Chapter Three
Flutter Widgets

Types of Widgets

State Management Widget StatelessWidget vs StatefulWidget

A widget is either stateful or stateless. If a widget can change—when a user interacts with it, for example—it's stateful.
A stateless widget never changes. Icon, IconButton, and Text are examples of stateless widgets. Stateless widgets subclass StatelessWidget.
A stateful widget is dynamic: for example, it can change its appearance in response to events triggered by user interactions or when it receives data. Checkbox, Radio, Slider, InkWell, Form, and TextField are examples of stateful widgets. Stateful widgets subclass StatefulWidget.
A widget's state is stored in a State object, separating the widget's state from its appearance. The state consists of values that can change, like a slider's current value or whether a checkbox is checked. When the widget's state changes, the state object calls setState(), telling the framework to redraw the widget.

Flutter Scaffold

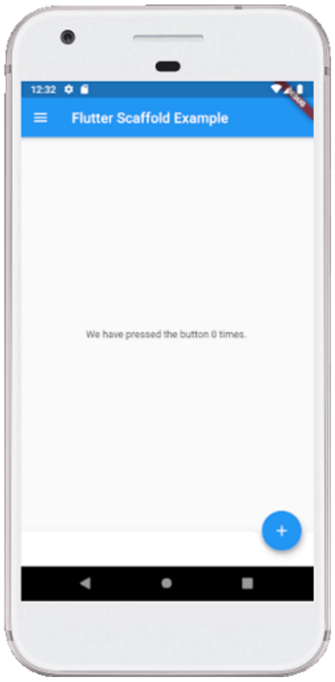
The Scaffold is a widget in Flutter used to implements the basic material design visual layout structure. It is quick enough to create a general-purpose mobile application and contains almost everything we need to create a functional and responsive Flutter apps. This widget is able to occupy the whole device screen. In other words, we can say that it is mainly responsible for creating a base to the app screen on which the child widgets hold on and render on the screen. It provides many widgets or APIs for showing Drawer, SnackBar, BottomNavigationBar, AppBar, FloatingActionButton, and many more.
The Scaffold class is a shortcut to set up the look and design of our app that allows us not to build the individual visual elements manually. It saves our time to write more code for the look and feel of the app. The following are the constructor and properties of the Scaffold widget class.

In this example, we are going to see a Scaffold widget with an AppBar, BottomAppBar, FloatingActionButton, floatingActionButtonLocation, and drawer properties.

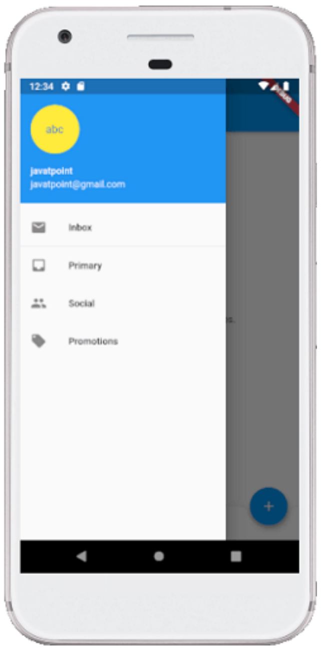
```
import 'package:flutter/material.dart';

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyStatefulWidget(),
    );
  }
}

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

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

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  int _count = 0;

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Scaffold Example'),
      ),
      body: Center(
        child: Text('We have pressed the button $_count times.'),
      ),
      bottomNavigationBar: BottomAppBar(
        shape: const CircularNotchedRectangle(),
        child: Container(
          height: 50.0,
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => setState(() {
          _count++;
        }),
        tooltip: 'Increment Counter',
        child: Icon(Icons.add),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.endDocked,
      drawer: Drawer(
        elevation: 20.0,
        child: Column(
          children: <Widget>[
            UserAccountsDrawerHeader(
              accountName: Text("javatpoint"),
              accountEmail: Text("javatpoint@gmail.com"),
              currentAccountPicture: CircleAvatar(
                backgroundColor: Colors.yellow,
                child: Text("abc"),
              ),
            ),
            ListTile(
              title: new Text("Inbox"),
              leading: new Icon(Icons.mail),
            ),
            Divider( height: 0.1,),
            ListTile(
              title: new Text("Primary"),
              leading: new Icon(Icons.inbox),
            ),
            ListTile(
              title: new Text("Social"),
              leading: new Icon(Icons.people),
            ),
            ListTile(
              title: new Text("Promotions"),
              leading: new Icon(Icons.local_offer),
            )
          ],
        ),
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyStatefulWidget(),
    );
  }
}

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

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

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  int _count = 0;

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Scaffold Example'),
      ),
      body: Center(
        child: Text('We have pressed the button $_count times.'),
      ),
      bottomNavigationBar: BottomAppBar(
        shape: const CircularNotchedRectangle(),
        child: Container(
          height: 50.0,
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => setState(() {
          _count++;
        }),
        tooltip: 'Increment Counter',
        child: Icon(Icons.add),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.endDocked,
      drawer: Drawer(
        elevation: 20.0,
        child: Column(
          children: <Widget>[
            UserAccountsDrawerHeader(
              accountName: Text("javatpoint"),
              accountEmail: Text("javatpoint@gmail.com"),
              currentAccountPicture: CircleAvatar(
                backgroundColor: Colors.yellow,
                child: Text("abc"),
              ),
            ),
            ListTile(
              title: new Text("Inbox"),
              leading: new Icon(Icons.mail),
            ),
            Divider( height: 0.1,),
            ListTile(
              title: new Text("Primary"),
              leading: new Icon(Icons.inbox),
            ),
            ListTile(
              title: new Text("Social"),
              leading: new Icon(Icons.people),
            ),
            ListTile(
              title: new Text("Promotions"),
              leading: new Icon(Icons.local_offer),
            )
          ],
        ),
      ),
    );
  }
}
```

Output:

When we run this project in the IDE, we will see the UI as the following screenshot.



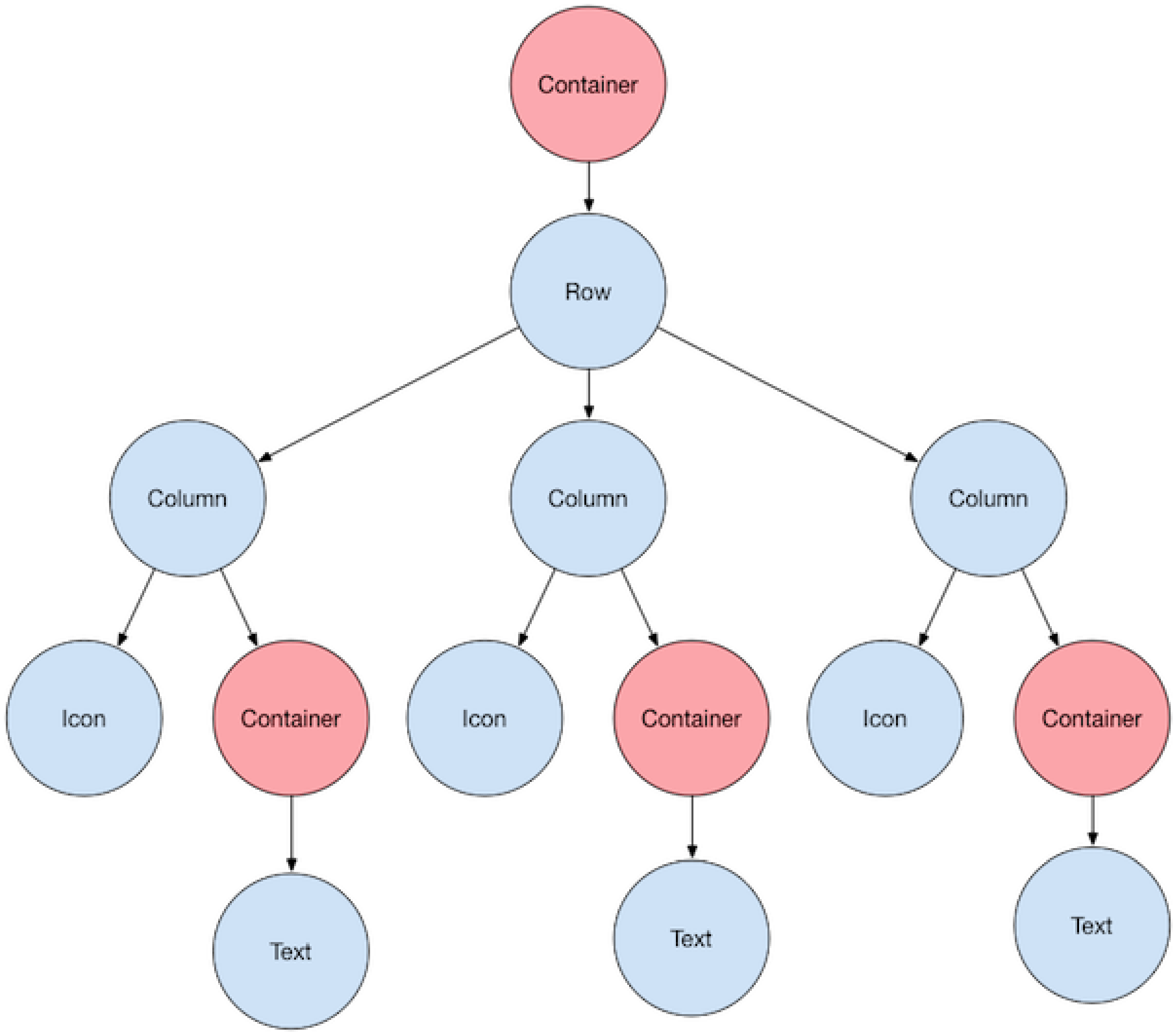
If we click on the three lines that can be seen in the top left corner of the screen, we will see the drawer. The drawer can be swiped right to left or left to right. See the below image.



Flutter Layouts

The core of Flutter's layout mechanism is widgets. In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets.

Here's a diagram of the widget tree for this UI:



Most of this should look as you might expect, but you might be wondering about the containers (shown in pink). Container is a widget class that allows you to customize its child widget. Use a Container when you want to add padding, margins, borders, or background color, to name some of its capabilities.

In this example, each Text widget is placed in a Container to add margins. The entire Row is also placed in a Container to add padding around the row.

The rest of the UI in this example is controlled by properties. Set an Icon's color using its color property. Use the Text.style property to set the font, its color, weight, and so on. Columns and rows have properties that allow you to specify how their children are aligned vertically or horizontally, and how much space the children should occupy.

Lay out a widget

How do you lay out a single widget in Flutter? This section shows you how to create and display a simple widget. It also shows the entire code for a simple Hello World app.

In Flutter, it takes only a few steps to put text, an icon, or an image on the screen.

1. Select a layout widget

Choose from a variety of layout widgets based on how you want to align or constrain the visible widget, as these characteristics are typically passed on to the contained widget.

This example uses Center which centers its content horizontally and vertically.

2. Create a visible widget

For example, create a Text widget:

```
Text('Hello World'),
```

Create an Image widget:

```
return Image.asset(
  image,
  fit: BoxFit.cover,
);
```

Create an Icon widget:

```
Icon(
  Icons.star,
  color: Colors.red[500],
),
```

3. Add the visible widget to the layout widget

All layout widgets have either of the following:

- A child property if they take a single child—for example, Center or Container
- A children property if they take a list of widgets—for example, Row, Column, ListView, or Stack.

Add the Text widget to the Center widget:

```
const Center(
  child: Text('Hello World'),
),
```

4. Add the layout widget to the page

A Flutter app is itself a widget, and most widgets have a build() method. Instantiating and returning a widget in the app's build() method displays the widget.

Material apps

For a Material app, you can use a Scaffold widget; it provides a default banner, background color, and has API for adding drawers, snack bars, and bottom sheets. Then you can add the Center widget directly to the body property for the home page.

lib/main.dart (MyApp)

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

  @override
  Widget build(BuildContext context) {
    const String appTitle = 'Flutter layout demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Cupertino apps

To create a Cupertino app, use CupertinoApp and CupertinoPageScaffold widgets.

Unlike Material, it doesn't provide a default banner or background color. You need to set these yourself.

- To set default colors, pass in a configured CupertinoThemeData to your app's theme property.
- To add an iOS-styled navigation bar to the top of your app, add a CupertinoNavigationBar widget to the navigationBar property of your scaffold. You can use the colors that CupertinoColors provides to configure your widgets to match iOS design.
- To lay out the body of your app, set the child property of your scaffold with the desired widget as its value, like Center or Column.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const CupertinoApp(
      title: 'Flutter layout demo',
      theme: CupertinoThemeData(
        brightness: Brightness.light,
        primaryColor: CupertinoColors.systemBlue,
      ),
      home: CupertinoPageScaffold(
        navigationBar: CupertinoNavigationBar(
          backgroundColor: CupertinoColors.systemGrey,
          middle: Text('Flutter layout demo'),
        ),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text('Hello World'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Non-Material apps

For a non-Material app, you can add the Center widget to the app's build() method:

lib/main.dart (MyApp)

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

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: const BoxDecoration(color: Colors.white),
      child: const Center(
        child: Text(
          'Hello World',
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 32,
            color: Colors.black87,
          ),
        ),
      ),
    );
  }
}
```

By default, a non-Material app doesn't include an AppBar, title, or background color. If you want these features in a non-Material app, you have to build them yourself. This app changes the background color to white and the text to dark grey to mimic a Material app.

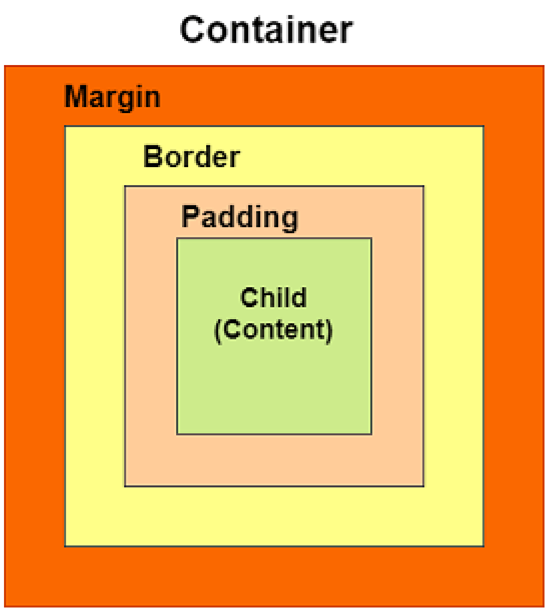
Flutter Container

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 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.
A container widget is same as <div> tag in html. If this widget does not contain any child widget, it will fill the whole area on the screen automatically. Otherwise, it will wrap the child widget according to the specified height & width. It is to note that this widget cannot render directly without any parent widget. We can use Scaffold widget, Center widget, Padding widget, Row widget, or Column widget as its parent widget.

Why we need a container widget in Flutter?

If we have a widget that needs some background styling may be a color, shape, or size constraints, we may try to wrap it in a container widget. This widget helps us to compose, decorate, and position its child widgets. If we wrap our widgets in a container, then without using any parameters, we would not notice any difference in its appearance. But if we add any properties such as color, margin, padding, etc. in a container, we can style our widgets on the screen according to our needs.

A basic container has a margin, border, and padding properties surrounding its child widget, as shown in the below image:



Properties of Container widget

Let us learn some of the essential properties of the container widget in detail.

1. child: This property is used to store the child widget of the container. Suppose we have taken a Text widget as its child widget that can be shown in the below example:

```
        Container(
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

```
Container(
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

2. color: This property is used to set the background color of the text. It also changes the background color of the entire container. See the below example:

```
Container(
    color: Colors.green,
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
Container(
    color: Colors.green,
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

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. See the below code:

```
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

4. margin: This property is used to surround the empty 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, as shown in the below example:

```
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    margin: EdgeInsets.all(20),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    margin: EdgeInsets.all(20),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

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. Here, we have used an EdgeInsets.all(35) that set the space between text and all four container directions.

```
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    padding: EdgeInsets.all(35),
    margin: EdgeInsets.all(20),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
Container(
    width: 200.0,
    height: 100.0,
    color: Colors.green,
    padding: EdgeInsets.all(35),
    margin: EdgeInsets.all(20),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

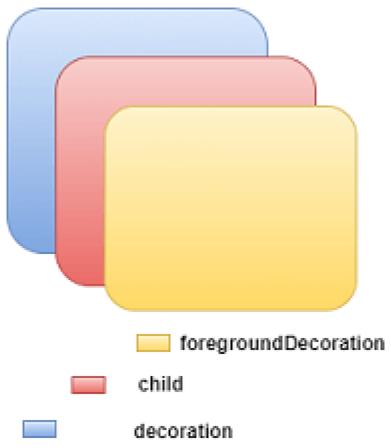
6. alignment: This property is used to set the position of the child within the container. Flutter allows the user to align its element in various ways such as center, bottom,

bottom center, topLeft, centerRight, left, right, and many more. In the below example, we are going to align its child into the bottom right position.

```
Container(
   width: 200.0,
   height: 100.0,
   color: Colors.green,
   padding: EdgeInsets.all(35),
   margin: EdgeInsets.all(20),
   alignment: Alignment.bottomRight,
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

```
Container(
   width: 200.0,
   height: 100.0,
   color: Colors.green,
   padding: EdgeInsets.all(35),
   margin: EdgeInsets.all(20),
   alignment: Alignment.bottomRight,
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

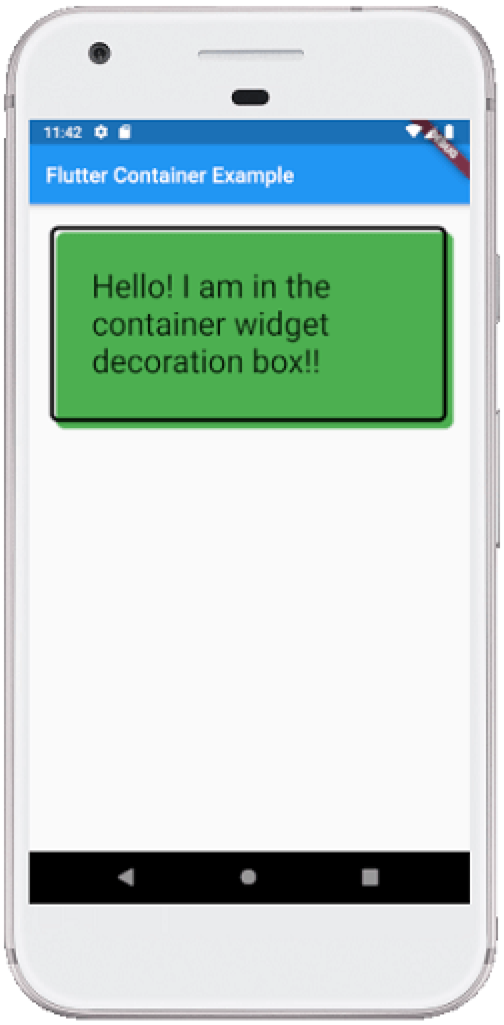
7. decoration: This property allows the developer to add decoration on the widget. It decorates or paint the widget behind the child. If we want to decorate or paint in front of a child, we need to use the forgroundDecoration parameter. The below image explains the difference between them where the foregroundDecoration covers the child and decoration paint behind the child.



The decoration property supported many parameters, such as color, gradient, background image, border, shadow, etc. It is to make sure that we can either use the color property in a container or decoration, but not in both. See the below code where we have added a border and shadow property to decorate the box:

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

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Flutter Container Example"),
        ),
        body: Container(
          padding: EdgeInsets.all(35),
          margin: EdgeInsets.all(20),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black, width: 4),
            borderRadius: BorderRadius.circular(8),
          boxShadow: [
            new BoxShadow(color: Colors.green, offset: new Offset(6.0, 6.0),),
          ],
        ),
        child: Text("Hello! I am in the container widget decoration box!!",
            style: TextStyle(fontSize: 30)),
      ),
    ),
  );
 }
}
import 'package:flutter/material.dart';

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Flutter Container Example"),
        ),
        body: Container(
          padding: EdgeInsets.all(35),
          margin: EdgeInsets.all(20),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black, width: 4),
            borderRadius: BorderRadius.circular(8),
            boxShadow: [
              new BoxShadow(color: Colors.green, offset: new Offset(6.0, 6.0),),
            ],
          ),
          child: Text("Hello! I am in the container widget decoration box!!",
            style: TextStyle(fontSize: 30)),
        ),
      ),
    );
  }
}
```

We will see the output as below screenshot:



Hello! I am in the container widget decoration box!!

8. transform: The transform property allows developers to rotate the container. It can rotate the container in any direction, i.e., change the container coordinate in the parent widget. In the below example, we will rotate the container in the z-axis.

```
Container(
   width: 200.0,
   height: 100.0,
   color: Colors.green,
   padding: EdgeInsets.all(35),
   margin: EdgeInsets.all(20),
   alignment: Alignment.bottomRight,
   transform: Matrix4.rotationZ(0.1),
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

```
Container(
   width: 200.0,
   height: 100.0,
   color: Colors.green,
   padding: EdgeInsets.all(35),
   margin: EdgeInsets.all(20),
   alignment: Alignment.bottomRight,
   transform: Matrix4.rotationZ(0.1),
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

9. constraints: This property is used when we want to add additional constraints to the child. It contains various constructors, such as tight, loose, expand, etc. Let's see how to use these constructors in our app:

tight: If we use size property in this, it will give fixed value to the child.

```
Container(
   color: Colors.green,
   constraints: BoxConstraints.tight(Size size)
      : minWidth = size.width, maxWidth = size.width,
        minHeight = size.height, maxHeight = size.height;
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

```
Container(
   color: Colors.green,
   constraints: BoxConstraints.tight(Size size)
       : minWidth = size.width, maxWidth = size.width,
         minHeight = size.height, maxHeight = size.height;
   child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

expand: Here, we can choose the height, width, or both values to the child.

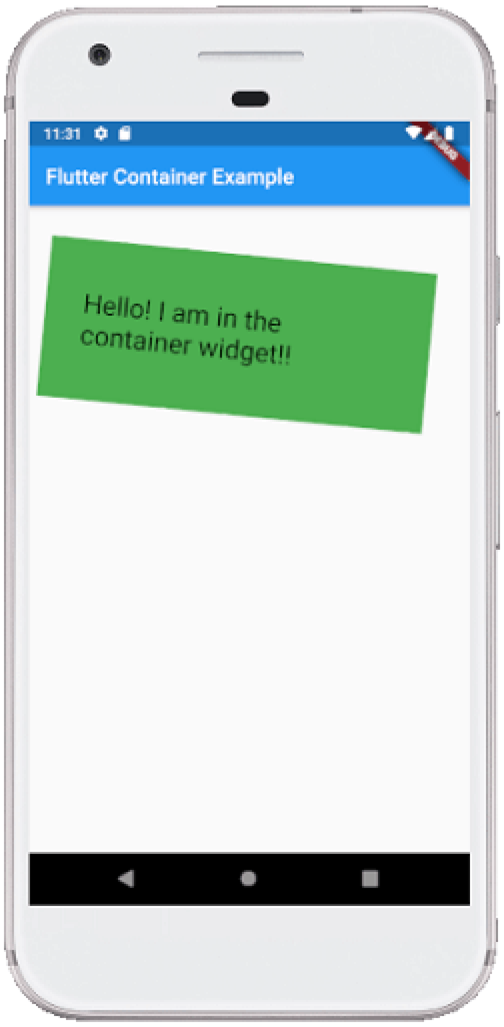
```
Container(
    color: Colors.green,
    constraints: BoxConstraints.expand(height: 60.0),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)

Container(
    color: Colors.green,
    constraints: BoxConstraints.expand(height: 60.0),
    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
)
```

Let us understand it with an example where we will try to cover most of the container properties. Open the main.dart file and replace it with the below code:

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

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyContainerWidget(),
    );
  }
}
```

```
class MyContainerWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Flutter Container Example"),
        ),
        body: Container(
          width: double.infinity,
          height: 150.0,
          color: Colors.green,
          margin: EdgeInsets.all(25),
          padding: EdgeInsets.all(35),
          alignment: Alignment.center,
          transform: Matrix4.rotationZ(0.1),
          child: Text("Hello! I am in the container widget!!",
              style: TextStyle(fontSize: 25)),
        ),
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

/// This Widget is the main application widget.
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyContainerWidget(),
    );
  }
}

class MyContainerWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Flutter Container Example"),
        ),
        body: Container(
          width: double.infinity,
          height: 150.0,
          color: Colors.green,
          margin: EdgeInsets.all(25),
          padding: EdgeInsets.all(35),
          alignment: Alignment.center,
          transform: Matrix4.rotationZ(0.1),
          child: Text("Hello! I am in the container widget!!",
              style: TextStyle(fontSize: 25)),
        ),
      ),
    );
  }
}
```

When we run this app, it will give the following screenshot:


11:31
Flutter Container Example
Hello! I am in the container widget!!

Flutter Row and Column

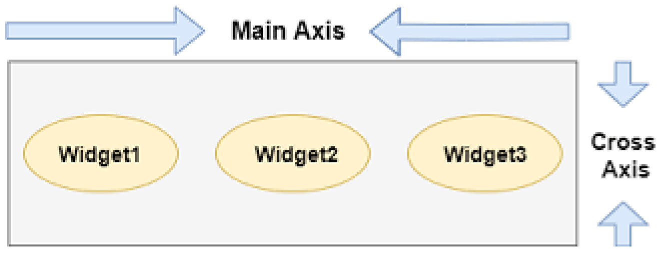
Row and column are the two essential widgets in Flutter that allows developers to align children horizontally and vertically according to our needs. These widgets are very necessary when we design the application user interface in Flutter.

Row Widget

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.

We can control how a row widget aligns its children based on our choice using the property crossAxisAlignment and mainAxisAlignment. The row's cross-axis will run vertically, and the main axis will run horizontally. See the below visual representation to understand it more clearly.



Note: Flutter row widget has several other properties like mainAxisSize, textDirection, verticalDirection, etc. Here, we will discuss only mainAxisAlignment and crossAxisAlignment properties.

We can align the row's children widget with the help of the following properties:

- start: It will place the children from the starting of the main axis.
- end: It will place the children at the end of the main axis.
- center: It will place the children in the middle of the main axis.
- spaceBetween: It will place the free space between the children evenly.
- spaceAround: It will place the free space between the children evenly and half of that space before and after the first and last children widget.
- spaceEvenly: It will place the free space between the children evenly and before and after the first and last children widget.

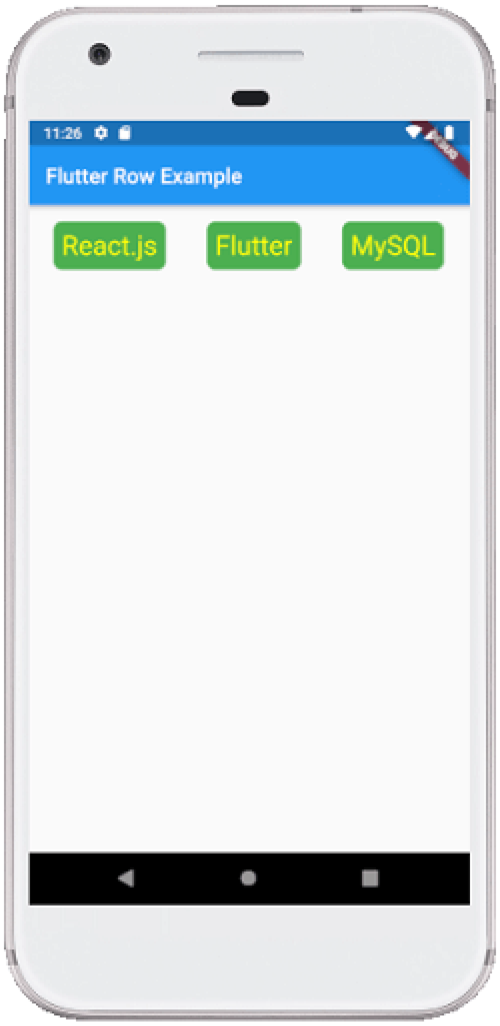
Let us understand it with the help of an example where we are going to align the content such that there is an even space around the children in a row:

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

void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage()
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Flutter Row Example"),
      ),
      body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children:<Widget>[
            Container(
              margin: EdgeInsets.all(12.0),
              padding: EdgeInsets.all(8.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.green
              ),
              child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),

            ),
            Container(
              margin: EdgeInsets.all(15.0),
              padding: EdgeInsets.all(8.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                color:Colors.green
              ),
              child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
            ),
            Container(
              margin: EdgeInsets.all(12.0),
              padding: EdgeInsets.all(8.0),
              decoration:BoxDecoration(
                borderRadius:BorderRadius.circular(8),
                color:Colors.green
              ),
              child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
            )
          ]
      ),
    );
  }
}
import 'package:flutter/material.dart';

void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage()
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

```
class _MyHomePageState extends State<MyHomePage> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text("Flutter Row Example"),
   ),
   body: Row(
     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
     children:<Widget>[
      Container(
       margin: EdgeInsets.all(12.0),
       padding: EdgeInsets.all(8.0),
       decoration:BoxDecoration(
         borderRadius:BorderRadius.circular(8),
         color:Colors.green
       ),
       child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),

      ),
      Container(
       margin: EdgeInsets.all(15.0),
       padding: EdgeInsets.all(8.0),
       decoration:BoxDecoration(
         borderRadius:BorderRadius.circular(8),
         color:Colors.green
       ),
       child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
      ),
      Container(
       margin: EdgeInsets.all(12.0),
       padding: EdgeInsets.all(8.0),
       decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.green
                ),
                child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
              )
            ]
          ),
        );
      }
    }
```

Output:

When we run this app, we should get the UI as the below screenshot.

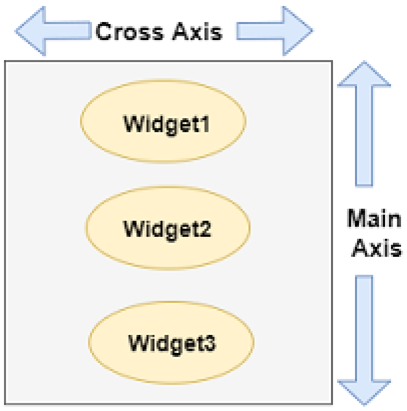


Column

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.

We can also control how a column widget aligns its children using the property mainAxisAlignment and crossAxisAlignment. The column's cross-axis will run horizontally, and the main axis will run vertically. The below visual representation explains it more clearly.



Note: Column widget also aligns its content by using the same properties as we have discussed in row widget such as start, end, center, spaceAround, spaceBetween, and spaceEvenly.

Let us understand it with the help of an example where we are going to align the content such that there is a free space between the children evenly in a column:

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

void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
    home: MyHomePage()
  );
 }
```

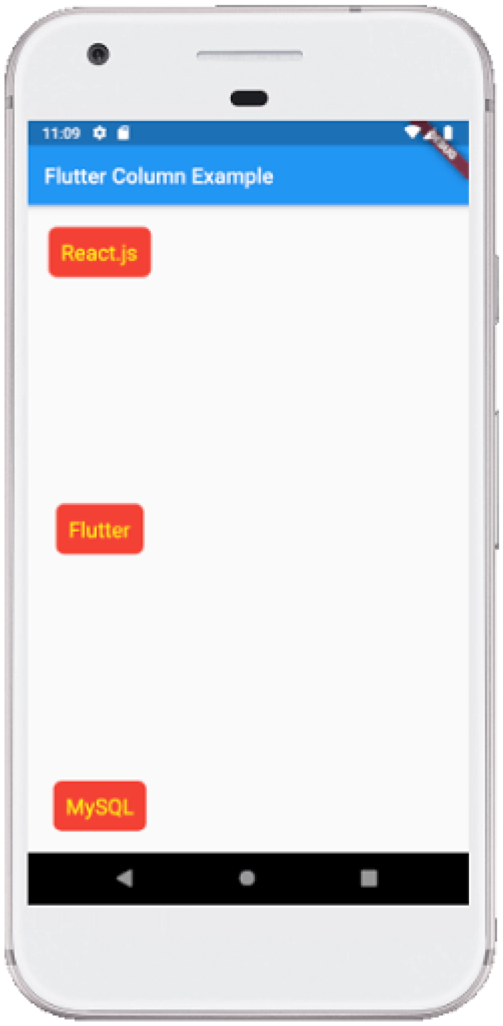
```
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Flutter Column Example"),
      ),
      body: Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children:<Widget>[
            Container(
              margin: EdgeInsets.all(20.0),
              padding: EdgeInsets.all(12.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.red
              ),
              child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),

            ),
            Container(
              margin: EdgeInsets.all(20.0),
              padding: EdgeInsets.all(12.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.red
              ),
            child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
          ),
          Container(
            margin: EdgeInsets.all(20.0),
            padding: EdgeInsets.all(12.0),
            decoration:BoxDecoration(
                borderRadius:BorderRadius.circular(8),
                color:Colors.red
            ),
            child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
          )
        ]
      ),
    );
  }
}
import 'package:flutter/material.dart';

void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: MyHomePage()
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
   appBar: AppBar(
     title: Text("Flutter Column Example"),
   ),
   body: Column(
       mainAxisAlignment: MainAxisAlignment.spaceBetween,
       children:<Widget>[
         Container(
           margin: EdgeInsets.all(20.0),
           padding: EdgeInsets.all(12.0),
           decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.red
           ),
           child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),

         ),
         Container(
           margin: EdgeInsets.all(20.0),
           padding: EdgeInsets.all(12.0),
           decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.red
           ),
           child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
         ),
         Container(
           margin: EdgeInsets.all(20.0),
           padding: EdgeInsets.all(12.0),
           decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.red
```

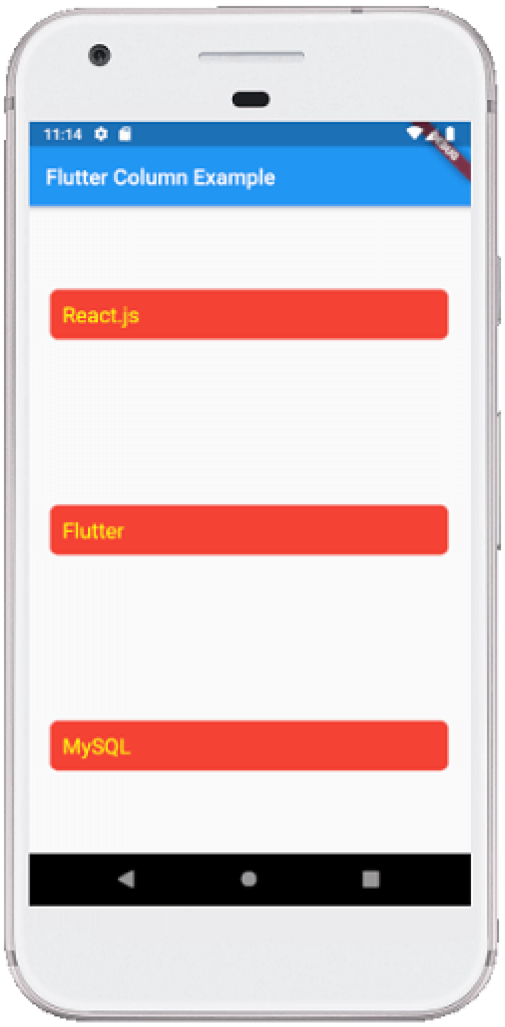
```
                ),
                child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
              )
            ]
          ),
        );
      }
    }
```

Output:

When we run this app, we should get the UI as the below screenshot.



Flutter also allows developers to align the child widget with a combination of crossAxisAlignment and mainAxisAlignment for both row and column widget. Let us take the above example of column widget where we will set mainAxisAlignment as MainAxisAlignment.spaceAround and crossAxisAlignment is CrossAxisAlignment.stretch. It will make the height of the column is equal to the height of the body. See the below screenshot.



Flutter 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. In this article, we are going to learn how to use a Text widget and how to style it in our application.

Here is a simple example to understand this widget. This example shows our project's title in the application bar and a message in the application's body.

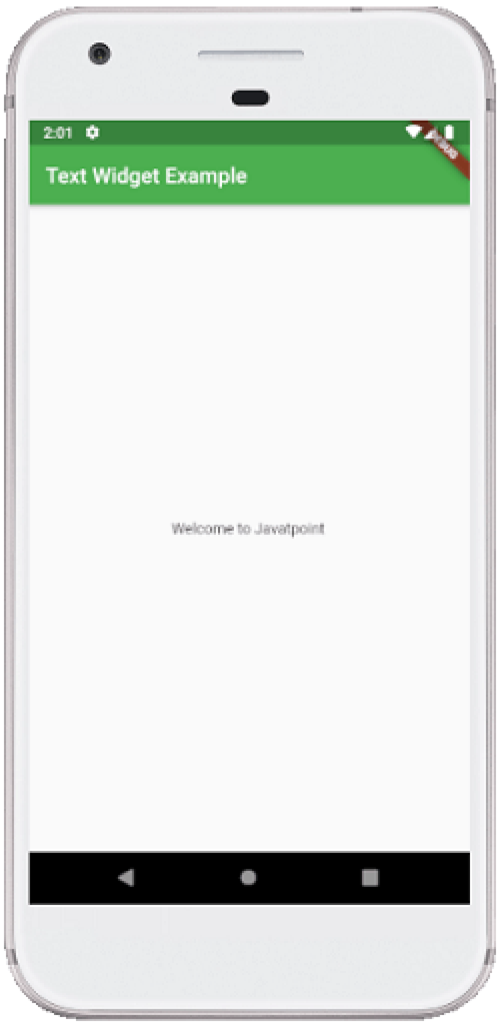
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
          primarySwatch: Colors.green,
        ),
        home: MyTextPage()
    );
  }
}
class MyTextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title:Text("Text Widget Example")
      ),
      body: Center(
          child:Text("Welcome to Javatpoint")
      ),
    );
  }
}
```

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

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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
          primarySwatch: Colors.green,
        ),
        home: MyTextPage()
    );
  }
}
class MyTextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title:Text("Text Widget Example")
      ),
      body: Center(
          child:Text("Welcome to Javatpoint")
      ),
    );
  }
}
```

In the above code, we have used a MaterialApp widget that calls the home screen using the MyTextPage() class. This class contains the scaffold widget, which has appBar and body where we have used the Text widget to display the title and body, respectively. It is a simple scenario of Text widget where we have to pass the string that we want to display on our page.

When we run this application in the emulator or device, we should get the UI similar to the below screenshot:



The following are the essential properties of the Text widget used in our application:

TextAlign: It is used to specify how our text is aligned horizontally. It also controls the text location.

TextDirection: It is used to determine how textAlign values control the layout of our text. Usually, we write text from left to right, but we can change it using this parameter.

Overflow: It is used to determine when the text will not fit in the available space. It means we have specified more text than the available space.

TextScaleFactor: It is used to determine the scaling to the text displayed by the Text widget. Suppose we have specified the text scale factor as 1.5, then our text will be 50 percent larger than the specified font size.

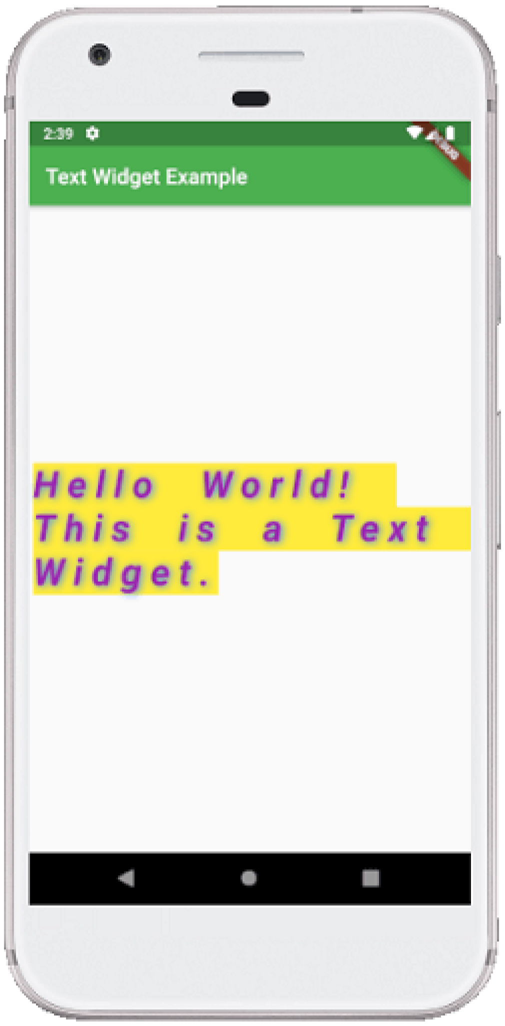
SoftWrap: It is used to determine whether or not to show all text widget content when there is not enough space available. If it is true, it will show all content. Otherwise, it will not show all content.

