

Name: Anshi Tiwari

Class: D15C

Roll No: 56

Experiment 04 : To create an interactive Form using form widget

Theory:

Form Widget: It groups multiple input fields like TextFormField and manages their state collectively. It helps in validating user input and preventing form submission if any field is invalid.

TextFormField: This widget is commonly used for text input. It supports validation, styling, and easy integration with controllers to read and manage user input.

Validation and Submission:

Validators can be added to check conditions like empty fields or incorrect formats (e.g., email validation).

The form can be submitted using a button, typically wrapped in an ElevatedButton or TextButton.

GlobalKey:

A GlobalKey<FormState> is used to identify the form and manage its state, including validation and resetting the form.

Code:

```
void main() {  
  runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      debugShowCheckedModeBanner: false,  
      title: 'Music Recommendation App',  
      theme: ThemeData.light().copyWith(  
        scaffoldBackgroundColor: Colors.white,  
        appBarTheme: const AppBarTheme(  

```

```

        backgroundColor: Colors.white,
        foregroundColor: Colors.black,
    ),
),
home: const LoginPage(),
);
}
}

```

```

class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

```

```

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

```

```

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  bool _isPasswordVisible = false; // To manage password visibility

```

```

  void _login() {
    String email = _emailController.text;
    String password = _passwordController.text;

```

```

    if (email == "test@gmail.com" && password == "password123") {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const MusicRecommendationPage()),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Invalid email or password")),
      );
    }
  }
}

```

```

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Login')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),

```

```

child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Image.network(
      'https://cdn-icons-png.flaticon.com/512/2907/2907253.png',
      height: 100,
      errorBuilder: (context, error, stackTrace) {
        return const Icon(Icons.error, size: 100, color: Colors.red);
      },
    ),
    const SizedBox(height: 20),
    TextField(
      controller: _emailController,
      decoration: const InputDecoration(
        labelText: "Email",
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(color: Colors.black54),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(color: Colors.black),
        ),
      ),
    ),
    const SizedBox(height: 10),
    TextField(
      controller: _passwordController,
      obscureText: !_isPasswordVisible,
      decoration: InputDecoration(
        labelText: "Password",
        enabledBorder: const OutlineInputBorder(
          borderSide: BorderSide(color: Colors.black54),
        ),
        focusedBorder: const OutlineInputBorder(
          borderSide: BorderSide(color: Colors.black),
        ),
      ),
      suffixIcon: IconButton(
        icon: Icon(
          _isPasswordVisible
            ? Icons.visibility
            : Icons.visibility_off,
          color: Colors.black54,
        ),
        onPressed: () {
          setState(() {

```

```

        _isPasswordVisible = !_isPasswordVisible;
    });
  },
),
),
),
const SizedBox(height: 20),
ElevatedButton(
  onPressed: _login,
  style: ElevatedButton.styleFrom(
    backgroundColor: const Color.fromARGB(255, 177, 209, 236),
  ),
  child: const Text("Login"),
),
const SizedBox(height: 20),
ElevatedButton(
  onPressed: _login,
  style: ElevatedButton.styleFrom(
    backgroundColor: const Color.fromARGB(255, 177, 209, 236),
  ),
  child: const Text("New User? Signup"),
),
],
),
),
);
}
}

```

Explanation:

Form Widget: The login page uses a Form to group the email and password fields, managing their state collectively. This makes it easier to validate all inputs before submission.

TextFormField:

Two TextFormField widgets are used for the Email and Password inputs.

They include properties like controller to capture user input and obscureText to hide the password for security.

Validation and Submission:

Basic validation is implemented to check if the entered email and password match predefined values.

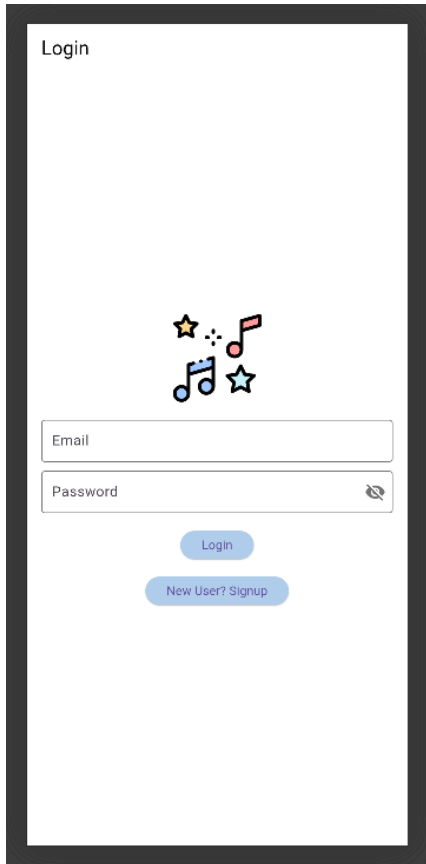
If the inputs are correct, the user is navigated to the next screen. If incorrect, an error message is displayed using SnackBar.

Interactive Elements:

A toggle for showing/hiding the password enhances user experience.

Buttons for Login and Signup trigger the form validation and submission process..

Output:



Conclusion:

This Flutter code builds a simple login screen for a music recommendation app. It verifies user credentials and navigates to the next page if they are correct. The design is clean, with input fields and password visibility toggle.