Weather Application

Using OpenWeatherMap API

(Cross-Platform)

- Technical Report -

Project Type: Build

Project Date: 14th January 2022

Table of Contents

List of Figures

List of Tables

1.0 Weather Application

1.1 Introduction

This project would be focused on developing a mobile weather application, based on flutter cross-platform development. Using this application, users would be able to view the weather information for the locations of various countries by searching the city name. Moreover, the application would be able to provide the weather information for an entire week for any selected location. The requirements of the applications are;

- Create a weather application using OpenWeatherMap API.
- Use of mobile development and UI/UX best practices for development.
- Use of suitable architecture, development methodologies and 3rd party libraries.

1.2 Features

01. Search a city for weather.

This feature allows users to search for any city location through the application, to view the weather details of the specific location. This location would be verified using the city name, latitude location as well as its longitude location. This data would be retrieved from the weather API, based on the selected location.

02. View current weather data.

This feature would allow the weather application to retrieve and display the weather information of the current day. This would view the current temperature, humidity percentage, rain probability and the wind speed of the location. This data would be retrieved from the weather API, based on the selected location.

03. Extended today weather details.

This feature allows the application to additionally display the weather data for multiple hours ahead from the current time. This would majorly display the temperature with the time of the current day.

04. Week's weather details.

This application feature displays the weather details up to seven days, from the current day. This would display the temperature, wind speed, humidity percentage and rain probability for the coming seven days.

1.3 Architecture

The selected system architectures for the development of Weather application are listed below.

Table 1: Selected tools and technologies

Application State	Cross-Platform (iOS and Android)
Front-End Language	Dart
Front- End Framework	Flutter
API	OpenWeatherMap API
Development Environment	Android Studio and Visual Studio Code

01. Language: Dart

Dart programming language has been selected to program the requirement Weather application. Dart is a free and open-source OOP (Object-Oriented Programming) language, consisted with C-based composition developed by Google LLC. Dart is majorly used to design the frontend user interfaces for mobile applications. Moreover, Dart is also a compiled language, the code requires to be compiled to machine code before being executed during development. [1]

02. Framework: Flutter

Flutter framework has been selected to assist the application development with Dart language. Flutter is an open-source and free mobile application user interface framework released by Google LLC company. Flutter is majorly popular among developers due to its ability to design a native mobile application using just one codebase. Eventually, this means that developers would find it simpler and more efficient to develop applications based on iOS and Android. Moreover, the Flutter framework is consisted of an SDK (Software Development Kit), which assists with tools used for development. It also consists a separate widget-based UI (User Interface) library, which contains a group of reusable UI elements. [2]

03. State Management: Cross-Platform

The Weather application will be implemented using cross-platform development technique. Cross-platform is a development methodology procedure that allows application implementation to be able work on multiple mobile operating systems concurrently, mainly including iOS, Android and Microsoft Windows. These applications are programmed with native codes along with independent codes supporting multiple platforms, which increases the development efficiency and effectiveness. [3]

04. Protocol: HTTP

Hypertext Transfer Protocol (HTTP) is an application-level protocol which allows the communication between resources among the internet. This protocol would establish the basic communication between internet clients and their specific requests [4]. HTTP requests have been linked in this application in order to link third party libraries and the weather API with the application.

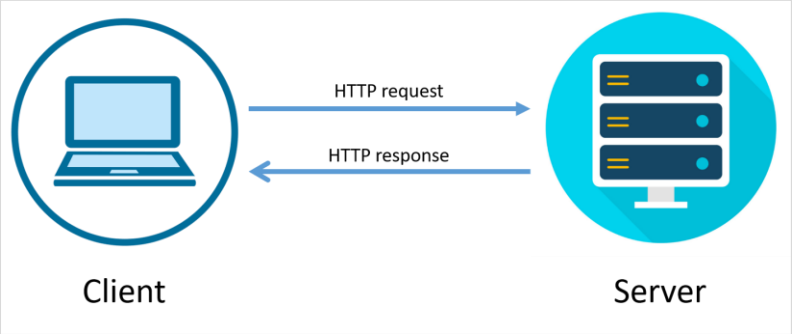


Figure 1: HTTP Request and Response

05. Dart Package: Cupertino Icons

A dart package available to add specific icons in the application. The icons were used to include specific weather-related icons in the weather application.

06. Dart Package: Flutter Glow

A dart package available to add visual glows to application widgets. Glows were used in the application for an attractive look and to separate weather components.

1.4 Third Party Library

OpenWeatherMap

OpenWeatherMap is considered as an online global weather information providing service. The service also provides business-based products using climate data. OpenWeatherMap contains real-time weather data for locations around the globe, using their own weather prediction system. OpenWeatherMap uses API keys to allow internet users to access their climate data, including current weather, hourly forecast, daily forecast, climatic forecast and even historical weather.

OpenWeatherMap users would be required to create an account on their website, receiving an API key to access the weather data from any application system. Users would also be able to generate multiple API keys as required. These weather data can be called using city names, city codes, zip codes or even geographical coordinates [5]. Similarly, this project application is developed using OpenWeatherMap API key, in order to receive the required weather data to be displayed in the weather application.

Figure 2: OpenWeatherMap

Countries States Cities Database Library

This database library, developed by [6], is a collection including the geographical data regarding countries, states and its cities. The library includes all the required data required to identify and verify geographical locations based on unique identities. The library was linked in the current project to assist the weather city searching feature of the application. Once a city name is searched in the application, the specific city's relevant geographical coordinates are retrieved from this library. Coordinates are verified, and then the relevant weather data would be retrieved from the OpenWeatherMap API source using the coordinates.

