**Ex-13**

**CONNECTIVITY VIA SOAP OR REST**

**Aim:**

To develope an application for connectivity via SOAP or REST

**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["dateAdded"]), 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://api.quotable.io/random%27)%3B) 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*,

),

))

],

),

),

),

)

],

),

),

);

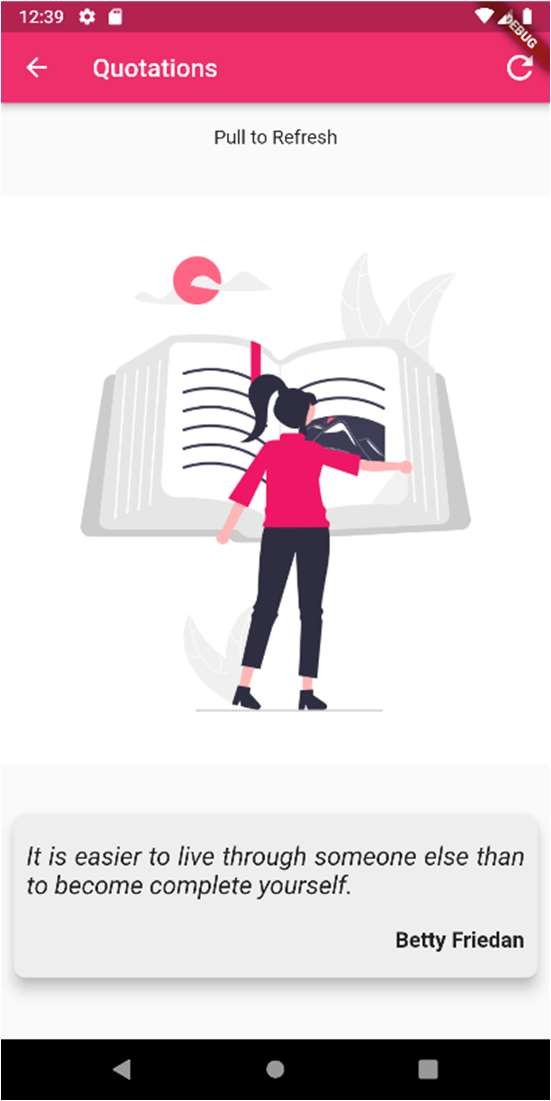
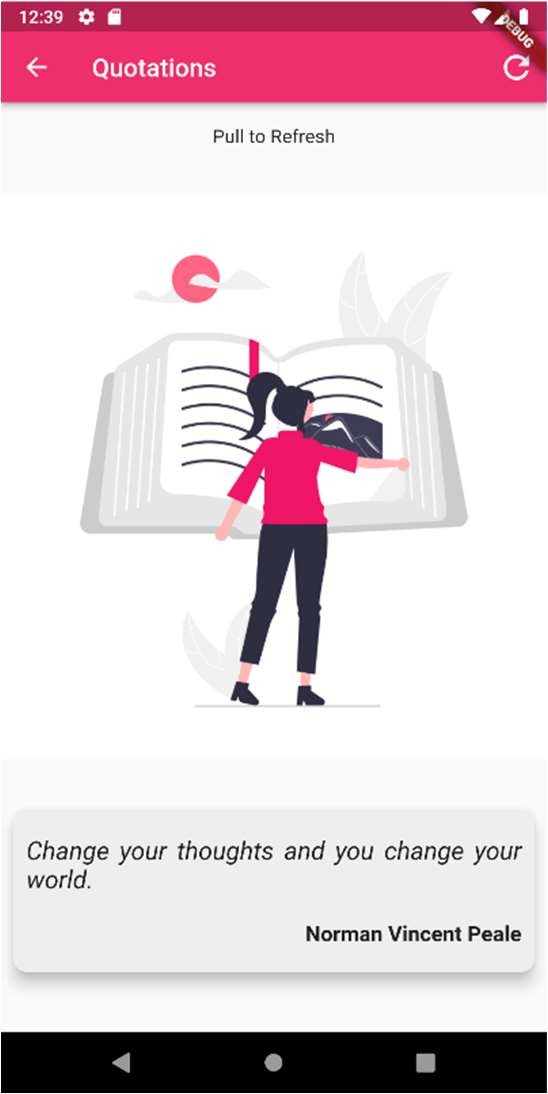
}

