Input & Form Validation

- **TextField, Form,**
- **Button, dan**
- **Validator**.

---

1. Input dalam Flutter

TextField

`TextField` adalah widget Flutter yang digunakan untuk menerima input teks dari pengguna.

Fitur utama TextField:

- Mengetik teks
- Mengatur jenis keyboard (numeric, email, dll.)
- Menggunakan controller untuk mengambil nilai
- Menangani perubahan teks dengan `onChanged`
- Mengatur dekorasi (hint, label, border)

Contoh penggunaan dasar:

```
TextField(
  decoration: InputDecoration(
    labelText: 'Nama',
    border: OutlineInputBorder(),
  ),
)
```

TextEditingController

Untuk membaca nilai input, gunakan `TextEditingController`.

```
final TextEditingController nameController = TextEditingController();

TextField(
  controller: nameController,
)
```

Untuk mengambil nilai:

```
print(nameController.text);
```

---

2. Form dalam Flutter

`Form` adalah wadah yang digunakan untuk mengelompokkan beberapa input (misalnya beberapa TextFormField).
Form memiliki sebuah `GlobalKey` untuk melakukan validasi secara keseluruhan.

Struktur umum Form

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

Form(
  key: _formKey,
  child: Column(
    children: [
      TextFormField(),
      TextFormField(),
      ElevatedButton(
        onPressed: () {
          if (_formKey.currentState!.validate()) {
            // valid
          }
        },
        child: Text('Submit'),
      ),
    ],
  ),
)
```

👉 **Form sangat mendukung validasi**, terutama ketika menggunakan `TextFormField`.

---

3. Button (Tombol) dalam Flutter

Flutter menyediakan beberapa jenis tombol:

✔ **`ElevatedButton`**

Tombol dengan background yang menonjol

```
ElevatedButton(
  onPressed: () {},
  child: Text('Kirim'),
)
```

✔ **`TextButton`**

Tombol tanpa background

✔ **`OutlinedButton`**

Tombol dengan border

Tombol sering digunakan untuk **mengirim data** atau **menjalankan validasi form**.

---

4. Validator dalam Flutter

Validator digunakan untuk **memeriksa apakah input sudah benar**.

Validator biasanya digunakan pada **TextFormField**, bukan TextField.

TextFormField dengan validator

```
TextFormField(
  decoration: InputDecoration(labelText: 'Email'),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Email tidak boleh kosong';
    }
    if (!value.contains('@')) {
      return 'Email tidak valid';
    }
    return null; // valid
  },
)
```

Cara menjalankan validator:

```
if (_formKey.currentState!.validate()) {
  print("Semua input valid");
}
```

---

5. Contoh Form Lengkap dengan Validasi

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

class MyFormPage extends StatefulWidget {
  @override
  _MyFormPageState createState() => _MyFormPageState();
}

class _MyFormPageState extends State<MyFormPage> {
```

```
final _formKey = GlobalKey<FormState>();
final TextEditingController nameController = TextEditingController();
final TextEditingController emailController = TextEditingController();

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("Input & Validation")),
    body: Padding(
      padding: EdgeInsets.all(16),
      child: Form(
        key: _formKey,
        child: Column(
          children: [
            TextFormField(
              controller: nameController,
              decoration: InputDecoration(labelText: 'Nama'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Nama wajib diisi';
                }
                return null;
              },
            ),

            SizedBox(height: 16),

            TextFormField(
              controller: emailController,
              decoration: InputDecoration(labelText: 'Email'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Email wajib diisi';
                }
                if (!value.contains('@')) {
                  return 'Format email salah';
                }
                return null;
              },
            ),

            SizedBox(height: 24),

            ElevatedButton(
              child: Text('Submit'),
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  print('Nama: ${nameController.text}');
                  print('Email: ${emailController.text}');
                }
              },
            )
          ],
        ),
      ),
    ),
  );
```

```
  }
}
```

Ringkasan

Komponen	Fungsi
TextField	Input teks dasar (tidak support validator langsung)
TextFormField	Input teks yang mendukung validasi
Form	Wadah untuk mengelompokkan input dan menjalankan validasi
Button	Eksekusi aksi, sering digunakan untuk submit form
Validator	Fungsi untuk memeriksa apakah input valid

Contoh Aplikasi Mini (Login & Register)

Aplikasi sederhana ini memiliki:

- Halaman Login
- Halaman Register
- Validasi email (regex)
- Validasi password kuat
- Navigasi antar halaman

---

main.dart

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: LoginPage(),
      routes: {
        '/login': (_) => LoginPage(),
        '/register': (_) => RegisterPage(),
```

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

---

login_page.dart

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

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  final emailController = TextEditingController();
  final passwordController = TextEditingController();

  // Regex email
  final emailRegex = RegExp(r'^[^@]+@[^@]+\.[^@]+');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Login")),
      body: Padding(
        padding: EdgeInsets.all(16),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: emailController,
                decoration: InputDecoration(labelText: 'Email'),
                validator: (value) {
                  if (value!.isEmpty) return "Email wajib diisi";
                  if (!emailRegex.hasMatch(value)) return "Format email
salah";
                  return null;
                },
              ),
              SizedBox(height: 16),

              TextFormField(
                controller: passwordController,
                obscureText: true,
                decoration: InputDecoration(labelText: 'Password'),
                validator: (value) {
                  if (value!.isEmpty) return "Password wajib diisi";
                  if (value.length < 6) return "Minimal 6 karakter";
                  return null;
                },
              ),
```

```
              SizedBox(height: 24),

              ElevatedButton(
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text("Login Berhasil")),
                    );
                  }
                },
                child: Text("Login"),
              ),

              TextButton(
                onPressed: () {
                  Navigator.pushNamed(context, '/register');
                },
                child: Text("Belum punya akun? Register"),
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

---

register_page.dart

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

class RegisterPage extends StatefulWidget {
  @override
  _RegisterPageState createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final _formKey = GlobalKey<FormState>();
  final emailController = TextEditingController();
  final passController = TextEditingController();
  final confirmController = TextEditingController();

  // Regex email
  final emailRegex = RegExp(r'^[^@]+@[^@]+\.[^@]+');

  // Regex password kuat:
  // minimal 8 karakter, ada huruf besar, kecil, angka, simbol
  final strongPass = RegExp(r'^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&]).{8,}$');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(title: Text("Register")),
      body: Padding(
        padding: EdgeInsets.all(16),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: emailController,
                decoration: InputDecoration(labelText: 'Email'),
                validator: (value) {
                  if (value!.isEmpty) return "Email wajib diisi";
                  if (!emailRegex.hasMatch(value)) return "Email tidak
valid";
                  return null;
                },
              ),

              SizedBox(height: 16),

              TextFormField(
                controller: passController,
                obscureText: true,
                decoration: InputDecoration(labelText: 'Password'),
                validator: (value) {
                  if (value!.isEmpty) return "Password wajib diisi";
                  if (!strongPass.hasMatch(value)) {
                    return "Password harus mengandung huruf besar, kecil,
angka & simbol, min 8 karakter";
                  }
                  return null;
                },
              ),

              SizedBox(height: 16),

              TextFormField(
                controller: confirmController,
                obscureText: true,
                decoration: InputDecoration(labelText: 'Konfirmasi
Password'),
                validator: (value) {
                  if (value != passController.text) return "Password tidak
sama";
                  return null;
                },
              ),

              SizedBox(height: 24),

              ElevatedButton(
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text("Registrasi Berhasil")),
                    );
                  }
```

```
                },
                child: Text("Register"),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

---

2. Validasi Kompleks (Regex Email & Password Kuat)

✔ Regex Email

```
final emailRegex = RegExp(r'^[^@]+@[^@]+\.[^@]+');
```

Cek email dengan:

```
if (!emailRegex.hasMatch(value)) return "Email tidak valid";
```

Regex Password Kuat

Syarat:

- Minimal 8 karakter
- Ada huruf besar
- Ada huruf kecil
- Ada angka
- Ada simbol

```
final strongPass =
    RegExp(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&]).{8,}$');
```

Validasi:

```
if (!strongPass.hasMatch(value)) {
  return "Password harus mengandung huruf besar, kecil, angka & simbol";
}
```

---

3. Perbedaan TextField vs TextFormField

Fitur	TextField	TextFormField
Mendukung validator	❌ Tidak	✅ Ya
Digunakan dengan Form	❌ Tidak cocok	✅ Ideal
Event validasi otomatis	Tidak ada	Ada (validate, save)
Implementasi lebih mudah untuk input sederhana	Ya	Tidak selalu
Mendapatkan nilai via controller	Ya	Ya

Kesimpulan:

- **TextField** → untuk input biasa (chat, search, dll.)
- **TextFormField** → untuk input yang butuh *validator* dalam *Form*

---

4. Penjelasan Lengkap tentang FormState

`FormState` adalah object yang mengontrol keadaan Form.

Biasanya diakses melalui:

```
_formKey.currentState
```

Fungsi penting dalam FormState

1. validate()

Menjalankan semua validator di setiap `TextFormField`.

