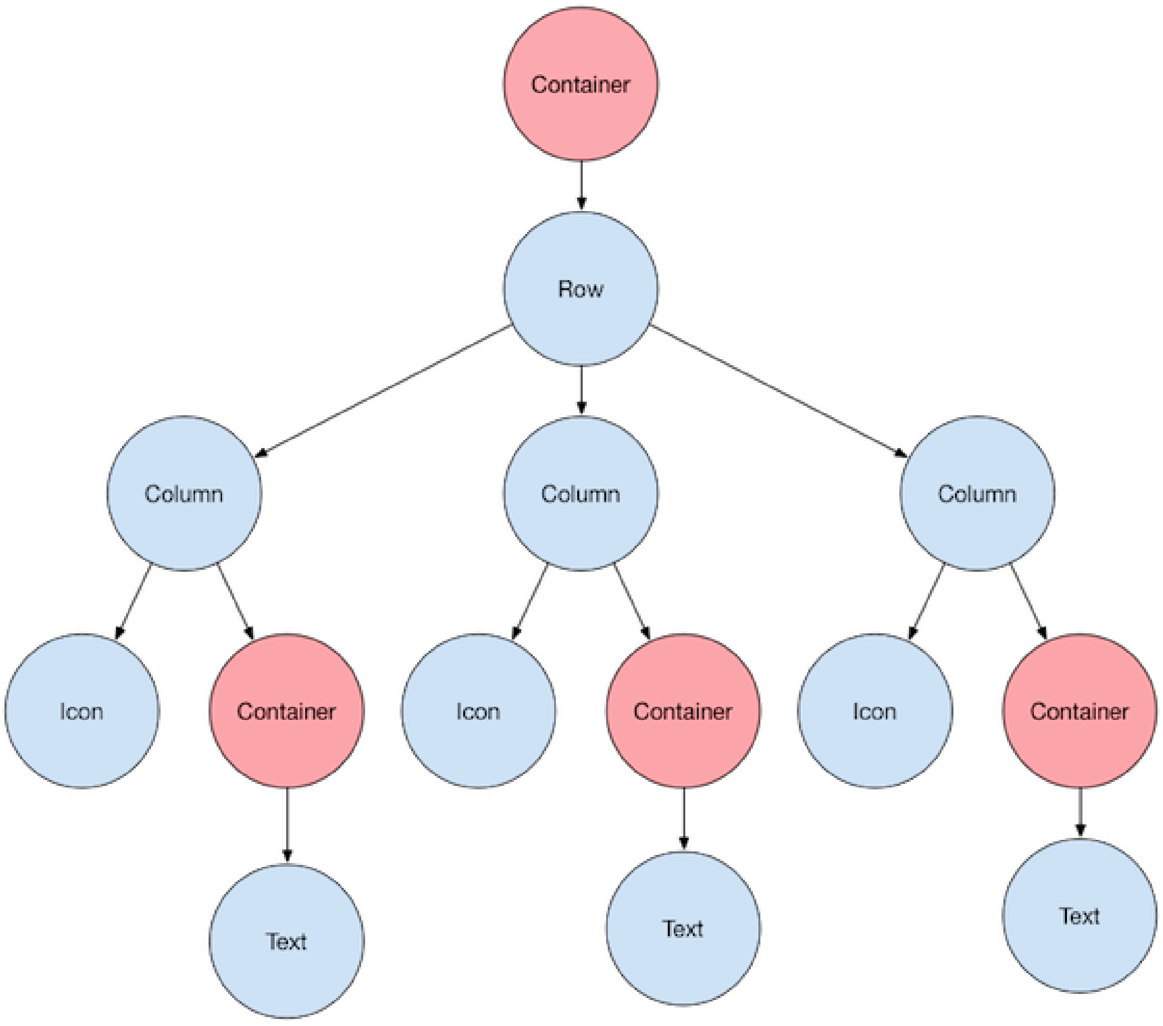


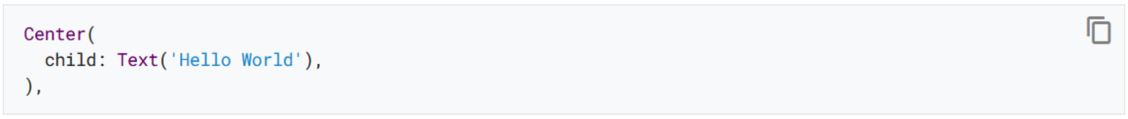
Figure 16 Widget Tree for above UI [7]

In this model, every Text widget is put in a Container to include edges. The whole Row is likewise positioned in a Container to include cushioning around the row.

Utilize the Text.style property to set the textual style, its shading, weight, etc. The remainder of the USER INTERFACE in this model is constrained by features. Set an Icon symbol's shading utilizing its shading feature. Columns and rows have properties that permit you to indicate how their youngsters are aligned vertically or evenly, and how much space the kids ought to possess.[7]

2.3.1 Create and Display a widget

1. Choose any widget with layout: You can opt from various different types layout widget provided by flutter based on alignment and constraints on your widget. These characters aare also incorporated in widget passed in layout widget.
2. Create visible widget: It displays text , image, icon or any other graphic item on your screen.
3. Incorporate visible widget in layout widget:



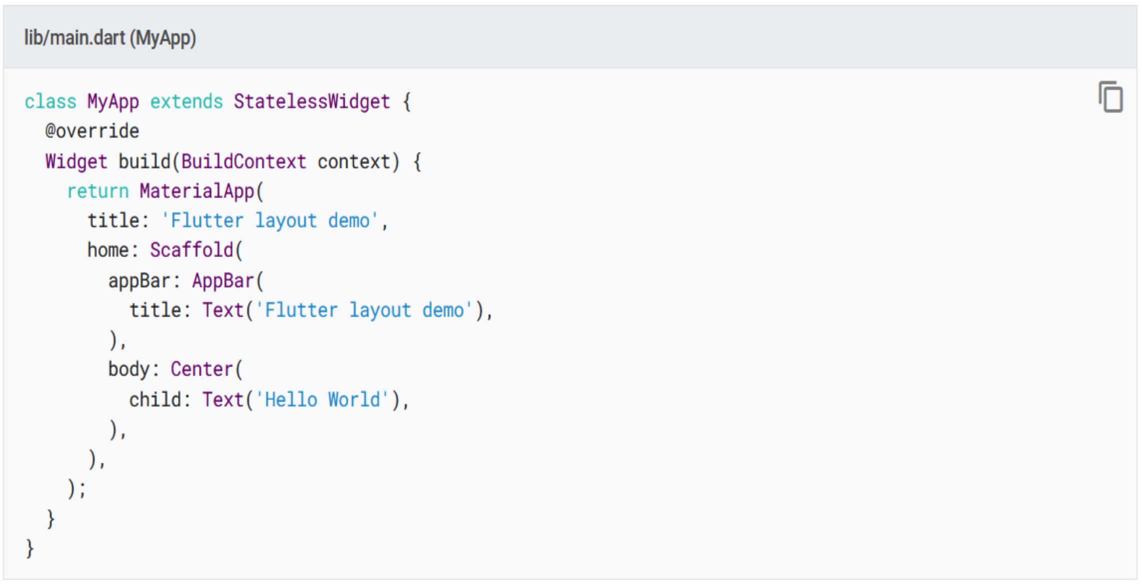
```
Center(
  child: Text('Hello World'),
),
```

4. **Figure 17 Text visible widget is incorporated in center layout widget [7]**

''

to Adding Layout widget in your screen: when you create flutter app it's a one widget, and and most widgets have a build ( ) strategy. Starting up and restoring a widget in the application's assemble() strategy shows the widget.

For a Material application, you can utilize a Framework widget; it gives a default standard, foundation shading, and has Programming interface for including drawers, café, and base sheets. At that point you can add the Middle widget legitimately to the body property for the landing page [7]


lib/main.dart (MyApp)

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter layout demo'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Figure 18 Layout widget in body of Scaffold[7]

2.3.2 Layout for Multiple Widget

1. Layout widgets vertical and on a level plane: One of the most notable structure plans is to organize widgets vertical or on a level plane. You can use a Line widget to arrange various widgets equally, and a Section widget to organize widgets vertically.

To make a line or segment in Flutter, you incorporate an overview of child widgets to a Line or Section widget. The going with model exhibits how it is possible to settle lines or sections inside lines or segments. This arrangement is sifted through as a Line. The line contains two kids: a segment on the left, and an image on the right. Along these lines, each youngster would itself have the option to be a line or section, and so forth. [8]

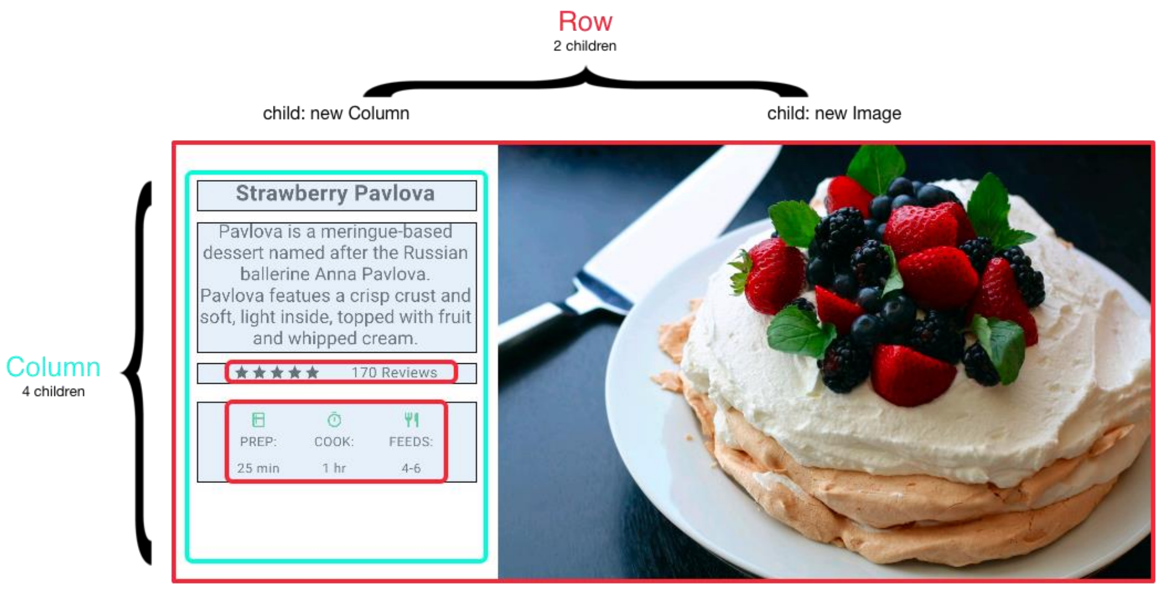

Row
2 children
child: new Column
child: new Image
Column
4 children
Strawberry Pavlova
Pavlova is a meringue-based dessert named after the Russian ballerine Anna Pavlova. Pavlova featues a crisp crust and soft, light inside, topped with fruit and whipped cream.
170 Reviews
PREP:
25 min
COOK:
1 hr
FEEDS:
4-6

Figure 19 Layout widget using Column and row [9]

1. Alignment of widget:

   mainAxisAlignment and crossAxisAlignment properties helps you to align widgets within column or row and also helps in maintaining space within widgets, they can be set to many values depending on hpw you would like your widgets to be displayed.

   Example in rows, the principle hub goes on a level plane and the cross pivot goes upright.

