Name : Pranav Trivedi
Roll No : 61
Lab : Mad Lab
Class : Teit2
Batch : I6

Experiment No : 05

Aim : To create an interactive sign in Form using form widget.

Theory :

Form Widget :
In Flutter, a form widget is a special widget that allows you to collect user input and validate it. A form widget can contain multiple form fields, such as text input fields, checkboxes, radio buttons, and dropdown menus.

The Form widget provides several important features, including:

1. Validation: You can validate user input by defining validators for each form field. These validators can check if the input is valid, such as if the user entered a valid email address or if the input is required.
2. Saving Form Data: Once the user submits the form, the form widget will automatically collect all the data entered by the user and store it in a map.
3. Submitting Form Data: The Form widget also provides a built-in way to submit the form data to a server or other backend system.

TextFormField :

In Flutter, a TextFormField widget is a type of form field widget that allows users to enter text input. TextFormField is a stateful widget that provides built-in features like input validation, error messages, and text formatting options.

To use a TextFormField widget, you need to provide it with a TextEditingController and define a validator function. The controller is used to control the text input in the text field and the validator is used to validate the input.

Validation of form:
In Flutter, validation of a form refers to the process of checking user input to ensure that it meets certain requirements or constraints. For example, you might want to ensure that a user enters a valid email address, or that a required field is not left blank.

Code :

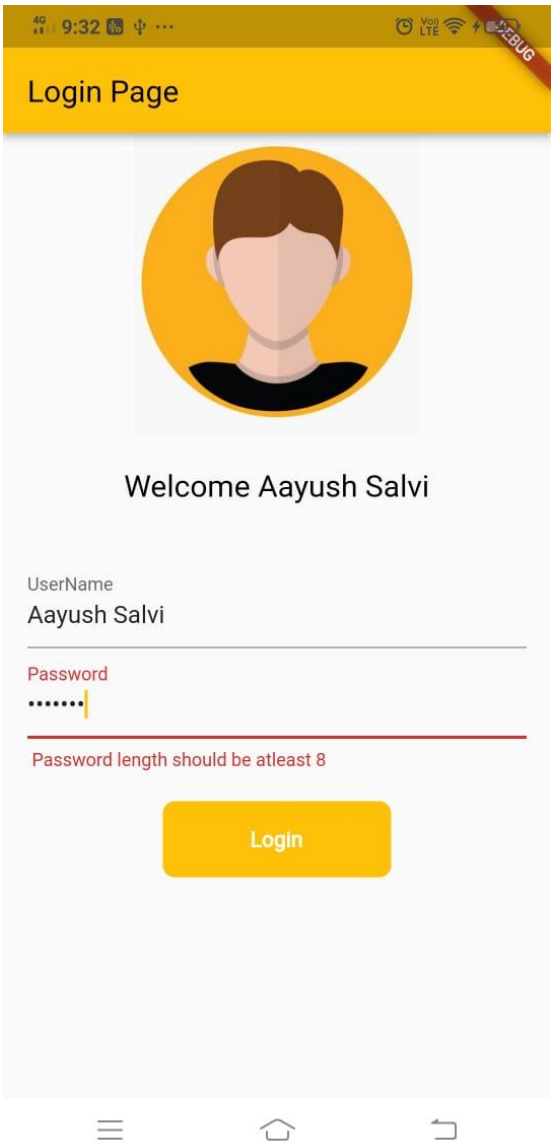
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.amber,
      ),
      home: const MyHomePage(title: 'Login Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  String name = " ";
  String pass = " ";
  bool changebutton = false;
```

```
final _formkey = GlobalKey<FormState>();

animate(BuildContext context) async {
  if (_formkey.currentState!.validate()) {
    setState(() {
      changebutton = true;
    });
    await Future.delayed(Duration(seconds: 1));
    setState(() {
      changebutton = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: SingleChildScrollView(
      child: Form(
        key: _formkey,
        child: Column(
          // ignore: prefer_const_literals_to_create_immutables
            children: [
              Image.asset(
                "assets/img/profile(1).png",
                height: 200,
                width: 275,
              ),
              SizedBox(height: 20),
              Text(
                "Welcome $name",
                style: TextStyle(
                    fontSize: 20,
                    color: Colors.black,
                    fontWeight: FontWeight.normal),
              ),
              SizedBox(height: 20),
              Padding(
```

```
padding: const EdgeInsets.all(16.0),
child: Column(
  children: [
    TextFormField(
      decoration: InputDecoration(
          hintText: "Enter UserName",
          labelText: "UserName "),
      validator: (value) {
        if (value!.isEmpty) {
          return "Username cannot be empty";
        }
        return null;
      },
      onChanged: (value) {
        name = value;
        setState(() {});
      },
    ),
    TextFormField(
      obscureText: true,
      decoration: InputDecoration(
          hintText: "Enter Password",
          labelText: "Password "),
      validator: (value) {
        if (value!.isEmpty) {
          return "Password cannot be empty";
        } else if (value.length < 8) {
          return " Password length should be atleast 8";
        }
        return null;
      },
      onChanged: (value) {
        pass = value;
        setState(() {});
      },
    ),
    SizedBox(
      height: 20,
    ),
    InkWell(
      onTap: () => {animate(context), print(name), print(pass)},
```

```
                    child: AnimatedContainer(
                        duration: Duration(seconds: 1),
                        width: changebutton ? 50 : 150,
                        height: 50,
                        alignment: Alignment.center,
                        child: changebutton
                            ? Icon(Icons.done, color: Colors.white)
                            : Text(
                          "Login",
                          style: TextStyle(
                              color: Colors.white,
                              fontWeight: FontWeight.bold),
                        ),
                        decoration: BoxDecoration(
                          color: Colors.amber,
                          borderRadius: BorderRadius.circular(
                              changebutton ? 50 : 8),
                        )),
                  )
                ],
              ),
            )
          ]),
      ),
    ),
  );
 }
}
```

Output :-



Conclusion :- Hence , create an interactive sign in Form using form widget

Name : Pranav Trivedi

Roll No : 61

Lab : MadLab

Class :TEIT2

Batch : I6

Experiment No : 01

Aim :

Theory :

Flutter is a mobile app SDK (software development kit) for building high-performance, high-fidelity apps for iOS and Android.

With powerful graphics and animation libraries, the Flutter framework makes it easy to build user interfaces that react smoothly in response totouch.

Flutter is built on the Dart programming language and provides a fast development workflow with hot reloading, so you can quickly iterate on yourcode.

Steps :

1. Install Jdk17 and set path to Environment variable
2. Install git and set path to Environment Variable
3. Install flutter from ‘docs.flutter.dev’ and set the path of it in the Environment variable.
4. Install Android studio
5. In Android studio go to more actions and then go to SDK tools and inthis install command line tools.
6. Now open cmd and enter flutter, here you will see information of flutter.
7. Now in cmd , enter ‘flutter doctor’, it will show that all the requirementsare installed properly or not.
8. Now in cmd, enter ‘flutter doctor –android licences’.
9. Again run ‘flutter doctor’, it will show all the green ticks if all the requirements are installed properly.
10. Go to android studio and then go to plugins and install flutter inthat.
11. Now go to create new project and select flutter and there fill allthe details (i.e Path of flutter,name of project ,etc) and click on next.

12. A flutter project will be created.

13. Now create a mobile Emulator

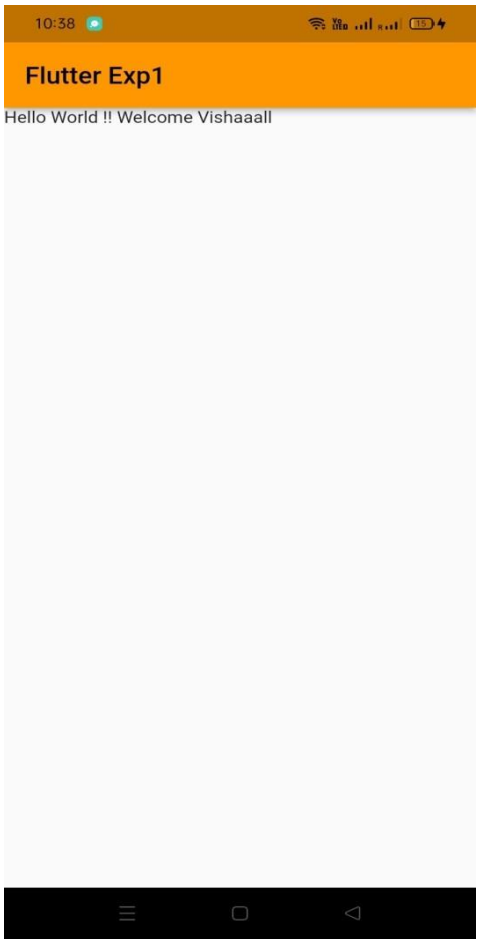
14. Run the project with the help of the emulator.

Code :

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.@override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Exp1 ',
      theme: ThemeData(

        primarySwatch:  Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Exp1'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});



  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {



  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.title:
        Text(widget.title),
      ),
      body: Column(
      children: [
        Text('Hello World !! Welcome Vishaaall')
      ],
```

```
    ),
      // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Output :



Conclusion : Therefore we have successfully installed Android studio andflutter and created first hello world app.

ATHARVA EDUCATIONAL TRUST'S
ATHARVA COLLEGE OF ENGINEERING
(Approved by AICTE, Recognized by Government of Maharashtra
& Affiliated to University of Mumbai - Estd. 1999 - 2000)
Department of Electronics and Telecommunication Engineering

14th February 2023

Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

EXPERIMENT NO. 3

AIM: To create Buttons, Icon button and display the Message box after pressing the button.

THEORY:

CODE:

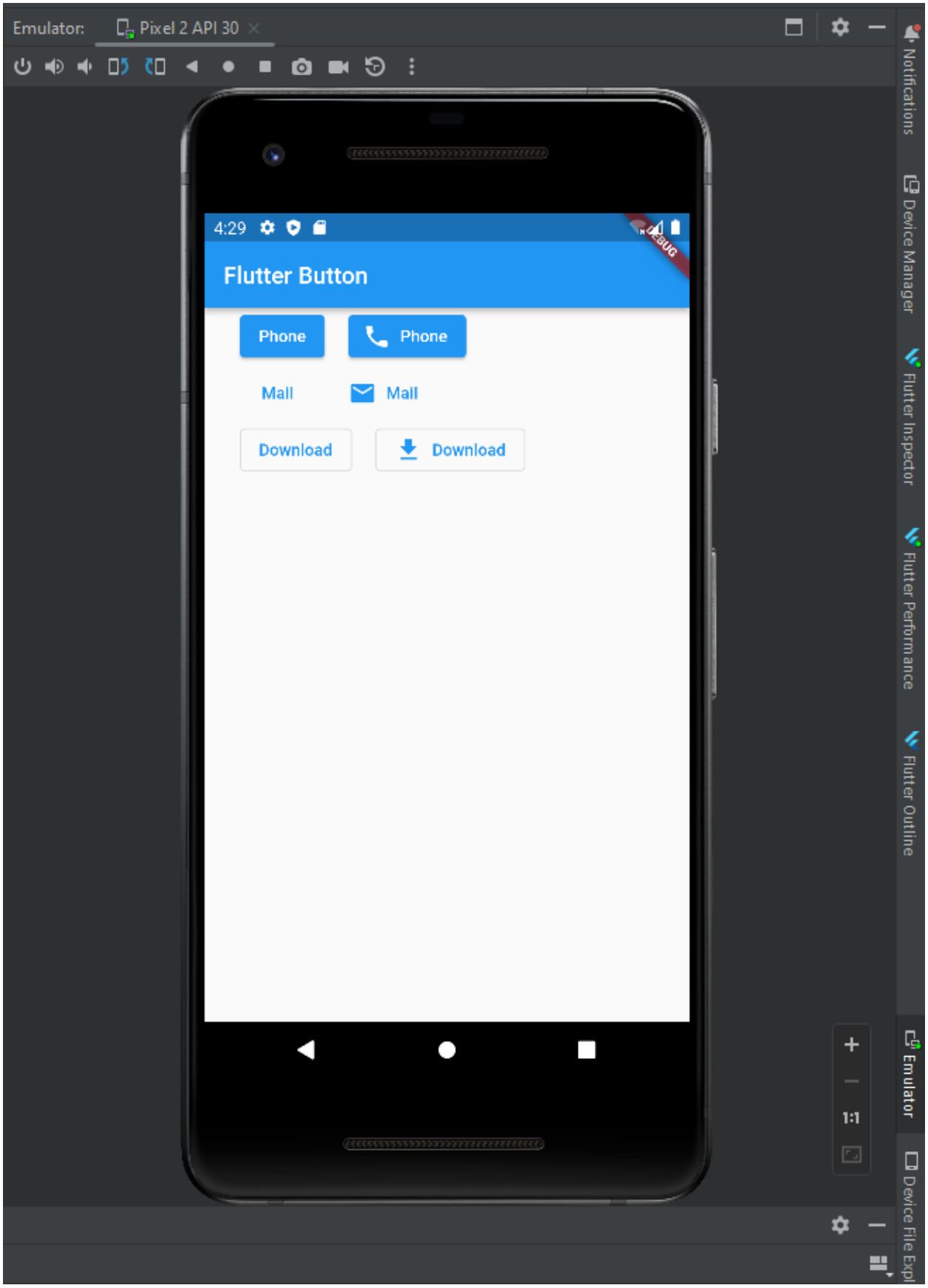
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Demo',
     theme: ThemeData(
       // This is the theme of your application.
       //
       // Try running your application with "flutter run". You'll see the
       // application has a blue toolbar. Then, without quitting the app, try
       // changing the primarySwatch below to Colors.green and then invoke
       // "hot reload" (press "r" in the console where you ran "flutter run",
       // or simply save your changes to "hot reload" in a Flutter IDE).
       // Notice that the counter didn't reset back to zero; the application
       // is not restarted.
       primarySwatch: Colors.blue,
     ),
     home: const MyHomePage(title: 'Flutter Demo Home Page'),
   );
 }
}
```