MaxLines: It is used to determine the maximum number of lines displayed in the text widget.
TextWidthBasis: It is used to control how the text width is defined.
TextHeightBehavior: It is used to control how the paragraph appears between the first line and descent of the last line.
Style: It is the most common property of this widget that allows developers to styling their text. It can do styling by specifying the foreground and background color, font size, font weight, letter and word spacing, locale, shadows, etc. See the table to understand it more easily:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
          primarySwatch: Colors.green,
        ),
        home: MyTextPage()
    );
  }
}
class MyTextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title:Text("Text Widget Example")
      ),
      body: Center(
          child: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)
              ]
            ),
          )
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
          primarySwatch: Colors.green,
        ),
        home: MyTextPage()
    );
  }
}
class MyTextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
     appBar: AppBar(
       title:Text("Text Widget Example")
     ),
     body: Center(
       child: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)
         ]
        ),
       )
     ),
    );
  }
}
```

Output:

When we run this application in the emulator or device, we should get the UI similar to the below screenshot:



Hello World! This is a Text Widget.

Flutter TextField

A TextField or TextBox is an input element which holds the alphanumeric data, such as name, password, address, etc. It is a GUI control element that enables the user to enter text information using a programmable code. It can be of a single-line text field (when only one line of information is required) or multiple-line text field (when more than one line of information is required).

TextField in Flutter is the most commonly used text input widget that allows users to collect inputs from the keyboard into an app. We can use the TextField widget in building forms, sending messages, creating search experiences, and many more. By default, Flutter decorated the TextField with an underline. We can also add several attributes with TextField, such as label, icon, inline hint text, and error text using an InputDecoration as the decoration. If we want to remove the decoration properties entirely, it is required to set the decoration to null.

The following code explains a demo example of TextFiled widget in Flutter:

```
TextField (
  decoration: InputDecoration(
    border: InputBorder.none,
    labelText: 'Enter Name',
    hintText: 'Enter Your Name'
  ),
);
```

```
TextField (
  decoration: InputDecoration(
    border: InputBorder.none,
    labelText: 'Enter Name',
    hintText: 'Enter Your Name'
  ),
);
```

Some of the most common attributes used with the TextField widget are as follows:

- decoration: It is used to show the decoration around TextField.
- border: It is used to create a default rounded rectangle border around TextField.
- labelText: It is used to show the label text on the selection of TextField.
- hintText: It is used to show the hint text inside TextField.
- icon: It is used to add icons directly to the TextField.

We are going to see how to use TextField widget in the Flutter app through the following steps:

Step 1: Create a Flutter project in the IDE you used. Here, I am going to use Android Studio.

Step 2: Open the project in Android Studio and navigate to the lib folder. In this folder, open the main.dart file and import the material.dart package as given below:

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

Step 3: Next, call the main MyApp class using void main run app function and then create your main widget class named as MyApp extends with StatefulWidget:

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

class MyApp extends StatefulWidget { }
```

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

class MyApp extends StatefulWidget { }
```

Step 4: Next, we need to create the Scaffold widget -> Column widget in the class widget build area as given below:

```
class MyApp extends StatefulWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
     title: Text('Flutter TextField Example'),
    ),
    body: Padding(
      padding: EdgeInsets.all(15),
      child: Column(
       children: <Widget> [


       ]
       )
      )
       )
    );
 }
}
class MyApp extends StatefulWidget {
 @override
 Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter TextField Example'),
      ),
      body: Padding(
          padding: EdgeInsets.all(15),
          child: Column(
            children: <Widget> [

            ]
            )
          )
            )
        );
  }
}
```

Step 5: Finally, create the TextField widget as the below code.

```
child: TextField(
                    obscureText: true,
                    decoration: InputDecoration(
                      border: OutlineInputBorder(),
                      labelText: 'Password',
                      hintText: 'Enter Password',
                    ),
                  ),
child: TextField(
                    obscureText: true,
                    decoration: InputDecoration(
                      border: OutlineInputBorder(),
                      labelText: 'Password',
          hintText: 'Enter Password',
         ),
        ),
```

Let us see the complete source code that contains the TextField Widget. This Flutter application takes two TextFields and one RaisedButton. After filling the details, the user clicks on the button. Since we have not specified any value in the onPressed () property of the button, it cannot print them to console.

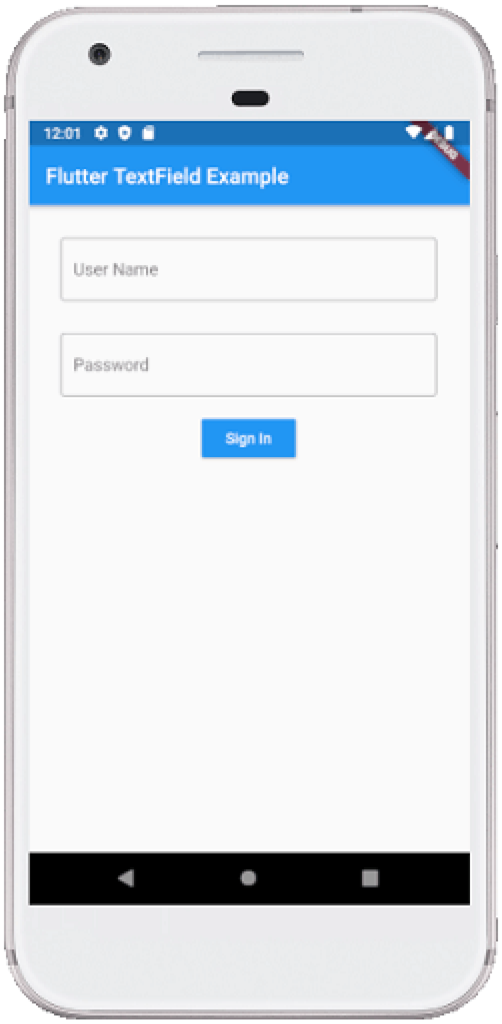
Replace the following code in the main.dart file and see the output.

```
import 'package:flutter/material.dart';
void main() {
 runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
     title: Text('Flutter TextField Example'),
    ),
    body: Padding(
      padding: EdgeInsets.all(15),
      child: Column(
       children: <Widget>[
        Padding(
         padding: EdgeInsets.all(15),
            child: TextField(
              decoration: InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'User Name',
                hintText: 'Enter Your Name',
              ),
            ),
          ),
          Padding(
            padding: EdgeInsets.all(15),
            child: TextField(
              obscureText: true,
              decoration: InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'Password',
                hintText: 'Enter Password',
              ),
            ),
          ),
          RaisedButton(
            textColor: Colors.white,
            color: Colors.blue,
            child: Text('Sign In'),
            onPressed: (){},
          )
        ],
      )
    )
  );
 }
}
```

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

void main() {
  runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter TextField Example'),
        ),
        body: Padding(
            padding: EdgeInsets.all(15),
            child: Column(
              children: <Widget>[
                Padding(
                  padding: EdgeInsets.all(15),
                  child: TextField(
                    decoration: InputDecoration(
                      border: OutlineInputBorder(),
                      labelText: 'User Name',
                      hintText: 'Enter Your Name',
                    ),
                  ),
                ),
                Padding(
                  padding: EdgeInsets.all(15),
                  child: TextField(
                    obscureText: true,
                    decoration: InputDecoration(
                      border: OutlineInputBorder(),
                      labelText: 'Password',
                      hintText: 'Enter Password',
                    ),
                  ),
                ),
                RaisedButton(
                  textColor: Colors.white,
                  color: Colors.blue,
                  child: Text('Sign In'),
                  onPressed: (){},
                )
              ],
            )
        )
    );
  }
}
```

Output

When we run the application in android emulator, we should get UI similar to the following screenshot:



How to retrieve the value of a TextField?

We know that Flutter does not have an ID like in Android for the TextField widget. Flutter allows the user to retrieve the text in mainly two ways: First is the onChanged method, and another is the controller method. Both are discussed below:

1. onChanged method: It is the easiest way to retrieve the text field value. This method store the current value in a simple variable and then use it in the TextField widget. Below is the sample example:

```
String value = "";
TextField(
  onChanged: (text) {
    value = text;
  },
)
String value = "";
TextField(
  onChanged: (text) {
    value = text;
  },
)
```

2. Controller method: It is a popular method to retrieve text field value using TextEditingController. It will be attached to the TextField widget and then listen to change and control the widget's text value. Below is the sample code:

```
TextEditingController mycontroller = TextEditingController();
TextField(
  controller: mycontroller,
)
TextEditingController mycontroller = TextEditingController();
TextField(
  controller: mycontroller,
)
```

Sample code for listening to the changes.

```
controller.addListener(() {
  // Do something here
});
controller.addListener(() {
  // Do something here
});
```

Sample code to get or set the value.

```
print(controller.text); // Print current value
controller.text = "Demo Text"; // Set new value
print(controller.text); // Print current value
controller.text = "Demo Text"; // Set new value
```

Let us see the second way in detail to retrieve the text field value in Flutter application with the help of following steps:

Create a TextEditingController.
Attach the TextEditingController to a TextField using controller property.
Retrieve the value of the TextField by using the text() method provided by the TextEditingController.
Now, create a new Flutter project in your IDE and open the main.dart file. Replace the below code in the main.dart file. In this example, we are going to display the alert dialog with the current value of the text field when the user taps on a button.

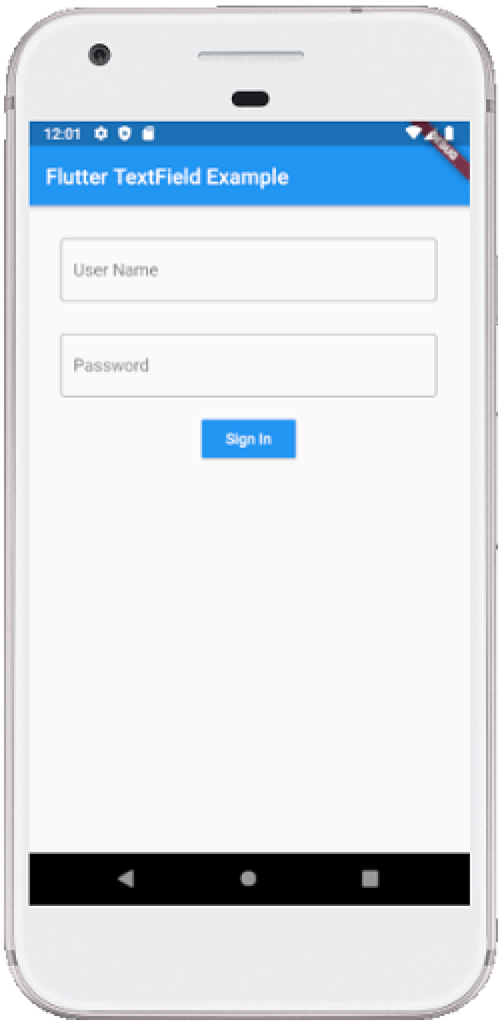
```
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
  TextEditingController nameController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  @override
  Widget build(BuildContext context) {
```

```
return Scaffold(
  appBar: AppBar(
    title: Text('Flutter TextField Example'),
  ),
  body: Padding(
    padding: EdgeInsets.all(15),
    child: Column(
      children: <Widget>[
        Padding(
          padding: EdgeInsets.all(15),
          child: TextField(
            controller: nameController,
            decoration: InputDecoration(
              border: OutlineInputBorder(),
              labelText: 'User Name',
              hintText: 'Enter Your Name',
            ),
          ),
        ),
        Padding(
          padding: EdgeInsets.all(15),
          child: TextField(
            controller: passwordController,
            obscureText: true,
            decoration: InputDecoration(
              border: OutlineInputBorder(),
              labelText: 'Password',
              hintText: 'Enter Password',
            ),
          ),
        ),
            RaisedButton(
              textColor: Colors.white,
              color: Colors.blue,
              child: Text('Sign In'),
              onPressed: (){
                return showDialog(
                  context: context,
                  builder: (context) {
                    return AlertDialog(
                      title: Text("Alert Message"),
                      // Retrieve the text which the user has entered by
                      // using the TextEditingController.
                      content: Text(nameController.text),
                      actions: <Widget>[
                        new FlatButton(
                          child: new Text('OK'),
                          onPressed: () {
                            Navigator.of(context).pop();
                          },
                        )
                      ],
                    );
                  },
                );
              },
            )
          ],
        )
      )
    );
  }
```

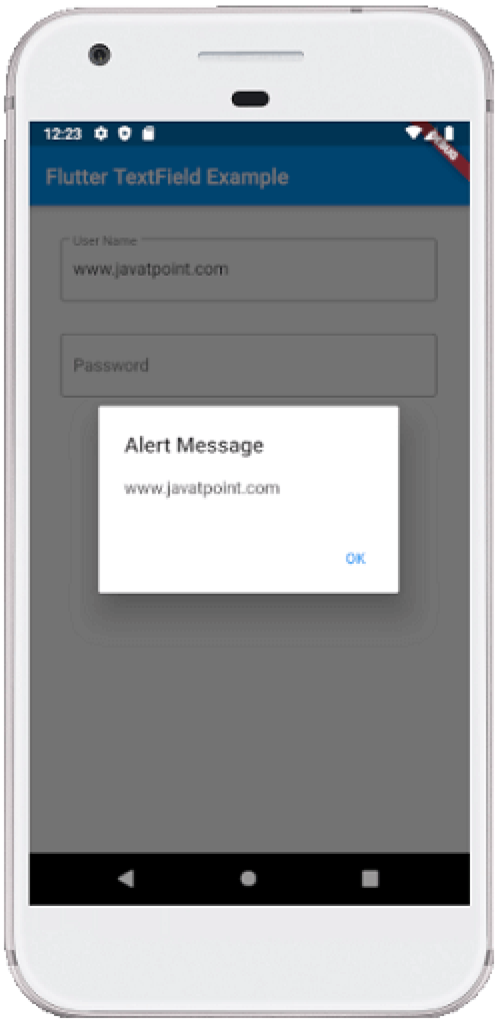
```
}
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
  TextEditingController nameController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter TextField Example'),
        ),
        body: Padding(
            padding: EdgeInsets.all(15),
            child: Column(
              children: <Widget>[
                Padding(
                  padding: EdgeInsets.all(15),
                  child: TextField(
                    controller: nameController,
                    decoration: InputDecoration(
                      border: OutlineInputBorder(),
      labelText: 'User Name',
      hintText: 'Enter Your Name',
    ),
  ),
),
Padding(
  padding: EdgeInsets.all(15),
  child: TextField(
    controller: passwordController,
    obscureText: true,
    decoration: InputDecoration(
      border: OutlineInputBorder(),
      labelText: 'Password',
      hintText: 'Enter Password',
    ),
  ),
),
RaisedButton(
  textColor: Colors.white,
  color: Colors.blue,
  child: Text('Sign In'),
  onPressed: (){
    return showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text("Alert Message"),
          // Retrieve the text which the user has entered by
          // using the TextEditingController.
          content: Text(nameController.text),
          actions: <Widget>[
                        new FlatButton(
                          child: new Text('OK'),
                          onPressed: () {
                            Navigator.of(context).pop();
                          },
                        )
                      ],
                    );
                  },
                );
              },
            )
          ],
        )
      )
    );
  }
}
```

Output:

When we run the application in android emulator, we should get UI similar to the following screenshot:



Click inside the text box and add values, as shown in the field. When we press the sign-in button, an alert dialog box appeared containing the text that the user has entered into the field. If we click on the OK button, the alert dialog will disappear. Look into the below screenshot:



Flutter Buttons

Buttons are the graphical control element that **provides 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.

Buttons are the Flutter widgets, which is a part of the material design library. Flutter provides several types of buttons that have different shapes, styles, and features.

Features of Buttons

The standard features of a button in Flutter are given below:

1. We can easily apply themes on buttons, shapes, color, animation, and behavior.
2. We can also theme icons and text inside the button.
3. Buttons can be composed of different child widgets for different characteristics.

Types of Flutter Buttons

Following are the different types of button available in Flutter:

- Flat Button
- Raised Button
- Floating Button
- Drop Down Button
- Icon Button
- Inkwell Button
- PopupMenu Button
- Outline Button

Let us discuss each button in detail.

1. Flat Button

It is a **text label button** that does not have much decoration and displayed **without any elevation**. The flat button has two required properties that are: **child and onPressed()**. It is mostly used in toolbars, dialogs, or inline with other content. By default, the flat button has no color, and its text is black. But, we can use color to the button and text using **color and textColor** attributes, respectively.

Example:

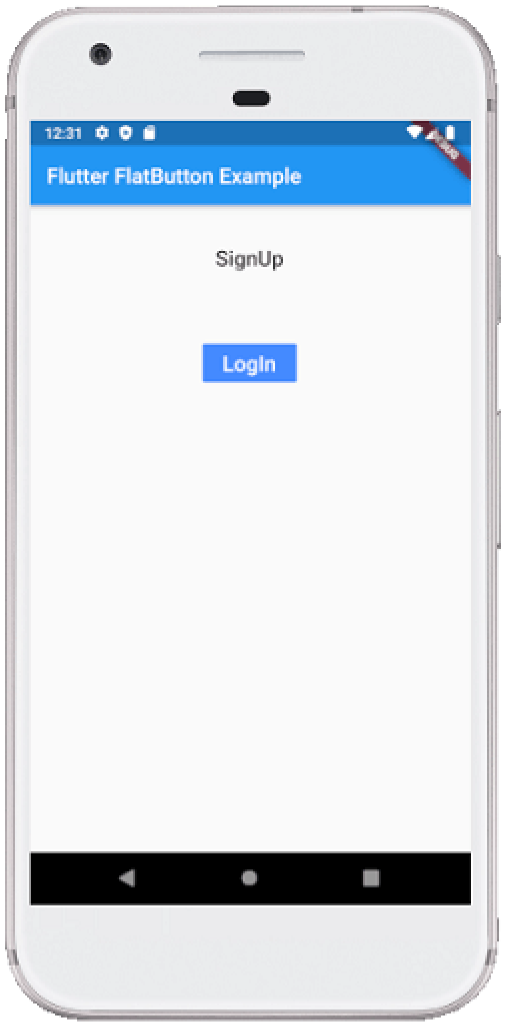
Open the **main.dart** file and replace it with the below code.

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

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

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

class _MyAppState extends State<MyApp> {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
     appBar: AppBar(
      title: Text('Flutter FlatButton Example'),
     ),
     body: Center(child: Column(children: <Widget>[
      Container(
       margin: EdgeInsets.all(25),
       child: FlatButton(
        child: Text('SignUp', style: TextStyle(fontSize: 20.0),),
        onPressed: () {},
       ),
      ),
      Container(
       margin: EdgeInsets.all(25),
       child: FlatButton(
        child: Text('LogIn', style: TextStyle(fontSize: 20.0),),
        color: Colors.blueAccent,
        textColor: Colors.white,
        onPressed: () {},
       ),
      ),
     ]
```