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-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) {

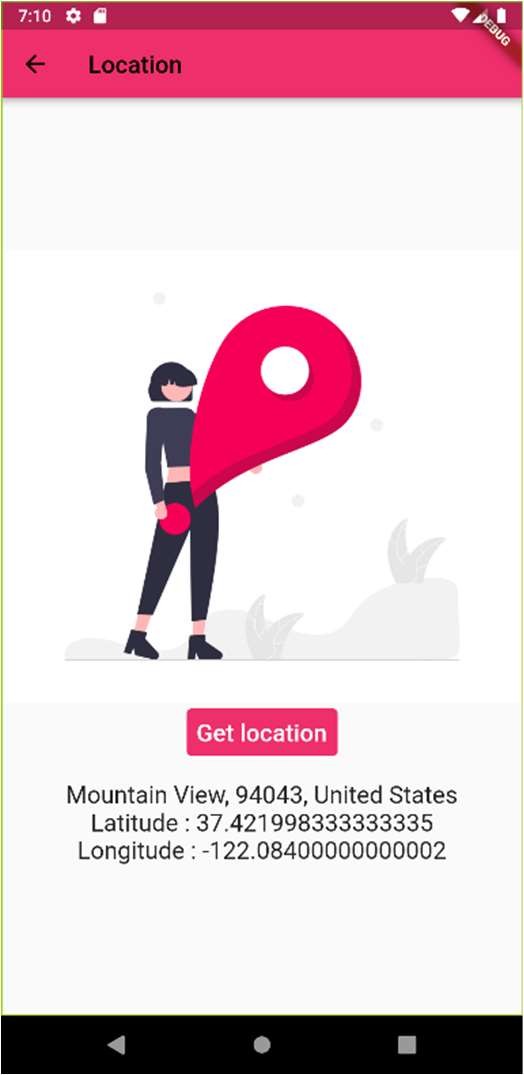
print(e);

}

}

}

**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,*

),

)

],

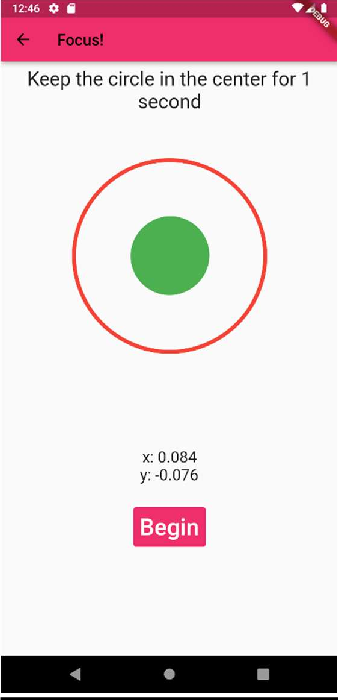
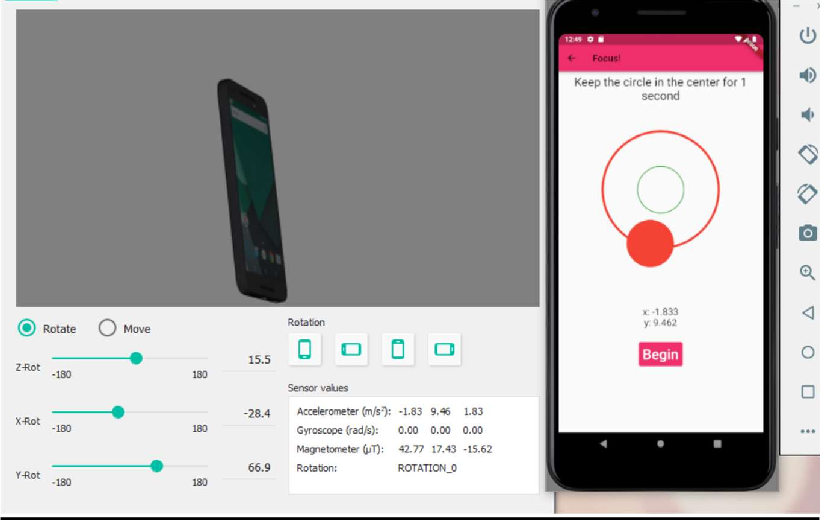
),