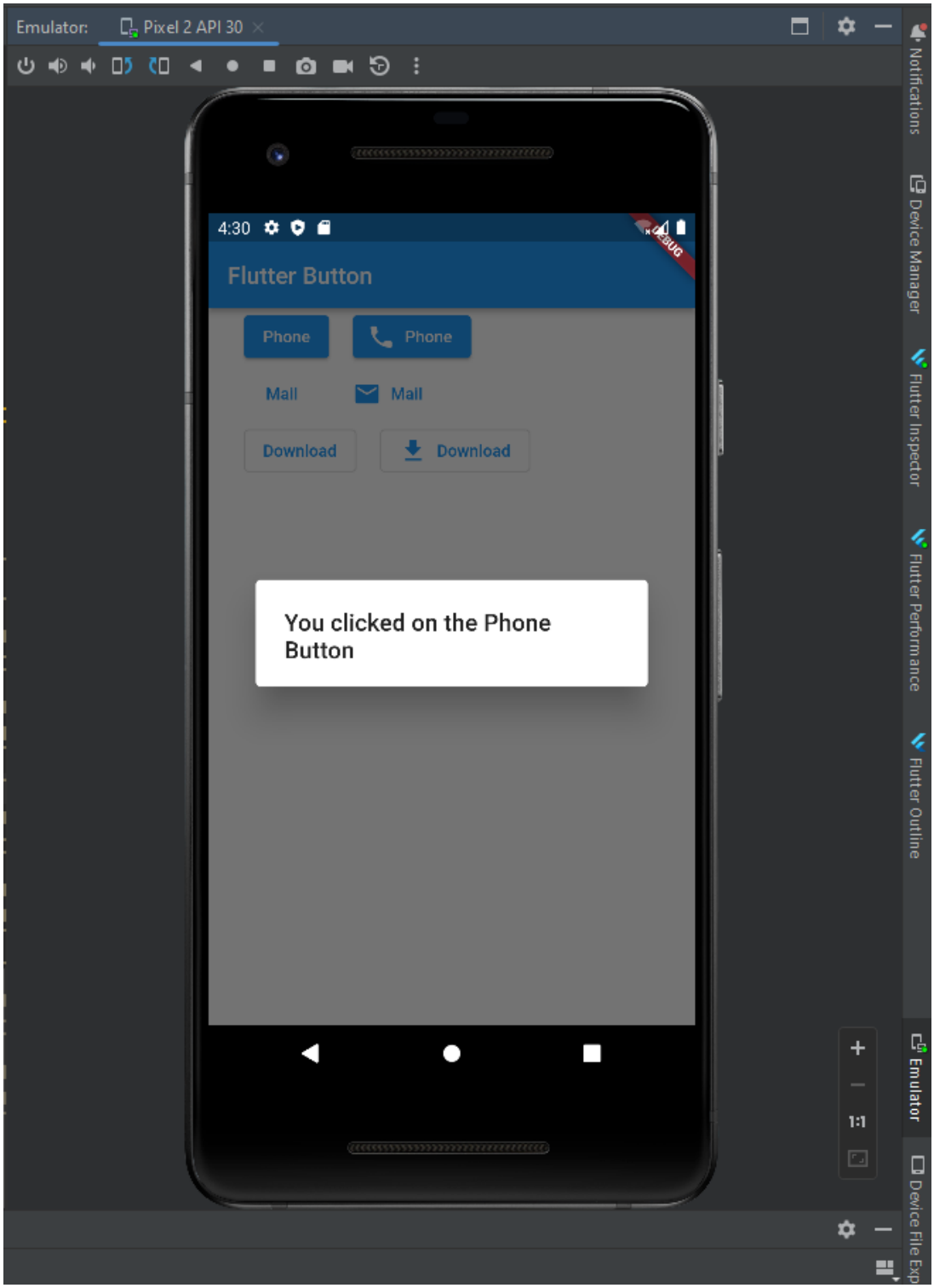
```
class MyHomePage extends StatefulWidget {
 const MyHomePage({super.key, required this.title});

 // This widget is the home page of your application. It is stateful, meaning
 // that it has a State object (defined below) that contains fields that
affect
 // how it looks.

 // This class is the configuration for the state. It holds the values (in
this
 // case the title) provided by the parent (in this case the App widget) and
 // used by the build method of the State. Fields in a Widget subclass are
 // always marked "final".

 final String title;

 @override
 State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
 int _counter = 0;

 void _incrementCounter() {
   setState(() {
     // This call to setState tells the Flutter framework that something has
     // changed in this State, which causes it to rerun the build method below
     // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not
be
     // called again, and so nothing would appear to happen.
     _counter++;
   });
 }

 @override
 Widget build(BuildContext context) {
   // This method is rerun every time setState is called, for instance as done
   // by the _incrementCounter method above.
   //
   // The Flutter framework has been optimized to make rerunning build methods
   // fast, so that you can just rebuild anything that needs updating rather
   // than having to individually change instances of widgets.
   return Scaffold(
     appBar: AppBar(
       title: Text('Flutter Button'),
     ),
     body: Column(
     children: [
       Row(
         children: [
           SizedBox(width: 30),
           ElevatedButton(
               onPressed:() {
                 showDialog(
                     context: context,
```

```
                    builder: (ctx)=> AlertDialog(
                        title: const Text('You clicked on the Phone Button')
                    ));
                },
                child: Text('Phone')
            ),
            SizedBox( width:20),
            ElevatedButton.icon(
                onPressed: (){
                  showDialog(
                      context: context,
                      builder: (ctx)=> AlertDialog(
                        title: const Text('You clicked on the Phone Button')
                      ));
                },
                icon: Icon(
                  Icons.phone
                ),
                label: Text('Phone'))
          ],
        ),
        Row(
          children: [
            SizedBox(width: 30),
            TextButton(
                onPressed:() {
                  showDialog(
                      context: context,
                      builder: (ctx)=> AlertDialog(
                          title: const Text('You clicked on the Mail Button')
                      ));
                },
                child: Text('Mail')
            ),
            SizedBox( width:20),
            TextButton.icon(
                onPressed: (){
                  showDialog(
                      context: context,
                      builder: (ctx)=> AlertDialog(
                          title: const Text('You clicked on the Mail Button')
                      ));
                },
                icon: Icon(
                    Icons.mail
                ),
                label: Text('Mail'))
          ],
        ),
        Row(
          children: [
            SizedBox(width: 30),
            OutlinedButton(
                onPressed:() {
                  showDialog(
                      context: context,
                      builder: (ctx)=> AlertDialog(
```

```
                                title: const Text('You clicked on the Download
Button')
                        ));
                },
                child: Text('Download')
            ),
            SizedBox( width:20),
            OutlinedButton.icon(
                onPressed: (){
                  showDialog(
                      context: context,
                      builder: (ctx)=> AlertDialog(
                                title: const Text('You clicked on the Download
Button')
                        ));
                },
                icon: Icon(
                    Icons.download
                ),
                label: Text('Download'))
          ],
        )
      ],
      )
      , // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

SCREENSHOT/OUTPUT:


Emulator:
Pixel 2 API 30
4:29
DEBUG
Flutter Button
Phone
Phone
Mail
Mail
Download
Download
1:1
Notifications
Device Manager
Flutter Inspector
Flutter Performance
Flutter Outline
Emulator
Device File Expl



CONCLUSION: Thus we created Buttons, Icon button and display the Message box after pressing the button.

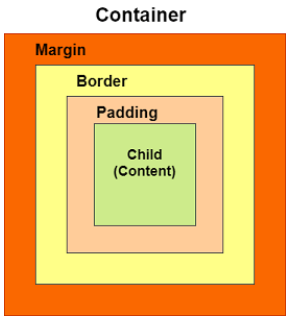
Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

EXPERIMENT NO. 2

AIM: To design flutter UI by including common Widgets like Container, Columns to display text and image.

THEORY:

The container in Flutter is a parent widget that can contain multiple child widgets and manage them efficiently through width, height, padding, background color, etc. It is a widget that combines common painting, positioning, and sizing of the child widgets. It is also a class to store one or more widgets and position them on the screen according to our needs. Generally, it is similar to a box for storing contents. It allows many attributes to the user for decorating its child widgets, such as using margin, which separates the container with other contents.



Properties of Container widget

1. **child:** This property is used to store the child widget of the container.
2. **color:** This property is used to set the **background color of the text**. It also changes the background color of the entire container.
3. **height and width:** This property is used to set the container's height and width according to our needs. By default, the container always takes the space based on its child widget.
4. **margin:** This property is used to surround the e**mpty space around the container**. We can observe this by seeing white space around the container. Suppose we have used the **EdgeInsets.all(25)** that set the equal margin in all four directions.
5. **padding:** This property is used to **set the distance** between the border of the container (all four directions) and its child widget. We can observe this by seeing the space between the container and the child widget.

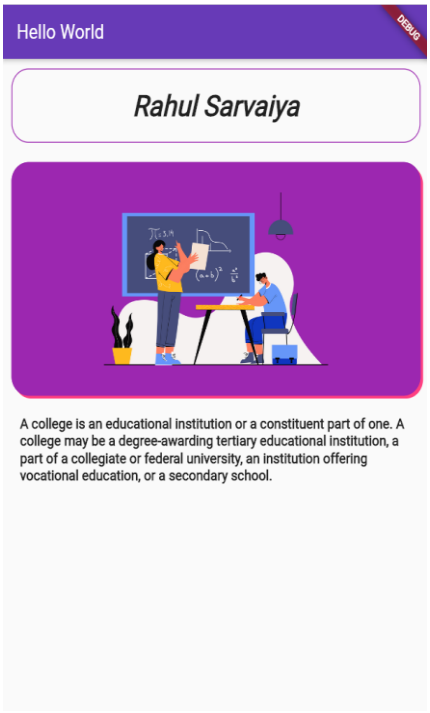
6. **alignment:** This property is used to **set the position** of the child within the container. Flutter allows the user to align its element in various ways such as center, bottom, bottom center, topLeft, centerRight, left, right, and many more.
7. **decoration:** This property allows the developer to **add decoration on the widget**. It decorates or paint the widget behind the child. If we want to decorate or paint in front of a child, we need to use the **forgroundDecoration** parameter.

Code:

```
class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Column(
        children: [
          Container(
            padding: EdgeInsets.all(20),
            margin: EdgeInsets.fromLTRB(10, 10, 10, 10),
            width: 500,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.purple),
              borderRadius: BorderRadius.circular(20),
            ),
            alignment: Alignment.center,
            child: Text("Rahul Sarvaiya",style: TextStyle(fontWeight: FontWeight.bold,fontSize: 30,fontStyle:
FontStyle.italic),),
          ),
          Container(
            padding: EdgeInsets.all(20),
            margin: EdgeInsets.fromLTRB(10, 10, 10, 10),
            width: 500,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.purple),
              borderRadius: BorderRadius.circular(20),
              color: Colors.purple,
              boxShadow: [
                new BoxShadow(color: Colors.pinkAccent, offset: new Offset(3.0, 3.0),),
              ],
            ),
            alignment: Alignment.center,
            child: Image.asset('assets/img/img.png',width: 300),
          ),
          Container(
            padding: EdgeInsets.all(10),
            margin: EdgeInsets.fromLTRB(10, 0, 10, 10),
            width: 500,
            child: Text("A college is an educational institution or a constituent part of one. A college may be a
degree-awarding tertiary educational institution, a part of a collegiate or federal university, an institution
```

```
offering vocational education, or a secondary school.",style: TextStyle(fontWeight: FontWeight.bold,fontSize:
15,),),
          )
        ],
      ),
      // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Output



CONCLUSION: From this experiment we successfully designed flutter UI by including common Widgets like Container, Columns to display text and image.

Name: Pranav Trivedi

Roll No: 61

Batch : I6

Sub : Mad lab

Aim: To connect flutter UI with firebase database.

Theory: Create a Firebase project

After setting up & creating a simple Flutter app, navigate to Firebase Official website

- Click on **Get started.**

- Create a new project by tapping on the **Add project** box.

- Enter the **Project name** & select **Analytics location.**
  Analytics location represents the country/region of your organization and sets the currency for revenue reporting.

- Tap on **Create project**. Following this, Firebase will set up a new project for you. This will take just a minute.

- After the project is created successfully, Firebase will show a prompt saying **Your new project is ready.** Click on **Continue** to complete the flow.

Configure an Android app

- In the **Add an app to get started** section, click on **Android** Icon to add an Android app to Firebase.

- On the next screen, enter your **Android package name** & **App nickname** and click on **Register App**. Your package name is generally the **applicationId** in your app-level **build.gradle** file. If specified, the app nickname will be used throughout the Firebase console to represent this app. Nicknames aren't visible to users.

- Download the **google-services.json** file & place it in your project's **app** root directory. Make sure the config file is not appended with additional characters, like **(2)**.

- The Google services plugin for Gradle loads the **google-services.json** file you just downloaded. Modify your **build.gradle** files to use the plugin.

- Back in the Firebase console-setup workflow, click **Next** to skip the remaining steps.

- Run **flutter packages get**.

Code:

Main.drt

```
// @dart=2.9
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'welcome_screen.dart';
import 'home_screen.dart';
import 'signup_screen.dart';
import 'login_screen.dart';

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

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: 'welcome_screen',
      routes: {
        'welcome_screen': (context) => WelcomeScreen(),
        'registration_screen': (context) => RegistrationScreen(),
        'login_screen': (context) => LoginScreen(),
        'home_screen': (context) => HomeScreen()
      },
    );
```

```
  }
}
```

Welcome_screen.dart

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


class WelcomeScreen extends StatefulWidget {
  @override
  _WelcomeScreenState createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: Colors.white,
        body: Padding(
          padding: EdgeInsets.symmetric(horizontal: 24.0),
          child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: <Widget>[
                ElevatedButton(
                  //colour: Colors.lightBlueAccent,
                  child: Text('Log In'),
                  onPressed: () {
                    Navigator.pushNamed(context, 'login_screen');
                  },
                ),
                ElevatedButton(
                    //colour: Colors.blueAccent,
                    child: Text('Register'),
                    onPressed: () {
                      Navigator.pushNamed(context, 'registration_screen');
                    }),
              ]),
        ));
  }
}
```

login_screen.dart

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

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

//code for designing the UI of our text field where the user writes his
email id or password

const kTextFieldDecoration = InputDecoration(
    hintText: 'Enter a value',
    hintStyle: TextStyle(color: Colors.grey),
    contentPadding: EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),
    border: OutlineInputBorder(
```

```
      borderRadius: BorderRadius.all(Radius.circular(32.0)),
    ),
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlueAccent, width: 1.0),
      borderRadius: BorderRadius.all(Radius.circular(32.0)),
    ),
    focusedBorder: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlueAccent, width: 2.0),
      borderRadius: BorderRadius.all(Radius.circular(32.0)),
    ));

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

final _auth = FirebaseAuth.instance;

class _LoginScreenState extends State<LoginScreen> {
  late String email;
  late String password;
  bool showSpinner = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: ModalProgressHUD(
        inAsyncCall: showSpinner,
        child: Padding(
          padding: EdgeInsets.symmetric(horizontal: 24.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              TextField(
                  keyboardType: TextInputType.emailAddress,
                  textAlign: TextAlign.center,
                  onChanged: (value) {
                    email = value;
                    //Do something with the user input.
                  },
                  decoration: kTextFieldDecoration.copyWith(
                    hintText: 'Enter your email',
                  )),
              SizedBox(
                height: 8.0,
              ),
              TextField(
                  obscureText: true,
                  textAlign: TextAlign.center,
                  onChanged: (value) {
                    password = value;
                    //Do something with the user input.
                  },
                  decoration: kTextFieldDecoration.copyWith(
                      hintText: 'Enter your password.')),
              SizedBox(
                height: 24.0,
              ),
              ElevatedButton(
                  //colour: Colors.lightBlueAccent,
```

```
                  child:Text('Log In'),
                  onPressed: () async {
                    setState(() {
                      showSpinner = true;
                    });
                    try {
                      final user = await _auth.signInWithEmailAndPassword(
                          email: email, password: password);
                      if (user != null) {
                        Navigator.pushNamed(context, 'home_screen');
                      }
                    } catch (e) {
                      print(e);
                    }
                    setState(() {
                      showSpinner = false;
                    });
                  }),
            ],
          ),
        ),
      ),
    );
  }
}
```

signup_screen.dart

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


//code for designing the UI of our text field where the user writes his
email id or password

const kTextFieldDecoration = InputDecoration(
  hintText: 'Enter a value',
  hintStyle: TextStyle(color: Colors.grey),
  contentPadding: EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),
  border: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),
  enabledBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Colors.lightBlueAccent, width: 1.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Colors.lightBlueAccent, width: 2.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),
);

class RegistrationScreen extends StatefulWidget {
  @override
  _RegistrationScreenState createState() => _RegistrationScreenState();
}

class _RegistrationScreenState extends State<RegistrationScreen> {
  final _auth = FirebaseAuth.instance;
```