   Example in columns, the principle hub operates upright and the cross hub goes parallel.[9]

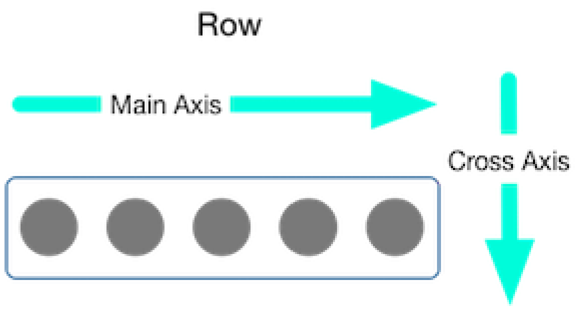


Figure 20 Alignment of widgets[9]

2.3.3 Sizing of widget

At the point when a design is too enormous to even think about fitting a gadget, a yellow and dark striped example shows up along the influenced side. Here is a case of a row which is excessively wide

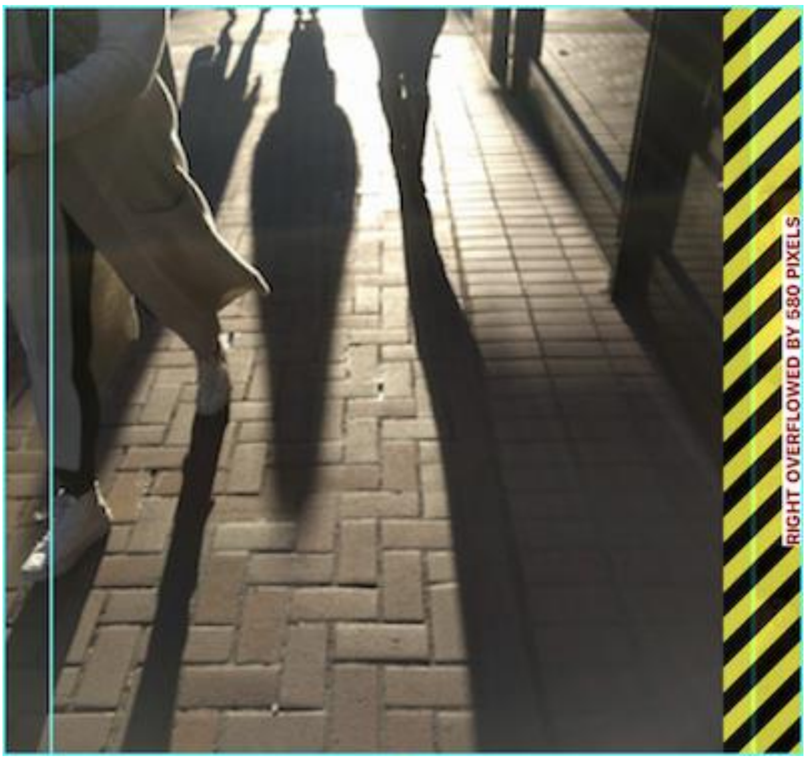
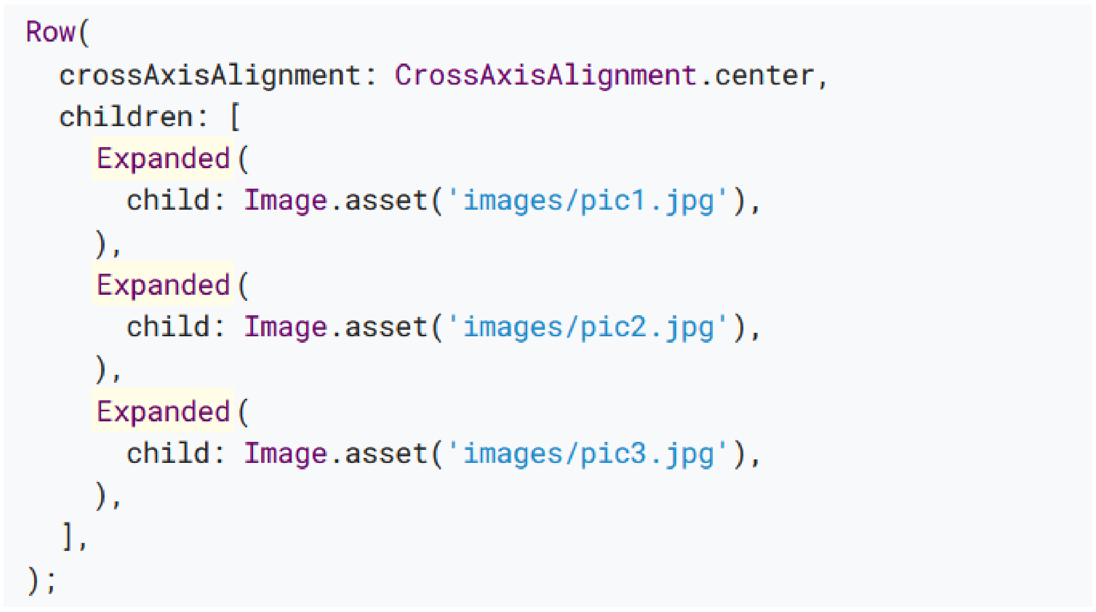


Figure 21. Overflowing row in a widget resulting a yellow and black striped pattern [9]

You can use Expanded widget to fit various different widgets in rows and columns and prevent them from leaking through your screen. In past, where the line of pictures is excessively wide for its renderer, create each image included within an Extended widget. Widgets can be estimated to fit inside a line or segment by using the Extended widget. [9]



```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

Figure 22. Widget Wrapped in Expanded to prevent overflowing[9]

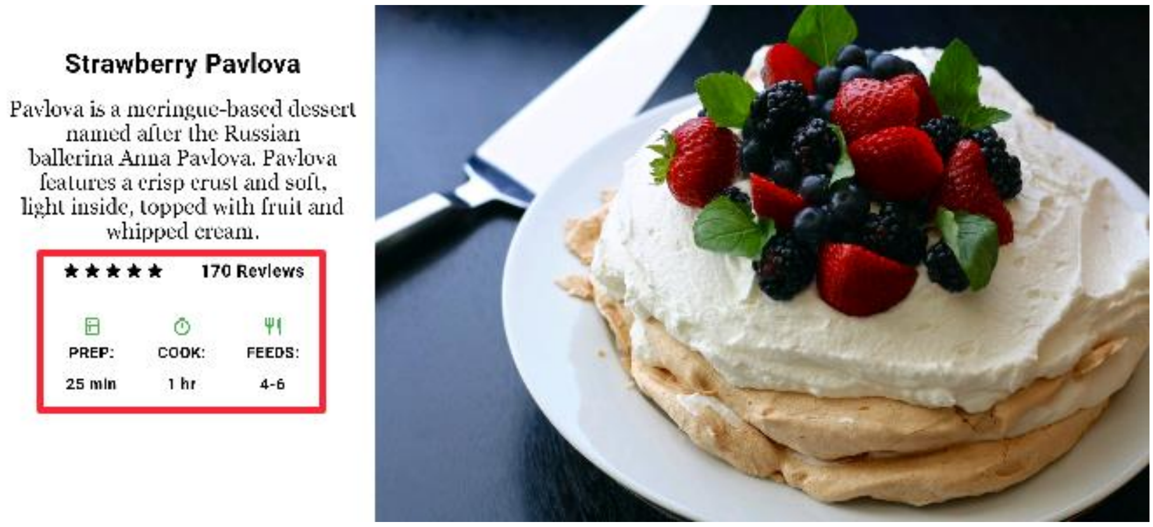
On the off chance that you need a widget to consume 2x as much space as its kin. For this, utilization the widget called Expanded widget having property called flex property, a whole number which decides the flex component for that containing widget. The default flex factor is 1. The accompanying code adjusts the flex of the center picture to two.[9]

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      flex: 2,
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

Figure 23. Widget Wrapped in Expanded with Flex: 2 to double its size[9]

2.3.4 Nesting of Rows and Columns

The format structure of flutter permits you to add as many as rows and columns within rows and columns as you are willing.



Strawberry Pavlova

Pavlova is a meringue-based dessert named after the Russian ballerina Anna Pavlova. Pavlova features a crisp crust and soft, light inside, topped with fruit and whipped cream.

★★★★★ 170 Reviews

PREP:	COOK:	FEEDS:
25 min	1 hr	4-6

Figure 24. Nesting of widgets marked with box with red outline[9]

The illustrated segment is actualized as two rows. The appraisals row contains column with stars of the rating and the quantity of surveys. The symbols row contains 3 columns of symbols and text.

The rating variable makes a row including a littler row of five Stars symbols, and bit of text.

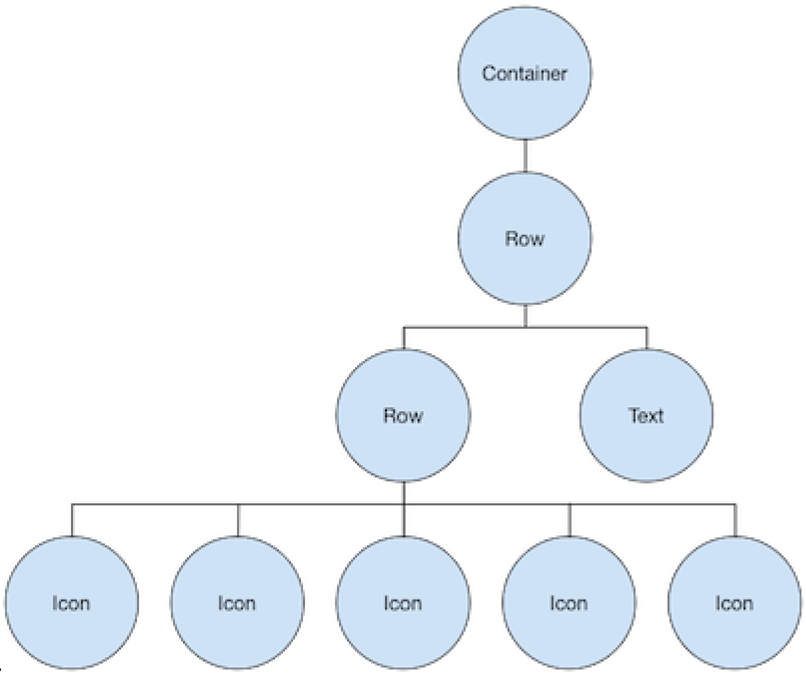


Figure 26. The widget tree for nesting of rows and columns [9]

1.3.5 Common Layout widgets

Fl Flutter has a rich library of plan widgets. Here are two or three those most normally used. The desire is to prepare you for activity as quick as could be normal considering the present situation, rather than overwhelm you with an absolute summary. For information on other open widgets, imply the Widget list, or use the Hunt encase the Programming interface reference docs. Also, the widget pages in the Programming interface docs consistently make proposition about similar widgets that may better suit your prerequisites.

The going with widgets fall into two classes: widgets from the widgets library called standard widgets, and widgets called explicit that are from the MaterialLibrary. Any application can use the widgets library anyway.15]

1.4 Packages in Flutter

Flutter emphasizes on utilizing various different packages developed by different developers working of flutter or dart applications. This permits faster assembling an application without creating everything from scratch. They are similar to libraries in other programing language.

All of the Packages are distributed from pub.dev . The Flutter home page on the given website shows all of the top and best packages that are working with Flutter , and boosts looking among every distributed package.

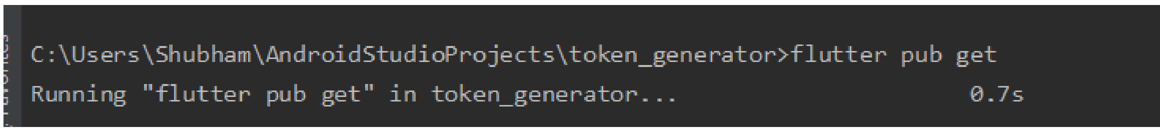
Top choices page on pub.dev records the modules and bundles that have been recognized as bundles you ought to at first consider using when forming your application. For additional information on being a Flutter Top choice, see the Flutter Top picks program.

You can likewise peruse the bundles on pub.dev by isolating on iOS modules , Android modules , web modules, or any blend thereof.[14]

Add Package to your Application:

To include the package, abc, in an application:

1. Open the pubspec.yaml document situated inside the application organizer, and include abc: under dependencies.
2. Go to terminal and run following command

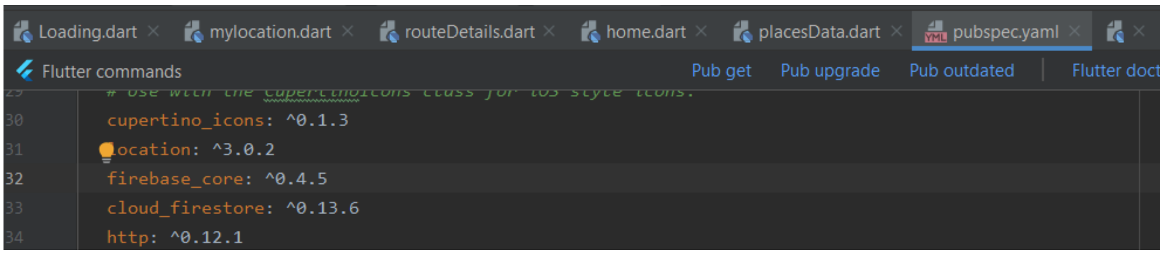


