

120A3051

Shreya Idate

Batch: E3

## Experiment No. : 2

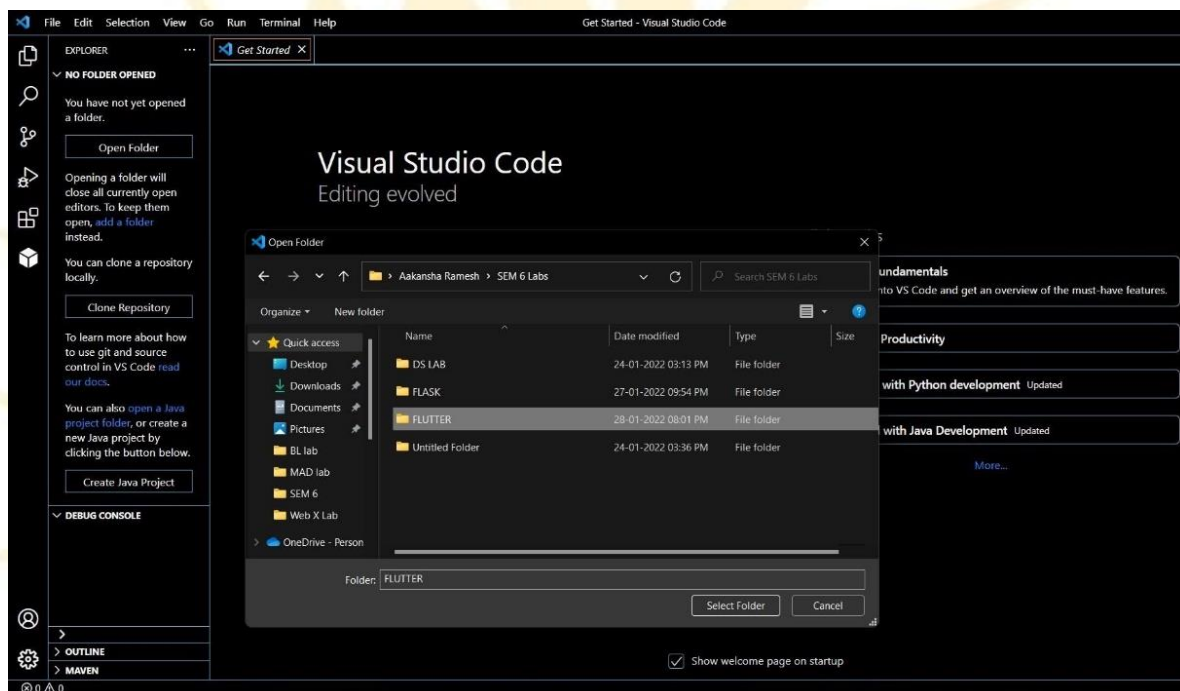
**Aim:** To design Flutter UI by including common widgets.

### Theory:

#### To Create Flutter Project:

##### (A) Opening a project from existing source code:

To create a new flutter project, for that we have to select a directory in which we are going to create the project. Click on the Open folder button and then choose a preferred location.



##### (B) Creating a new project:

**Step-1:** To create a new flutter project, click on view in the visual studio Codetoolbar and go to Command Palette. Choose Flutter: New Application Project. After that project will be created inside the test project directory.

**Step-2:** Select a folder to create the project in.

**Step-3:** Enter a name for your project

Now, the project has been created successfully.

Flutter project is created now we have to run this program in order to check that if it is working or not. We can run a flutter program in android Emulator or we can run this in our browser as well.