);

}

}

# 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),)

),

]

)

)

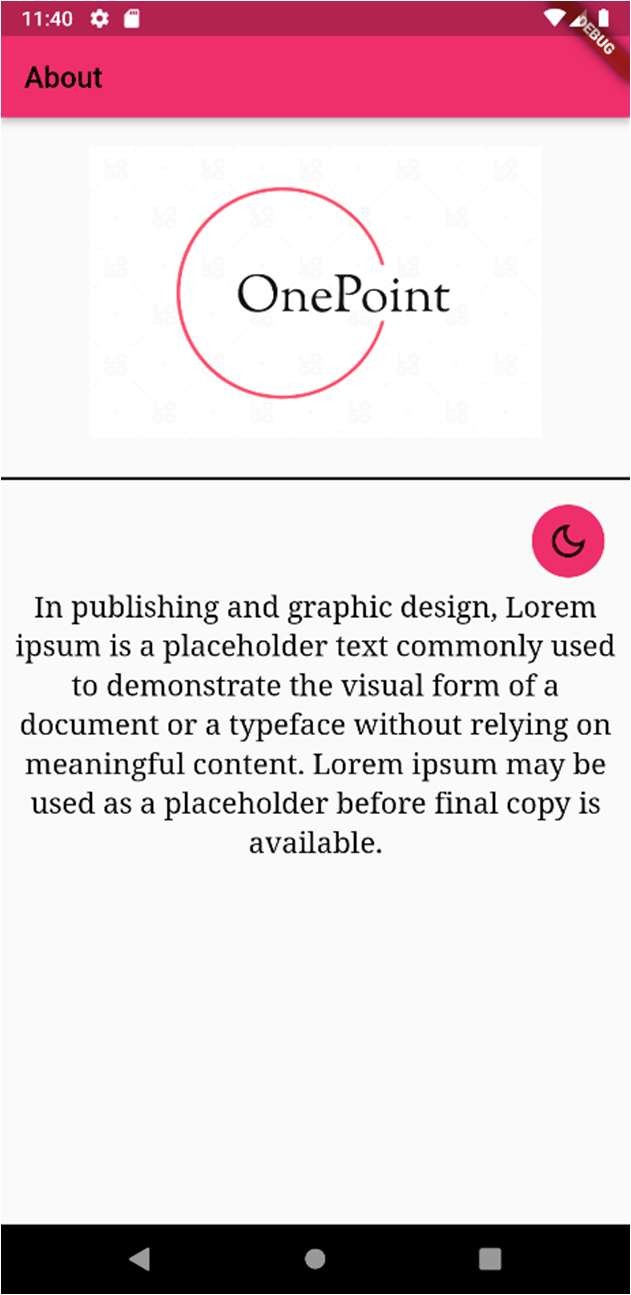
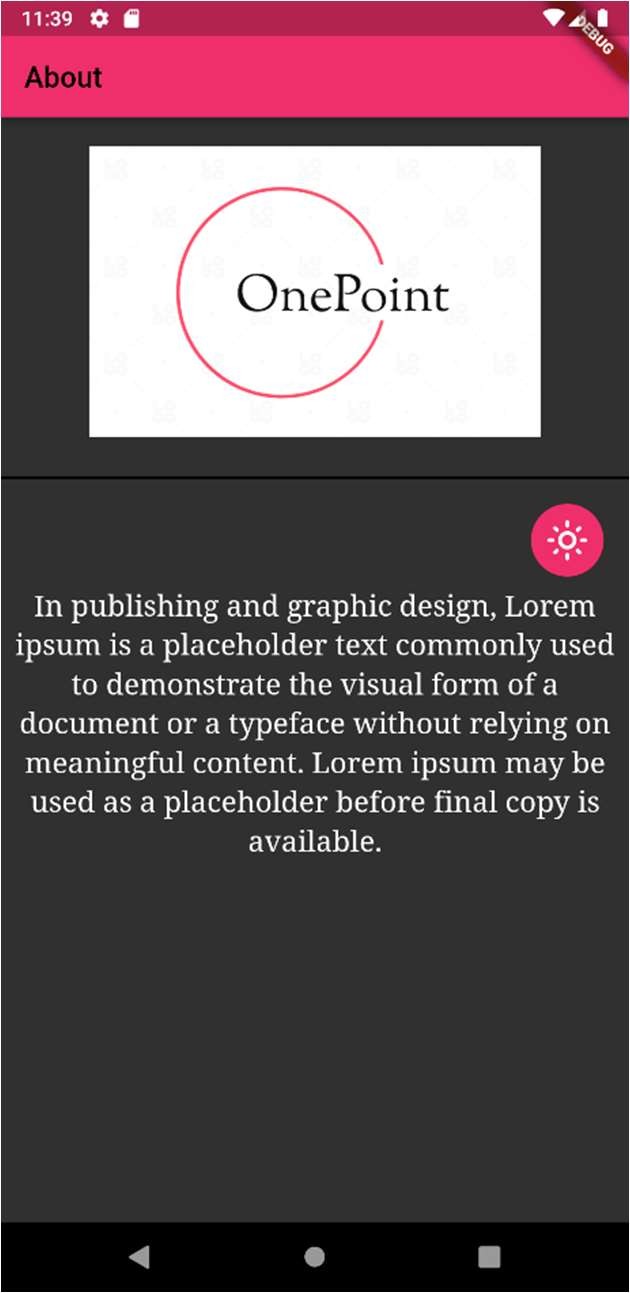
);

}

}

# OUTPUT:

Dark mode Light mode



# RESULT:

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

# Ex-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),

),

),

),

],

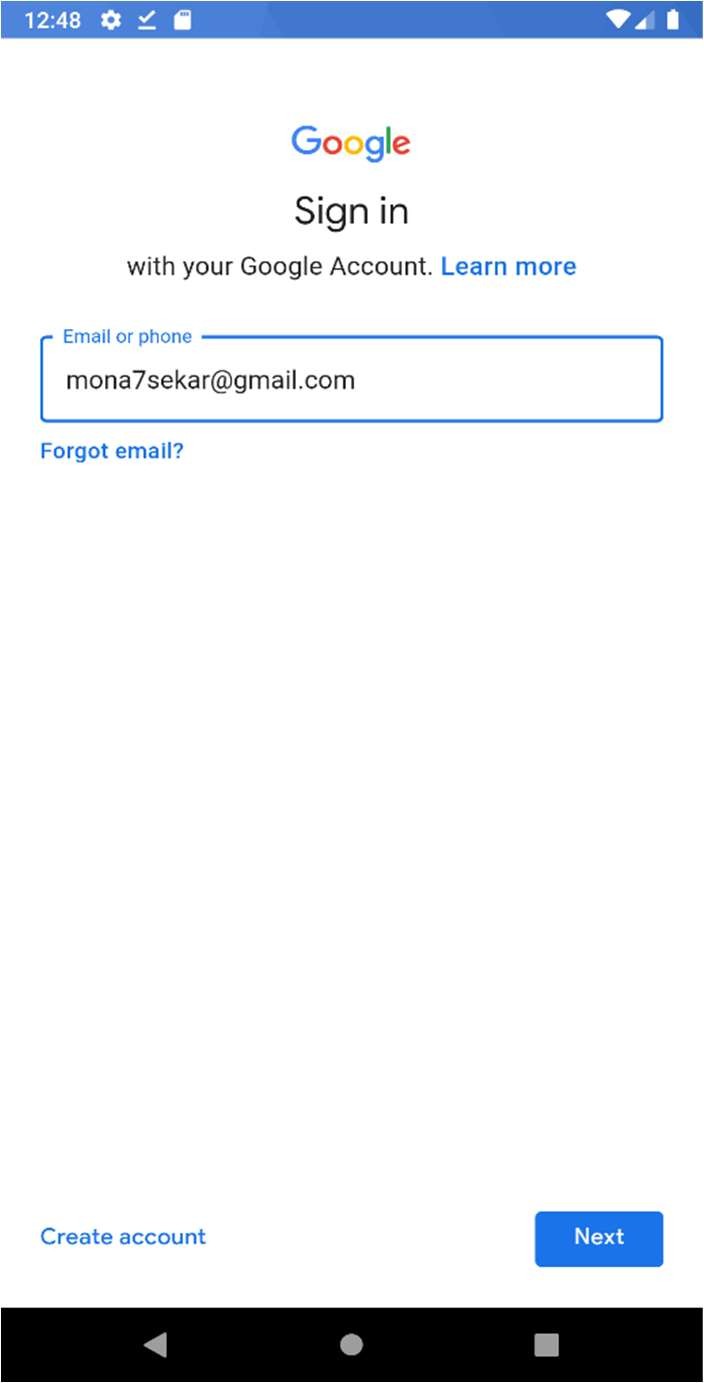
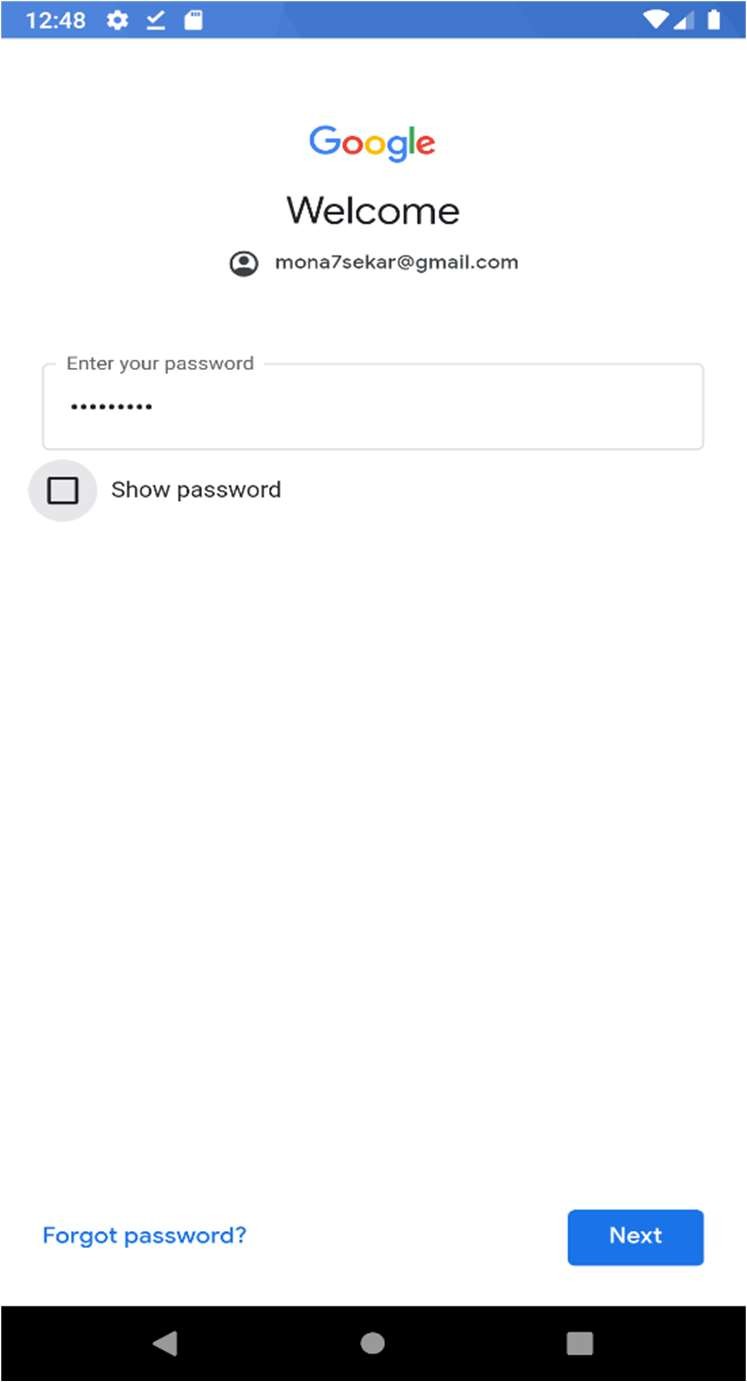
),

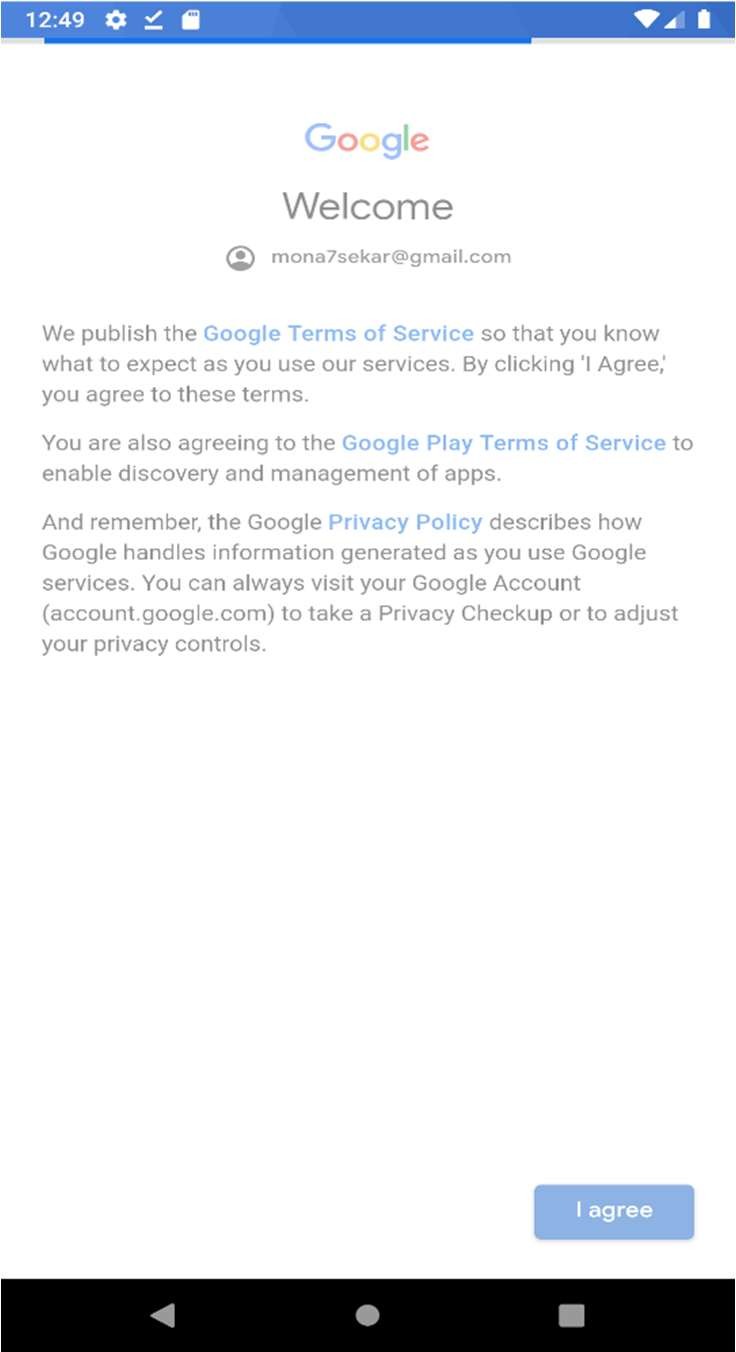