1.5 UI/UX Design

Wireframe User Interface Design

01. View current weather details and hourly forecast:

Figure 3: Wireframe - View current weather details and hourly forecast

02. View weather details for coming week:

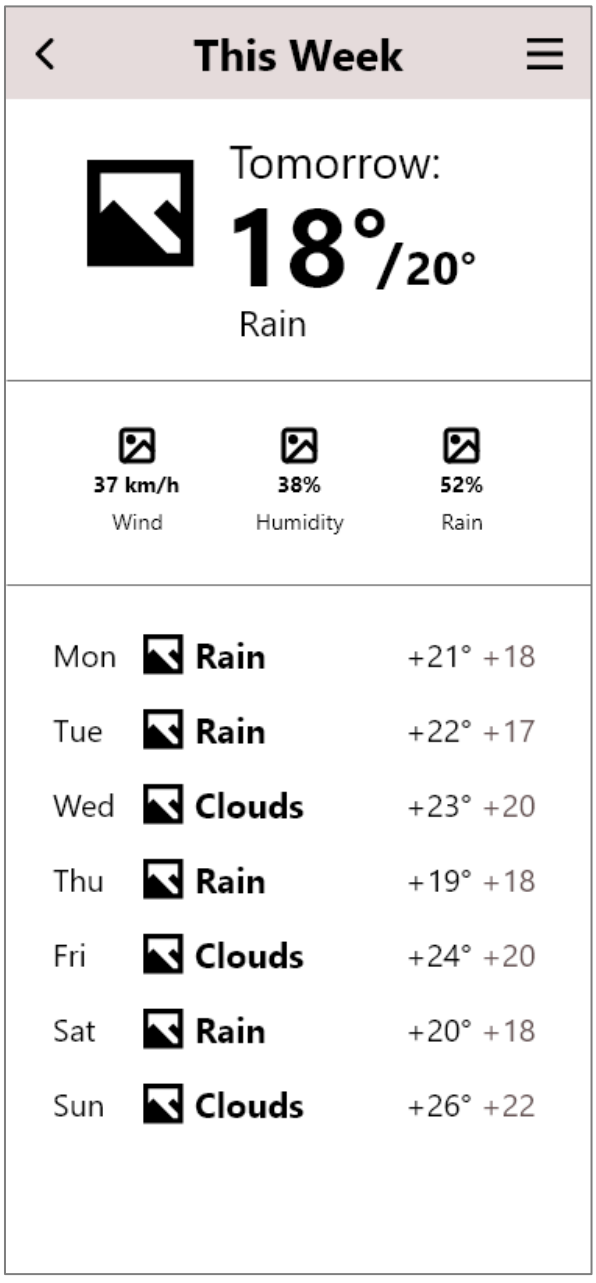


Figure 4: Wireframe - View weather details for coming week

03. Search for a specific city:

Figure 5: Wireframe - Search for a specific city

Implemented User Interface Design

01. View current weather details and hourly forecast:

Figure 6: Implemented Design - View current weather details and hourly forecast

02. View weather details for coming week:

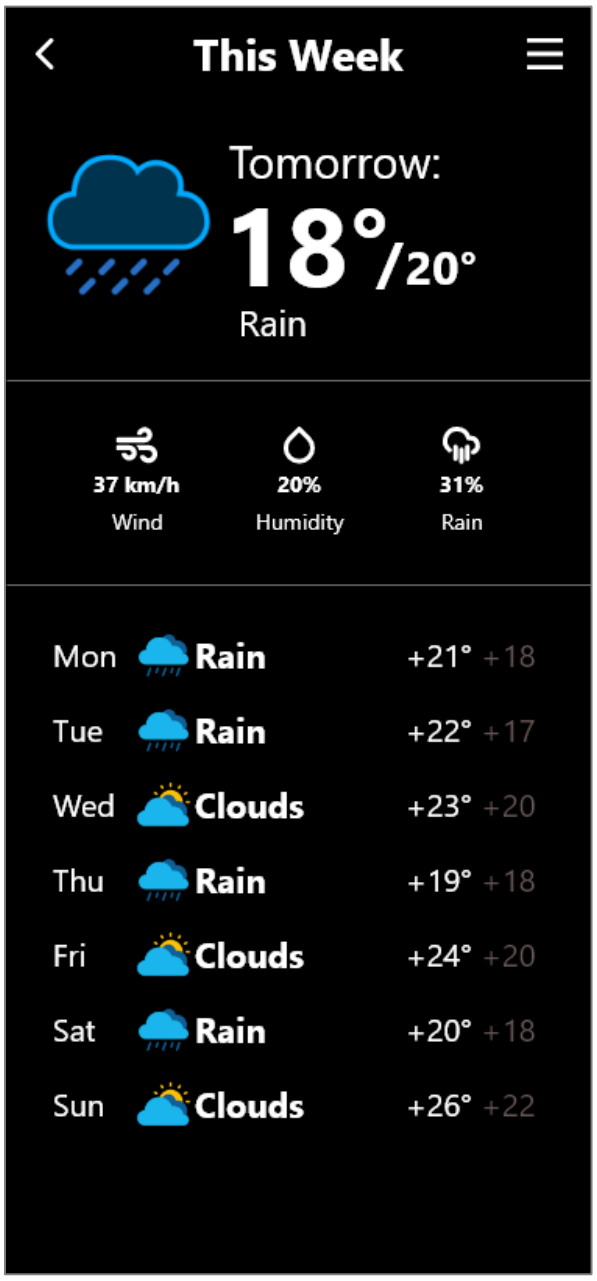


Figure 7: Implemented Design - View weather details for coming week

03. Search for a specific city:

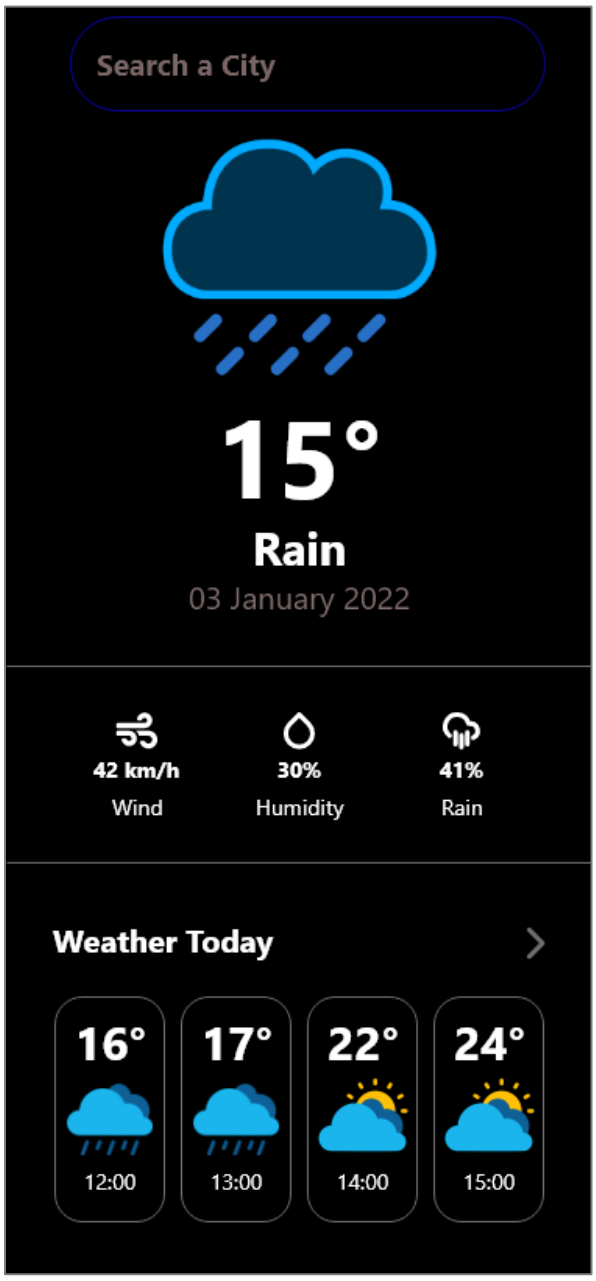


Figure 8: Implemented Design - Search for a specific city

1.6 Implementation

View current weather details and hourly forecast function:

Figure 9: Function - View current weather details and hourly forecast function

View weather details for coming week function:

Figure 10: Function - View weather details for coming week function

Search for a specific city function:

Figure 11: Function - Search for a specific city function

Incorrect city name error:

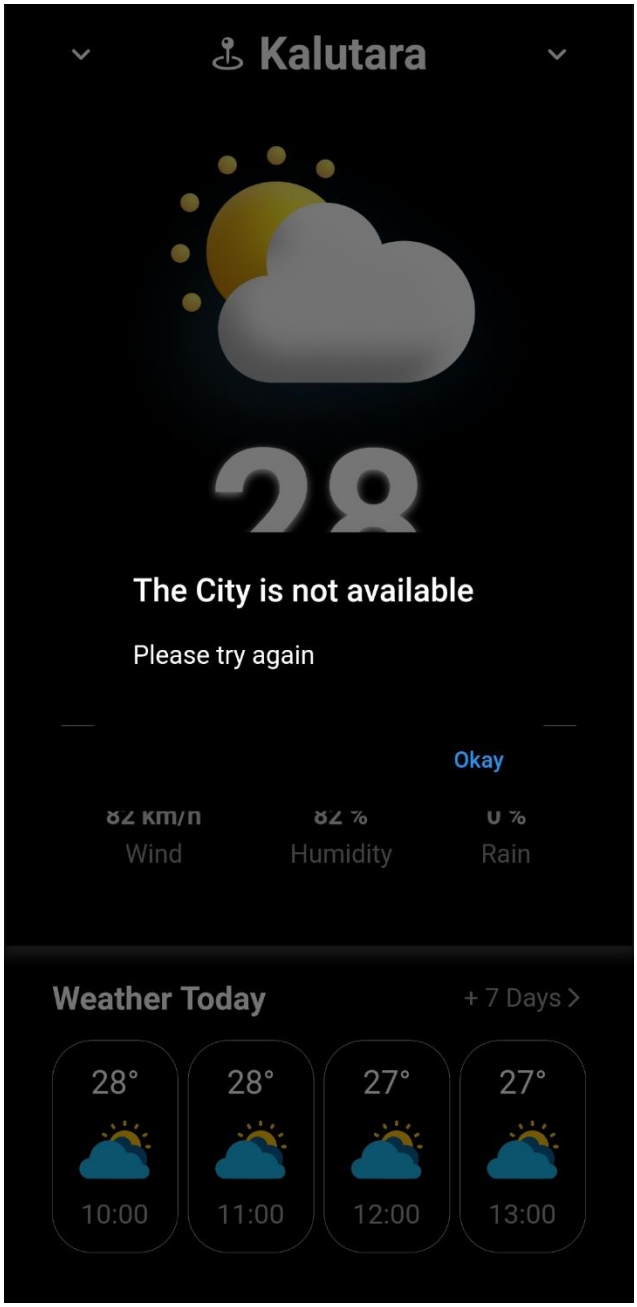


Figure 12: Function - Incorrect city name error

Folder Structure:

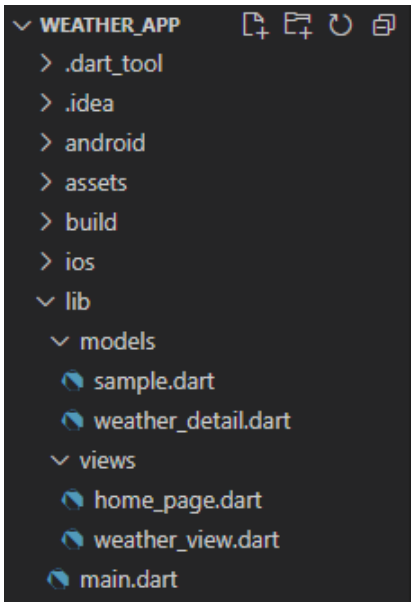


Figure 13: Folder Structure

Codes snippets:

- lib > main.dart:

```dart
import 'package:flutter/material.dart';
import 'Views/home_page.dart';


void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Weather',
      theme: ThemeData(
          textTheme: Theme.of(context)
              .textTheme
              .apply(bodyColor: Colors.white, displayColor: Colors.blue)),
      debugShowCheckedModeBanner: false,
      home: Home(),
    );
  }
}
```

Figure 14: Code Snippet - main.dart

- lib > views > home_page.dart:

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_glow/flutter_glow.dart';
import 'package:weather_app/models/sample.dart';
import 'package:weather_app/models/weather_detail.dart';
import 'package:weather_app/views/weather_view.dart';

Weather temperatureNow;
Weather temperatureTomorrow;
List<Weather> weatherToday;
List<Weather> weatherWeek;
String lat = "6.93106260";
String lon = "79.97944220";
String city = "Kaduwela";

class Home extends StatefulWidget {
  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  getData() async {
    fetchData(lat, lon, city).then((value) {
      temperatureNow = value[0];
      weatherToday = value[1];
      temperatureTomorrow = value[2];
      weatherWeek = value[3];
      setState(() {});
    });
  }