```
  late String email;
  late String password;
  bool showSpinner = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: ModalProgressHUD(
        inAsyncCall: showSpinner,
        child: Padding(
          padding: EdgeInsets.symmetric(horizontal: 24.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              TextField(
                  keyboardType: TextInputType.emailAddress,
                  textAlign: TextAlign.center,
                  onChanged: (value) {
                    email = value;
                  },
                  decoration: kTextFieldDecoration.copyWith(
                      hintText: 'Enter your email')),
              SizedBox(
                height: 8.0,
              ),
              TextField(
                  obscureText: true,
                  textAlign: TextAlign.center,
                  onChanged: (value) {
                    password = value;
                  },
                  decoration: kTextFieldDecoration.copyWith(
                      hintText: 'Enter your Password')),
              SizedBox(
                height: 24.0,
              ),
              ElevatedButton(
                //colour: Colors.blueAccent,
                child: Text('Register'),
                onPressed: () async {
                  setState(() {
                    showSpinner = true;
                  });
                  try {
                    final newUser = await
_auth.createUserWithEmailAndPassword(
                        email: email, password: password);
                    if (newUser != null) {
                      Navigator.pushNamed(context, 'home_screen');
                    }
                  } catch (e) {
                    print(e);
                  }
                  setState(() {
                    showSpinner = false;
                  });
                },
              )
            ],
          ),
```

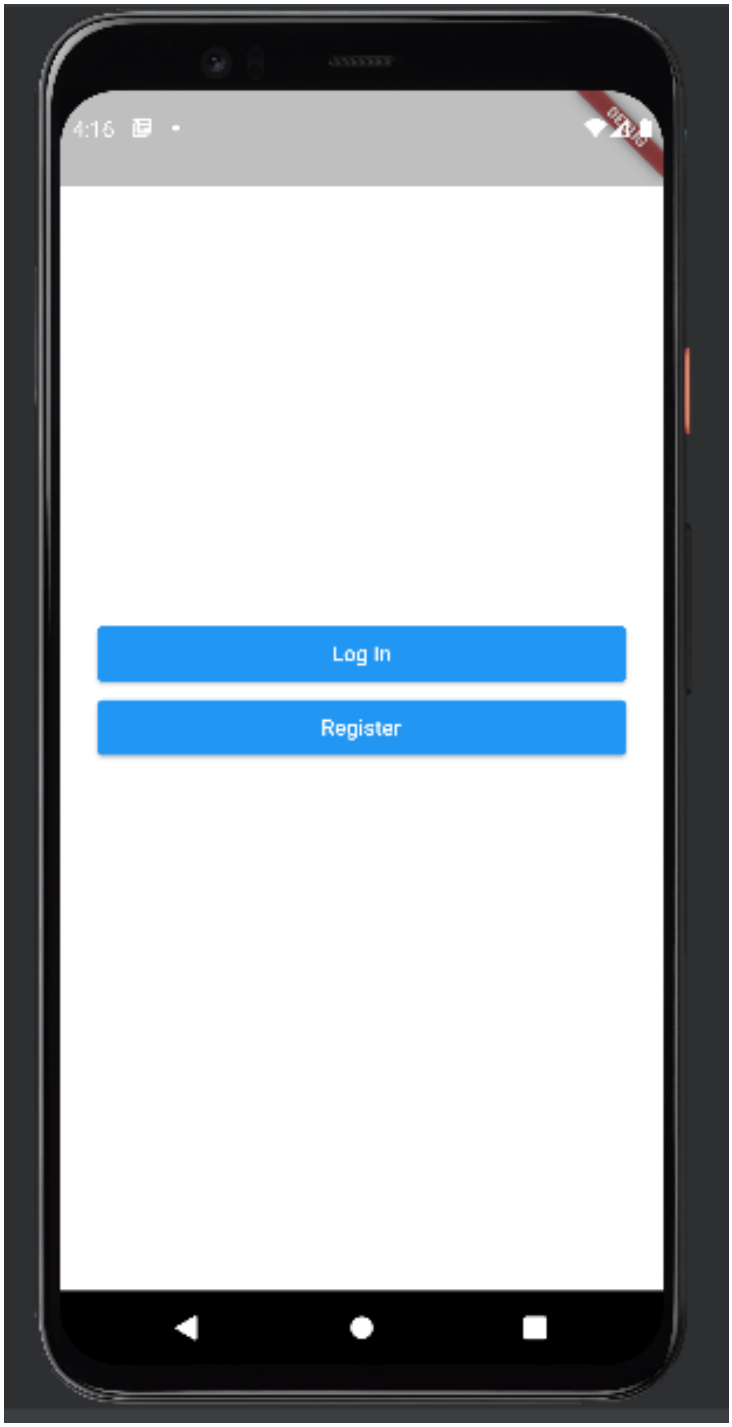
```
        ),
      ),
    );
  }
}
```

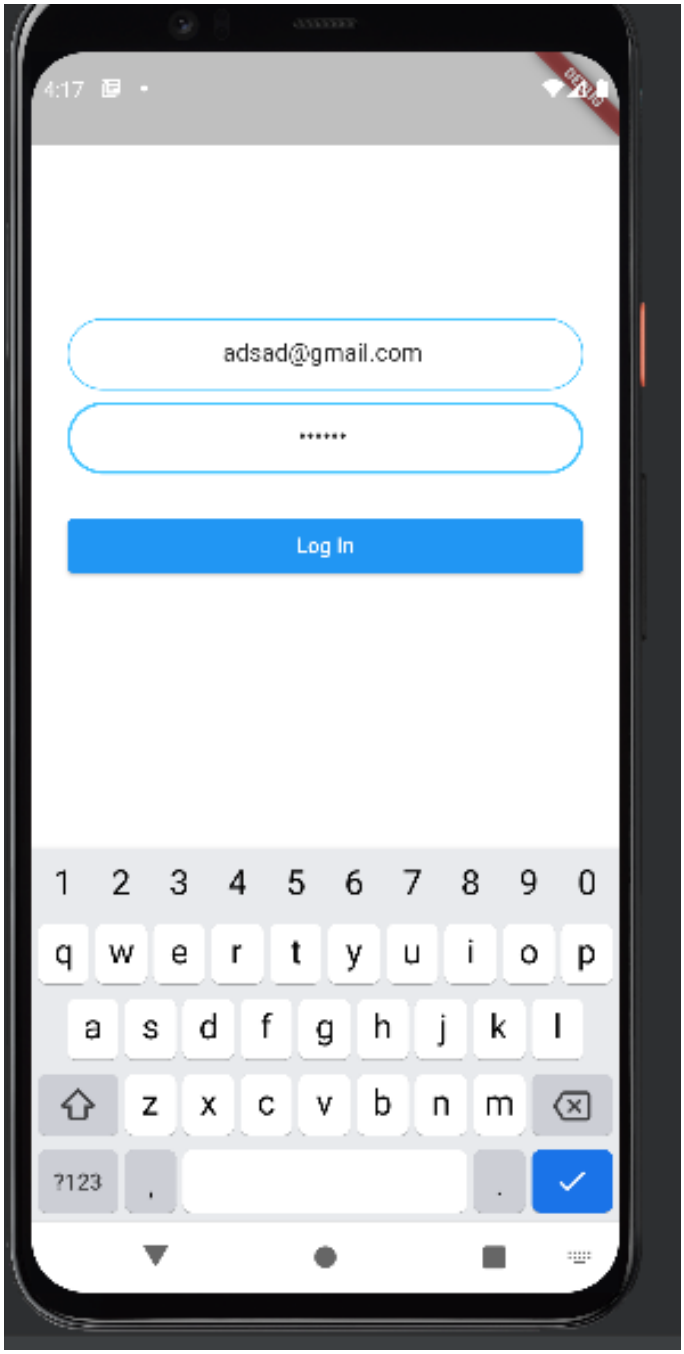
home_screen.dart

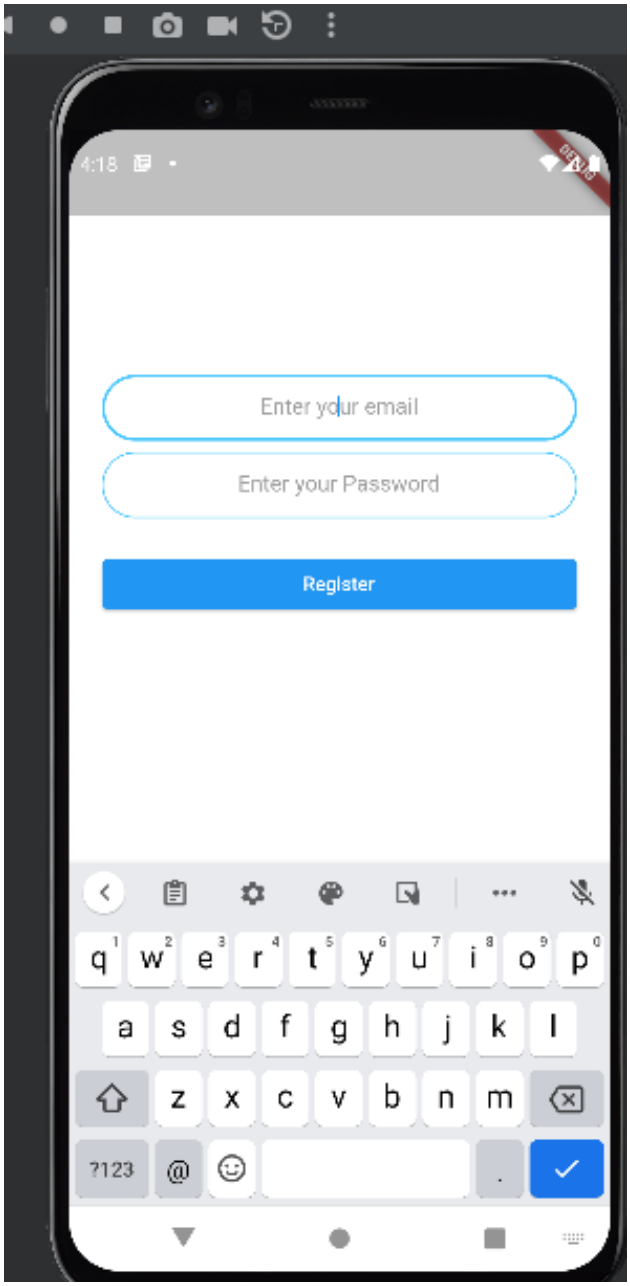
```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';

User loggedinUser= loggedinUser;

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final _auth = FirebaseAuth.instance;

  void initState() {
    super.initState();
    getCurrentUser();
  }

  //using this function you can use the credentials of the user
  void getCurrentUser() async {
    try {
      final user = await _auth.currentUser;
      if (user != null) {
        loggedinUser = user;
      }
    } catch (e) {
      print(e);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: null,
        actions: <Widget>[
          IconButton(
              icon: Icon(Icons.close),
              onPressed: () {
                _auth.signOut();
                Navigator.pop(context);

                //Implement logout functionality
              }),
        ],
        title: Text('Home Page'),
        backgroundColor: Colors.lightBlueAccent,
      ),
      body: Center(
        child: Text(
          "Welcome User",
          style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.bold),
```

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

Output:


4:16
DEBUG
Log In
Register


4:17
adsad@gmail.com
Log In
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123



Conclusion : Hence we have successfully connected flutter UI with firebase database

Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

Experiment No. 09

Aim : To test and deploy production ready Flutter App on Android platform.

Theory :

Spotify is a popular music streaming platform that has revolutionized the way people consume and discover music. A Spotify clone app is a music streaming application that replicates the functionalities and features of Spotify.

The Spotify clone app allows users to browse and search for their favorite music tracks, albums, and playlists. Users can also create their own playlists, follow other users' playlists, and share their favorite tracks with friends and family.

The app also offers personalized recommendations based on the user's listening history, allowing them to discover new artists and genres.

Additionally, the Spotify clone app includes features such as music playback controls, song lyrics, and the ability to save music for offline listening.

Overall, a Spotify clone app can provide music lovers with a seamless and enjoyable music streaming experience, allowing them to access their favorite tracks anytime, anywhere.

Code :

```
import …
Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
```

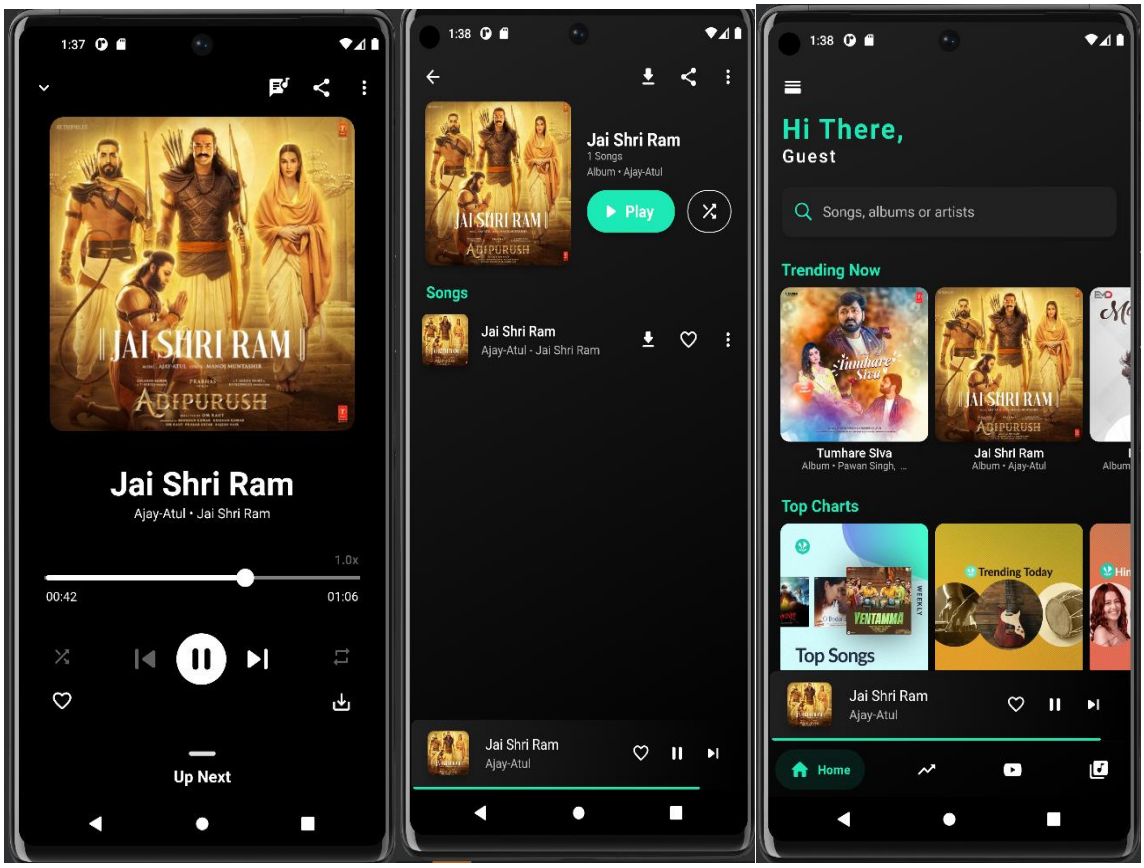
```
 Paint.enableDithering = true;

 if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
   await Hive.initFlutter('BlackHole');
 } else {
   await Hive.initFlutter();
 }
 await openHiveBox('settings');
 await openHiveBox('downloads');
 await openHiveBox('stats');
 await openHiveBox('Favorite Songs');
 await openHiveBox('cache', limit: true);
 await openHiveBox('ytlinkcache', limit: true);
 if (Platform.isAndroid) {
   setOptimalDisplayMode();
 }
 await startService();
 runApp(MyApp());
}

Future<void> setOptimalDisplayMode() async {
 await FlutterDisplayMode.setHighRefreshRate();
}

Future<void> startService() async {
 await initializeLogging();
 final AudioPlayerHandler audioHandler = await AudioService.init(
   builder: () => AudioPlayerHandlerImpl(),
   config: AudioServiceConfig(
     androidNotificationChannelId: 'com.shadow.blackhole.channel.audio',
     androidNotificationChannelName: 'BlackHole',
     androidNotificationIcon: 'drawable/ic_stat_music_note',
     androidShowNotificationBadge: true,
     androidStopForegroundOnPause: false,
     // Hive.box('settings').get('stopServiceOnPause', defaultValue: true)
as bool,
     notificationColor: Colors.grey[900],
   ),
 );
 GetIt.I.registerSingleton<AudioPlayerHandler>(audioHandler);
 GetIt.I.registerSingleton<MyTheme>(MyTheme());
}

Future<void> openHiveBox(String boxName, {bool limit = false}) async {
 final box = await Hive.openBox(boxName).onError((error, stackTrace) async
{
   Logger.root.severe('Failed to open $boxName Box', error, stackTrace);
   final Directory dir = await getApplicationDocumentsDirectory();
   final String dirPath = dir.path;
   File dbFile = File('$dirPath/$boxName.hive');
   File lockFile = File('$dirPath/$boxName.lock');
   if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
     dbFile = File('$dirPath/BlackHole/$boxName.hive');
     lockFile = File('$dirPath/BlackHole/$boxName.lock');
   }
   await dbFile.delete();
   await lockFile.delete();
   await Hive.openBox(boxName);
   throw 'Failed to open $boxName Box\nError: $error';
 });
 // clear box if it grows large
```

```
 if (limit && box.length > 500) {
   box.clear();
 }
}

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

 static _MyAppState of(BuildContext context) =>
     context.findAncestorStateOfType<_MyAppState>()!;
}

class _MyAppState extends State<MyApp> {
 Locale _locale = const Locale('en', '');
 late StreamSubscription _intentTextStreamSubscription;
 late StreamSubscription _intentDataStreamSubscription;
 final GlobalKey<NavigatorState> navigatorKey =
GlobalKey<NavigatorState>();

 @override
 void dispose() {
   _intentTextStreamSubscription.cancel();
   _intentDataStreamSubscription.cancel();
   super.dispose();
 }

 @override
 void initState() {
   super.initState();
   final String systemLangCode = Platform.localeName.substring(0, 2);
   if (ConstantCodes.languageCodes.values.contains(systemLangCode)) {
     _locale = Locale(systemLangCode);
   } else {
     final String lang =
         Hive.box('settings').get('lang', defaultValue: 'English') as
String;
     _locale = Locale(ConstantCodes.languageCodes[lang] ?? 'en');
   }

   AppTheme.currentTheme.addListener(() {
     setState(() {});
   });

   // For sharing or opening urls/text coming from outside the app while
the app is in the memory
   _intentTextStreamSubscription =
ReceiveSharingIntent.getTextStream().listen(
     (String value) {
       Logger.root.info('Received intent on stream: $value');
       handleSharedText(value, navigatorKey);
     },
     onError: (err) {
       Logger.root.severe('ERROR in getTextStream', err);
     },
   );


   // For sharing files coming from outside the app while the app is in the
memory
   _intentDataStreamSubscription =
```

```
      ReceiveSharingIntent.getMediaStream().listen(
    (List<SharedMediaFile> value) {
      if (value.isNotEmpty) {
        for (final file in value) {
          if (file.path.endsWith('.json')) {
            final List playlistNames = Hive.box('settings')
                    .get('playlistNames')
                    ?.toList() as List? ??
                ['Favorite Songs'];
            importFilePlaylist(
              null,
              playlistNames,
              path: file.path,
              pickFile: false,
            ).then(
              (value) =>
navigatorKey.currentState?.pushNamed('/playlists'),
            );
          }
        }
      }
    },
    onError: (err) {
      Logger.root.severe('ERROR in getDataStream', err);
    },
  );

   void setLocale(Locale value) {
  setState(() {
    _locale = value;
  });
 }

 Widget initialFuntion() {
  return Hive.box('settings').get('userId') != null
      ? HomePage()
      : AuthScreen();
 }

 @override
 Widget build(BuildContext context) {
  SystemChrome.setSystemUIOverlayStyle(
    SystemUiOverlayStyle(
      statusBarColor: Colors.transparent,
      systemNavigationBarColor: AppTheme.themeMode == ThemeMode.dark
          ? Colors.black38
          : Colors.white,
      statusBarIconBrightness: AppTheme.themeMode == ThemeMode.dark
          ? Brightness.light
          : Brightness.dark,
      systemNavigationBarIconBrightness: AppTheme.themeMode ==
ThemeMode.dark
          ? Brightness.light
          : Brightness.dark,
    ),
  );
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
    DeviceOrientation.landscapeLeft,
    DeviceOrientation.landscapeRight,
```

```
    ]);

    return MaterialApp(
      title: 'BlackHole',
      restorationScopeId: 'blackhole',
      debugShowCheckedModeBanner: false,
      themeMode: AppTheme.themeMode,
      theme: AppTheme.lightTheme(
        context: context,
      ),
      darkTheme: AppTheme.darkTheme(
        context: context,
      ),
      locale: _locale,
      localizationsDelegates: const [
        AppLocalizations.delegate,
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      supportedLocales: ConstantCodes.languageCodes.entries
          .map((languageCode) => Locale(languageCode.value, ''))
          .toList(),
      routes: {
        '/': (context) => initialFuntion(),
        '/pref': (context) => const PrefScreen(),
        '/setting': (context) => const SettingPage(),
        '/about': (context) => AboutScreen(),
        '/playlists': (context) => PlaylistScreen(),
        '/nowplaying': (context) => NowPlaying(),
        '/recent': (context) => RecentlyPlayed(),
        '/downloads': (context) => const Downloads(),
        '/stats': (context) => const Stats(),
      },
      navigatorKey: navigatorKey,
      onGenerateRoute: (RouteSettings settings) {
        if (settings.name == '/player') {
          return PageRouteBuilder(
            opaque: false,
            pageBuilder: (_, __, ___) => const PlayScreen(),
          );
        }
        return HandleRoute.handleRoute(settings.name);
      },
    );
  }
}
```

Output :-



Conclusion : Hence we have successfully tested and deployed production ready Flutter App on Android platform.

Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

Experiment No. 10

Aim : To create a responsive User Interface for Ecommerce applications.

Theory :

Benefits of responsive web design for ecommerce

Of course, building a website from scratch and using responsive web design right from the get-go introduces a number of benefits. Here are some of them to expect:

Optimal User Experience

Research shows that 90% of smartphone users have their phones within reach at all times. This makes it vital for websites to be accessible on mobile devices.

By utilizing a responsive design, the website will automatically adjust to any device's screen size and resolution.

Improved Brand Reputation

Customers' expectations have changed – now, a business's website should look great on any device. If that's not the case, it might negatively impact a brand's reputation. A responsive website design contributes to a pleasant user experience, meeting or exceeding customers' expectations.

Flexible Site Management

Maintaining two versions of your website – desktop and mobile – means extra effort to manage two different sites, running the risk of creating duplicate content.

Opting for a responsive web design instead will take up less time and resources, as you'll only have to manage a single website.

Higher Conversion Rates

According to a study, 61% of customers are more likely to make purchases on mobile-friendly websites. Thus, fostering a robust mobile experience by implementing responsive web design can increase conversion rates substantially.

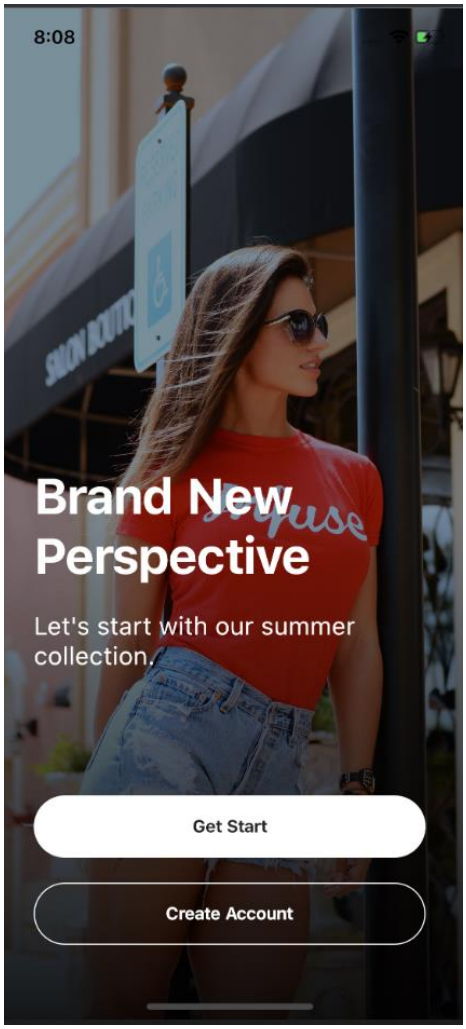
In addition, due to a seamless shopping experience, your eCommerce store will experience reduced abandoned cart and bounce rates.

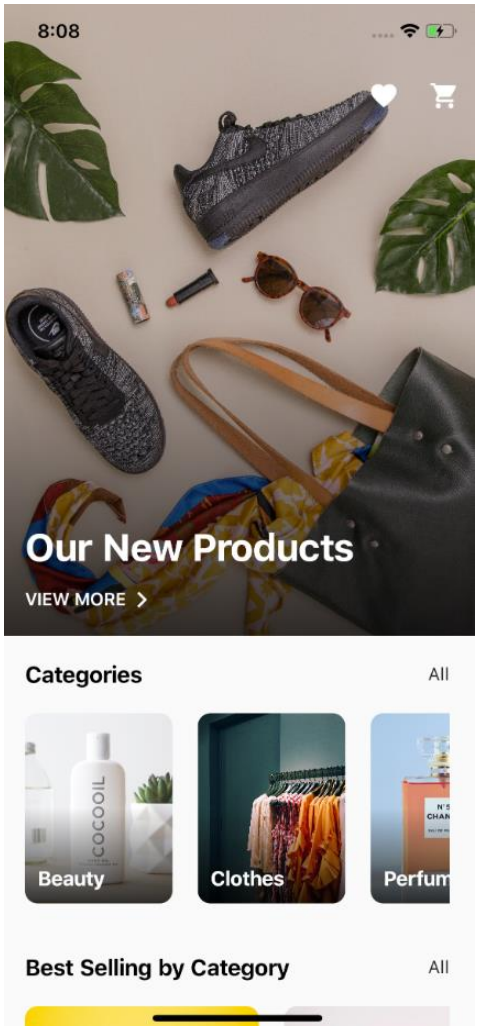
Code :

```
import 'package:day16_shopping/Pages/home_page.dart';
import 'package:day16_shopping/Pages/not_found.dart';
import 'package:day16_shopping/Pages/welcome.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import 'app_state.dart';

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

class _AppState extends State<App> {
  Map<String, WidgetBuilder> _routes;

  AppState _state;

  @override
  void initState() {
    super.initState();
    _state = AppState();

    _routes = {
      WelcomePage.routeName: (_) => WelcomePage(),
      HomePage.routeName: (_) => HomePage(),
    };
  }
```

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

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        theme: ThemeData(fontFamily: 'Gotham'),
        title: 'Shop',
        onGenerateRoute: (settings) {

          if (_routes.containsKey(settings.name)) {
            final builder = _routes[settings.name];
            return inPageRoute(
                builder(context), RouteSettings(name: settings.name));
          }

          return inPageRoute(NotFound());
        },
        // Mirgration to Navigator 2.0
        // merge request: https://github.com/flutter/flutter/pull/51435
        // migration guide: https://flutter.dev/docs/release/breaking-
changes/route-navigator-refactoring
        onGenerateInitialRoutes: (initialRoute) {
          return <Route>[
            inPageRoute(
                WelcomePage(), RouteSettings(name: WelcomePage.routeName))
          ];
        }

        // supportedLocales: [//TODO DEFINE SUPOORTED LANGSSSSS],
      ),
      providers: [
        inProvider<AppState>(_state),
      ],
    );
  }
}

bool neverNotify(_, __) => false;

Provider<T> inProvider<T>(T value) =>
    Provider<T>.value(value: value, updateShouldNotify: neverNotify);

MaterialPageRoute inPageRoute(Widget child, [RouteSettings settings]) =>
    MaterialPageRoute(builder: (context) => child, settings: settings);
```

Output :-





Conclusion : Hence we have successfully created a responsive User Interface for Ecommerce application.

Name : Pranav Trivedi

Roll No. : 61

Batch : I6

Sub : MAD Lab

Experiment No. 11

Aim : To study and implement deployment of your app to GitHub Pages.

Theory :

Step 1: Creating a flutter web project.
To create a flutter web project, you have to be in one of the following flutter channels : beta, dev or master. Change your channel and upgrade it, if you were in stable and then proceed to the next step.
Enable support for flutter-web in terminal/command-prompt:
$ flutter config --enable-web
Now, create your flutter project like you usually do and it will have web support. If you want to add web support to an existing project you can use the following command inside the directory:
$ flutter create .

After creating your flutter project, you can check if you have the web directory. It shows that your project supports flutter-web.
The sources for the examples are at the end of this page.
Now, you can also publish it on GitHub.
Step 2: Making your flutter-web build.
You can make a release build for the flutter-web using the command:
$ flutter build web --release
Now, you will have a new directory named build and you will find your web build in it, like this.

Now if you look inside that web folder, you will find the build files. As the dart code trans compiles into javascript code with HTML and CSS, the starting point of the build naturally, is the index.html .
Step 3: Publishing the build to GitHub.
You can now start a new repository or clone a repository you already have in a new folder/location .

This is how the cloned repository will look like (mine is just empty).
Then rename it to web and paste it into flutter-project/build/ directory, before really building the web build.

Now, that you have done everything, you can edit your flutter code just like you want, anytime from the flutter-project and build it. Then the changes can be committed and pushed from flutter-project/build/web . This reduces our worry to handle the build files to

host and maintain two different repositories, one for the flutter project and the other exclusively for hosting.
tip: You can add build directory to your .gitignore file in your flutter project folder to avoid confusion and repetition.
Step 4: Hosting it on GitHub Pages.
After you have successfully pushed the build files you got from the flutter project, go to the repository page. Then, navigate to the settings, you will find a title “GitHub Pages”. Select your source as main branch and save.

Code :

```
import …
Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
 Paint.enableDithering = true;

 if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
   await Hive.initFlutter('BlackHole');
 } else {
   await Hive.initFlutter();
 }
 await openHiveBox('settings');
 await openHiveBox('downloads');
 await openHiveBox('stats');
 await openHiveBox('Favorite Songs');
 await openHiveBox('cache', limit: true);
 await openHiveBox('ytlinkcache', limit: true);
 if (Platform.isAndroid) {
   setOptimalDisplayMode();
 }
 await startService();
 runApp(MyApp());
}

Future<void> setOptimalDisplayMode() async {
 await FlutterDisplayMode.setHighRefreshRate();
}

Future<void> startService() async {
 await initializeLogging();
 final AudioPlayerHandler audioHandler = await AudioService.init(
   builder: () => AudioPlayerHandlerImpl(),
   config: AudioServiceConfig(
     androidNotificationChannelId: 'com.shadow.blackhole.channel.audio',
     androidNotificationChannelName: 'BlackHole',
     androidNotificationIcon: 'drawable/ic_stat_music_note',
     androidShowNotificationBadge: true,
     androidStopForegroundOnPause: false,
     // Hive.box('settings').get('stopServiceOnPause', defaultValue: true)
as bool,
     notificationColor: Colors.grey[900],
   ),
 );
 GetIt.I.registerSingleton<AudioPlayerHandler>(audioHandler);
 GetIt.I.registerSingleton<MyTheme>(MyTheme());
}
```

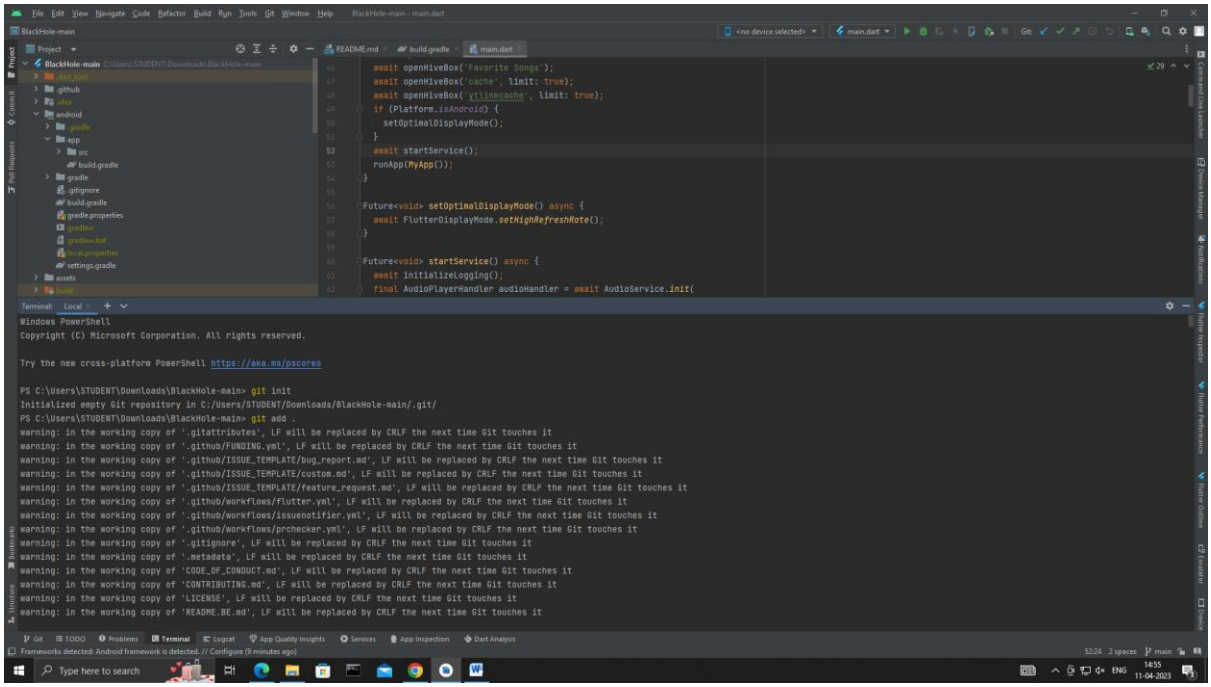
```
Future<void> openHiveBox(String boxName, {bool limit = false}) async {
 final box = await Hive.openBox(boxName).onError((error, stackTrace) async
{
   Logger.root.severe('Failed to open $boxName Box', error, stackTrace);
   final Directory dir = await getApplicationDocumentsDirectory();
   final String dirPath = dir.path;
   File dbFile = File('$dirPath/$boxName.hive');
   File lockFile = File('$dirPath/$boxName.lock');
   if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
     dbFile = File('$dirPath/BlackHole/$boxName.hive');
     lockFile = File('$dirPath/BlackHole/$boxName.lock');
   }
   await dbFile.delete();
   await lockFile.delete();
   await Hive.openBox(boxName);
   throw 'Failed to open $boxName Box\nError: $error';
 });
 // clear box if it grows large
 if (limit && box.length > 500) {
   box.clear();
 }
}

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

 static _MyAppState of(BuildContext context) =>
     context.findAncestorStateOfType<_MyAppState>()!;
}

class _MyAppState extends State<MyApp> {
 Locale _locale = const Locale('en', '');
 late StreamSubscription _intentTextStreamSubscription;
 late StreamSubscription _intentDataStreamSubscription;
 final GlobalKey<NavigatorState> navigatorKey =
GlobalKey<NavigatorState>();

 @override
 void dispose() {
   _intentTextStreamSubscription.cancel();
   _intentDataStreamSubscription.cancel();
   super.dispose();
 }

 @override
 void initState() {
   super.initState();
   final String systemLangCode = Platform.localeName.substring(0, 2);
   if (ConstantCodes.languageCodes.values.contains(systemLangCode)) {
     _locale = Locale(systemLangCode);
   } else {
     final String lang =
         Hive.box('settings').get('lang', defaultValue: 'English') as
String;
     _locale = Locale(ConstantCodes.languageCodes[lang] ?? 'en');
   }

   AppTheme.currentTheme.addListener(() {
     setState(() {});
   });
```

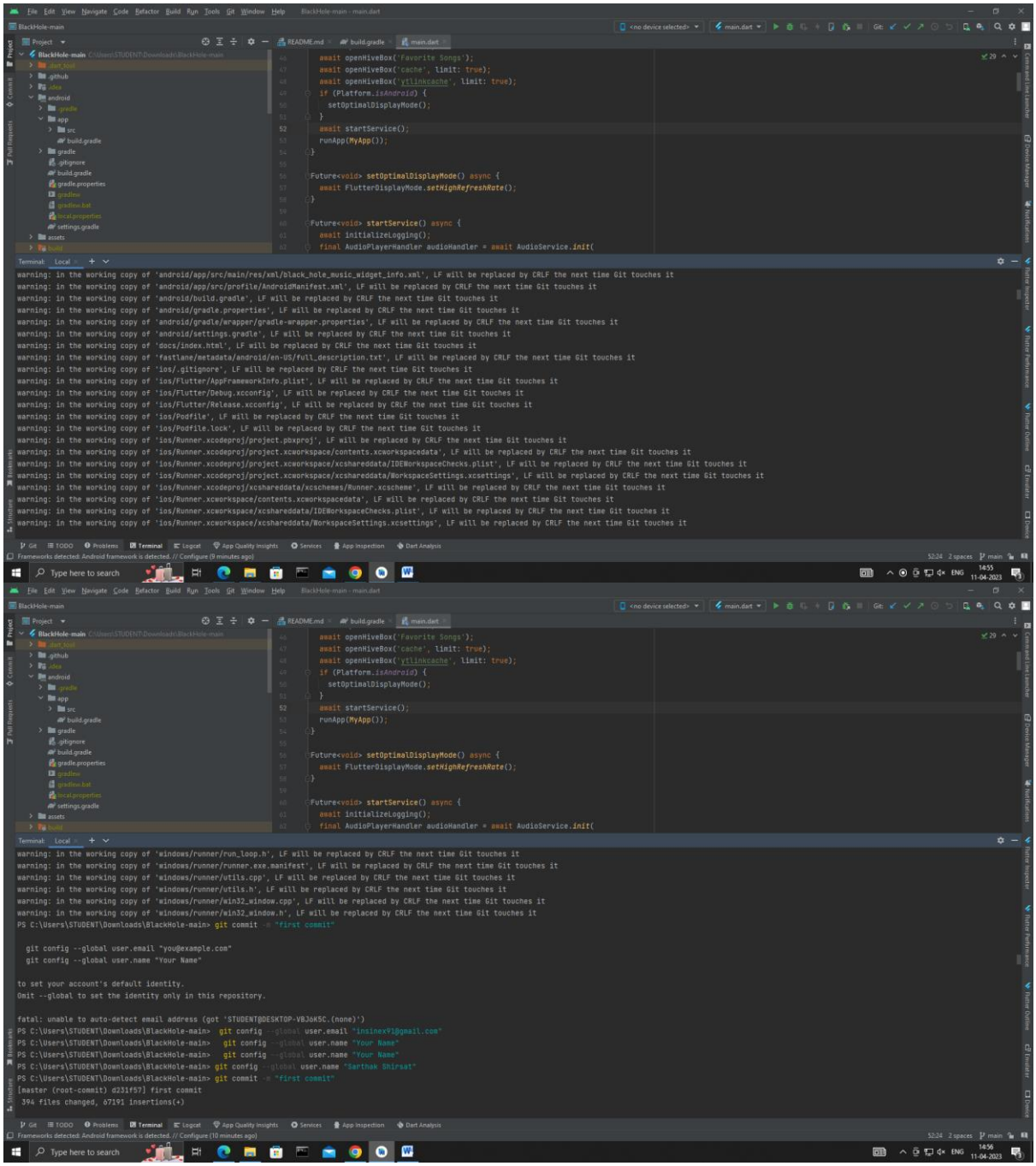
```
   // For sharing or opening urls/text coming from outside the app while
the app is in the memory
   _intentTextStreamSubscription =
ReceiveSharingIntent.getTextStream().listen(
     (String value) {
       Logger.root.info('Received intent on stream: $value');
       handleSharedText(value, navigatorKey);
     },
     onError: (err) {
       Logger.root.severe('ERROR in getTextStream', err);
     },
   );


   // For sharing files coming from outside the app while the app is in the
memory
   _intentDataStreamSubscription =
       ReceiveSharingIntent.getMediaStream().listen(
     (List<SharedMediaFile> value) {
       if (value.isNotEmpty) {
         for (final file in value) {
           if (file.path.endsWith('.json')) {
             final List playlistNames = Hive.box('settings')
                     .get('playlistNames')
                     ?.toList() as List? ??
                 ['Favorite Songs'];
             importFilePlaylist(
               null,
               playlistNames,
               path: file.path,
               pickFile: false,
             ).then(
               (value) =>
navigatorKey.currentState?.pushNamed('/playlists'),
             );
           }
         }
       }
     },
     onError: (err) {
       Logger.root.severe('ERROR in getDataStream', err);
     },
   );

    void setLocale(Locale value) {
   setState(() {
     _locale = value;
   });
 }

 Widget initialFuntion() {
   return Hive.box('settings').get('userId') != null
       ? HomePage()
       : AuthScreen();
 }

 @override
 Widget build(BuildContext context) {
   SystemChrome.setSystemUIOverlayStyle(
     SystemUiOverlayStyle(
```

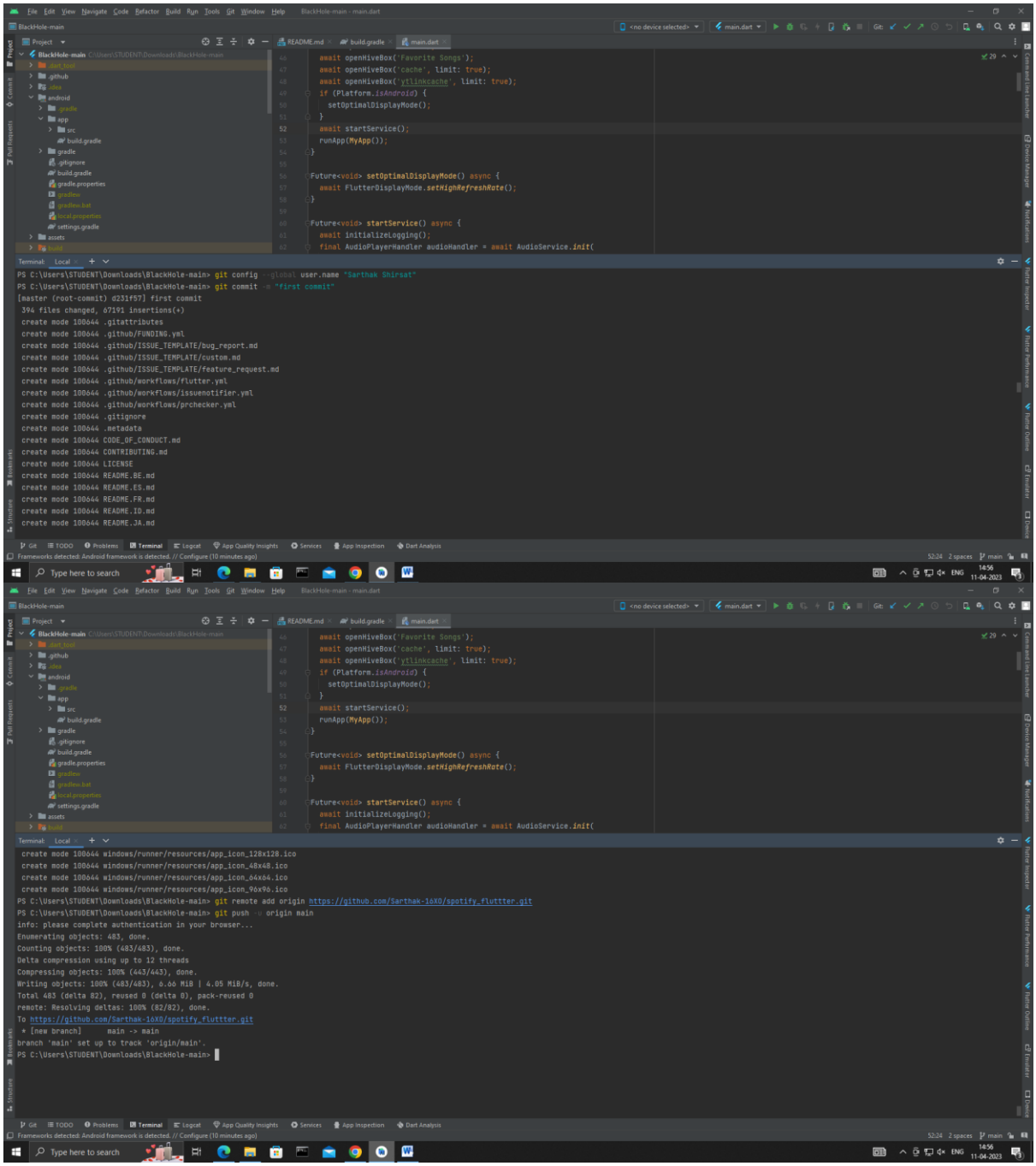
```
      statusBarColor: Colors.transparent,
      systemNavigationBarColor: AppTheme.themeMode == ThemeMode.dark
          ? Colors.black38
          : Colors.white,
      statusBarIconBrightness: AppTheme.themeMode == ThemeMode.dark
          ? Brightness.light
          : Brightness.dark,
      systemNavigationBarIconBrightness: AppTheme.themeMode ==
ThemeMode.dark
          ? Brightness.light
          : Brightness.dark,
    ),
  );
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
    DeviceOrientation.landscapeLeft,
    DeviceOrientation.landscapeRight,
  ]);

  return MaterialApp(
    title: 'BlackHole',
    restorationScopeId: 'blackhole',
    debugShowCheckedModeBanner: false,
    themeMode: AppTheme.themeMode,
    theme: AppTheme.lightTheme(
      context: context,
    ),
    darkTheme: AppTheme.darkTheme(
      context: context,
    ),
    locale: _locale,
    localizationsDelegates: const [
      AppLocalizations.delegate,
      GlobalMaterialLocalizations.delegate,
      GlobalWidgetsLocalizations.delegate,
      GlobalCupertinoLocalizations.delegate,
    ],
    supportedLocales: ConstantCodes.languageCodes.entries
        .map((languageCode) => Locale(languageCode.value, ''))
        .toList(),
    routes: {
      '/': (context) => initialFuntion(),
      '/pref': (context) => const PrefScreen(),
      '/setting': (context) => const SettingPage(),
      '/about': (context) => AboutScreen(),
      '/playlists': (context) => PlaylistScreen(),
      '/nowplaying': (context) => NowPlaying(),
      '/recent': (context) => RecentlyPlayed(),
      '/downloads': (context) => const Downloads(),
      '/stats': (context) => const Stats(),
    },
    navigatorKey: navigatorKey,
    onGenerateRoute: (RouteSettings settings) {
      if (settings.name == '/player') {
        return PageRouteBuilder(
          opaque: false,
          pageBuilder: (_, __, ___) => const PlayScreen(),
        );
      }
      return HandleRoute.handleRoute(settings.name);
```

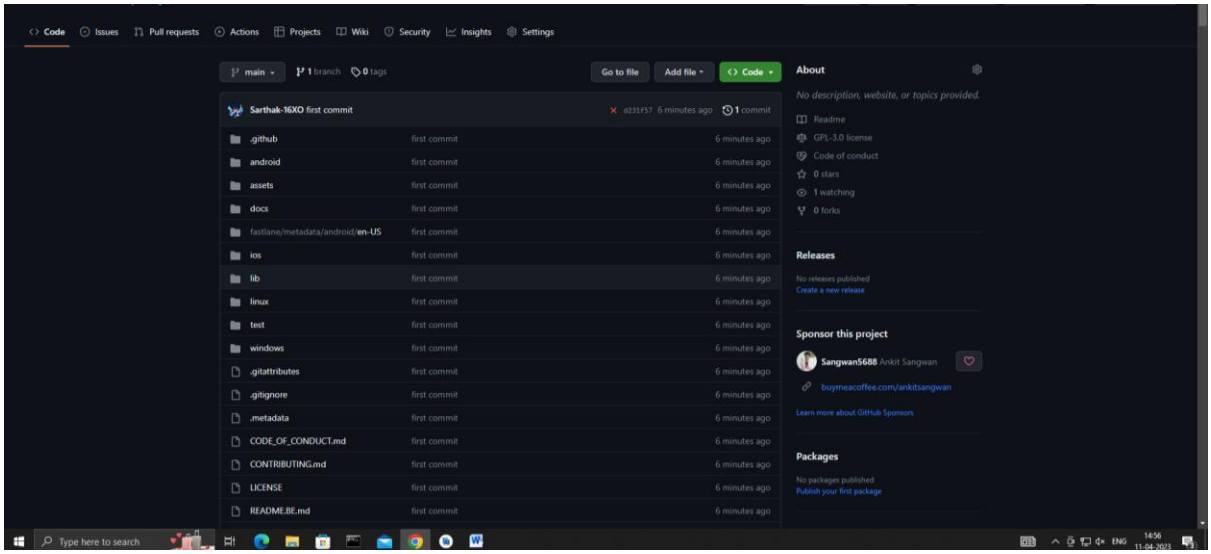
```
        },
      );
    }
}
```

Output :-


Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\STUDENT\Downloads\BlackHole-main> git init
Initialized empty Git repository in C:/Users/STUDENT/Downloads/BlackHole-main/.git/
PS C:\Users\STUDENT\Downloads\BlackHole-main> git add .
warning: in the working copy of '.gitattributes', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/FUNDING.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/ISSUE_TEMPLATE/bug_report.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/ISSUE_TEMPLATE/custom.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/ISSUE_TEMPLATE/feature_request.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/workflows/flutter.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/workflows/issuenotifier.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.github/workflows/prchecker.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.metadata', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'CODE_OF_CONDUCT.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'CONTRIBUTING.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'LICENSE', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.BE.md', LF will be replaced by CRLF the next time Git touches it


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
BlackHole-main - main.dart
BlackHole-main
README.md
build.gradle
main.dart
Project
BlackHole-main
.dart_tool
.github
.idea
android
.gradle
app
src
build.gradle
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
assets
build
await openHiveBox('Favorite Songs');
await openHiveBox('cache', limit: true);
await openHiveBox('ytlinkcache', limit: true);
if (Platform.isAndroid) {
setOptimalDisplayMode();
}
await startService();
runApp(MyApp());
}
Future<void> setOptimalDisplayMode() async {
await FlutterDisplayMode.setHighRefreshRate();
}
Future<void> startService() async {
await initializeLogging();
final AudioPlayerHandler audioHandler = await AudioService.init(
Terminal: Local
warning: in the working copy of 'android/app/src/main/res/xml/black_hole_music_widget_info.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'android/app/src/profile/AndroidManifest.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'android/build.gradle', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'android/gradle.properties', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'android/gradle/wrapper/gradle-wrapper.properties', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'android/settings.gradle', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'docs/index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'fastlane/metadata/android/en-US/full_description.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Flutter/AppFrameworkInfo.plist', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Flutter/Debug.xcconfig', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Flutter/Release.xcconfig', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Podfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Podfile.lock', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcodeproj/project.pbxproj', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcodeproj/project.xcworkspace/contents.xcworkspacedata', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcodeproj/project.xcworkspace/xcshareddata/IDEWorkspaceChecks.plist', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcodeproj/project.xcworkspace/xcshareddata/WorkspaceSettings.xcsettings', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcodeproj/xcshareddata/xcschemes/Runner.xcscheme', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcworkspace/contents.xcworkspacedata', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcworkspace/xcshareddata/IDEWorkspaceChecks.plist', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ios/Runner.xcworkspace/xcshareddata/WorkspaceSettings.xcsettings', LF will be replaced by CRLF the next time Git touches it
Git TODO Problems Terminal Logcat App Quality Insights Services App Inspection Dart Analysis
Frameworks detected: Android framework is detected. // Configure (9 minutes ago)
52:24 2 spaces main
Type here to search
ENG
14:55
11-04-2023
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
BlackHole-main - main.dart
BlackHole-main
README.md
build.gradle
main.dart
Project
BlackHole-main
.dart_tool
.github
.idea
android
.gradle
app
src
build.gradle
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
assets
build
await openHiveBox('Favorite Songs');
await openHiveBox('cache', limit: true);
await openHiveBox('ytlinkcache', limit: true);
if (Platform.isAndroid) {
setOptimalDisplayMode();
}
await startService();
runApp(MyApp());
}
Future<void> setOptimalDisplayMode() async {
await FlutterDisplayMode.setHighRefreshRate();
}
Future<void> startService() async {
await initializeLogging();
final AudioPlayerHandler audioHandler = await AudioService.init(
Terminal: Local
warning: in the working copy of 'windows/runner/run_loop.h', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/runner/runner.exe.manifest', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/runner/utils.cpp', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/runner/utils.h', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/runner/win32_window.cpp', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/runner/win32_window.h', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\STUDENT\Downloads\BlackHole-main> git commit -m "first commit"
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
to set your account's default identity.
Omit --global to set the identity only in this repository.
fatal: unable to auto-detect email address (got 'STUDENT@DESKTOP-VBJ0K5C.(none)')
PS C:\Users\STUDENT\Downloads\BlackHole-main> git config --global user.email "insinex91@gmail.com"
PS C:\Users\STUDENT\Downloads\BlackHole-main> git config --global user.name "Your Name"
PS C:\Users\STUDENT\Downloads\BlackHole-main> git config --global user.name "Your Name"
PS C:\Users\STUDENT\Downloads\BlackHole-main> git config --global user.name "Sarthak Shirsat"
PS C:\Users\STUDENT\Downloads\BlackHole-main> git commit -m "first commit"
[master (root-commit) d231f57] first commit
394 files changed, 67191 insertions(+)
Git TODO Problems Terminal Logcat App Quality Insights Services App Inspection Dart Analysis
Frameworks detected: Android framework is detected. // Configure (10 minutes ago)
52:24 2 spaces main
Type here to search
ENG
14:56
11-04-2023


BlackHole-main - main.dart
BlackHole-main
Project
README.md
build.gradle
main.dart
<no device selected>
main.dart
Git:
BlackHole-main C:\Users\STUDENT\Downloads\BlackHole-main
.dart_tool
.github
.idea
android
.gradle
app
src
build.gradle
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
assets
build
await openHiveBox('Favorite Songs');
await openHiveBox('cache', limit: true);
await openHiveBox('ytlinkcache', limit: true);
if (Platform.isAndroid) {
setOptimalDisplayMode();
}
await startService();
runApp(MyApp());
}
Future<void> setOptimalDisplayMode() async {
await FlutterDisplayMode.setHighRefreshRate();
}
Future<void> startService() async {
await initializeLogging();
final AudioPlayerHandler audioHandler = await AudioService.init(
Terminal: Local
PS C:\Users\STUDENT\Downloads\BlackHole-main> git config --global user.name "Sarthak Shirsat"
PS C:\Users\STUDENT\Downloads\BlackHole-main> git commit -m "first commit"
[master (root-commit) d231f57] first commit
394 files changed, 67191 insertions(+)
create mode 100644 .gitattributes
create mode 100644 .github/FUNDING.yml
create mode 100644 .github/ISSUE_TEMPLATE/bug_report.md
create mode 100644 .github/ISSUE_TEMPLATE/custom.md
create mode 100644 .github/ISSUE_TEMPLATE/feature_request.md
create mode 100644 .github/workflows/flutter.yml
create mode 100644 .github/workflows/issuenotifier.yml
create mode 100644 .github/workflows/prchecker.yml
create mode 100644 .gitignore
create mode 100644 .metadata
create mode 100644 CODE_OF_CONDUCT.md
create mode 100644 CONTRIBUTING.md
create mode 100644 LICENSE
create mode 100644 README.BE.md
create mode 100644 README.ES.md
create mode 100644 README.FR.md
create mode 100644 README.ID.md
create mode 100644 README.JA.md
Git
TODO
Problems
Terminal
Logcat
App Quality Insights
Services
App Inspection
Dart Analysis
Frameworks detected: Android framework is detected. // Configure (10 minutes ago)
52:24
2 spaces
main
Type here to search
ENG
14:56
11-04-2023
create mode 100644 windows/runner/resources/app_icon_128x128.ico
create mode 100644 windows/runner/resources/app_icon_48x48.ico
create mode 100644 windows/runner/resources/app_icon_64x64.ico
create mode 100644 windows/runner/resources/app_icon_96x96.ico
PS C:\Users\STUDENT\Downloads\BlackHole-main> git remote add origin https://github.com/Sarthak-16X0/spotify_fluttter.git
PS C:\Users\STUDENT\Downloads\BlackHole-main> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 483, done.
Counting objects: 100% (483/483), done.
Delta compression using up to 12 threads
Compressing objects: 100% (443/443), done.
Writing objects: 100% (483/483), 6.66 MiB | 4.05 MiB/s, done.
Total 483 (delta 82), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (82/82), done.
To https://github.com/Sarthak-16X0/spotify_fluttter.git
* [new branch] main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\STUDENT\Downloads\BlackHole-main>



Conclusion : Hence we have successfully tested and deployed production ready Flutter App on Android platform.

Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

EXPERIMENT NO. 6

AIM: To include bar charts in Flutter application

THEORY:

A chart is a graphical representation of data where data is represented by a symbol such as a line, bar, pie, etc. In Flutter, the chart behaves the same as a normal chart. We use a chart in Flutter to represent the data in a graphical way that allows the user to understand them in a simple manner.

Flutter supports mainly three types of charts, and each chart comes with several configuration options. The following are the chart used in Flutter application:

Line Chart:

A line chart is a graph that uses lines for connecting individual data points. It displays the information in a series of data points. It is mainly used to track changes over a short and long period of time.

```
LineChart(
  LineChartData(
    // write your logic
  ),
);
```

Bar Chart:

It is a graph that represents the categorical data with rectangular bars. It can be horizontal or vertical.

```
BarChart(
  BarChartData(
    // write your logic
  ),
);
```

Pie or Donut Chart:

It is a graph that displays the information in a circular graph. In this graph, the circle is divided into sectors, and each shows the percentage or proportional data.

PieChart(

PieChartData(

// write your logic

),

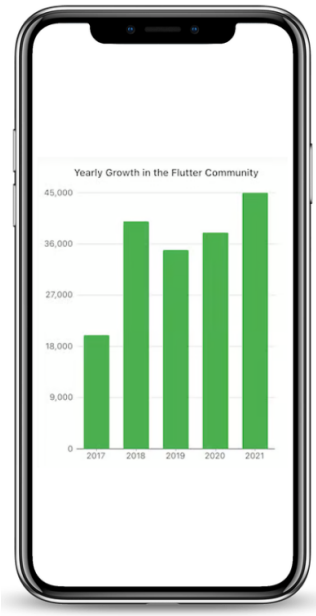
);

CODE:

```
import 'package:flutter/material.dart';
import 'package:charts_flutter/flutter.dart' as charts;
// @dart=2.9
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  late List<charts.Series<dynamic, String>> seriesList;
  static List<charts.Series<Sales, String>> _createRandomData()
  {
    final desktopSalesData = [
      Sales( '2017',19000),
      Sales( '2018',38000),
      Sales( '2019',35000),
      Sales( '2020',37000),
      Sales( '2021',45000),
    ];return[
```

```
      charts.Series<Sales, String>(
        id: 'Sales',
        domainFn: (Sales sales,_) =>sales.year,
        measureFn: (Sales sales,_) =>sales.sales,
        data: desktopSalesData,
        fillColorFn: (Sales sales, _)
        {return (sales.year == '2023')
            ?charts.MaterialPalette.green.shadeDefault
            :charts.MaterialPalette.green.shadeDefault;},)];}
  barChart() {return charts.BarChart(
      seriesList = _createRandomData()
  );
  }
  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Container(
      padding: EdgeInsets.all(20.0),
      child:barChart(),
    ),
  );

}
class Sales{
  final String year;
  final int sales;
  Sales(this.year,this.sales);
}
```

OUTPUT:



CONCLUSION: From this experiment we successfully included bar charts in Flutter application.

Name : Pranav Trivedi
Roll No. : 61
Batch : I6
Sub : MAD Lab

EXPERIMENT NO. 7

AIM: To apply navigation, routing and gestures in Flutter Application.

THEORY:

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. **For** example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.

Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Navigator.push()

The Navigator.push() method is used to navigate/switch to a new route/page/screen. Here, the push() method adds a page/route on the stack and then manage it by using the Navigator. Again we use MaterialPageRoute class that allows transition between the routes using a platform-specific animation. The below code explain the use of the Navigator.push() method.

Navigator.pop()

Navigator.pop() method is used to close the second route and return to the first route. The pop() method allows us to remove the current route from the stack, which is managed by the Navigator.

CODE:

```
import 'package:flutter/material.dart';
import 'package:untitled5/SecondScreen.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  void initState() {
    super.initState();
    // _navigation();
  }
  // _navigation()async{
  //   await Future.delayed(Duration(milliseconds: 3000),(){});
  //   Navigator.pushReplacement(this.context, MaterialPageRoute(builder:
(context)=>SecondScreen()));
  // }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Hello, World"),
            SizedBox(
              height: 20,
            ),
            ElevatedButton(onPressed:(){
              Navigator.push(this.context, MaterialPageRoute(builder:
(context)=>SecondScreen()));
            }, child:
            Text("Next"))
          ],
        ),
      ),
    );
  }
}
```

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

```dart
import 'main.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Navigation"),
      ),
      body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text("Bye Guys"),
          SizedBox(
            height: 20,
          ),
          ElevatedButton(onPressed:(){
            Navigator.pop(context);
          }, child:
          Text("Back")),
        ],
      ),
      ),
    );
  }
}
```

OUTPUT:

CONCLUSION: From this experiment we successfully applied navigation, routing and gestures in Flutter Application.