Name:- Roshni Talreja Class:- D15B Roll No:- 66

Experiment No: 4

Aim: To create a simple interactive form in Flutter, including form validation and data processing on submission. The form includes text input fields, a dropdown menu, and a checkbox.

Theory:
In Flutter, a form is a collection of input fields and interactive widgets that allow users to enter and submit data. Forms are commonly used in mobile applications to collect user information, preferences, or feedback. Flutter provides the `Form` widget to manage form state, validate input, and handle form submission efficiently.

Key concepts and terms related to forms in Flutter include:

1. Form Widget: The `Form` widget is a container for form fields and validators. It manages the state of the form, including validation and submission. The `Form` widget should typically be placed within a `Scaffold` widget to provide a material design layout.

2. FormField Widget: The `FormField` widget represents an individual input field within a form. It is a generic widget that can hold various types of form fields such as text fields, dropdowns, checkboxes, radio buttons, and more.

3. TextFormField Widget: The `TextFormField` widget is a specialized form field widget used to capture text input from the user. It provides built-in validation capabilities for common input types like email, password, and numeric fields. Developers can customize validation logic using the `validator` parameter.

4. DropdownButtonFormField Widget: The `DropdownButtonFormField` widget is used to create a dropdown menu within a form. It allows users to select one option from a list of predefined choices. Developers can specify the list of items, as well as handle the selection and validation of the dropdown value.

5. CheckboxListTile Widget: The `CheckboxListTile` widget represents a checkbox that the user can toggle on or off. It includes a title, subtitle, and trailing icon for enhanced usability. Developers can use the `value` and `onChanged` properties to control the state of the checkbox.

6. Form Validation: Form validation is the process of ensuring that user input meets specified criteria or constraints before allowing submission. Flutter provides a built-in mechanism for form validation using the `validator` property of form fields. Developers

can define custom validation functions and error messages to handle different validation scenarios.

7. Form Submission: Form submission refers to the process of capturing user input and processing it in response to a user action, such as pressing a submit button. In Flutter, developers can use the `onPressed` event handler of a button widget to trigger form submission. Within the event handler, they can access the current state of the form and perform data processing or submit data to a backend server.

CODE & OUTPUT :-

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const MyLogInForm(),
      theme: ThemeData(
        primaryColor: Colors.blue, // Changed primary color to
blue
        colorScheme: ColorScheme.fromSwatch(primarySwatch:
Colors.blue), // Adjusted color scheme
        fontFamily: 'Arial',
      ),
    );
  }
}

class MyLogInForm extends StatefulWidget {
```

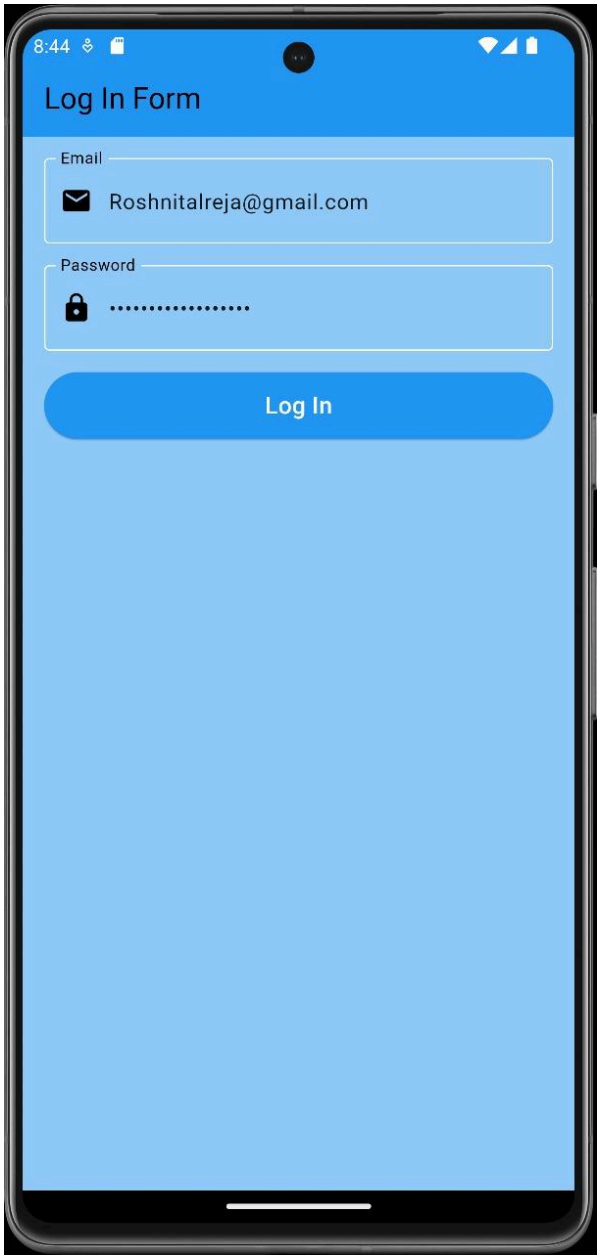
```
  const MyLogInForm({Key? key}) : super(key: key);