  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    getData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      backgroundColor: Colors.black,
      body: temperatureNow == null
          ? Center(
              child: CircularProgressIndicator(),
            ) // Center
          : Column(
              children: [WeatherNow(getData), WeatherToday()],
            ), // Column
    ); // Scaffold
  }
}
```

Figure 15: Code Snippet - home_page.dart #01

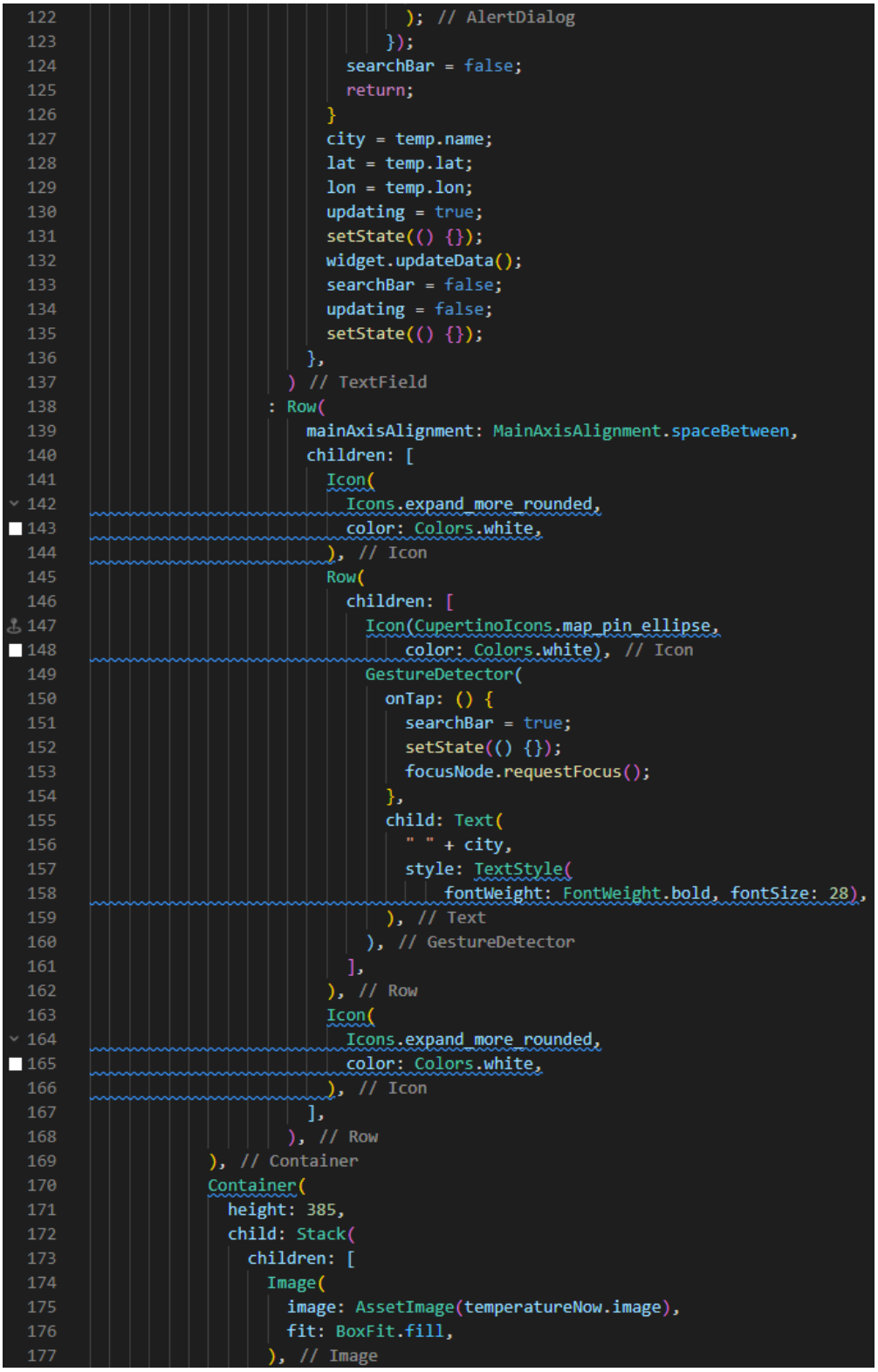
```dart
class WeatherNow extends StatefulWidget {
  final Function() updateData;
  WeatherNow(this.updateData);

  @override
  State<WeatherNow> createState() => _WeatherNowState();
}

class _WeatherNowState extends State<WeatherNow> {
  bool searchBar = false;
  bool updating = false;
  var focusNode = FocusNode();

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        if (searchBar)
          setState(() {
            searchBar = false;
          });
      },
      child: GlowContainer(
        height: MediaQuery.of(context).size.height - 250,
        margin: EdgeInsets.all(1),
        padding: EdgeInsets.only(top: 50, left: 35, right: 35),
        glowColor: Colors.white.withOpacity(0.2),
        borderRadius: BorderRadius.only(
            bottomLeft: Radius.circular(0.1),
            bottomRight: Radius.circular(0.1)), // BorderRadius.only
        color: Colors.black,
        spreadRadius: 5,
        child: Column(
          children: [
            Container(
              child: searchBar
                  ? TextField(
                      focusNode: focusNode,
                      decoration: InputDecoration(
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(50),
                        ), // OutlineInputBorder
                        fillColor: Colors.black,
                        filled: true,
                        hintText: 'Search a City',
                        hintStyle: TextStyle(color: Colors.white54),
                      ), // InputDecoration
                      textInputAction: TextInputAction.search,
                      onSubmitted: (value) async {
                        CityModel temp = await fetchCity(value);
                        if (temp == null) {
                          showDialog(
                              context: context,
                              builder: (BuildContext context) {
                                return AlertDialog(
                                  backgroundColor: Colors.black,
                                  title: Text("The City is not available"),
                                  content: Text("Please try again"),
                                  actions: [
                                    TextButton(
                                        onPressed: () {
                                          Navigator.of(context).pop();
                                        },
                                        child: Text("Okay")) // TextButton
                                  ],
```

Figure 16: Code Snippet - home_page.dart #02

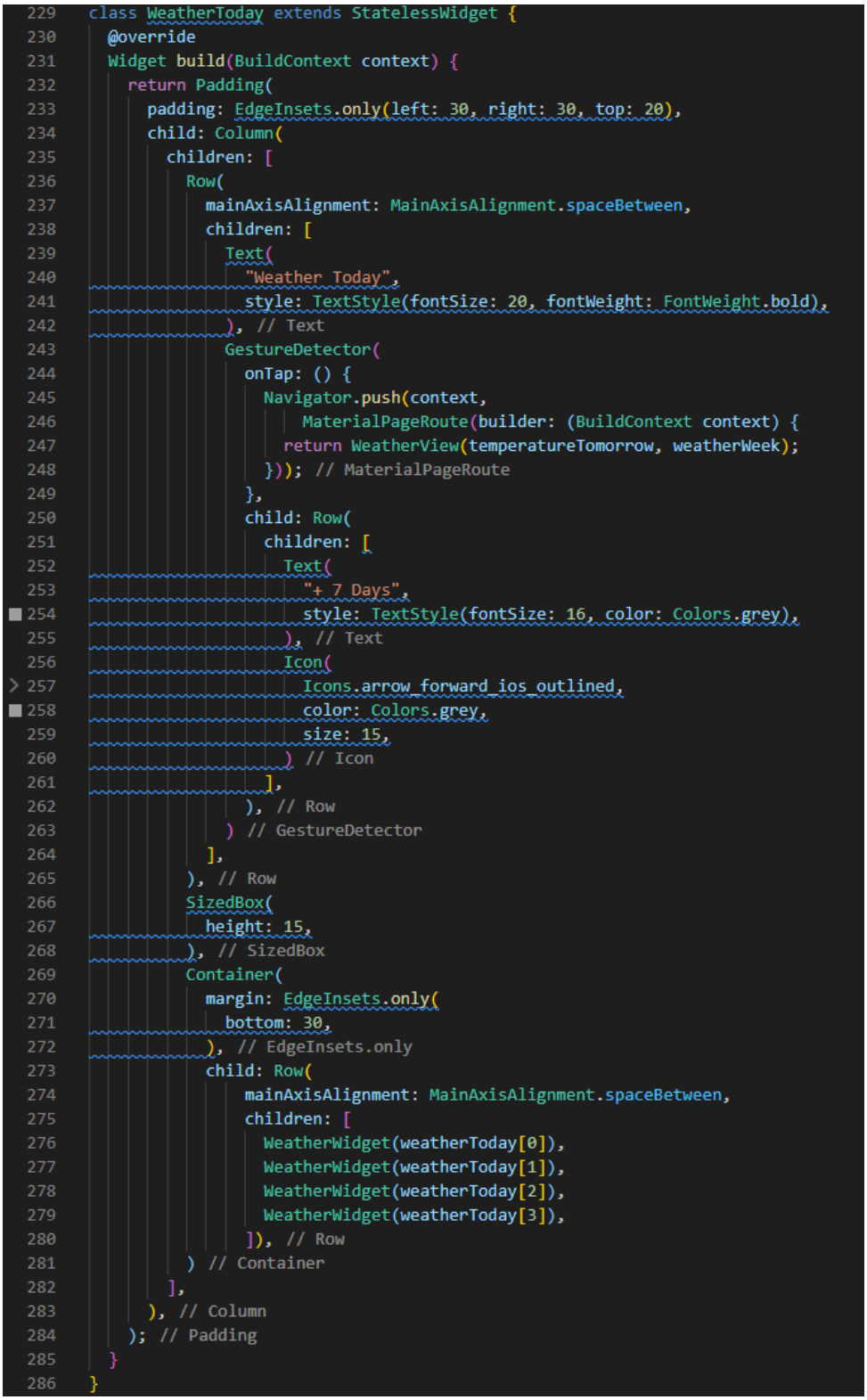


```
                                ); // AlertDialog
                              });
                          searchBar = false;
                          return;
                        }
                        city = temp.name;
                        lat = temp.lat;
                        lon = temp.lon;
                        updating = true;
                        setState(() {});
                        widget.updateData();
                        searchBar = false;
                        updating = false;
                        setState(() {});
                      },
                    ) // TextField
                  : Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: [
                        Icon(
                          Icons.expand_more_rounded,
                          color: Colors.white,
                        ), // Icon
                        Row(
                          children: [
                            Icon(CupertinoIcons.map_pin_ellipse,
                                color: Colors.white), // Icon
                            GestureDetector(
                              onTap: () {
                                searchBar = true;
                                setState(() {});
                                focusNode.requestFocus();
                              },
                              child: Text(
                                " " + city,
                                style: TextStyle(
                                    fontWeight: FontWeight.bold, fontSize: 28),
                              ), // Text
                            ), // GestureDetector
                          ],
                        ), // Row
                        Icon(
                          Icons.expand_more_rounded,
                          color: Colors.white,
                        ), // Icon
                      ],
                    ), // Row
            ), // Container
            Container(
              height: 385,
              child: Stack(
                children: [
                  Image(
                    image: AssetImage(temperatureNow.image),
                    fit: BoxFit.fill,
                  ), // Image
```

Figure 17: Code Snippet - home_page.dart #03

```
178                   Positioned(
179                     bottom: 0,
180                     right: 0,
181                     left: 0,
182                     child: Center(
183                       child: Column(
184                         children: [
185                           GlowText(
186                             temperatureNow.current.toString(),
187                             style: TextStyle(
188                               height: 1.0,
189                               fontSize: 120,
190                               fontWeight: FontWeight.bold), // TextStyle
191                           ), // GlowText
192                           Text(
193                             temperatureNow.name,
194                             style: TextStyle(
195                               fontSize: 22,
196                             ), // TextStyle
197                           ), // Text
198                           SizedBox(
199                             height: 1.0,
200                           ), // SizedBox
201                           Text(
202                             temperatureNow.day,
203                             style: TextStyle(
204                               fontSize: 15,
205                             ), // TextStyle
206                           ), // Text
207                         ],
208                       ), // Column
209                     ), // Center
210                   ) // Positioned
211                 ],
212               ), // Stack
213             ), // Container
214             Divider(
215               color: Colors.white,
216               height: 35,
217             ), // Divider
218             SizedBox(
219               height: 1,
220             ), // SizedBox
221             WeatherDetail(temperatureNow)
222           ],
223         ), // Column
224       ), // GlowContainer
225     ); // GestureDetector
226   }
227 }
```

Figure 18: Code Snippet - home_page.dart #04



```dart
class WeatherToday extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.only(left: 30, right: 30, top: 20),
      child: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(
                "Weather Today",
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
              ), // Text
              GestureDetector(
                onTap: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (BuildContext context) {
                    return WeatherView(temperatureTomorrow, weatherWeek);
                  })); // MaterialPageRoute
                },
                child: Row(
                  children: [
                    Text(
                      "+ 7 Days",
                      style: TextStyle(fontSize: 16, color: Colors.grey),
                    ), // Text
                    Icon(
                      Icons.arrow_forward_ios_outlined,
                      color: Colors.grey,
                      size: 15,
                    ) // Icon
                  ],
                ), // Row
              ) // GestureDetector
            ],
          ), // Row
          SizedBox(
            height: 15,
          ), // SizedBox
          Container(
            margin: EdgeInsets.only(
              bottom: 30,
            ), // EdgeInsets.only
            child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  WeatherWidget(weatherToday[0]),
                  WeatherWidget(weatherToday[1]),
                  WeatherWidget(weatherToday[2]),
                  WeatherWidget(weatherToday[3]),
                ]), // Row
          ) // Container
        ],
      ), // Column
    ); // Padding
  }
}
```

Figure 19: Code Snippet - home_page.dart #05