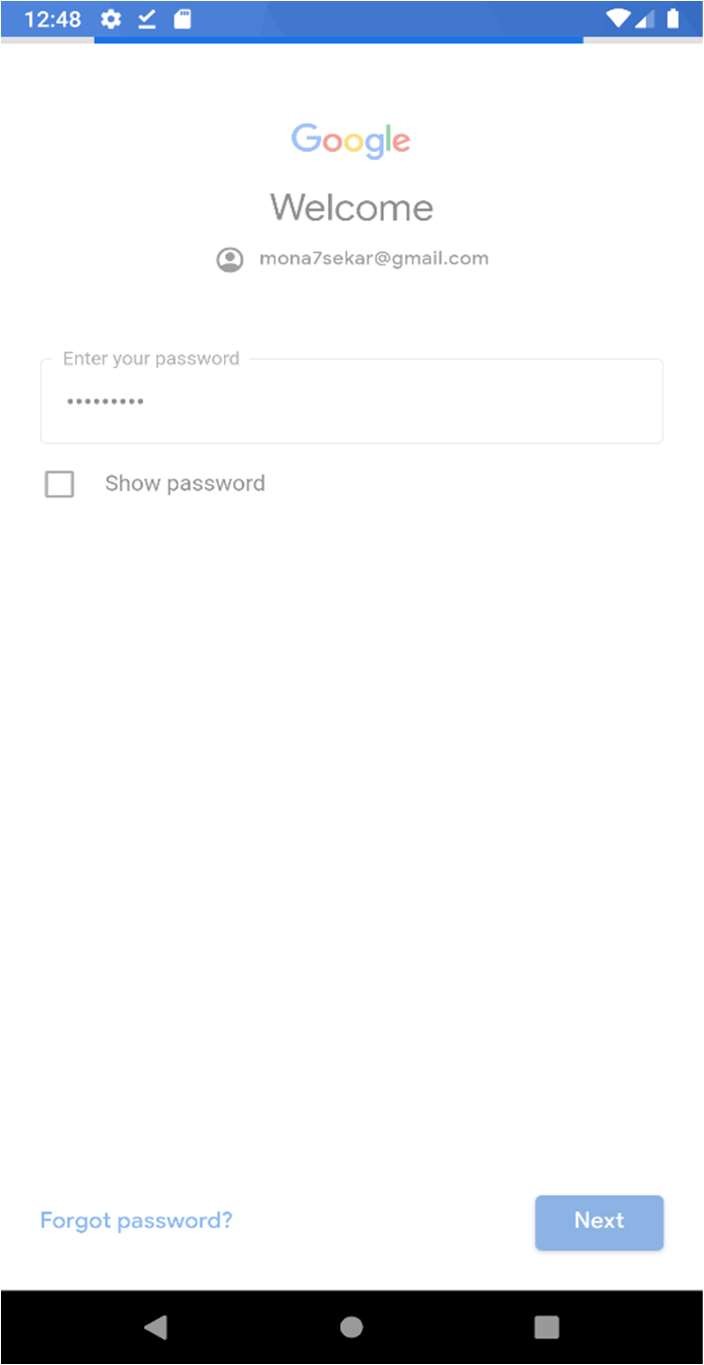
);

