Experiment No. : 04

Aim: To create an interactive Form using form widget

Theory:

Flutter Forms

Forms are an integral part of all modern mobile and web applications. It is mainly used to interact with the app as well as gather information from the users. They can perform many tasks, which depend on the nature of your business requirements and logic, such as authentication of the user, adding user, searching, filtering, ordering, booking, etc. A form can contain text fields, buttons, checkboxes, radio buttons, etc.

Creating Form

Flutter provides a Form widget to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. 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.

Form validation

Validation is a method, which allows us to correct or confirms a certain standard. It ensures the authentication of the entered data.
Validating forms is a common practice in all digital interactions. To validate a form in a flutter, we need to implement mainly three steps.

Now let's go through the key components and features used in the code below:
1.Material Design: The app follows the Material Design principles provided by the Flutter framework, ensuring a consistent and visually appealing user interface.
2.StatefulWidget: The SignUpForm class is a stateful widget, allowing it to maintain state (such as the user input) and update the UI accordingly.

3.Form Widget: The Form widget is used to group the form fields together. It allows us to perform form validation and submission easily.
4.GlobalKey: The GlobalKey<FormState> is used to uniquely identify the form widget. This key is necessary for performing operations like form validation and saving.
5.TextFormField: These widgets are used for user input. EachTextFormField represents a field in the form, such as username, email, and password. They include properties like decoration for styling and validator for input validation.
6.Padding: The Padding widget adds padding around its child widget, ensuring proper spacing between UI elements.
7.SizedBox: These widgets are used to add empty space (vertical height in this case) between UI elements. They help improve the layout and readability of the form.
8.ElevatedButton: This button widget triggers the sign-up action when pressed. It is styled as an elevated button, following the Material Design guidelines.
9.AlertDialog: When the sign-up process is complete, an AlertDialog is shown to inform the user about the successful sign-up. It includes a title, content, and an "OK" button to close the dialog.
10.showDialog: The showDialog function is used to display the AlertDialog on the screen. It takes the BuildContext and a builder function that returns the dialog widget.
11.Image.asset: This widget is used to display the logo image at the top of the sign-up form. It loads the image from the assets directory using the provided image path.
12.TextButton: This button widget is used inside the AlertDialog for dismissing the dialog when pressed.
Overall, these components work together to create an interactive sign-up form with input validation and a user-friendly interface.

Code:

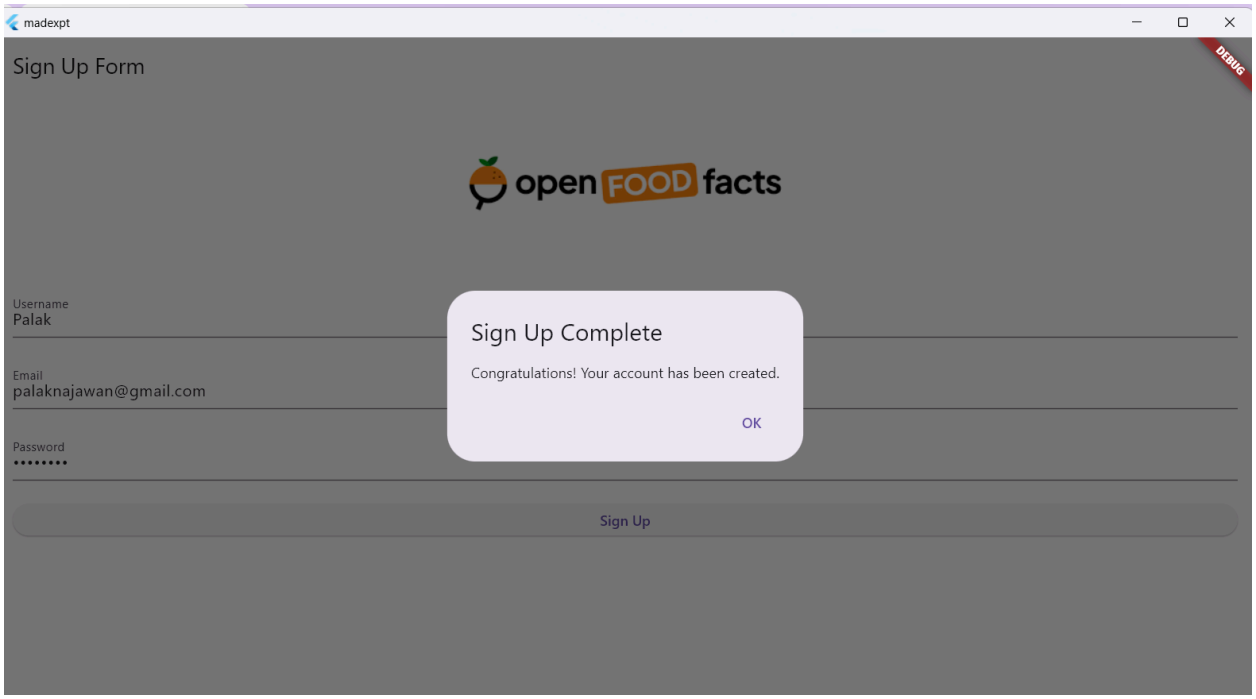
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
```

```
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Sign Up Form',
     theme: ThemeData(
       primarySwatch: Colors.blue,
       visualDensity: VisualDensity.adaptivePlatformDensity,
     ),
     home: SignUpForm(),
   );
 }
}

class SignUpForm extends StatefulWidget {
 @override
 _SignUpFormState createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {
 final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
 TextEditingController _usernameController = TextEditingController();
 TextEditingController _emailController = TextEditingController();
 TextEditingController _passwordController = TextEditingController();

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Sign Up Form'),
     ),
     body: Padding(
       padding: const EdgeInsets.all(16.0),
       child: Form(
         key: _formKey,
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.stretch,
           children: <Widget>[
             Image(
               image: AssetImage('images/download.png'),
               height: 150.0,
               width: 150.0,
             ),
             SizedBox(height: 24.0),
             TextFormField(
               controller: _usernameController,
               decoration: InputDecoration(labelText: 'Username'),
               validator: (value) {
                 if (value == null || value.isEmpty) {
                   return 'Please enter your username';
```

```
                }
                return null;
              },
            ),
            SizedBox(height: 16.0),
            TextFormField(
              controller: _emailController,
              decoration: InputDecoration(labelText: 'Email'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your email';
                }
                if (!value.contains('@')) {
                  return 'Please enter a valid email';
                }
                return null;
              },
            ),
            SizedBox(height: 16.0),
            TextFormField(
              controller: _passwordController,
              decoration: InputDecoration(labelText: 'Password'),
              obscureText: true,
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your password';
                }
                if (value.length < 6) {
                  return 'Password must be at least 6 characters long';
                }
                return null;
              },
            ),
            SizedBox(height: 24.0),
            ElevatedButton(
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  _showSignUpCompleteDialog();
                }
              },
              child: Text('Sign Up'),
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
void _showSignUpCompleteDialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Sign Up Complete'),
        content: Text('Congratulations! Your account has been created.'),
        actions: <Widget>[
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: Text('OK'),
          ),
        ],
      );
    },
  );
}

@override
void dispose() {
  _usernameController.dispose();
  _emailController.dispose();
  _passwordController.dispose();
  super.dispose();
}
}
```

Output:



Conclusion:

In conclusion, creating an interactive form using the form widget in Flutter enables developers to build robust, user-friendly applications that efficiently collect and process user input. This approach enhances usability, improves data accuracy, and contributes to a positive user experience.