```dart
class WeatherWidget extends StatelessWidget {
  final Weather weather;

  WeatherWidget(this.weather);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(14),
      decoration: BoxDecoration(
          border: Border.all(width: 0.3, color: Colors.white),
          borderRadius: BorderRadius.circular(25)), // BoxDecoration
      child: Column(
        children: [
          Text(
            weather.current.toString() + "\u00B0",
            style: TextStyle(fontSize: 20.0),
          ), // Text
          SizedBox(
            height: 6,
          ), // SizedBox
          Image(
            image: AssetImage(weather.image),
            width: 50,
            height: 50,
          ), // Image
          SizedBox(
            height: 5,
          ), // SizedBox
          Text(
            weather.time,
            style: TextStyle(fontSize: 17, color: Colors.grey),
          ) // Text
        ],
      ), // Column
    ); // Container
  }
}
```

Figure 20: Code Snippet - home_page.dart #06

- lib > views > weather_view.dart:

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_glow/flutter_glow.dart';
import 'package:weather_app/models/sample.dart';
import 'package:weather_app/models/weather_detail.dart';

class WeatherView extends StatelessWidget {
  final Weather temperatureTomorrow;
  final List<Weather> weatherWeek;

  WeatherView(this.temperatureTomorrow, this.weatherWeek);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Column(
        children: [WeatherTomorrow(temperatureTomorrow), Week(weatherWeek)],
      ), // Column
    ); // Scaffold
  }
}

class WeatherTomorrow extends StatelessWidget {
  final Weather temperatureTomorrow;
  WeatherTomorrow(this.temperatureTomorrow);

  @override
  Widget build(BuildContext context) {
    return GlowContainer(
      color: Colors.black,
      glowColor: Colors.white,
      borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(0.1), bottomRight: Radius.circular(0.1)),
      child: Column(
        children: [
          Padding(
            padding: EdgeInsets.only(top: 50, right: 30, left: 30, bottom: 20),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                GestureDetector(
                    onTap: () {
                      Navigator.pop(context);
                    },
                    child: Icon(
                      Icons.arrow_back_ios,
                      color: Colors.white,
                    )), // Icon // GestureDetector
```

Figure 21: Code Snippet - weather_view.dart #01

```
                    Row(
                      children: [
                        Icon(
                          Icons.calendar_today,
                          color: Colors.white,
                        ), // Icon
                        Text(
                          " This Week", //Weather this week text
                          style:
                              TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
                        ) // Text
                      ],
                    ), // Row
                    Icon(
                      Icons.more_vert,
                      color: Colors.white,
                    ) // Icon
                  ],
                ), // Row
              ), // Padding
              Padding(
                padding: EdgeInsets.all(8),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Container(
                      width: MediaQuery.of(context).size.width / 2.3,
                      height: MediaQuery.of(context).size.width / 2.3,
                      decoration: BoxDecoration(
                          image: DecorationImage(
                              image: AssetImage(temperatureTomorrow.image))), // Dec
                    ), // Container
                    Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Text(
                          "Tomorrow:",
                          style: TextStyle(fontSize: 25, height: 0.1),
                        ), // Text
                        Container(
                          height: 105,
                          child: Row(
                            crossAxisAlignment: CrossAxisAlignment.end,
                            children: [
                              GlowText(
                                temperatureTomorrow.max.toString(),
                                style: TextStyle(
                                    fontSize: 100, fontWeight: FontWeight.bold), //
                              ), // GlowText
                              Text(
                                "/" + temperatureTomorrow.min.toString() + "\u00B0",
                                style: TextStyle(
                                    color: Colors.white70.withOpacity(0.3),
                                    fontSize: 40,
                                    fontWeight: FontWeight.bold), // TextStyle
                              ), // Text
                            ],
                          ), // Row
                        ), // Container
```

Figure 22: Code Snippet - weather_view.dart #02

```
                    SizedBox(
                      height: 10,
                    ), // SizedBox
                    Text(
                      " " + temperatureTomorrow.name,
                      style: TextStyle(
                        fontSize: 18,
                      ), // TextStyle
                    ) // Text
                  ],
                ) // Column
              ],
            ), // Row
          ), // Padding
          Padding(
            padding: EdgeInsets.only(
              bottom: 20,
              right: 50,
              left: 50,
            ), // EdgeInsets.only
            child: Column(
              children: [
                Divider(
                  color: Colors.white,
                ), // Divider
                SizedBox(
                  height: 10,
                ), // SizedBox
                WeatherDetail(temperatureTomorrow)
              ],
            ), // Column
          ) // Padding
        ],
      ), // Column
    ); // GlowContainer
  }
}

class Week extends StatelessWidget {
  List<Weather> weatherWeek;
  Week(this.weatherWeek);
```

Figure 23: Code Snippet - weather_view.dart #03

```dart
@override
Widget build(BuildContext context) {
  return Expanded(
    child: ListView.builder(
        itemCount: weatherWeek.length,
        itemBuilder: (BuildContext context, int index) {
          return Padding(
              padding: EdgeInsets.only(left: 20, right: 20, bottom: 25),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    weatherWeek[index].day,
                    style: TextStyle(fontSize: 20),
                  ), // Text
                  Container(
                    width: 135,
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.start,
                      children: [
                        Image(
                          image: AssetImage(weatherWeek[index].image),
                          width: 40,
                        ), // Image
                        SizedBox(width: 15),
                        Text(
                          weatherWeek[index].name,
                          style: TextStyle(fontSize: 20),
                        ) // Text
                      ],
                    ), // Row
                  ), // Container
                  Row(
                    children: [
                      Text(
                        "+" + weatherWeek[index].max.toString() + "\u00B0",
                        style: TextStyle(fontSize: 20),
                      ), // Text
                      SizedBox(
                        width: 5,
                      ), // SizedBox
                      Text(
                        "+" + weatherWeek[index].min.toString() + "\u00B0",
                        style: TextStyle(fontSize: 20, color: Colors.grey),
                      ), // Text
                    ],
                  ) // Row
                ],
              )); // Row // Padding
        }), // ListView.builder
  ); // Expanded
}
}
```