```
if (_formKey.currentState!.validate()) {
  print("Semua input valid!");
}
```

2. save()

Menjalankan `onSaved` dari setiap `TextFormField`.

```
_formKey.currentState!.save();
```

3. reset()

Mengembalikan semua input ke kondisi awal.

```
_formKey.currentState!.reset();
```

Cara FormState bekerja

1. Setiap `TextFormField` memiliki **validator**, **onChanged**, dan **onSaved**.
2. Saat `validate()` dipanggil:
    - Semua validator dijalankan
    - Jika ada 1 saja yang error → validasi gagal
3. Jika validasi lolos:
    - Aplikasi bisa lanjut (submit data)

Contoh real yang meliputi:

UI lebih modern (Material 3)

Firebase Auth (Login & Register sungguhan)

State management (Provider, Bloc, & Riverpod – kamu bisa pilih)

Clean Architecture untuk Form Validation

Semua disusun dengan sangat rapi sesuai standar industri.
Arsitektur → UI → Auth → State Management.

---

1. STRUKTUR CLEAN ARCHITECTURE

Berikut struktur folder recommended untuk aplikasi Flutter modern:

```
lib/
 ├── core/
 │    ├── validators/
 │    │     ├── email_validator.dart
 │    │     └── password_validator.dart
 │    ├── error/
 │    └── utils/
 ├── features/
 │    └── auth/
 │         ├── data/
 │         │    ├── auth_firebase_service.dart
 │         ├── domain/
 │         │    ├── entities/
 │         │    │    └── user_entity.dart
 │         │    ├── repositories/auth_repository.dart
 │         │    └── usecases/
 │         │         ├── login_usecase.dart
 │         │         └── register_usecase.dart
 │         ├── presentation/
 │         │    ├── provider/ (atau bloc/ atau riverpod/)
 │         │    ├── pages/
 │         │    │    ├── login_page.dart
 │         │    │    └── register_page.dart
 │         │    └── widgets/
 ├── app.dart
 └── main.dart
```

---

2. CLEAN VALIDATION (Domain Layer)

email_validator.dart

```
class EmailValidator {
  static bool isValid(String email) {
    final regex = RegExp(r'^[^@]+@[^@]+\.[^@]+');
    return regex.hasMatch(email);
  }
}
```

password_validator.dart

```
class PasswordValidator {
  static bool isStrong(String password) {
    final regex =
      RegExp(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&]).{8,}$');
    return regex.hasMatch(password);
```

```
  }
}
```

---

3. FIREBASE AUTH (Data Layer)

auth_firebase_service.dart

```
import 'package:firebase_auth/firebase_auth.dart';

class AuthFirebaseService {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  Future<User?> login(String email, String password) async {
    final result = await _auth.signInWithEmailAndPassword(
        email: email, password: password);
    return result.user;
  }

  Future<User?> register(String email, String password) async {
    final result = await _auth.createUserWithEmailAndPassword(
        email: email, password: password);
    return result.user;
  }

  Future<void> logout() => _auth.signOut();
}
```

---

4. USE CASE (Domain Layer)

login_usecase.dart

```
import '../repositories/auth_repository.dart';

class LoginUseCase {
  final AuthRepository repo;
  LoginUseCase(this.repo);

  Future call(String email, String password) {
    return repo.login(email, password);
  }
}
```

register_usecase.dart

```
class RegisterUseCase {
  final AuthRepository repo;
  RegisterUseCase(this.repo);
```

```
  Future call(String email, String password) {
    return repo.register(email, password);
  }
}
```

---

5. REPOSITORY ABSTRACTION

auth_repository.dart

```
abstract class AuthRepository {
  Future<dynamic> login(String email, String password);
  Future<dynamic> register(String email, String password);
}
```

---

6. REPOSITORY IMPLEMENTATION

auth_repository_impl.dart

```
import '../../data/auth_firebase_service.dart';
import '../repositories/auth_repository.dart';

class AuthRepositoryImpl implements AuthRepository {
  final AuthFirebaseService service;
  AuthRepositoryImpl(this.service);

  @override
  Future login(String email, String password) {
    return service.login(email, password);
  }

  @override
  Future register(String email, String password) {
    return service.register(email, password);
  }
}
```

---

7. UI MODERN (Material 3)

Aktifkan Material 3:

main.dart

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

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}
```

---

app.dart

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        useMaterial3: true,
        colorSchemeSeed: Colors.deepPurple,
      ),
      home: LoginPage(),
    );
  }
}
```

---

8. STATE MANAGEMENT (3 Pilihan)

Saya buatkan tiga versi: **Provider**, **Bloc**, dan **Riverpod**.
Kamu bisa pilih mana yang mau dipakai.

OPTION A – PROVIDER (paling sederhana)

auth_provider.dart

```
import 'package:flutter/material.dart';
import '../../../../core/validators/email_validator.dart';
import '../../../../core/validators/password_validator.dart';
import '../../../auth/domain/usecases/login_usecase.dart';
import '../../../auth/domain/usecases/register_usecase.dart';

class AuthProvider with ChangeNotifier {
  final LoginUseCase loginUseCase;
  final RegisterUseCase registerUseCase;
```

```
  bool isLoading = false;
  String? error;

  AuthProvider(this.loginUseCase, this.registerUseCase);

  Future<bool> login(String email, String password) async {
    if (!EmailValidator.isValid(email)) {
      error = "Email tidak valid";
      notifyListeners();
      return false;
    }

    isLoading = true;
    notifyListeners();

    try {
      await loginUseCase(email, password);
      return true;
    } catch (e) {
      error = e.toString();
      return false;
    } finally {
      isLoading = false;
      notifyListeners();
    }
  }

  Future<bool> register(String email, String password) async {
    if (!PasswordValidator.isStrong(password)) {
      error = "Password terlalu lemah";
      notifyListeners();
      return false;
    }

    isLoading = true;
    notifyListeners();

    try {
      await registerUseCase(email, password);
      return true;
    } catch (e) {
      error = e.toString();
      return false;
    } finally {
      isLoading = false;
      notifyListeners();
    }
  }
}
```

OPTION B – BLoC (skala besar)

Jika kamu mau, saya bisa buatkan folder:

```
presentation/bloc/
  ├── auth_bloc.dart
  ├── auth_event.dart
  └── auth_state.dart
```

Untuk menjaga jawaban tetap fokus, saya siap buatkan lengkap bila kamu pilih “BLoC”.

OPTION C – Riverpod (paling modern)

Saya juga bisa siapkan versi **Riverpod + AsyncValue** yang clean.

Contoh provider:

```
final authProvider = StateNotifierProvider<AuthController, AsyncValue<void>>(
  (ref) => AuthController(ref),
);
```

Jika kamu ingin Riverpod lengkap: tinggal bilang.

---

9. LOGIN PAGE (Material 3 + Provider)

login_page.dart

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

class LoginPage extends StatelessWidget {
  LoginPage({super.key});

  final emailCtrl = TextEditingController();
  final passCtrl = TextEditingController();

  @override
  Widget build(BuildContext context) {
    final auth = context.watch<AuthProvider>();

    return Scaffold(
      appBar: AppBar(title: Text("Login")),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          children: [
```

```
            TextField(
              controller: emailCtrl,
              decoration: InputDecoration(
                labelText: "Email",
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 16),
            TextField(
              controller: passCtrl,
              obscureText: true,
              decoration: InputDecoration(
                labelText: "Password",
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 24),

            auth.isLoading
                ? CircularProgressIndicator()
                : FilledButton(
                    onPressed: () async {
                      bool success = await auth.login(
                        emailCtrl.text.trim(),
                        passCtrl.text.trim(),
                      );

                      if (success) {
                        ScaffoldMessenger.of(context).showSnackBar(
                          SnackBar(content: Text("Login berhasil")),
                        );
                      } else {
                        ScaffoldMessenger.of(context).showSnackBar(
                          SnackBar(content: Text(auth.error ?? "Error")),
                        );
                      }
                    },
                    child: Text("Login"),
                  ),

            TextButton(
              onPressed: () {},
              child: Text("Belum punya akun? Register"),
            ),
          ],
        ),
      ),
    );
  }
}
```

10. REGISTER PAGE (Material 3 + Validation)

Mirip login page, tinggal mengganti fungsi register.