```
        ))
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Flutter FlatButton Example'),
          ),
          body: Center(child: Column(children: <Widget>[
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text('SignUp', style: TextStyle(fontSize: 20.0),),
                onPressed: () {},
              ),
            ),
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text('LogIn', style: TextStyle(fontSize: 20.0),),
                color: Colors.blueAccent,
                textColor: Colors.white,
                onPressed: () {},
              ),
            ),
          ]
        ))
      ),
    );
  }
}
```

Output:

If we run this app, we will see the following screen:



2. Raised Button

It is a button, which is based on the material widget and has a **rectangular body**. It is similar to a flat button, but it **has an elevation** that will increases when the button is pressed. It adds dimension to the UI along Z-axis. It has several properties like text color, shape, padding, button color, the color of a button when disabled, animation time, elevation, etc.

This button has **two callback functions**.

onPressed(): It is triggered when the button is pressed.

onLongPress(): It is triggered when the button is long pressed.

It is to note that this button is in a **disabled state** if onPressed() and onLongPressed() callbacks are not specified.

Example:

Open the **main.dart** file and replace it with the below code.

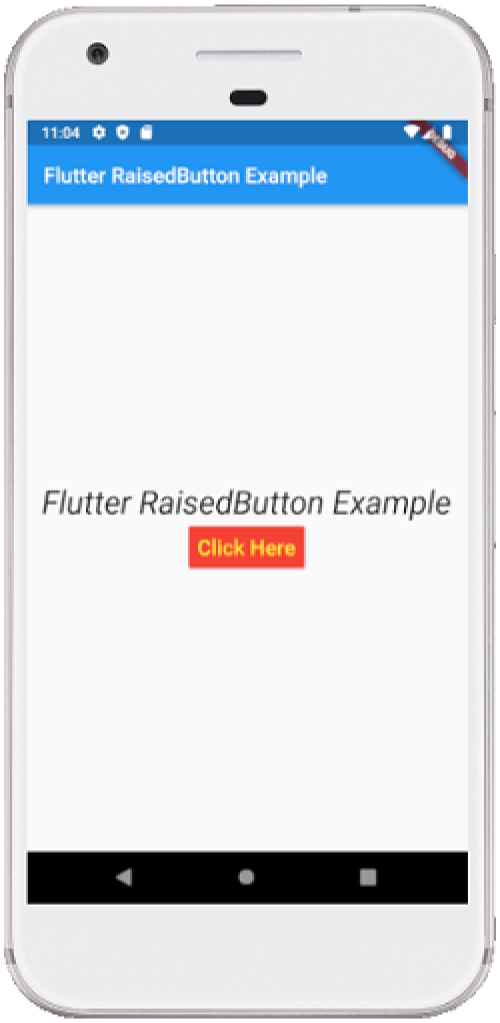
```
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  String msg = 'Flutter RaisedButton Example';
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Flutter RaisedButton Example'),
          ),
        body: Container(
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(msg, style: TextStyle(fontSize: 30, fontStyle: FontStyle.italic),),
                RaisedButton(
              child: Text("Click Here", style: TextStyle(fontSize: 20),),
              onPressed: _changeText,
              color: Colors.red,
              textColor: Colors.yellow,
              padding: EdgeInsets.all(8.0),
              splashColor: Colors.grey,
             )
            ],
           ),
          ),
         ),
        ),
       );
      }
      _changeText() {
       setState(() {
        if (msg.startsWith('F')) {
         msg = 'We have learned FlutterRaised button example.';
        } else {
         msg = 'Flutter RaisedButton Example';
        }
       });
      }
     }
```

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

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

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

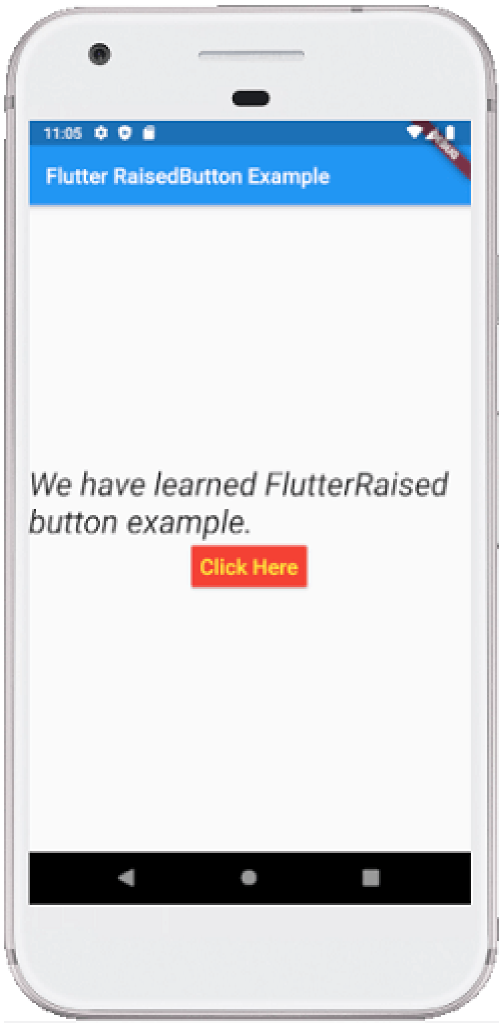
```
class _MyAppState extends State<MyApp> {
 String msg = 'Flutter RaisedButton Example';
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
     appBar: AppBar(
      title: Text('Flutter RaisedButton Example'),
     ),
    body: Container(
     child: Center(
      child: Column(
       mainAxisAlignment: MainAxisAlignment.center,
       children: [
        Text(msg, style: TextStyle(fontSize: 30, fontStyle: FontStyle.italic),),
        RaisedButton(
         child: Text("Click Here", style: TextStyle(fontSize: 20),),
         onPressed: _changeText,
         color: Colors.red,
         textColor: Colors.yellow,
         padding: EdgeInsets.all(8.0),
         splashColor: Colors.grey,
        )
       ],
      ),
     ),
    ),
   ),
  );
 }
 _changeText() {
  setState(() {
   if (msg.startsWith('F')) {
          msg = 'We have learned FlutterRaised button example.';
        } else {
          msg = 'Flutter RaisedButton Example';
        }
      });
    }
  }
```

Output:

When we run this example, it will give the below screenshot. If we click on the "**Click Here**" button, it will change the text message. Show the second screenshot.



Flutter RaisedButton Example



We have learned FlutterRaised button example.

3. Floating Action Button

A FAB button is a **circular icon button** that triggers the primary action in our application. It is the most used button in today's applications. We can use this button for adding, refreshing, or sharing the content. Flutter suggests using at most one FAB button per screen. There are two types of Floating Action Button:

FloatingActionButton: It creates a simple circular floating button with a child widget inside it. It must have a child parameter to display a widget.

FloatingActionButton.extended: It creates a wide floating button along with an icon and a label inside it. Instead of a child, it uses labels and icon parameters.

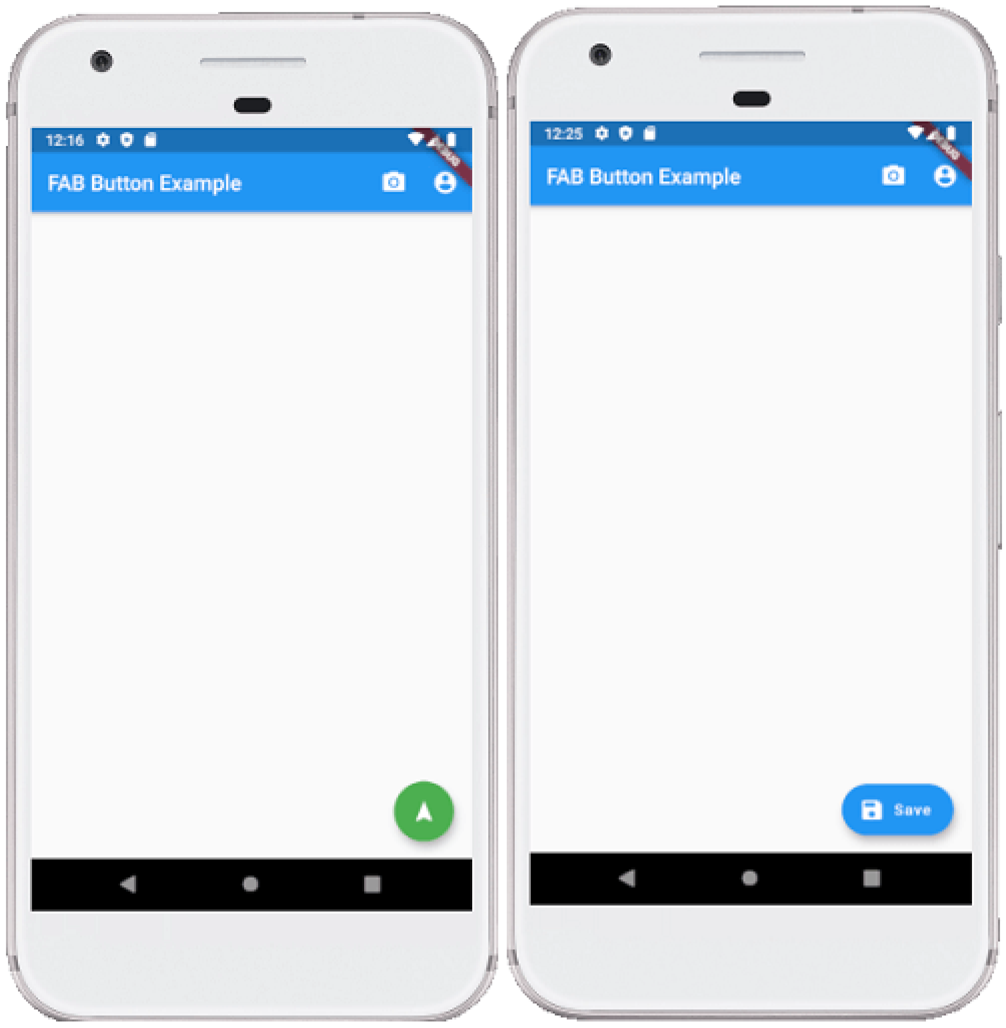
Example:

Open the **main.dart** file and replace it with the below code.

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

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

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

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold(
      appBar: AppBar(
        title: Text("FAB Button Example"),
        backgroundColor: Colors.blue,
        actions: <Widget>[
          IconButton(icon: Icon(Icons.camera_alt), onPressed: () => {}),
          IconButton(icon: Icon(Icons.account_circle), onPressed: () => {})
        ],
      ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.navigation),
        backgroundColor: Colors.green,
        foregroundColor: Colors.white,
        onPressed: () => {},
      ),
      /*floatingActionButton:FloatingActionButton.extended(
        onPressed: () {},
        icon: Icon(Icons.save),
        label: Text("Save"),
      ), */
    ),
    );
  }
}
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold(
      appBar: AppBar(
        title: Text("FAB Button Example"),
        backgroundColor: Colors.blue,
        actions: <Widget>[
          IconButton(icon: Icon(Icons.camera_alt), onPressed: () => {}),
          IconButton(icon: Icon(Icons.account_circle), onPressed: () => {})
        ],
      ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.navigation),
        backgroundColor: Colors.green,
        foregroundColor: Colors.white,
        onPressed: () => {},
      ),
      /*floatingActionButton:FloatingActionButton.extended(
        onPressed: () {},
        icon: Icon(Icons.save),
        label: Text("Save"),
      ), */
    ),
    );
  }
}
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot. The second image is an output of the **FAB.extended** button. Its coding can be seen in the above code's comment section.



4. DropDown Button

A drop-down button is used to create a nice overlay on the screen that allows the user to select any item from multiple options. Flutter allows a simple way to implement a drop-down box or drop-down button. This button shows the currently selected item and an arrow that opens a menu to select an item from multiple options.

Flutter provides a **DropdownButton widget** to implement a drop-down list. We can place it anywhere in our app.

Example

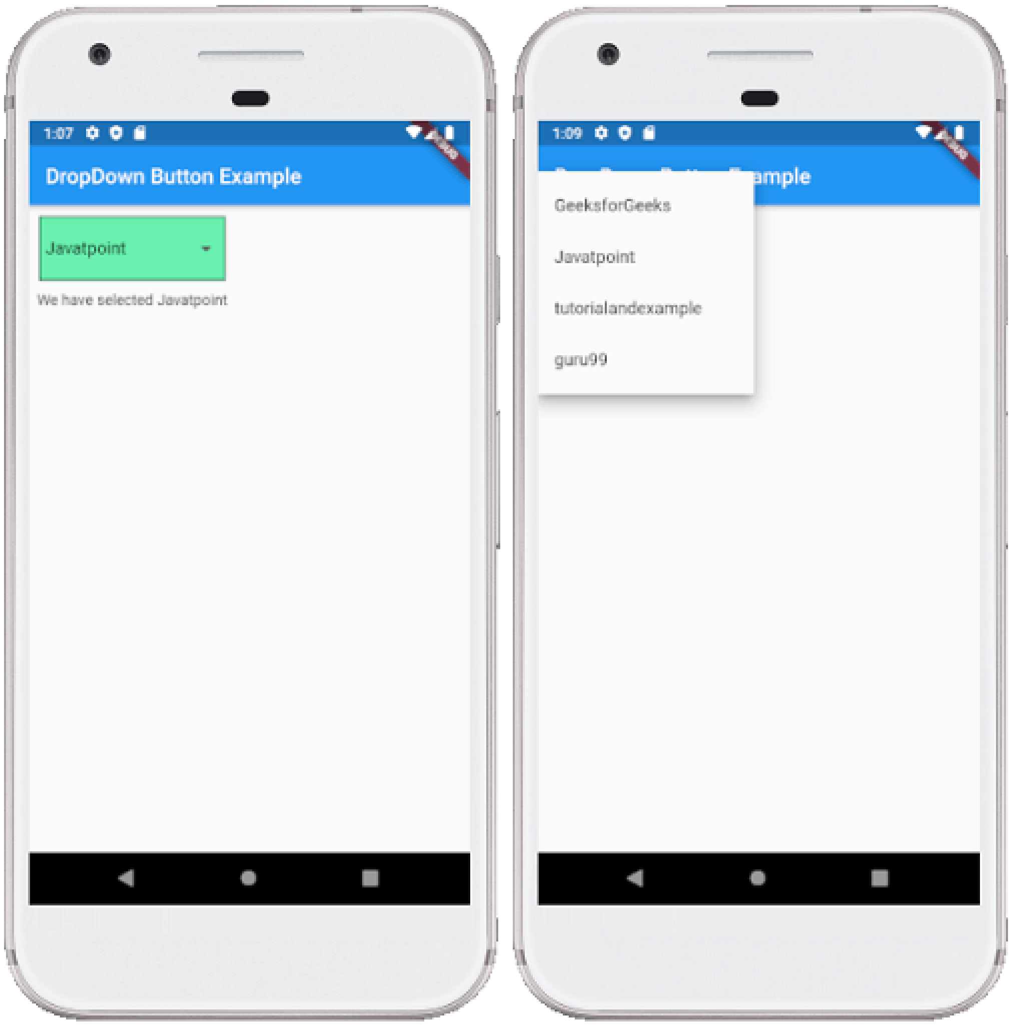
Open the **main.dart** file and replace it with the below code.

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

void main() => runApp(MaterialApp(
  home: MyApp(),
));

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

class _MyAppState extends State<MyApp> {
  List<ListItem> _dropdownItems = [
    ListItem(1, "GeeksforGeeks"),
    ListItem(2, "Javatpoint"),
    ListItem(3, "tutorialandexample"),
    ListItem(4, "guru99")
  ];

  List<DropdownMenuItem<ListItem>> _dropdownMenuItems;
  ListItem _itemSelected;

  void initState() {
    super.initState();
    _dropdownMenuItems = buildDropDownMenuItems(_dropdownItems);
    _itemSelected = _dropdownMenuItems[1].value;
  }

  List<DropdownMenuItem<ListItem>> buildDropDownMenuItems(List listItems) {

    List<DropdownMenuItem<ListItem>> items = List();
    for (ListItem listItem in listItems) {
      items.add(
        DropdownMenuItem(
        child: Text(listItem.name),
        value: listItem,
      ),
    );
  }
  return items;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("DropDown Button Example"),
    ),
    body: Column(
      children: <Widget>[
        Padding(
          padding: const EdgeInsets.all(10.0),
          child: Container(
            padding: const EdgeInsets.all(5.0),
            decoration: BoxDecoration(
                color: Colors.greenAccent,
                border: Border.all()),
            child: DropdownButtonHideUnderline(
              child: DropdownButton(
                  value: _itemSelected,
                  items: _dropdownMenuItems,
                  onChanged: (value) {
                    setState(() {
                      _itemSelected = value;
                    });
                  }),
            ),
          ),
          ),
          Text("We have selected ${_itemSelected.name}"),
        ],
      ),
    );
  }
}

class ListItem {
  int value;
  String name;

  ListItem(this.value, this.name);
}
import 'package:flutter/material.dart';

void main() => runApp(MaterialApp(
  home: MyApp(),
));

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

class _MyAppState extends State<MyApp> {
  List<ListItem> _dropdownItems = [
    ListItem(1, "GeeksforGeeks"),
    ListItem(2, "Javatpoint"),
    ListItem(3, "tutorialandexample"),
    ListItem(4, "guru99")
  ];

  List<DropdownMenuItem<ListItem>> _dropdownMenuItems;
```

```
ListItem _itemSelected;

void initState() {
  super.initState();
  _dropdownMenuItems = buildDropDownMenuItems(_dropdownItems);
  _itemSelected = _dropdownMenuItems[1].value;
}

List<DropdownMenuItem<ListItem>> buildDropDownMenuItems(List listItems) {

  List<DropdownMenuItem<ListItem>> items = List();
  for (ListItem listItem in listItems) {
    items.add(
      DropdownMenuItem(
        child: Text(listItem.name),
        value: listItem,
      ),
    );
  }
  return items;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("DropDown Button Example"),
    ),
    body: Column(
      children: <Widget>[
        Padding(
          padding: const EdgeInsets.all(10.0),
          child: Container(
            padding: const EdgeInsets.all(5.0),
              decoration: BoxDecoration(
                color: Colors.greenAccent,
                border: Border.all()),
              child: DropdownButtonHideUnderline(
                child: DropdownButton(
                  value: _itemSelected,
                  items: _dropdownMenuItems,
                  onChanged: (value) {
                    setState(() {
                      _itemSelected = value;
                    });
                  }),
              ),
            ),
          ),
          Text("We have selected ${_itemSelected.name}"),
        ],
      ),
    );
  }
}

class ListItem {
  int value;
  String name;

  ListItem(this.value, this.name);
}
```

Output

Run the application in android emulator, and it will give the UI similar to the following screenshot. The second image is an output of the list contains in the drop drown button.



5. Icon Button

An IconButton is a **picture printed** on the Material widget. It is a useful widget that gives the Flutter UI a material design feel. We can also customize the look and feel of this button. In simple terms, it is an icon that reacts when the user will touch it.

Example:

Open the **main.dart** file and replace it with the below code.

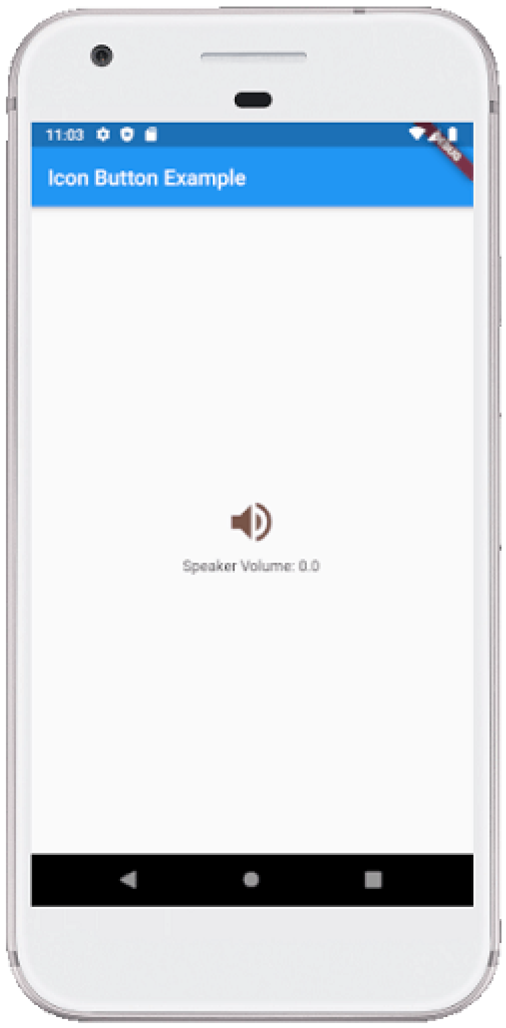
```
import 'package:flutter/material.dart';

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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Icon Button Example"),
        ),
        body: Center(
          child: MyStatefulWidget(),
        ),
      ),
    );
  }
}
double _speakervolume = 0.0;

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

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

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  Widget build(BuildContext context) {
    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        IconButton(
          icon: Icon(Icons.volume_up),
          iconSize: 50,
          color: Colors.brown,
          tooltip: 'Increase volume by 5',
          onPressed: () {
            setState(() {
              _speakervolume += 5;
            });
          },
        ),
        Text('Speaker Volume: $_speakervolume')
      ],
    );
  }
}
```

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Icon Button Example"),
        ),
        body: Center(
          child: MyStatefulWidget(),
        ),
      ),
    );
  }
}
double _speakervolume = 0.0;

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

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

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  Widget build(BuildContext context) {
    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        IconButton(
          icon: Icon(Icons.volume_up),
          iconSize: 50,
          color: Colors.brown,
          tooltip: 'Increase volume by 5',
          onPressed: () {
            setState(() {
              _speakervolume += 5;
            });
          },
        ),
        Text('Speaker Volume: $_speakervolume')
      ],
    );
  }
}
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot. When we press the **volume button**, it will always increase by 5.



6. Inkwell Button

InkWell button is a material design concept, which is used for **touch response**. This widget comes under the Material widget where the ink reactions are actually painted. It creates the app UI interactive by adding gesture feedback. It is mainly used for adding **splash ripple effect**.

Example:

Open the **main.dart** file and replace it with the below code.

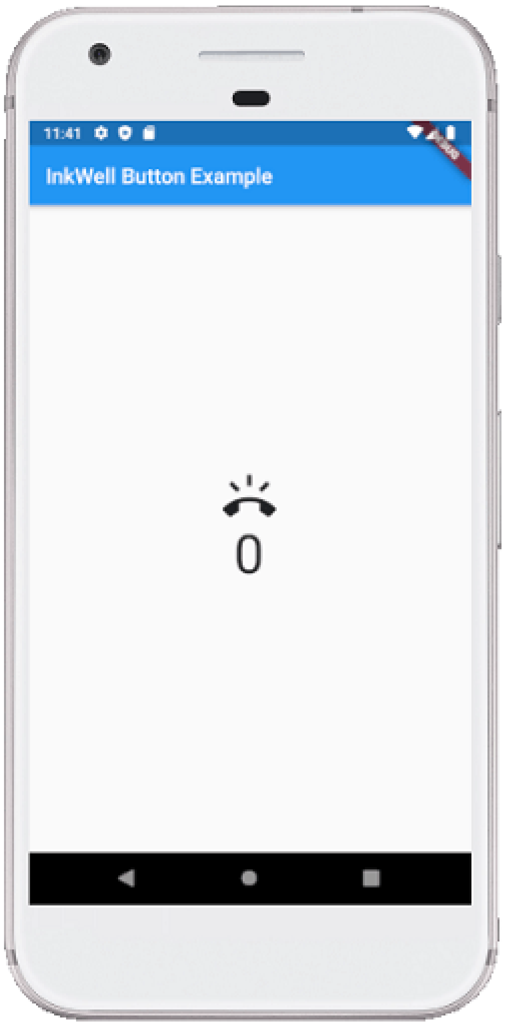
```
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  int _volume = 0;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('InkWell Button Example'),
        ),
        body: Center(
          child: new Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              InkWell(
                splashColor: Colors.green,
                highlightColor: Colors.blue,
                child: Icon(Icons.ring_volume, size: 50),
                onTap: () {
                  setState(() {
                    _volume += 2;
                  });
                },
              ),
              Text (
                  _volume.toString(),
                  style: TextStyle(fontSize: 50)
              ),
          ],
        ),
      ),
    ),
  );
 }
}
```

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

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

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

class _MyAppState extends State<MyApp> {
 int _volume = 0;

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(
     title: Text('InkWell Button Example'),
    ),
    body: Center(
     child: new Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
       InkWell(
        splashColor: Colors.green,
        highlightColor: Colors.blue,
        child: Icon(Icons.ring_volume, size: 50),
          onTap: () {
            setState(() {
              _volume += 2;
            });
          },
        ),
        Text (
            _volume.toString(),
            style: TextStyle(fontSize: 50)
        ),
      ],
    ),
  ),
),
);
}
}
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot. Every time we press the ring volume button, it will increase the volume by 2.



7. PopupMenu Button

It is a button that **displays the menu** when it is pressed and then calls the **onSelected** method the menu is dismissed. It is because the item from the multiple options is selected. This button contains a text and an image. It will mainly use with **Settings** menu to list all options. It helps in making a great user experience.

Example:

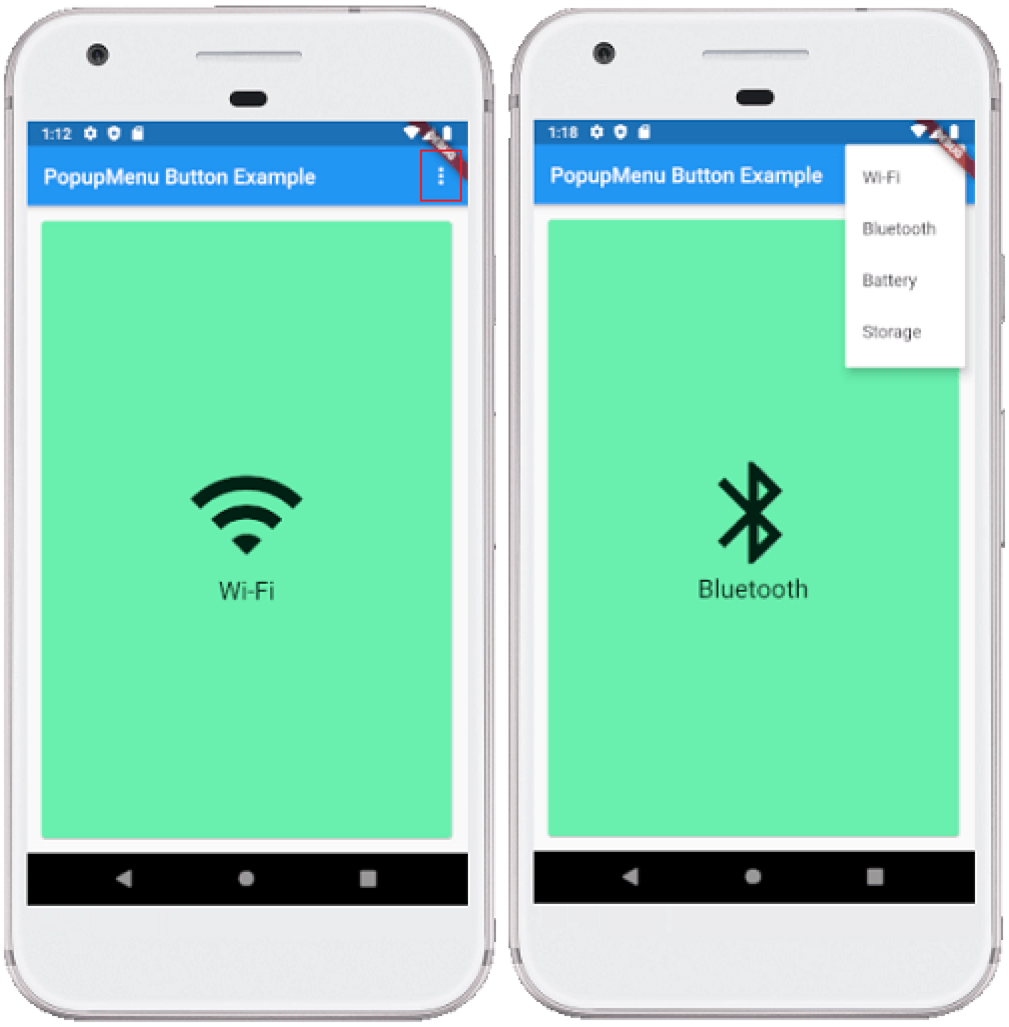
Open the **main.dart** file and replace it with the below code.

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

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

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

class _MyAppState extends State<MyApp> {
 Choice _selectedOption = choices[0];

 void _select(Choice choice) {
  setState(() {
   _selectedOption = choice;
  });
 }
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(
     title: const Text('PopupMenu Button Example'),
     actions: <Widget>[
      PopupMenuButton<Choice>(
       onSelected: _select,
       itemBuilder: (BuildContext context) {
        return choices.skip(0).map((Choice choice) {
         return PopupMenuItem<Choice>(
          value: choice,
          child: Text(choice.name),
         );
        }).toList();
       },
      ),
     ],
        ),
        body: Padding(
          padding: const EdgeInsets.all(10.0),
          child: ChoiceCard(choice: _selectedOption),
        ),
      ),
    );
  }
}

class Choice {
  const Choice({this.name, this.icon});
  final String name;
  final IconData icon;
}

const List<Choice> choices = const <Choice>[
  const Choice(name: 'Wi-Fi', icon: Icons.wifi),
  const Choice(name: 'Bluetooth', icon: Icons.bluetooth),
  const Choice(name: 'Battery', icon: Icons.battery_alert),
  const Choice(name: 'Storage', icon: Icons.storage),
];

class ChoiceCard extends StatelessWidget {
  const ChoiceCard({Key key, this.choice}) : super(key: key);

  final Choice choice;

  @override
  Widget build(BuildContext context) {
    final TextStyle textStyle = Theme.of(context).textTheme.headline;
    return Card(
      color: Colors.greenAccent,
      child: Center(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            Icon(choice.icon, size: 115.0, color: textStyle.color),
            Text(choice.name, style: textStyle),
          ],
        ),
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  Choice _selectedOption = choices[0];

  void _select(Choice choice) {
    setState(() {
      _selectedOption = choice;
    });
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('PopupMenu Button Example'),
          actions: <Widget>[
            PopupMenuButton<Choice>(
              onSelected: _select,
              itemBuilder: (BuildContext context) {
                return choices.skip(0).map((Choice choice) {
                  return PopupMenuItem<Choice>(
                    value: choice,
                    child: Text(choice.name),
                  );
                }).toList();
              },
            ),
          ],
        ),
        body: Padding(
          padding: const EdgeInsets.all(10.0),
          child: ChoiceCard(choice: _selectedOption),
        ),
      ),
    );
  }
}

class Choice {
  const Choice({this.name, this.icon});
  final String name;
  final IconData icon;
}

const List<Choice> choices = const <Choice>[
  const Choice(name: 'Wi-Fi', icon: Icons.wifi),
  const Choice(name: 'Bluetooth', icon: Icons.bluetooth),
  const Choice(name: 'Battery', icon: Icons.battery_alert),
  const Choice(name: 'Storage', icon: Icons.storage),
];

class ChoiceCard extends StatelessWidget {
  const ChoiceCard({Key key, this.choice}) : super(key: key);

  final Choice choice;

  @override
  Widget build(BuildContext context) {
    final TextStyle textStyle = Theme.of(context).textTheme.headline;
    return Card(
      color: Colors.greenAccent,
      child: Center(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            Icon(choice.icon, size: 115.0, color: textStyle.color),
            Text(choice.name, style: textStyle),
          ],
        ),
      ),
    );
  }
}
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot. When we click the **three dots** shown at the top left corner of the screen, it will pop up the multiple options. Here, we can select any option, and it will keep it in the card, as shown in the second image.



8. Outline Button

It is similar to the flat button, but it contains a thin grey rounded rectangle border. Its outline border is defined by the shape attribute.

Example:

Open the **main.dart** file and replace it with the below code.

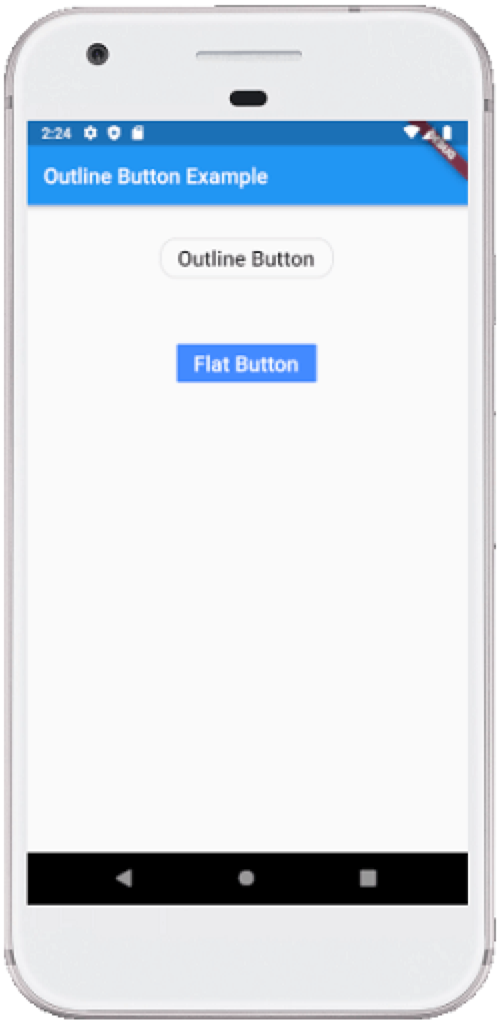
```
import 'package:flutter/material.dart';

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

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

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Outline Button Example'),
          ),
          body: Center(child: Column(children: <Widget>[
            Container(
              margin: EdgeInsets.all(25),
              child: OutlineButton(
                child: Text("Outline Button", style: TextStyle(fontSize: 20.0),),
                highlightedBorderColor: Colors.red,
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(15)),
                onPressed: () {},
              ),
            ),
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text('Flat Button', style: TextStyle(fontSize: 20.0),),
                color: Colors.blueAccent,
                textColor: Colors.white,
                onPressed: () {},
              ),
            ),
          ]
          ))
        ),
      );
    }
  }
```

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

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

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

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Outline Button Example'),
          ),
          body: Center(child: Column(children: <Widget>[
            Container(
              margin: EdgeInsets.all(25),
              child: OutlineButton(
                child: Text("Outline Button", style: TextStyle(fontSize: 20.0),),
                highlightedBorderColor: Colors.red,
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(15)),
                onPressed: () {},
              ),
            ),
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text('Flat Button', style: TextStyle(fontSize: 20.0),),
                color: Colors.blueAccent,
                textColor: Colors.white,
                onPressed: () {},
              ),
            ),
          ]
          ))
      ),
    );
  }
}
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot.



