

## EXPERIMENT NO: - 04

**Name:-** Kunal Punjabi

**Class:-** D15A

**Roll:No: -** 43

**AIM: -** To create an interactive Form using form widget.

---

### **Theory: -**

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays 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 manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

### **Creation of a Form**

While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.

A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.

The **TextFormField widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.

To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.

Validation plays a crucial role in forms, and the **validator property** within **TextFormField** ensures user input meets specific criteria before submission.

Different types of input require appropriate **keyboard types**, such as `TextInputType.number` for numeric fields or `TextInputType.emailAddress` for email fields.

Proper **state management** is needed to store and retrieve user input, ensuring the form data is processed correctly.

A **submit button** is necessary to trigger form validation and submit the collected data for further processing.

### Some Properties of Form Widget

- **key:** A `GlobalKey` that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- **child:** The child widget that contains the form fields. Typically, this is a `Column`, `ListView`, or another widget that allows you to arrange the form fields vertically.
- **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

### Some Methods of Form Widget

- **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- **save():** This method is used to save the current values of all form fields. It invokes the `onSaved` callback for each field. Typically, this method is called after validation succeeds.
- **reset():** Resets the form to its initial state, clearing any user-entered data.
- **currentState:** A getter that returns the current `FormState` associated with the Form.

Code: -

```
import
'package:flutter/material.d
art';
import 'homepage.dart'; //
Correct import

class LoginPage extends
StatefulWidget {
  @override
```

```
_LoginPageState
createState() =>
  _LoginPageState();
}

class _LoginPageState extends
State<LoginPage> {
  final TextEditingController
_emailController =
TextEditingController();
```

```

    final
    TextEditingController
    _passwordController =
    TextEditingController();

    final
    GlobalKey<FormState>
    _formKey =
    GlobalKey<FormState>();

    void _login() {
      if
      (_formKey.currentState!.v
      alidate()) {
        String email =
        _emailController.text.trim(
        );

        String password =
        _passwordController.text.t
        rim();

        if (email ==
        "test@example.com" &&
        password ==
        "password123") {

          Navigator.pushReplaceme
          nt(
            context,

            MaterialPageRoute(builde
            r: (context) =>
            HomePage()), // cartItems
            removed
          );

```

```

    } else {

      ScaffoldMessenger.of(context)
      .showSnackBar(
        const SnackBar(content:
        Text("Invalid email or
        password")),
      );
    }
  }

  @override
  Widget build(BuildContext
  context) {
    return Scaffold(
      appBar: AppBar(title:
      const Text('Login')),
      body: Padding(
        padding: const
        EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
            CrossAxisAlignment.start,
            children: [
              TextFormField(
                controller:
                _emailController,
                decoration: const
                InputDecoration(labelText:
                'Email'),
                validator: (value) {

```

```

        if (value ==
null || value.isEmpty) {
            return 'Please
enter your email';
        }
        return null;
    },
),
    TextFormField(
        controller:
_passwordController,
        decoration: const
InputDecoration(labelText
: 'Password'),
        obscureText:
true,
        validator:
(value) {
            if (value ==
null || value.isEmpty) {
                return 'Please
enter your password';
            }
            return null;
        },
    ),
    const
SizedBox(height: 20),
    ElevatedButton(
        onPressed:
_login,
        child: const
Text('Login'),
    ),

```

```

    ],
),
),
),
);
}
}
signup.dart
import
'package:flutter/material.dart';
import 'homepage.dart';

class SignUpPage extends
StatefulWidget {
    @override
    _SignUpPageState
createState() =>
    _SignUpPageState();
}

class _SignUpPageState
extends State<SignUpPage> {
    final _formKey =
GlobalKey<FormState>();
    final TextEditingController
_passwordController =
TextEditingController();
    final TextEditingController
_confirmPasswordController
= TextEditingController();

    bool _isPasswordVisible =
false;
    bool

```

```

_isConfirmPasswordVisible
e = false;

void _signUp() {
  if
(_formKey.currentState!.va
alidate()) {

ScaffoldMessenger.of(cont
ext).showSnackBar(
  const
SnackBar(content:
Text("Sign-up successful!
Redirecting...")),
  );

  Future.delayed(const
Duration(seconds: 2), () {

Navigator.pushReplaceme
nt(
  context,
  MaterialPageRoute(
    builder: (context)
=> const HomePage(), //
Removed cartItems
argument
  ),
  );
});
}
}

@override

```

```

Widget build(BuildContext
context) {
  return Scaffold(
    appBar: AppBar(title:
const Text('Sign Up')),
    body:
SingleChildScrollView(
  child: Padding(
    padding: const
EdgeInsets.all(16.0),
    child: Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          const Text("Create
Account", style:
TextStyle(fontSize: 22,
fontWeight:
FontWeight.bold)),
          const
SizedBox(height: 10),

          // Name Field
          TextFormField(
            decoration: const
InputDecoration(
              labelText: 'Full
Name',
              border:
OutlineInputBorder(),
              prefixIcon:
Icon(Icons.person),

```

```

    ),
    validator:
(value) {
    if (value ==
null || value.isEmpty) {
        return
'Please enter your name';
    }
    return null;
},
),
const
SizedBox(height: 12),

```

```

// Email Field
TextFormField(
    decoration:
const InputDecoration(
    labelText:
'Email Address',
    border:
OutlineInputBorder(),
    prefixIcon:
Icon(Icons.email),
),
    keyboardType:
TextInputType.emailAddre
ss,
    validator:
(value) {
    if (value ==
null || value.isEmpty) {
        return
'Please enter your email';
    }
}
),
const
SizedBox(height: 12),

```

```

    },
    validator:
(value) {
    if (!RegExp(r'^[a-
zA-Z0-9_+-.]+@[a-zA-Z0-9-
]+\.[a-zA-Z0-9-
.]+$', value).hasMatch(value)) {
        return 'Enter a
valid email';
    }
    return null;
},
),
const
SizedBox(height: 12),

```

```

// Password Field
TextFormField(
    controller:
_passwordController,
    decoration:
InputDecoration(
    labelText:
'Password',
    border: const
OutlineInputBorder(),
    prefixIcon: const
Icon(Icons.lock),
    suffixIcon:
IconButton(
    icon:
Icon(_isPasswordVisible ?
Icons.visibility :
Icons.visibility_off),
    onPressed: () =>
setState(() =>

```

```

        !_isPasswordVisible =
!_isPasswordVisible),
    ),
    obscureText:
!_isPasswordVisible,
    validator:
(value) {
    if (value ==
null || value.isEmpty) {
        return
'Please enter a password';
    }
    if
(value.length < 6) {
        return
'Password must be at least
6 characters';
    }
    return null;
},
),
const
SizedBox(height: 12),

    // Confirm
Password Field
    TextFormField(
        controller:
!_confirmPasswordControll
er,
        decoration:
InputDecoration(
            labelText:

```

```

'Confirm Password',
        border: const
OutlineInputBorder(),
        prefixIcon: const
Icon(Icons.lock),
        suffixIcon:
IconButton(
            icon:
Icon(!_isConfirmPasswordVisi
ble ? Icons.visibility :
Icons.visibility_off),
            onPressed: () =>
setState(() =>
!_isConfirmPasswordVisible =
!_isConfirmPasswordVisible),
        ),
    ),
    obscureText:
!_isConfirmPasswordVisible,
    validator: (value) {
        if (value == null ||
value.isEmpty) {
            return 'Please
confirm your password';
        }
        if (value !=
!_passwordController.text) {
            return
'Passwords do not match';
        }
        return null;
    },
),
const

```

```
    SizedBox(height: 20),

    // Sign-Up
    Button
      SizedBox(
        width:
double.infinity,
        child:
ElevatedButton(
          onPressed:
            _signUp,
          child: const
            Text('Sign Up', style:
              TextStyle(fontSize: 16)),
        ),
      ),
    ],
  ),
),
),
),
);
}
```



SCREENSHOTS :


The image displays two side-by-side web forms for user authentication. The left form is titled 'Create account' and includes fields for 'Email' and 'Password', a 'Create account' button, and a 'Sign up with Google' button. The right form is titled 'Log in' and includes fields for 'Email' and 'Password', a 'Log in' button, and a 'Log in with Google' button. At the bottom of each form, there are links for users who already have an account or do not. In the 'Create account' form, the 'Log in' link is circled in orange. In the 'Log in' form, the 'Sign up' link is circled in orange. Two orange arrows originate from these circled links: one points from the 'Log in' link on the left to the 'Sign up with Google' button on the left, and the other points from the 'Sign up' link on the right to the 'Log in with Google' button on the right.

## Create account

Email

Password

Create account

 Sign up with Google

Have an account? [Log in](#)


## Log in

Email

Password

[Forgot your password](#)

Log in

 Log in with Google

No account? [Sign up](#)