FAKULTAS TEKNOLOGI DAN INFORMATIKA
Bright Future with FTI, be a Young Entrepreneur with UNIBI.

PROGRAM STUDI INFORMATIKA
UNIVERSITAS INFORMATIKA DAN BISNIS INDONESIA

PRODI SISTEM INFORMASI
FAKULTAS TEKNOLOGI DAN INFORMATIKA

FutureProvider, Adaptive dan Responsive

Arif Bakti Nugraha, S.T., M.Kom.

FutureProvider

Tujuan Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan mampu :

1. Anda mengerti penggunaan FutureProvider dalam Flutter.
2. Anda memahami cara membuat halaman login dengan parameter username dan password yang dikirim ke server API.
3. Anda dapat menyimpan informasi user dalam Riverpod Provider sehingg a bisa diakses di setiap halaman.
4. Anda tahu cara mengambil informasi user di halaman lain setelah login.

Riverpod Singkat

- Riverpod adalah state management yang dikembangkan sebagai penerus Provider dengan perbaikan.
- Dengan Riverpod, state management jadi lebih aman, testable, dan modular.
- Provider bisa digunakan untuk berbagai tipe data termasuk state biasa, async data, dll.

FutureProvider di Riverpod

- FutureProvider adalah salah satu cara untuk mengelola data asynchronous di Flutter menggunakan Riverpod.
- Ini memungkinkan kita mengambil data dari API atau operasi async lainnya dan menyediakan data tersebut untuk UI.
- Cara kerja: FuturePr ovider menjalankan sebuah fungsi async dan memberikan status data (data, error) yang bisa dikonsumsi widget.

Halaman Login dengan Parameter Username & Password

- Halaman login berfungsi menerima input dari user berupa username dan password.
- Data tersebut dikirim via HTTP POST ke server API.
- Server akan merespon dengan status login berhasil atau gagal, dan data user jika berhasil.

Latihan 1

1. Buat projek Flutter dengan nama `latihanmandiri14`
2. Instal paket `http` dan `flutter_riverpod` dengan mengetikkan perintah berikut ini

```
flutter pub add http
flutter pub add flutter_riverpod
```

3. Buat file `login_model.dart` di direktori `/lib`. File ini akan digunakan untuk membuat model `user` yang merupakan data user yang akan login pada aplikasi. Di dalam file ini juga akan dibuat model respon login. Isi file tersebut dengan model keduanya.

```
//Model untuk response token
class LoginResponse {
  final String token;
  LoginResponse({required this.token});
  factory LoginResponse.fromJson(Map<String, dynamic> json) {
    return LoginResponse(token: json['token']);
  }
}
```

Latihan 1

```
class User {
  final int id;
  final String email;
  final String name;

  User({required this.id, required this.email, required this.name});
  factory User.fromJson(Map<String, dynamic> json) {
    return User(id: json['id'], email: json['email'], name: json['name']);
  }
}
```

4. Buat file dengan nama `api_service.dart` di direktori `/lib` File ini akan digunakan untuk menyimpan `loginProvider` yang berupa `FutureProvider` untuk mengakses data `user` yang login dan data daftar user.

```
import 'dart:convert';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_riverpod/legacy.dart';
import 'package:http/http.dart' as http;
```

Latihan 1

```
final usernameProvider = StateProvider<String>((ref) => '');
final passwordProvider = StateProvider<String>((ref) => '');
final tokenProvider = StateProvider<String>((ref) => '');

class LoginResponse {
  final String token;
  LoginResponse({required this.token});
  factory LoginResponse.fromJson(Map<String, dynamic> json) {
    return LoginResponse(token: (json['token'] ?? '').toString());
  }
}
final loginProvider = FutureProvider.autoDispose
    .family<LoginResponse, Map<String, String>>((ref, credentials) async {
      final username = credentials['username'] ?? '';
      final password = credentials['password'] ?? '';
      const apiKey = 'YOUR_REQRES_API_KEY_HERE';
      final response = await http.post(
        Uri.parse('https://reqres.in/api/login'),
```

Latihan 1

```
 headers: {'Content-Type': 'application/json', 'x-api-key': apiKey},
          body: jsonEncode({'email': username, 'password': password}),
        );
        if (response.statusCode == 200) {
          return LoginResponse.fromJson(jsonDecode(response.body));
        }
        try {
          final Map<String, dynamic> err = jsonDecode(response.body);
          throw Exception((err['error'] ?? 'Login gagal').toString());
        } catch (_) {
          throw Exception('Login gagal (HTTP ${response.statusCode})');
        }
      });
class User {
  final int id;
  final String name;
  final String email;
  User({required this.id, required this.name, required this.email});
```

Latihan 1

```
 factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: (json['id'] ?? 0) as int,
      name: (json['name'] ?? '').toString(),
      email: (json['email'] ?? '').toString(),
    );
  }
}
final userProvider = StateProvider<List<User>>((ref) => []);
Future<List<User>> fetchUsers() async {
  final response = await http.get(
    Uri.parse('https://jsonplaceholder.typicode.com/users'),
  );
  if (response.statusCode == 200) {
    final List<dynamic> body = jsonDecode(response.body);
    return body.map((item) => User.fromJson(item)).toList();
  } else {
    throw Exception('Failed to load users (HTTP ${response.statusCode})');
```

Latihan 1

```
  }
}
```

5. Buat halaman `login_page.dart` yang akan digunakan untuk menampilkan form login. Isi dengan kode berikut.

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:latihanmandiri14/api_service.dart';
import 'package:latihanmandiri14/tab_page.dart';

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

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final username = ref.watch(usernameProvider);
    final password = ref.watch(passwordProvider);

    return Scaffold(
```

Latihan 1

```
 appBar: AppBar(title: const Text('Login Flutter Riverpod')),
      body: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          children: [
            TextField(
              decoration:   const   InputDecoration(labelText:   'Username
(email)'),
              onChanged: (value) =>
                  ref.read(usernameProvider.notifier).state = value,
            ),
            TextField(
              decoration: const InputDecoration(labelText: 'Password'),
              obscureText: true,
              onChanged: (value) =>
                  ref.read(passwordProvider.notifier).state = value,
            ),
            const SizedBox(height: 20),
            ElevatedButton(
```

Latihan 1

```
onPressed: () async {
                try {
                  // Pakai akun contoh Reqres yang valid:
                  // eve.holt@reqres.in / cityslicka
                  final loginResponse = await ref.read(
                    loginProvider({
                      'username': username.isEmpty
                          ? 'eve.holt@reqres.in'
                          : username,
                      'password':   password.isEmpty   ?   'cityslicka'   :
password,
                    }).future,
                  );

                  ref.read(tokenProvider.notifier).state                    =
loginResponse.token;

                  if (!context.mounted) return;
                  Navigator.pushReplacement(
```

Latihan 1

```
 context,
                    MaterialPageRoute(builder:    (context)    =>    const
TabPage()),
                );
              } catch (e) {
                if (!context.mounted) return;
                showDialog(
                  context: context,
                  builder: (context) => AlertDialog(
                    title: const Text('Login Gagal'),
                    content: Text(e.toString()),
                    actions: [
                      TextButton(
                        onPressed: () => Navigator.pop(context),
                        child: const Text('OK'),
                      ),
                    ],
                  ),
                );
```

Latihan 1

```
        }
                    },
                    child: const Text('Login'),
                  ),
                ],
              ),
            ),
          );
        }
      }
```

6. Buat halaman `tab_page.dart` yang akan digunakan untuk menyimpan halaman dengan dengan widget Scaffold dan dan didalamnya terdapat widget AppBar BottomNavigationBar. Adapun isi dari parameter **body** akan dibuat kemudian.

```
import 'package:flutter_riverpod/legacy.dart';
import 'package:latihanmandiri14/api_service.dart';
import 'package:latihanmandiri14/home_page.dart';
import 'package:latihanmandiri14/profile_page.dart';
import 'package:flutter/material.dart';
```

Latihan 1

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

final bottomNavIndexProvider = StateProvider<int>((ref) => 0);
class TabPage extends ConsumerWidget {
  const TabPage({super.key});

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final username = ref.watch(usernameProvider);
    final currentIndex = ref.watch(bottomNavIndexProvider);
    final pages = [
      HomePage(),
      ProfilPage(), // sesuaikan dengan nama class kamu
    ];
    final clean = username.trim();
    final initials = clean.isEmpty
        ? '?'
        : clean.substring(0, clean.length >= 2 ? 2 : 1).toUpperCase();
```

Latihan 1

```
return Scaffold(
  appBar: AppBar(
    leading: Padding(
      padding: const EdgeInsets.all(4.0),
      child: CircleAvatar(
        backgroundColor: const Color.fromARGB(255, 8, 128, 139),
        child: Text(
          initials,
          style: const TextStyle(color: Colors.white, fontSize: 24),
        ),
      ),
    ),
    title: const Text("Contoh App"),
    backgroundColor: const Color.fromARGB(154, 255, 118, 64),
  ),
  body: pages[currentIndex],
  bottomNavigationBar: BottomNavigationBar(
    currentIndex: currentIndex,
```

Latihan 1

```
 onTap: (value) {
          ref.read(bottomNavIndexProvider.notifier).state = value;
        },
        items: const [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: "Rumah"),
          BottomNavigationBarItem(icon:    Icon(Icons.person_2),    label:
"Profil"),
        ],
      ),
    );
  }
}
```

7. Buat halaman `home_page` yang akan digunakan untuk mengisi body dari halaman `tabe_page dart`. Dan isi dengan kode berikut ini.

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:latihanmandiri14/api_service.dart';
```

Latihan 1

```
class HomePage extends ConsumerStatefulWidget {
  const HomePage({super.key});
  @override
  ConsumerState<HomePage> createState() => _HomePageState();
}
class _HomePageState extends ConsumerState<HomePage> {
  bool _loading = false;
  @override
  void initState() {
    super.initState();
    _fetchData();
  }
  Future<void> _fetchData() async {
    setState(() => _loading = true);
    try {
      final users = await fetchUsers();
      ref.read(userProvider.notifier).state = users;
    } catch (e) {
      if (!mounted) return;
```

Latihan 1

```
ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Gagal memuat data: ${e.toString()}')),
      );
    } finally {
      if (mounted) setState(() => _loading = false);
    }
  }
  @override
  Widget build(BuildContext context) {
    final users = ref.watch(userProvider);
    if (_loading && users.isEmpty) {
      return const Center(child: CircularProgressIndicator());
    }
    return RefreshIndicator(
      onRefresh: _fetchData,
      child: ListView.builder(
        itemCount: users.length + 1,
        itemBuilder: (context, index) {
          if (index == 0) {
```

Latihan 1

```
return const Padding(
                padding: EdgeInsets.all(8.0),
                child: Text("Daftar User", style: TextStyle(fontSize: 24)),
              );
            }
            final user = users[index - 1];
            return Card(
              color: const Color.fromARGB(255, 177, 239, 240),
              child: ListTile(
                title: Text(user.name),
                subtitle: Text(user.email),
                leading: CircleAvatar(child: Text(user.id.toString())),
              ),
            );
          },
        ),
      );
    }
  }
```

Latihan 1

8. Buat file `profile_page.dart` yang digunakan untuk menampilkan data pengguna yang login. Dan isi dengan kode berikut.

```
import 'package:latihanmandiri14/api_service.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class ProfilPage extends ConsumerWidget {
  const ProfilPage();
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final username = ref.watch(usernameProvider);
    final password = ref.watch(passwordProvider);
    final token = ref.watch(tokenProvider);
    return Padding(
      padding: EdgeInsets.all(20),
      child: ListView(
        children: [
          Card(
```

Latihan 1

```
color: const Color.fromARGB(255, 177, 239, 240),
          child: ListTile(
            title: Text(
              'Username: $username',
              style: TextStyle(fontSize: 18),
            ),
          ),
        ),
        Card(
          color: const Color.fromARGB(255, 177, 239, 240),
          child: ListTile(
            title: Text(
              'Password: $password',
              style: TextStyle(fontSize: 18),
            ),
          ),
        ),
        Card(
```

Latihan 1

```
 color: const Color.fromARGB(255, 177, 239, 240),
              child: ListTile(
                title: Text(
                  'Token: $token',
                  style:       TextStyle(fontSize:       18,       fontWeight:
FontWeight.bold),
                ),
              ),
            ),
          ],
        ),
      );
    }
}
```

9. Ubah isi file `main.dart` dengan kode berikut untuk menyatukan semua kode.

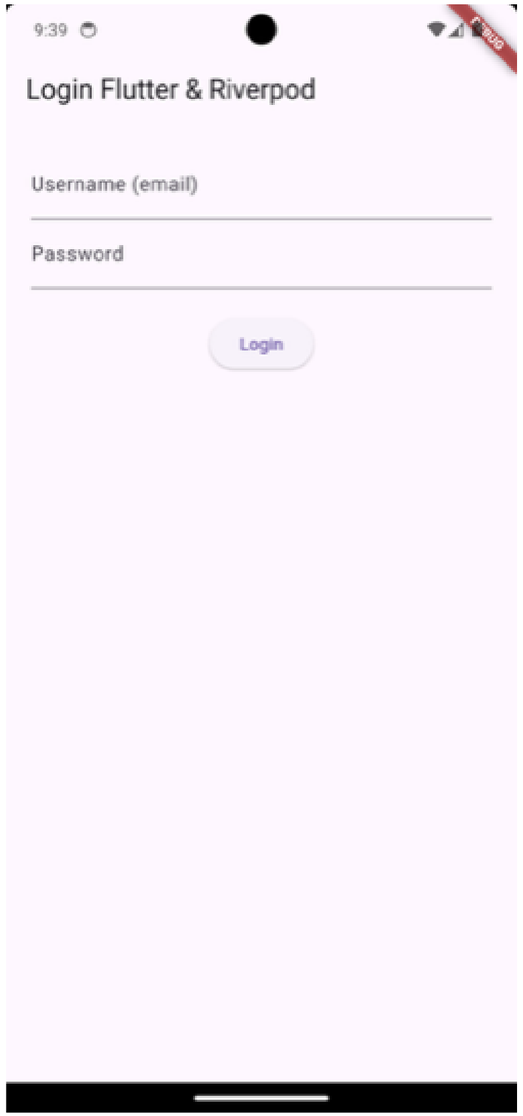
```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:latihanmandiri14/login_page.dart';
```

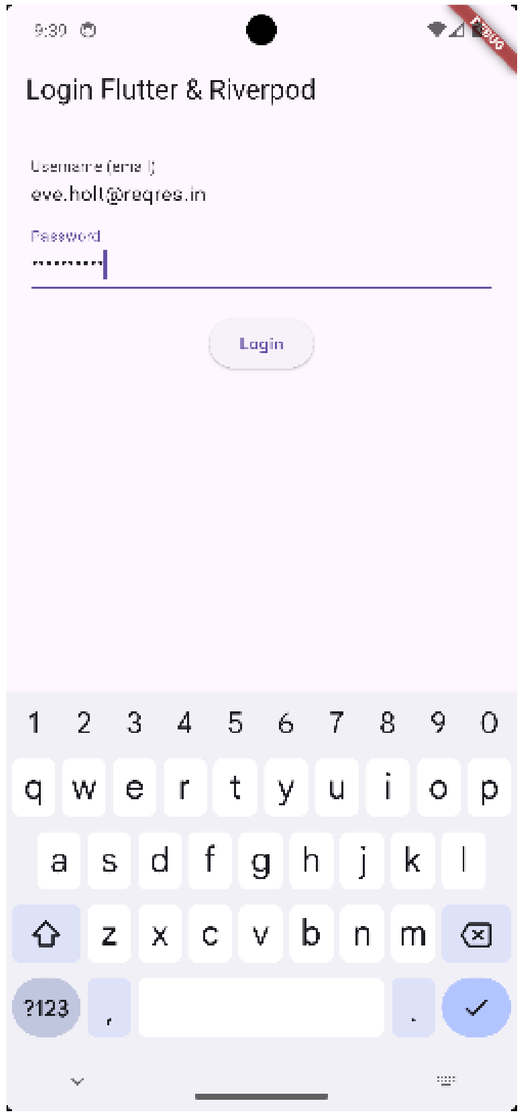
Latihan 1

```
void main() {
  runApp(const ProviderScope(child: MyApp()));
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(home: LoginPage());
  }
}
```

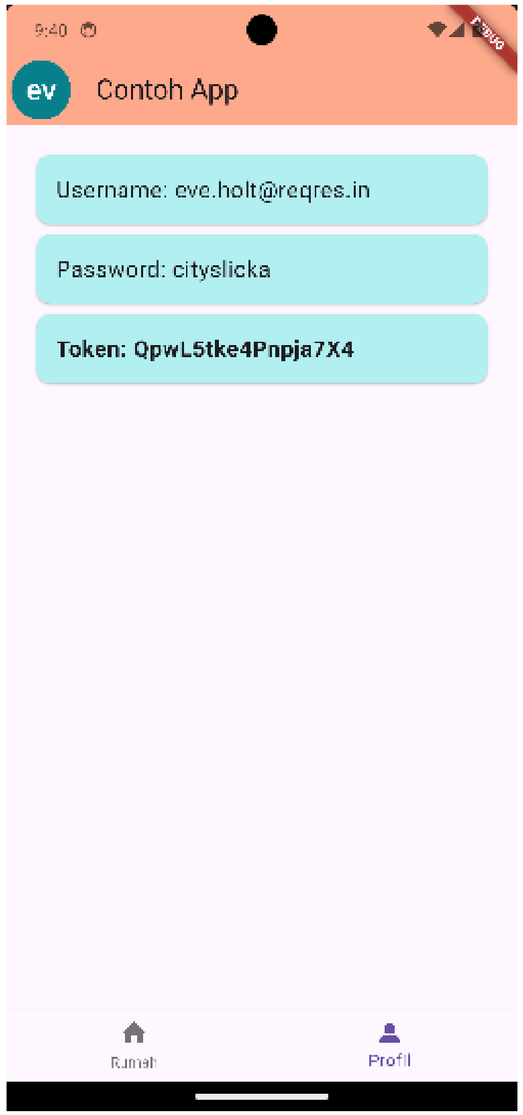
10. Hubungkan proyek ke emulator dan jalankan proyek tersebut. Jika tidak ada kesalahan, akan diperoleh gambar seperti pada Latihan Mandiri 1 4 ini.

Latihan 1


9:39
Login Flutter & Riverpod
Username (email)
Password
Login


9:39
Login Flutter & Riverpod
eve.holt@reqres.in
Password
Login


9:40
ev
Contoh App
Daftar User
1
Leanne Graham
Sincere@april.biz
2
Ervin Howell
Shanna@melissa.tv
3
Clementine Bauch
Nathan@yesenia.net
4
Patricia Lebsack
Julianne.OConner@kory.org
5
Chelsey Dietrich
Lucio_Hettinger@annie.ca
6
Mrs. Dennis Schulist
Karley_Dach@jasper.info
7
Kurtis Weissnat
Telly.Hoeger@billy.biz
8
Nicholas Runolfsdottir V
Sherwood@rosamond.me
Rumah
Profil


9:40
ev
Contoh App
Username: eve.holt@reqres.in
Password: cityslicka
Token: QpwL5tke4Pnpja7X4
Rumah
Profil

Adaptive dan Responsive

Tujuan Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan mampu :

1. Memahami konsep dasar Adaptive dan Responsive dalam pengembangan aplikasi mobile.
2. Membuat aplikasi Flutter yang dapat tampil optimal pada berbagai ukuran layar menggunakan prinsip responsive.
3. Membuat aplikasi Flutter yang dapat menyesuaikan tampilan berdasarkan jenis perangkat atau kondisi tertentu menggunakan prinsip adaptive.
4. Mengimplementasikan widget seperti Container dan GridView agar menyesuaikan ukuran layar dan orientasi perangkat.

Pengertian Responsive Design

Responsive design adalah pendekatan dalam pengembangan aplikasi yang memungkinkan UI menyesuaikan tata letak dan ukuran elemen secara dinamis berdasarkan ukuran layar perangkat. Tujuannya agar tampilan tetap optimal dan nyaman dilihat baik di layar kecil (smartphone) maupun layar besar (tablet, desktop).

Pengertian Adaptive Design

Adaptive design adalah pendekatan d imana aplikasi menyediakan beberapa layout tetap (preset) dan memilih yang paling sesuai tergantung jenis perangkat, ukuran layar, atau kondisi lain. Jadi, bukan hanya scaling elemen UI tapi menyesuaikan struktur tampilan secara spesifik.

Perbedaan Utama Adaptive dan Responsive

Aspek	Responsive	Adaptive
Penyesuaian Layout	Dinamis, mengikuti ukuran layar	Berdasarkan preset layout tertentu
Fleksibilitas	High, elemen UI skala dan posisinya menyesuaikan	Medium, pilih layout berbeda yang sudah didesain
Kompleksitas Implementasi	Lebih sederhana dengan layout fleksibel	Lebih kompleks karena harus mendesain beberapa layout
Contoh	UI berbasis MediaQuery, Flexible, Expanded	UI dengan layout berbeda untuk mobile dan tablet

Menerapkan Responsive Design di Flutter

Menggunakan MediaQuery

MediaQuery menyediakan informasi ukuran layar dan orientasi perangkat sehingga UI dapat menyesuaikan ukuran widget atau tata letak.

```
final screenWidth = MediaQuery.of(context).size.width;
final screenHeight = MediaQuery.of(context).size.height;
```

Widget Responsif

- Flexible Expanded: Membuat widget fleksibel dan mengisi ruang sesuai proporsi tertentu.
- LayoutBuilder: Memberikan constraints ukuran kepada widget agar dapat membuat keputusan layout berdasarkan ukuran yang tersedia.
- AspectRatio: Menjaga rasio aspek widget tetap konsisten.

Contoh

```
Container(
   width: MediaQuery.of(context).size.width*0.8, //80% dari lebar layar
   height: 200, color: Colors.blue,
);
```

Menerapkan Adaptive Design di Flutter

Deteksi Jenis Perangkat

Flutter bukan hanya untuk mobile; aplikasi dapat berjalan di banyak platform. Dengan mengenali ukuran atau platform, aplikasi bisa memilih layout tertentu.

```
Widget build(BuildContext context){
   final screenWidth = MediaQuery.of(context).size.width;

   if(screenWidth < 600){
       return mobileLayout();
   }else{
       return tabletLayout();
   }
}
```

Adaptive dengan Widget Responsif Khusus

Misalnya, menampilkan daftar dalam bentuk kolom tunggal di smartphone, tetapi menggunakan grid dua kolom di tablet.

Konsep Tema (Theme) dalam Aplikasi Flutter

Pengenalan Tema di Flutter

Tema dalam Flutter adalah sekumpulan aturan styilng (gaya visual) yang mengontrol tampilan seluruh aplikasi, seperti warna latar, warna teks, warna tombol, dan elemen visual lainnya. Dengan menggunakan tema, pengembang dapat membuat tampilan aplikasi yang konsisten dan estetis.

Manfaat Penggunaan Tema

- Konsistensi UI Warna, font, dan elemen UI lain seragam di seluruh aplikasi.
- Mudah Pemeliharaan: Mengubah gaya global cukup dengan mengubah tema saja.
- Dukungan Tema Gelap/Terang: Mempermudah pembuatan mode gelap dan terang.

Membuat Tema Terang dan Tema Gelap dengan ThemeData dan ColorScheme

ThemeData

ThemeData adalah class di Flutter yang menyimpan data konfigurasi tema seperti warna utama (warna latar, warna teks, gaya teks, ikon, dan sebagainya.

ColorScheme

ColorScheme adalah cara yang lebih modern dan terstruktur untuk menentukan warna tema, dengan 13 warna standar seperti primary, secondary, background, surface, onPrimar y, dan lainnya yang membantu mempertahankan kontras dan keterbacaan.

Membuat Tema Terang

Tema terang menggunakan ThemeData.light() yang dapat dikustomisasi dengan ColorScheme untuk mengganti warna dasar.

Membuat Tema Gelap

Tema gelap menggunakan ThemeData. dark() yang juga dapat disesuaikan ColorScheme dengan warna yang lebih gelap dan nyaman dilihat di tempat gelap.

Membuat Tema Terang dan Tema Gelap dengan ThemeData dan ColorScheme

Contoh:

```
final ThemeData lightTheme = ThemeData(
   colorScheme: ColorScheme.light(
       primary: Colors.blue,
       secondary: Colors.blueAccent,
   ),
);
final ThemeData darkTheme = ThemeData(
   colorScheme: ColorScheme.dark(
       primary: Colors.blueGrey,
       secondary: Colors.lightBlueAccent,
   ),
);
```

Mengatur Perpindahan Tema Otomatis Berdasarkan Preferensi Sistem

Tema Otomatis Sesuai Sistem

Flutter dapat mendeteksi mode gelap atau terang sistem operasi (Android,iOS) menggunakan MediaQuery atau lebih umum melalui property themeMode di MaterialApp.

Pengaturan themeMode

- ThemeMode.light: Selalu theme terang.
- ThemeMode.dark: Selalu theme gelap.
- ThemeMode.system: Otomatis ikut pengaturan sistem.

Contoh Penentuan MateriApp :

