Name: Akanksha Shinde   Class: D15C   Roll No: 53

AIM: To create an interactive Form using form widget

THEORY:

1. Form

The Form widget is used to group multiple form fields (e.g., TextFormField) and provides a way to validate all the fields in a single go. It uses a GlobalKey<FormState> to uniquely identify the form and manage its state, including validation.

2. TextFormField

TextFormField widgets are used to accept input from the user. In this example, there are two TextFormField widgets: one for the username and another for the password. Each field is styled with custom text colors and an outline border. Validators are attached to each field to ensure that they are not left empty.

3. Validators

Validators are functions that check the input of form fields for errors. If the input is invalid, a validator returns an error string that is displayed below the field. If the input is valid, it returns null. In this code, simple validators are used to ensure that the username and password fields are not left empty.

4. ElevatedButton

An ElevatedButton is used to trigger form submission. The button's onPressed callback checks if the form is valid by using the form's validate method. If the form is valid, the app could then proceed with the login process (not implemented in this snippet).

5. TextEditingController

TextEditingController objects are used to control and listen to the text that users type into TextFormField widgets. They can be used to fetch the current value of a text field, clear its content, or listen for changes.

Key Concepts

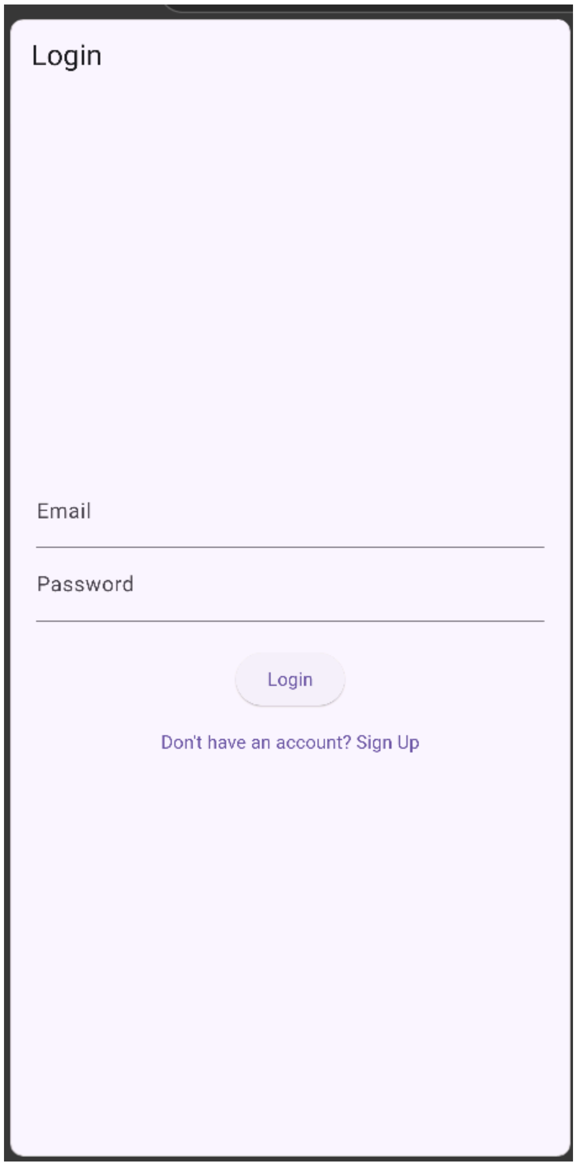
- State Management: The use of GlobalKey to manage form state and validation is a simple example of state management in Flutter.
- Validation: Demonstrates how to enforce input rules before processing data.

CODE:

```
import 'package:flutter/material.dart';
import 'home.dart'; // Import your HomePage
import 'start.dart'; // Import your StartPage

class AuthPage extends StatefulWidget {
  @override
  _AuthPageState createState() => _AuthPageState();
}

class _AuthPageState extends State<AuthPage> {
  bool isLogin = true;  // Toggle state

  void toggleAuthMode() {
    setState(() {
      isLogin = !isLogin;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(isLogin ? "Login" : "Sign Up")),
      body: Padding(
        padding: EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              decoration: InputDecoration(labelText: "Email"),
            ),
            TextField(
              decoration: InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            if (!isLogin)  // Show only in Sign Up mode
              TextField(
                decoration: InputDecoration(labelText: "Confirm Password"),
                obscureText: true,
              ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
```

```
          // After successful login/signup, navigate to StartPage
          Navigator.pushReplacementNamed(context, '/start');
         },
         child: Text(isLogin ? "Login" : "Sign Up"),
        ),
        TextButton(
         onPressed: toggleAuthMode,
         child: Text(isLogin
           ? "Don't have an account? Sign Up"
           : "Already have an account? Login"),
        ),
       ],
      ),
     ),
    );
   }
  }
```

OUTPUT:



CONCLUSION:

It utilizes GlobalKey<FormState> for form management, TextFormField for user input, and validators to ensure data validity. The ElevatedButton triggers form submission, while TextEditingController handles text input. The code introduces simple state management through GlobalKey, showcasing essential Flutter concepts for creating a functional form.