Button Bar

Flutter provides the flexibility to **arrange the buttons in a bar or a row**. ButtonBar widget contains three properties: **alignment, children, and mainAxisSize**.

- Alignment is used to present the aligning option to the entire button bar widget.
- Children attribute is used to take the number of buttons in a bar.
- mainAxisSize attribute is used to provide the horizontal space for the button bar.

Example:

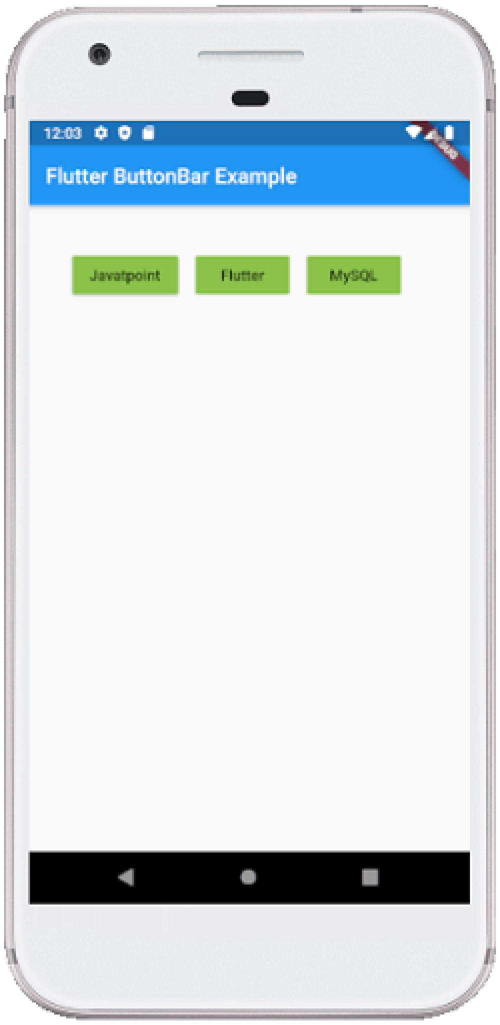
Open the **main.dart** file and replace it with the below code.

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

void main() {
  runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter ButtonBar Example'),
        ),
        body: Padding(
            padding: EdgeInsets.all(10),
            child: Column(
              children: <Widget>[
                Padding(
                  padding: EdgeInsets.all(15),
                  child: new ButtonBar(
                    mainAxisSize: MainAxisSize.min,
                    children: <Widget>[
                      RaisedButton(
                        child: new Text('Javatpoint'),
                        color: Colors.lightGreen,
                          onPressed: () {/** */},
                      ),
                      FlatButton(
                        child: Text('Flutter'),
                        color: Colors.lightGreen,
                        onPressed: () {/** */},
                      ),
                      FlatButton(
                        child: Text('MySQL'),
                        color: Colors.lightGreen,
                        onPressed: () {/** */},
                      ),
                    ],
                  ),
                ),
              ],
            )
        )
    );
  }
}
```

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

void main() {
  runApp(MaterialApp( home: MyApp(),));
}

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

class _State extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter ButtonBar Example'),
        ),
        body: Padding(
          padding: EdgeInsets.all(10),
          child: Column(
            children: <Widget>[
              Padding(
                padding: EdgeInsets.all(15),
                child: new ButtonBar(
                  mainAxisSize: MainAxisSize.min,
                  children: <Widget>[
                    RaisedButton(
                      child: new Text('Javatpoint'),
                      color: Colors.lightGreen,
                        onPressed: () {/** */},
                    ),
                    FlatButton(
                      child: Text('Flutter'),
                      color: Colors.lightGreen,
                      onPressed: () {/** */},
                    ),
                    FlatButton(
                      child: Text('MySQL'),
                      color: Colors.lightGreen,
                      onPressed: () {/** */},
                    ),
                  ],
                ),
              ),
            ],
          )
        )
      );
    }
  }
```

Output:

Run the application in android emulator, and it will give the UI similar to the following screenshot. Here, we can see that the three buttons are placed in a horizontal bar or row.



Flutter Forms

Forms are an integral part of all modern mobile and web applications. It is mainly used to interact with the app as well as gather information from the users. They can perform many tasks, which depend on the nature of your business requirements and logic, such as authentication of the user, adding user, searching, filtering, ordering, booking, etc. A form can contain text fields, buttons, checkboxes, radio buttons, etc.

Creating Form

Flutter provides a Form widget to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields.

The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Let us create a form. First, create a Flutter project and replace the following code in the main.dart file. In this code snippet, we have created a custom class named MyCustomForm. Inside this class, we define a global key as _formKey. This key holds a FormState and can use to retrieve the form widget. Inside the build method of this class, we have added some custom style and use the TextFormField widget to provide the form fields such as name, phone number, date of birth, or just a normal field. Inside the TextFormField, we have used InputDecoration that provides the look and feel of your form properties such as borders, labels, icons, hint, styles, etc. Finally, we have added a button to submit the form.

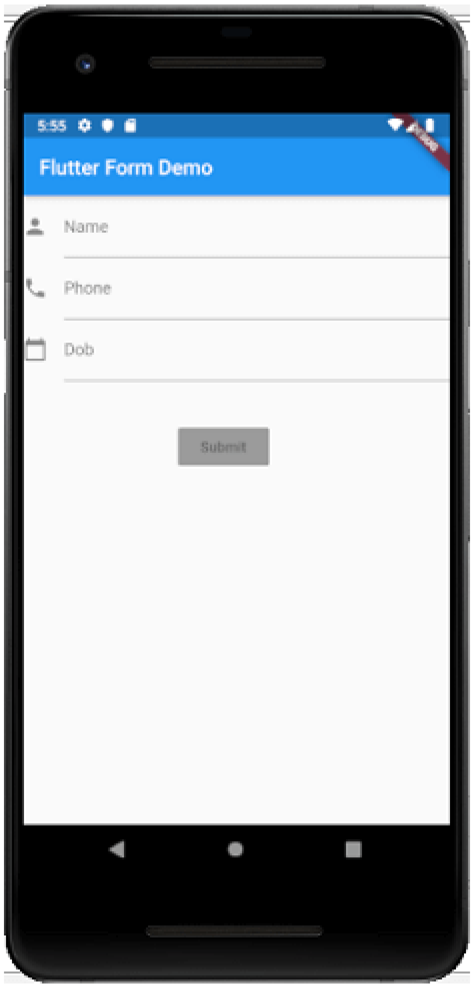
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final appTitle = 'Flutter Form Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: Text(appTitle),
        ),
        body: MyCustomForm(),
      ),
    );
  }
}
// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
// Create a corresponding State class. This class holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    // Build a Form widget using the _formKey created above.
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          TextFormField(
            decoration: const InputDecoration(
              icon: const Icon(Icons.person),
              hintText: 'Enter your name',
              labelText: 'Name',
            ),
          ),
          TextFormField(
            decoration: const InputDecoration(
              icon: const Icon(Icons.phone),
              hintText: 'Enter a phone number',
              labelText: 'Phone',
            ),
          ),
          TextFormField(
            decoration: const InputDecoration(
            icon: const Icon(Icons.calendar_today),
            hintText: 'Enter your date of birth',
            labelText: 'Dob',
            ),
           ),
          new Container(
              padding: const EdgeInsets.only(left: 150.0, top: 40.0),
              child: new RaisedButton(
                child: const Text('Submit'),
                  onPressed: null,
              )),
        ],
      ),
    );
  }
}
import 'package:flutter/material.dart';

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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final appTitle = 'Flutter Form Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: Text(appTitle),
        ),
        body: MyCustomForm(),
      ),
    );
  }
}
// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
// Create a corresponding State class. This class holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    // Build a Form widget using the _formKey created above.
```

```
return Form(
  key: _formKey,
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      TextFormField(
        decoration: const InputDecoration(
          icon: const Icon(Icons.person),
          hintText: 'Enter your name',
          labelText: 'Name',
        ),
      ),
      TextFormField(
        decoration: const InputDecoration(
          icon: const Icon(Icons.phone),
          hintText: 'Enter a phone number',
          labelText: 'Phone',
        ),
      ),
      TextFormField(
        decoration: const InputDecoration(
        icon: const Icon(Icons.calendar_today),
        hintText: 'Enter your date of birth',
        labelText: 'Dob',
        ),
       ),
      new Container(
          padding: const EdgeInsets.only(left: 150.0, top: 40.0),
          child: new RaisedButton(
            child: const Text('Submit'),
              onPressed: null,
          )),
        ],
      ),
    );
  }
}
```

Output

Now, run the app, you can see the following screen in your Android Emulator. This form contains three field name, phone number, date of birth, and submit button.



Form validation

Validation is a method, which allows us to correct or confirms a certain standard. It ensures the authentication of the entered data.

Validating forms is a common practice in all digital interactions. To validate a form in a flutter, we need to implement mainly three steps.

Step 1: Use the Form widget with a global key.

Step 2: Use TextFormField to give the input field with validator property.

Step 3: Create a button to validate form fields and display validation errors.

Let us understand it with the following example. In the above code, we have to use validator() function in the TextFormField to validate the input properties. If the user gives the wrong input, the validator function returns a string that contains an error message; otherwise, the validator function return null. In the validator function, make sure that the TextFormField is not empty. Otherwise, it returns an error message.

The validator() function can be written as below code snippets:

```
validator: (value) {
    if (value.isEmpty) {
        return 'Please enter some text';
    }
    return null;
},
validator: (value) {
    if (value.isEmpty) {
        return 'Please enter some text';
    }
    return null;
},
```

Now, open the main.dart file and add validator() function in the TextFormField widget. Replace the following code with the main.dart file.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final appTitle = 'Flutter Form Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: Text(appTitle),
        ),
        body: MyCustomForm(),
      ),
    );
  }
}
// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
// Create a corresponding State class, which holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  final _formKey = GlobalKey<FormState>();

  @override
```

```
Widget build(BuildContext context) {
 // Build a Form widget using the _formKey created above.
 return Form(
  key: _formKey,
  child: Column(
   crossAxisAlignment: CrossAxisAlignment.start,
   children: <Widget>[
    TextFormField(
     decoration: const InputDecoration(
      icon: const Icon(Icons.person),
      hintText: 'Enter your full name',
      labelText: 'Name',
     ),
     validator: (value) {
      if (value.isEmpty) {
       return 'Please enter some text';
      }
      return null;
     },
    ),
    TextFormField(
     decoration: const InputDecoration(
      icon: const Icon(Icons.phone),
      hintText: 'Enter a phone number',
      labelText: 'Phone',
     ),
     validator: (value) {
      if (value.isEmpty) {
       return 'Please enter valid phone number';
      }
      return null;
```

```
          },
        ),
        TextFormField(
          decoration: const InputDecoration(
          icon: const Icon(Icons.calendar_today),
          hintText: 'Enter your date of birth',
          labelText: 'Dob',
          ),
          validator: (value) {
            if (value.isEmpty) {
              return 'Please enter valid date';
            }
            return null;
          },
         ),
        new Container(
            padding: const EdgeInsets.only(left: 150.0, top: 40.0),
            child: new RaisedButton(
              child: const Text('Submit'),
              onPressed: () {
                // It returns true if the form is valid, otherwise returns false
                if (_formKey.currentState.validate()) {
                  // If the form is valid, display a Snackbar.
                  Scaffold.of(context)
                      .showSnackBar(SnackBar(content: Text('Data is in processing.')));
                }
              },
            )),
      ],
    ),
  );
  }
}
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final appTitle = 'Flutter Form Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: Text(appTitle),
        ),
        body: MyCustomForm(),
      ),
    );
  }
}
// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
// Create a corresponding State class, which holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    // Build a Form widget using the _formKey created above.
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          TextFormField(
            decoration: const InputDecoration(
              icon: const Icon(Icons.person),
              hintText: 'Enter your full name',
              labelText: 'Name',
            ),
            validator: (value) {
              if (value.isEmpty) {
                return 'Please enter some text';
              }
              return null;
            },
          ),
          TextFormField(
            decoration: const InputDecoration(
              icon: const Icon(Icons.phone),
              hintText: 'Enter a phone number',
              labelText: 'Phone',
            ),
            validator: (value) {
```

```
    if (value.isEmpty) {
      return 'Please enter valid phone number';
    }
    return null;
  },
),
TextFormField(
  decoration: const InputDecoration(
  icon: const Icon(Icons.calendar_today),
  hintText: 'Enter your date of birth',
  labelText: 'Dob',
  ),
  validator: (value) {
    if (value.isEmpty) {
      return 'Please enter valid date';
    }
    return null;
  },
 ),
new Container(
    padding: const EdgeInsets.only(left: 150.0, top: 40.0),
    child: new RaisedButton(
      child: const Text('Submit'),
      onPressed: () {
        // It returns true if the form is valid, otherwise returns false
        if (_formKey.currentState.validate()) {
          // If the form is valid, display a Snackbar.
          Scaffold.of(context)
              .showSnackBar(SnackBar(content: Text('Data is in processing.')));
        }
      },
            )),
          ],
        ),
      );
    }
  }
```