

**SEB-312 Mobile Application Development**

**LAB # 06**

**LAB Title**

|  |
| --- |
| Creating a multi-screen app with named routes. Create a splash screen, passing data from one screen to another. |

**Assessment of CLO: 03, PLO: 05**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** |  | | |
| **Roll No.** |  | | |
| **Semester** |  | **Session** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Perf. Level**  **Criteria** | **Excellent**  **(2.5)** | **Good**  **(2)** | **Satisfactory**  **(1.5)** | **Needs Improvement**  **(0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional, or not Performed. |  |
| **2** | Results & Debugging  Or Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem-Solving & Adaptability  (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

**Experiment evaluated by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor’s Name** | **Sidra Khatoon** | | |
| **Date** |  | **Signature** |  |

**Objective**

The objective of lab is exploring inkwell widgets and its usages. Also learn how to use ScrollView, ListView and ListTile widget.

**Instructions**

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

**Flutter Splash Screen**

A splash screen is a launch screen, start screen, or boot screen, which is a graphical control element containing the image, logo, and current version of the software. It is the first screen of the app that displays whenever the application is loading. It can also be the app's welcome screen that provides a simple initial experience when a mobile game or program is launching. The splash screen is just a display screen that allows users to look something while the hardware is loading to present software to the user.

The common elements of a splash screen contain a company name and logo or a title. The most common example of a splash screen is the Flutter logo on starting the Flutter application or Microsoft logo while starting the Microsoft operating system. In this tutorial, we are going to see how a splash screen is created in Flutter application.

**Splash Screen Characteristics**

The following are the essential characteristics of the splash screen:

* It is mainly used for branding or identity recognition of the application and puts the branding impression to users.
* It can also be used to show some loading progress indicator while the hardware is loading to present software to the user.
* When the loading of the splash screen completes, the user gets another functional screen that would be home screen or dashboard, then is forgotten. Once the loading completes, we cannot press the back button to return the splash screen.

**How to Create a Splash Screen?**

**Step 1:** First we have to create new project by selecting application in flutter: New Project. Remove MyHomePage class form this code. Your code only consist on main function and MyApp code. In MyApp class, you have call Splash screen class in home parameter as highlighted below. Because Splash screen in execute first we run the application.

import 'package:flutter/material.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  // This widget is the root of your application.

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

        useMaterial3: true,

      ),

      home: const SplashScreen(),

    );

  }

}

**Step 2:** Create a UI for splash Screen. You can create this class in main.dart file and you can also create new file in lib folder under the main.dart. Below code create simple UI for Splash Screen.

class SplashScreen extends StatelessWidget {

  const SplashScreen({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        color: Colors.cyan,

        child: Center(

            child: Text(

          "Classico",

          style: TextStyle(

              fontSize: 34, fontWeight: FontWeight.bold, color: Colors.white),

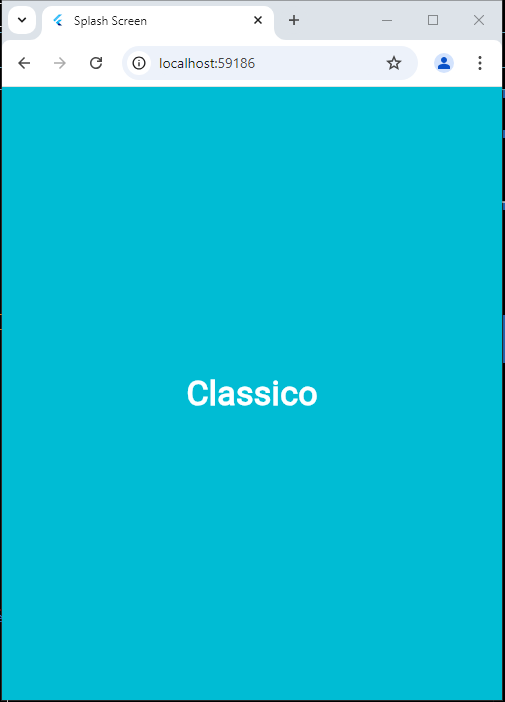
        )),

      ),

    );

  }

}

****

**Step 3:** Now we creating new class for first screen. After some duration, splash screen navigates to the first screen. You can create this class in main.dart file and you can also create new file in lib folder under the main.dart.

class FirstScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

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

        //title: Center(child: Text("Classico")),

        title: Center(child: Text("FirstScreen")),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Container(

              height: 100,

              width: 200,

              color: Colors.blueAccent,

              child: Center(

                child: Padding(

                  padding: const EdgeInsets.all(8.0),

                  child: Text(

                    "Welcome to Classico",

                    style: TextStyle(fontSize: 18,

                        fontWeight: FontWeight.w400,

                        color: Colors.white),

                  ),

                ),

              ),

            ),

            SizedBox(

              height: 11,

            ),

            Text("You have learned Splash Screen concept",

            style: TextStyle(fontSize: 24,fontWeight: FontWeight.w800,color: Colors.black87),)

          ],

        ),

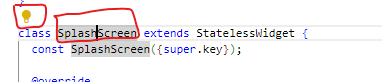
      ),

    );

  }

}

**Step 4:** Now we covert over Splash Screen class into stateful widget. You have select class name now click on the bulb icon. A dialog box is open in you have select covert into stateful widget.



Now your code looks like this

class SplashScreen extends StatefulWidget {

  const SplashScreen({super.key});

  @override

  State<SplashScreen> createState() => \_SplashScreenState();

}

class \_SplashScreenState extends State<SplashScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        color: Colors.cyan,

        child: Center(

            child: Text(

          "Classico",

          style: TextStyle(

              fontSize: 34, fontWeight: FontWeight.bold, color: Colors.white),

        )),

      ),

    );

  }

}

**Step 5:** For Splash Screen we are using initState() method of stateful widget. initState() method called once when the stateful widget is inserted in the widget tree. initState() first call super.initState() and then Timer of duration 4 seconds(Timer function has 2 arguments,first is Duration and second is action to be performed). After 4 seconds,Current screen will be replaced by new Screen i.e. FirstScreen() . We have import library of dart:sync to run Timer function.

class SplashScreen extends StatefulWidget {

  const SplashScreen({super.key});

  @override

  State<SplashScreen> createState() => \_SplashScreenState();

}

class \_SplashScreenState extends State<SplashScreen> {

  void initState() {

    super.initState();

    Timer(Duration(seconds: 3),

          ()=>Navigator.pushReplacement(context,

                                        MaterialPageRoute(builder:

                                                          (context) =>

                                                          FirstScreen()

                                                         )

                                       )

         );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        color: Colors.cyan,

        child: Center(

            child: Text(

          "Classico",

          style: TextStyle(

              fontSize: 34, fontWeight: FontWeight.bold, color: Colors.white),

        )),

      ),

    );

  }

}

In your app after 3 second your app is navigating to the first screen

**Send Data from One Screen to Another Screen**

As a developer, most of the time, we need to pass a data from one screen to another. For this we need to add a attribute in Second Screen. Before that let’s change the First Screen to StatefulWidget and Second Screen as well so that we can use some reactive widget. Let’s add one text field and send the value of the field to second Screen when button pressed.

Step 1: Create the Second Screen (Basic Static View)

In the previous lab, we created a splash screen and a basic first screen. Now, we extend that by creating a second screen.

class SecondScreen extends StatelessWidget {

const SecondScreen({super.key});

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

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

title: Center(child: Text("Second Screen")),

),

body: Container(

color: Colors.purpleAccent,

child: Center(

child: Container(

height: 100,

width: 200,

color: Colors.blue,

child: Center(

child: Text(

'Welcome to Second Screen',

style: TextStyle(

fontSize: 12,

fontWeight: FontWeight.bold,

color: Colors.white),

),

),

),

),

),

);

}

}

📝 Explanation: This creates a simple second screen with a welcome message and styled layout. Initially, no data is passed from the first screen.

**Step 2:** Convert Screens to StatefulWidgets

To handle user input and dynamic updates, we convert both FirstScreen and SecondScreen into StatefulWidgets. This allows us to manage state, like storing user-entered text.

**Step 3:** Add TextField in First Screen for User Input

We modify FirstScreen to include a TextField so the user can input their name, which we’ll pass to the second screen.

class FirstScreen extends StatefulWidget {

const FirstScreen({super.key});

@override

State<FirstScreen> createState() => \_FirstScreenState();

}

class \_FirstScreenState extends State<FirstScreen> {

final TextEditingController \_nameController = TextEditingController();

@override

void dispose() {

\_nameController.dispose(); // Clean up the controller

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

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

title: Center(child: Text("First Screen")),

),

body: Center(

child: SizedBox(

width: 200,

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Container(

height: 50,

color: Colors.blueAccent,

child: Center(

child: Text(

"Welcome to Classico",

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.w400,

color: Colors.white),

),

),

),

SizedBox(height: 11),

TextField(

controller: \_nameController,

decoration: InputDecoration(hintText: "Enter your name"),

),

SizedBox(height: 11),

ElevatedButton(

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) =>

SecondScreen(textName: \_nameController.text),

),

);

},

child: Text("Go to Second Screen"),

),

],

),

),

),

);

}

}

**Explanation:**

We used TextEditingController to get user input. When the button is clicked, the text is passed to the SecondScreen using its constructor.

**Step 4:** Update Second Screen to Accept and Display Input

Now we update SecondScreen to receive the name from FirstScreen and display it in a welcome message.

class SecondScreen extends StatefulWidget {

final String? textName; // Data from FirstScreen

const SecondScreen({super.key, this.textName});

@override

State<SecondScreen> createState() => \_SecondScreenState();

}

class \_SecondScreenState extends State<SecondScreen> {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

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

title: Center(child: Text("Second Screen")),

),

body: Container(

color: Colors.purpleAccent,

child: Center(

child: Container(

height: 100,

width: 200,

color: Colors.blue,

child: Center(

child: Text(

'Welcome to ${widget.textName} Second Screen',

style: TextStyle(

fontSize: 12,

fontWeight: FontWeight.bold,

color: Colors.white),

),

),

),

),

),

);

}

}

**Explanation:**

The name from FirstScreen is received using a constructor parameter and accessed using widget.textName. A dynamic greeting message is shown using the passed value.

**Assessment**

**Task 1: Splash Screen with Welcome Message**

**Objective:**

* Design and implement a **Splash Screen** in Flutter.
* Display a **welcome message dynamically**.

**Task: Develop an Animated Splash Screen**

Create a Splash Screen that:  
✅ Displays an **app logo**.  
✅ Shows a **welcome message**.  
✅ **Automatically navigates** to the **Main Screen** after 4 seconds.  
✅ Provides a **personal introduction on the second screen**.

**Instructions:**

1. **Design a Splash Screen** that includes the **app name and a welcome message**.
2. Use a **Timer** to transition from the Splash Screen to the **Main Screen** after the animation completes.
3. Create a **Main Screen** that displays "Welcome to My App!" and includes a **button** to navigate to the **Introduction Screen**.
4. Develop an **Introduction Screen** that showcases **your name, class, semester, and hobbies**.
5. Use **Containers, Columns, and Text Widgets** to create a well-structured and visually appealing UI.

**Task 2:**

Create a simple Flutter app that demonstrates passing data between two screens. The app should have a form on the first screen where users can enter their name and age. Upon submitting, it should navigate to the second screen and display a personalized message with the entered details.

**Requirements**

1. Screen 1 - InputScreen

* A TextField for the user’s name.
* A TextField for the user’s age.
* A RaisedButton or ElevatedButton to submit and navigate to the next screen.

2. Screen 2 - DisplayScreen

* Displays a greeting message like "Hello [Name], you are [Age] years old!"

I**nstructions**

1. Create a Flutter Project:

* Set up a new Flutter project (or use an existing one) in your IDE.

2. Design the Input Screen:

* Add two TextField widgets for the user’s name and age.
* Add a button to navigate to the next screen. Ensure the button is only enabled when both fields are filled.

3. Pass Data to the Second Screen:

* Use the Navigator.push() method to navigate from InputScreen to DisplayScreen.
* Pass the entered name and age as arguments to DisplayScreen.

4. Display Data on the Second Screen:

* Retrieve the passed arguments in DisplayScreen.
* Use them to display a personalized message.