Department of Information Technology

Academic Year: 2024-25
Semester: VI
Class / Branch / Div: TE- IT A/B/C
Subject: MAD & PWA Lab
Name of Instructor: Mrs. Sujata Oak

Name of Student:
Student ID:
Roll No.
Date of Submission:

Experiment No.:2

Aim: To design Flutter UI by including common widgets.

Theory:

Everything in Flutter is a widget and their different types available in the Flutter framework

Whenever you are going to code for building anything in Flutter, it will be inside a widget. The central purpose is to build the app out of widgets. It describes how your app view should look like with their current configuration and state. When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.

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

Widget Build Visualization

In Flutter, widgets can be grouped into multiple categories based on their features, as listed below −

1. Platform specific widgets
2. Layout widgets
3. State maintenance widgets
4. Platform independent / basic widgets

A. Platform specific widgets

Flutter has widgets specific to a particular platform - Android or iOS.

Android specific widgets are designed in accordance with Material design guideline by Android OS. Android specific widgets are called as Material widgets.

iOS specific widgets are designed in accordance with Human Interface Guidelines by Apple and they are called as Cupertino widgets.

Some of the most used material widgets are as follows −

Scaffold, AppBar, BottomNavigationBar, TabBar, TabBarView, ListTile, RaisedButton, FloatingActionButton, FlatButton, IconButton, DropdownButton, PopupMenuButton ButtonBar, TextField, Checkbox, Radio, Switch, Slider, Date & Time Pickers, SimpleDialog, AlertDialog

Some of the most used Cupertino widgets are as follows −

CupertinoButton, CupertinoPicker, CupertinoDatePicker, CupertinoTimerPicker, CupertinoNavigationBar, CupertinoTabBar, CupertinoTabScaffold, CupertinoTabVie, CupertinoTextField, CupertinoDialog, etc.

B. Layout widgets

In Flutter, a widget can be created by composing one or more widgets. To compose multiple widgets into a single widget, Flutter provides large number of widgets with layout feature. For example, the child widget can be centered using Center widget.

Some of the popular layout widgets are as follows −

1. Container − A rectangular box decorated using BoxDecoration widgets with background, border and shadow.
2. Center − Center its child widget.
3. Row − Arrange its children in the horizontal direction.
4. Column − Arrange its children in the vertical direction.
5. Stack − Arrange one above the another.

C. State maintenance widgets

In Flutter, all widgets are either derived from StatelessWidget or StatefulWidget.

Widget derived from StatelessWidget does not have any state information but it may contain widget derived from StatefulWidget. The dynamic nature of the application is through interactive behavior of the widgets and the state changes during interaction. For example, tapping a counter button will increase / decrease the internal state of the counter by one and reactive nature of the Flutter widget will auto re-render the widget using new state information.

D. Platform independent / basic widgets

Flutter provides large number of basic widgets to create simple as well as complex user interface in a platform independent manner. Let us see some of the basic widgets in this chapter.

A] TEXT Widgets

Text widget is used to display a piece of string. The style of the string can be set by using style property and TextStyle class. The sample code for this purpose is as follows −

```
Text('Hello World!', style: TextStyle(fontWeight: FontWeight.bold))
```

Text widget has a special constructor, Text.rich, which accepts the child of type TextSpan to specify the string with different style. TextSpan widget is recursive in nature and it accepts TextSpan as its children. The sample code for this purpose is as follows –

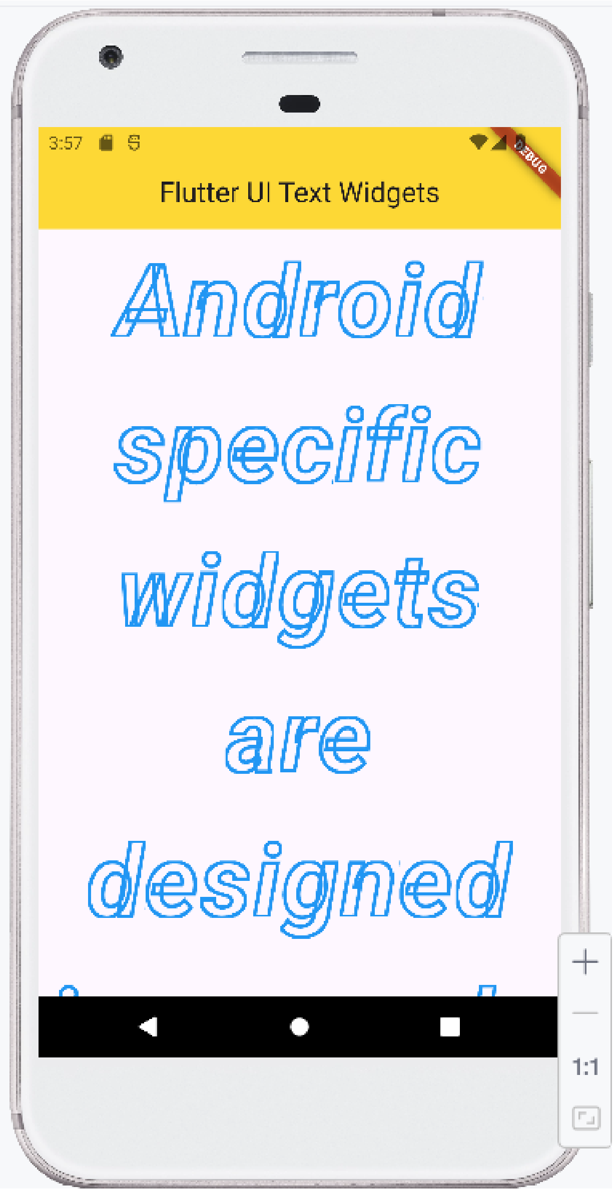
```
import 'package:flutter/material.dart';

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

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar:AppBar(
          title:Text('Flutter UI Text Widgets'),
          centerTitle: true,
          backgroundColor: Colors.yellow[600],
        ),
        body: Center(
          child:Text(
            "Android specific widgets are designed in accordance with Material design guideline
by Android OS. Android specific widgets are called as Material widgets iOS specific widgets
are designed .",
            //textDirection: TextDirection.ltr,
            textAlign: TextAlign.center,
            //overflow: TextOverflow.ellipsis,
            //textScaleFactor: 2,
            //softWrap: true,
           // maxLines: 3,
            style: TextStyle(
              fontSize: 40.0,
              //fontWeight: FontWeight.bold,
              //letterSpacing: 2.0,
             // wordSpacing: 12,
              //color:Colors.red[400],
              //foreground: Paint()
             // ..color = Colors.blue
             // ..strokeWidth=3.0
             // ..style=PaintingStyle.stroke,
             // background: Paint()
             // ..color = Colors.blue
             // ..strokeWidth=1.0
             // ..style=PaintingStyle.stroke,
             // backgroundColor: Colors.greenAccent[200],
             //   fontStyle: FontStyle.italic,

              //fontFamily:'Jersey15',
            ),
          ),
        ),

    );
  }
}
```



3:57

Flutter UI Text Widgets

Android specific widgets are designed

The most important properties of the Text widget are as follows −

maxLines, int − Maximum number of lines to show

overflow, TextOverFlow − Specify how visual overflow is handled using TextOverFlow class

style, TextStyle − Specify the style of the string using TextStyle class

textAlign, TextAlign − Alignment of the text like right, left, justify, etc., using TextAlign class

textDirection, TextDirection − Direction of text to flow, either left-to-right or right-to-left

.

B] IMAGE Widgets

Image widget is used to display an image in the application. Image widget provides different constructors to load images from multiple sources and they are as follows −

Image − Generic image loader using ImageProvider

Image.asset − Load image from flutter project's assets

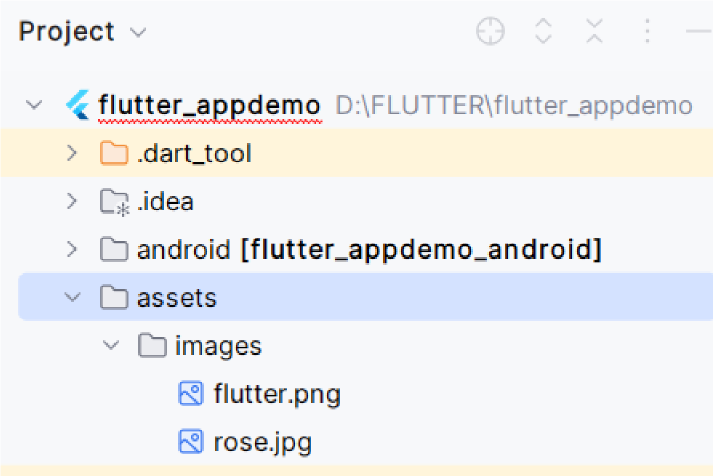
Image.file − Load image from system folder

Image.memory − Load image from memory

Image.Network − Load image from network

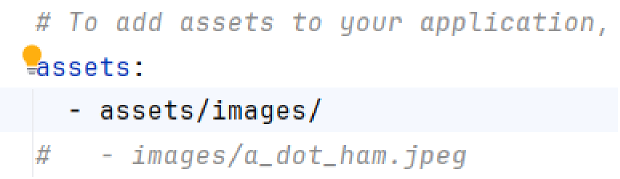
The easiest option to load and display an image in Flutter is by including the image as assets of the application and load it into the widget on demand.

Create a folder, assets in the project folder and place the necessary images.



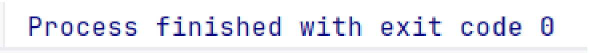
```
Project
flutter_appdemo  D:\FLUTTER\flutter_appdemo
  .dart_tool
  .idea
  android [flutter_appdemo_android]
  assets
    images
      flutter.png
      rose.jpg
```

Specify the assets in the **pubspec.yaml** as shown below –



```
# To add assets to your application,
assets:
  - assets/images/
#   - images/a_dot_ham.jpeg
```

Click on pub get option on top : You get the output as



```
Process finished with exit code 0
```

Now, load and display the image in the application.

Image.asset('assets/images/flutter.png')

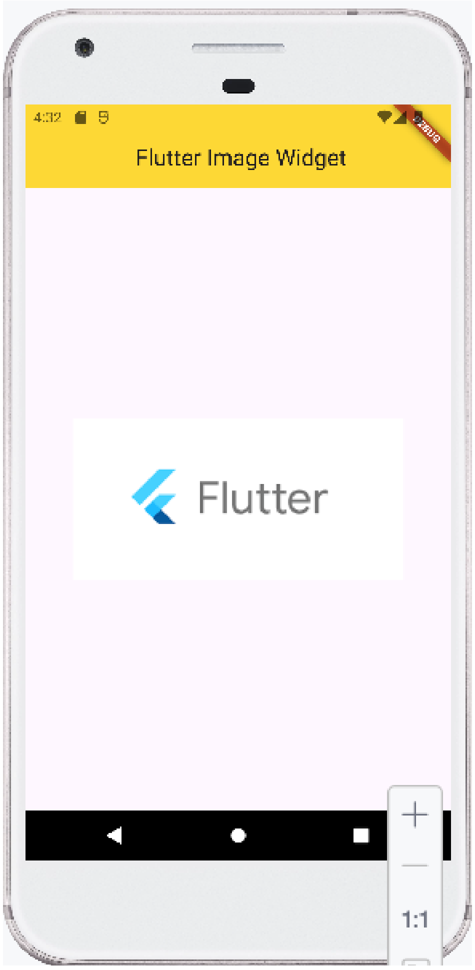
The complete source code of Image widget is as shown below –.

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

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

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:AppBar(
        title:Text(' Flutter Image Widget'),
        centerTitle: true,
        backgroundColor: Colors.yellow[600],
      ),
      body: Center(
      child:Image.asset('assets/images/flutter.png'),
     ),
    );
   }
  }
```

The loaded image is as shown below –

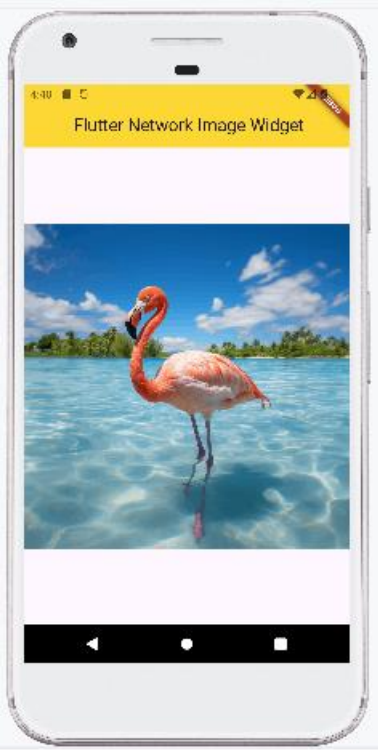


Network Image Widget:

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

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

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:AppBar(
        title:Text(' Flutter Network Image Widget'),
        centerTitle: true,
        backgroundColor: Colors.yellow[600],
      ),
      body: Center(
        child:Image(
          image:NetworkImage('https://ix-
marketing.imgix.net/genfill.png?auto=format,compress&w=1946'),
        ),
      ),
    );
  }
}
```



The most important properties of the Image widget are as follows –

image, ImageProvider – Actual image to load

width, double – Width of the image

height, double – Height of the image

alignment, AlignmentGeometry – How to align the image within its bounds

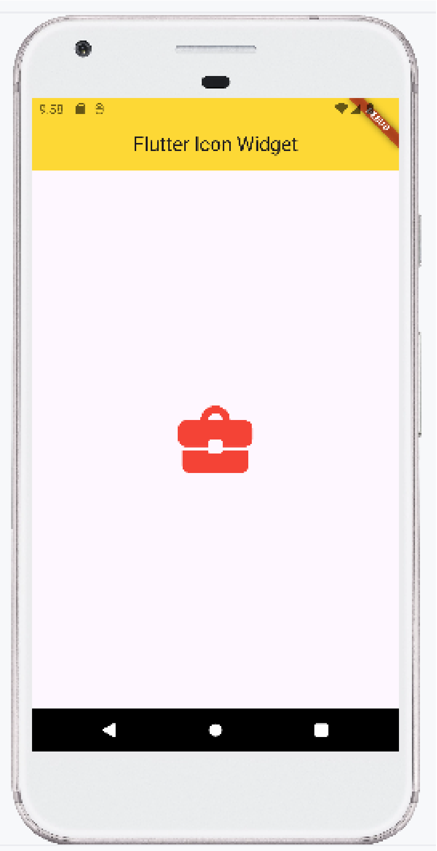
C] ICON Widgets

Icon widget is used to display a glyph from a font described in IconData class. The code to load a simple email icon is as follows −

Icon(Icons.email)

```
import 'package:flutter/material.dart';
void main() =>runApp(MaterialApp(
  home:Home(),
));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:AppBar(
        title:Text('Flutter Icon Widget'),
        centerTitle: true,
        backgroundColor: Colors.yellow[600],
      ),
      body: Center(
        child:Icon(
            Icons.business_center_rounded,
            size:100,
            color:Colors.red,
         ),
        ),
      );
   }
}
```



D]BUTTON Widgets

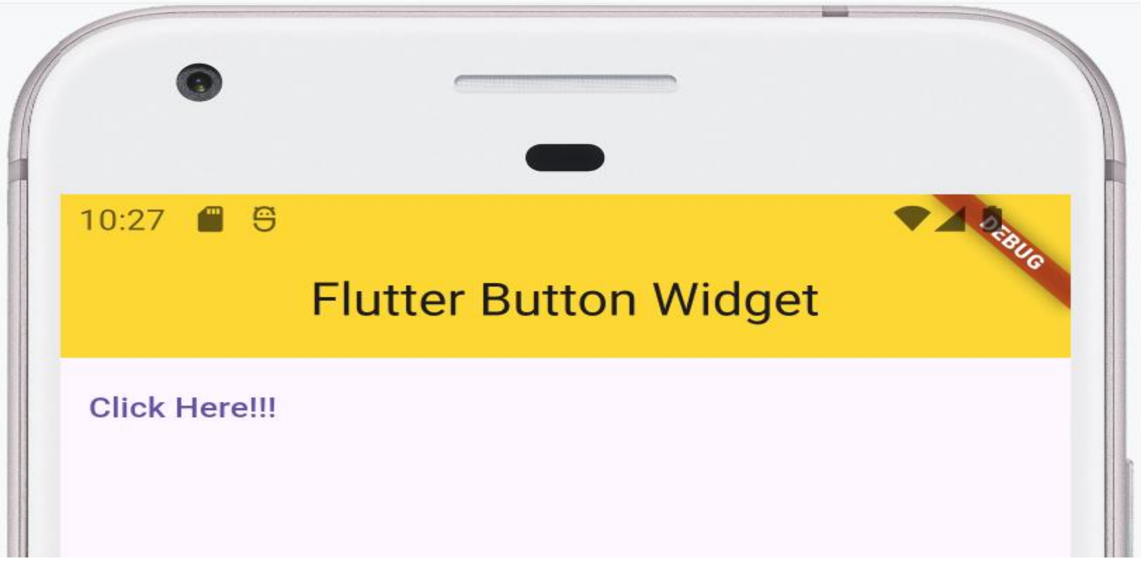
This widget allows you to perform some action on click. Flutter does not allow you to use the Button widget directly; instead, it uses a type of buttons like a Text/FlatButton and a Elevated/RaisedButton. We can use it as like below code snippets.

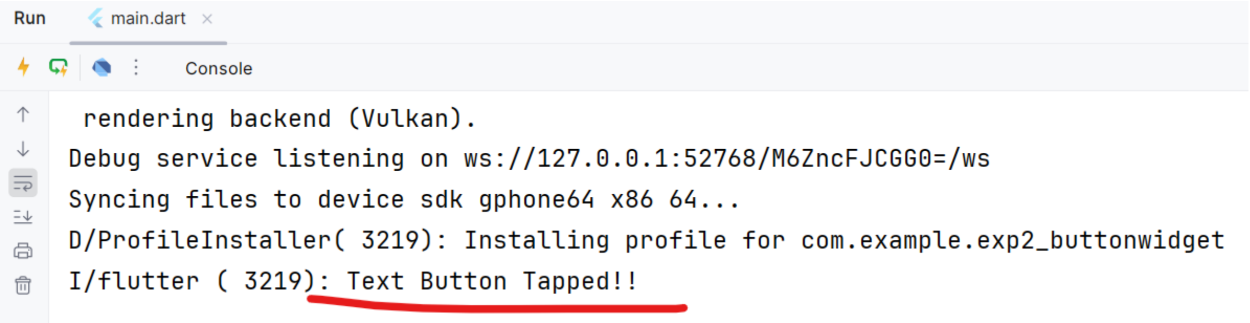
//TextButton Example

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

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

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter Button Widget'),
          centerTitle: true,
          backgroundColor: Colors.yellow[600],
        ),
        body: TextButton(
          child: Text('Click Here!!!'),
          onPressed: () {
            print('Text Button Tapped!!');
          },
        )
      );
    }
  }
```





```
 rendering backend (Vulkan).
Debug service listening on ws://127.0.0.1:52768/M6ZncFJCGG0=/ws
Syncing files to device sdk gphone64 x86 64...
D/ProfileInstaller( 3219): Installing profile for com.example.exp2_buttonwidget
I/flutter ( 3219): Text Button Tapped!!
```

//ElevatedButton

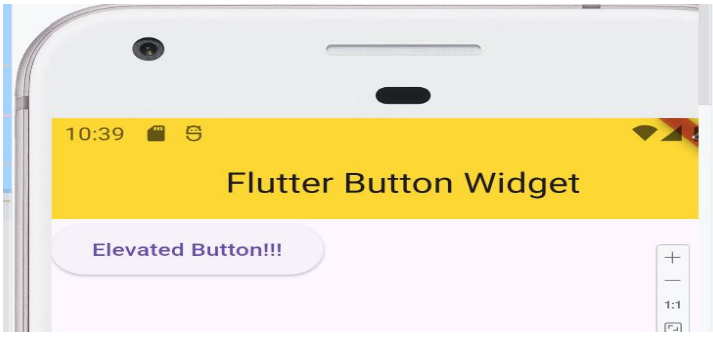
Example

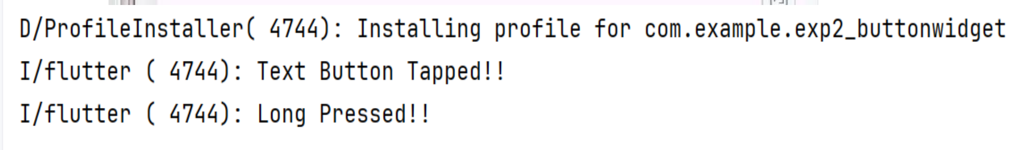
```
import 'package:flutter/material.dart';

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

class Home extends StatelessWidget {
  @override
Widget build(BuildContext context) {
 return Scaffold(
     appBar: AppBar(
       title: Text('Flutter Button Widget'),
       centerTitle: true,
       backgroundColor: Colors.yellow[600],
     ),
     body: ElevatedButton(
       child:Text('Elevated Button!!!'),
       onPressed: (){
         print('Text Button Tapped!!');
     },
         onLongPress: () {
           print('Long Pressed!!');
         }

     )
 );
}
}
```

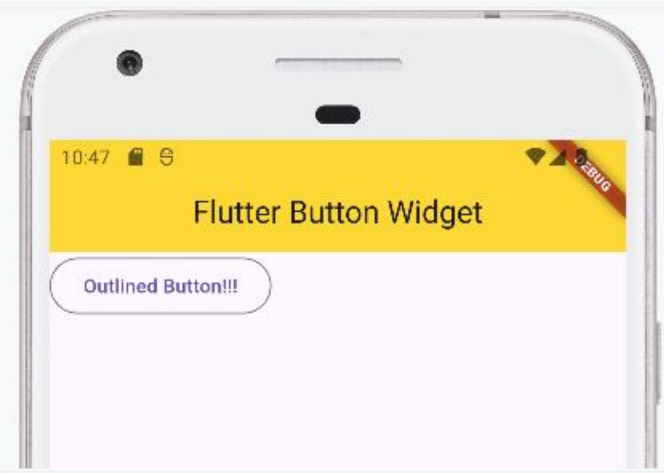


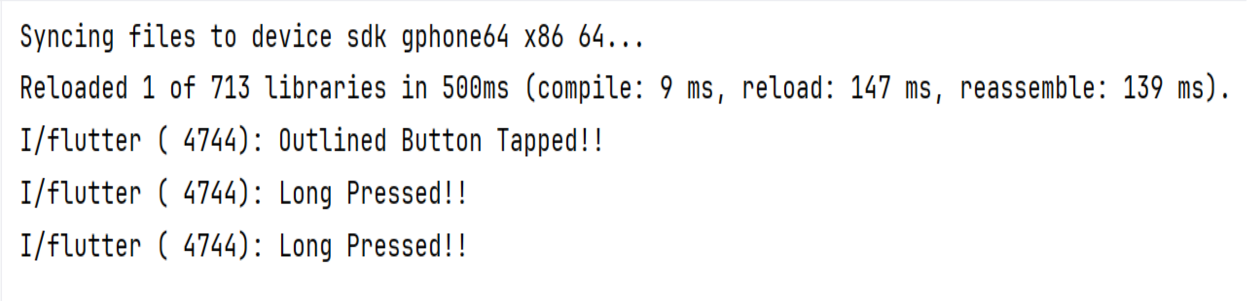


```
D/ProfileInstaller( 4744): Installing profile for com.example.exp2_buttonwidget
I/flutter ( 4744): Text Button Tapped!!
I/flutter ( 4744): Long Pressed!!
```

```
//OutlinedButton
Example
import
'package:flutter/material.dart'
;
void main()
=>runApp(MaterialApp(
  home:Home(),
));
class Home extends
StatelessWidget {
  @override
  Widget
build(BuildContext
context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Flutter Button
Widget'),
          centerTitle: true,
          backgroundColor:
Colors.yellow[600],
        ),
        body:
OutlinedButton(

child:Text('Outlined
Button!!!'),
          onPressed: (){
            print('Outlined
Button Tapped!!');
         },
            onLongPress: () {
              print('Long
Pressed!!');
            }
         )
      );
   }
}
```





```
Syncing files to device sdk gphone64 x86 64...
Reloaded 1 of 713 libraries in 500ms (compile: 9 ms, reload: 147 ms, reassemble: 139 ms).
I/flutter ( 4744): Outlined Button Tapped!!
I/flutter ( 4744): Long Pressed!!
I/flutter ( 4744): Long Pressed!!
```

Conclusion: In this experiment we have created user interface by using flutter common widgets like text, image, Icon and Button.