Experiment 4

Aim:To create an interactive Form using form widget

Theory:

- Form Widget: The Form widget is the container for form fields. It manages the form state and provides methods to validate, save, reset, and submit the form.
- Form Fields: Form fields are input fields where users can enter data. Flutter provides
 various types of form fields, such as TextFormField, DropdownButtonFormField,
 CheckboxFormField, RadioFormField, etc. Each form field has its own properties and
 behavior.
- Submission: Form submission involves taking the data entered by the user and
 processing it, such as sending it to a server, saving it locally, or performing other actions
 based on the application's logic. In Flutter, you can handle form submission by
 implementing a callback function that gets triggered when the form is submitted.
- Error Handling: Error handling in forms involves displaying error messages to the user when the data entered is invalid or when there are issues with form submission

Code:

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

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

class MyApp extends StatelessWidget {
   @override
   Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
      theme: ThemeData(
         primaryColor: Colors.blue,
      hintColor: Colors.blueAccent,
      inputDecorationTheme: InputDecorationTheme(
          border: OutlineInputBorder(),
```

```
contentPadding: EdgeInsets.symmetric(vertical: 12.0,
horizontal: 16.0),
        ),
      ),
   );
  }
}
class MyHomePage extends StatefulWidget {
  @override
  MyHomePageState createState() => MyHomePageState();
}
class MyHomePageState extends State<MyHomePage> {
  final formKey = GlobalKey<FormState>();
  String name = "";
  String email = "";
  String password = "";
  String confirmPassword = "";
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('ToDo App Validation Form'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Form (
          key: formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              TextFormField(
                decoration: InputDecoration(
                  labelText: "Name",
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
```

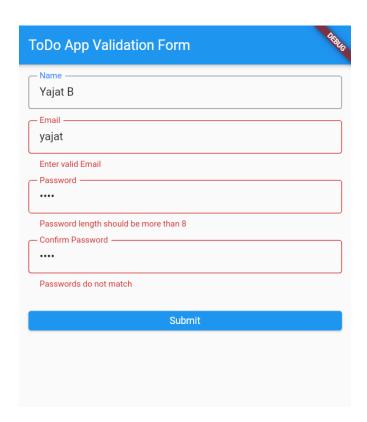
Roll No:5

```
return "Please enter your name";
                  }
                  if (RegExp(r'\d').hasMatch(value)) {
                    return "Name should not contain numbers";
                  return null;
                },
                onSaved: (value) => name = value!,
              ),
              SizedBox (height: 16.0),
              TextFormField(
                decoration: InputDecoration(
                  labelText: "Email",
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return "Please enter your email";
                  }
                  if (!RegExp(
r"^{[a-zA-Z0-9.a-zA-Z0-9.!\#\$\%\&"*+/=?^{[a-zA-Z0-9-]+}.[a-zA-Z]}
+")
                      .hasMatch(value!)) {
                    return "Enter valid Email";
                  return null;
                },
                onSaved: (value) => email = value!,
              ),
              SizedBox (height: 16.0),
              TextFormField(
                decoration: InputDecoration (
                  labelText: "Password",
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return "Please enter your password";
                  if (value.length < 8) {</pre>
```

return "Password length should be more than 8";

```
}
                  return null;
                },
                obscureText: true,
                onSaved: (value) => password = value!,
              ),
              SizedBox (height: 16.0),
              TextFormField(
                decoration: InputDecoration (
                  labelText: "Confirm Password",
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return "Please confirm your password";
                  if (value != password) {
                    return "Passwords do not match";
                  }
                  return null;
                },
                obscureText: true,
                onSaved: (value) => confirmPassword = value!,
              SizedBox (height: 32.0),
              ElevatedButton(
                onPressed: () {
                  final isValid = formKey.currentState!.validate();
                  if (isValid) {
                    formKey.currentState!.save();
                    print(
                        "Name: $ name, Email: $ email,
Password:$ password");
                    // Handle successful form submission here
                  }
                },
                child: Text('Submit'),
              ),
            ],
```

```
),
),
);
}
```



Conclusion: In summary, using a form widget to create interactive forms enhances user engagement and data collection. It streamlines information input, offers customization options, and improves overall user experience. Interactive forms are versatile tools for surveys, registrations, and feedback mechanisms, facilitating efficient data capture and processing in digital environments.