Figure 24: Code Snippet - weather_view.dart #04

- lib > models > sample.dart:

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

class Weather {
  final int max;
  final int min;
  final int current;
  final String name;
  final String day;
  final int wind;
  final int humidity;
  final int rainChance;
  final String image;
  final String time;
  final String location;

  Weather(
      {this.max,
      this.min,
      this.name,
      this.day,
      this.wind,
      this.humidity,
      this.rainChance,
      this.image,
      this.current,
      this.time,
      this.location});
}

String appId = "b7a3104a6fea4aa1cdadd1e2a6c3be3c";

Future<List> fetchData(String lat, String lon, String city) async {
  var url =
      "https://api.openweathermap.org/data/2.5/onecall?lat=$lat&lon=$lon&units=metric&appid=$appId";
  var response = await http.get(Uri.parse(url));
  DateTime date = DateTime.now();
  if (response.statusCode == 200) {
    var res = json.decode(response.body);

    //Temperature Now
    var current = res["current"];
    Weather temperatureNow = Weather(
        current: current["temp"]?.round() ?? 0,
        name: current["weather"][0]["main"].toString(),
        day: DateFormat("EEEE dd MMMM").format(date),
        wind: current["wind_speed"]?.round() ?? 0,
        humidity: current["humidity"]?.round() ?? 0,
        rainChance: current["uvi"]?.round() ?? 0,
        location: city,
        image: findIcon(current["weather"][0]["main"].toString(), true)); // Weather
```

Figure 25: Code Snippet - sample.dart #01

```dart
    //Weather Today
    List<Weather> todayWeather = [];
    int hour = int.parse(DateFormat("hh").format(date));
    for (var i = 0; i < 4; i++) {
      var temp = res["hourly"];
      var hourly = Weather(
          current: temp[i]["temp"]?.round() ?? 0,
          image: findIcon(temp[i]["weather"][0]["main"].toString(), false),
          time: Duration(hours: hour + i + 1).toString().split(":")[0] + ":00"); // Weather
      todayWeather.add(hourly);
    }

    //Weather Tomorrow
    var daily = res["daily"][0];
    Weather temperatureTomorrow = Weather(
        max: daily["temp"]["max"]?.round() ?? 0,
        min: daily["temp"]["min"]?.round() ?? 0,
        image: findIcon(daily["weather"][0]["main"].toString(), true),
        name: daily["weather"][0]["main"].toString(),
        wind: daily["wind_speed"]?.round() ?? 0,
        humidity: daily["rain"]?.round() ?? 0,
        rainChance: daily["uvi"]?.round() ?? 0);

    //Week's Weather
    List<Weather> weatherWeek = [];
    for (var i = 1; i < 8; i++) {
      String day = DateFormat("EEEE")
          .format(DateTime(date.year, date.month, date.day + i + 1))
          .substring(0, 3);
      var temp = res["daily"][i];
      var hourly = Weather(
          max: temp["temp"]["max"]?.round() ?? 0,
          min: temp["temp"]["min"]?.round() ?? 0,
          image: findIcon(temp["weather"][0]["main"].toString(), false),
          name: temp["weather"][0]["main"].toString(),
          day: day);
      weatherWeek.add(hourly);
    }
    return [temperatureNow, todayWeather, temperatureTomorrow, weatherWeek];
  }
  return [null, null, null, null];
}

//Icon Identifier
String findIcon(String name, bool type) {
  if (type) {
    switch (name) {
      case "Clouds":
        return "assets/cloudstate.png";
        break;
      case "Rain":
        return "assets/rainstate.png";
        break;
      case "Drizzle":
        return "assets/rainstate.png";
        break;
      case "Thunderstorm":
        return "assets/storm.png";
        break;
      case "Snow":
        return "assets/snowstate.png";
        break;
      default:
        return "assets/cloudstate.png";
    }
```

Figure 26: Code Snippet - sample.dart #02

```dart
  } else {
    switch (name) {
      case "Clouds":
        return "assets/cloudstate_extended.png";
        break;
      case "Rain":
        return "assets/rainstate_extended.png";
        break;
      case "Drizzle":
        return "assets/rainstate_extended.png";
        break;
      case "Thunderstorm":
        return "assets/storm_extended.png";
        break;
      case "Snow":
        return "assets/snowstate_extended.png";
        break;
      default:
        return "assets/cloudstate_extended.png";
    }
  }
}

class CityModel {
  final String name;
  final String lat;
  final String lon;
  CityModel({this.name, this.lat, this.lon});
}

var cityJSON;

Future<CityModel> fetchCity(String cityName) async {
  if (cityJSON == null) {
    String link =
        "https://raw.githubusercontent.com/dr5hn/countries-states-cities-database/master/cities.json";
    var response = await http.get(Uri.parse(link));
    if (response.statusCode == 200) {
      cityJSON = json.decode(response.body);
    }
  }
  for (var i = 0; i < cityJSON.length; i++) {
    if (cityJSON[i]["name"].toString().toLowerCase() ==
        cityName.toLowerCase()) {
      return CityModel(
          name: cityJSON[i]["name"].toString(),
          lat: cityJSON[i]["latitude"].toString(),
          lon: cityJSON[i]["longitude"].toString());
    }
  }
  return null;
}
```

Figure 27: Code Snippet - sample.dart #03

- lib > models > weather_detail.dart:

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:weather_app/models/sample.dart';

class WeatherDetail extends StatelessWidget {
  final Weather temp;
  WeatherDetail(this.temp);

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: [
        Column(
          children: [
            Icon(
              CupertinoIcons.wind,
              color: Colors.white,
            ), // Icon
            SizedBox(
              height: 4,
            ), // SizedBox
            Text(
              temp.humidity.toString() + " km/h ",
              style: TextStyle(fontWeight: FontWeight.w700, fontSize: 16),
            ), // Text
            SizedBox(
              height: 5,
            ), // SizedBox
            Text(
              "Wind",
              style: TextStyle(color: Colors.grey, fontSize: 16),
            ) // Text
          ],
        ), // Column
```

Figure 28: Code Snippet - weather_detail.dart #01

```dart
          Column(
            children: [
              Icon(
                CupertinoIcons.drop,
                color: Colors.white,
              ), // Icon
              SizedBox(
                height: 4,
              ), // SizedBox
              Text(
                temp.humidity.toString() + " %",
                style: TextStyle(fontWeight: FontWeight.w700, fontSize: 16),
              ), // Text
              SizedBox(
                height: 5,
              ), // SizedBox
              Text(
                "Humidity",
                style: TextStyle(color: Colors.grey, fontSize: 16),
              ) // Text
            ],
          ), // Column
          Column(
            children: [
              Icon(
                CupertinoIcons.cloud_heavyrain,
                color: Colors.white,
              ), // Icon
              SizedBox(
                height: 4,
              ), // SizedBox
              Text(
                temp.rainChance.toString() + " %",
                style: TextStyle(fontWeight: FontWeight.w700, fontSize: 16),
              ), // Text
              SizedBox(
                height: 5,
              ), // SizedBox
              Text(
                "Rain",
                style: TextStyle(color: Colors.grey, fontSize: 16),
              ) // Text
            ],
          ) // Column
        ],
      ); // Row
    }
}
```

Figure 29: Code Snippet - weather_detail.dart #02

1.7 Test Case

Test Scope:

The major scope of the current testing design is to make sure that the mobile weather application properly functions correspond with the third-party libraries, as well as meet the proper requirements.