To do so type:

```
flutter run
```

Now we are ready to see the output, by default it has a program in which there is a button by clicking on the that a counter will be displayed in the center.

```
PS C:\Users\exam\Desktop\120A3051> cd basickwidgets
PS C:\Users\exam\Desktop\120A3051\basickwidgets> flutter create myapp
Wrote 127 files.

All done!
In order to run your application, type:

$ cd myapp
$ flutter run

Your application code is in myapp\lib\main.dart.

PS C:\Users\exam\Desktop\120A3051\basickwidgets> cd myapp
PS C:\Users\exam\Desktop\120A3051\basickwidgets\myapp> flutter run
Using hardware rendering with device AOSP on IA Emulator. If you notice graphics artifacts, consider enabling software rendering with "--enable-software-rendering".
Launching lib\main.dart on AOSP on IA Emulator in debug mode...
Running Gradle task 'assembleDebug'... 32.1s
✓ Built build\app\outputs\flutter-apk\app-debug.apk.
Installing build\app\outputs\flutter-apk\app.apk... 1,573ms
Syncing files to device AOSP on IA Emulator... 83ms

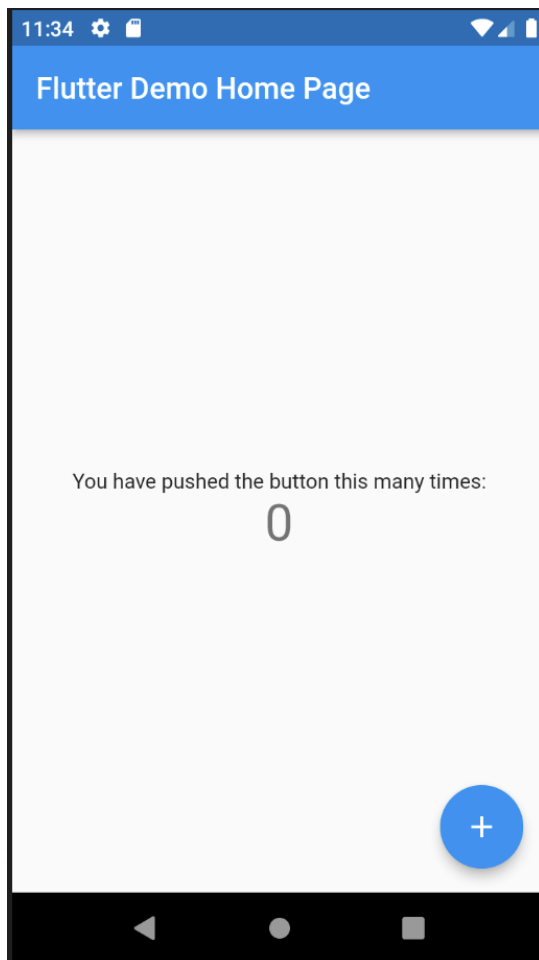
Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

Running with sound null safety

An Observatory debugger and profiler on AOSP on IA Emulator is available at: http://127.0.0.1:60297/cyKR2PdRhgm=/
D/eglCodecCommon( 6699): setVertexArrayObject: set vao to 0 (0) 1 0
The Flutter DevTools debugger and profiler on AOSP on IA Emulator is available at: http://127.0.0.1:9101?uri=http://127.0.0.1:60297/cyKR2PdRhgm=/
D/EGL_emulation( 6699): eglMakeCurrent: 0xdf105720: ver 2 0 (tinfo 0xdf103c90)

Taking screenshot for AOSP on IA Emulator... 366ms
Screenshot written to flutter_01.png (41kB).

Performing hot reload...
Reloaded 0 libraries in 107ms (compile: 8 ms, reload: 0 ms, reassemble: 27 ms).
D/EGL_emulation( 6699): eglMakeCurrent: 0xdf105720: ver 2 0 (tinfo 0xdf103c90)
[]
```



Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

### Types of Widgets:

There are two types of Widgets:

#### A. Visible widget (Output and Input) :

The visible widgets are related to the user input and output data. Some of the important types of this widget are:

1. **Text**: A Text widget holds some text to display on the screen. We can align the text widget by using *textAlign* property, and style property allow the customization of Text that includes *font*, *font weight*, *font style*, *letter spacing*, *colour*, and many more.

```
Text(  
  "Hello World! This is a Text Widget.",  
  style: TextStyle(  
    fontSize: 35,  
    color: Colors.purple,  
    fontWeight: FontWeight.w700,  
    fontStyle: FontStyle.italic,  
    letterSpacing: 8,  
    wordSpacing: 20,  
    backgroundColor: Colors.yellow,
```

```
    shadows: [  
      Shadow(color: Colors.blueAccent, offset: Offset(2,1), blurRadius:10)  
    ]  
  ),
```

2. **Image:** This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL. It provides many constructors for loading image, which are given below:

- **Image:** It is a generic image loader, which is used by ImageProvider.
- **asset:** It load image from your project asset folder.
- **file:** It loads images from the system folder.
- **memory:** It load image from memory.
- **network:** It loads images from the network.

**To add an image in the project, you need first to create an assets folder where you keep your images and then add the below line in pubspec.yaml file.**

3. **Icon:** This widget acts as a container for storing the Icon in the Flutter. Note. Image and icons are covered later.

## **B. Invisible widget (Layout and Control):**

The invisible widgets are related to the layout and control of widgets. It provides controlling how the widgets actually behave and how they will look onto the screen. Some of the important types of these widgets are:

1. **Column:** A column widget is a type of widget that arranges all its children's widgets in a vertical alignment. It provides spacing between the widgets by using the **mainAxisAlignment** and **crossAxisAlignment** properties. In these properties, the main axis is the vertical axis, and the cross axis is the horizontal axis.
2. **Row:** The row widget is similar to the column widget, but it constructs a widget horizontally rather than vertically. Here, the main axis is the horizontal axis, and the cross axis is the vertical axis.
3. **Container:** 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
4. **Center:** This widget is used to center the child widget, which comes inside it. All the previous examples contain inside the center widget.
5. **Padding:** This widget wraps other widgets to give them padding in specified directions. You can also provide padding in all directions. We can understand it from the below example that gives the text widget padding of 6.0 in all directions.