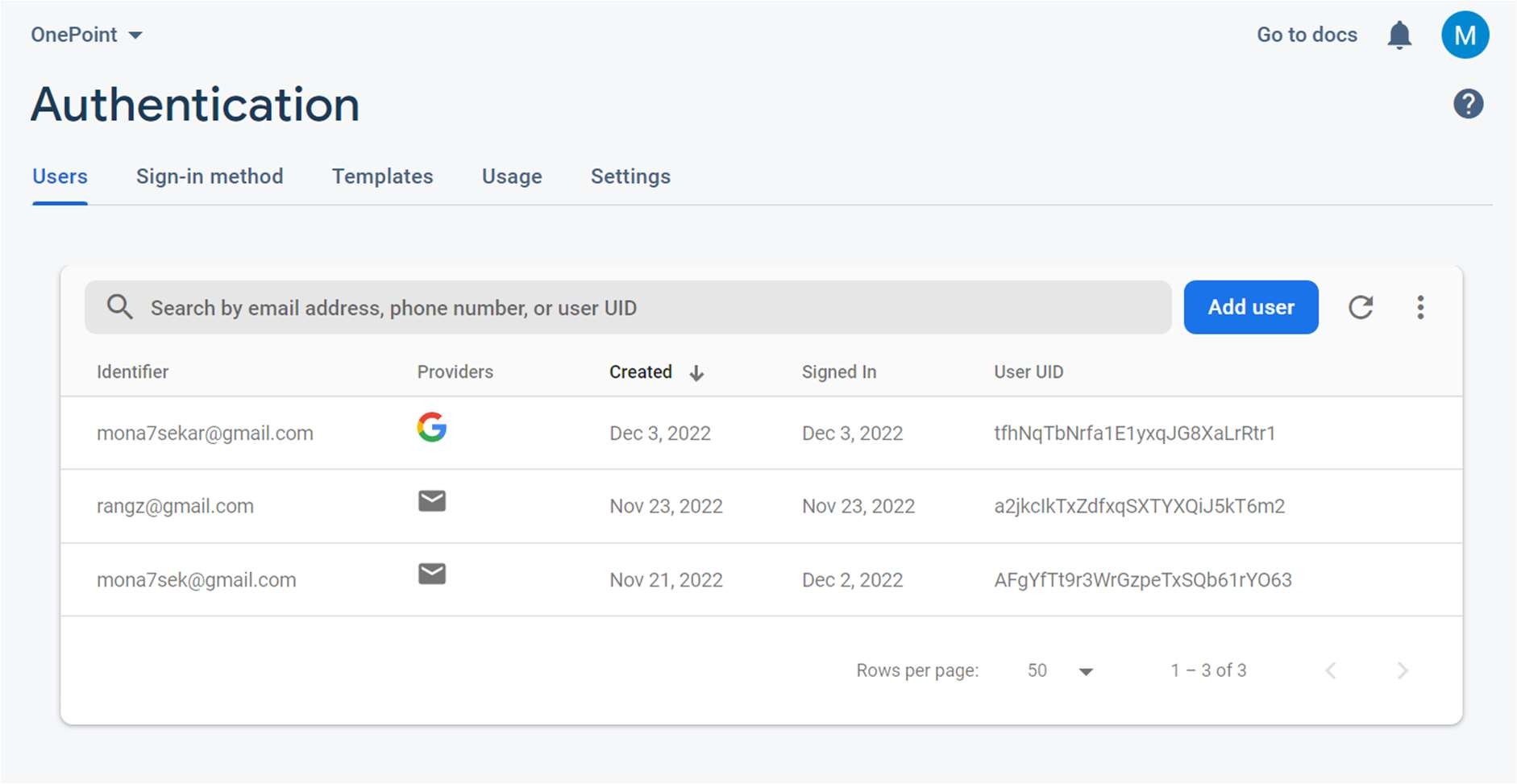
}

}

# Output:





# RESULT:

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