Experiment 4

Aim: Creating an Interactive Form Using Form Widget in Flutter

Guidelines:

Introduction:

Flutter is a UI toolkit that enables the development of natively compiled applications for mobile, web, and desktop from a single codebase. To create an interactive form using a form widget in Flutter, we can utilize the `Form` widget along with various form-related widgets provided by Flutter.

Guidelines for Creating an Interactive Form:

1. Import Flutter Packages:
Start by importing the necessary Flutter packages. The `flutter/material.dart` package is essential for building the user interface.

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

2. Create a StatefulWidget:
Use the `StatefulWidget` to create a stateful widget that will contain the form and handle its state changes.

~~~
```dart
class InteractiveForm extends StatefulWidget {
  @override
  _InteractiveFormState createState() => _InteractiveFormState();
}
```
~~~

3. Create State Class:
Inside the state class, define the variables and controllers for form elements.

~~~
```dart
class _InteractiveFormState extends State<InteractiveForm> {
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
~~~

~~~
    // Add more controllers for other form elements
  }
  ```
~~~

4. Build Form Widget:
  Use the `Form` widget to wrap the form elements. Utilize various form-related widgets like `TextFormField`, `Checkbox`, `Radio`, `DropdownButton`, etc.

```dart
  @override
  Widget build(BuildContext context) {
    return Form( child: Column(
        children: [
          TextFormField( controller: _nameController,
            decoration: InputDecoration(labelText:
            'Name'),
          ),
          TextFormField( controller: _emailController,
            decoration: InputDecoration(labelText:
            'Email'),
          ),
          // Add more form elements
          ElevatedButton(
            onPressed: () {
              // Handle form submission
            },
            child: Text('Submit'),
          ),
        ],
      ),
    );
  }
```

5. Form Validation:
  Implement validation logic using the `validator` property of the `TextFormField` widget or custom validation functions.

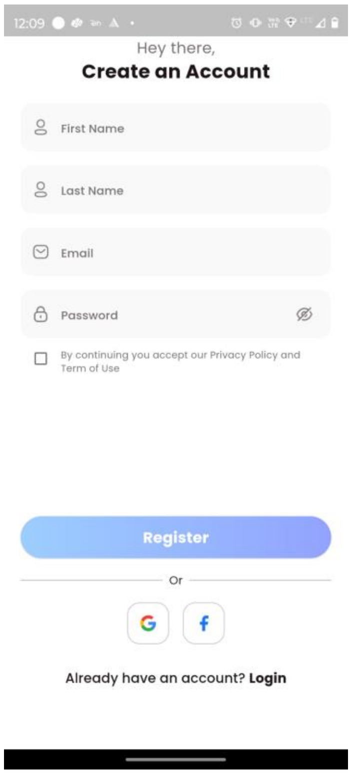
```dart
  TextFormField( controller: _emailController,
    decoration: InputDecoration(labelText: 'Email'),
    validator: (value) {
      if (value.isEmpty || !value.contains('@')) {
```

```
        return 'Invalid email format';
      }
      return null;
    },
  ),
  ```
```

6. Handle Form Submission:
  Add a callback function to handle the form submission when the submit button is pressed.

```
  ```dart
  ElevatedButton(
    onPressed: () {
      if (Form.of(context).validate()) {
        // Form is valid, handle submission
        // Access form field values using controllers (e.g., _nameController.text)
      }
    },
    child: Text('Submit'),
  ),
  ```
```

Screenshot:
```


12:09
Hey there,
Create an Account
First Name
Last Name
Email
Password
By continuing you accept our Privacy Policy and Term of Use
Register
Or
Already have an account? Login

Conclusion:

Creating an interactive form using the form widget in Flutter involves utilizing various form-related widgets, handling form validation, and implementing a submission mechanism. This experiment provides a practical guide to building interactive forms in Flutter, facilitating user input and enhancing the user experience in Flutter applications.