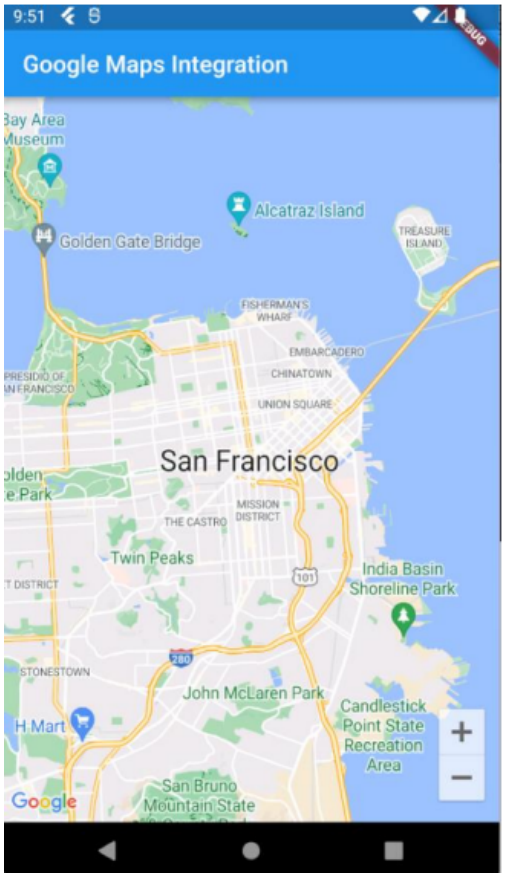
1. Develop an application by integrating Google maps
And
16 Develop an application by integrating Google maps

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


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


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyMap(),
    );
  }
}


class MyMap extends StatefulWidget {
  @override
  _MyMapState createState() => _MyMapState();
}


class _MyMapState extends State<MyMap> {
  GoogleMapController? mapController;
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Google Maps Integration'),
      ),
      body: GoogleMap(
        onMapCreated: (controller) {
          setState(() {
            mapController = controller;
          });
        },
        initialCameraPosition: CameraPosition(
          target: LatLng(12.17, 79.04), // Default location (San Francisco)
```

```
          zoom: 12.0,
        ),
      ),   );
  }
}
```

Output:



2. Study and installation of Flutter/Kotlin multi-platform environment.
And
17 Study and installation of Flutter/Kotlin multi-platform environment.

1. Install Required Software:

- Flutter SDK: Download and install the Flutter SDK from the official Flutter website (https://flutter.dev/docs/get-started/install).
- Android Studio or IntelliJ IDEA: Install one of these IDEs as they provide good support for Flutter and Kotlin development.
- JDK (Java Development Kit): Ensure you have JDK installed as Kotlin requires it.

2. Configure Flutter:

- After installing the Flutter SDK, add Flutter's `bin` directory to your system's `PATH` variable.
- Run `flutter doctor` in your terminal to check if there are any dependencies you need to install.
- Install any missing dependencies as instructed.

3. Set up Android Development Environment:

- For Flutter development, you need to set up Android Studio or IntelliJ IDEA with the Flutter and Dart plugins.
- Configure a virtual device for testing Flutter apps or connect a physical device.

5. Create a Multi-Platform Project:

- With your IDE open, create a new Flutter project. This will create the Flutter project structure.
- Inside your Flutter project, you can add Kotlin support. You can do this by adding Kotlin files or modules to your Flutter project.

6. Write and Test Code:

- Write your Flutter UI code in Dart.
- Write your business logic and platform-specific code in Kotlin.

3. Develop a native calculator application.
And
6. Mini Projects involving Flutter/Kotlin multi-platform
And
7. Develop an application that uses Widgets, GUI components, Fonts, and Colors.
And
11 Mini Projects involving Flutter/Kotlin multi-platform
And
12 Develop an application that uses Widgets, GUI components, Fonts, and Colors
And
18 Develop a native calculator application.

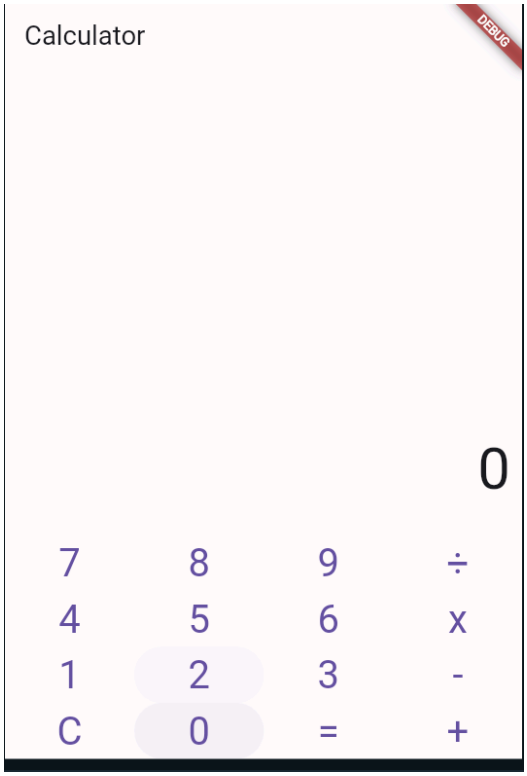
```
import 'package:flutter/material.dart';
```

```
void main() {
  runApp(CalculatorApp());
}


class CalculatorApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CalculatorScreen(),
    );
  }
}


class CalculatorScreen extends StatefulWidget {
  @override
  _CalculatorScreenState createState() => _CalculatorScreenState();
}


class _CalculatorScreenState extends State<CalculatorScreen> {
  String _output = "0";
  String _expression = "";


  void _onPressed(String buttonText) {
    setState(() {
      if (buttonText == "C") {
        _output = "0";
        _expression = "";
      } else if (buttonText == "=") {
        _expression = _expression.replaceAll("x", "*");
        _expression = _expression.replaceAll("÷", "/");
        try {
          _output = (evalExpression(_expression)).toString();
        } catch (e) {
          _output = "Error";
        }
      } else {
        _expression += buttonText;
        _output = _expression;
```

```
    }
  });
}


double evalExpression(String exp) {
  return double.parse(exp);
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Calculator'),
    ),
    body: Column(
      children: <Widget>[
        Expanded(
          child: Container(
            padding: EdgeInsets.symmetric(horizontal: 10, vertical: 20),
            alignment: Alignment.bottomRight,
            child: Text(
              _output,
              style: TextStyle(fontSize: 48),
            ),
          ),
        ),
        Row(
          children: <Widget>[
            buildButton("7"),
            buildButton("8"),
            buildButton("9"),
            buildButton("÷"),
          ],
        ),
        Row(
          children: <Widget>[
            buildButton("4"),
            buildButton("5"),
            buildButton("6"),
            buildButton("x"),
          ],
        ),
```

```
        Row(
          children: <Widget>[
            buildButton("1"),
            buildButton("2"),
            buildButton("3"),
            buildButton("-"),
          ],
        ),
        Row(
          children: <Widget>[
            buildButton("C"),
            buildButton("0"),
            buildButton("="),
            buildButton("+"),
          ],
        ),
      ],
    ),
  );
 }


 Widget buildButton(String buttonText) {
  return Expanded(
    child: TextButton(
      onPressed: () => _onPressed(buttonText),
      child: Text(
        buttonText,
        style: TextStyle(fontSize: 32),
      ),
    ),
  );
 }
}
```

Output:



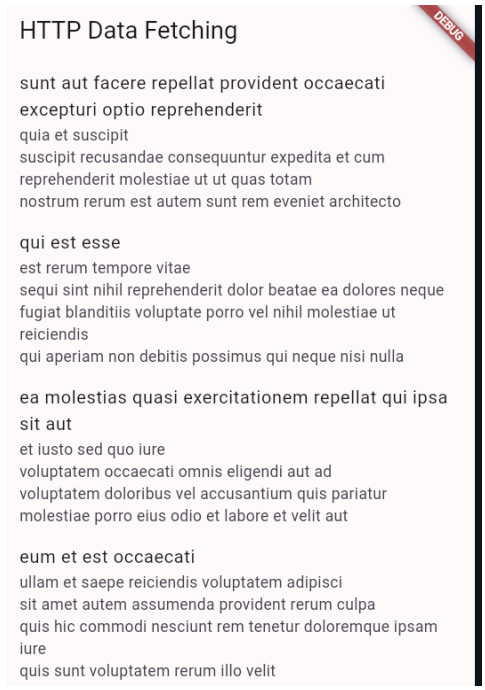
9. Develop an application to connect to a web service and to retrieve data with HTTP
And
14 Develop an application to connect to a web service and to retrieve data with HTTP

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


void main() => runApp(MyApp());


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'HTTP Data Fetching',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomePage(),
    );
  }
```

```
}


class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}


class _HomePageState extends State<HomePage> {
  List<dynamic> _data = [];


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


  Future<void> fetchData() async {
    final response =
        await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));


    if (response.statusCode == 200) {
      setState(() {
        _data = json.decode(response.body);
      });
    } else {
      throw Exception('Failed to load data');
    }
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('HTTP Data Fetching'),
      ),
      body: _data.isEmpty
          ? Center(child: CircularProgressIndicator())
          : ListView.builder(
```

```
            itemCount: _data.length,
            itemBuilder: (context, index) {
              return ListTile(
                title: Text(_data[index]['title']),
                subtitle: Text(_data[index]['body']),
              );
            },
          ),
    );
  }
}
```

Output:



10. Design a web server supporting push notifications.
And
15 Design a web server supporting push notifications.

```
const express = require("express");
const bodyParser = require("body-parser");
const admin = require("firebase-admin");
// Initialize Firebase Admin SDK
const serviceAccount = require("./path/to/serviceAccountKey.json");
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
```

```
});


const app = express();
const port = 3000;


app.use(bodyParser.json());


// Endpoint to handle registration of device tokens
app.post("/register", (req, res) => {
  const { token, userId } = req.body;


  // Store the device token in your database along with the user ID
  // For simplicity, you can use an in-memory object here
  devices[userId] = token;


  res.status(200).send("Device registered successfully");
});


// Endpoint to send push notification
app.post("/send-notification", (req, res) => {
  const { userId, message } = req.body;


  // Retrieve the device token for the specified user ID from the database
  const token = devices[userId];


  if (token) {
    // Send a push notification to the device using Firebase Cloud Messaging (FCM)
    const payload = {
      notification: {
        title: "New Notification",
        body: message,
      },
    };


    admin
```

```
      .messaging()
      .sendToDevice(token, payload)
      .then((response) => {
        console.log("Notification sent successfully:", response);
        res.status(200).send("Notification sent successfully");
      })
      .catch((error) => {
        console.error("Error sending notification:", error);
        res.status(500).send("Error sending notification");
      });
  } else {
    res.status(404).send("Device token not found for user");
  }
});


// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Output: