Nama : Syava Aprilia Puspitasari

Kelas : SIB 3E

NIM : 2241760129

No.Abs : 24

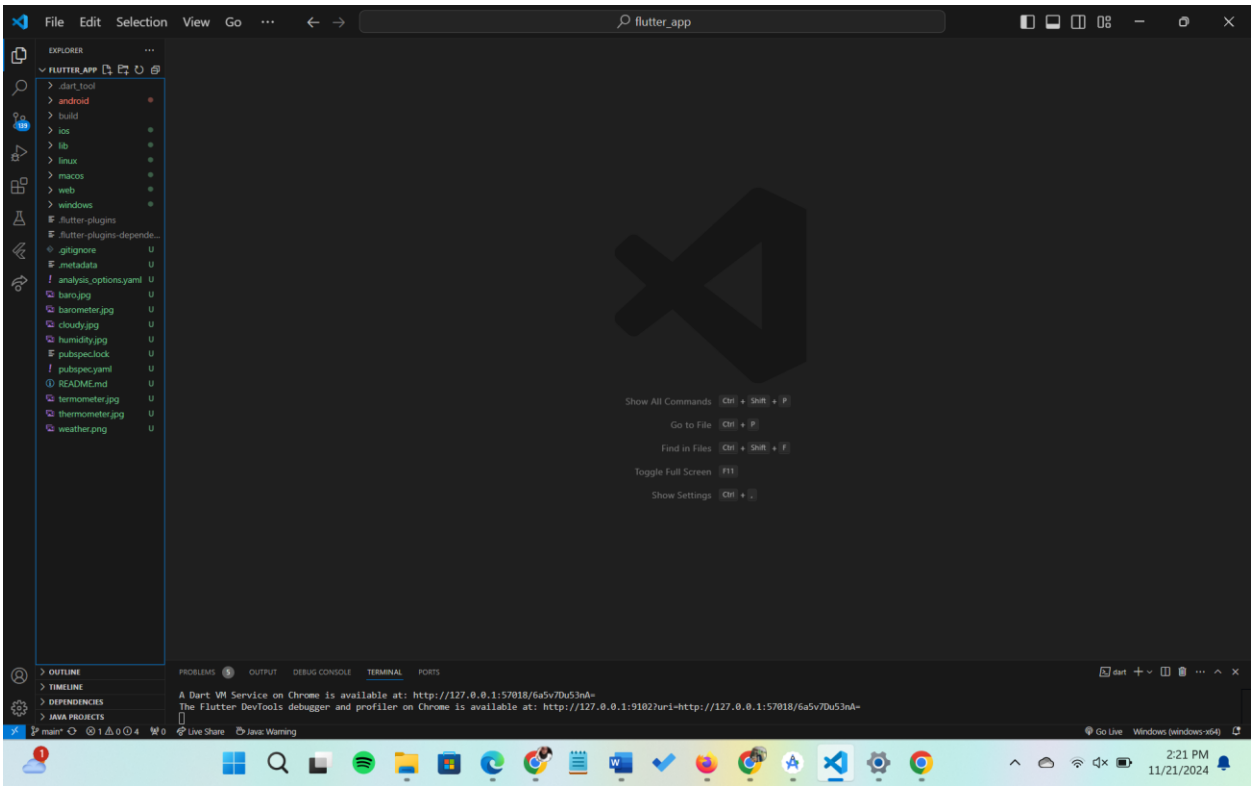
Mata Kuliah : Pemrograman Mobile

Pertemuan 12

Weather app using API integration in Flutter

Creating a New Flutter Project

Open up android studio (or visual studio) to create a new flutter project(If you don't have flutter installed then follow steps in this video and proceed). Choose Start a new flutter project options



Fetching the packages

Flutter provides various packages for simplifying the app building process. Here, we make use of a few packages like http and geolocator. The http package aids in establishing a connection

between our Flutter app with the internet and the geocoding package aids in obtaining the location data. We can incorporate the packages into our app by providing the package name with version in the pubspec.yaml in our flutter project, that is the configuration file. dependencies:

geolocator:

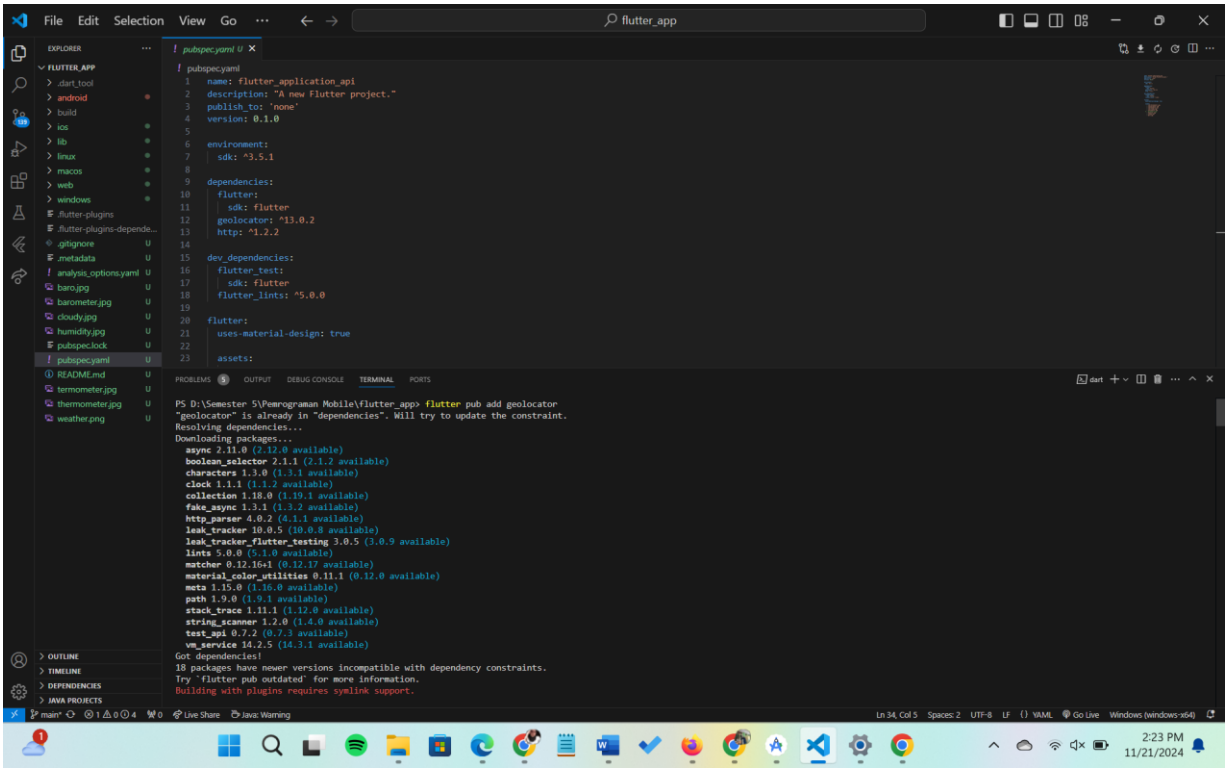
http:

Here I have not specified the version number of the packages. This enables to obtain the latest compatible versions of the dependencies on running pub get command. In order to see the exact version obtained you can refer the pubspec.lock file. For this project I have used version 8.0.5 of geolocator package and version 0.13.3 of http package.

We will be implementing the code as functions. So our aim is to display the current location weather conditions on opening the app and then to get the current weather conditions of any other locations or city.

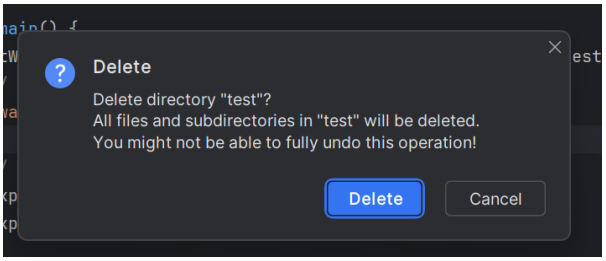
In order to achieve this initially we will have to get the current location's latitude and longitude. We can achieve this with the help of geolocator package. However in order to use the package there are certain configurations for android as well as iOS platforms which can be referred in the readme section of geolocator package.

Once the configurations are done we can proceed to get the current location latitude and longitude.



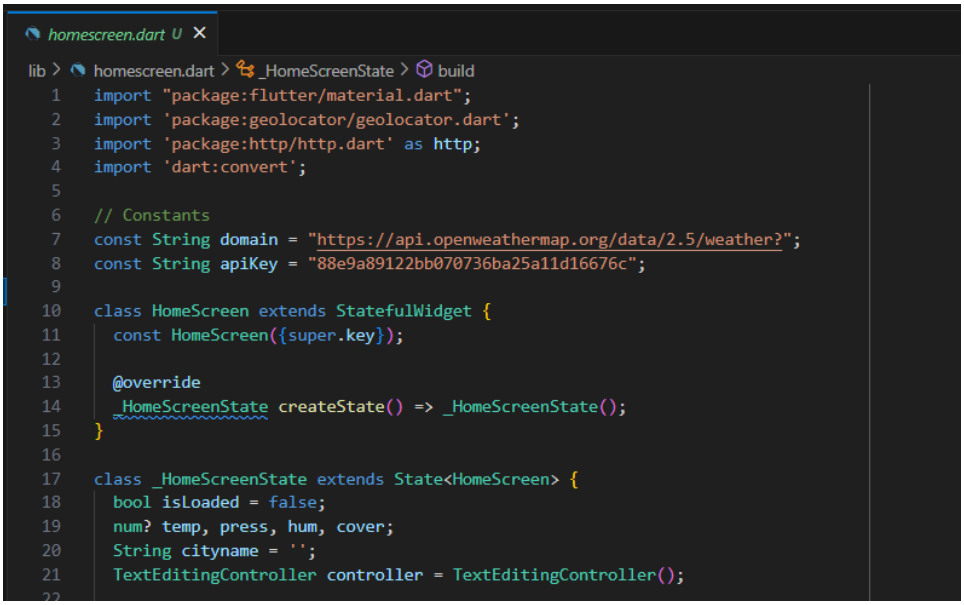
Building the App

In the default main.dart file we will be removing the unnecessary code and keep the necessary code only. Here we will be working with MaterialApp instead of MyApp. Therefore we delete the MyApp references in main.dart and delete the test file under projects section.



Now create a new dart file by right clicking on lib >New>Dart File

Inside this dart file import the material.dart package. We will now create a Stateful widget to build our app. The shortcut to create one is to simply type stful and we will get a skeletal code of the Stateful widget. Replace the YourWidgetName with your own custom names.



Now once that is done, import this dart file in main.dart and set the home property of MaterialApp() to the name given for the Stateful widget created. The home property is used to display the starting screen in an app when the app involves only a single screen. Here I have given the name as HomeScreen() and the dart file name as homescreen.dart.

```dart
import 'package:flutter/material.dart';
import 'homescreen.dart';

void main() {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const HomeScreen(),
      theme: ThemeData(
        primaryColor: Colors.white, colorScheme: ColorScheme.fromSwatch().copyWith(secondary: Colors.white),
      ), // ThemeData
    ), // MaterialApp
  );
}
```

Apart from the home property of MaterialApp we will also be specifying the debugShowCheckedModeBanner property to false so the debug banner won't be visible and also setting the primary and accent theme colors to white using the theme property.

Step 1: Getting Current location co-ordinates

It is inside the homescreen.dart we will be obtaining the current location latitude and longitude. The latitude and longitude is not displayed in the final version of the app so we will just try to print the values in terminal.

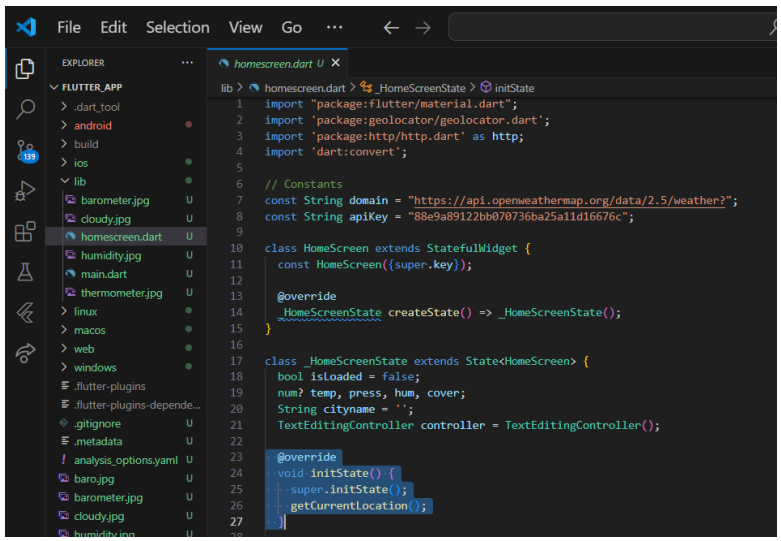
For this first we have to import the geolocator package in homescreen.dart

```dart
import "package:flutter/material.dart";
import 'package:geolocator/geolocator.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

// Constants
const String domain = "https://api.openweathermap.org/data/2.5/weather?";
const String apiKey = "88e9a89122bb070736ba25a11d16676c";

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

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

class _HomeScreenState extends State<HomeScreen> {
  bool isloaded = false;
  num? temp, press, hum, cover;
  String cityname = '';
  TextEditingController controller = TextEditingController();

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

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

  void getCurrentLocation() async {
    var position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.low,
    );
    getCurrentCityWeather(position);
    }
```