Example:

Row Widget	Column Widget
<pre> @override Widget build(BuildContext context) {return MaterialApp(   title: "Row-Col   Demo",home:   Scaffold(  body: Row( children: [   Containe r(     child: const Text('text 1'),     padding: const EdgeInsets.all(10),     margin: const EdgeInsets.all(40),color: Colors.green,   ),   Container(     child: const Text('text 2'),padding: const EdgeInsets.all(10),     margin: const EdgeInsets.all(40),color: Colors.lightGreen,   ), ], ), ), ); </pre>	<pre> @override Widget build(BuildContext context) {return MaterialApp(   title: "Row-Col   Demo",home:   Scaffold(     body: Column( //mainAxisAlignment:MainAxisAlignment.spaceEvenly, // crossAxisAlignment: CrossAxisAlignment.stretc h,     // mainAxisAlignment: MainAxisAlignment.max,  children: &lt;Widget&gt;[   Container(     child: const Text('text 1'),     padding: const EdgeInsets.all(10),margin: const EdgeInsets.all(40), color: Colors.green,   ),   Container(     child: const Text('text 2'),     padding: const EdgeInsets.all(10),margin: const EdgeInsets.all(40), color: Colors.lightGreen,   ), ], ), ), ); } </pre>

6. **Scaffold:** This widget provides a framework that allows you to add common materialdesign elements like AppBar, Floating Action Buttons, Drawers, etc.

```

@override
Widget build(BuildContext context) { return
MaterialApp(
  debugShowCheckedModeBanner: false, title:
  "My First App",
  theme: ThemeData(primarySwatch: Colors.blueGrey),
  home: Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.blueGrey,title:
      const Text("My AppBar"),

```



```
),  
body: const Center(  
  child: Text(  
    "Hello, theree!",  
    textAlign: TextAlign.center,  
    textScaleFactor: 1.5, overflow:  
    TextOverflow.fade, textDirection:  
    TextDirection.rtl, style: TextStyle(  
      backgroundColor: Colors.yellow,  
      fontStyle: FontStyle.italic, fontSize:  
      50,  
      color: Colors.blue,  
      letterSpacing: 20.0,  
      wordSpacing: 5,  
      shadows: [ Shadow(  
        color: Colors.blueAccent,  
        offset: Offset(2, 1),  
        blurRadius: 10)  
      ],  
    ),  
  ),  
),  
))),  
);  
}
```

7. **Stack:** It is an essential widget, which is mainly used for overlapping a widget, such as a button on a background gradient.

For every widget, we have 3 objects, One in each tree.

### 1. Widget tree

The first tree is called the Widget tree. This is used to configure the User Interface. There, we can configure the properties of the widget and decide how it should look.

### 2. Element tree

The second tree is called the Element tree. This is used to manage and update everything i.e., the trees and all references. In the element tree are elements. An element is a specific instance of a widget.

### 3. Render Tree

The last missing piece is Render Object. Inside of the Render tree are render objects and this is basically what we are seeing on the screen. We don't see widgets. We see render objects. Render objects will take care of the size, layout, and the actual painting to the screen.

Flutter goes through 3 stages when rendering the widgets to the screen.

**Stage-1: Configure**

The widget tree holds the widgets and, in this state, it's all about the configuration of widgets. Via an API we assign properties and values to our widgets which get hold by the widget tree.

**Stage-2: Life Cycle**

Here the whole life cycle of the UI gets managed. Also, here it determines the components that are existing in our UI hierarchy. We can imagine this state as the glue between state 1 and 3.

**Stage-3: Paint**

Here the Render tree comes into play. All relevant things about painting will be done here. It takes care of constraints, how the children of widgets will actually look like, how big they should be. This is where the render objects get painted on the screen.

**Basic Widgets in Flutter:**

- **AppBar:** It is usually the topmost component of the app (or sometimes the bottom-most), it contains the toolbar and some other common action buttons. As all the components in a flutter application is a widget or a combination of widgets. So AppBar is also a built-in class or widget in flutter which gives the functionality of the AppBar out of the box. The AppBar widget is based on Material Design and much of the information is already provided by other classes like MediaQuery, Scaffold as to where the content of the AppBar should be placed.

```
appBar: new AppBar(  
  title: new Text('AppBar'),  
  backgroundColor: Colors.green,  
  elevation: 30.0,  
  centerTitle: true,  
  titleSpacing: 70.0,  
  toolbarOpacity: 0.5,  
),
```

