EX.NO: 1 GUI COMPONENTS, FONTS AND COLORS

AIM:

To implement an application that uses GUI components, Font, Colors.

PROCEDURE:

GUI components:

- Scaffold()
  - Creates a visual scaffold for Material Design widgets
  - appBar() id used to specify the title and background of the top bar.
  - body() is used to contain the primary content of the scaffold.
- MaterialApp()
  - contains widgets that are used for the material design of an application.
  - theme property is used to set the theme of the application to dark or light.
  - Home property defines the starting point of the application. It usually contains Scaffold.
- Text():
  - import 'package:*flutter*/material.dart';
  - specify the string to be displayed, withing quotes inside Text().
  - Style property can be used to add TextStyle like fontSize, color.
  - textAlign property can be used for alignment of specified text
- GridView.count()
  - creates a layout with a fixed number of tiles in the cross axis
  - children property is used to specify the widgets to be included in the layout. (Eg: containers)
  - To set spacing between items along main axis or cross axis, set the required double values for mainAxisSpacing property and crossAxisSpacing property respectively
- Container()
  - Helps to create a rectangular visual element.
  - The margin property uses EdgeInsets to set the margin for the four directions (LTRB).
  - Image or icon or text can be included placed inside the container using child parameter:
  - Decoration (BoxDecoration) can be used to give shape, backgroundColor etc. to a container.

Font:

- Style property can be used to add TextStyle like fontSize, color.
- To use google fonts,
  - Install using ‘flutter pub add google_fonts’
  - import 'package:google_fonts/google_fonts.dart';
  - Specify the font name in the style property of Text().
  - textStyle attribute can be used to format the text.
  - style: GoogleFonts.*rockSalt*(textStyle: const TextStyle(color: Colors.*black*,fontSize: 20)

Colors:

- Color property can be used to specify the color using the Colors class.
- It can also be represented in the format of #RRGGBB where RR represents Red color, GG represents the Green color and BB represents the Blue color.

CODE:

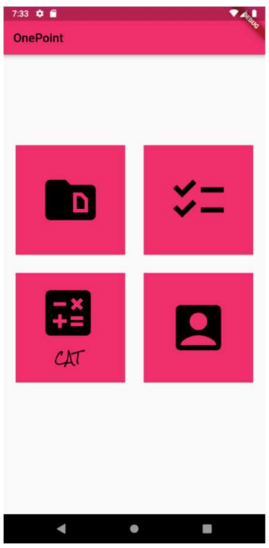
```
import 'package:flutter/material.dart';
import 'package:onep_v2/account/my_account.dart';
import './todo/todo_list.dart';
import './cat/cat_marks.dart';
import './account/my_account.dart';
import './upload_files/sample.dart';
import 'package:google_fonts/google_fonts.dart';
//firebase
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';



Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}


class MyApp extends StatelessWidget {
  const MyApp({super.key});
  static const String _title = 'OnePoint';

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
       title: _title,
       // home: MoodyGradient());
       home : Scaffold(
              appBar: AppBar(title: const Text(_title,style: TextStyle(color: Colors.black)),
backgroundColor: Color(0xffef2e6c),),
              body: const Center(
              child: MyStatefulWidget(),
              ),
    ),
    );
  }
}

//------------------------------------------------------------------------------------

class MyStatefulWidget extends StatefulWidget {
```

```
  const MyStatefulWidget({super.key});

  @override
  State<MyStatefulWidget> createState() => _MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  @override
  Widget build(BuildContext context) {
    return GridView.count(
      primary: false,
      // padding: const EdgeInsets.all(20),
      padding: const EdgeInsets.fromLTRB(20, 150, 20, 30),
      crossAxisSpacing: 30,
      mainAxisSpacing: 30,
      crossAxisCount: 2,
      children: <Widget>[
        Container(
          padding: const EdgeInsets.all(8),
          color: Color(0xffef2e6c),
          child: Ink(
            decoration: const ShapeDecoration(
              color: Colors.lightBlue,
              shape: CircleBorder(),
            ),
            child: IconButton(
              icon: const Icon(
                Icons.snippet_folder,
                color: Colors.black,
                size: 100.0,
              ),
              color: Colors.white,
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => FourthRoute()),
                );
              },
            ),
          ),
        ),
        Container(
          padding: const EdgeInsets.all(8),
          color: Color(0xffef2e6c),
          child: Ink(
            decoration: const ShapeDecoration(
              color: Colors.lightBlue,
              shape: CircleBorder(),
            ),
            child: IconButton(
```

```
            icon: const Icon(
              Icons.checklist,
              color: Colors.black,
              size: 100.0,
            ),
            color: Colors.white,
            onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SecondRoute()),
            );
            }),
      ),
    ),
    Container(
      padding: const EdgeInsets.all(8),
      color: Color(0xffef2e6c),
      child: Ink(
        decoration: const ShapeDecoration(
          color: Colors.lightBlue,
          shape: CircleBorder(),
        ),
        child:  TextButton.icon(
          onPressed: () => {},
          icon: Column(
            children: [
              Icon(
                Icons.calculate,
                color: Colors.black,
                size: 100,
              ),
              Text(
                'CAT',
                style: GoogleFonts.rockSalt(textStyle: const TextStyle(color:
Colors.black,fontSize: 20)),
              ),
            ],
          ),
          label: Text(
            '', //'Label',
            style: TextStyle(
              color: Colors.black,
            ),
          ),
        ),
      ),

    ),
    Container(
      padding: const EdgeInsets.all(8),
```

```
          color: Color(0xffef2e6c),
          child: Ink(
            decoration: const ShapeDecoration(
              color: Colors.lightBlue,
              shape: CircleBorder(),
            ),
            child: IconButton(
              icon: const Icon(
                Icons.account_box_rounded,
                color: Colors.black,
                size: 100.0,
              ),
              color: Colors.white,
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => FifthRoute()),
                );
              },
            ),
          ),
        ),
      ],
    );
  }
}
```

OUTPUT:



RESULT:

Thus, GUI components, Font and Colors have been implemented using Flutter.

EX.NO:2 LAYOUT MANAGERS AND EVENT LISTENERS

AIM:

To implement an application that uses layout managers and event listeners.

PROCEDURE:

- Layout managers:
  - Column() class is used to display its children in a vertical way.
  - Children property is used to specify its descendants.
  - ListTile is a fixed-height row that typically contains some text as well as leading or trailing icon.
  - The icons (or other widgets) for the tile are defined with the leading and trailing parameters.
- Event listeners:
  - onPressed() property is used to assign a callback function to the button or icon.
  - The application executes this function whenever the user presses taps the chip.
  - If onPressed() is null, then it denotes disabled.

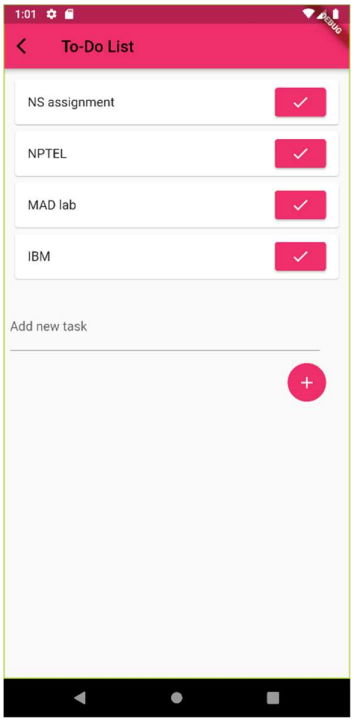
CODE:

```
import 'package:flutter/material.dart';
class SecondRoute extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<SecondRoute> {
  // List<Person> persons = [];
  List<Task> tasks = [];

  final taskController = TextEditingController();

  @override
  void dispose() {
    // Clean up the controller when the widget is disposed.
    taskController.dispose();
    super.dispose();
  }

  @override
  void initState() {
    tasks.add(Task(tname: "NS assignment"));
    tasks.add(Task(tname: "NPTEL"));
    tasks.add(Task(tname: "MAD lab"));
    tasks.add(Task(tname: "IBM"));
    super.initState();
  }
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
    appBar: AppBar(
     leading: GestureDetector(
       child: Icon( Icons.arrow_back_ios, color: Colors.black,  ),
       onTap: () {
         Navigator.pop(context);
       } ,
     ) ,
     title: Text("To-Do List",style:TextStyle(color:Colors.black)),
     backgroundColor: Color(0xffef2e6c),
    ),
    body: Column(children: [
     SingleChildScrollView(
       child: Container(
         padding: EdgeInsets.all(10),
         child: Column(
           children: tasks.map((taskone) {
             return Container(
               child: Card(
                 child: ListTile(
                   title: Text(taskone.tname),
                   trailing: ElevatedButton(
                     style: ElevatedButton.styleFrom(
                        primary: Color(0xffef2e6c)),
                     child: Icon(Icons.check),
                     onPressed: () {
                       tasks.removeWhere((element) {
                         return element.tname == taskone.tname;
                       });
                       setState(() {
                         //refresh UI after deleting element from list
                       });
                     },
                   ),
                 ),
               ),
             );
           }).toList(),
         ),
       ),
     ),
     Center(
       child: Container(
          margin: const EdgeInsets.only(right: 30.0),
          child: Column(
           children: [
             Padding(
```

```
                padding: const EdgeInsets.symmetric(
                    horizontal: 8, vertical: 16),
                child: TextFormField(
                  controller: taskController,
                  decoration: const InputDecoration(
                    border: UnderlineInputBorder(),
                    labelText: 'Add new task',
                  ),
                ),
              ),
              Align(
                alignment: Alignment.bottomRight,
                child: Ink(
                  decoration: const ShapeDecoration(
                    color: Color(0xffef2e6c),
                    shape: CircleBorder(),
                  ),
                  child: IconButton(
                    icon: const Icon(Icons.add),
                    color: Colors.white,
                    // onPressed: () { debugPrint(taskController.text); },
                    onPressed: () {
                      debugPrint(taskController.text);
                      tasks.add(Task(tname: taskController.text));
                      taskController.text = "";
                      setState(() {
                        //refresh UI after deleting element from list
                      });
                    },
                  ),
                ),
              ),
            ],
          )),
      )
    ]));
  }
}

class Task {
  String tname = "";
  Task({required this.tname});
}
```

OUTPUT:



RESULT:

Thus, an application that uses layout managers and event listeners has been implemented using Flutter.

EX.NO: 3 **SIMPLE CALCULATOR**

AIM:

To develop a naive calculator application.

PROCEDURE:

- Initialize num1, num2 and res (result) as 0
- Declare a function for each of the basic arithmetic operations (+ , - , * , /) which takes two operands as parameters and returns the result.
- Use the TextField, to get num1 and num2 as input.
- TextEditingController is used to retrieve the values of the TextField(s).
- Use another non-editable TextField to display the result.
- Use MaterialButton to perform the labelled arithmetic operation.

CODE:

main.dart

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

import 'homePage.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Simple Calculator',
      debugShowCheckedModeBanner: false,
      theme: ThemeData.light(),
      home: HomePage(),
    );
  }
}
```

homepage.dart

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

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

class _HomePageState extends State<HomePage> {
  double num1 = 0, num2 = 0, res = 0;


  TextEditingController t1 = TextEditingController(text: '');
```

```
TextEditingController t2 = TextEditingController(text: '');

doAddition() {
  setState(() {
    num2 = double.parse(t2.text);
    num1 = double.parse(t1.text);
    res = num1 + num2;
  });
}

doSub() {
  setState(() {
    num2 = double.parse(t2.text);
    num1 = double.parse(t1.text);
    res = num1 - num2;
  });
}

doMul() {
  setState(() {
    num2 = double.parse(t2.text);
    num1 = double.parse(t1.text);
    res = num1 * num2;
  });
}

doDiv() {
  setState(() {

    num2 = double.parse(t2.text);
    num1 = double.parse(t1.text);
    res = (num1 / num2);
  });
}

doClear() {
  setState(() {
    t1 = TextEditingController(text: '');
    t2 = TextEditingController(text: '');
    res = 0;
  });
}

doDecimal() {
  setState(() {
    //TODO:.....................
  });
}

@override
```

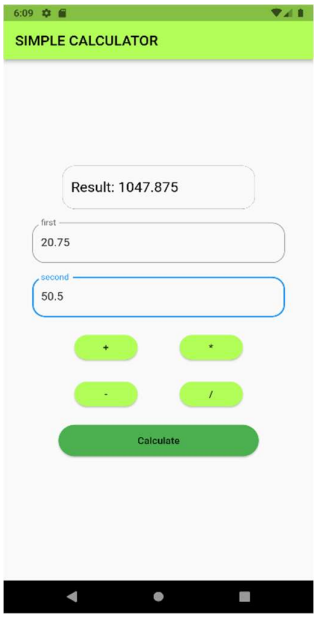
```
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: Text('SIMPLE CALCULATOR',style: TextStyle(fontSize: 20.0, color:
Colors.black)),
        backgroundColor: Colors.lightGreenAccent,
      ),
      body: Container(
        padding: EdgeInsets.only(bottom:40.0, top: 15.0, left: 40.0, right: 40.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.symmetric(horizontal:42.0),
              child: TextField(
                keyboardType: TextInputType.number,
                cursorColor: Colors.indigo,
                enabled: false,
                //enableInteractiveSelection: false,
                decoration: InputDecoration(
                  fillColor: Colors.white,
                  hintText: 'Result: $res', hintStyle: TextStyle(fontSize: 20.0, color: Colors.black),
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(18.0),
                  ),
                ),
              ),
            ),
            Padding(
              padding: EdgeInsets.only(top: 20.0),
            ),
            //The Text field for the First number
            TextField(
              keyboardType: TextInputType.number,
              cursorColor: Colors.tealAccent,
              controller: t1,
              decoration: InputDecoration(
                labelText: 'first',
                fillColor: Colors.white,
                hintText: 'Enter your First number',
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(18.0),
                ),
              ),
            ),
            Padding(
              padding: EdgeInsets.only(top: 20.0),
            ),
            //The Text field for the second number
```

```
TextField(
  keyboardType: TextInputType.number,
  cursorColor: Colors.tealAccent,
  controller: t2,
  decoration: InputDecoration(
    labelText: 'second',
    fillColor: Colors.white,
    hintText: 'Enter your Second number',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(18.0),
    ),
  ),
),

Padding(
  padding: EdgeInsets.only(top: 20.0),
),

Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    MaterialButton(
      child: Text('+'),
      shape: StadiumBorder(),
      color: Colors.lightGreenAccent,
      onPressed: () {
        //TODO:
        doAddition();
      },
    ),
    MaterialButton(
      child: Text('*'),
      shape: StadiumBorder(),
      color: Colors.lightGreenAccent,
      onPressed: () {
        //TODO:
        doMul();
      },
    ),
  ],
),

Padding(
  padding: EdgeInsets.only(top: 20.0),
),

Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    MaterialButton(
```

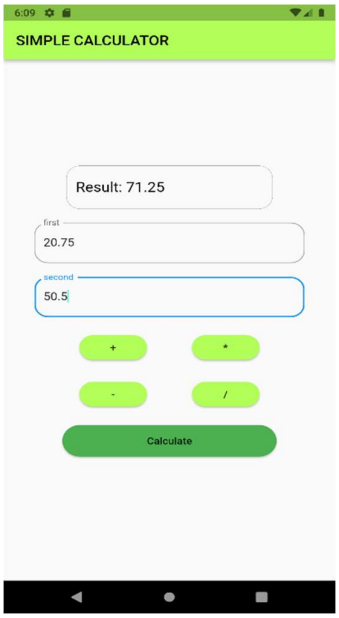
```
              child: Text('-'),
              color: Colors.lightGreenAccent,
              shape: StadiumBorder(),
              onPressed: () {
                //TODO:
                doSub();
              },
            ),
            MaterialButton(
              child: Text('/'),
              shape: StadiumBorder(),
              color: Colors.lightGreenAccent,
              onPressed: () {
                //TODO:
                doDiv();
              },
            ),
          ],
        ),
        Padding(
          padding: EdgeInsets.only(top: 20.0),
        ),

        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            MaterialButton(
              child: Text('Calculate', textAlign: TextAlign.center,),
              color: Colors.green,
              shape: StadiumBorder(),
              padding: EdgeInsets.only(left: 110.0, right: 110.0, top: 15.0, bottom: 15.0),
              onPressed: () {
                //TODO:
                doClear();
              },
            ),
          ],
        ),
      ],
    ),
  ),
 );
 }
}
```

OUTPUT:

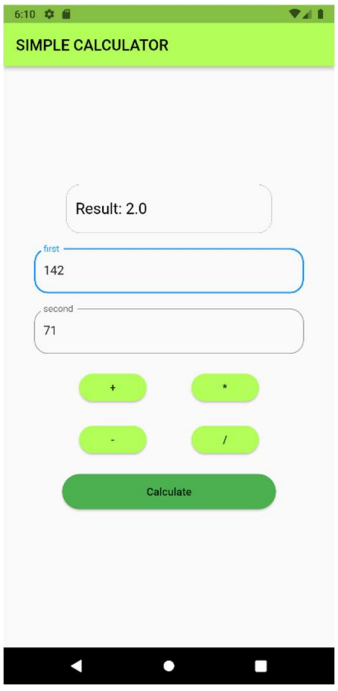
Multiplication



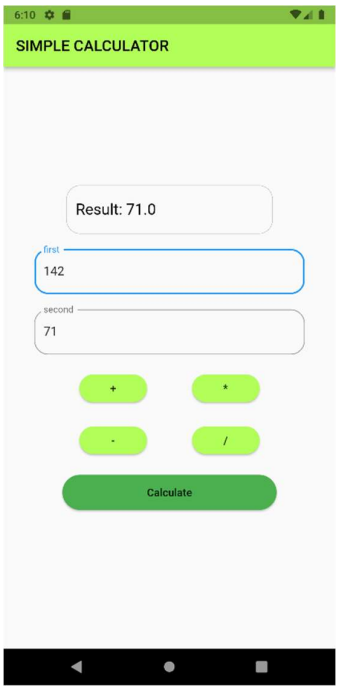
Addition



Division



Subtraction



RESULT:

Thus, a simple naive calculator application is developed using Flutter.

EX.NO:4 BASIC GRAPHICAL PRIMITIVES

AIM:

To write an application that draws basic graphical primitives on the screen.

PROCEDURE:

- Declare a class for each graphical primitive.
- The CustomPainter class is used.
- The paint method takes canvas and size as parameters.
- Create an instance of Paint() class.
- canvas.drawRect() is used to draw a rectangle.
- Similarly, for line drawLine() is used.
- For circle and arc, drawCircle() and drawArc() are used respectively.
- Inside the scaffold, the required class is called by specifying it as the painter of CustomPaint class.

CODE:

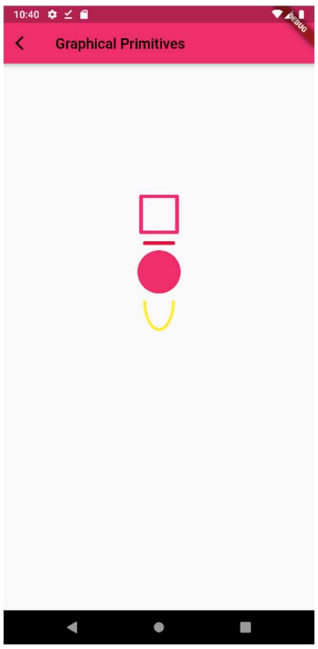
```
import 'package:flutter/material.dart';

class Rectangle extends CustomPainter {
  bool? isFilled;
  Rectangle({this.isFilled});
  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = Paint();
    paint.color = Color(0xffef2e6c);
    if(isFilled != null){
      paint.style = PaintingStyle.fill;
    }
    else {
      paint.style = PaintingStyle.stroke;
    }
    paint.strokeCap = StrokeCap.round;
    paint.strokeJoin = StrokeJoin.round;
    paint.strokeWidth = 5;
    Offset offset = Offset(size.width * 0.5, size.height);

    Rect rect = Rect.fromCenter(center: offset, width: 50, height: 50);
    canvas.drawRect(rect, paint);
  }

  @override
  bool shouldRepaint(covariant Rectangle oldDelegate) {
    return false;
  }
}

class Line extends CustomPainter {
```

```
  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = Paint();
    paint.color = const Color.fromARGB(255, 226, 19, 64);
    paint.strokeWidth = 5;
    paint.strokeCap = StrokeCap.round;

    Offset startingOffset = Offset(0, size.height);
    Offset endingOffset = Offset(size.width, size.height);

    canvas.drawLine(startingOffset, endingOffset, paint);
  }

  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) {
    return false;
  }
}

class Circle extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = Paint();
    paint.color = Color(0xffef2e6c);
    paint.style = PaintingStyle.fill;
    paint.strokeCap = StrokeCap.round;
    paint.strokeJoin = StrokeJoin.round;

    Offset offset = Offset(size.width * 0.5, size.height);
    canvas.drawCircle(offset, 30, paint);
  }

  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) {
    return false;
  }
}

class Arc extends CustomPainter {

  double _degreeToRadians(num degree) {
    return (degree * 3.14) / 180.0;
  }

  @override
  void paint(Canvas canvas, Size size) {
    Rect rect = Rect.fromLTRB(0, 0, size.width, size.height * 2);
    double startAngle = _degreeToRadians(0);
    double sweepAngle = _degreeToRadians(180);
    const useCenter = false;
```

```
    Paint paint = Paint();
    paint.color = Colors.yellow;
    paint.style = PaintingStyle.stroke;
    paint.strokeWidth = 4;
    canvas.drawArc(rect, startAngle, sweepAngle, useCenter, paint);
  }

  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) {
    return false;
  }
}


class ShapesPage extends StatefulWidget {
  @override
  _ShapesPageState createState() => _ShapesPageState();
}

class _ShapesPageState extends State<ShapesPage> {

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: GestureDetector(
          child: Icon( Icons.arrow_back_ios, color: Colors.black,  ),
          onTap: () {
            Navigator.pop(context);
          } ,
        ),
        title: Text('Graphical Primitives', style: TextStyle(color: Colors.black)),
        backgroundColor: Color(0xffef2e6c),
      ),
      body: Container(
          padding: EdgeInsets.fromLTRB(5,30.0,5,30.0),
          child: Center(
              child: Column(
          children: [
            CustomPaint(
```

```
            size: Size(160,180),
            painter: Rectangle(),
          ),
          CustomPaint(
            size: Size(40.0, 40.0),
            painter: Line(),
          ),
          CustomPaint(
            size: Size(40.0, 40.0),
            painter: Circle(),
          ),
          CustomPaint(
            size: Size(40.0, 40.0),
            painter: Arc(),
          ),

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

OUTPUT:



RESULT:

Hence, an application that draws basic graphical primitives on the screen has been implemented using Flutter.

EX.NO:5 DATABASE CONNECTION

AIM:

To develop an application that makes use of database.

PROCEDURE:

- Install the following packages:
  - npm install firebase-tools
  - flutter pub add firebase_core
  - flutter pub add firebase_auth
- Use ‘firebase login’ command to login to google account
- Use ‘flutterfire configure’ to add a firebase project to the application.
- Import the generated ‘firebase options’ file to main.dart file.
- FirebaseAuth.*instance*.currentUser is used to get the current user object
- Use FilePicker to select files from the device.
- storage.ref().child() is used to store the chosen file to Firebase storage.

CODE:

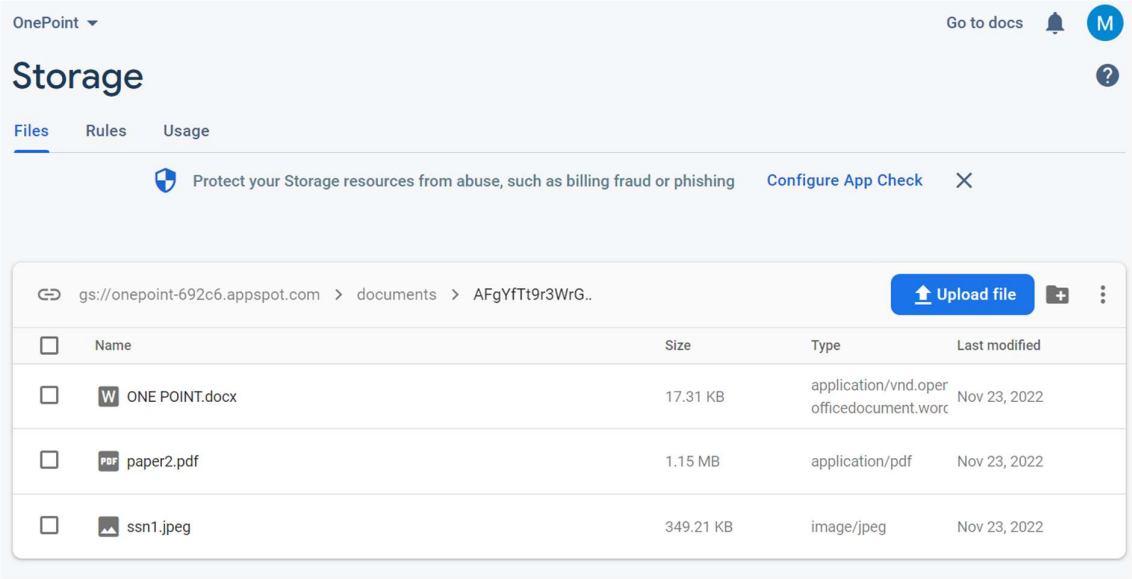
```
import 'package:flutter/material.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:image_picker/image_picker.dart';
import 'package:path/path.dart' as path;
import 'package:flutter/foundation.dart';
import 'dart:io';
import 'package:file_picker/file_picker.dart';
import 'package:get/get.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';

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

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

class _FourthRoute extends State<FourthRoute> {
  FirebaseStorage storage = FirebaseStorage.instance;


  Future<void> _upload() async {
    FilePickerResult? result = await FilePicker.platform.pickFiles();
    final User? auth = FirebaseAuth.instance.currentUser;
    final uid = auth?.uid;

    if (result != null) {
      PlatformFile file = result.files.first;
      debugPrint(file.name);
```

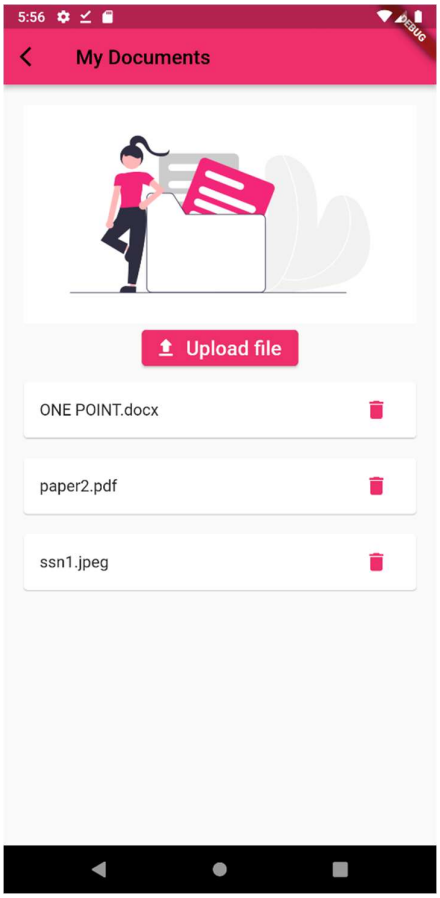
```
      debugPrint(file.bytes.toString());
      debugPrint(file.size.toString());
      debugPrint(file.extension);
      debugPrint(file.path);
      // storage.ref(file.name).putFile(File(file.path.toString()),
      //     SettableMetadata(customMetadata: {'uploaded_by': uid.toString()}));
      storage.ref().child("documents/" + uid! + "/" +
file.name).putFile(File(file.path.toString()));

    }
  }

  Future<List<Map<String, dynamic>>> _list() async{
    // List userFiles = [];
    List<Map<String, dynamic>> userFiles = [];
    final User? auth = FirebaseAuth.instance.currentUser;
    final uid = auth?.uid;

    final storageRef = FirebaseStorage.instance.ref().child("documents/" + uid! + "/");
    final listResult = await storageRef.listAll();
    for (var item in listResult.items) {
      debugPrint(item.name);
      userFiles.add({
        "fName": item.name,
       "fPath" : item.fullPath});
    }
    return userFiles;
  }


  // Delete the selected image
  // This function is called when a trash icon is pressed
  Future<void> _delete(String ref) async {
    await storage.ref(ref).delete();
    // Rebuild the UI
    setState(() {});
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text("My Documents", style: TextStyle(color: Colors.black)),
        leading: GestureDetector(
          child: Icon( Icons.arrow_back_ios, color: Colors.black,  ),
          onTap: () {
            Navigator.pop(context);
          } ,
        ),
          backgroundColor: Color(0xffef2e6c),
```

```
),
body: Padding(
  padding: const EdgeInsets.all(20),
  child: Column(
    children: [
      Image.asset('assets/images/undraw_my_files_swob.png'),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: [
          ElevatedButton.icon(
              onPressed: () => _upload(),
              icon: const Icon(Icons.file_upload_rounded),
              label: const Text('Upload file',style:TextStyle(fontSize: 20)),
              style: ElevatedButton.styleFrom(backgroundColor:Color(0xffef2e6c) )),
          // ElevatedButton.icon(
          //     onPressed: () => _list(),
          //     icon: const Icon(Icons.view_list_rounded),
          //     label: const Text('View files'),
          //     style: ElevatedButton.styleFrom(backgroundColor:Color(0xffef2e6c) )),
        ],
      ),
      Expanded(
        child: FutureBuilder(
          future: _list(),
          builder: (context,
              AsyncSnapshot<List<Map<String, dynamic>>> snapshot) {
            if (snapshot.connectionState == ConnectionState.done) {
              return ListView.builder(
                itemCount: snapshot.data?.length ?? 0,
                itemBuilder: (context, index) {
                  final Map<String, dynamic> uFile=
                      snapshot.data![index];

                  return Card(
                    margin: const EdgeInsets.symmetric(vertical: 10),
                    child: ListTile(
                      dense: false,
                      // leading: fname,
                      title: Text(uFile['fName']),
                      //subtitle: Text(image['uploaded_by']),
                      trailing: IconButton(
                        onPressed: () => _delete(uFile['fPath']),
                        icon: const Icon(
                          Icons.delete,
                          color: Color(0xffef2e6c),
                        ),
                      ),
                    ),
                  );
                },
```

```
                );
              }

              return const Center(
                child: CircularProgressIndicator(),
              );
            },
          ),
        ),
      ],
    ),
  ),
 );
 }
}
```

OUTPUT:





RESULT:

Thus, an application that makes use of Firebase Storage (Database) for storing, retrieving and deleting files of each user has been implemented using Flutter.

EX.NO:6 **RSS FEED**

AIM:

To develop an application that makes use of RSS Feed.

PROCEDURE:

- *Import packages.*
  import 'package:webfeed/webfeed.dart';
  import 'package:http/http.dart' as http;
  import 'package:url_launcher/url_launcher.dart';
- *Define RSS Feed URL (FEED_URL)*
- *Create a variable to hold our RSS feed data. (_feed)*
- *Create a place holder for our title (_title)*
- *Create a method to navigate to the selected RSS item* (openFeed)
- Use RssFeed.parse(response.body)to *grab the RSS data from the provided URL.*
- *Create the UI for the ListView and plug in the retrieved RSS data*

CODE:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_rss_reader/colors.dart';
import 'package:webfeed/webfeed.dart';
import 'package:http/http.dart' as http;
import 'package:url_launcher/url_launcher.dart';

class RSSReader extends StatefulWidget {
  RSSReader() : super();.
  final String title = 'Jobs Feed';

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

class RSSReaderState extends State<RSSReader> {

  static const String loadingMessage = 'Loading Feed...';
  static const String feedLoadErrorMessage = 'Error Loading Feed.';
  static const String feedOpenErrorMessage = 'Error Opening Feed.';

  GlobalKey<RefreshIndicatorState> _refreshKey;

  updateTitle(title) {
    setState(() {
      _title = title;
    });
  }


  updateFeed(feed) {
```

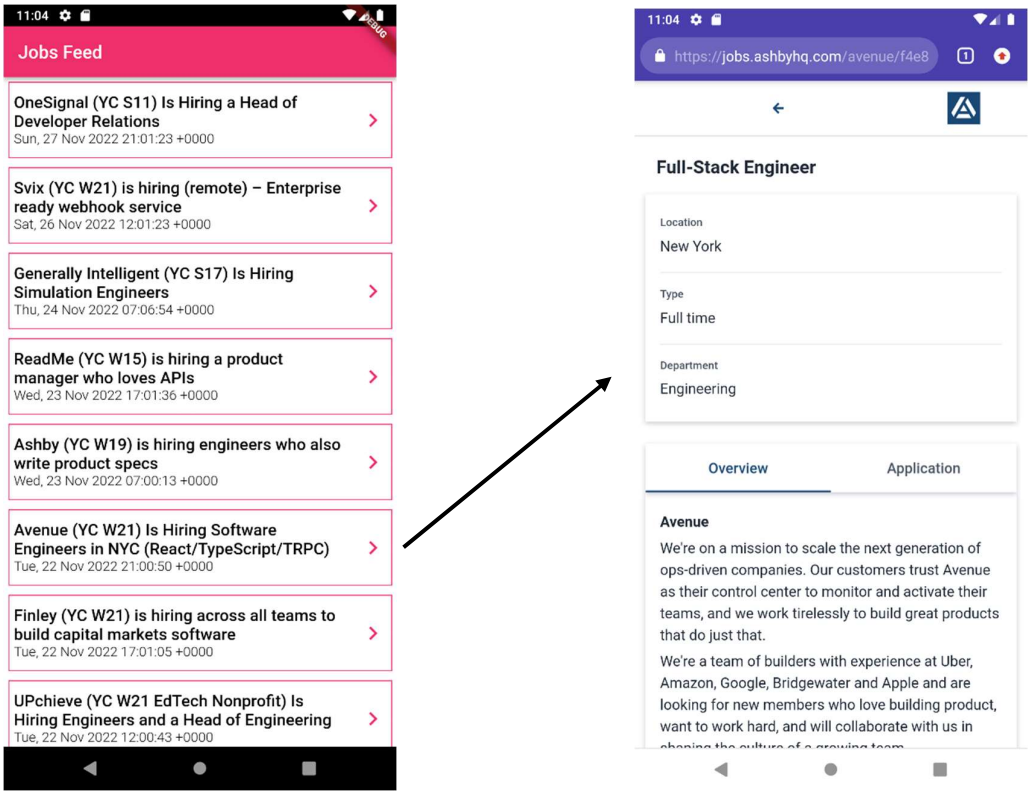
```
  setState(() {
    _feed = feed;
  });
 }
 Future<void> openFeed(String url) async {
  if (await canLaunch(url)) {
    await launch(
      url,
      forceSafariVC: true,
      forceWebView: false,
    );
    return;
  }
  updateTitle(feedOpenErrorMessage);
 }

 load() async {
  updateTitle(loadingMessage);
  loadFeed().then((result) {
    if (null == result || result.toString().isEmpty) {
      // Notify user of error.
      updateTitle(feedLoadErrorMessage);
      return;
    }
    // If there is no error, load the RSS data into the _feed object.
    updateFeed(result);
    // Reset the title.
    updateTitle("Jobs Feed");
  });
 }

 Future<RssFeed> loadFeed() async {
  try {
    final client = http.Client();
    final response = await client.get(FEED_URL);
    return RssFeed.parse(response.body);
  } catch (e) {
    // handle any exceptions here
  }
  return null;
 }

 @override
 void initState() {
  super.initState();
  _refreshKey = GlobalKey<RefreshIndicatorState>();
  updateTitle(widget.title);
  load();
 }
```

```
isFeedEmpty() {
  return null == _feed || null == _feed.items;
}

body() {
  return isFeedEmpty()
      ? Center(
          child: CircularProgressIndicator(),
        )
      : RefreshIndicator(
          key: _refreshKey,
          child: list(),
          onRefresh: () => load(),
        );
}
@override
Widget build(BuildContext context) {
  return SafeArea(
    child: Scaffold(
      backgroundColor: colorHackerBackground,
      appBar: AppBar(
        backgroundColor: Color(0xffef2e6c),
        title: Text(_title),
      ),
      body: body(),
    ),
  );
}
list() {
  return Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: <Widget>[
        Expanded(
          flex: 3,
          child: Container(
            child: ListView.builder(
              padding: EdgeInsets.all(5.0),
              itemCount: _feed.items.length,
              itemBuilder: (BuildContext context, int index) {
                final item = _feed.items[index];
                return Container(
                  margin: EdgeInsets.only(
                    bottom: 10.0,
                  ),
                  decoration: customBoxDecoration(),
                  child: ListTile(
                    title: title(item.title),
                    subtitle: subtitle(item.pubDate),
                    trailing: rightIcon(),
```

```
                contentPadding: EdgeInsets.all(5.0),
                onTap: () => openFeed(item.link),
              ),
            );
          },
        ),
      ),
    ),
  ]);
}

// Method that returns the Text Widget for the title of our RSS data.
title(title) {
  return Text(
    title,
    style: TextStyle(
        fontSize: 18.0,
        fontWeight: FontWeight.w500,
        color: colorHackerHeading),
    maxLines: 2,
    overflow: TextOverflow.ellipsis,
  );
}

// Method that returns the Text Widget for the subtitle of our RSS data.
subtitle(subTitle) {
  return Text(
    subTitle,
    style: TextStyle(
        fontSize: 15.0,
        fontWeight: FontWeight.w300,
        color: colorHackerHeading),
    maxLines: 1,
    overflow: TextOverflow.ellipsis,
  );
}

// Method that returns Icon Widget.
rightIcon() {
  return Icon(
    Icons.keyboard_arrow_right,
    color: colorHackerBorder,
    size: 30.0,
  );
}

// Custom box decoration for the Container Widgets.
BoxDecoration customBoxDecoration() {
  return BoxDecoration(
    border: Border.all(
```

```
        color: colorHackerBorder, // border color
        width: 1.0,
      ),
    );
  }

}
```

OUTPUT:



RESULT:

Thus, an application that uses RSS feed has been developed using Flutter.

EX.NO:7 **MULTI-THREADING**

AIM:

To write an application that implements multi-threading.

PROCEDURE:

- Install the following packages:
  - npm install firebase-tools
  - flutter pub add firebase_core
  - flutter pub add firebase_auth
- Use 'firebase login' command to login to google account
- Use 'flutterfire configure' to add a firebase project to the application.
- Import the generated 'firebase options' file to main.dart file.
- FirebaseAuth.*instance*.currentUser is used to get the current user object
- Use FilePicker to select files from the device.
- storage.ref().child() is used to store the chosen file to Firebase storage.
- 'async' enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished.
- 'await' keyword is used before a call to a function that returns a promise. This makes the code wait at that point until the promise is settled, at which point the fulfilled value of the promise is treated as a return value, or the rejected value is thrown.

CODE:

main.dart

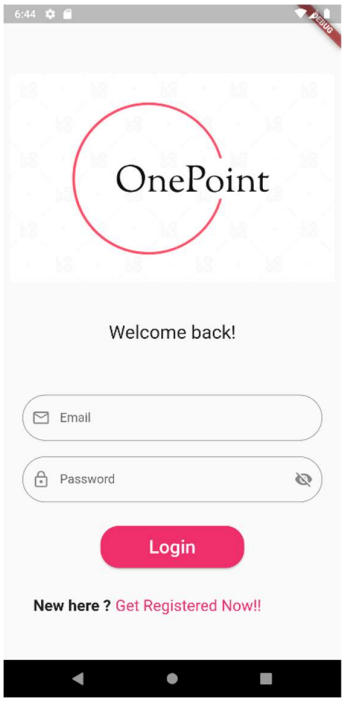
```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'login.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter auth Demo',
      home: Login(),
    );
  }
}
```

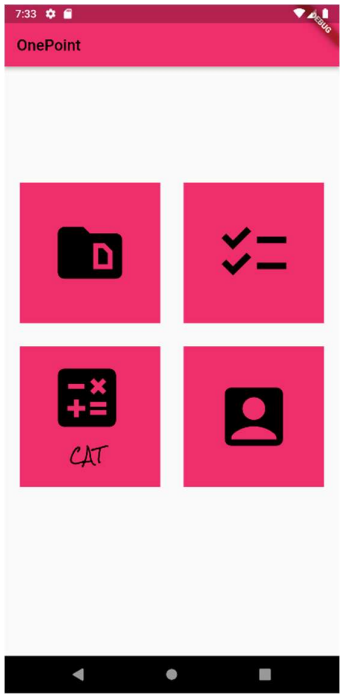
authentication.dart

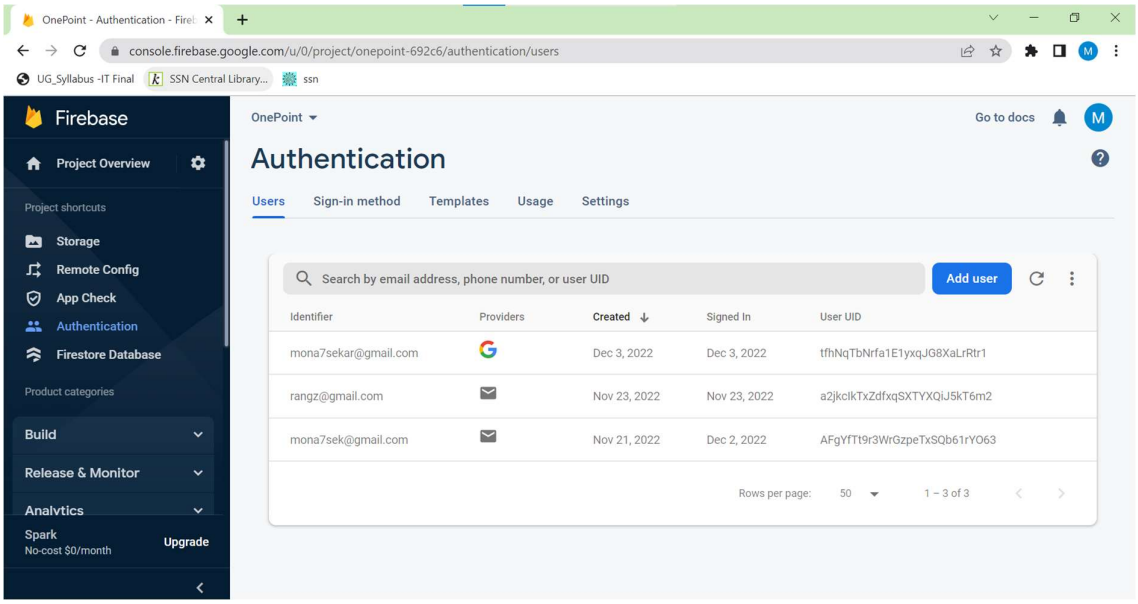
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

class AuthenticationHelper {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  get user => _auth.currentUser;

  Future<String?> signInWithGoogle() async {
    // Trigger the authentication flow
    final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();

    // Obtain the auth details from the request
    final GoogleSignInAuthentication? googleAuth = await googleUser?.authentication;

    // Create a new credential
    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth?.accessToken,
      idToken: googleAuth?.idToken,
    );

    // Once signed in, return the UserCredential
    await FirebaseAuth.instance.signInWithCredential(credential);
    return null;
  }


//SIGN UP METHOD
  Future<String?> signUp({required String email, required String password}) async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

  //SIGN IN METHODJ
  Future<String?> signIn({required String email, required String password}) async {
    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }
```

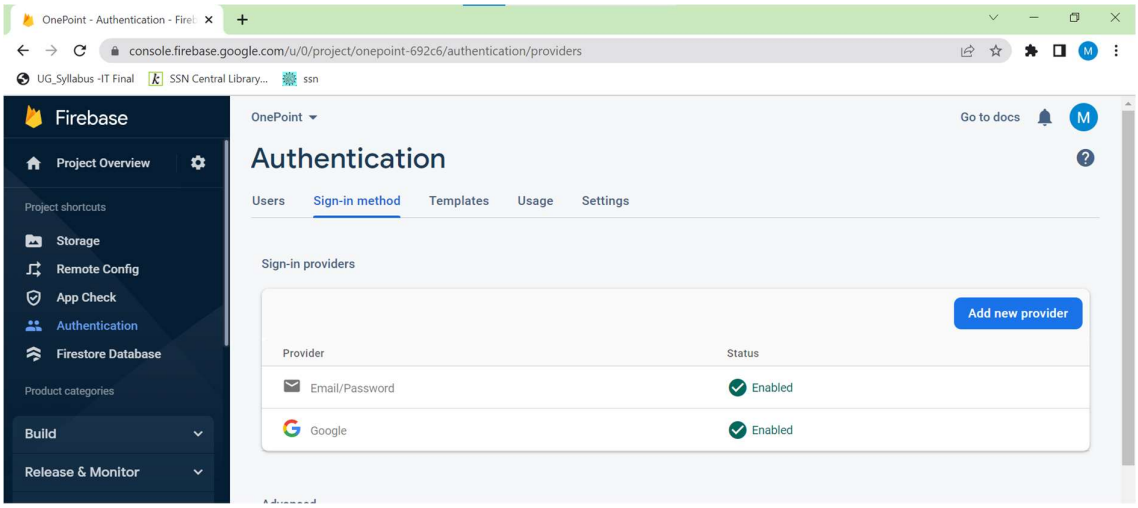
```
  //SIGN OUT METHOD
  Future<void> signOut() async {
    await _auth.signOut();

    print('signout');
  }
}
```

OUTPUT:









RESULT:

Thus, an application that implements multithreading is implemented using Flutter and Firebase.

EX.NO:8 GPS LOCATION INFORMATION

AIM:

To develop a native application that uses GPS location information.

PROCEDURE:

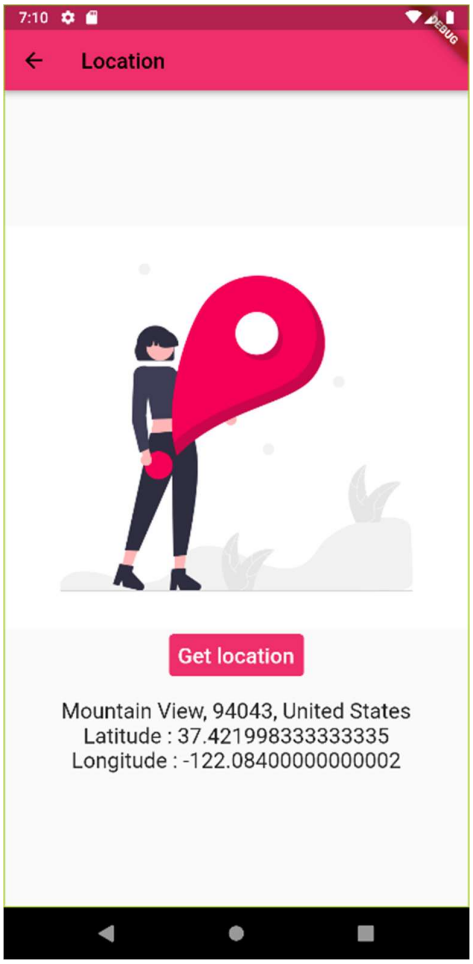
- Install the following packages: geolocator & geocoding
- Import them using,
    - import 'package:geocoding/geocoding.dart';
    - import 'package:geolocator/geolocator.dart';
- Get current location of the device, by creating an instance of Geolocator and calling getCurrentPosition.
- Convert latitude and longitude values into address using placemarkFromCoordinates().

CODE:

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

class LocationPage extends StatefulWidget {
  @override
  _LocationPageState createState() => _LocationPageState();
}

class _LocationPageState extends State<LocationPage> {
  Position? _currentPosition;
  String _currentAddress = '';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: Colors.black, //change your color here
        ),
        backgroundColor: Color(0xffef2e6c),
        title: Text("Location",style:TextStyle(color:Colors.black)),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image.asset('assets/images/undraw_Current_location_re_j130.png'),
            TextButton(
              style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xffef2e6c))),
              child: Text("Get location",style:TextStyle(fontSize: 20,color:Colors.white)),
              onPressed: () {
```

```
            _getCurrentLocation();
          },
        ),
        Divider(color:Colors.transparent,thickness: 150),
        if (_currentAddress != null) Text(
          _currentAddress,style: TextStyle(fontSize: 20),
        ),
        if (_currentPosition != null) Text( 'Latitude : ' +
          _currentPosition!.latitude.toString(),style: TextStyle(fontSize: 20),
        ),
        if (_currentPosition != null) Text( 'Longitude : ' +
            _currentPosition!.longitude.toString(),style: TextStyle(fontSize: 20),
        ),
       ],
      ),
    ),
   );
  }

  _getCurrentLocation() {
   Geolocator
       .getCurrentPosition(desiredAccuracy: LocationAccuracy.best,
forceAndroidLocationManager: true)
       .then((Position position) {
     setState(() {
       _currentPosition = position;
       _getAddressFromLatLng();
     });
   }).catchError((e) {
     print(e);
   });
  }

  _getAddressFromLatLng() async {
   try {
     List<Placemark> placemarks = await placemarkFromCoordinates(
         _currentPosition!.latitude,
         _currentPosition!.longitude
     );

     Placemark place = placemarks[0];

     setState(() {
       _currentAddress = "${place.locality}, ${place.postalCode}, ${place.country}";
     });
   } catch (e) {
     print(e);
   }
  }
}
```

OUTPUT:



RESULT:

Thus, a native application that uses GPS location information has been developed.

EX.NO:9 **WRITING TO SD CARD**

AIM:

To implement an application that writes to SD card.

PROCEDURE:

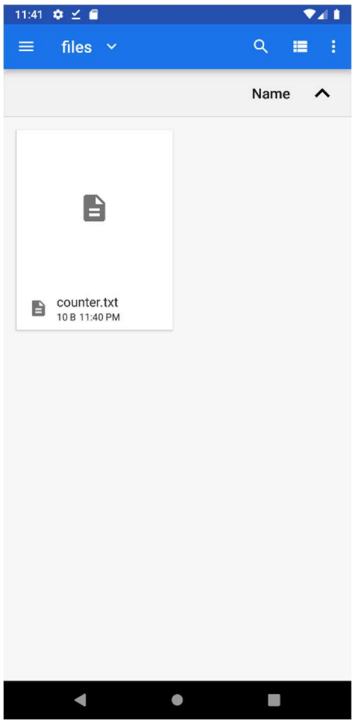
- Install path_provider package
- The path where is file is to be written is obtained using getExternalStorageDirectory() function.
- writeAsString(<String>) is used to write contents into a text file.
- readAsString() is used to read the contents of the file.

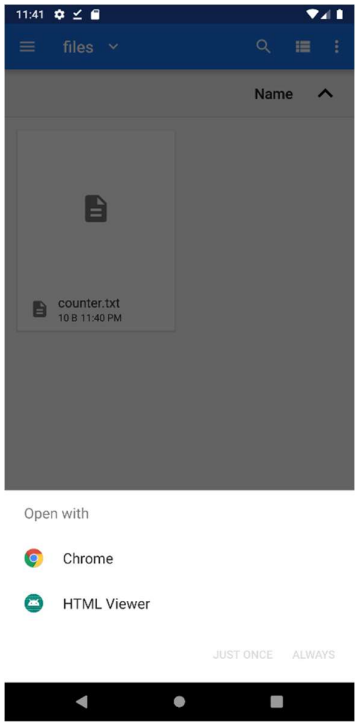
CODE:

```
import 'dart:async';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';


class SDcard extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<SDcard> {
  String data='';

  Future<String> get _localPath async {
    final directory = await getExternalStorageDirectory();
    print(directory?.path);
    return directory!.path;
  }

  Future<File> get _localFile async {
    final path = await _localPath;
    return File('$path/counter.txt');
  }

  Future<String> readContent() async {
    try {
      final file = await _localFile;
      // Read the file
      String contents = await file.readAsString();
      // Returning the contents of the file
      return contents;
    } catch (e) {
      // If encountering an error, return
      return 'Error!';
    }
```

```
  }

  Future<File> writeContent() async {
    final file = await _localFile;
    // Write the file
    return file.writeAsString('One Point App');
  }

  @override
  void initState() {
    super.initState();
    writeContent();
    readContent().then((String value) {
      setState(() {
        data = value;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Write to SD Card", style: TextStyle(color: Colors.black)),
        leading: GestureDetector(
          child: Icon( Icons.arrow_back_ios, color: Colors.black,  ),
          onTap: () {
            Navigator.pop(context);
          } ,
        ),
        backgroundColor: Color(0xffef2e6c),
      ),
      body: Center(
        child: Text(
          'Data read from a file: \n $data',style:TextStyle(fontSize: 40)
        ),
      ),
    );
  }
}
```

OUTPUT:





RESULT:

Hence, an application that writes to SD card has been implemented using Flutter.

EX.NO: 10 ALERT BOX

AIM:

To implement an application that creates an alert upon receiving a message.

PROCEDURE:

- On the To-do list page, create a TextButton labelled ‘ADD’ to add a new task.
- In the onPressed() property, use showDialog to specify the alert box contents.
- AlertDialog() is used to create the alert message box.
    - The content property is used to specify the message using Text(). In this case, the message displayed is “Task added”.
    - The action property is used to specify the buttons in the alert box using TextButton().

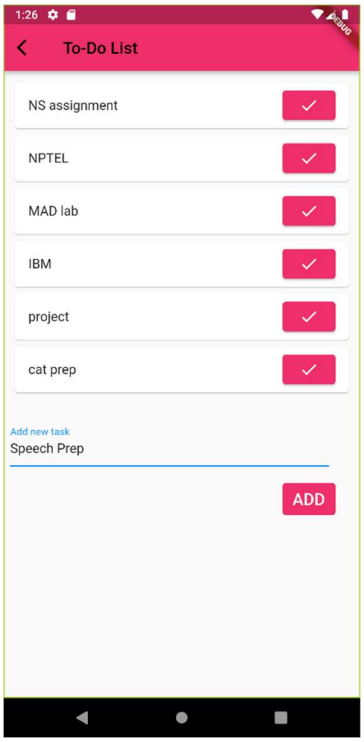
CODE:

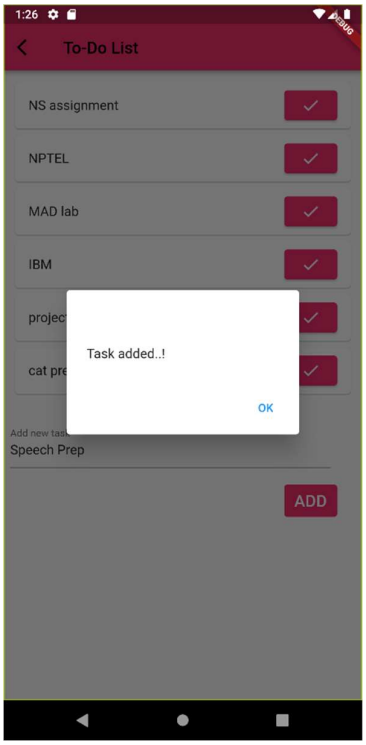
```
import 'package:flutter/material.dart';

class SecondRoute extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<SecondRoute> {

  List<Task> tasks = [];

  final taskController = TextEditingController();

  @override
  void dispose() {
    // Clean up the controller when the widget is disposed.
    taskController.dispose();
    super.dispose();
  }

  @override
  void initState() {
    tasks.add(Task(tname: "NS assignment"));
    tasks.add(Task(tname: "NPTEL"));
    tasks.add(Task(tname: "MAD lab"));
    tasks.add(Task(tname: "IBM"));
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          leading: GestureDetector(
```

```
      child: Icon(
        Icons.arrow_back_ios,
        color: Colors.black,
      ),
      onTap: () {
        Navigator.pop(context);
      },
    ),
    title: Text("To-Do List", style: TextStyle(color: Colors.black)),
    backgroundColor: Color(0xffef2e6c),
  ),
  body: Column(children: [
    SingleChildScrollView(
      child: Container(
        padding: EdgeInsets.all(10),
        child: Column(
          children: tasks.map((taskone) {
            return Container(
              child: Card(
                child: ListTile(
                  title: Text(taskone.tname),
                  trailing: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                        primary: Color(0xffef2e6c)),
                    child: Icon(Icons.check),
                    onPressed: () {
                      tasks.removeWhere((element) {
                        return element.tname == taskone.tname;
                      });
                      setState(() {
                        //refresh UI after deleting element from list
                      });
                    },
                  ),
                ),
              ),
            );
          }).toList(),
        ),
      ),
    ),
    Center(
      child: Container(
          margin: const EdgeInsets.only(right: 30.0),
          child: Column(
            children: [
              Padding(
                padding: const EdgeInsets.symmetric(
                    horizontal: 8, vertical: 16),
                child: TextFormField(
```

```
                    controller: taskController,
                    decoration: const InputDecoration(
                      border: UnderlineInputBorder(),
                      labelText: 'Add new task',
                    ),
                  ),
                ),
                Align(
                    alignment: Alignment.bottomRight,
                    child: TextButton(
                      style: ButtonStyle(backgroundColor:
MaterialStateProperty.all(Color(0xffef2e6c))),
                      onPressed: () => showDialog<String>(
                        context: context,
                        builder: (BuildContext context) => AlertDialog(
                          title: const Text(''),
                          content: const Text('Task added..!'),
                          actions: <Widget>[
                            TextButton(
                              onPressed: () {
                                debugPrint(taskController.text);
                                tasks.add(Task(tname: taskController.text));
                                taskController.text = "";
                                setState(() {});
                                Navigator.pop(context, 'OK');
                              },
                              child: const Text('OK'),
                            ),
                          ],
                        ),
                      ),
                      child: const Text('ADD',style:TextStyle(color:Colors.white,fontSize: 20.0)),
                    )

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

class Task {
  String tname = "";
  Task({required this.tname});
}
```

OUTPUT:





RESULT:

Thus, an application that creates an alert upon receiving a message is implemented using Flutter.

EX.NO:11 ALARM CLOCK

AIM:

To write a mobile application that creates an alarm clock.

PROCEDURE:

- Install the flutter_alarm_clock package using
  - flutter pub add flutter_alarm_clock
- Import it using
  - import 'package:flutter_alarm_clock/flutter_alarm_clock.dart';
- The FlutterAlarmClock.createAlarm() that takes hours and minutes as parameters.
- Hours and minutes are taken as input from user, using TextField().
- On clicking on “Create Alarm” button, a snackbar is displayed which appears when an alarm is set.
- The “Show Alarms” button, opens the clock application of the device which shows the created alarms.

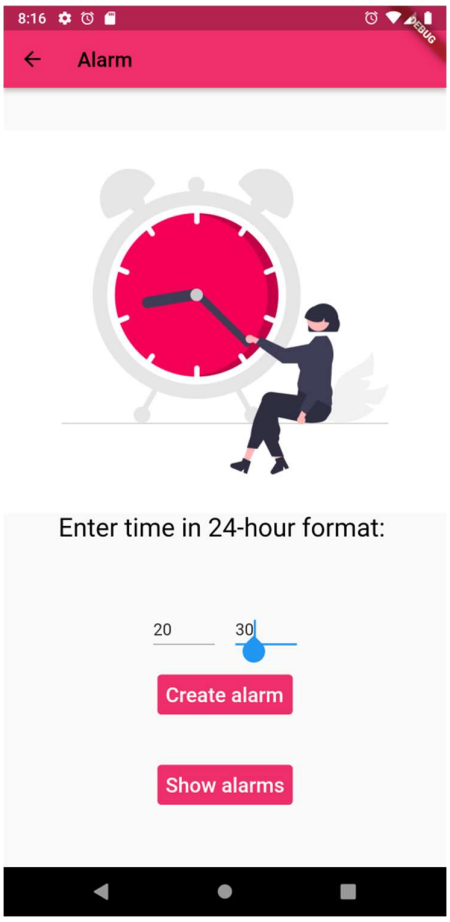
CODE:

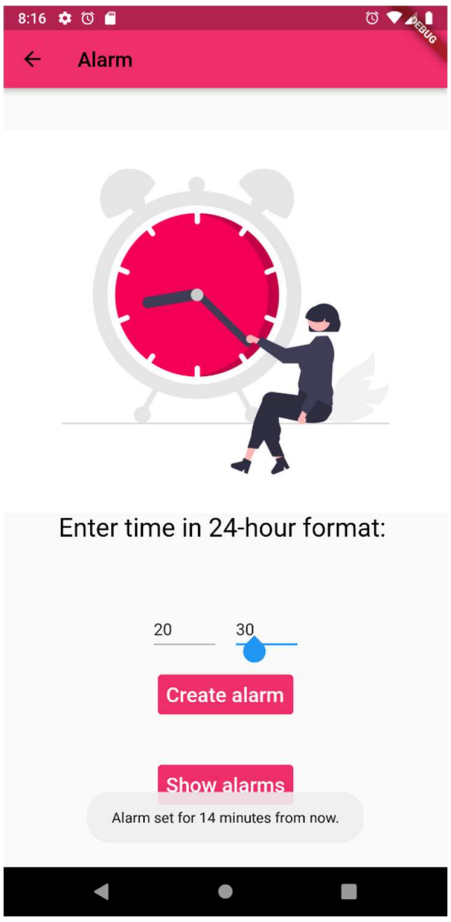
```
import 'package:flutter/material.dart';
import 'package:flutter_alarm_clock/flutter_alarm_clock.dart';


class AlarmPage extends StatefulWidget {
  @override
  _AlarmPageState createState() => _AlarmPageState();
}

class _AlarmPageState extends State<AlarmPage> {
  TextEditingController hourController = TextEditingController();
  TextEditingController minuteController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: Colors.black, //change your color here
        ),
        backgroundColor: Color(0xffef2e6c),
        title: Text("Alarm", style: TextStyle(color: Colors.black)),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image.asset('assets/images/undraw_Time_management_re_tk5w.png'),
            Text('Enter time in 24-hour format: \n',style:TextStyle(fontSize:
25,color:Colors.black)),
```

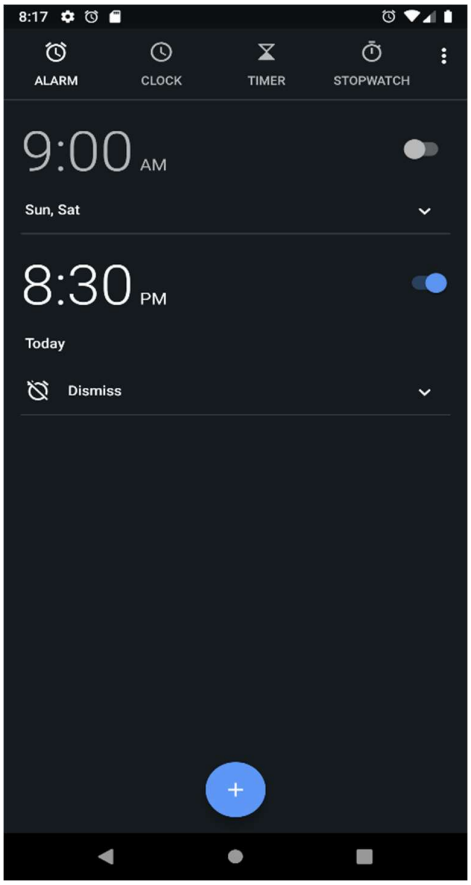
```
          SizedBox(height: 30),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                height: 40,
                width: 60,
                child: Center(
                  child: TextField(
                    controller: hourController,
                    keyboardType: TextInputType.number,
                  ),
                ),
              ),
              SizedBox(width: 20),
              Container(
                height: 40,
                width: 60,
                child: Center(
                  child: TextField(
                    controller: minuteController,
                    keyboardType: TextInputType.number,
                  ),
                ),
              ),
            ],
          ),
          Container(
            margin: const EdgeInsets.all(25),
            child: TextButton(
              style: ButtonStyle(backgroundColor:
MaterialStateProperty.all(Color(0xffef2e6c))),
              child: const Text(
                'Create alarm',
                style: TextStyle(fontSize: 20.0,color:Colors.white),
              ),
              onPressed: () {
                int hour;
                int minutes;
                hour = int.parse(hourController.text);
                minutes = int.parse(minuteController.text);
                FlutterAlarmClock.createAlarm(hour, minutes);
              },
            ),
          ),
          Container(
            margin: const EdgeInsets.all(15),
            child: TextButton(
              style: ButtonStyle(backgroundColor:
MaterialStateProperty.all(Color(0xffef2e6c))),
```

```
            child: const Text(
              'Show alarms',
              style: TextStyle(fontSize: 20.0,color:Colors.white),
            ),
            onPressed: () {
              FlutterAlarmClock.showAlarms();
            },
          ),
        ),
      ],
    ),
  ),
 );
 }
}
```

OUTPUT:







RESULT:

Thus, a mobile application that creates an alarm clock is implemented using Flutter.

EX.NO:12 SIMPLE GAME WITH MULTIMEDIA SUPPORT

AIM:

To implement a simple gaming application with multimedia support.

PROCEDURE:

- Create a class TileModel for each tile, which has the following as members
  - ImageAssetPath
  - IsSelected
- Create a list called 'pairs' which contains a pair of each tile of a specific image.
- Use GridView to display the tiles as a 4x4 grid.
- Initialize points as 0 using setState().
- For every matched tile, increment points by 100.
- Play until points == 800.
- Click on replay to restart the game.

CODE:

data.dart

```
import 'package:memory_game/models/TileModel.dart';

String selectedTile = "";
int selectedIndex ;
bool selected = true;
int points = 0;

List<TileModel> myPairs = new List<TileModel>();
List<bool> clicked = new List<bool>();

List<bool> getClicked(){

  List<bool> yoClicked = new List<bool>();
  List<TileModel> myairs = new List<TileModel>();
  myairs = getPairs();
  for(int i=0;i<myairs.length;i++){
    yoClicked[i] = false;
  }

  return yoClicked;
}

List<TileModel>  getPairs(){

  List<TileModel> pairs = new List<TileModel>();

  TileModel tileModel = new TileModel();

  //1
  tileModel.setImageAssetPath("assets/fox.png");
```

```
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//2
tileModel.setImageAssetPath("assets/hippo.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//3
tileModel.setImageAssetPath("assets/horse.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//4
tileModel.setImageAssetPath("assets/monkey.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();
//5
tileModel.setImageAssetPath("assets/panda.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//6
tileModel.setImageAssetPath("assets/parrot.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//7
tileModel.setImageAssetPath("assets/rabbit.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
pairs.add(tileModel);
tileModel = new TileModel();

//8
tileModel.setImageAssetPath("assets/zoo.png");
tileModel.setIsSelected(false);
pairs.add(tileModel);
```

```
  pairs.add(tileModel);
  tileModel = new TileModel();

  return pairs;
}

List<TileModel>  getQuestionPairs(){

  List<TileModel> pairs = new List<TileModel>();

  TileModel tileModel = new TileModel();

  //1
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  //2
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  //3
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  //4
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();
  //5
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  //6
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
```

```
  pairs.add(tileModel);
  tileModel = new TileModel();

  //7
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  //8
  tileModel.setImageAssetPath("assets/question.png");
  tileModel.setIsSelected(false);
  pairs.add(tileModel);
  pairs.add(tileModel);
  tileModel = new TileModel();

  return pairs;
}
```

TileModel.dart

```
class TileModel{

  String imageAssetPath;
  bool isSelected;

  TileModel({this.imageAssetPath, this.isSelected});

  void setImageAssetPath(String getImageAssetPath){
    imageAssetPath = getImageAssetPath;
  }

  String getImageAssetPath(){
    return imageAssetPath;
  }

  void setIsSelected(bool getIsSelected){
    isSelected = getIsSelected;
  }

  bool getIsSelected(){
    return isSelected;
  }
}
```

main.dart

```
import 'dart:async';
```

```
import 'package:flutter/material.dart';
import 'package:memory_game/data/data.dart';
import 'package:memory_game/models/TileModel.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: 'Card Memory Game',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        // primaryColor: Color(0xffef2e6c),
        primarySwatch: Colors.red,
      ),
      home: Home(),
    );
  }
}

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

class _HomeState extends State<Home> {
  List<TileModel> gridViewTiles = new List<TileModel>();
  List<TileModel> questionPairs = new List<TileModel>();

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

    myPairs = getPairs();
    myPairs.shuffle();

    gridViewTiles = myPairs;
    Future.delayed(const Duration(seconds: 5), () {
// Here you can write your code
      setState(() {
        print("2 seconds done");
        // Here you can write your code for open new view
        questionPairs = getQuestionPairs();
        gridViewTiles = questionPairs;
```

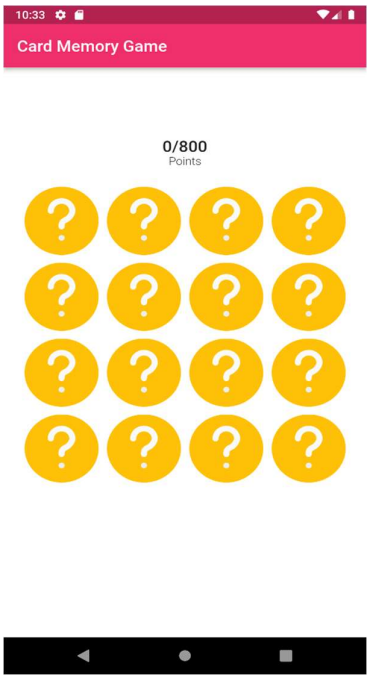
```
      selected = false;
    });
  });
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Card Memory Game'),
      backgroundColor:Color(0xffef2e6c) ,
    ),
    backgroundColor: Colors.white,
    body: SingleChildScrollView(
      child: Container(
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 50),
        child: Column(
          children: <Widget>[
            SizedBox(
              height: 40,
            ),
            points != 800 ? Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: <Widget>[
                Text(
                  "$points/800",
                  style: TextStyle(
                      fontSize: 20, fontWeight: FontWeight.w500),
                ),
                Text(
                  "Points",
                  textAlign: TextAlign.start,
                  style: TextStyle(
                      fontSize: 14, fontWeight: FontWeight.w300),
                ),
              ],
            ) : Container(),
            SizedBox(
              height: 20,
            ),
            points != 800 ? GridView(
              shrinkWrap: true,
              //physics: ClampingScrollPhysics(),
              scrollDirection: Axis.vertical,
              gridDelegate: SliverGridDelegateWithMaxCrossAxisExtent(
                  mainAxisSpacing: 0.0, maxCrossAxisExtent: 100.0),
              children: List.generate(gridViewTiles.length, (index) {
                return Tile(
                  imagePathUrl: gridViewTiles[index].getImageAssetPath(),
```

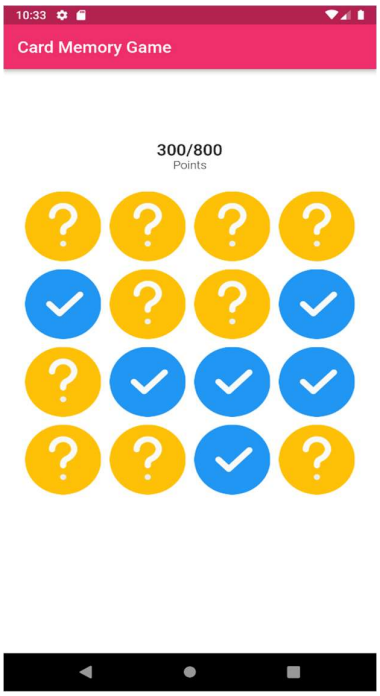
```
                tileIndex: index,
                parent: this,
              );
            }),
          ) : Container(
            child: Column(
              children: <Widget>[
                GestureDetector(
                  onTap: (){
                    setState(() {
                      points = 0;
                      reStart();
                    });
                  },
                  child: Container(
                    height: 50,
                    width: 200,
                    alignment: Alignment.center,
                    decoration: BoxDecoration(
                      color: Color(0xffef2e6c),
                      borderRadius: BorderRadius.circular(24),
                    ),
                    child: Text("Replay", style: TextStyle(
                        color: Colors.white,
                        fontSize: 17,
                        fontWeight: FontWeight.w500
                    ),),
                  ),
                ),
                SizedBox(height: 20,),
              ],
            )
          )
        ],
      ),
    ),
  ),
 );
 }
}



class Tile extends StatefulWidget {
  String imagePathUrl;
  int tileIndex;
  _HomeState parent;

  Tile({this.imagePathUrl, this.tileIndex, this.parent});
```

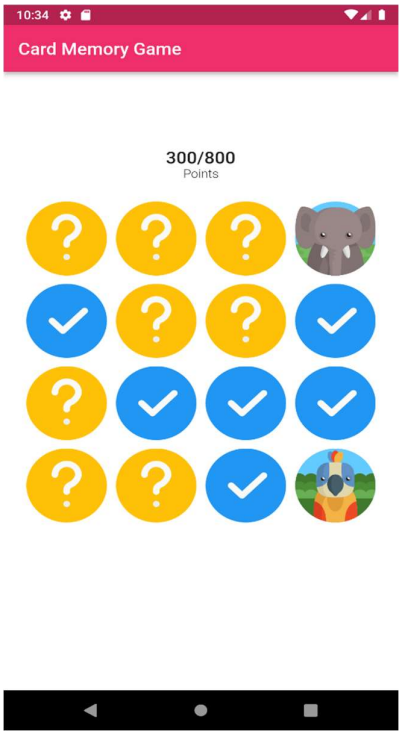
```
  @override
  _TileState createState() => _TileState();
}

class _TileState extends State<Tile> {
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        if (!selected) {
          setState(() {
            myPairs[widget.tileIndex].setIsSelected(true);
          });
          if (selectedTile != "") {
            /// testing if the selected tiles are same
            if (selectedTile == myPairs[widget.tileIndex].getImageAssetPath()) {
              print("add point");
              points = points + 100;
              print(selectedTile + " thishis" + widget.imagePathUrl);

              TileModel tileModel = new TileModel();
              print(widget.tileIndex);
              selected = true;
              Future.delayed(const Duration(seconds: 2), () {
                tileModel.setImageAssetPath("");
                myPairs[widget.tileIndex] = tileModel;
                print(selectedIndex);
                myPairs[selectedIndex] = tileModel;
                this.widget.parent.setState(() {});
                setState(() {
                  selected = false;
                });
                selectedTile = "";
              });
            } else {
              print(selectedTile +
                  " thishis " +
                  myPairs[widget.tileIndex].getImageAssetPath());
              print("wrong choice");
              print(widget.tileIndex);
              print(selectedIndex);
              selected = true;
              Future.delayed(const Duration(seconds: 2), () {
                this.widget.parent.setState(() {
                  myPairs[widget.tileIndex].setIsSelected(false);
                  myPairs[selectedIndex].setIsSelected(false);
                });
                setState(() {
                  selected = false;
                });
```

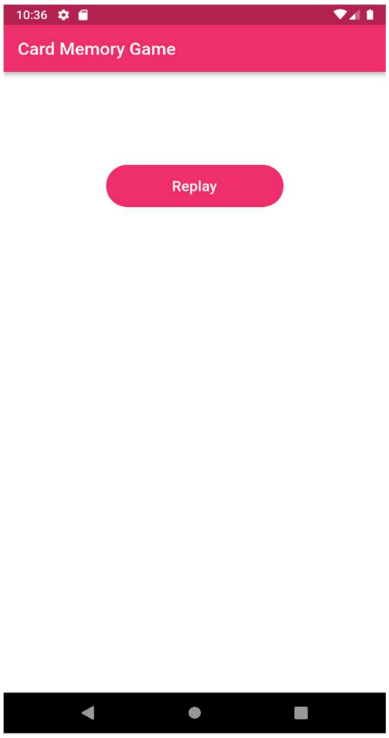
```
            });

            selectedTile = "";
           }
          } else {
           setState(() {
            selectedTile = myPairs[widget.tileIndex].getImageAssetPath();
            selectedIndex = widget.tileIndex;
           });

           print(selectedTile);
           print(selectedIndex);
          }
         }
        },
        child: Container(
         margin: EdgeInsets.all(5),
         child: myPairs[widget.tileIndex].getImageAssetPath() != ""
            ? Image.asset(myPairs[widget.tileIndex].getIsSelected()
               ? myPairs[widget.tileIndex].getImageAssetPath()
               : widget.imagePathUrl)
            : Container(
               color: Colors.white,
               child: Image.asset("assets/correct.png"),
              ),
        ),
       );
      }
    }
```

OUTPUT:









RESULT:

Thus, a simple gaming application that supports multimedia is implemented using Flutter.

EX.NO:13 CONNECTIVITY VIA SOAP OR REST

AIM:

To a mobile application for data handling and connectivity via SOAP or REST to backend services potentially hosted in a cloud environment.

PROCEDURE:

- Import,
  - http.dart
  - dart:convert
- Specify the URL of the API within "Uri.*parse*(<>)"
- http.get() is used to fetch url contents.

CODE:

quotes.dart

```
// To parse this JSON data, do
//
//    final quotes = quotesFromJson(jsonString);

import 'dart:convert';

Quotes quotesFromJson(String str) => Quotes.fromJson(json.decode(str));

String quotesToJson(Quotes data) => json.encode(data.toJson());

class Quotes {
  Quotes({
    this.id,
    this.tags,
    this.content = '',
    this.author = '',
    this.authorSlug,
    this.length,
    this.dateAdded,
    this.dateModified,
  });

  String? id;
  List<String>? tags;
  String content;
  String author;
  String? authorSlug;
  int? length;
  DateTime? dateAdded;
  DateTime? dateModified;

  factory Quotes.fromJson(Map<String, dynamic> json) => Quotes(
```

```
    id: json["_id"],
    tags: List<String>.from(json["tags"].map((x) => x)),
    content: json["content"],
    author: json["author"],
    authorSlug: json["authorSlug"],
    length: json["length"],
    dateAdded: DateTime.parse(json["dateAded"]),
    dateModified: DateTime.parse(json["dateModified"]),
  );

  Map<String, dynamic> toJson() => {
    "_id": id,
    "tags": List<dynamic>.from(tags!.map((x) => x)),
    "content": content,
    "author": author,
    "authorSlug": authorSlug,
    "length": length,
    "dateAdded":
    "${dateAdded!.year.toString().padLeft(4, '0')}-${dateAdded!.month.toString().padLeft(2,
'0')}-${dateAdded!.day.toString().padLeft(2, '0')}",
    "dateModified":
    "${dateModified!.year.toString().padLeft(4, '0')}-
${dateModified!.month.toString().padLeft(2, '0')}-${dateModified!.day.toString().padLeft(2,
'0')}",
  };
}
```

api.dart

```
import 'dart:convert';

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

class Api {
  static Future<Quotes?> getQuotes() async {
    Uri url = Uri.parse('http://api.quotable.io/random');
    http.Response response = await http.get(url);

    if (response.statusCode == 200) {
      print("success");
      return Quotes.fromJson(jsonDecode(response.body));
    } else {
      print("error in getting data");
    }
  }
}
```

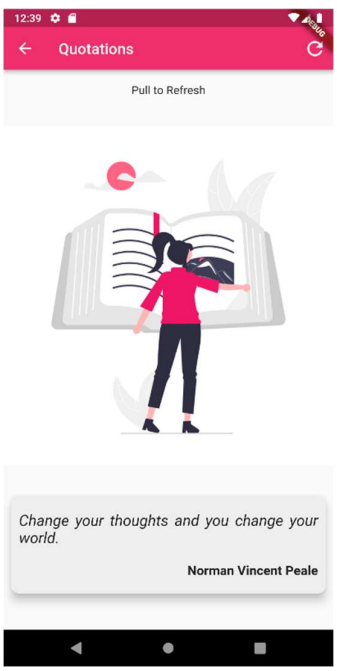
quotes_page.dart

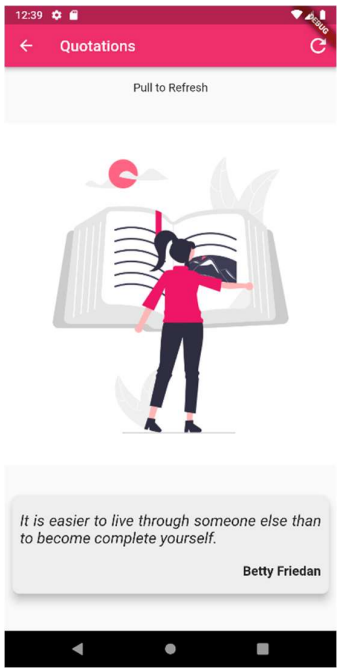
```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'quotes.dart';
import 'api.dart';

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

  @override
  State<QuotesScreen> createState() => _QuotesScreenState();
}

class _QuotesScreenState extends State<QuotesScreen> {
  var size, height, width;
  Quotes? data;
  @override
  Widget build(BuildContext context) {
    size = MediaQuery.of(context).size;
    height = size.height;
    width = size.width;
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color(0xffef2e6c),
        title: Text("Quotations"),
        actions: [
          IconButton(
            icon: Icon(
              Icons.refresh_outlined,
            ),
            iconSize: 30,
            onPressed: () {
              print("icon refresh");
              getQuotes();
            },
          ),
        ],
      ),
      body: RefreshIndicator(
        onRefresh: getQuotes,
        child: ListView(
          children: [
            Padding(
              padding: const EdgeInsets.all(18.0),
              child: Text(
                "Pull to Refresh",
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 15,
```

```
            ),
          ),
        ),
        SizedBox(height: 20),
        Image.asset('assets/images/undraw_Bibliophile_re_xarc.png'),
        SizedBox(height: 20),
        Container(
          padding: EdgeInsets.symmetric(
            horizontal: 10,
          ),
          width: width / 2,
          child: Card(
            margin: EdgeInsets.only(top: 20),
            color: Color(0XFFeeeeee),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(10.0),
            ),
            elevation: 10,
            child: Padding(
              padding: EdgeInsets.symmetric(horizontal: 10, vertical: 20),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    '${data?.content ?? "Don't talk about what you have done or what you are
going to do."}',
                    textAlign: TextAlign.justify,
                    style: TextStyle(
                      fontSize: 20,
                      fontStyle: FontStyle.italic,
                    ),
                  ),
                  SizedBox(height: 22),
                  Align(
                      alignment: Alignment.bottomRight,
                      child: Text(
                        data?.author ?? "Thomas Jefferson",
                        textAlign: TextAlign.justify,
                        style: TextStyle(
                          fontSize: 17,
                          fontWeight: FontWeight.bold,
                        ),
                      ))
                ],
              ),
            ),
          ),
        )
      ],
    ),
```

```
      ),
    );
  }

  Future<Null> getQuotes() async {
    data = await Api.getQuotes();
    setState(() {});
  }
}
```

OUTPUT:





RESULT:

Hence, a mobile application for data handling and connectivity via SOAP or REST to backend services potentially hosted in a cloud environment.

EX.NO:14 GEO-POSITIONING, ACCELEROMETER AND RICH GESTURE BASED UI

AIM:

To write a mobile application that will take advantage of underlying phone functionality including GEO positioning, accelerometer, and rich gesture-based UI handling.

PROCEDURE:

Geo-positioning:

- Install the following packages: geolocator & geocoding
- Import them using,
    - import 'package:geocoding/geocoding.dart';
    - import 'package:geolocator/geolocator.dart';
- Get current location of the device, by creating an instance of Geolocator and calling getCurrentPosition.
- Convert latitude and longitude values into address using placemarkFromCoordinates().

Accelerometer:

- Install the sensors package.
- Import it using, 'import 'package:sensors/sensors.dart';'
- accelerometer readings tell if the device is moving in a particular direction.

Gesture-based UI:

- In the onTap() property of the GestureDetector(), pass the function to be performed.
- In this case, it reverses the boolean value isLightsOn.
- This is used to switch the theme of the screen as dark or light.
- The child property of GestureDetector() is used to specify icon, on clicking which the action is to be performed.

Geo-positioning:

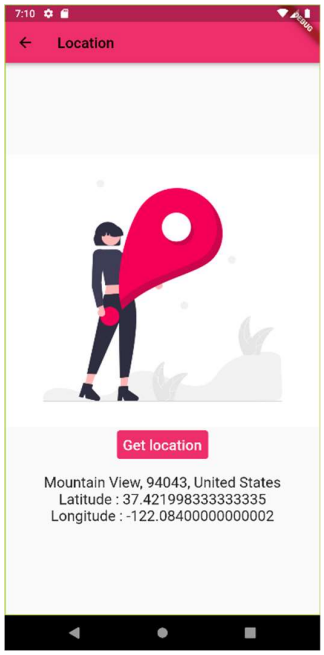
CODE:

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

class LocationPage extends StatefulWidget {
  @override
  _LocationPageState createState() => _LocationPageState();
}

class _LocationPageState extends State<LocationPage> {
  Position? _currentPosition;
  String _currentAddress = '';
```

```
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    iconTheme: IconThemeData(
     color: Colors.black, //change your color here
    ),
    backgroundColor: Color(0xffef2e6c),
    title: Text("Location",style:TextStyle(color:Colors.black)),
   ),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: <Widget>[
      Image.asset('assets/images/undraw_Current_location_re_j130.png'),
      TextButton(
       style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xffef2e6c))),
       child: Text("Get location",style:TextStyle(fontSize: 20,color:Colors.white)),
       onPressed: () {
        _getCurrentLocation();
       },
      ),
      Divider(color:Colors.transparent,thickness: 150),
      if (_currentAddress != null) Text(
       _currentAddress,style: TextStyle(fontSize: 20),
      ),
      if (_currentPosition != null) Text( 'Latitude : ' +
       _currentPosition!.latitude.toString(),style: TextStyle(fontSize: 20),
      ),
      if (_currentPosition != null) Text( 'Longitude : ' +
        _currentPosition!.longitude.toString(),style: TextStyle(fontSize: 20),
      ),
     ],
    ),
   ),
  );
 }

 _getCurrentLocation() {
  Geolocator
    .getCurrentPosition(desiredAccuracy: LocationAccuracy.best,
forceAndroidLocationManager: true)
    .then((Position position) {
   setState(() {
    _currentPosition = position;
    _getAddressFromLatLng();
   });
  }).catchError((e) {
   print(e);
  });
```

```
  }

  _getAddressFromLatLng() async {
    try {
      List<Placemark> placemarks = await placemarkFromCoordinates(
          _currentPosition!.latitude,
          _currentPosition!.longitude
      );

      Placemark place = placemarks[0];

      setState(() {
        _currentAddress = "${place.locality}, ${place.postalCode}, ${place.country}";
      });
    } catch (e) {
      print(e);
    }
  }
}
```

OUTPUT:



Accelerometer:

CODE:

```
import 'dart:async';

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

```
class FocusPage extends StatefulWidget {

  final String title='Focus!';

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

class FocusPageState extends State<FocusPage> {
  // color of the circle
  Color color = Colors.greenAccent;

  // event returned from accelerometer stream
  AccelerometerEvent? event;

  // hold a refernce to these, so that they can be disposed
  Timer? timer;
  StreamSubscription? accel;

  // positions and count
  double top = 125;
  double? left;
  int count = 0;

  // variables for screen size
  double? width;
  double? height;

  setColor(AccelerometerEvent event) {
    // Calculate Left
    double x = ((event.x * 12) + ((width! - 100) / 2));
    // Calculate Top
    double y = event.y * 12 + 125;

    // find the difference from the target position
    var xDiff = x.abs() - ((width! - 100) / 2);
    var yDiff = y.abs() - 125;

    // check if the circle is centered, currently allowing a buffer of 3 to make centering easier
    if (xDiff.abs() < 3 && yDiff.abs() < 3) {
      // set the color and increment count
      setState(() {
        color = Colors.greenAccent;
        count += 1;
      });
    } else {
      // set the color and restart count
      setState(() {
        color = Colors.red;
```

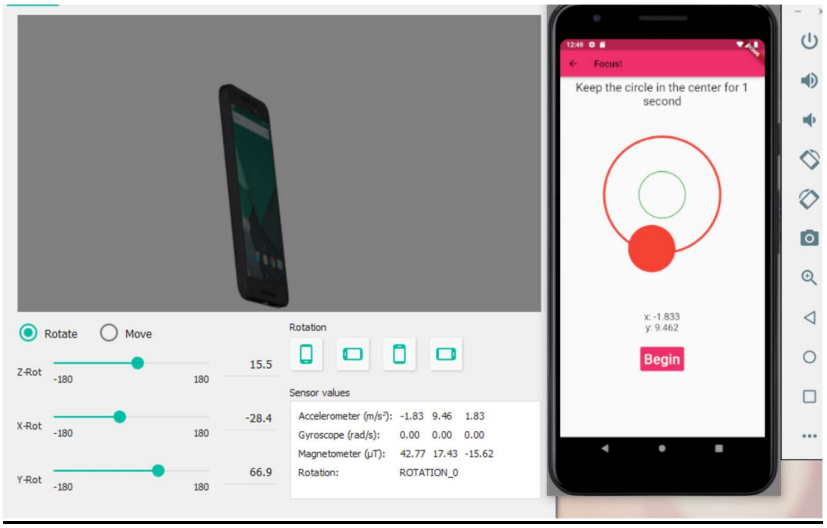
```
        count = 0;
      });
    }
  }

  setPosition(AccelerometerEvent event) {
    if (event == null) {
      return;
    }

    // When x = 0 it should be centered horizontally
    // The left positin should equal (width - 100) / 2
    // The greatest absolute value of x is 10, multipling it by 12 allows the left position to move a total of 120 in either direction.
    setState(() {
      left = ((event.x * 12) + ((width! - 100) / 2));
    });

    // When y = 0 it should have a top position matching the target, which we set at 125
    setState(() {
      top = event.y * 12 + 125;
    });
  }


  startTimer() {
    // if the accelerometer subscription hasn't been created, go ahead and create it
    if (accel == null) {
      accel = accelerometerEvents.listen((AccelerometerEvent eve) {
        setState(() {
          event = eve;
        });
      });
    } else {
      // it has already ben created so just resume it
      accel?.resume();
    }

    // Accelerometer events come faster than we need them so a timer is used to only proccess them every 200 milliseconds
    if (timer == null || !timer!.isActive) {
      timer = Timer.periodic(Duration(milliseconds: 200), (_) {
        // if count has increased greater than 3 call pause timer to handle success
        if (count > 3) {
          pauseTimer();
        } else {
          // proccess the current event
          setColor(event!);
          setPosition(event!);
        }
```

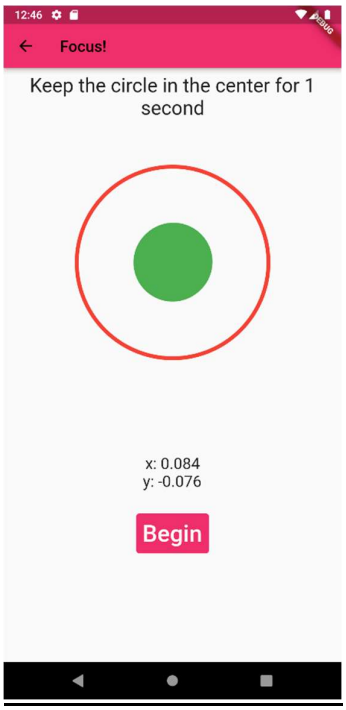
```
    });
   }
  }

  pauseTimer() {
   // stop the timer and pause the accelerometer stream
   timer?.cancel();
   accel?.pause();

   // set the success color and reset the count
   setState(() {
    count = 0;
    color = Colors.green;
   });
  }

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

  @override
  Widget build(BuildContext context) {
   // get the width and height of the screen
   width = MediaQuery.of(context).size.width;
   height = MediaQuery.of(context).size.height;

   return Scaffold(
    appBar: AppBar(
     iconTheme: IconThemeData(
      color: Colors.black, //change your color here
     ),
     title: Text(widget.title,style:TextStyle(color:Colors.black)),
     backgroundColor : Color(0xffef2e6c),
    ),
    body: Column(
     children: [
      Padding(
       padding: const EdgeInsets.all(8.0),
       child: Text('Keep the circle in the center for 1 second',textAlign:
TextAlign.center,style: TextStyle(fontSize:25)),
      ),
      Stack(
       children: [
        // This empty container is given a width and height to set the size of the stack
        Container(
         height: height! / 2,
         width: width,
```

```
    ),

    // Create the outer target circle wrapped in a Position
    Positioned(
      // positioned 50 from the top of the stack
      // and centered horizontally, left = (ScreenWidth - Container width) / 2
      top: 50,
      left: (width! - 250) / 2,
      child: Container(
        height: 250,
        width: 250,
        decoration: BoxDecoration(
          border: Border.all(color: Colors.red, width: 5.0),
          borderRadius: BorderRadius.circular(125),
        ),
      ),
    ),
    // This is the colored circle that will be moved by the accelerometer
    // the top and left are variables that will be set
    Positioned(
      top: top,
      left: left ?? (width! - 100) / 2,
      // the container has a color and is wrappeed in a ClipOval to make it round
      child: ClipOval(
        child: Container(
          width: 100,
          height: 100,
          color: color,
        ),
      ),
    ),
    // inner target circle wrapped in a Position
    Positioned(
      top: 125,
      left: (width! - 100) / 2,
      child: Container(
        height: 100,
        width: 100,
        decoration: BoxDecoration(
          border: Border.all(color: Colors.green, width: 2.0),
          borderRadius: BorderRadius.circular(50),
        ),
      ),
    ),
  ],
),
Text('x: ${(event?.x ?? 0).toStringAsFixed(3)}',style:TextStyle(fontSize: 20)),
Text('y: ${(event?.y ?? 0).toStringAsFixed(3)}',style:TextStyle(fontSize: 20)),
Padding(
  padding: EdgeInsets.symmetric(horizontal: 16.0, vertical: 30.0),
```

```
          child: TextButton(
            style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xffef2e6c))),
            onPressed: startTimer,
            child: Text('Begin.!!',style: TextStyle(fontSize: 30.0,color:Colors.white),),
            // color: Theme.of(context).primaryColor,
            // textColor: Colors.white,
          ),
        )
      ],
    ),
  );
 }
}
```

OUTPUT:





Gesture based UI:

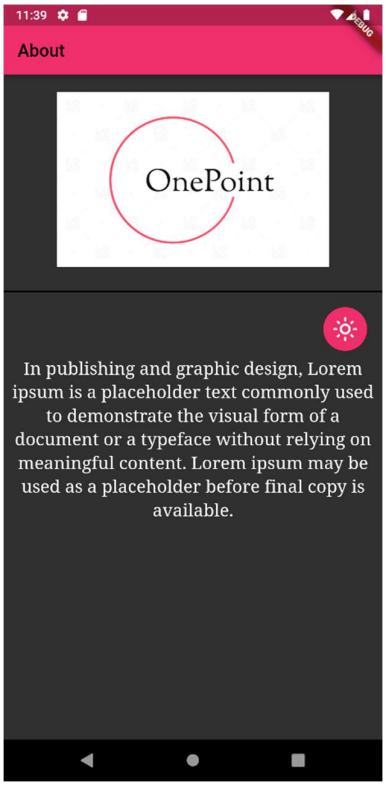
CODE:

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


class AboutPage extends StatefulWidget {
  @override
  _AboutPageState createState() => _AboutPageState();
}

class _AboutPageState extends State<AboutPage> {
  bool _lightIsOn = false;

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: _lightIsOn ? ThemeData.dark() : ThemeData.light(),
        home: Scaffold(
        appBar: AppBar(
          title: Text('About', style: TextStyle(color: Colors.black)),
          backgroundColor: Color(0xffef2e6c),
        ),
        body: Column(children: <Widget>[
          Container(
            margin: EdgeInsets.all(20),
            height: 200,
            width: 350,
            child: Image.asset('assets/images/logo.png'),
          ),
          Divider(color:Colors.black,thickness: 2,),
          Container(
            // alignment: FractionalOffset.center,
            child: Column(
              // mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                GestureDetector(
                  onTap: () {
                    setState(() {
```

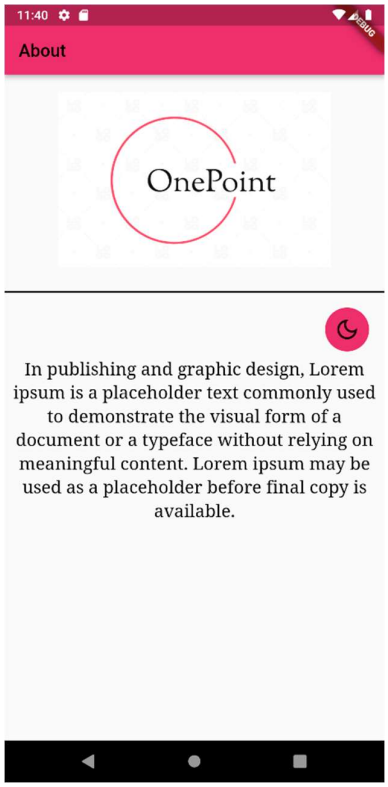
```
              // Toggle light when tapped.
              _lightIsOn = !_lightIsOn;
            });
          },

