Shravani Anil Patil **D15A-37**

EXPERIMENT NO:04

Aim: To create an interactive Form using form widget.

Theory:

Understanding Forms in Flutter:

A **Form** in Flutter is a structured container used to collect user input through various fields like **text fields, dropdowns, checkboxes, and buttons**. Forms play a crucial role in applications that require user data entry, such as **login pages, registration forms, and feedback submissions**. Flutter provides the **Form** widget, which works alongside **TextFormField** and other input elements to handle:

- **Validation** – Ensuring data accuracy before submission.
- **State management** – Storing and retrieving user input.
- **Error handling** – Displaying validation errors when needed.

By implementing form validation techniques, developers can improve data reliability and enhance the overall user experience.

Creating a Form in Flutter

When creating a form in Flutter, several essential components come into play:

1. The Form Widget

- The **Form** widget acts as a **container** for grouping multiple form fields and managing their validation.

2. GlobalKey for Identification

- A **GlobalKey<FormState>** is required to uniquely identify the form and enable operations like validation and data retrieval.

3. TextFormField for User Input

- The **TextFormField** widget allows users to enter data such as names, phone numbers, and email addresses.
- It renders a material design text field and provides validation error messages when necessary.

4. Customizing Input Fields with InputDecoration

- The **InputDecoration** property enhances usability by customizing:
    - Labels
    - Icons
    - Borders
    - Hint text

5. Implementing Form Validation

- The **validator** property ensures user input meets specific criteria before submission.
- Different input types require appropriate keyboard settings, such as:
    - **TextInputType.number** for numeric fields.
    - **TextInputType.emailAddress** for email fields.

6. Managing State Efficiently

- Proper **state management** is required to store and process user input accurately.

7. Adding a Submit Button

- A **submit button** is essential to trigger form validation and process the collected data.

Key Properties of the Form Widget

1. **key** – A **GlobalKey** that uniquely identifies the form, enabling operations like validation, resetting, and saving.
2. **child** – Contains the form fields, typically wrapped in a **Column** or **ListView** for structured layout.
3. **autovalidateMode** – Controls when the form fields should be automatically validated.

Important Methods of the Form Widget

1. **validate()**
    - Checks if all form fields are valid.
    - Returns **true** if valid, otherwise **false**.
    - Useful for ensuring form correctness before submission.
2. **save()**
    - Saves the current values of all form fields.
    - Calls the **onSaved** callback for each field.
    - Typically used after successful validation.
3. **reset()**
    - Resets the form to its **initial state**, clearing user-entered data.
4. **currentState**
    - Returns the current **FormState** associated with the form, allowing interaction with form elements.
    -

Login.dart:

```
import 'package:flutter/material.dart';
import 'home.dart'; // Import the HomePage file

class LoginPage extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.white,
     appBar: AppBar(
       backgroundColor: Colors.white,
       elevation: 0,
       leading: IconButton(
```

```
    icon: Icon(Icons.arrow_back, color: Colors.black),
    onPressed: () {
      Navigator.pop(context);
    },
  ),
),
body: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 20),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Image.asset('assets/logo.png', width: 120, height: 120),
      SizedBox(height: 20),
      Text(
        "Welcome Back!",
        style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
      ),
      SizedBox(height: 10),
      Text(
        "Login to continue",
        style: TextStyle(fontSize: 14, color: Colors.black54),
      ),
      SizedBox(height: 30),
      TextField(
        controller: emailController,
        decoration: InputDecoration(
          labelText: "Email",
          border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 15),
      TextField(
        controller: passwordController,
        obscureText: true,
        decoration: InputDecoration(
          labelText: "Password",
        border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 20),
```

```
ElevatedButton(
  onPressed: () {
    String email = emailController.text;
    String password = passwordController.text;

    if (email.isNotEmpty && password.isNotEmpty) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomePage()),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Please enter email and password")),
      );
    }
  },
  child: Text("Login"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.red,
    foregroundColor: Colors.white,
    minimumSize: Size(double.infinity, 50),
  ),
),
SizedBox(height: 15),
TextButton(
  onPressed: () {},
  child: Text(
    "Forgot Password?",
    style: TextStyle(color: Colors.red),
  ),
),
SizedBox(height: 20),
Text("Or login with"),
SizedBox(height: 10),
ElevatedButton.icon(
  onPressed: () {},
  icon: Image.asset('assets/Google.png', width: 24, height: 24),
  label: Text("Login with Google"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.white,
```

```
        foregroundColor: Colors.black,
        minimumSize: Size(double.infinity, 50),
        side: BorderSide(color: Colors.black26),
      ),
    ),
  ],
```

OUTPUT: