

Experiment No: 4

AIM: To create an interactive Form using form widget

THEORY:

Flutter Form

Apps often require users to enter information into a text field. For example, you might require users to log in with an email address and password combination.

To make apps secure and easy to use, check whether the information the user has provided is valid. If the user has correctly filled out the form, process the information. If the user submits incorrect information, display a friendly error message letting them know what went wrong.

In this example, add validation to a form that has a single text field using the following steps:

1. Create a Form with a GlobalKey.
2. Add a TextFormField with validation logic.
3. Create a button to validate and submit the form.

1. Create a Form with a GlobalKey

First, create a Form. The Form widget acts as a container for grouping and validating multiple form fields.

When creating the form, provide a GlobalKey. This uniquely identifies the Form, and allows validation of the form in a later step.

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

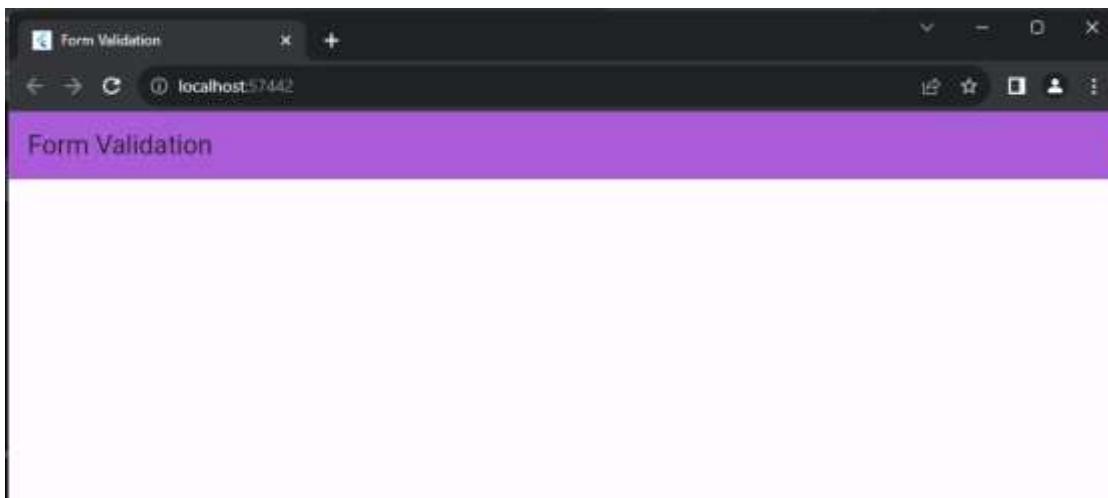
void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    const appTitle = 'Form Validation';

    return MaterialApp(
      title: appTitle,
      theme: ThemeData(primarySwatch: Colors.blue),
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
          backgroundColor: Color.fromARGB(255, 161, 101, 220),
        ),
        body: const MyCustomForm(),
      ),
    );
  }
}
```

```
    );  
  }  
}  
  
class MyCustomForm extends StatefulWidget {  
  const MyCustomForm({Key? key}) : super(key: key);  
  
  @override  
  MyCustomFormState createState() {  
    return MyCustomFormState();  
  }  
}  
  
class MyCustomFormState extends State<MyCustomForm> {  
  final _formKey = GlobalKey<FormState>();  
  
