UIT UNIVERSITY

**CSC-318 – Mobile Application Development**

**LAB 2:** **Adding interactivity to the Flutter app using Stateful widgets**

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Section : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semester : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fall 2024

Computer Science Department

LAB-2 | Adding interactivity to the Flutter app using Stateful widgets

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Stateful and stateless widgets:

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.

# Creating a stateful widget

* A stateful widget is implemented by two classes: a subclass of StatefulWidget and a subclass of State.
* The state class contains the widget's mutable state and the widget's build() method.
* When the widget's state changes, the state object calls setState(), telling the framework to redraw the widget.

Implementing a custom stateful widget requires creating two classes:

* A subclass of StatefulWidget that defines the widget.
* A subclass of State that contains the state for that widget and defines the widget's build() method.

# Subclass StatefulWidget

The MyHomePage class manages its own state, so it overrides createState() to create a State object. The framework calls createState() when it wants to build the widget. In this example, createState() returns an instance of \_MyHomePageState.

class MyHomePage extends StatefulWidget {

  @override

  \_MyHomePageState createState() => \_MyHomePageState();

}

# Subclass State

The \_MyHomePageState class stores the mutable data that can change over the lifetime of the widget.

class \_MyHomePageState extends State<MyHomePage> {

  int \_counter = 0;

# build() method

The class \_MyHomePageState also defines a build() method, which creates a column containing a ElevatedButton and a Text widget within the body. The ElevatedButton has an onPressed property that defines the callback function (\_incrementCounter) for handling a tap.

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Theme.of(context).colorScheme.inversePrimary,

        title:const Text("LAB # 2 --- USING STATEFUL WIDGET"),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            Text(

              '  You have pushed the ADD (+) button $\_counter times:', style: Theme.of(context).textTheme.headlineMedium,

            ),

            ElevatedButton(onPressed: \_incrementCounter, child: const Icon(Icons.add),),

          ],

        ),

      ),

    );

  }

The \_incrementCounter method, which is called when the  ElevatedButton is pressed, calls setState(). Calling [setState](https://api.flutter.dev/flutter/widgets/State/setState.html) notifies the framework that the internal state of this object has changed in a way that might impact the user interface in this subtree, which causes the framework to schedule a [build](https://api.flutter.dev/flutter/widgets/State/build.html) for this [State](https://api.flutter.dev/flutter/widgets/State-class.html) object.

void \_incrementCounter() {

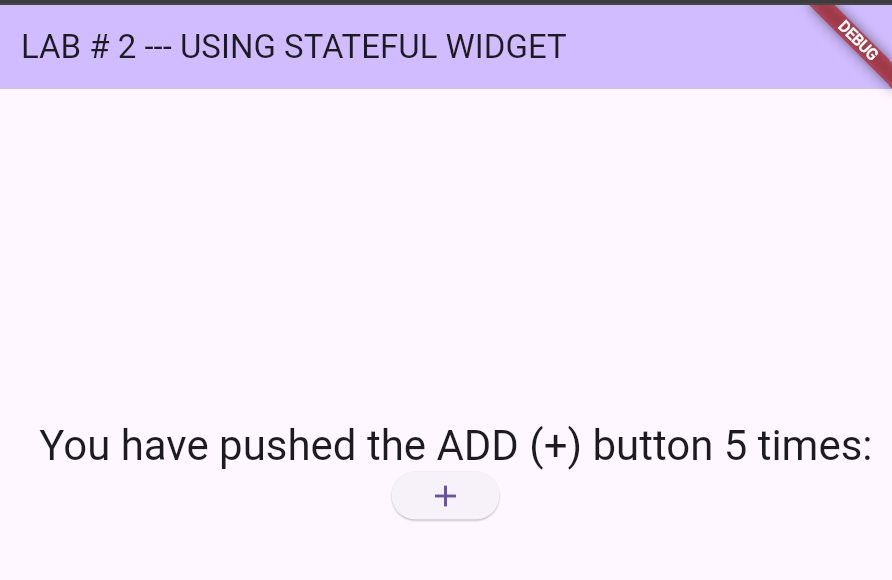
    setState(() {

      \_counter++;

    });

  }

# Running the application-i



Clicking the Elevated Button increments the counter value, which causes the state of the Widget to change. The setState method is called to trigger a rebuild, and then the widget is rebuilt to reflect this update.

# Stateful widget with a text field

Another example of a Flutter application that includes a stateful widget with a text field. The app will display the text entered in the text field below it when you press a button. The build method constructs the UI using:

 A TextField for user input.

 An ElevatedButton to submit the input.

 A Text widget to display the entered text.

Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Theme.of(context).colorScheme.inversePrimary,

        title:const Text("LAB2--USING STATEFUL WIDGET"),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            TextField(

          controller: textController,

          maxLines: null,

          decoration: const InputDecoration(

          hintStyle: TextStyle(color: Colors.blue),

          hintText: "Enter your name"

        ),

        ),

        ElevatedButton(onPressed: (){

          setState(() {

            displayText = textController.text;

          });

        }, child: const Text("Show Text")),

        Text(displayText,style: TextStyle(fontSize: 20),)

          ],

        ),

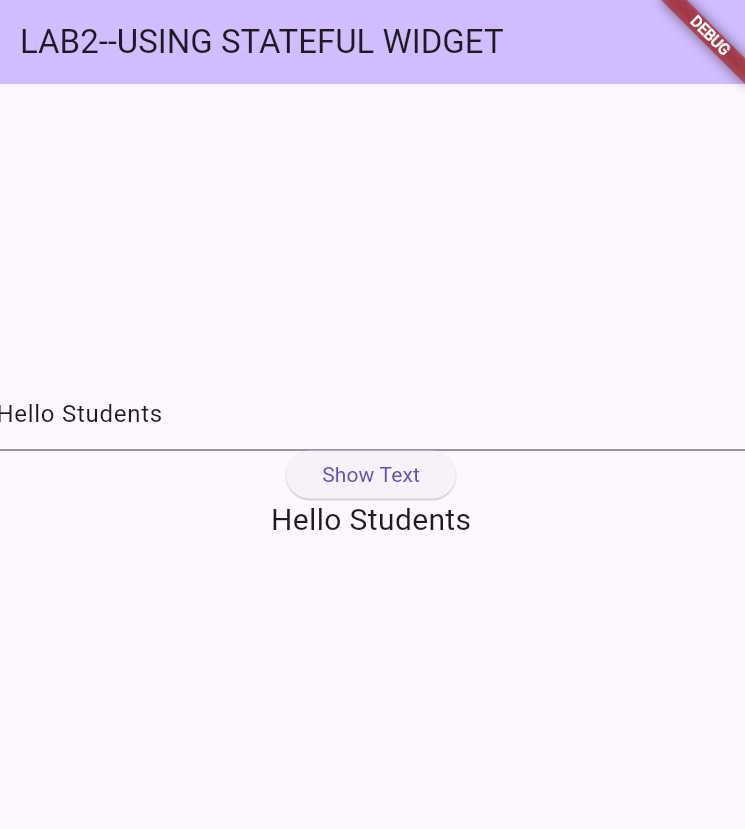
      ),

**State Variables**: The state class defines displayText to hold the text entered by the user and uses a TextEditingController to control the text field.

TextEditingController textController = TextEditingController();

  String displayText = "";

# Running the application-ii



# Exercises:

* 1. Develop a Flutter application that utilizes a stateful widget to implement a simple counter. The app should allow users to increase and decrease the counter value through button interactions, following the design shown below.

A screenshot of a phone

Description automatically generated

* 1. Develop a Flutter project that includes a stateful widget named MyApp with an associated state class called \_MyState. This state class should have a changeColor function to manage the widget's state. The application should change the background color of a container from yellow to red when the FloatingActionButton is pressed.
  2. Use any three of the provided material components to enhance interactivity in your Flutter application
* DropdownButton
* TextButton
* IconButton
* Radio
* Slider
* Switch
* Checkbox