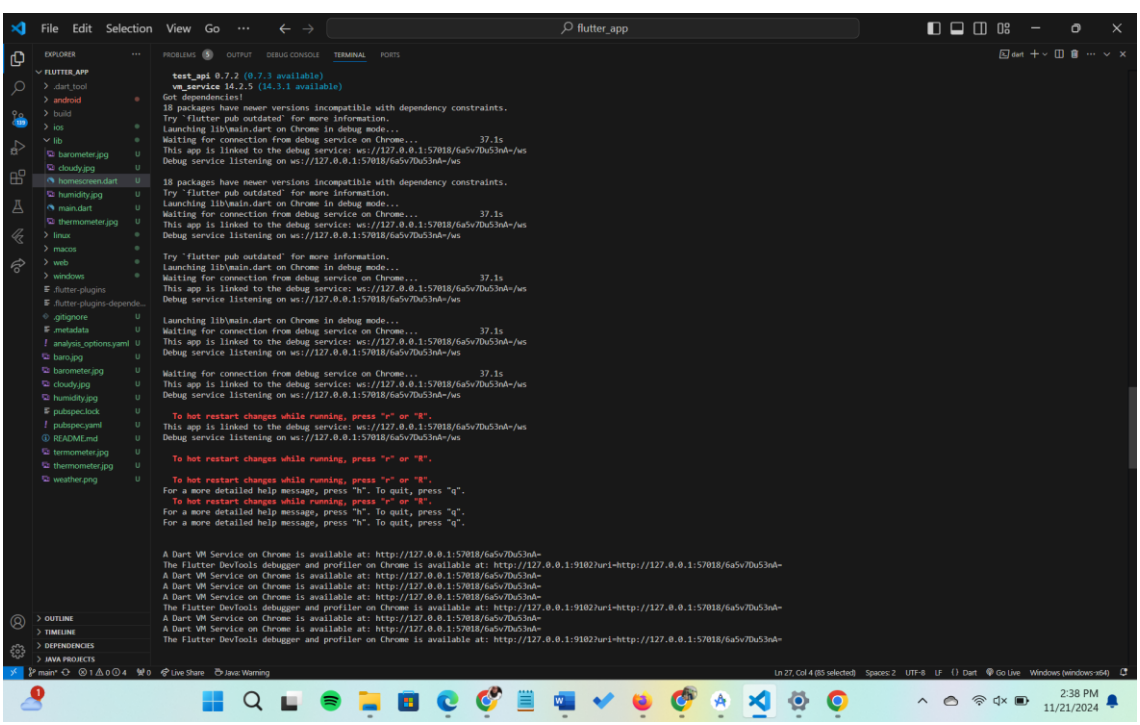
```
PS D:\Semester 5\Pemrograman Mobile\flutter_app> flutter pub add geolocator
"geolocator" is already in "dependencies". Will try to update the constraint.
Resolving dependencies...
Downloading packages...
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.18.0 (1.19.1 available)
```

The accuracy of the position can set by the desiredAccuracy property and the location manager of android can be set by the forceAndroidLocationManager property. The returned Position value consists of the details of the current location details of the device. To obtain the latitude and longitude, the corresponding parameters of the Position value is called and printed accordingly.

This function is then called in the initState() method of the Stateful() widget as shown and The build method is returned with a simple Scaffold() wrapped in SafeArea() as shown.

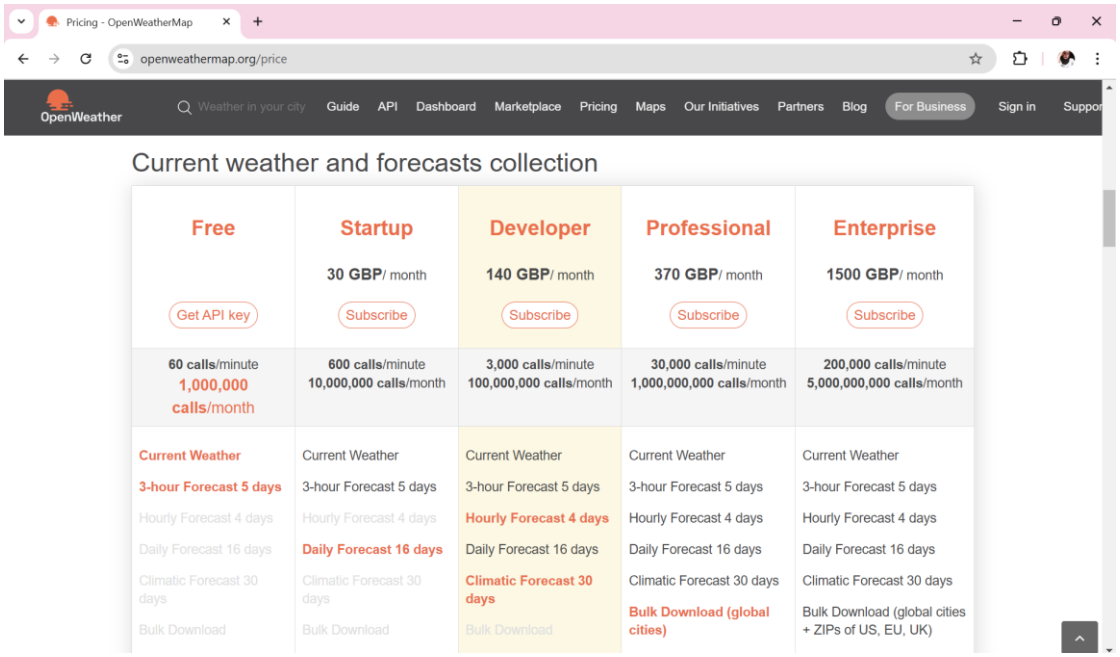


With this run the application and we will obtain the latitude and longitude of the current location.