**Icons:** Flutter provides an Icon Widget to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or with the custom icons. Flutter provides the list of all icons in the Icons class. The following code snippet demonstrates Flutter icons in the application.

```
appBar: new AppBar(  
  title: new Text('AppBar'),
```

```
leading: new Icon(Icons.menu),
actions: <Widget>[
  new IconButton(icon: new Icon(Icons.arrow_forward),onPressed: (){print('you pressed
  Arrow Button');}),
  new IconButton(icon: new Icon(Icons.add),onPressed: (){print('you pressed Addition
  Button');}),
  new IconButton(icon: new Icon(Icons.close),onPressed: (){print('you pressed Close
  Button');}),
],
),
```

- **Row:** This widget arranges its children in a horizontal direction on the screen. In other words, it will expect child widgets in a horizontal array. If the child widgets need to fill the available horizontal space, we must wrap the children widgets in an Expanded widget. A row widget does not appear scrollable because it displays the widgets within the visible view. So, it is considered wrong if we have more children in a row which will not fit in the available space. If we want to make a scrollable list of row widgets, we need to use the ListView widget.
- **Column:** This widget arranges its children in a vertical direction on the screen. In other words, it will expect a vertical array of children widgets. If the child widgets need to fill the available vertical space, we must wrap the children widgets in an Expanded widget. A column widget does not appear scrollable because it displays the widgets within the visible view. So, it is considered wrong if we have more children in a column which will not fit in the available space. If we want to make a scrollable list of column widgets, we need to use the ListView Widget. 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.
- **Buttons:** They are the Flutter widgets, which are a part of the material design library. It allows a user to trigger an event; such as taking actions, making choices, searching things, and many more. They can be placed anywhere in our UI like dialogs, forms, cards, toolbars, etc. Flutter provides several types of buttons that have different shapes, styles, and features.
- **Text:** A Text is a widget in Flutter that allows us to display a string of text with a single line in our application. Depending on the layout constraints, we can break the string across multiple lines or might all be displayed on the same line. If we do not specify any styling to the text widget, it will use the closest DefaultTextStyle class style. This class does not have any explicit style. The following are the essential properties of the Text widget: TextAlign, TextDirection, Overflow, TextScaleFactor, SoftWrap, MaxLines,



TextWidthBasis, TextHeightBehavior, Style.

- **Center:** This widget comes built-in with flutter; it aligns its child widget to the center of the available space on the screen. The size of this widget will be as big as possible if the widthFactor and heightFactor properties are set to null and the dimensions are constrained. And in case the dimensions are not constrained and the widthFactor and HeightFactor are set to null then the Center widget takes the size of its child widget.

## Programs:

### 1) Text, Scaffold and AppBar:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "My First App",
      theme: ThemeData(primarySwatch: Colors.indigo),
      home: Scaffold(
        appBar: AppBar(
          title: const Text("TITLE"),
          elevation: 20,
          leading: IconButton(
            icon: const Icon(Icons.arrow_back),
            onPressed: () {
              print('you pressed Back Button');
            },
          ),
          actions: <Widget>[
            // new Icon(Icons.menu),
            IconButton(
              icon: const Icon(Icons.add),
              onPressed: () {
                print('you pressed Addition Button');
```

```
    }),  
    IconButton(  
      icon: const Icon(Icons.menu),  
      onPressed: () {  
        print('you pressed Menu Button');  
      },  
    ),  
  ],  
),  
body: const Center(  
  child: Text(  
    "Hello World!!",  
    style: TextStyle(  
      backgroundColor: Colors.indigo,  
      fontWeight: FontWeight.w400,  
      color: Colors.white,  
      fontSize: 50,  
      letterSpacing: 5,  
      //fontStyle: FontStyle.italic,  
    ),  
  )),  
));  
}  
}
```

## 2) Container:

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

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

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

```

```
debugShowCheckedModeBanner: false,  
title: "My First App",  
theme: ThemeData(primarySwatch: Colors.indigo),  
home: Scaffold(  
  appBar: AppBar(  
    title: const Text("My First App"),  
    elevation: 20,  
    leading: IconButton(  
      icon: const Icon(Icons.arrow_back),  
      onPressed: () {  
        print('you pressed Back Button');  
      }  
    ),  
    actions: <Widget>[  
      // new Icon(Icons.menu),  
      IconButton(  
        icon: const Icon(Icons.add),  
        onPressed: () {  
          print('you pressed Addition Button');  
        }  
      ),  
      IconButton(  
        icon: const Icon(Icons.menu),  
        onPressed: () {  
          print('you pressed Menu Button');  
        }  
      ),  
    ],  
  ),  
  body: Container(  
    padding: const EdgeInsets.all(10),  
    margin: const EdgeInsets.all(10),  
    alignment: Alignment.topCenter,  
    decoration: BoxDecoration(  
      border: Border.all(color: Colors.black, width: 4),  
      borderRadius: BorderRadius.circular(8),  
      boxShadow: const [  
        // 2nd  
        BoxShadow(  

```

```
        color: Colors.indigo,
        offset: Offset(6.0, 6.0),
      ),
    ],
  ),
  child: const Text(
    "This is a container example",
    style: TextStyle(
      fontWeight: FontWeight.w400,
      color: Colors.white,
      fontSize: 50,
      letterSpacing: 5,
      //fontStyle: FontStyle.italic,
    ),
  )),
));
}
```

### 3) Column:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: "My First App",
      theme: ThemeData(primarySwatch: Colors.indigo),
      home: Scaffold(
        appBar: AppBar(
```



```
title: const Text("My First App"),
elevation: 20,
leading: IconButton(
  icon: const Icon(Icons.arrow_back),
  onPressed: () {
    print('you pressed Back Button');
  }),
actions: <Widget>[
  // new Icon(Icons.menu),
  IconButton(
    icon: const Icon(Icons.add),
    onPressed: () {
      print('you pressed Addition Button');
    }),
  IconButton(
    icon: const Icon(Icons.menu),
    onPressed: () {
      print('you pressed Menu Button');
    },
  ),
],
),
body: Column(
  children: [
    Expanded(
      child: Container(
        padding: const EdgeInsets.all(10),
        margin: const EdgeInsets.all(10),
        decoration: BoxDecoration(
          border: Border.all(color: Colors.black, width: 4),
          borderRadius: BorderRadius.circular(8),
          boxShadow: const [
            // 2nd
            BoxShadow(
              color: Colors.indigo,
              offset: Offset(6.0, 6.0),
            ),
          ],
        ),
      ),
    ],
  ),
),
```

```
    ],  
  ),  
  child: const Text(  
    "Child 1",  
    style: TextStyle(  
      fontWeight: FontWeight.w400,  
      color: Colors.white,  
      fontSize: 50,  
      letterSpacing: 5,  
      //fontStyle: FontStyle.italic,  
    ),  
  )),  
),  
Expanded(  
  child: Container(  
    padding: const EdgeInsets.all(10),  
    margin: const EdgeInsets.all(10),  
    decoration: BoxDecoration(  
      border: Border.all(color: Colors.black, width: 4),  
      borderRadius: BorderRadius.circular(8),  
      boxShadow: const [  
        // 2nd  
        BoxShadow(  
          color: Colors.indigo,  
          offset: Offset(6.0, 6.0),  
        ),  
      ],  
    ),  
  ),  
  child: const Text(  
    "Child 2",  
    style: TextStyle(  
      fontWeight: FontWeight.w400,  
      color: Colors.white,  
      fontSize: 50,  
      letterSpacing: 5,  
      //fontStyle: FontStyle.italic,  
    ),  
  ),  
),
```

```
    )),  
  ),  
  Expanded(  
    child: Container(  
      padding: const EdgeInsets.all(10),  
      margin: const EdgeInsets.all(10),  
      decoration: BoxDecoration(  
        border: Border.all(color: Colors.black, width: 4),  
        borderRadius: BorderRadius.circular(8),  
        boxShadow: const [  
          // 2nd  
          BoxShadow(  
            color: Colors.indigo,  
            offset: Offset(6.0, 6.0),  
          ),  
        ],  
      ),  
      child: const Text(  
        "Child 3",  
        style: TextStyle(  
          fontWeight: FontWeight.w400,  
          color: Colors.white,  
          fontSize: 50,  
          letterSpacing: 5,  
          //fontStyle: FontStyle.italic,  
        ),  
      )),  
    ),  
    Expanded(  
      child: Container(  
        padding: const EdgeInsets.all(10),  
        margin: const EdgeInsets.all(10),  
        decoration: BoxDecoration(  
          border: Border.all(color: Colors.black, width: 4),  
          borderRadius: BorderRadius.circular(8),  
          boxShadow: const [  
            // 2nd
```

```
BoxShadow(  
  color: Colors.indigo,  
  offset: Offset(6.0, 6.0),  
),  
],  
),  
child: const Text(  
  "Child 4",  
  style: TextStyle(  
    fontWeight: FontWeight.w400,  
    color: Colors.white,  
    fontSize: 50,  
    letterSpacing: 5,  
    //fontStyle: FontStyle.italic,  
  ),  
)),  
),  
Expanded(  
  child: Container(  
    padding: const EdgeInsets.all(10),  
    margin: const EdgeInsets.all(10),  
    decoration: BoxDecoration(  
      border: Border.all(color: Colors.black, width: 4),  
      borderRadius: BorderRadius.circular(8),  
      boxShadow: const [  
        // 2nd  
        BoxShadow(  
          color: Colors.indigo,  
          offset: Offset(6.0, 6.0),  
        ),  
      ],  
    ),  
    child: const Text(  
      "Child 5",  
      style: TextStyle(  
        fontWeight: FontWeight.w400,  
        color: Colors.white,
```



```
        fontSize: 50,  
        letterSpacing: 5,  
        //fontStyle: FontStyle.italic,  
      ),  
    )),  
  ),  
],  
),  
));  
}  
}
```

#### 4) Row:

```
import 'package:flutter/material.dart';  
  
void main() {  
  runApp(const MyApp());  
}  
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      debugShowCheckedModeBanner: false,  
      title: "My First App",  
      theme: ThemeData(primarySwatch: Colors.indigo),  
      home: Scaffold(  
        appBar: AppBar(  
          title: const Text("My First App"),  
          elevation: 20,  
          leading: IconButton(  
            icon: const Icon(Icons.arrow_back),  
            onPressed: () {  
              print('you pressed Back Button');  
            }  
          )),  
        actions: <Widget>[  
          // new Icon(Icons.menu),  
          IconButton(  

```

```
        icon: const Icon(Icons.add),
        onPressed: () {
          print('you pressed Addition Button');
        }),
      IconButton(
        icon: const Icon(Icons.menu),
        onPressed: () {
          print('you pressed Menu Button');
        },
      ),
    ],
  ),
  body: Row(
    children: [
      Expanded(
        child: Container(
          padding: const EdgeInsets.all(10),
          margin: const EdgeInsets.all(10),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black, width: 4),
            borderRadius: BorderRadius.circular(8),
            boxShadow: const [
              // 2nd
              BoxShadow(
                color: Colors.indigo,
                offset: Offset(6.0, 6.0),
              ),
            ],
          ),
          child: const Text(
            "Child 1",
            style: TextStyle(
              fontWeight: FontWeight.w400,
              color: Colors.white,
              fontSize: 20,
              letterSpacing: 5,
              fontStyle: FontStyle.italic,
```

```
    ),  
    )),  
  ),  
  Expanded(  
    child: Container(  
      padding: const EdgeInsets.all(10),  
      margin: const EdgeInsets.all(10),  
      decoration: BoxDecoration(  
        border: Border.all(color: Colors.black, width: 4),  
        borderRadius: BorderRadius.circular(8),  
        boxShadow: const [  
          // 2nd  
          BoxShadow(  
            color: Colors.indigo,  
            offset: Offset(6.0, 6.0),  
          ),  
        ],  
      ),  
      child: const Text(  
        "Child 2",  
        style: TextStyle(  
          fontWeight: FontWeight.w400,  
          color: Colors.white,  
          fontSize: 20,  
          letterSpacing: 5,  
          //fontStyle: FontStyle.italic,  
        ),  
      )),  
    ),  
    Expanded(  
      child: Container(  
        padding: const EdgeInsets.all(10),  
        margin: const EdgeInsets.all(10),  
        decoration: BoxDecoration(  
          border: Border.all(color: Colors.black, width: 4),  
          borderRadius: BorderRadius.circular(8),  
          boxShadow: const [  

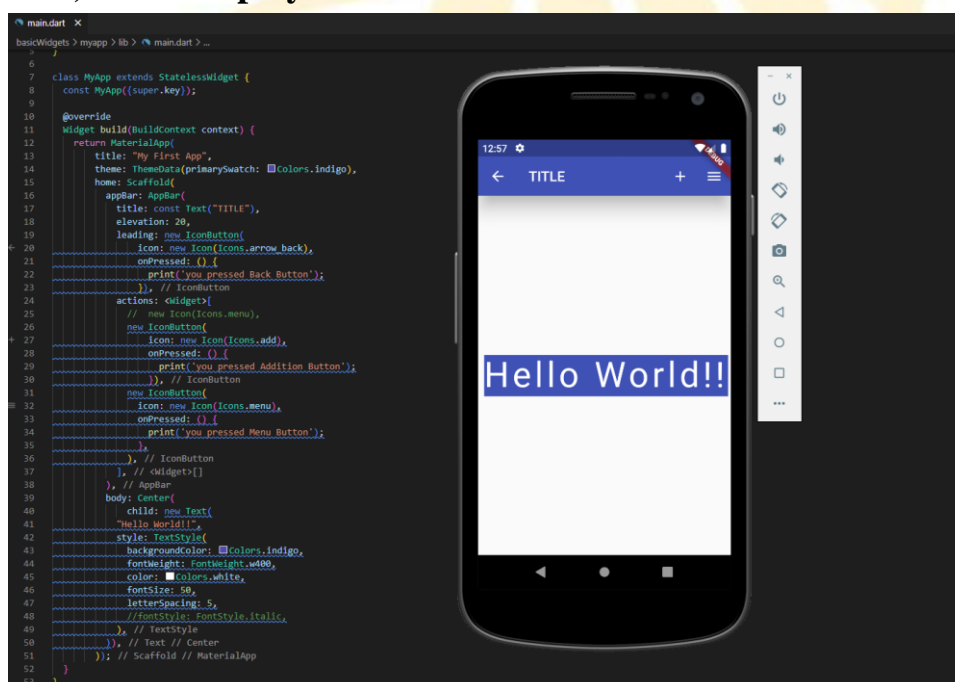
```

// 2nd

```
BoxShadow(
  color: Colors.indigo,
  offset: Offset(6.0, 6.0),
),
],
),
child: const Text(
  "Child 3",
  style: TextStyle(
    fontWeight: FontWeight.w400,
    color: Colors.white,
    fontSize: 20,
    letterSpacing: 5,
    //fontStyle: FontStyle.italic,
  ),
),
),
),
);
```

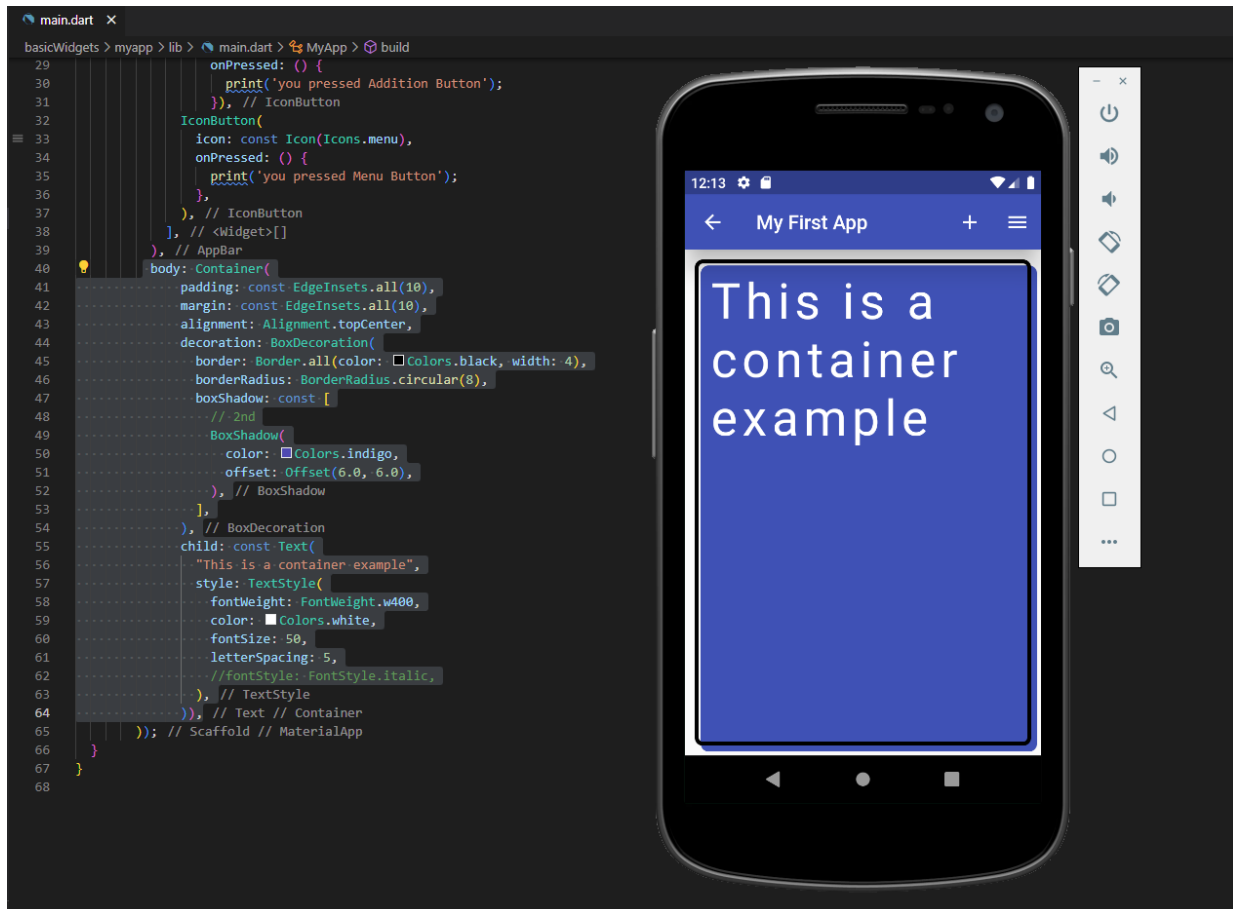
## Output:

### 1) Text Display





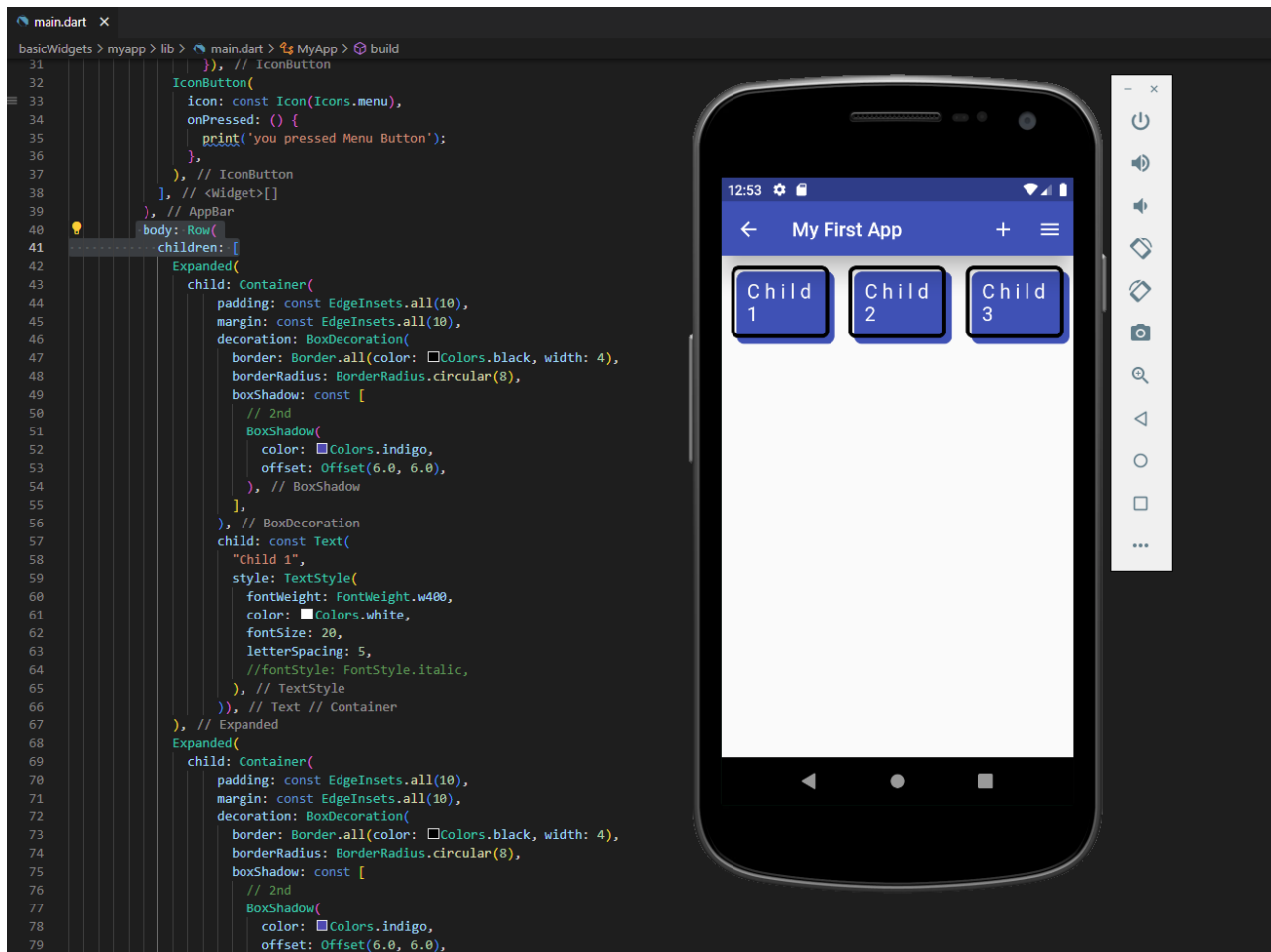
## 2) Container



## 3) Column:



#### 4) Row:



#### Conclusion:

Hence, we have successfully designed a Flutter UI by including common widgets. And we have learned that Flutter is one of the best solutions to develop apps for Android and iOS, without having to write in a different codebase for each platform. The smartphone versions of these apps function as true, native apps on Apple and Android devices and are compiled for the respective platform before publication. They do not need a runtime module or a browser.

Using the same codebase, it is also possible to create web apps for browsers as well as native programs for Windows, Linux and macOS. The popularity of Flutter app development is constantly growing. Now Flutter is widely used to create apps for the likes of Alibaba, Yandex, Airbnb, Uber, eBay and other leading companies.