```
C:\Users\Shubham\AndroidStudioProjects\token_generator>flutter pub get
Running "flutter pub get" in token_generator...                    0.7s
```

OR

In Android Studio/IntelliJ:

Open "pubspec.yaml." Click pub get button



```
  # use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.3
  location: ^3.0.2
  firebase_core: ^0.4.5
  cloud_firestore: ^0.13.6
  http: ^0.12.1
```

OR

In VS Code:

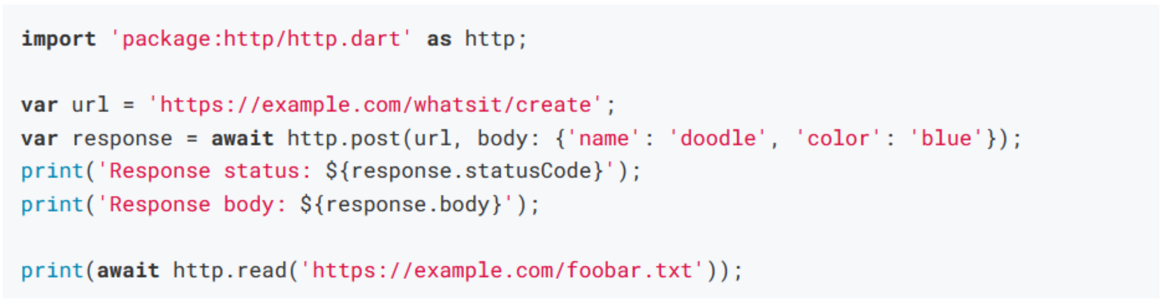
Choose get packages situated in right side of the activity strip at the highest point of "pubspec.yaml".

3. Import it in your required dart file
4. Stop and restart the application, if essential [14]

2.4.1 http Package

This package contains a lot of elevated level capacities and classes that make it simple to devour HTTP assets. It's foundation autonomous, and can be utilized on both the order line and the browser.

The least demanding approach to utilize this library is by means of the top-level capacities. They permit you to make singular HTTP demands with negligible problem:[17]



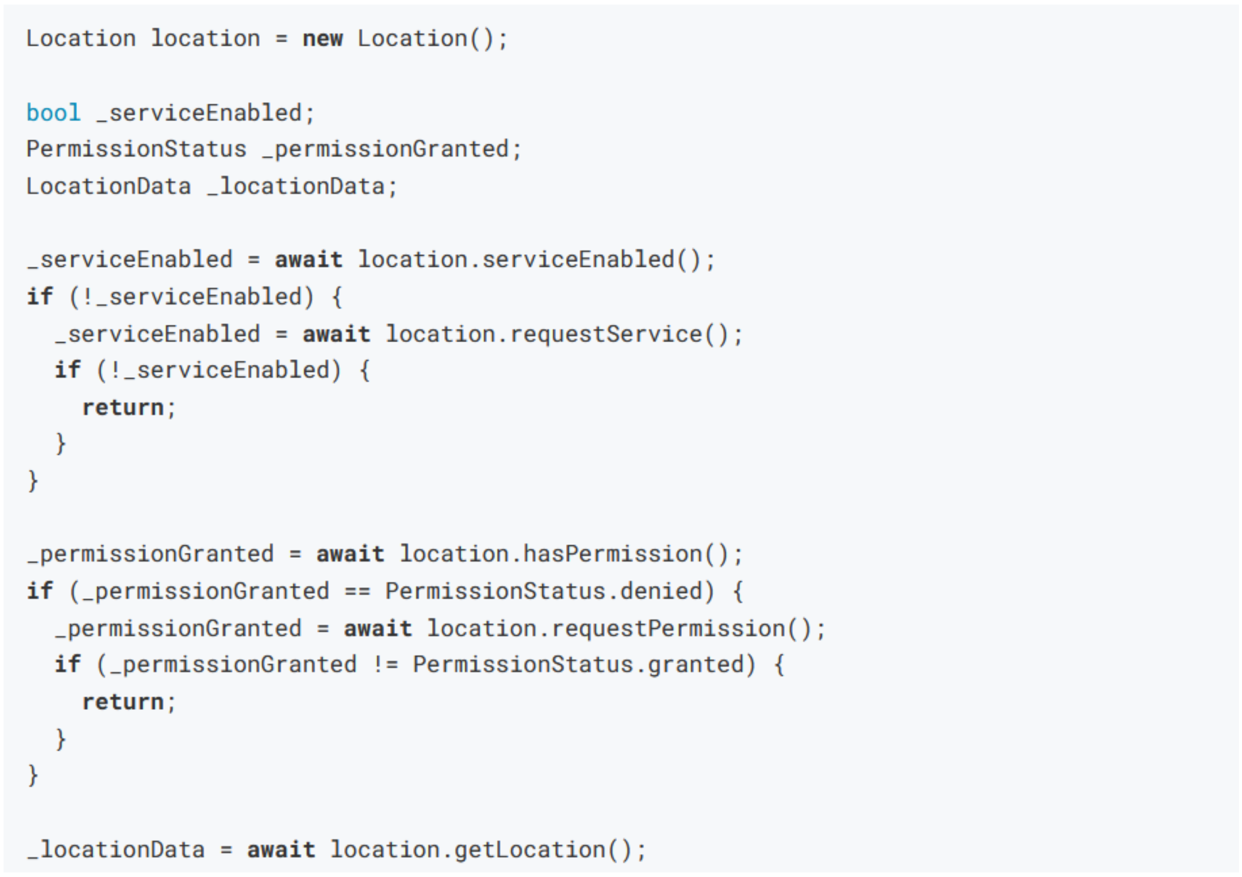
```
import 'package:http/http.dart' as http;

var url = 'https://example.com/whatsit/create';
var response = await http.post(url, body: {'name': 'doodle', 'color': 'blue'});
print('Response status: ${response.statusCode}');
print('Response body: ${response.body}');

print(await http.read('https://example.com/foobar.txt'));
```

Figure 26. Example of Usage of http Package [17]

2.4.2 Location Package

This plugin for Flutter handles getting area on Android and iOS. It additionally gives callbacks when area is changed.

In order to request location, you should all the time check manually for the Location Service status and Permission status from your android device.[16]



```
Location location = new Location();

bool _serviceEnabled;
PermissionStatus _permissionGranted;
LocationData _locationData;

_serviceEnabled = await location.serviceEnabled();
if (!_serviceEnabled) {
  _serviceEnabled = await location.requestService();
  if (!_serviceEnabled) {
    return;
  }
}

_permissionGranted = await location.hasPermission();
if (_permissionGranted == PermissionStatus.denied) {
  _permissionGranted = await location.requestPermission();
  if (_permissionGranted != PermissionStatus.granted) {
    return;
  }
}

_locationData = await location.getLocation();
```

Figure 27. Example on usage of Location package in flutter [16]

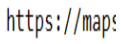
1.5 API (Application Programming Interface)

Application Programming Interface is what API is the abbreviation for, which acts like middle man between two applications and helps them to talk to one another without contacting each other and middle man keeps working of each party a secret. Each time you try to work with an app like Facebook, google, weather app or send a message, or check the weather using internet, you would be using some form of an API. [13]

Exactly when you use an application on your PDA, the application interface with the Web and sends some data to a server in far away area. The server by then recuperates that data, disentangles it, plays out the crucial exercises and sends it back to your cell phone. The application by then unravels that data and presents you with the information you required in a noticeable way. This is what a Programming interface is - the total of this occurs through Programming interface. [13]

Your phone's data is never at any point totally introduced legitimately to the server, and in like way the server is rarely totally introduced or associated with your phone. Or maybe, each talks with little packages of data, sharing only that which is significant—like mentioning takeout.[12]

Figure 28. Example of API[12]



https://maps

2.5.1 Google Maps API

Programming interface keys are required for applications and ventures that utilization the Google Maps Platform APIs and SDKs.Programming interface keys are venture driven accreditations that fill two needs:

- project Identification. - Recognize the application or the undertaking that is making a call to the API or SDK.

- project Authorization. - Check whether the calling application has been conceded access to call the API or SDK and has empowered the API or SDK in the task.

At the point when an API key is made, it is related with a task. By distinguishing the calling venture, an API key empowers utilization data to be related with that task, and guarantees calls from different undertakings are dismissed.

The Maps API lets you tweak maps with your own substance and symbolism for show on web pages and internet devices.[12]

2.5.2 Places API

The Places API is an assistance that gives data about places utilizing HTTP requests. Places are characterized inside this API as establishment, geographic areas, or interest points.

Places API has following place requests:

Place Details - returns increasingly nitty gritty data about a particular place, including client surveys.

Place Autocomplete - naturally fills in the name or potentially address of a place as user type.

Place Search - restores a rundown of places dependent on a client's area or search string.

Place Photos – allows you access to the a many place related photos that are included in database made by google.

Every one of the services is gotten to as a http requests returns either a JSON file or XML file as a reaction.

The Places API utilizes a place ID to interestingly distinguish a place.[10]

2.5.3 Directions API

The Directions API is an assistance that ascertains directions among two or more different places. You can look for directions for a few methods of transportation, including travel, driving, strolling, or cycling.

You get to the Directions API through a HTTP interface, with demands built as a URL string, utilizing text strings or scope/longitude directions to recognize the areas, alongside your API key.[11]

2.6 Firebase

Firebase is a versatile and web application development platform created by Firebase, Inc. in 2011. Later in 2014 it was bought by Google Inc.

A Firebase venture is the top-level substance for Firebase. In your undertaking, you can include Firebase applications that can be iOS, Android, or Web applications. After you arrange your applications to utilize Firebase, you would then be able to include the Firebase SDKs for any number of Firebase items, as Analytics, Cloud Firestore, Performance Monitoring, or Remote Config.[18]

2.6.1 Cloud Firestore:

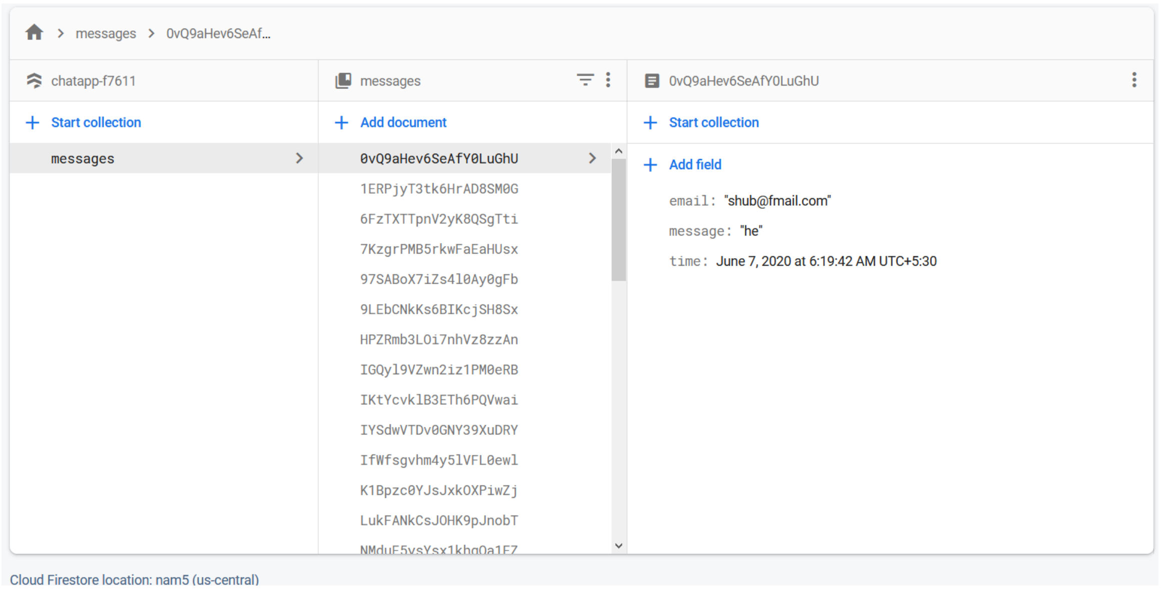
It store and sync data by using versatile, adaptable NoSQL cloud database for server side unexpected turn of events.

It is a versatile, adaptable database for invaluable, web and webserver progress from Google Cloud Stage and firebase. Like firebase Realtime Database, it in like way keeps your information in a condition of sync across different various applications through persistent customers/clients and offers speedier and less mind boggling help for being developed so you can without an entirely striking stretch make responsive-applications that work distributing basically no remaining to filter through inaction or your structure. Cloud Firestore other than offers steady perception with different Firebase things and other GCP things.

Firebase offers 2 cloud based, client accessible database plans that help to get predictable data and work on that:

Cloud Firestore: It is firebase most current database for versatile application/web improvement. It develops the achievements of the past Reatime Database with another, constantly instinctual data model. Cloud Firestore in like manner unites significantly increasingly snappier, liberal requests and scales farther than the Realtime Database. [20]

Realtime Database: It is one of Firebase excellent database. It\'s a profitable, less-inaction answer for versatile applications that require balanced states around clients in authentic time.[21]



```
messages

0vQ9aHev6SeAfY0LuGhU
1ERPjyT3tk6HrAD8SM0G
6FzTXTTpnV2yK8QSgTti
7KzgrPMB5rkwFaEaHUsx
97SABoX7iZs410Ay0gFb
9LEbCNkKs6BIKcjSH8Sx
HPZRmb3LOi7nhVz8zzAn
IGQyl9VZwn2iz1PM0eRB
IKtYcvklB3ETh6PQVwai
IYSdwVTDv0GNY39XuDRY
IfWfsgvhm4y5lVFL0ewl
K1Bpzc0YJsJxkOXPiwZj
LukFANkCsJOHK9pJnobT
NMduF5vsYsx1khgQa1FZ

email: "shub@fmail.com"
message: "he"
time: June 7, 2020 at 6:19:42 AM UTC+5:30
```

Cloud Firestore location: nam5 (us-central)

Figure 29. Example of Cloud Firestore [19]

Chapter 3 SYSTEM DEVELOPMENT

3.1 Silvers

A sliver is a part of a scrollable area. You can utilize slivers to accomplish custom scrolling effects. For data on actualizing slivers in Flutter, including SliverList, SliverGrid, and SliverAppBar. You can sort of consider Slivers a lower-level interface, giving better grained control on actualizing scrollable area. Since slivers can sluggishly construct every thing similarly as it scrolls into see, slivers are especially valuable for productively scrolling through huge quantities of children.

These sliver Components go inside a CustomScrollView. You can reinvent a ListView by putting a SliverList inside a CustomScrollView.

SliverAppBar:

It is a app bar with material design that comes under CustomScrollView.This aapbar consists of toolbar and many other widgets like tabBar and a FlexibleSpaceBar.appBar is then followed by popup memubutoon for less common operations.

Sliver application bars are commonly utilized as the primary child of a CustomScrollView, which lets the application bar coordinate with the scroll see so it can change in stature as per the scroll balance or buoy over the other substance in the scroll see. For a fixed-tallness application bar at the highest point of the screen see AppBar, which is utilized in the Scaffold.appBar space.

The AppBar shows the toolbar widgets, driving, title, and activities, over the base (assuming any). On the off chance that a flexibleSpace widget is indicated, at that point it is stacked behind the toolbar and the base widget.

SliverList:

A sliver that places various box children in a straight cluster along the primary hub.

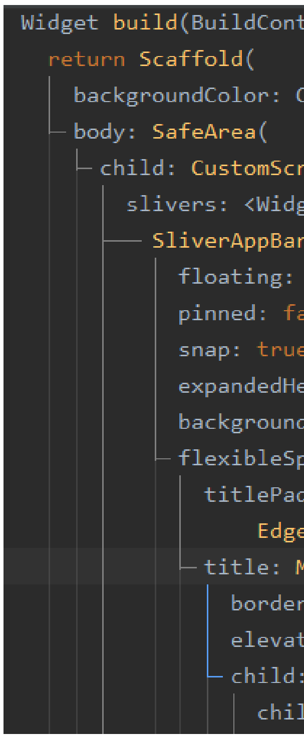
Every child is compelled to have the SliverConstraints.crossAxisExtent in the cross hub yet decides its own primary hub degree. SliverList decides its scroll counterbalance by "dead reckoning" since children outside the noticeable piece of the sliver are not materialized, which implies SliverList can't become familiar with their fundamental hub degree. Rather, recently materialized children are set nearby existing children.[22]

3.2 OnPressed

On the off chance that onPressed is set, at that point this callback will be considered when the client taps on the name or symbol parts of the chip. On the off chance that onPressed is invalid, at that point the chip will be impaired.

Following is the home screen of the application:[22]

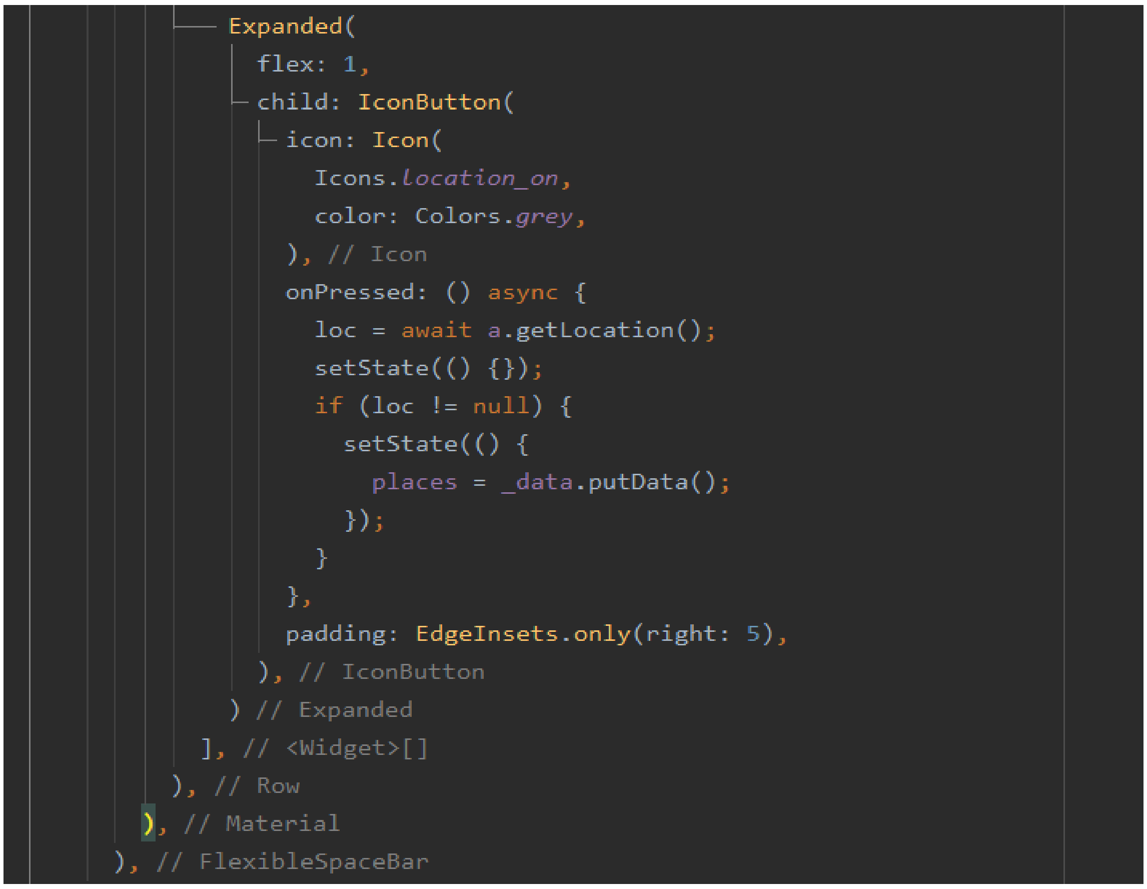
.



```
Widget build(BuildCont
  return Scaffold(
    backgroundColor: C
    body: SafeArea(
      child: CustomScr
        slivers: <Widg
          SliverAppBar
            floating:
            pinned: fa
            snap: true
            expandedHe
            backgrounc
            flexibleSp
              titlePac
                  Edge
              title: M
                border
                elevat
                child:
                  chil
```

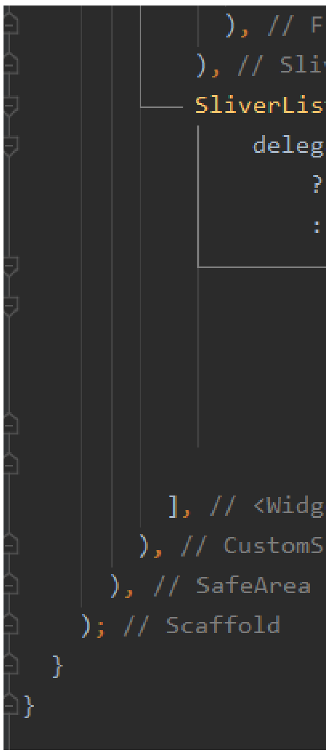
Figure 30. Home Page of the application (home.dart {1})

.



```
        Expanded(
          flex: 1,
          child: IconButton(
            icon: Icon(
              Icons.location_on,
              color: Colors.grey,
            ), // Icon
            onPressed: () async {
              loc = await a.getLocation();
              setState(() {});
              if (loc != null) {
                setState(() {
                  places = _data.putData();
                });
              }
            },
            padding: EdgeInsets.only(right: 5),
          ), // IconButton
        ) // Expanded
      ], // <Widget>[]
    ), // Row
  ), // Material
), // FlexibleSpaceBar
```

Figure 31. Home Page of the application (home.dart {2})



```
              ), // F
            ), // Sli
            SliverLis
              deleg
                  ?
                  :



          ], // <Widg
        ), // CustomS
      ), // SafeArea
    ); // Scaffold
  }
}
```

Figure 32. Home Page of the application (home.dart {3})

3.3 Location

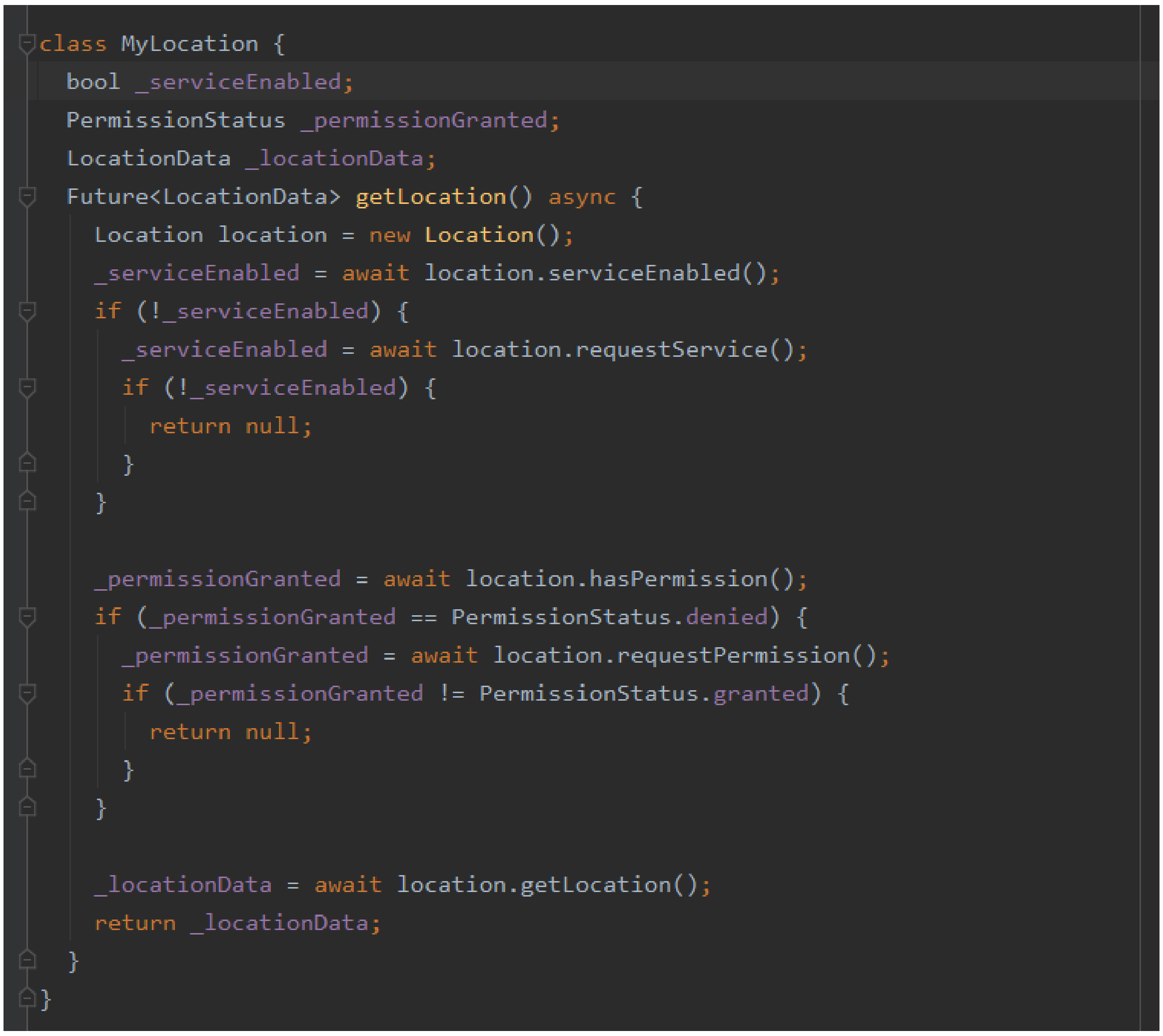
To retrieve location of the device add the following package to your pubspec.yaml file:

Like below:

```
dependencies:
  location: ^3.0.0
```

To retrieve location in your screen import the package using following line

import 'package:location/location.dart';



```dart
class MyLocation {
  bool _serviceEnabled;
  PermissionStatus _permissionGranted;
  LocationData _locationData;
  Future<LocationData> getLocation() async {
    Location location = new Location();
    _serviceEnabled = await location.serviceEnabled();
    if (!_serviceEnabled) {
      _serviceEnabled = await location.requestService();
      if (!_serviceEnabled) {
        return null;
      }
    }

    _permissionGranted = await location.hasPermission();
    if (_permissionGranted == PermissionStatus.denied) {
      _permissionGranted = await location.requestPermission();
      if (_permissionGranted != PermissionStatus.granted) {
        return null;
      }
    }

    _locationData = await location.getLocation();
    return _locationData;
  }
}
```

Figure 33. Code to retrieve location from device (location.dart)

3.4 Navigate to next screen

Most applications contain a few screens for showing various kinds of data. For instance, an application may have a screen that shows items. At the point when the client taps the picture of an item, another screen shows insights regarding the item.

In Flutter, screens and pages are called routes. The rest of this formula alludes to routes.

In Android, a route is comparable to an Activity. In iOS, a route is comparable to a ViewController. In Flutter, a route is only a widget.

This formula utilizes the Navigator to explore to another route.

The following not many areas tell the best way to explore between two routes, utilizing these means:

-Make two routes(Screens).

-Explore to the subsequent route utilizing Navigator.push().

-Come back to the principal route utilizing Navigator.pop().

Navigator.push():

To change to another route, utilize the Navigator.push() technique. The push() technique adds a Route to the stack of routes oversaw by the Navigator. You can make your own, or utilize a MaterialPageRoute, which is helpful on the grounds that it advances to the new route utilizing a platform-explicit animation. [23]

For Example:

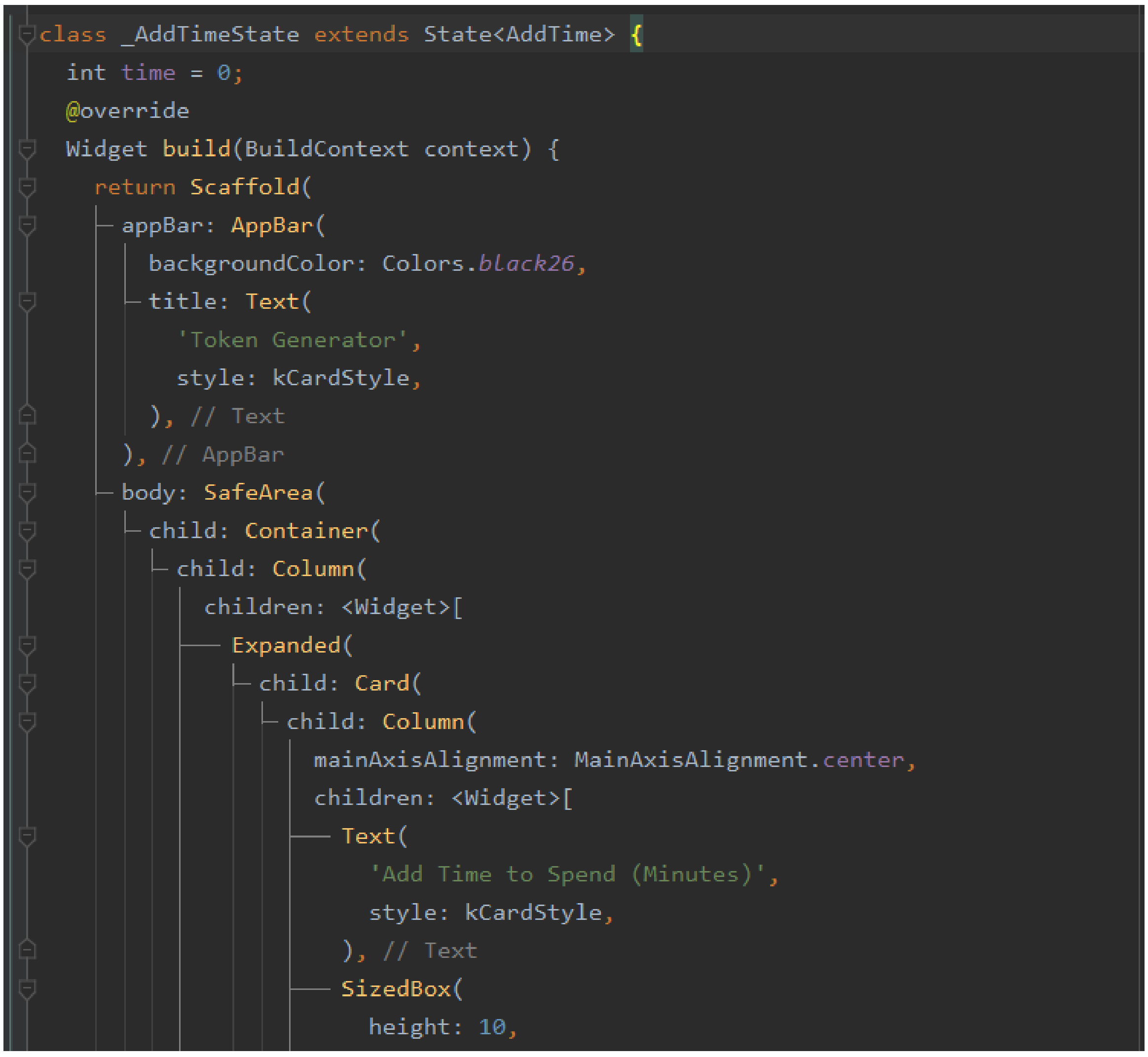
```
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondRoute()),
  );
}
```

Navigator.pop():

To go back to previous route use Navigator.pop() method, the pop method removes the current route from the given stack of routes managed by the navigator.

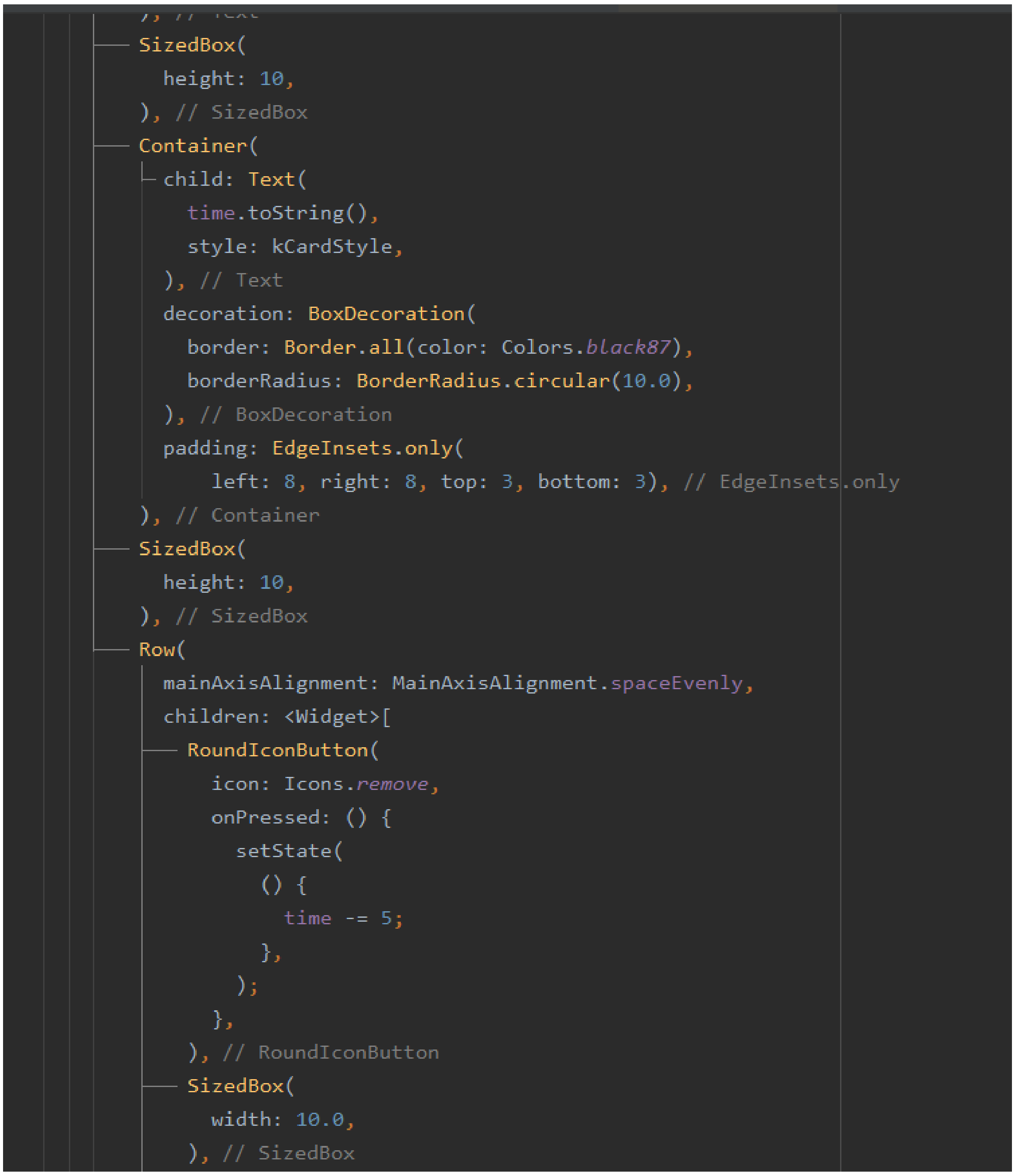
For Example:

```
onPressed: () {
  Navigator.pop(context);
}
```



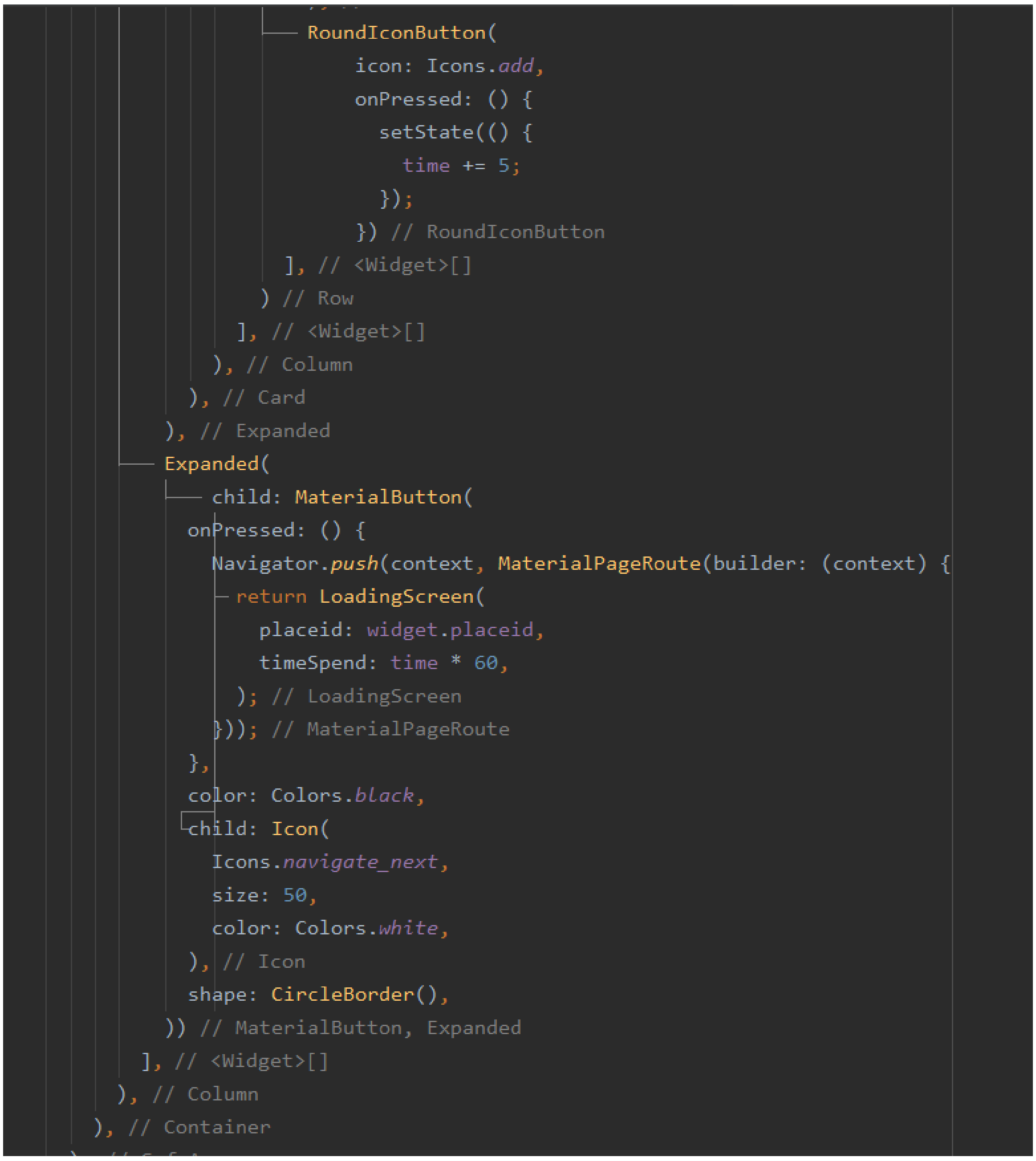
```dart
class _AddTimeState extends State<AddTime> {
  int time = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black26,
        title: Text(
          'Token Generator',
          style: kCardStyle,
        ), // Text
      ), // AppBar
      body: SafeArea(
        child: Container(
          child: Column(
            children: <Widget>[
              Expanded(
                child: Card(
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                      Text(
                        'Add Time to Spend (Minutes)',
                        style: kCardStyle,
                      ), // Text
                      SizedBox(
                        height: 10,
```

Figure 34. code to ask user to add time he will spend addTime.dart{1}



```
      SizedBox(
        height: 10,
      ), // SizedBox
      Container(
        child: Text(
          time.toString(),
          style: kCardStyle,
        ), // Text
        decoration: BoxDecoration(
          border: Border.all(color: Colors.black87),
          borderRadius: BorderRadius.circular(10.0),
        ), // BoxDecoration
        padding: EdgeInsets.only(
            left: 8, right: 8, top: 3, bottom: 3), // EdgeInsets.only
      ), // Container
      SizedBox(
        height: 10,
      ), // SizedBox
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          RoundIconButton(
            icon: Icons.remove,
            onPressed: () {
              setState(
                () {
                  time -= 5;
                },
              );
            },
          ), // RoundIconButton
          SizedBox(
            width: 10.0,
          ), // SizedBox
```

Figure 35. code to ask user to add time he will spend addTime.dart{2}