  @override  
  Widget build(BuildContext context) {  
    return Form(  
      key: _formKey,  
      child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [],  
      ),  
    );  
  }  
}
```



2. Add a TextFormField with validation logic

Although the Form is in place, it doesn't have a way for users to enter text. That's the job of a TextFormField. The **TextFormField** widget renders a material design text field and can display validation errors when they occur.

Validate the input by providing a **validator()** function to the TextFormField. If the user's input isn't valid, the validator function returns a String containing an error message. If there are no errors, the validator must return null.

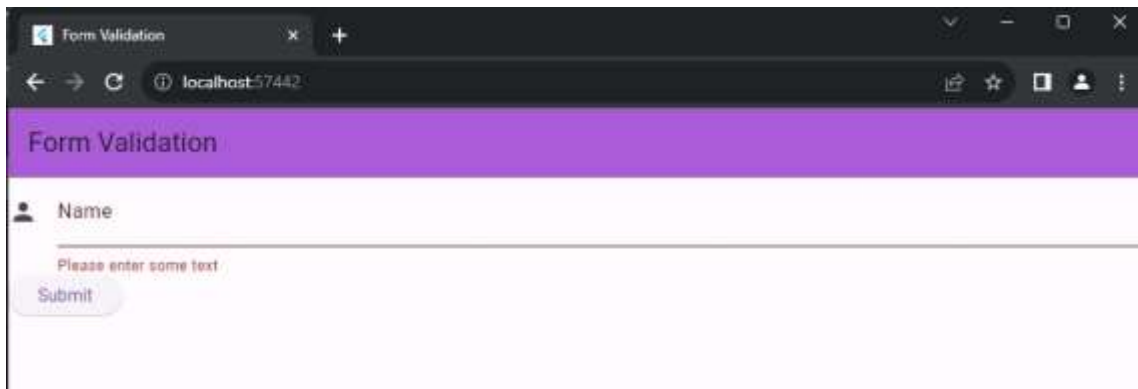
For this example, create a validator that ensures the TextFormField isn't empty. If it is empty, return a friendly error message.

First decorate your TextField to display hint text , icon and label text.

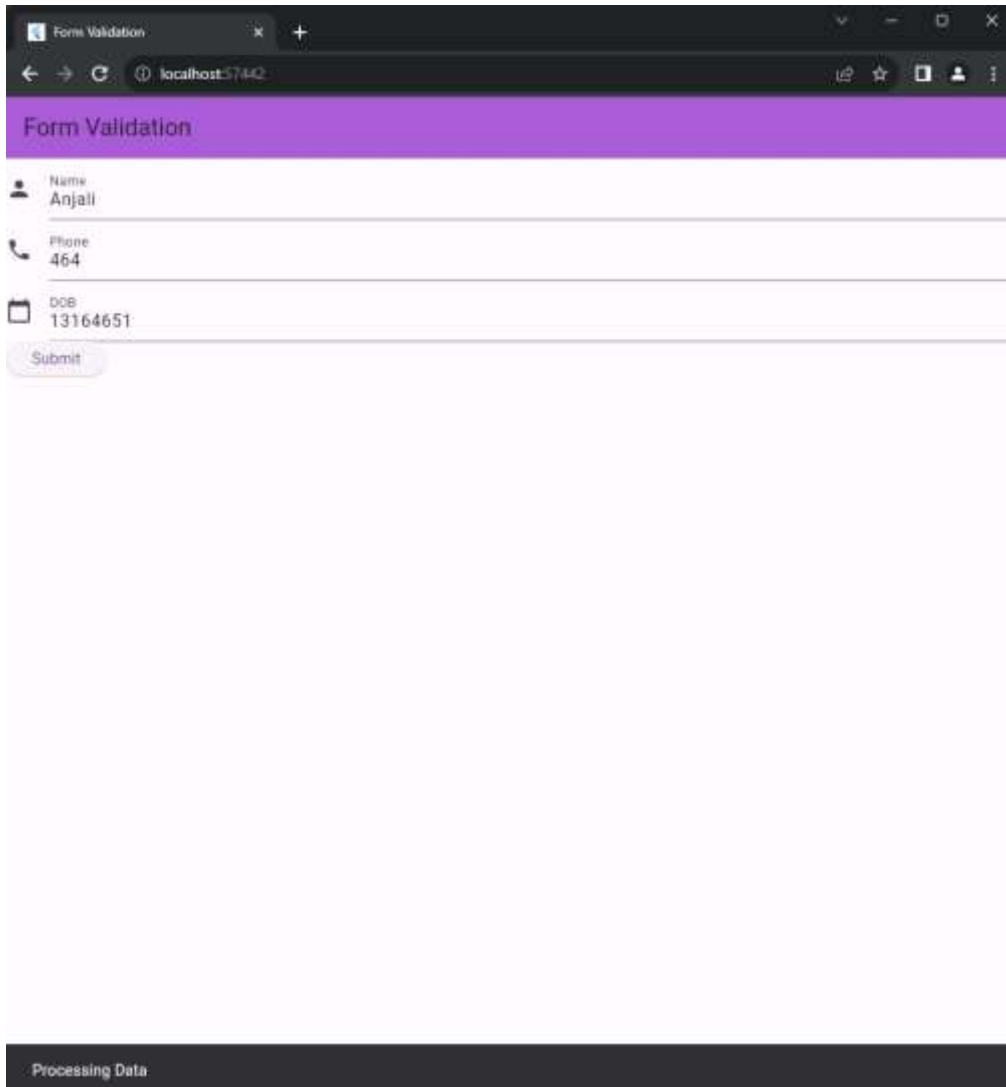
```
child: TextFormField(
  decoration: const InputDecoration(
    border: UnderlineInputBorder(),
    labelText: 'Enter your username',
  ),

class MyCustomFormState extends State<MyCustomForm> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          TextFormField(
            decoration: const InputDecoration(
              icon: Icon(Icons.person),
              hintText: 'Enter your name',
              labelText: 'Name',
            ),
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter some text';
              }
              return null;
            },
          ),
        ],
      ),
    );
  }
}
```

```
validator: (value) {  
  if (value == null || value.isEmpty) {  
    return 'Please enter some text';  
  }  
  return null;  
},  
,  
TextFormField(  
  decoration: const InputDecoration(  
    icon: Icon(Icons.phone),  
    hintText: 'Enter a phone number',  
    labelText: 'Phone',  
  ),  
,  
,  
TextFormField(  
  decoration: const InputDecoration(  
    icon: Icon(Icons.calendar_today),  
    hintText: 'Enter your date of birth',  
    labelText: 'DOB',  
  ),  
,  
,  
ElevatedButton(  
  onPressed: () {  
    // Validate will return true if the form is valid, or false if the form is invalid.  
    if (_formKey.currentState!.validate()) {  
      ScaffoldMessenger.of(context).showSnackBar(  
        const SnackBar(content: Text('Processing Data')),  
      );  
    }  
  },  
  child: const Text('Submit'),  
,  
,  
),  
,  
);  
}
```



Form Validation

Name
Anjali

Phone
464

DOB
13164651

Submit

Processing Data

```
children: [  
  TextFormField(  
    decoration: const InputDecoration(  
      icon: Icon(Icons.person),  
      hintText: 'Enter your name',  
      labelText: 'Name',  
    ),  
    validator: (value) {  
      if (value == null || value.isEmpty) {  
        return 'Please enter some text';  
      }  
      return null;  
    },  
  ),  
  TextFormField(  
    decoration: const InputDecoration(  
      icon: Icon(Icons.phone),  
      hintText: 'Enter a phone number',  
      labelText: 'Phone',  
    ),  
    validator: (value) {  
      if (value == null || value.isEmpty || value.length < 10) {  
        return 'Please enter phone number';  
      }  
    }  
  )  
]
```

```
        return null;
      },
    ),
    TextFormField(
      decoration: const InputDecoration(
        icon: Icon(Icons.calendar_today),
        hintText: 'Enter your date of birth',
        labelText: 'DOB',
      ),
    ),
    ElevatedButton(
      onPressed: () {
        // Validate will return true if the form is valid, or false if the form is invalid.
        if (_formKey.currentState!.validate()) {
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Processing Data')),
          );
        }
      },
      child: const Text('Submit'),
    ),
  ],
),
);
}
```

The screenshot shows a web browser window with the title 'Form Validation'. The address bar shows 'localhost:57442'. The page has a purple header with the text 'Form Validation'. Below the header, there are three input fields: 'Name' with the value 'anjali', 'Phone' with the value '123456', and 'DOB' with the value '01/09/97'. The 'Phone' field has a red error message 'Please enter phone number' below it. At the bottom, there is a 'Submit' button.

The screenshot shows a web browser window with the title 'Form Validation'. The address bar displays 'localhost:57442'. The page has a purple header with the text 'Form Validation'. Below the header, there are three input fields with labels and icons: 'Name' with a person icon, 'Phone' with a phone icon, and 'DOB' with a calendar icon. The values entered are 'anjali', '1234567890', and '01/09/97' respectively. A 'Submit' button is located below the DOB field. At the bottom of the page, there is a dark grey bar with the text 'Processing Data'.

Conclusion: - Hence we have successfully designed an interactive Form using form widget.

POSTLAB EXERCISE:

- Use of FormBuilder and FormBuilderValidators

https://pub.dev/packages/flutter_form_builder

[https://pub.dev/packages/form_builder_validators/](https://pub.dev/packages/form_builder_validators)



```
import 'package:flutter/material.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:form_builder_validators/form_builder_validators.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
```

```
void main() => runApp(const MyApp());
```

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});
```

```
  @override
```

```
  Widget build(BuildContext context) {
    const appTitle = 'Form Validation Demo';
```

```
    return MaterialApp(
      title: appTitle,
      localizationsDelegates: const [
        FormBuilderLocalizations.delegate,
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
```

```
    ],
    supportedLocales: const [
```

```
      Locale('en', ''),
      Locale('es', ''),
      Locale('fa', ''),
      Locale('fr', ''),
      Locale('ja', ''),
      Locale('pt', ''),
      Locale('sk', ''),
      Locale('pl', ''),
```

```
    ],
    home: Scaffold(
      appBar: AppBar(
        title: const Text(appTitle),
      ),
      body: const MyCustomForm(),
    ),
  );
}
```

```
}
```

```
// Create a Form widget.
```

```
class MyCustomForm extends StatefulWidget {
  const MyCustomForm({super.key});
```

```
  @override
```

```
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
```

```
}
```

```
}
```

```
// Create a corresponding State class.
```

```
// This class holds data related to the form.
```

```
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
```

PRN: 121A3014

```
// Note: This is a GlobalKey<FormState>,
// not a GlobalKey<MyCustomFormState>.
final _formKey = GlobalKey<FormState>();

var genderOptions = ['Male', 'Female', 'Others'];

//code for sample form

@override
Widget build(BuildContext context) {
  return Column(
    children: <Widget>[
      FormBuilder(
        key: _formKey,
        //autovalidateMode: AutovalidateMode.always,
        child: Column(
          children: <Widget>[
            FormBuilderFilterChip(
              name: 'filter_chip',
              decoration: const InputDecoration(
                labelText: "Select many options",
              ),
              options:const [
                //FormBuilderFieldChipOption(value: 'Test', child: Text('Test')),
                /* FormBuilderFieldOption(
                  value: 'Test 1', child: Text(AppLocalizations(myLocale)
                    .translate('mobile')),), */
                FormBuilderChipOption(
                  value: 'Test 1', child: Text('Test 1')),
                FormBuilderChipOption(
                  value: 'Test 2', child: Text('Test 2')),
                /* FormBuilderFieldOption(
                  value: 'Test 3', child: Text('Test 3')),
                FormBuilderFieldOption(
                  value: 'Test 4', child: Text('Test 4')), */
              ],
            ),
            FormBuilderChoiceChip(
              name: 'choice_chip',
              decoration: const InputDecoration(
                labelText: 'Select an option',
              ),
              options: const [
                FormBuilderChipOption(
                  value: 'Test 1', child: Text('Test 1')),
                FormBuilderChipOption(
                  value: 'Test 2', child: Text('Test 2')),
              ],
            ),
            FormBuilderDateTimePicker(
              name: 'date',
              // onChanged: _onChanged,
              inputType: InputType.time,
              decoration: const InputDecoration(
                labelText: 'Appointment Time',
              ),
            ),
            initialTime: const TimeOfDay(hour: 8, minute: 0),
            // initialValue: DateTime.now(),
```

```
// enabled: true,
),
FormBuilderDateRangePicker(
  name: 'date_range',
  firstDate: DateTime(1970),
  lastDate: DateTime(2030),
  //format: DateFormat('yyyy-MM-dd'),
  //onChanged: _onChanged,
  decoration: const InputDecoration(
    labelText: 'Date Range',
    helperText: 'Helper text',
    hintText: 'Hint text',
  ),
),
FormBuilderSlider(
  name: 'slider',
  validator: FormBuilderValidators.compose([
    FormBuilderValidators.min(6),
  ]),
  //onChanged: _onChanged,
  min: 0.0,
  max: 10.0,
  initialValue: 7.0,
  divisions: 20,
  activeColor: Colors.red,
  inactiveColor: Colors.pink[100],
  decoration: const InputDecoration(
    labelText: 'Number of things',
  ),
),
FormBuilderCheckbox(
  name: 'accept_terms',
  initialValue: false,
  //onChanged: _onChanged,
  title: RichText(
    text: const TextSpan(
      children: [
        TextSpan(
          text: 'I have read and agree to the ',
          style: TextStyle(color: Colors.black),
        ),
        TextSpan(
          text: 'Terms and Conditions',
          style: TextStyle(color: Colors.blue),
        ),
      ],
    ),
  ),
  validator: FormBuilderValidators.equal(
    true,
    errorText: 'You must accept terms and conditions to continue',
  ),
),
FormBuilderTextField(
  name: 'age',
  decoration: const InputDecoration(
```

```
      labelText:
        'This value is passed along to the [Text.maxLines] attribute of the [Text] widget used to display the hint
text.',
    ),
    //onChanged: _onChanged,
    // valueTransformer: (text) => num.tryParse(text),
    validator: FormBuilderValidators.compose([
      FormBuilderValidators.required(),
      FormBuilderValidators.numeric(),
      FormBuilderValidators.max(70),
    ]),
    keyboardType: TextInputType.number,
  ),
  FormBuilderDropdown(
    name: 'gender',
    decoration: const InputDecoration(
      labelText: 'Gender',
    ),
    // initialValue: 'Male',
    /* allowClear: true,
    hint: const Text('Select Gender'), */
    validator: FormBuilderValidators.compose(
      [FormBuilderValidators.required()]),
    items: genderOptions
      .map((gender) => DropdownMenuItem(
        value: gender,
        child: Text(gender),
      ))
      .toList(),
  ),
],
),
),
),
Row(
  children: <Widget>[
    Expanded(
      child: MaterialButton(
        color: Theme.of(context).colorScheme.secondary,
        child: const Text("Submit",
          style: TextStyle(color: Colors.white),
        ),
        onPressed: () {
          _formKey.currentState?.save();
          bool? t = _formKey.currentState?.validate();
          if (t == true) {
            ScaffoldMessenger.of(context).showSnackBar(
              const SnackBar(content: Text('Processing Data')));
          } else {
            ScaffoldMessenger.of(context).showSnackBar(
              const SnackBar(content: Text('Incorrect Data')));
          }
        },
      ),
    ),
  ],
  const SizedBox(width: 20),

```

```
child: MaterialButton(  
  color: Theme.of(context).colorScheme.secondary,  
  child: const Text(  
    "Reset",  
    style: TextStyle(color: Colors.white),  
  ),  
  onPressed: () {  
    _formKey.currentState?.reset();  
  },  
),  
),  
],  
)  
],  
);  
}  
}
```

Form Validation Demo

Select many options

✓ Test 1 ✓ Test 2

Select an option

Test 1 ✓ Test 2

Appointment Time

18:25

Date Range

2/22/2024 - 2/29/2024

Helper text

Number of things

0 5 10

☒ I have read and agree to the [Terms and Conditions](#)

This value is passed along to the [Text.maxLines] attribute of the [Text] widget used to display the hint text.

Gender

Female

Submit Reset

Output:

Conclusion: In conclusion, utilizing the Flutter Form widget streamlines user input handling and validation, fostering a structured and efficient development process. The integration of key components like `TextFormField` ensures a user-friendly interface, while the use of `GlobalKey<FormState>` enables effective form state management. The addition of a `FloatingActionButton` provides an intuitive submission trigger. Overall, Flutter's form implementation offers a versatile and responsive solution for creating dynamic user interactions in mobile applications.