          child: Container(
            margin: EdgeInsets.fromLTRB(350, 10, 3, 6),
            width : 50,
            height:50,
            padding: const EdgeInsets.all(8),
            // Change button text when light changes state.
            decoration: BoxDecoration(
              shape : BoxShape.circle,
              color: Color(0xffef2e6c),
            ),
            child: Icon(
                _lightIsOn ? Icons.light_mode_outlined : Icons.dark_mode_outlined,
                size: 30),
            ),
          ),
        ],
      ),
    ),
    Text('In publishing and graphic design, '
      'Lorem ipsum is a placeholder text commonly used to demonstrate '
      'the visual form of a document or a typeface without relying on '
      'meaningful content. Lorem ipsum may be used as a placeholder '
      'before final copy is available.',
      textAlign: TextAlign.center,
      softWrap: true,
      style: GoogleFonts.notoSerif(textStyle: TextStyle( color: _lightIsOn ? Colors.white :
Colors.black,fontSize: 20),)
    ),
    ]
    )
  )
  );
 }
}
```

OUTPUT:

Dark mode



Light mode



RESULT:

Thus, GEO positioning, accelerometer, and rich gesture-based UI handling have been implemented using Flutter.

EX.NO:15 SOCIAL MEDIA INTEGRATION

AIM:

To write an application for integrating mobile applications in the market, including social networking software integration with Google.

PROCEDURE:

- Download the following packages using flutter pub add.
  - firebase_auth
  - firebase_core
  - google_sign_in
- In the firebase console, enable Google as a provider under Authentication-> Sign In method.
- Get SHA key, by using the command gradlew signingReport at the android directory of the flutter application.
- Add SHA-1 fingerprint to the application.
- Now, get Google user credential using the await GoogleSignIn().signIn();
- Obtain the auth details from the request.
- Obtain the auth details from the request

CODE:

authentication.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';


class AuthenticationHelper {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  get user => _auth.currentUser;

  Future<String?> signInWithGoogle() async {
    final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();

    final GoogleSignInAuthentication? googleAuth = await googleUser?.authentication;

    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth?.accessToken,
      idToken: googleAuth?.idToken,
    );

    await FirebaseAuth.instance.signInWithCredential(credential);
    return null;
  }
```

```
  Future<UserCredential> signInWithFacebook() async {
    // Trigger the sign-in flow
    final LoginResult loginResult = await FacebookAuth.instance.login();

    // Create a credential from the access token
    final OAuthCredential facebookAuthCredential =
FacebookAuthProvider.credential(loginResult.accessToken.token);

    // Once signed in, return the UserCredential
    return FirebaseAuth.instance.signInWithCredential(facebookAuthCredential);
  }

//SIGN UP METHOD
  Future<String?> signUp({required String email, required String password}) async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

  //SIGN IN METHODJ
  Future<String?> signIn({required String email, required String password}) async {
    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

  //SIGN OUT METHOD
  Future<void> signOut() async {
    await _auth.signOut();

    print('signout');
  }
}
```

login.dart

```
import 'package:flutter/material.dart';
import './authentication.dart';
import './home.dart';
import './signup.dart';
```

```
class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: ListView(
        padding: EdgeInsets.all(8.0),
        children: <Widget>[
          SizedBox(height: 80),
          // logo
          Column(
            children: [
              Image.asset('assets/images/logo.png'),
              SizedBox(height: 50),
              Text(
                'Welcome back!',
                style: TextStyle(fontSize: 24),
              ),
            ],
          ),

          SizedBox(
            height: 50,
          ),

          Padding(
            padding: const EdgeInsets.all(16.0),
            child: LoginForm(),
          ),

          SizedBox(height: 20),
          Row(
            children: <Widget>[
              SizedBox(width: 30),
              Text('New here ? ',
                  style: TextStyle(fontWeight: FontWeight.bold, fontSize: 20)),
              GestureDetector(
                onTap: () {
                  Navigator.pushReplacement(context,MaterialPageRoute(builder: (context) =>
Signup()));
                },
                child: Text('Get Registered Now..',
                    style: TextStyle(fontSize: 20, color: Color(0xffef2e6c))),
              )
            ],
          ),
          Row(
            children: <Widget>[
              SizedBox(width: 30),
              GestureDetector(
```

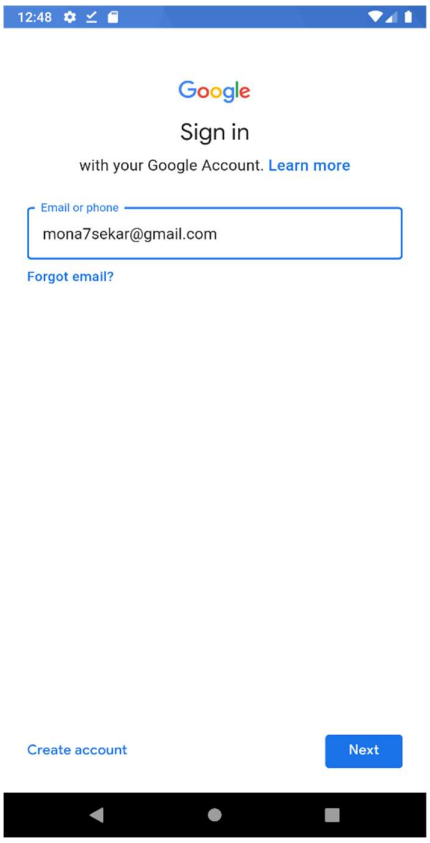
```
            onTap: () {
               AuthenticationHelper()
                  .signInWithGoogle()
                  .then((result) {
                 if (result == null) {
                  Navigator.pushReplacement(context,
                     MaterialPageRoute(builder: (context) => MyApp()));
                 } else {
                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                   content: Text(
                    result,
                    style: TextStyle(fontSize: 16),
                   ),
                  ));
                 }
               });
                },
            child: Text('Sign in with Google',
               style: TextStyle(fontSize: 20, color: Color(0xffef2e6c))),
           )
          ],
        ),
       ],
     ),
   );
  }


}

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

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

class _LoginFormState extends State<LoginForm> {
  final _formKey = GlobalKey<FormState>();

  String? email;
  String? password;

  bool _obscureText = true;

  @override
  Widget build(BuildContext context) {
   return Form(
     key: _formKey,
     child: Column(
```

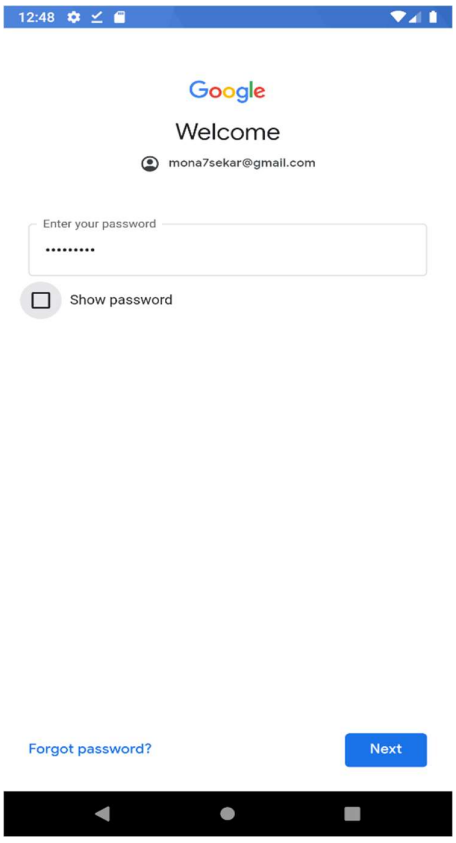
```
mainAxisAlignment: MainAxisAlignment.spaceAround,
children: <Widget>[
 // email
 TextFormField(
  // initialValue: 'Input text',
  decoration: InputDecoration(
   prefixIcon: Icon(Icons.email_outlined,color:Colors.black),
   labelText: 'Email',
   labelStyle: TextStyle(
    color: Color(0xffef2e6c),
   ),
   enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(
     const Radius.circular(100.0),
    ),
   ),
   focusedBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(
     const Radius.circular(100.0),
    ),
    borderSide:  BorderSide(color: Color(0xffef2e6c) ),

   ),
  ),
  validator: (value) {
   if (value!.isEmpty) {
    return 'Please enter some text';
   }
   return null;
  },
  onSaved: (val) {
   email = val;
  },
 ),
 SizedBox(
  height: 20,
 ),

 // password
 TextFormField(
  // initialValue: 'Input text',
  decoration: InputDecoration(
   labelText: 'Password',
   labelStyle: TextStyle(
    color: Color(0xffef2e6c),
   ),
   prefixIcon: Icon(Icons.lock_outline,color:Colors.black),
   enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(
     const Radius.circular(100.0),
```

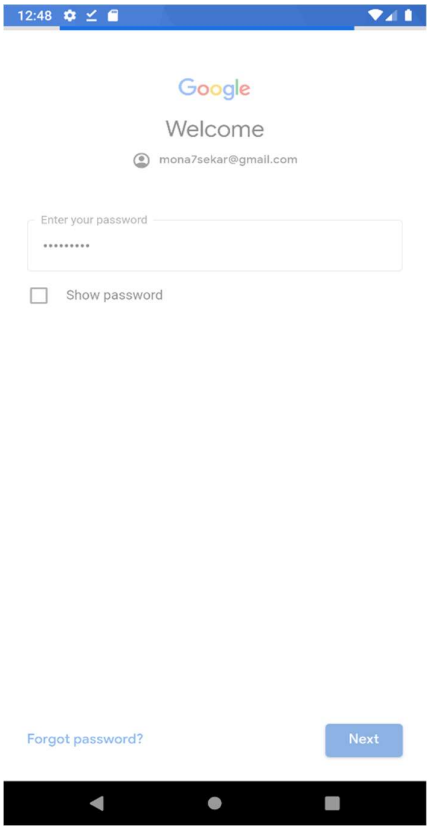
```
        ),
      ),
      focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.all(
          const Radius.circular(100.0),
        ),
        borderSide:  BorderSide(color: Color(0xffef2e6c) ),

      ),
      suffixIcon: GestureDetector(
        onTap: () {
          setState(() {
            _obscureText = !_obscureText;
          });
        },
        child: Icon(
          _obscureText ? Icons.visibility_off : Icons.visibility,
        ),
      ),
    ),
    obscureText: _obscureText,
    onSaved: (val) {
      password = val;
    },
    validator: (value) {
      if (value!.isEmpty) {
        return 'Please enter some text';
      }
      return null;
    },
  ),

  SizedBox(height: 30),

  SizedBox(
    height: 54,
    width: 184,
    child: ElevatedButton(
      onPressed: () {
        // Respond to button press

        if (_formKey.currentState!.validate()) {
          _formKey.currentState!.save();

          AuthenticationHelper()
              .signIn(email: email!, password: password!)
              .then((result) {
            if (result == null) {
              Navigator.pushReplacement(context,
                  MaterialPageRoute(builder: (context) => MyApp()));
```

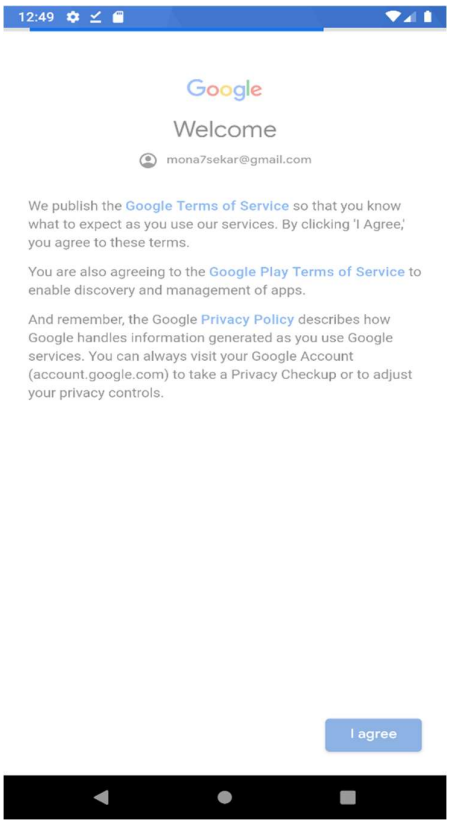
```
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                    content: Text(
                      result,
                      style: TextStyle(fontSize: 16),
                    ),
                  ));
                }
              });
            }
          },
          style: ElevatedButton.styleFrom(
              shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.all(Radius.circular(24.0))),
          backgroundColor: Color(0xffef2e6c)),
          child: Text(
            'Login',
            style: TextStyle(fontSize: 24),
          ),
        ),
      ),
    ],
  ),
 );
 }
}
```

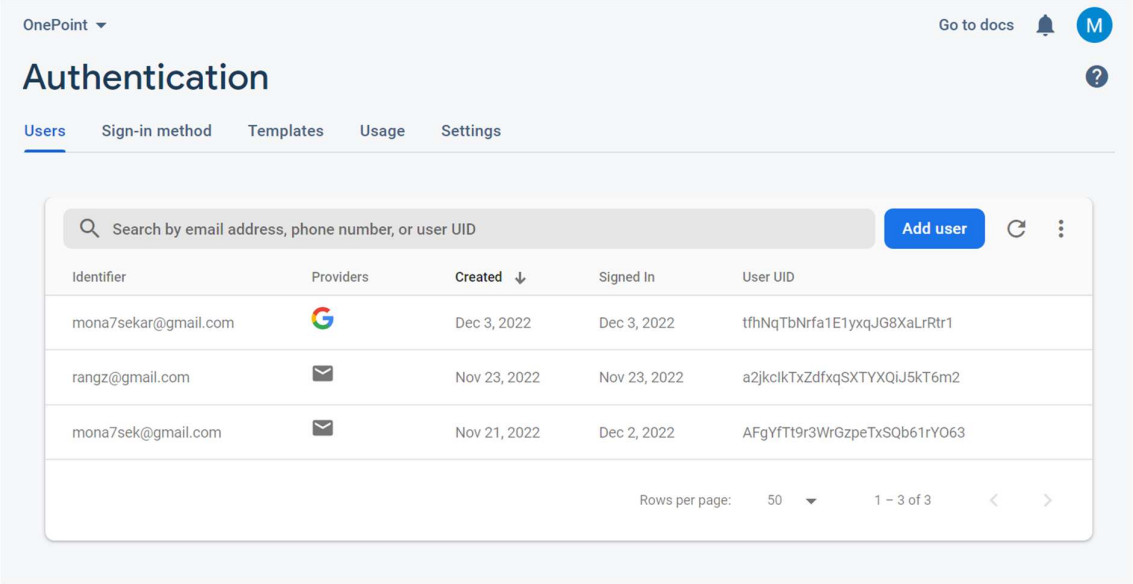
OUTPUT:


12:48
Google
Sign in
with your Google Account. Learn more
Email or phone
mona7sekar@gmail.com
Forgot email?
Create account
Next


12:48
Google
Welcome
mona7sekar@gmail.com
Enter your password
Show password
Forgot password?
Next


12:48
Google
Welcome
mona7sekar@gmail.com
Enter your password
Show password
Forgot password?
Next


12:49
Google
Welcome
mona7sekar@gmail.com
We publish the Google Terms of Service so that you know what to expect as you use our services. By clicking 'I Agree,' you agree to these terms.
You are also agreeing to the Google Play Terms of Service to enable discovery and management of apps.
And remember, the Google Privacy Policy describes how Google handles information generated as you use Google services. You can always visit your Google Account (account.google.com) to take a Privacy Checkup or to adjust your privacy controls.
I agree



RESULT:

Thus, an application that uses social networking software (Google) for authentication has been implemented.