```
         RoundIconButton(
           icon: Icons.add,
           onPressed: () {
            setState(() {
             time += 5;
            });
           }) // RoundIconButton
        ], // <Widget>[]
       ) // Row
      ], // <Widget>[]
     ), // Column
    ), // Card
   ), // Expanded
   Expanded(
     child: MaterialButton(
    onPressed: () {
     Navigator.push(context, MaterialPageRoute(builder: (context) {
      return LoadingScreen(
       placeid: widget.placeid,
       timeSpend: time * 60,
      ); // LoadingScreen
     })); // MaterialPageRoute
    },
    color: Colors.black,
    child: Icon(
     Icons.navigate_next,
     size: 50,
     color: Colors.white,
    ), // Icon
    shape: CircleBorder(),
   )) // MaterialButton, Expanded
  ], // <Widget>[]
 ), // Column
), // Container
```

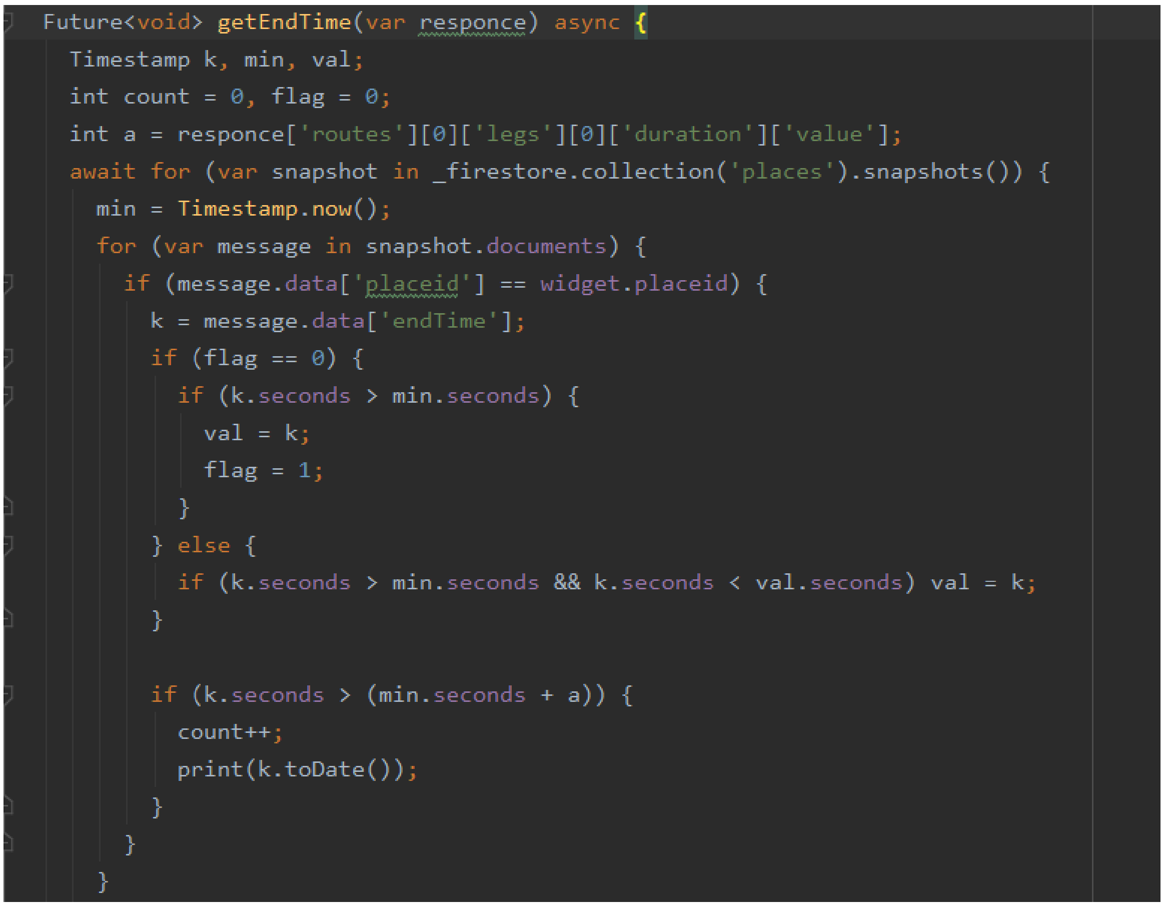
Figure 36. code to ask user to add time he will spend addTime.dart{3}

3.4 Algorithm

Algorithm to find the time user reach the store and user will leave store and add that to the collective database.

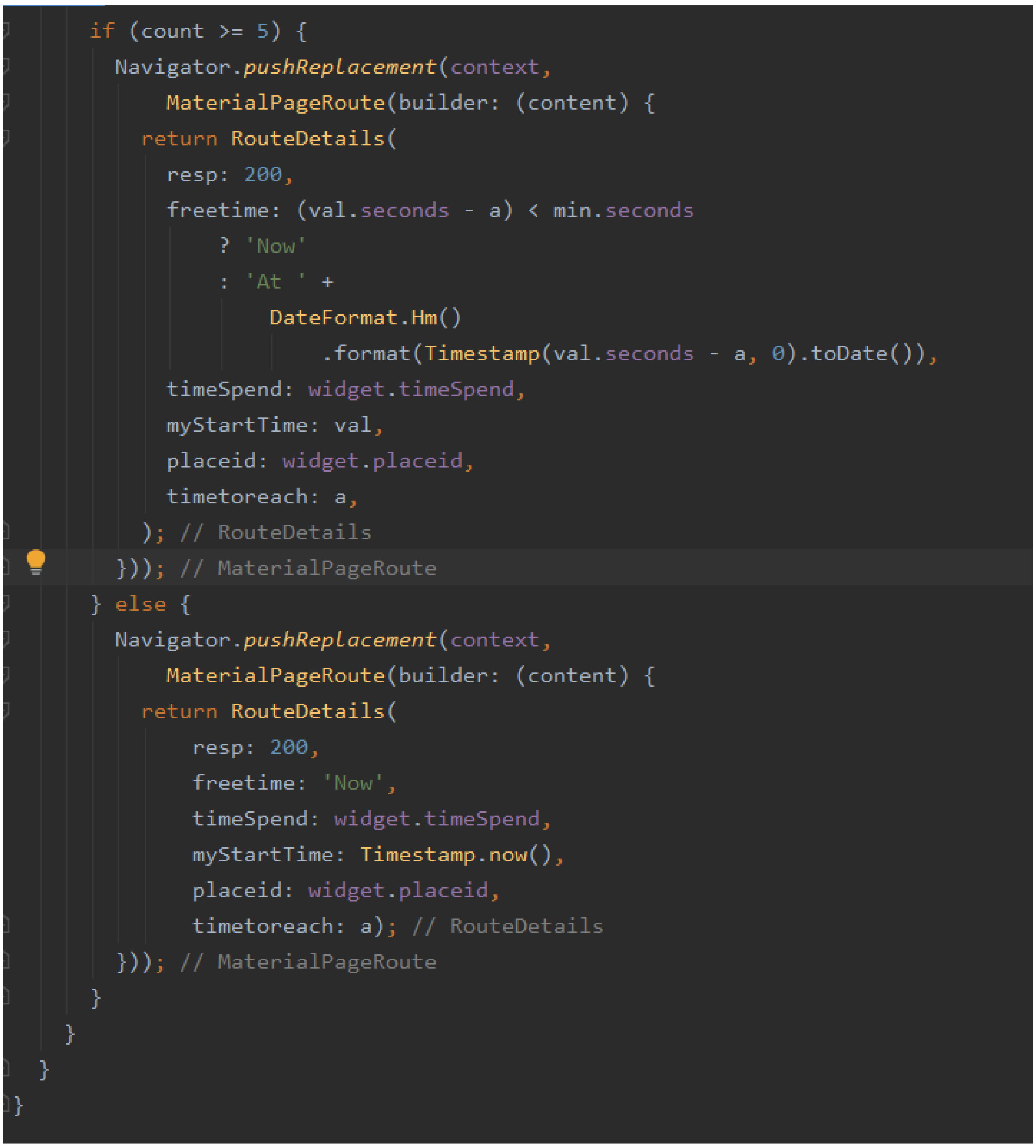
We use get requests to get the required data from google maps API using a custom API Key.

We firebase plugin and cloud firestore plugin to store the required data in firebase no-sql database on cloud.



```
Future<void> getEndTime(var responce) async {
  Timestamp k, min, val;
  int count = 0, flag = 0;
  int a = responce['routes'][0]['legs'][0]['duration']['value'];
  await for (var snapshot in _firestore.collection('places').snapshots()) {
    min = Timestamp.now();
    for (var message in snapshot.documents) {
      if (message.data['placeid'] == widget.placeid) {
        k = message.data['endTime'];
        if (flag == 0) {
          if (k.seconds > min.seconds) {
            val = k;
            flag = 1;
          }
        } else {
          if (k.seconds > min.seconds && k.seconds < val.seconds) val = k;
        }

        if (k.seconds > (min.seconds + a)) {
          count++;
          print(k.toDate());
        }
      }
    }
```

Figure 37: Algorithm to find Time user reaches and leaves the store{1}



```
    if (count >= 5) {
      Navigator.pushReplacement(context,
          MaterialPageRoute(builder: (content) {
        return RouteDetails(
          resp: 200,
          freetime: (val.seconds - a) < min.seconds
              ? 'Now'
              : 'At ' +
                  DateFormat.Hm()
                      .format(Timestamp(val.seconds - a, 0).toDate()),
          timeSpend: widget.timeSpend,
          myStartTime: val,
          placeid: widget.placeid,
          timetoreach: a,
        ); // RouteDetails
      })); // MaterialPageRoute
    } else {
      Navigator.pushReplacement(context,
          MaterialPageRoute(builder: (content) {
        return RouteDetails(
            resp: 200,
            freetime: 'Now',
            timeSpend: widget.timeSpend,
            myStartTime: Timestamp.now(),
            placeid: widget.placeid,
            timetoreach: a); // RouteDetails
      })); // MaterialPageRoute
    }
  }
 }
}
```

Figure 38: Algorithm to find Time user reaches and leaves the store{2}

3.8 Hardware Requirements

1. OS- Windows/Ubuntu/Mac
2. RAM- 6 GB or higher
3. GPU- Intel integrated Gpu or higher
4. Processor- Quad-Core i3 or higher
5. 8 GB Disk Space or higher

3.9 Software Requirements

1. Android Studio or any other IDE
2. Dart Plugin
3. Flutter Plugin
4. Flutter SDK
5. Android SDK
6. Web Browser
7. Android Virtual Device

Chapter 4 PERFORMANCE ANALYSIS

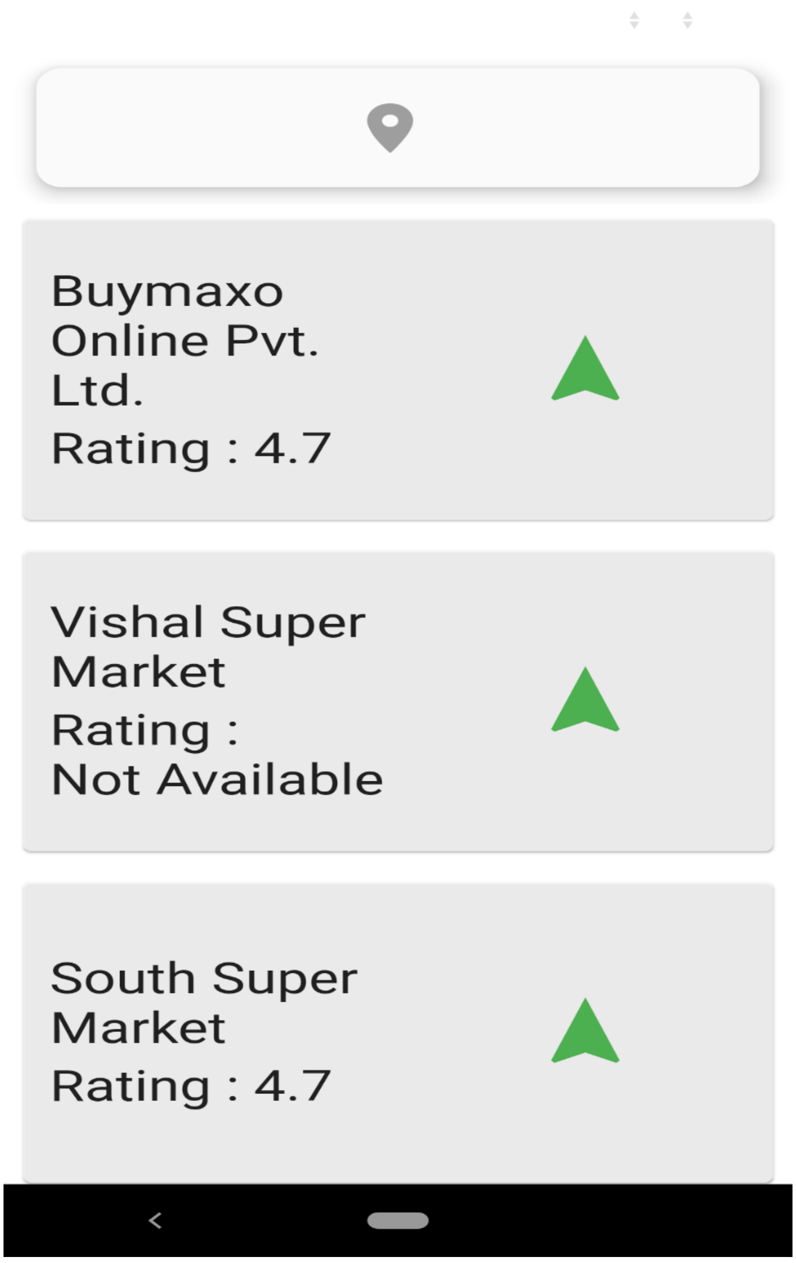
4.1 Home Page of the Application

Home of the Application displays nothing till location of the device is received by my application. It only displays a button on top to manually try to get location of device if user denies to give permission first time.

Nothing to Display

Figure 39. Snapshots of application with home page(when location is not available)

Once Location of device is received it displays the nearby stores at your location within radius of 1 KM. You can select which location you want o visit next, from this screen of the application



Buymaxo Online Pvt. Ltd.
Rating : 4.7

Vishal Super Market
Rating : Not Available

South Super Market
Rating : 4.7

Figure 40 Snapshots of application with home page(when location is available)

4.2 Add Time

Once you have selected the location you want to visit by tapping on the location icon. You will be sent to next screen where you can add the maximum time you will spend in the given selected store once You have selected the time by pressing the plus and minus icon present on the screen. You can Click next to go on next screen.

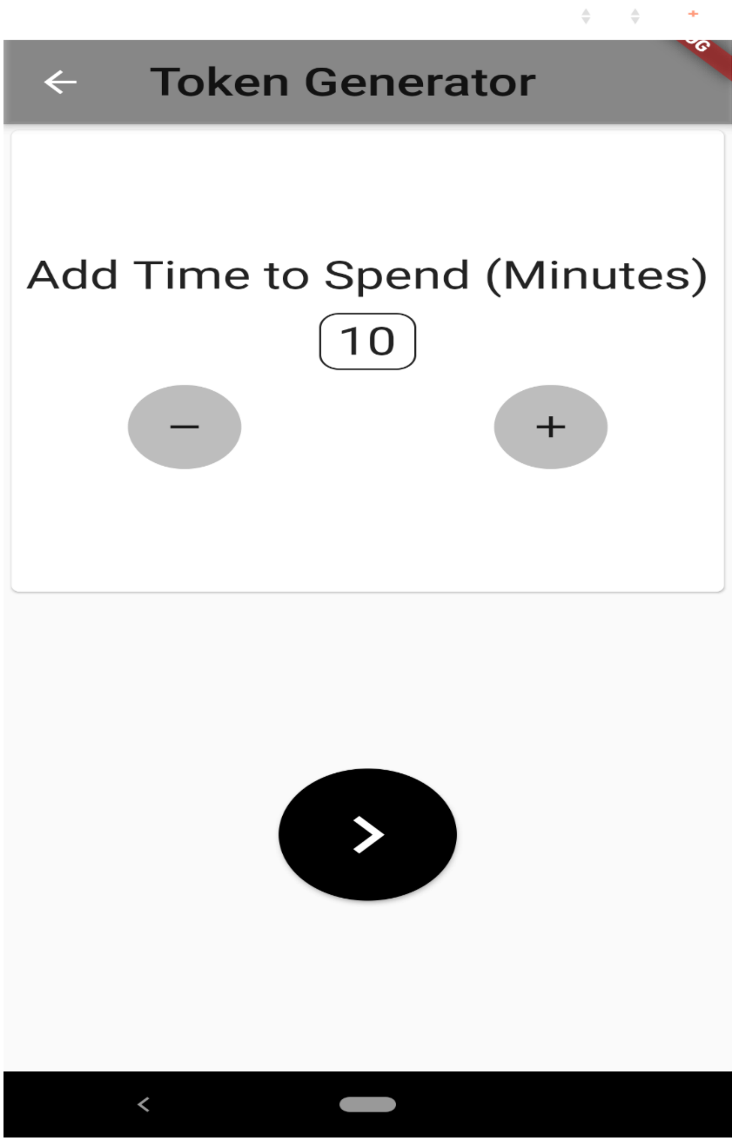
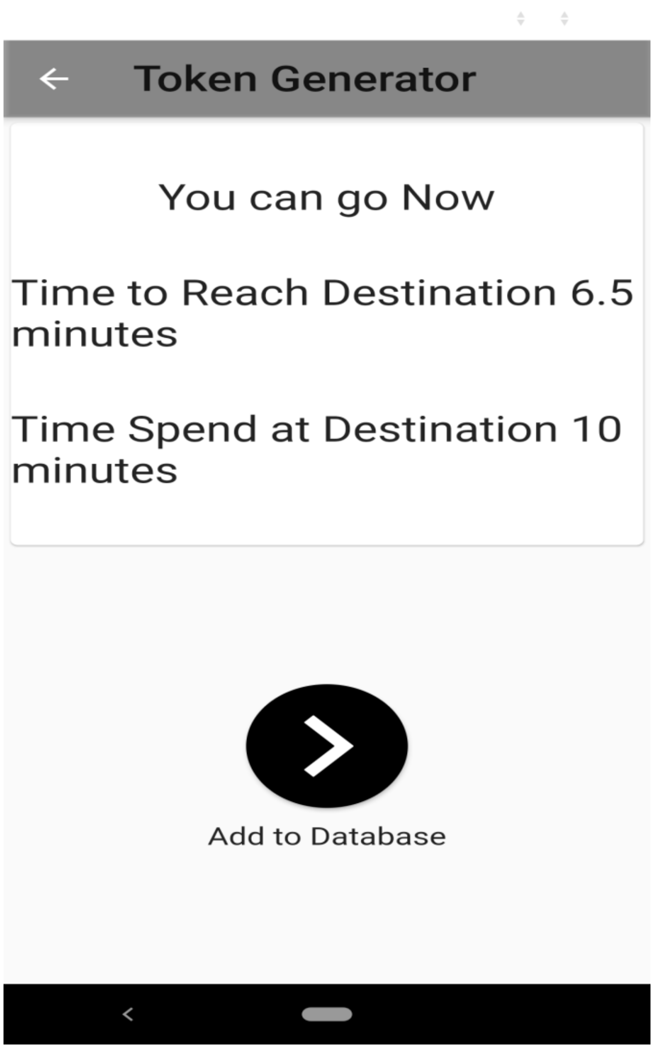


Figure 41. Snapshots of application with page to add max time user will spend at the store

4.3 Result Page

This pagedisplays the time you can leave your house to get to your destination on time.

Time to reach the store you have selected.


Token Generator

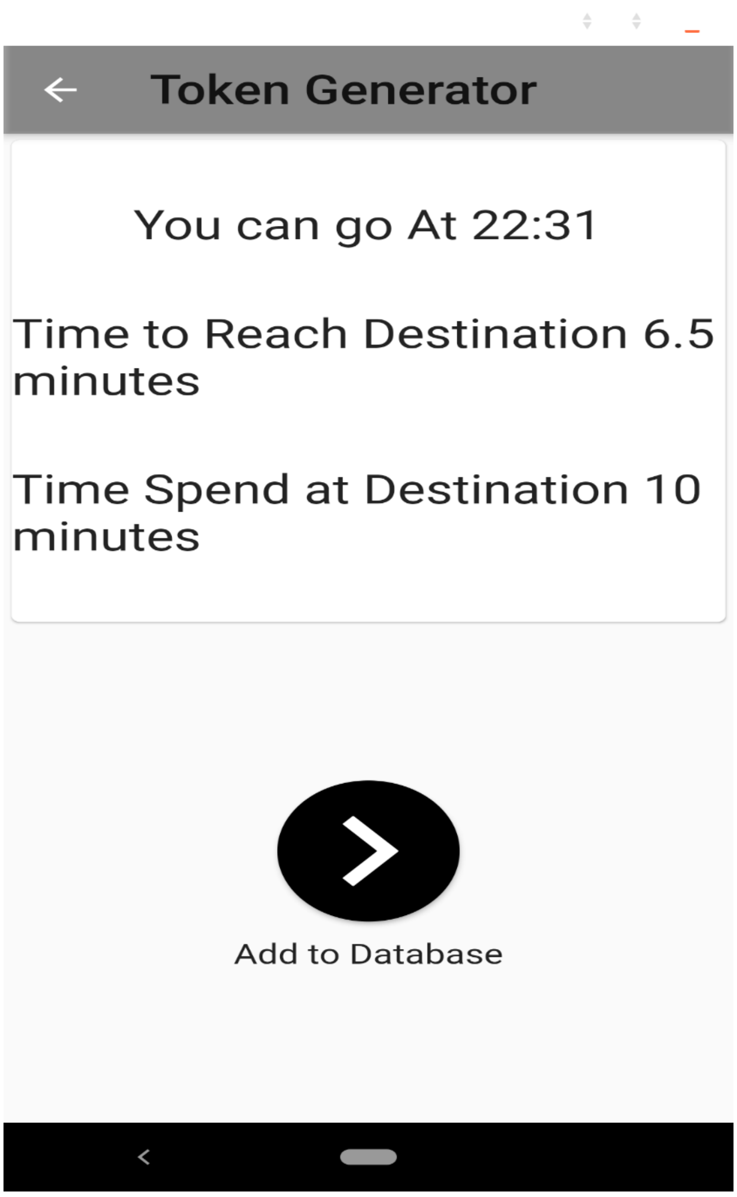
You can go Now

Time to Reach Destination 6.5 minutes

Time Spend at Destination 10 minutes

Figure 42. Snapshots of application with result page giving details about your journey

Once More than 5 People will be present in store it will give you the next time you need to leave your house to get to store in time.


Token Generator

You can go At 22:31

Time to Reach Destination 6.5 minutes

Time Spend at Destination 10 minutes

Figure 43. Snapshots of application with result page giving details about your journey(With Time)

Chapter 5 CONCLUSIONS

5.1 Conclusions

We propose a working application that can get you to the store and tell you the approximate time to reach the store and prevent the local stores from being overcrowded. This application will allow at max 5 people at one single store and person who will come after 5th will be given time the earliest person will leave and So on on for the next users who click on the same store.

5.2 Future Work

I will add a cleaner USER INTERFACE, increase to total performance of the application.

More features could be added to application like integration of google maps within the application.

Add notification options, making it visually more beautiful and with a tutorial for the usage of the application.

implementing search option within the app and making a cancel button if user is unwilling to the store.

5.2.1 Application

It will reduce the number of people in the store at a time of epidemic, and making social distancing possible in small stores.

References

1. "A tour of the Dart language," *Dart*, 2020. [Online]. Available: https://dart.dev/guides/language/language-tour.
2. "Dart programming language," *Dart*, 2020. [Online]. Available: https://dart.dev/.
3. "Commonly used packages," *Dart*, 2020. [Online]. Available: https://dart.dev/guides/libraries/useful-libraries
4. "Dart API docs," *Dart*, 2020. [Online]. Available: https://api.dart.dev/stable/2.8.4/index.html.
5. "core library," *dart*, 2020. [Online]. Available: https://api.dart.dev/stable/2.8.4/dart-core/dart-core-library.html
6. "Technical overview," *Flutter*, 2020. [Online]. Available: https://flutter.dev/docs/resources/technical-overview
7. "Layouts in Flutter," *Flutter,* 2020. [Online]. Available: https://flutter.dev/docs/development/ui/layout.
8. "Beautiful native apps in record time," *Flutter*, 2020. [Online]. Available: https://flutter.dev/.
9. "Building layouts," Flutter, 2020. [Online]. Available: https://flutter.dev/docs/development/ui/layout/tutorial.
10. "Overview | Places API | Google Developers," Google, 2020. [Online]. Available: https://developers.google.com/places/web-service/intro.
11. Google, 2020. [Online]. Available: https://developers.google.com/maps/documentation/directions/start.
12. Google, 2020. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial.
13. "What is an API? (Application Programming Interface)," MuleSoft, 2020. [Online]. Available: https://www.mulesoft.com/resources/api/what-is-an-api.

14. “Using packages,” Flutter, 2020. [Online]. Available: https://flutter.dev/docs/development/packages-and-plugins/using-packages.

15. “Introduction to widgets,” *Flutter*, 2020. [Online]. Available: https://flutter.dev/docs/development/ui/widgets-intro.

16. “location: Flutter Package,” Dart packages, 03-Apr-2020. [Online]. Available: https://pub.dev/packages/location.

17. “http: Dart Package,” Dart packages, 27-Apr-2020. [Online]. Available: https://pub.dev/packages/http.

18. Google, 2020. [Online]. Available: https://firebase.google.com

19. Google Accounts, 2020. [Online]. Available: https://console.firebase.google.com/

20. “Cloud Firestore | Firebase,” Google, 2020. [Online]. Available: https://firebase.google.com/docs/firestore.

21. “Firebase Realtime Database,” Google, 2020. [Online]. Available: https://firebase.google.com/docs/database.

22. “Slivers,” Flutter, 2020. [Online]. Available: https://flutter.dev/docs/development/ui/advanced/slivers.

23. “Navigate to a new screen and back,” Flutter, 2020. [Online]. Available: https://flutter.dev/docs/cookbook/navigation/navigation-basics.