Test Objectives:

- Test essential parts of the application for proper functionalities.
- Check if the application properly connects with third-party libraries and retrieve data.
- Check the efficiency and effectiveness of the application.

Test Approach:

The current testing design for the mobile weather application;

1. Uses a common test case template for each test case.
2. Test fundamental structures and functions of the application.
3. Test one section of every essential task in the database system.

Test Case Template:

Table 2: Test case template

Test Description				
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**

Testing Targets:

Table 3: Testing targets

Test Explanation for Mobile Weather Application		
Test Case ID	**Test Description**	**Test Date**
01	Retrieve the specified default city's current and today's weather.	13/01/2021
02	Retrieve the default city's weather this week.	13/01/2021
03	Search for a city's current and today's weather.	13/01/2021
04	View the searched city's weather this week.	13/01/2021
05	Verify that the search function with false data.	13/01/2021

Test Implementation:

Table 4: Test Case 01 - Retrieve the specified default city's current and today's weather.

Test Description	Retrieve the specified default city's current and today's weather.			
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**
01	**City:** "Kaduwela" **Latitude:** "6.93106260" **Longitude:** "79.97944220"	View "Kaduwela" city's current and today's weather.	View "Kaduwela" city's current and today's weather.	PASS

Evidence:

Figure 30: Test Case 01 - Test Evidence

Table 5: Test Case 02 - Retrieve the default city's weather this week

Test Description	Retrieve the default city's weather this week.			
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**
02	**City:** "Kaduwela" **Latitude:** "6.93106260" **Longitude:** "79.97944220"	View "Kaduwela" city's weather this week.	View "Kaduwela" city's weather this week.	PASS

Evidence:

Figure 31: Test Case 02 - Test Evidence

Table 6: Test Case 03 - Search for a city's current and today's weather

Test Description	Search for a city's current and today's weather.			
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**
03	**City:** "Kandy"	View "Kandy" city's current and today's weather.	View "Kandy" city's current and today's weather.	PASS

Evidence:

Figure 33: Test Case 03 - Test Evidence #01

Figure 32: Test Case 02 - Test Evidence #02

Table 7: Test Case 04 - View the searched city's weather this week

Test Description	View the searched city's weather this week.			
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**
04	**City:** "Kandy"	View "Kandy" city's weather this week.	View "Kandy" city's weather this week.	PASS

Evidence:

Figure 34: Test Case 04 - Test Evidence

Table 8: Test Case 05 - Verify that the search function with false data

Test Description	Verify that the search function with false data.			
Test Case	**Input Data**	**Expected Outcome**	**Actual Outcome**	**Outcome Result (Pass/Fail)**
05	**City:** "Wrong City"	Error message stating "The city is not available."	Error message stating "The city is not available."	PASS

Evidence:

Figure 36: Test Case 05 - Test Evidence #01

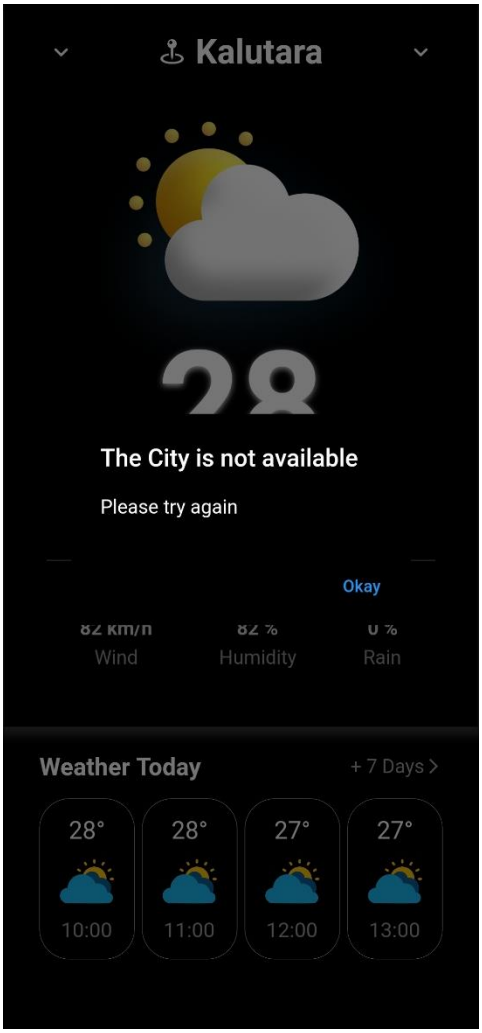


Figure 35: Test Case 05 - Test Evidence #02

References

[1] javaTpoint, "What is dart programming?," [Online]. Available: https://www.javatpoint.com/flutter-dart-programming. [Accessed 12 January 2022].

[2] G. Thomas, "What is Flutter and Why You Should Learn it in 2020," freeCodeCamp, 12 December 2019. [Online]. Available: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/. [Accessed 12 January 2022].

[3] N. Sakovich, "Cross-Platform Mobile Development: Five Best Frameworks," 22 June 2018. [Online]. Available: https://www.sam-solutions.com/blog/cross-platform-mobile-development/. [Accessed 12 January 2022].

[4] TutorialsPoint, "HTTP Tutorial," [Online]. Available: https://www.tutorialspoint.com/http/index.htm. [Accessed 12 January 2022].

[5] "OpenWeather," OpenWeather.org, [Online]. Available: https://openweathermap.org/. [Accessed 08 January 2022].

[6] D. Gada, "Countries States Cities Database," dr5hn, 11 December 2021. [Online]. Available: https://countrystatecity.in/docs/. [Accessed 08 January 2022].