

EXPERIMENT NO. 4

Aim: To create an interactive Form using a form widget

Theory:

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.

Form Widget:

The Form widget in Flutter provides a way to manage and validate a group of form fields. It is a container for various form-related widgets.

A form usually consists of multiple form fields, such as text fields, checkboxes, radio buttons, and buttons.

Form Fields:

1.TextFormField:

The TextFormField widget is used for single-line text input. It automatically handles validation, error messages, and updating the form state.

Example:

```
TextFormField(  
  decoration: InputDecoration(labelText: 'Username'),  
  validator: (value) {  
    if (value == null || value.isEmpty) {  
      return 'Please enter your username';  
    }  
    return null;  
  },  
)
```

2.DropdownButtonFormField:

The DropdownButtonFormField widget creates a dropdown menu and automatically manages its state.

Example:

```
DropdownButtonFormField<String>(  
  value: selectedCountry,  
  items: ['USA', 'Canada', 'UK'].map((String country) {  
    return DropdownMenuItem<String>(  
      value: country,
```

```
        child: Text(country),
      );
    }).toList(),
    onChanged: (value) {
      setState(() {
        selectedCountry = value;
      });
    },
    decoration: InputDecoration(labelText: 'Country'),
  )
```

3.CheckboxFormField:

The CheckboxFormField widget is used for checkbox input.

Example:

```
CheckboxFormField(
  initialValue: false,
  title: 'Accept Terms and Conditions',
  validator: (value) {
    if (value != true) {
      return 'Please accept the terms and conditions';
    }
    return null;
  },
)
```

Form Validation:

Form validation ensures that the user input meets specific criteria before allowing form submission. It helps maintain data integrity.

Validation is typically done using the validator property of form fields. Validators are functions that return an error message if the input is invalid or null if it's valid.

1.Handling Form Submission:

Use the onSave property of form fields to specify a function that will be called when the form is saved or submitted.

Example:

```
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      // If the form is valid, save or process the data.
      _formKey.currentState!.save();
      // Perform form submission logic here
    }
  },
)
```

```
    }  
  },  
  child: Text('Submit'),  
)
```

2.GlobalKey<FormState>:

The GlobalKey<FormState> is used to uniquely identify the Form widget. It allows access to the Form widget's state, which is necessary for form validation and submission.

Example:

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

Code:

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

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

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

```
  @override  
  Widget build(BuildContext context) {  
    return const MaterialApp(  
      home: MyForm(),  
    );  
  }  
}
```

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

```
  @override  
  // ignore: library_private_types_in_public_api  
  _MyFormState createState() => _MyFormState();  
}
```

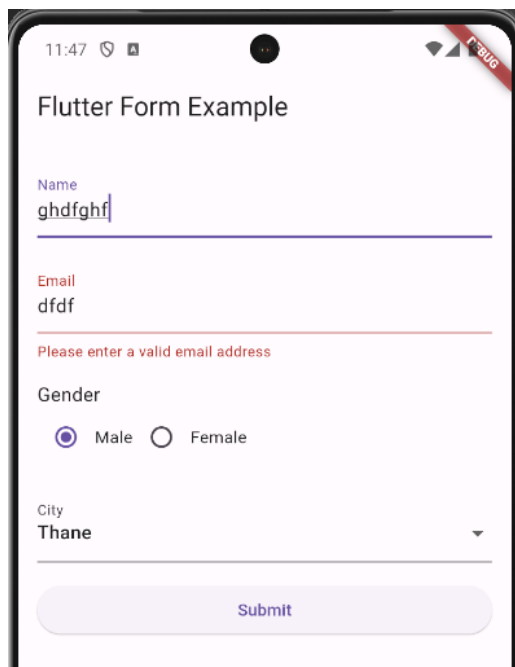
```
class _MyFormState extends State<MyForm> {  
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  String? _gender;  
  String? _selectedCity;  
  
  final List<String> _cities = ['Mumbai', 'Thane', 'Chembur', 'Kurla'];
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Flutter Form Example'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextFormField(
              decoration: const InputDecoration(labelText: 'Name'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your name';
                }
                return null;
              },
              onSave: (value) {
                // ...
              },
            ),
            const SizedBox(height: 16),
            TextFormField(
              decoration: const InputDecoration(labelText: 'Email'),
              validator: (value) {
                if (value == null || value.isEmpty || !value.contains('@')) {
                  return 'Please enter a valid email address';
                }
                return null;
              },
              onSave: (value) {
                // ...
              },
            ),
            const SizedBox(height: 16),
            const Text('Gender', style: TextStyle(fontSize: 16)),
            Row(
              children: [
                Radio(
                  value: 'Male',
                  groupValue: _gender,
```

```
        onChanged: (value) {
          setState(() {
            _gender = value.toString();
          });
        },
      ),
      const Text('Male'),
      Radio(
        value: 'Female',
        groupValue: _gender,
        onChanged: (value) {
          setState(() {
            _gender = value.toString();
          });
        },
      ),
      const Text('Female'),
    ],
  ),
  const SizedBox(height: 16),
  DropdownButtonFormField<String>(
    value: _selectedCity,
    items: _cities.map((city) {
      return DropdownMenuItem<String>(
        value: city,
        child: Text(city),
      );
    }).toList(),
    onChanged: (value) {
      setState(() {
        _selectedCity = value;
      });
    },
    decoration: const InputDecoration(labelText: 'City'),
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please select a city';
      }
      return null;
    },
  ),
  const SizedBox(height: 16),
  ElevatedButton(
    onPressed: () {
```

```
        _submitForm();
      },
      child: const Text('Submit'),
    ),
  ],
),
),
),
);
}
```

```
void _submitForm() {
  if (_formKey.currentState!.validate()) {
    _formKey.currentState!.save();
  }
}
```

Output:

The screenshot shows a mobile application titled "Flutter Form Example". It features a form with the following elements:

- Name:** A text input field containing "ghdfghf".
- Email:** A text input field containing "dfdf". Below it, a red error message reads "Please enter a valid email address".
- Gender:** Two radio button options: "Male" (selected) and "Female".
- City:** A dropdown menu showing "Thane".
- Submit:** A light blue button with the text "Submit".

Conclusion:

In summary, this experiment has demonstrated the effective utilization of form widgets in Flutter to construct interactive and user-friendly forms within applications. We created a form with a Global field and then added a textform field with validation logic then created a button to validate and submit the form. We can add the form functionality to create our login and sign-up page while creating our project and do user authentication.