Name: Bhavishka Mulchandani Roll No.: 40 Class: D15B

Experiment No: 4

Experiment No 4 **Aim: To create an interactive Form using form widget**	
ROLL NO	**40**
NAME	**Bhavishka Mulchandani**
CLASS	**D15-B**
SUBJECT	**MAD & PWA Lab**
LO-MAPPED	

Name: Bhavishka Mulchandani		Roll No.: 40		Class: D15B

Aim: To create an interactive Form using form widget

Theory:

- Form Widget:

  The Form widget in Flutter acts as a container for multiple FormField widgets. It tracks the form's global state, facilitating scenarios like form validation, resetting, and saving of form data. The Form widget is essential for grouping related form fields together, making collective operations on these fields more manageable.

- GlobalKey<FormState>:

  To interact with the form, such as validating fields or resetting the form, a GlobalKey<FormState> is used. This key provides access to the FormState, which is where the logic for these operations resides. By associating a GlobalKey<FormState> with a Form, you gain the ability to programmatically control the form's state from other parts of your widget tree.

- TextFormField:

  TextFormField is a specialized FormField widget for text input. It integrates seamlessly with the Form widget, supporting features like validation, auto-validation, and saving. Each TextFormField can have its own validation logic, defined through its validator property, which can enforce specific rules (e.g., required fields, email format) and provide user feedback.

- Validation Logic:

  Validation is a critical aspect of handling forms. The validator function within FormField widgets allows you to define validation logic for each input. This function is called for each form field when attempting to validate the form. If the input is valid, the validator should return null; otherwise, it returns a string with an error message. Validation can be triggered programmatically (e.g., when a button is pressed) or automatically based on user interaction, depending on the form's configuration.

- Managing Form Input:

Upon validation, the form's data needs to be processed or stored. This can involve sending the data to a backend service, saving it locally, or using it within the app. TextEditingController instances associated with TextFormField widgets can be used to access the current value of form fields.

- Implementing Submission Logic:

Form submission typically involves validating the form fields and, if validation passes, processing the data. Submission is usually triggered by a button press. Flutter's ElevatedButton (or other button widgets) can be used to execute submission logic, which includes calling the form's validate method through its FormState.

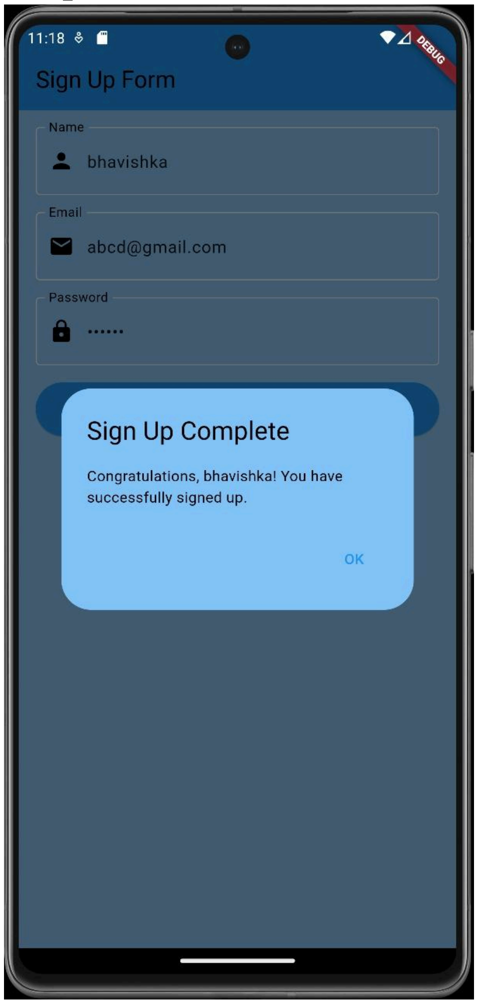
Code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MySignUpForm(),
      theme: ThemeData(
        primaryColor: Colors.blue,
        colorScheme: ColorScheme.fromSwatch(primarySwatch:
Colors.blue),
        fontFamily: 'Arial',
      ),
    );
  }
}

class MySignUpForm extends StatefulWidget {
  @override
  _MySignUpFormState createState() =>
_MySignUpFormState();
}

class _MySignUpFormState extends State<MySignUpForm> {
  final GlobalKey<FormState> _formKey =
GlobalKey<FormState>();
  final TextEditingController _nameController =
TextEditingController();
  String _email = '';
  String _password = '';

  String? _validateName(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your name';
    }
    return null;
  }

  String? _validateEmail(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    } else if
(!RegExp(r'^[\w-]+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,7}$')
        .hasMatch(value)) {
      return 'Please enter a valid email address';
    }
    return null;
  }

  String? _validatePassword(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your password';
    } else if (value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  }

  void _submitForm() {
    if (_formKey.currentState?.validate() ?? false) {
      _formKey.currentState?.save();
      _showSignUpCompleteDialog(_nameController.text);
    }
  }

  void _showSignUpCompleteDialog(String name) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Sign Up Complete'),
          content: Text('Congratulations, $name! You have
successfully signed up.'),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('OK'),
            ),
          ],
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Sign Up Form'),
        backgroundColor: Colors.blue, // Changed to blue
      ),
      body: Padding(
```

Name: Bhavishka Mulchandani Roll No.: 40 Class: D15B

```
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              TextFormField(
                controller: _nameController,
                decoration: InputDecoration(
                  labelText: 'Name',
                  hintText: 'Enter your name',
                  border: OutlineInputBorder(),
                  prefixIcon: Icon(Icons.person),
                ),
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black87,
                ),
                validator: _validateName,
                onSaved: (value) {
                  _nameController.text = value ?? '';
                },
              ),
              SizedBox(height: 16),
              TextFormField(
                decoration: InputDecoration(
                  labelText: 'Email',
                  hintText: 'Enter your email',
                  border: OutlineInputBorder(),
                  prefixIcon: Icon(Icons.email),
                ),
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black87,
                ),
                validator: _validateEmail,
                onSaved: (value) {
                  _email = value ?? '';
                },
              ),
              SizedBox(height: 16),
              TextFormField(
                obscureText: true,
                decoration: InputDecoration(
                  labelText: 'Password',
                  hintText: 'Enter your password',
                  border: OutlineInputBorder(),
                  prefixIcon: Icon(Icons.lock),
                ),
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black87,
                ),
                validator: _validatePassword,
                onSaved: (value) {
                  _password = value ?? '';
                },
              ),
              SizedBox(height: 16),
              ElevatedButton(
                onPressed: _submitForm,
                child: Text(
                  'Sign Up',
                  style: TextStyle(
                    fontSize: 18,
                    color: Colors.white,
                  ),
                ),
                style: ButtonStyle(
                  backgroundColor:
MaterialStateProperty.all(Colors.blue), // Changed to blue
                  padding: MaterialStateProperty.all(
                    EdgeInsets.symmetric(vertical: 12),
                  ),
                ),
                  ),
                ],
              ),
            ),
          ),
        );
  }
}
```

Output:



Sign Up Complete

Congratulations, bhavishka! You have successfully signed up.

Conclusion:

Interactive forms are essential for user input in mobile apps. Flutter offers a robust suite of widgets and tools for form creation, validation, and management, enabling developers to build complex forms with ease. Mastering these tools is crucial for designing intuitive and efficient forms in Flutter apps.