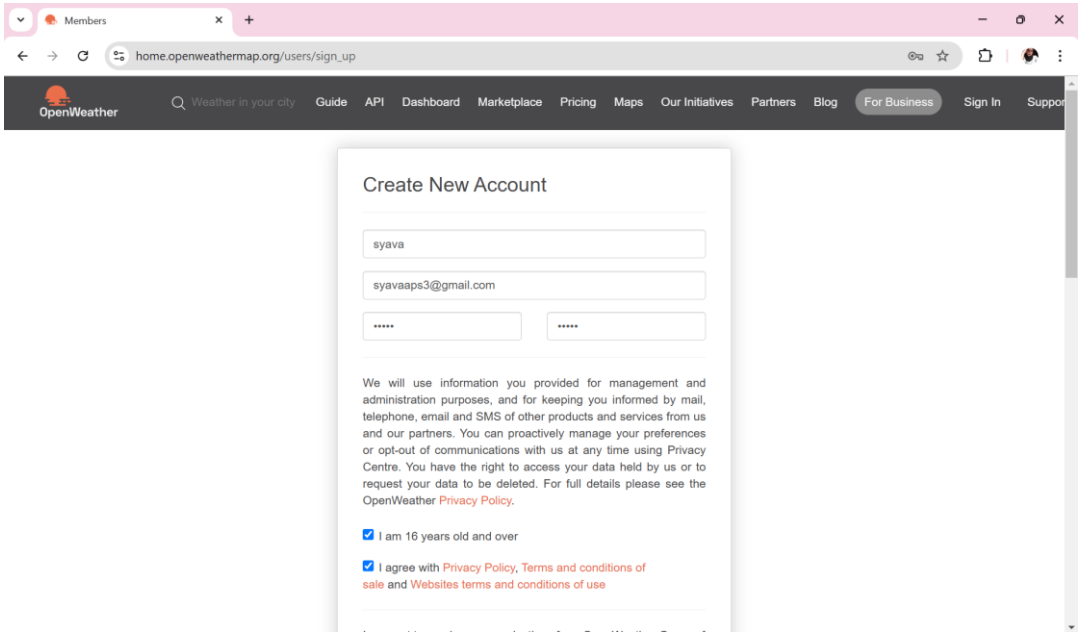


STEP 2: Getting API key from OpenWeatherMap

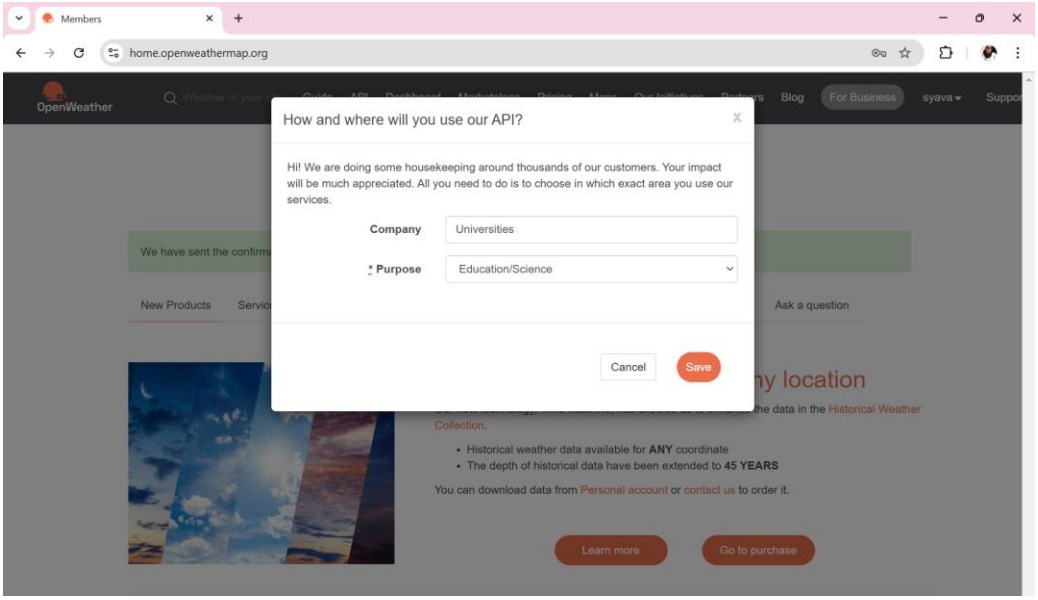
For getting the real-time weather data we will be making use of OpenWeatherMap API. To access the API data we require an API key. In order to do that, choose the Pricing section from the top menu.



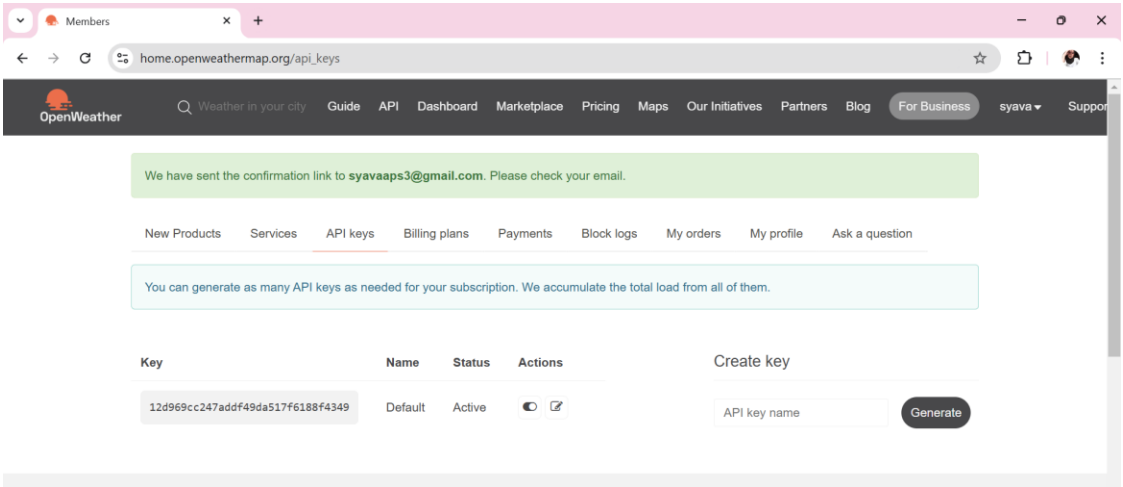
Once that is done, a new window comes up where you have to create an account. Provide the necessary information and create your account.



Once the account is created, a pop-up will come up asking about the purpose for which we intend to use our API key



Here specifying the company is optional while specifying the purpose is mandatory. If you don't know which purpose to choose you can go with Education/Science. Then click save. Now in your dashboard go to the API keys section. Here you will find all your API keys under the Key section. Note that the API key here is unique to you and you shouldn't share it with anyone else



Once you get your API key you can proceed to call the API to get the weather data. There are various formats of calling the API and how the response will be, all of which can be referred here. For our purpose we will be calling the API via the latitude longitude format and the city name format as given below API call by latitude longitude format https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}API call by city name format

https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}

As both API calls have same API key and domain, we will store those values in a new dart file named constants under lib folder. For creating the file refer the Building App section above. In the created constants.dart file add the following lines of code.

```
const String domain = "https://api.openweathermap.org/data/2.5/weather?";
const String apiKey = "12d969cc247addf49da517f6188f4349";
```

Now we will try obtaining the weather data which will be a JSON response.

STEP 3: Getting weather data of Current location

After successfully completing the above steps, we will try obtaining the weather data. For this inside the homescreen.dart we will write a function to first connect our app to the internet and then obtain weather data. This is where we will be implementing http package. So first we import the package as http so it becomes easier to access the different fields in the package

```
import "package:flutter/material.dart";
import 'package:geolocator/geolocator.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
```

```
18 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
PS D:\Semester 5\Pemrograman Mobile\flutter_app> flutter pub add http
"http" is already in "dependencies". Will try to update the constraint.
Resolving dependencies...
Downloading packages...
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.18.0 (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
  http_parser 4.0.2 (4.1.1 available)
  leak_tracker 10.0.5 (10.0.8 available)
  leak_tracker_flutter_testing 3.0.5 (3.0.9 available)
  lints 5.0.0 (5.1.0 available)
  matcher 0.12.16+1 (0.12.17 available)
  material_color_utilities 0.11.1 (0.12.0 available)
  meta 1.15.0 (1.16.0 available)
  path 1.9.0 (1.9.1 available)
  stack_trace 1.11.1 (1.12.0 available)
  string_scanner 1.2.0 (1.4.0 available)
  test_api 0.7.2 (0.7.3 available)
  vm_service 14.2.5 (14.3.1 available)
Got dependencies!
18 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
```

Similarly we also import the constants.dart where we copied the API key and domain link. Along with it as we are dealing with getting a single JSON response for each call, we use the convert library of dart for decoding the JSON response we obtain.

API call by latitude longitude format

https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}

Next we will write the function to connect our app to internet. This function will also be asynchronous as it is returning a future. Here we first create a Client object so we don't have to open and close ports every time we call the get method. Then we provide our URI (A URI is a character sequence that helps identify a logical or physical resource connected to the internet) address which is the API call format providing the necessary fields.

```dart
const String apiKey = "88e9a89122bb070736ba25a11d16676c";

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

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

class _HomeScreenState extends State<HomeScreen> {
  bool isLoaded = false;
  num? temp, press, hum, cover;
  String cityname = '';
  TextEditingController controller = TextEditingController();

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

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

  void getCurrentLocation() async {
    var position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.low,
    );
    getCurrentCityWeather(position);
    }

  Future<void> getCurrentCityWeather(Position position) async {
    var uri =
        '${domain}lat=${position.latitude}&lon=${position.longitude}&appid=$apiKey';
    var url = Uri.parse(uri);
    var response = await http.get(url);
    if (response.statusCode == 200) {
      var data = json.decode(response.body);
      updateUI(data);
      setState(() {
        isLoaded = true;
      });
    } else {
      print(response.statusCode);
    }
  }
```

STEP 4: Getting weather data of various cities

Similar to STEP 3 we will implement getting the weather data of a particular city based on the city name. Here the only difference is instead of providing latitude and longitude in the

API call we provide the city name.

```dart
class _HomeScreenState extends State<HomeScreen> {

  Future<void> getCurrentCityWeather(Position position) async {
    var uri =
        '${domain}lat=${position.latitude}&lon=${position.longitude}&appid=$apiKey';
    var url = Uri.parse(uri);
    var response = await http.get(url);
    if (response.statusCode == 200) {
      var data = json.decode(response.body);
      updateUI(data);
      setState(() {
        isLoaded = true;
      });
    } else {
      print(response.statusCode);
    }
  }

  Future<void> getCityWeather(String cityName) async {
    var uri = '${domain}q=$cityName&appid=$apiKey';
    var url = Uri.parse(uri);
    var response = await http.get(url);
    if (response.statusCode == 200) {
      var data = json.decode(response.body);
      updateUI(data);
      setState(() {
        isLoaded = true;
      });
    } else {
      print(response.statusCode);
    }
  }
```

he code for obtaining complete background gradient is as shown below. Here we will set the resizeToAvoidBottomInset property of Scaffold() widget to false so that resizing of the widgets when the keyboard pops up is avoided

```
lib > homescreen.dart > _HomeScreenState
17  class _HomeScreenState extends State<HomeScreen> {
91    @override
92    Widget build(BuildContext context) {
93      return SafeArea(
94        child: Scaffold(
95          resizeToAvoidBottomInset: false,
96          body: Container(
97            width: double.infinity,
98            height: double.infinity,
99            padding: const EdgeInsets.all(20),
100           decoration: const BoxDecoration(
101             gradient: LinearGradient(
102               colors: [
103                 Color(0xfffA8BFF),
104                 Color(0xff2BD2FF),
105                 Color(0xff2BFF88),
106               ],
107               begin: Alignment.bottomLeft,
108               end: Alignment.topRight,
109             ), // LinearGradient
110           ), // BoxDecoration
```

Then we will provide a few variables to store values of temperature, pressure, humidity, cloud cover, city name and the state of the data that is whether the data is fetched and ready to be used by the app. These variables are declared inside the Stateful widget before the initState method

```
lib > homescreen.dart > _HomeScreenState
17  class _HomeScreenState extends State<HomeScreen> {
58    Future<void> getCityWeather(String cityName) async {
59      var url = '${domain}q=$cityName&appid=$apiKey';
60      var uri = Uri.parse(uri);
61      var response = await http.get(uri);
62      if (response.statusCode == 200) {
63        var data = json.decode(response.body);
64        updateUI(data);
65        setState(() {
66          isLoaded = true;
67        });
68      } else {
69        print(response.statusCode);
70      }
71    }
72
73    void updateUI(var decodedData) {
74      setState(() {
75        if (decodedData == null) {
76          temp = 0;
77          press = 0;
78          hum = 0;
79          cover = 0;
80          cityname = 'Not available';
81        } else {
82          temp = decodedData['main']['temp'] - 273;
83          press = decodedData['main']['pressure'];
84          hum = decodedData['main']['humidity'];
85          cover = decodedData['clouds']['all'];
86          cityname = decodedData['name'];
87        }
88      });
89    }
90
91    @override
92    Widget build(BuildContext context) {
93      return SafeArea(
94        child: Scaffold(
95          resizeToAvoidBottomInset: false,
96          body: Container(
97            width: double.infinity,
98            height: double.infinity,
99            padding: const EdgeInsets.all(20),
100           decoration: const BoxDecoration(
101             gradient: LinearGradient(
102               colors: [
103                 Color(0xfffA8BFF),
104                 Color(0xff2BD2FF),
105                 Color(0xff2BFF88),
106               ],
```

Next we will add a Visibility widget as the child of the Container widget. So only when there is data will the weather data is displayed else it shows a loading indicator. For this the value of isLoaded is dynamically changed in the functions using setState method. The code is as given below.

```
lib > homescreen.dart > _HomeScreenState > updateUI
17  class _HomeScreenState extends State<HomeScreen> {
91    @override
92    Widget build(BuildContext context) {
93      return SafeArea(
94        child: Scaffold(
95          resizeToAvoidBottomInset: false,
96          body: Container(
97            width: double.infinity,
98            height: double.infinity,
99            padding: const EdgeInsets.all(20),
100           decoration: const BoxDecoration(
101             gradient: LinearGradient(
102               colors: [
103                 Color(0xffFA8BFF),
104                 Color(0xff2BD2FF),
105                 Color(0xff2BFF88),
106               ],
107               begin: Alignment.bottomLeft,
108               end: Alignment.topRight,
109             ), // LinearGradient
110           ), // BoxDecoration
111           child: Visibility(
112             visible: isLoaded,
113             replacement: const Center(child: CircularProgressIndicator()),
114             child: Column(
115               children: [
116                 // Search Input
117                 Container(
118                   width: MediaQuery.of(context).size.width * 0.85,
119                   height: MediaQuery.of(context).size.height * 0.09,
120                   padding: const EdgeInsets.symmetric(horizontal: 10),
121                   decoration: BoxDecoration(
122                     color: Colors.black.withOpacity(0.3),
123                     borderRadius: BorderRadius.circular(20),
124                   ), // BoxDecoration
125                   child: Center(
126                     child: TextFormField(
127                       onFieldSubmitted: (String s) {
128                         setState(() {
129                           cityname = s;
130                           getCityWeather(s);
131                           isLoaded = false;
132                           controller.clear();
133                         });
134                       },
135                       controller: controller,
136                       cursorColor: Colors.white,
137                       style: const TextStyle(
138                         fontSize: 20,
139                         fontWeight: FontWeight.w600,
```

Now to store the weather data into the variables we will provide another function to update the variables. This function will have the decoded JSON data as the parameter and based on the value of it, the parameter values are set.

```
homescreen.dart U
lib > homescreen.dart > _HomeScreenState > updateUI
17  class _HomeScreenState extends State<HomeScreen> {
58    Future<void> getCityWeather(String cityName) async {
64        updateUI(data);
65        setState(() {
66          isLoaded = true;
67        });
68      } else {
69        print(response.statusCode);
70      }
71    }
72
73    void updateUI(var decodedData) {
74      setState(() {
75        if (decodedData == null) {
76          temp = 0;
77          press = 0;
78          hum = 0;
79          cover = 0;
80          cityname = 'Not available';
81        } else {
82          temp = decodedData['main']['temp'] - 273;
83          press = decodedData['main']['pressure'];
84          hum = decodedData['main']['humidity'];
85          cover = decodedData['clouds']['all'];
86          cityname = decodedData['name'];
87        }
88      });
89    }
```

This function will be called in both the API calling functions if the status code of received data is 200

```
homescreen.dart U ×
lib > homescreen.dart > _HomeScreenState > getCurrentCityWeather
17  class _HomeScreenState extends State<HomeScreen> {
41
42    Future<void> getCurrentCityWeather(Position position) async {
43      var uri =
44          '${domain}lat=${position.latitude}&lon=${position.longitude}&appid=$apiKey';
45      var url = Uri.parse(uri);
46      var response = await http.get(url);
47      if (response.statusCode == 200) {
48        var data = json.decode(response.body);
49        updateUI(data);
50        setState(() {
51          isLoaded = true;
52        });
53      } else {
54        print(response.statusCode);
55      }
56    }
57
58    Future<void> getCityWeather(String cityName) async {
59      var uri = '${domain}q=$cityName&appid=$apiKey';
60      var url = Uri.parse(uri);
61      var response = await http.get(url);
62      if (response.statusCode == 200) {
63        var data = json.decode(response.body);
64        updateUI(data);
65        setState(() {
66          isLoaded = true;
67        });
68      } else {
69        print(response.statusCode);
70      }
71    }
```

Inside the column of Visibilty widget we will add a TextFormField as the first child. For the controller of this TextFormField a controller is also provided, which is declared along with the variables.

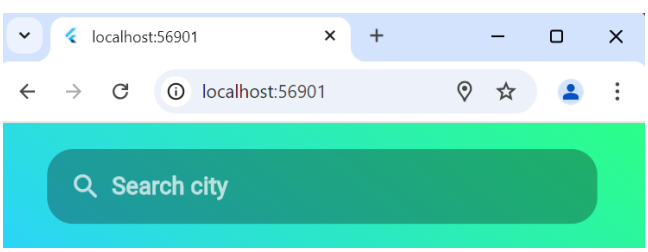
TextEditingController controller = TextEditingController();

The code of the TextFormField is as given below. Note that we have set the value of isLoaded to false once the city name is entered. This provides a loading indicator to the user while fetching the data, so the user will know there is some process going on.

```
homescreen.dart U ×
lib > homescreen.dart > _HomeScreenState
1   import "package:flutter/material.dart";
2   import 'package:geolocator/geolocator.dart';
3   import 'package:http/http.dart' as http;
4   import 'dart:convert';
5
6   // Constants
7   const String domain = "https://api.openweathermap.org/data/2.5/weather?";
8   const String apiKey = "88e9a89122bb070736ba25a11d16676c";
9
10  class HomeScreen extends StatefulWidget {
11    const HomeScreen({super.key});
12
13    @override
14    _HomeScreenState createState() => _HomeScreenState();
15  }
16
17  class _HomeScreenState extends State<HomeScreen> {
18    bool isLoaded = false;
19    num? temp, press, hum, cover;
20    String cityname = '';
21    TextEditingController controller = TextEditingController();
22
23    @override
24    void initState() {
25      super.initState();
26      getCurrentLocation();
27    }
```

```dart
class _HomeScreenState extends State<HomeScreen> {
  Widget build(BuildContext context) {
                // Search Input
                Container(
                  width: MediaQuery.of(context).size.width * 0.85,
                  height: MediaQuery.of(context).size.height * 0.09,
                  padding: const EdgeInsets.symmetric(horizontal: 10),
                  decoration: BoxDecoration(
                    color: Colors.black.withOpacity(0.3),
                    borderRadius: BorderRadius.circular(20),
                  ), // BoxDecoration
                  child: Center(
                    child: TextFormField(
                      onFieldSubmitted: (String s) {
                        setState(() {
                          cityname = s;
                          getCityWeather(s);
                          isLoaded = false;
                          controller.clear();
                        });
                      },
                      controller: controller,
                      cursorColor: Colors.white,
                      style: const TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.w600,
                        color: Colors.white,
                      ), // TextStyle
                      decoration: InputDecoration(
                        hintText: 'Search city',
                        hintStyle: TextStyle(
                          fontSize: 18,
                          color: Colors.white.withOpacity(0.7),
                          fontWeight: FontWeight.w600,
                        ), // TextStyle
                        prefixIcon: Icon(
                          Icons.search_rounded,
                          size: 25,
                          color: Colors.white.withOpacity(0.7),
                        ), // Icon
                        border: InputBorder.none,
                      ), // InputDecoration
                    ), // TextFormField
                  ), // Center
                ), // Container
                const SizedBox(height: 30),
                // City Name
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Row(
```

Result :



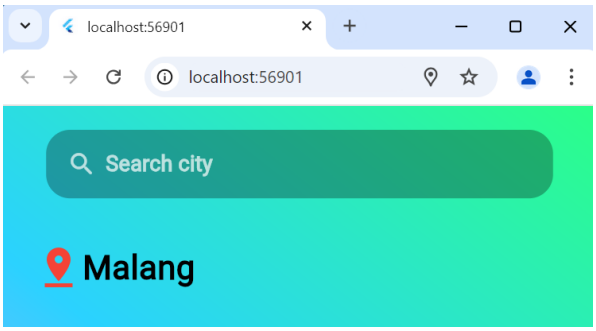
Afterwards in order for minimum memory usage, we will dispose the controller in the dispose method of Stateful widget as shown below.

```dart
const String apiKey = "88e9a89122bb070736ba25a11d16676c";

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

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

class _HomeScreenState extends State<HomeScreen> {
  bool isLoaded = false;
  num? temp, press, hum, cover;
  String cityname = '';
  TextEditingController controller = TextEditingController();

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

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

  void getCurrentLocation() async {
    var position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.low,
    );
    getCurrentCityWeather(position);
    }

  Future<void> getCurrentCityWeather(Position position) async {
```

Next we will add the next child of the Column() which is a SizedBox widget for adding necessary space between the components.

```dart
class _HomeScreenState extends State<HomeScreen> {
  Widget build(BuildContext context) {
                    ),
                    controller: controller,
                    cursorColor: Colors.white,
                    style: const TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.w600,
                      color: Colors.white,
                    ), // TextStyle
                    decoration: InputDecoration(
                      hintText: 'Search city',
                      hintStyle: TextStyle(
                        fontSize: 18,
                        color: Colors.white.withOpacity(0.7),
                        fontWeight: FontWeight.w600,
                      ), // TextStyle
                      prefixIcon: Icon(
                        Icons.search_rounded,
                        size: 25,
                        color: Colors.white.withOpacity(0.7),
                      ), // Icon
                      border: InputBorder.none,
                    ), // InputDecoration
                  ), // TextFormField
                ), // Center
              ), // Container
              const SizedBox(height: 30),
              // City Name
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  crossAxisAlignment: CrossAxisAlignment.end,
                  children: [
                    const Icon(Icons.pin_drop, color: Colors.red, size: 40),
```

This is followed by the City name data.



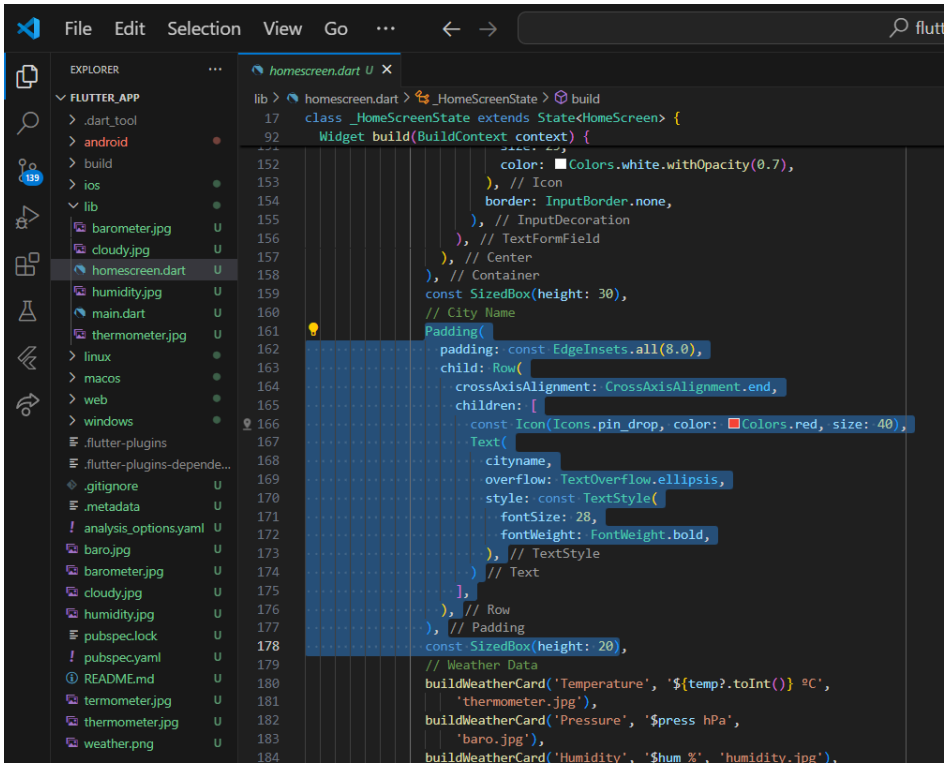
It is implemented by use of a Row() widget wrapped with a Padding widget. It comprises of an Icon and a Text and the code is as given below

This is again followed by a SizedBox widget

SizedBox(

height: 20,

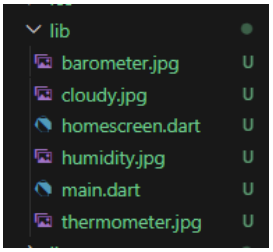
),



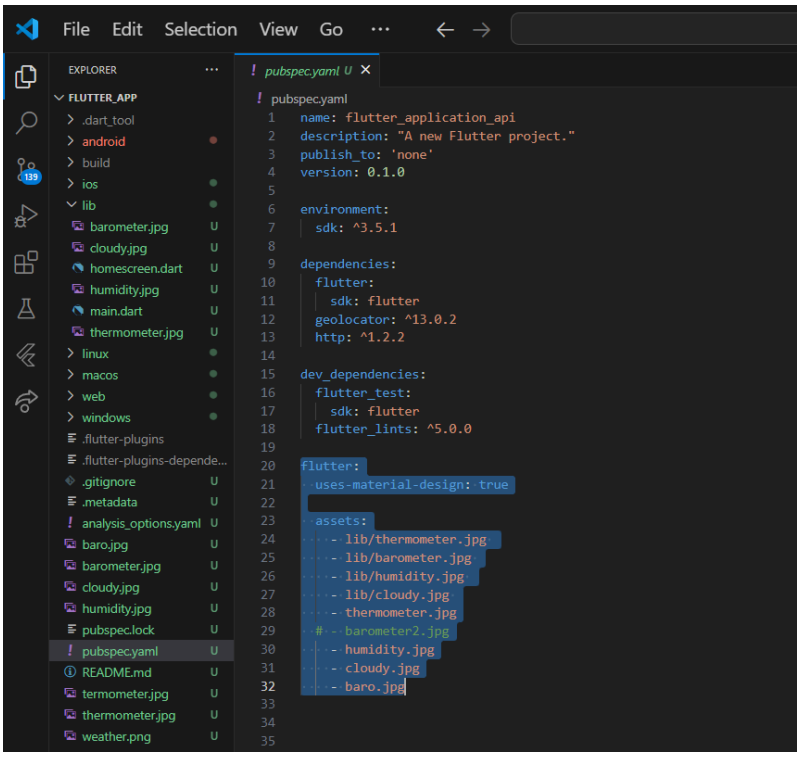
Next comes the weather data section displayed as cards. Here I have used Container for creating custom cards. For the data I have used Image and Text widgets.

Note that the images used here are from the project file itself. So in order to do that we will have to configure those images. For this first create a new directory in the project folder under project_name>New>Directory

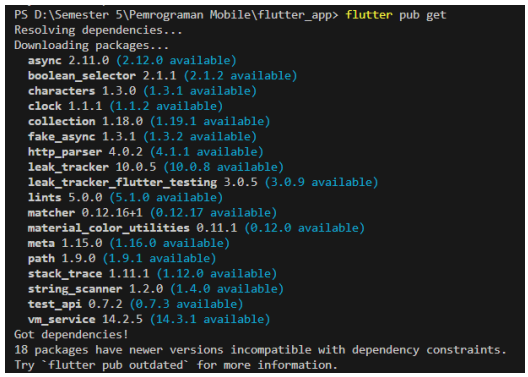
Name the new directory as images. Then once the directory is created add the necessary images.



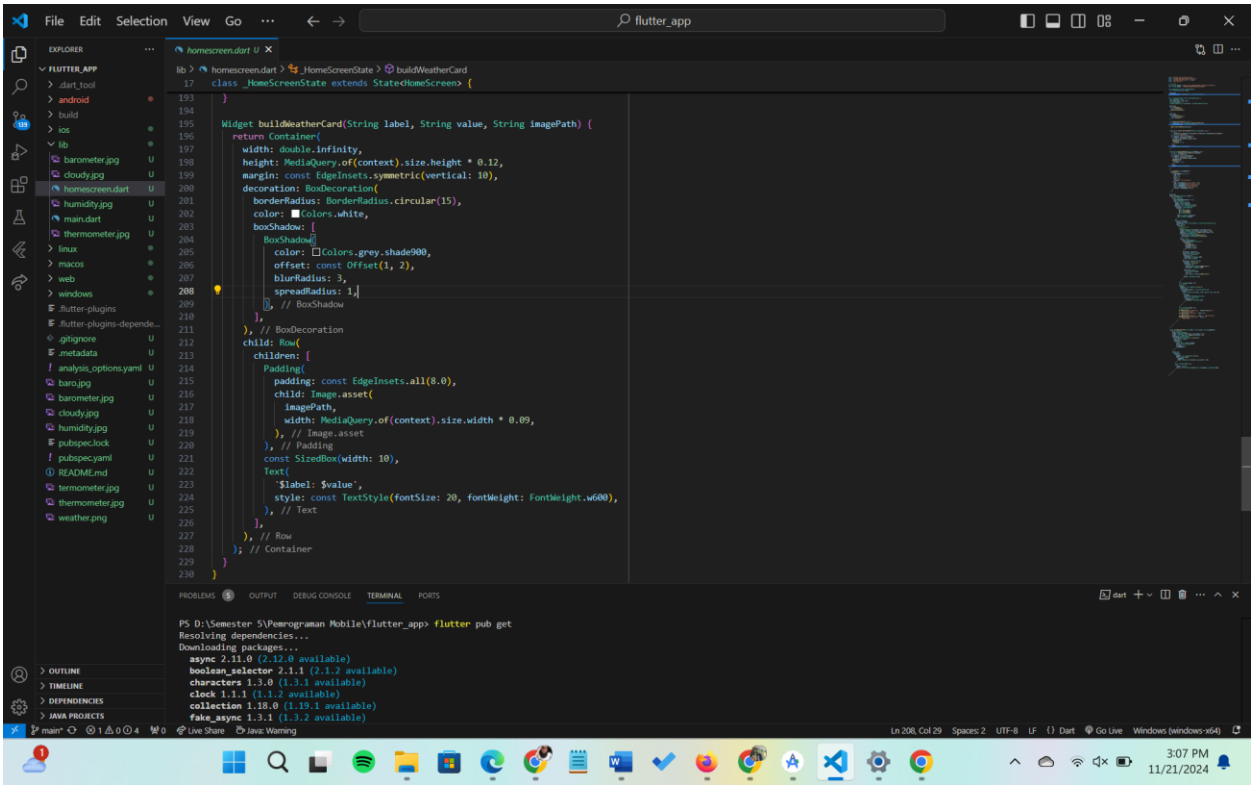
Afterwards, go to your pubspec.yaml file. Scroll down to the asset section where images are incorporated. Uncomment the lines and make sure the spacing is correct as shown below. The below code includes all the files under images directory.



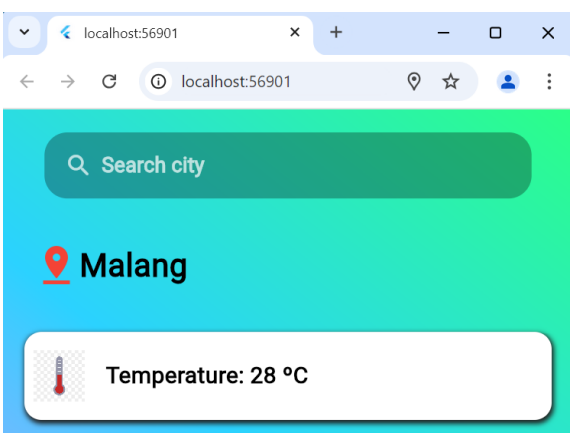
Then run pub get command at the top right corner.



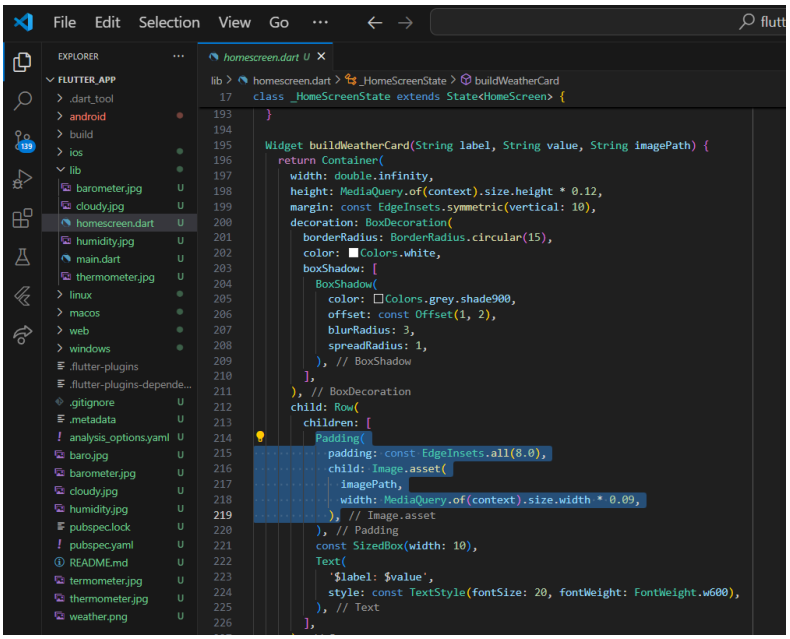
Once that is done, the images will be configured in the app. Now in homescreen.dart we will add the code for creating the card display.



Once the above code is added stop and cold start the app for the configuration changes to be included. When the app is loaded we will get a card display as shown below



Again 3 more cards are added by replacing the image and text for the respective weather parameters. For the next 3 card displays, the images are wrapped with a padding on all sides.



Once that is done we will get an output as shown below