MPL Lab 4
Mayur Jaiswal
D15B
24

Aim: Develop an interactive form in Flutter utilizing the Form and TextFormField widgets to efficiently handle user input and validation.

Theory: In Flutter, forms are essential for collecting and validating user input. The Form widget acts as a container for form fields, managing their state and validation. TextFormField is a commonly used widget for single-line text input, offering built-in validation and state management. By combining these widgets, developers can create robust forms that provide real-time feedback to users, enhancing the overall user experience.

Steps to Perform:

1. **Set Up a New Flutter Project:**
    - Create a new Flutter application using the command: flutter create form_app.
    - Navigate to the project directory: cd form_app.
2. **Design the Form Interface:**

In lib/main.dart, import the necessary packages:

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

    -

Define the main function and set the home to MyFormPage:

```
void main() {
  runApp(MaterialApp(
    home: MyFormPage(),
  ));
}
```

    -
    - Create a stateful widget MyFormPage that returns a Scaffold with an AppBar and a Form widget.
3. **Implement Form Fields with Validation:**
    - Within the Form widget, add TextFormField widgets for each input field (e.g., name, email, password).
    - Assign a TextEditingController to each field to manage the input.

Add validation logic using the validator property:

```
TextFormField(
  controller: _emailController,
  decoration: InputDecoration(labelText: 'Email'),
  validator: (value) {
```

```
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    }
    if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
      return 'Enter a valid email';
    }
    return null;
  },
),
```

4. **Add a Submit Button:**

Include an ElevatedButton that, when pressed, triggers the form's validation and processes the input if valid:

```
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      // Process data
    }
  },
  child: Text('Submit'),
),
```

5. **Manage Form State:**

Use a GlobalKey<FormState> to manage the form's state and validation:

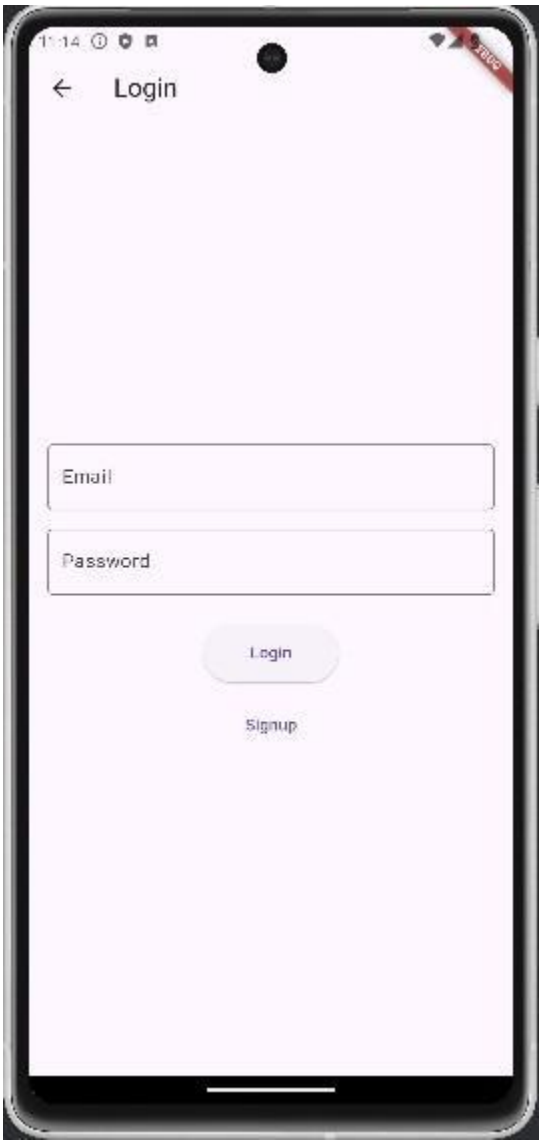
```
final _formKey = GlobalKey<FormState>();
```

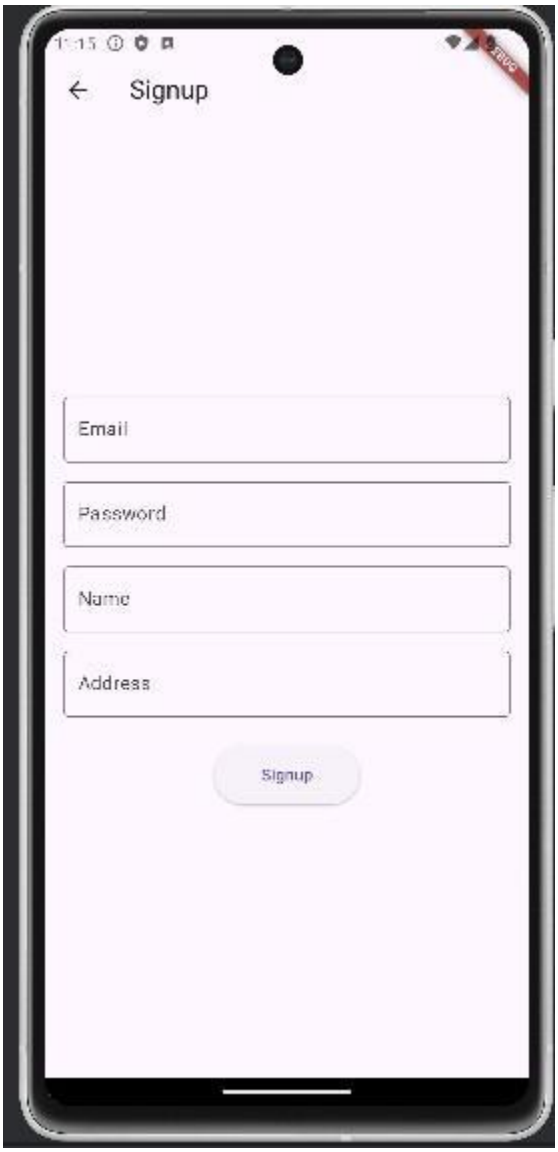
- Wrap the Form widget with a Form widget and assign the _formKey to it.

Key Features:

- **State Management:** Utilizes GlobalKey to manage form state effectively.
- **Validation:** Provides real-time validation feedback to users.
- **User Experience:** Enhances user interaction with clear error messages and responsive input fields.

Output:






```
class LoginScreen extends StatefulWidget {
 @override
 _LoginScreenState createState() => _LoginScreenState();
}

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

 Future<void> _login() async {
   if (_formKey.currentState!.validate()) {
     try {
```

```
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _emailController.text,
        password: _passwordController.text,
      );
      Navigator.pushReplacement(
          context, MaterialPageRoute(builder: (context) => HomeScreen()));
    } on FirebaseAuthException catch (e) {
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Failed to login: ${e.message}')),
        );
      }
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Login')),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextFormField(
              controller: _emailController,
              decoration: InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
              ),
              validator: (value) => value!.isEmpty ? 'Enter email' : null,
            ),
            SizedBox(height: 16),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
              ),
```