  @override
  _MyLogInFormState createState() => _MyLogInFormState();
}

class _MyLogInFormState extends State<MyLogInForm> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();

  String? _validateEmail(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    } else if
(!RegExp(r'^[\w-]+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,7}$').hasMat
ch(value)) {
      return 'Please enter a valid email address';
    }
    return null;
  }

  String? _validatePassword(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your password';
    } else if (value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  }

  void _submitForm() {
```

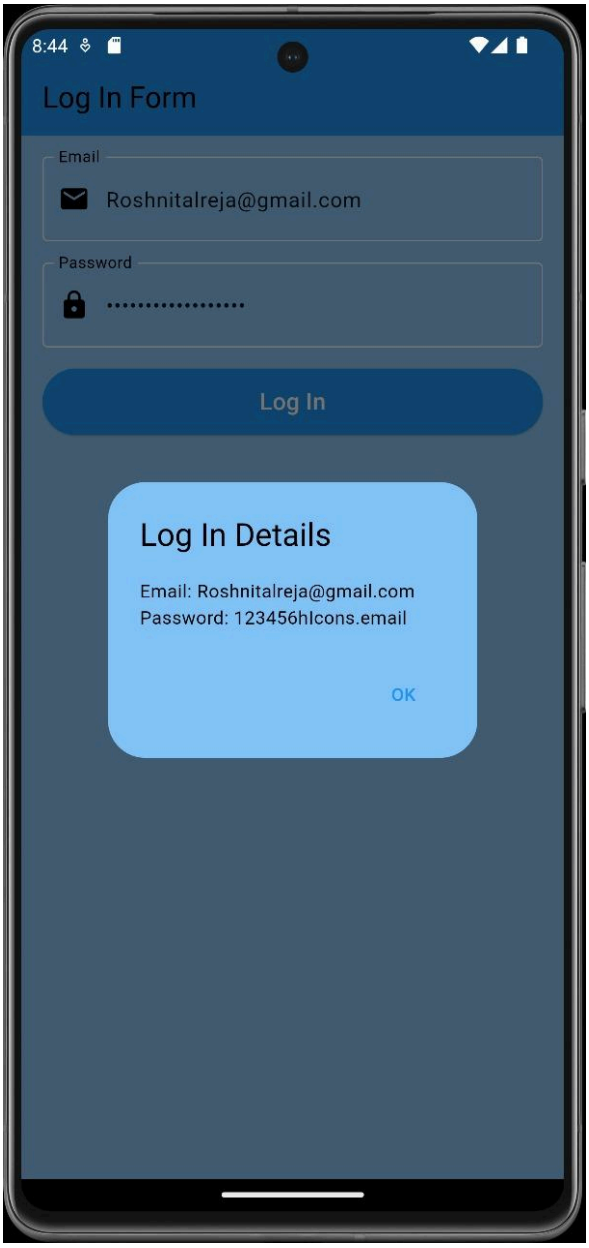
```
    if (_formKey.currentState?.validate() ?? false) {
      _formKey.currentState?.save();
      _showLogInDetails();
    }
  }

  void _showLogInDetails() {
    String email = _emailController.text;
    String password = _passwordController.text;
    // In a real app, you would handle authentication here
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Log In Details'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text('Email: $email'),
              Text('Password: $password'),
            ],
          ),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('OK'),
            ),
          ],
        );
      },
    );
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Log In Form'),
      backgroundColor: Colors.blue, // Changed to blue
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextFormField(
              controller: _emailController,
              decoration: const InputDecoration(
                labelText: 'Email',
                hintText: 'Enter your email',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.email),
              ),
              style: const TextStyle(
                fontSize: 16,
                color: Colors.black87,
              ),
              validator: _validateEmail,
            ),
            const SizedBox(height: 16),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
```

```
                decoration: const InputDecoration(
                  labelText: 'Password',
                  hintText: 'Enter your password',
                  border: OutlineInputBorder(),
                  prefixIcon: Icon(Icons.lock),
                ),
                style: const TextStyle(
                  fontSize: 16,
                  color: Colors.black87,
                ),
                validator: _validatePassword,
              ),
              const SizedBox(height: 16),
              ElevatedButton(
                onPressed: _submitForm,
                style: ButtonStyle(
                  backgroundColor:
MaterialStateProperty.all(Colors.blue), // Changed to blue
                  padding: MaterialStateProperty.all(
                    const EdgeInsets.symmetric(vertical: 12),
                  ),
                ),
                child: const Text(
                  'Log In',
                  style: TextStyle(
                    fontSize: 18,
                    color: Colors.white,
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
```

```
      );
    }
}
```





Conclusion: In this code, we have successfully created a simple interactive form in Flutter with text input fields for email and password. The form includes form validation to ensure that the user enters valid data before submission.