**Name:Ajinkya Sunil Patil  
D15B/42  
Lab-mpl**

### **Experiment 02: To Design Flutter UI by Including Common Widgets**

#### **Introduction:**

Flutter is an open-source UI software development kit created by Google. It is primarily used to develop applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. Flutter's appeal lies in its ability to craft visually stunning user interfaces with smooth animations, coupled with the fact that it uses Dart as its programming language. In this experiment, the focus is on understanding how to design user interfaces by incorporating various common widgets available in Flutter.

#### **Objective:**

The aim of this experiment is to design and develop a user-friendly Flutter UI using common widgets such as containers, rows, columns, buttons, text fields, and other essential components. This helps in gaining a clear understanding of how to efficiently utilize these widgets to create interactive and responsive UIs.

#### **Theory:**

Flutter revolves around a widget tree, where everything is a widget, from the structural elements like containers and rows to the interactive components such as buttons and text fields. Widgets can be broadly categorized into two types: **Stateful** and **Stateless** widgets.

* **Stateless Widgets**: These widgets are immutable, meaning their properties cannot change during the runtime of the app. Common examples include Text, Icons, and Containers.
* **Stateful Widgets**: These widgets maintain state, meaning their properties can change dynamically during the app’s lifecycle. Examples include Checkbox, Switch, and Slider.

Some of the most commonly used widgets in Flutter include:

* **Container**: A flexible widget used for layout control, it can contain other widgets and control the padding, alignment, and size.
* **Row and Column**: These widgets are used for arranging child widgets in a horizontal (Row) or vertical (Column) direction.
* **Text**: This widget displays a string of text and can be customized with various properties like font size, style, and alignment.
* **Buttons**: Buttons like ElevatedButton, TextButton, and IconButton are used to handle user interactions.
* **Image**: Displays an image, which can be loaded from assets or the internet.
* **TextField**: Allows users to input text, a key component for forms.

By combining these widgets, developers can design UIs that are both aesthetic and functional. Flutter’s rich collection of pre-built widgets provides great flexibility in UI design, enabling developers to create complex interfaces with ease.

**Factors use in my code:**

**Scaffold**: Provides the basic structure for the UI, including the app bar, body, and navigation buttons.

**AppBar**: The navigation bar at the top with options like Home, All, and Settings.

**TextButton.icon**: Displays a button with an icon and a label in the app bar.

**Container**: Used for structuring and styling the display area and buttons.

**GestureDetector**: Wraps widgets (such as buttons) and adds touch detection, calling a function when the user interacts with it.

**GridView.builder**: Dynamically creates a grid layout for the calculator buttons.

**FaIcon**: Displays Font Awesome icons, providing better customization options for icons.

#### **Conclusion:**

#### Designing a Flutter UI involves an understanding of the different common widgets and how they can be organized within the widget tree. By using widgets like Container, Row, Column, Text, Buttons, and TextField, developers can create responsive, interactive, and visually appealing UIs. This experiment helps in mastering the fundamental components of Flutter UI design, providing the basis for building more complex user interfaces in future projects.

**Code:**

import 'package:flutter/material.dart';

import 'package:font\_awesome\_flutter/font\_awesome\_flutter.dart';

import 'all\_calculations\_screen.dart'; // Import the AllCalculatorsScreen

class HomeScreen extends StatefulWidget {

@override

\_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

bool isScientific = false; // Toggle between basic and scientific modes

String expression = ""; // Holds the current input or result

// Handle button presses

void onButtonPressed(String value) {

setState(() {

if (value == "C") {

expression = ""; // Clear expression

} else if (value == "=") {

try {

expression = evaluateExpression(expression); // Calculate result

} catch (e) {

expression = "Error"; // Handle calculation errors

}

} else {

expression += value; // Append button value to expression

}

});

}

// Dummy evaluate function for simplicity

String evaluateExpression(String expr) {

try {

return (double.parse(expr)).toString(); // Convert expression to number

} catch (e) {

return "Error";

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.black,

appBar: AppBar(

backgroundColor: Colors.grey[900],

title: Text("Smart Calculator"), // App bar title

actions: [

\_navBarButton("Home", FontAwesomeIcons.home, () {}), // Home button

\_navBarButton("All", FontAwesomeIcons.list, () {

Navigator.push(context, MaterialPageRoute(builder: (context) => AllCalculatorsScreen())); // Navigate to AllCalculatorsScreen

}),

\_navBarButton("Settings", FontAwesomeIcons.cog, () {}), // Settings button

],

),

body: Column(

children: [

\_buildDisplay(), // Display for showing expression or result

\_buildToggleButton(), // Button to toggle between basic and scientific modes

\_buildButtonGrid(), // Grid of calculator buttons

],

),

);

}

**// Navigation bar buttons**

Widget \_navBarButton(String label, IconData icon, VoidCallback onPressed) {

return Padding(

padding: EdgeInsets.symmetric(horizontal: 8.0),

child: TextButton.icon(

onPressed: onPressed,

icon: FaIcon(icon, color: Colors.white, size: 18)**, // Icon from FontAwesome**

label: Text(label, style: TextStyle(color: Colors.white, fontSize: 16)), // Button label

),

);

}

// Display widget for showing the calculator expression/result

Widget \_buildDisplay() {

return Container(

width: double.infinity, // Take full width

padding: EdgeInsets.all(16), // Padding for spacing

alignment: Alignment.centerRight, // Align text to the right

height: 100, // Fixed height for display area

decoration: BoxDecoration(

color: Colors.grey[800], // Background color for display

borderRadius: BorderRadius.vertical(bottom: Radius.circular(10)), **// Rounded bottom corners**

),

child: Text(

expression, // Show the current expression/result

style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, color: Colors.white), **// Text style for display**

),

);

}

**// Toggle button between Basic and Scientific mode**

Widget \_buildToggleButton() {

return Padding(

padding: EdgeInsets.symmetric(vertical: 8),

child: GestureDetector(

onTap: () {

setState(() {

isScientific = !isScientific; // Toggle the mode

});

},

child: Container(

padding: EdgeInsets.symmetric(vertical: 6, horizontal: 12),

decoration: BoxDecoration(

color: Colors.orange, // Button color

borderRadius: BorderRadius.circular(8), // Rounded corners

),

child: Text(

isScientific ? "Basic Mode" : "Scientific Mode", // Toggle label

style: TextStyle(color: Colors.white, fontSize: 16), // Button text style

),

),

),

);

}

// Grid of calculator buttons

Widget \_buildButtonGrid() {

// **Basic calculator buttons**

List<String> basicButtons = [

"7", "8", "9", "C",

"4", "5", "6", "/",

"1", "2", "3", "\*",

"0", ".", "=", "+",

"-", "(", ")", "%"

]

// Additional buttons for scientific mode

List<String> scientificButtons = [

"sin", "cos", "tan", "log",

"√", "π", "e", "^"

];

// Combine buttons based on mode

List<String> buttons = isScientific ? scientificButtons + basicButtons : basicButtons;

return Padding(

padding: EdgeInsets.symmetric(horizontal: 12, vertical: 8),

child: GridView.builder(

shrinkWrap: true, // Prevents overflow issues in layout

physics: NeverScrollableScrollPhysics(), // Avoid scrolling in GridView

gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: 4, // 4 buttons per row

childAspectRatio: 1.2, // Aspect ratio for buttons

crossAxisSpacing: 10, // Horizontal space between buttons

mainAxisSpacing: 10, // Vertical space between buttons

),

itemCount: buttons.length, // Total number of buttons

itemBuilder: (context, index) {

return \_buildButton(buttons[index]); // Build each button

},

),

);

}

**// Button widget with styling**

Widget \_buildButton(String value) {

bool isSpecial = (value == "C" || value == "="); **// Special buttons (Clear and Equals)**

return GestureDetector(

onTap: () => onButtonPressed(value), // Handle button press

child: Container(

decoration: BoxDecoration(

color: isSpecial ? Colors.orange : Colors.grey[900]**, // Special button color**

borderRadius: BorderRadius.circular(8), // Rounded corners for buttons

),

alignment: Alignment.center, // Center button text

child: Text(

value, // Button label (number or operator)

style: TextStyle(

fontSize: 22, // Font size for button text

fontWeight: FontWeight.bold, // Bold text for button

color: Colors.white, // Text color

),

),

),

);

}

}

**Factors use in my code:**

**Scaffold**: Provides the basic structure for the UI, including the app bar, body, and navigation buttons.

**AppBar**: The navigation bar at the top with options like Home, All, and Settings.

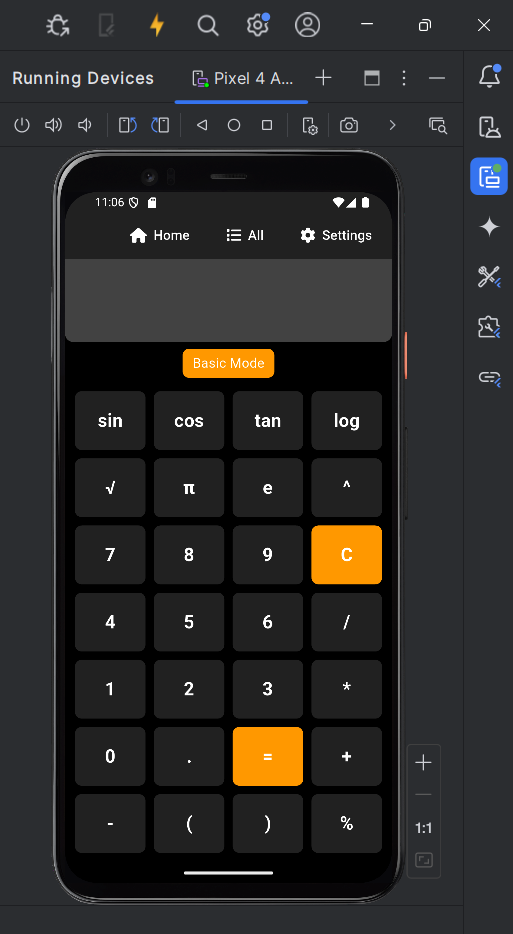
**TextButton.icon**: Displays a button with an icon and a label in the app bar.

**Container**: Used for structuring and styling the display area and buttons.

**GestureDetector**: Wraps widgets (such as buttons) and adds touch detection, calling a function when the user interacts with it.

**GridView.builder**: Dynamically creates a grid layout for the calculator buttons.

**FaIcon**: Displays Font Awesome icons, providing better customization options for icons.



**OUTPUT:**