```
MaterialApp(
   theme: lightTheme,
   darkTheme: darkTheme,
   themeMode: ThemeMode.system, //mengikuti pengaturan sistem
   home: MyHomePage(),
);
```

Mengatur Perpindahan Tema Otomatis Berdasarkan Preferensi Sistem

Personalisasi Warna Tema menggunakan ColorScheme

Kenapa Menggunakan ColorScheme?

ColorScheme memudahkan pengelolaan warna pada berbagai bagian UI seperti warna latar, teks, border, button, dan masih banyak lagi dengan struktur warna yang lengkap.

Cara Implementasi Personalisasi Warna

- Buat instance ColorScheme.light() atau ColorScheme.dark() dengan warna yang diinginkan.
- Gunakan sebagai parameter di ThemeData.colorScheme.
- Gunakan warna dari ColorScheme dalam widget dengan mengambil dari Theme.of(context).colorScheme.

Contoh Penggunaan Warna ColorScheme di Widget

```
Container(
   color: Theme.of(context).colorScheme.primary,
   child: Text(
       'Contoh teks’,
       style: TextStyle(color: Theme.of(context).colorScheme.onPrimary),
   ),
);
```

Mengatur Perpindahan Tema Otomatis Berdasarkan Preferensi Sistem

Penjelasan Istilah dalam ColorScheme

- **Primary**

  Warna utama aplikasi Anda, biasanya digunakan pada elemen elemen yang paling menonjol seperti AppBar, tombol utama, dan elemen interaktif penting.

- **OnPrimary**

  Warna yang digunakan untuk konten yang diletakkan di atas warna primary, misalnya teks atau ikon di atas AppBar berwarna primary. Warna ini harus memiliki kontras yang cukup agar terbaca jelas.

- **Secondary**

  Warna pendukung yang digunakan pada elemen elemen yang tidak terlalu penting seperti tombol sekunder, aksen, atau fitur tambahan.

- **OnSecondary**

Warna untuk konten di atas elemen dengan warna secondary, mirip konsep onPrimary, untuk memastikan kontras dan keterbacaan.

- **Background**

  Warna latar belakang utama aplikasi, biasanya latar dari area konten.

Mengatur Perpindahan Tema Otomatis Berdasarkan Preferensi Sistem

Penjelasan Istilah dalam ColorScheme

- **OnBackground**
  Warna yang digunakan untuk konten yang berada di atas latar background, seperti teks dan ikon utama dalam konten aplikasi.
- **Surface**
  Warna untuk permukaan elemen yang mengambang di atas background, seperti kartu, dialog, dan bottom sheets.
- **OnSurface**
  Warna untuk konten di atas surface, misalnya teks dan ikon di dalam kartu atau dialog.
- **Error**
  Warna yang digu nakan untuk menunjukkan status error atau kesalahan.
- **OnError**
  Warna untuk konten yang muncul di atas warna error, memastikan teks error mudah dibaca.

Latihan 2

1. Buat proyek Flutter kosong dan beri `nama latihanmandiri14b`
2. Buat file `responsive_home_page.dart` yang akan diisi dengan widget yang berubah komposisi ketika ukuran layar berubah. Isi file tersebut dengan kode berikut.

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

class ResponsiveHomePage extends StatelessWidget {
  const ResponsiveHomePage({super.key});
  @override
  Widget build(BuildContext context) {
    var size = MediaQuery.of(context).size;
    double containerWidth = size.width * 0.8;
    //Tentukan jumlah kolom grid sesuai ukuran layar
    int crossAxisCount = 2;
    if (size.width > 600) {
      crossAxisCount = 4;
    } else if (size.width > 400) {
      crossAxisCount = 3;
    }
```

Latihan 2

```
      return Center(
        child: Column(
          children: [
            SizedBox(height: 20),
            Container(
              width: containerWidth,
              height: 150,
              color: Colors.blue,
              child: Center(
                child: Text(
                  'Responsive                    Container                    nLebar:
${containerWidth.toStringAsFixed(1)}',
                  style: TextStyle(color: Colors.white, fontSize: 18),
                  textAlign: TextAlign.center,
                ),
              ),
            ),
            SizedBox(height: 20),
```

Latihan 2

```
Expanded(
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: GridView.builder(
              itemCount: 12,
              gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: crossAxisCount,
                crossAxisSpacing: 8,
                mainAxisSpacing: 8,
              ),
              itemBuilder: (context, index) {
                return Container(
                  color: Colors.teal[200],
                  child: Center(
                    child: Text(
                      'Item ${index + 1}',
                      style: TextStyle(fontWeight: FontWeight.bold),
                    ),
```

Latihan 2

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

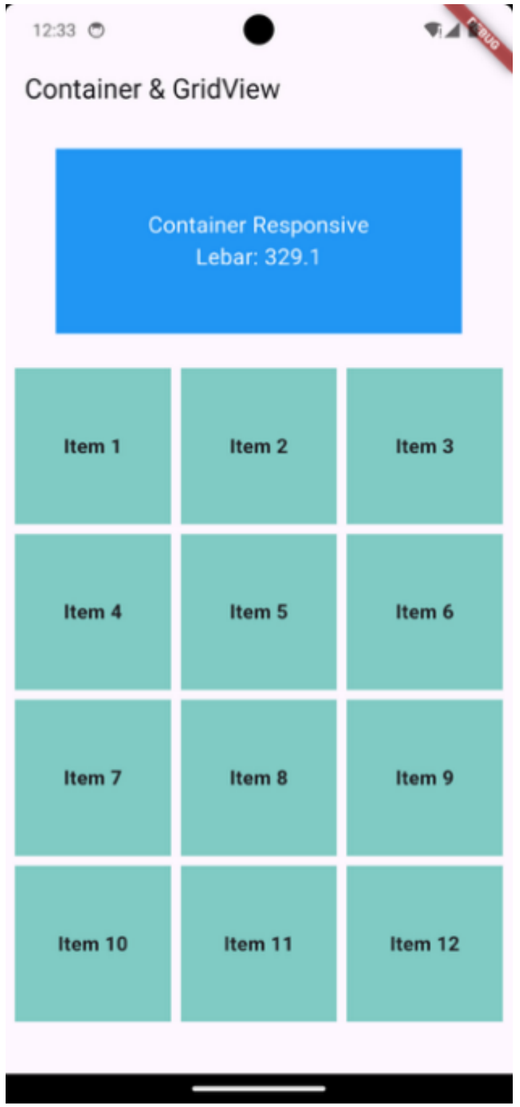
3. Ubah `main.dart` dengan kode berikut.

```
import 'package:flutter/material.dart';
import 'package:latihanmandiri14b/responsive_home_page.dart';

void main() {
  runApp(ResponsiveApp());
}
```

Latihan 2

```
class ResponsiveApp extends StatelessWidget {
  const ResponsiveApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Demo Responsive',
      home: Scaffold(
        appBar: AppBar(title: Text('Container GridView')),
        body: ResponsiveHomePage(),
      ),
    );
  }
}
```



Latihan 3

1. Buat proyek Flutter kosong dan beri naman `latihanmandiri14c`
2. Buat dokumen `adaptive_home_page.dart` untuk diisi dengan halaman yang akan menampilkan perubahan tata letak seiring orientasi emulator berubah atau laya yang akses berbeda. Isi dengan kode berikut

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

class AdaptiveHomePage extends StatelessWidget {
  const AdaptiveHomePage({super.key});
  @override
  Widget build(BuildContext context) {
    var size = MediaQuery.of(context).size;
    var orientation = MediaQuery.of(context).orientation;
    bool isLargeScreen = size.width > 600;
    Widget body;
    if (isLargeScreen && orientation == Orientation.portrait) {
      body = Container(
        color: Colors.blueAccent,
        child: Center(
```

Latihan 3

```
      child: Text(
                'Adaptive Layout: Large Screen Portrait',
                style: TextStyle(fontSize: 24, color: Colors.white),
                textAlign: TextAlign.center,
              ),
            ),
          );
        } else if (isLargeScreen && orientation == Orientation.landscape) {
          body = Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Flexible(
                child: Container(
                  color: Colors.green,
                  height: 300,
                  child: Center(
                    child: Text(
                      'Panel 1 (Large Landscape)',
```

Latihan 3

```
      style: TextStyle(fontSize: 20, color: Colors.white),
                      textAlign: TextAlign.center,
                    ),
                  ),
                ),
              ),
              SizedBox(width: 20),
              Flexible(
                child: Container(
                  color: Colors.teal,
                  height: 300,
                  child: Center(
                    child: Text(
                      'Panel 2 (Large Landscape)',
                      style: TextStyle(fontSize: 20, color: Colors.white),
                      textAlign: TextAlign.center,
                    ),
                  ),
                ),
```

Latihan 3

```
      ),
          ],
        );
      } else if (!isLargeScreen && orientation == Orientation.portrait) {
        body = Container(
          color: Colors.orange,
          child: Center(
            child: Text(
              'Adaptive Layout: Small Screen Portrait',
              style: TextStyle(fontSize: 18, color: Colors.white),
              textAlign: TextAlign.center,
            ),
          ),
        );
      } else {
        body = Container(
          color: Colors.red,
          child: Center(
            child: Text(
```

Latihan 3

```
      'Adaptive Layout: Small Screen Landscape',
                  style: TextStyle(fontSize: 16, color: Colors.white),
                  textAlign: TextAlign.center,
                ),
              ),
            );
          }
          return Scaffold(
            appBar: AppBar(title: Text('Adaptive Demo')),
            body: Center(child: body),
          );
        }
      }
```

4. Ubah `main.dart` sesuai dengan kode berikut ini.

```
import 'package:latihanmandiri14c/adaptive_home_page.dart';
import 'package:flutter/material.dart';
void main() {
  runApp(AdaptiveApp());
}
```

Latihan 3

```
class AdaptiveApp extends StatelessWidget {
  const AdaptiveApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(title: 'Adaptive Demo', home: AdaptiveHomePage());
  }
}
```

Latihan 4

1. Buatlah proyek Flutter dengan `latihanmandiri14d`
2. Buat file `my_home_page.dart` yang akan digunakan untuk menampilkan contoh contoh widget dengan warna sesuai tema dan peletakkan tombol untuk pengaturan tema.

```
import 'package:latihanmandiri14d/main.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class MyHomePage extends ConsumerWidget {
  const MyHomePage({super.key});
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final bool mode = ref.watch(isLightProvider);
    return Scaffold(
      appBar: AppBar(
        title: Text('Contoh Light / Dark Theme'),
        actions: [
```

Latihan 4

```
IconButton(
            onPressed: () {
              ref.read(isLightProvider.notifier).state = !mode;
            },
            icon: Icon(!mode ? Icons.light_mode : Icons.dark_mode),
          ),
        ],
      ),
      body: Center(
        child: Card(
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              'Teks ini berubah warna sesuai tema yang digunakan',
              style: Theme.of(context).textTheme.bodyLarge,
              textAlign: TextAlign.center,
            ),
          ),
```

Latihan 4

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

3. Ubah `main.dart` dengan kode berikut untuk mengatur tema.

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_riverpod/legacy.dart';
import 'package:projflutter14_d/my_home_page.dart';

final isLightProvider = StateProvider<bool>((ref) => true);
void main() {
  runApp(const ProviderScope(child: MyApp()));
}
class MyApp extends ConsumerWidget {
  const MyApp({super.key});
  ThemeData get lightTheme => ThemeData(
    textTheme: const TextTheme(
```

Latihan 3

```
    bodyLarge: TextStyle(fontSize: 20),
        bodySmall: TextStyle(
          fontSize: 10,
          color: Colors.deepPurple,
          fontWeight: FontWeight.normal,
        ),
      ),
      cardTheme:  const  CardThemeData(color:  Colors.amberAccent,  elevation:
  5),
      colorScheme: ColorScheme.light(
        primary: Colors.blue,
        secondary: Colors.green,
        surface: Colors.grey.shade100,
        onPrimary: Colors.white,
        onSecondary: Colors.white,
        onSurface: Colors.black,
        brightness: Brightness.light,
      ),
```

Latihan 3

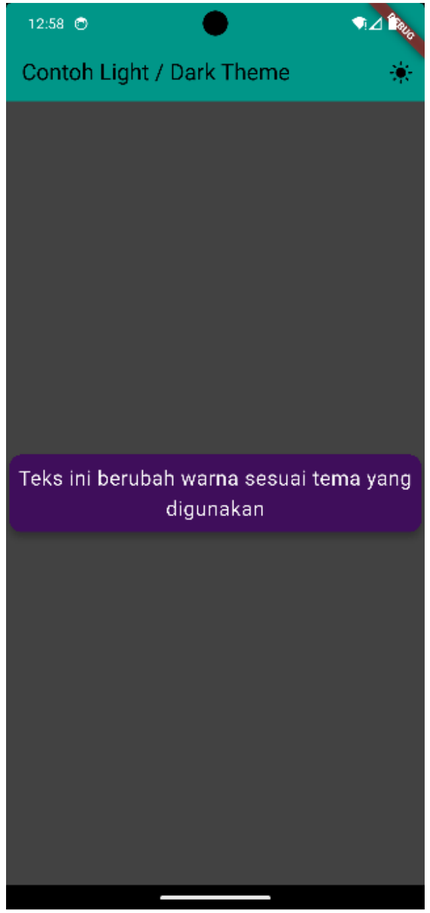
```
appBarTheme: const AppBarTheme(
      backgroundColor: Colors.blue,
      foregroundColor: Colors.white,
    ),
  );
  ThemeData get darkTheme => ThemeData(
    textTheme: const TextTheme(
      bodyLarge: TextStyle(fontSize: 20),
      bodySmall: TextStyle(
        fontSize: 10,
        color: Color.fromARGB(255, 216, 202, 241),
        fontWeight: FontWeight.normal,
      ),
    ),
    cardTheme: const CardThemeData(
      color: Color.fromARGB(255, 63, 14, 91),
      elevation: 5,
    ),
```

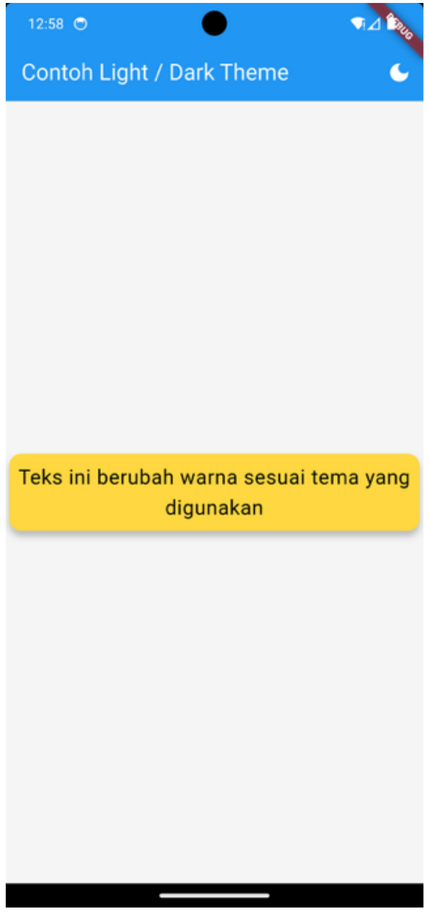
Latihan 3

```
    colorScheme: ColorScheme.dark(
          primary: Colors.teal,
          secondary: Colors.orange,
          surface: Colors.grey.shade800,
          onPrimary: Colors.black,
          onSecondary: Colors.black,
          onSurface: Colors.white,
          brightness: Brightness.dark,
        ),
        appBarTheme: const AppBarTheme(
          backgroundColor: Colors.teal,
          foregroundColor: Colors.black,
        ),
      );
      @override
      Widget build(BuildContext context, WidgetRef ref) {
        final isLight = ref.watch(isLightProvider);
        return MaterialApp(
          title: 'Flutter Light/Dark Theme Demo',
```

Latihan 3

```
    theme: lightTheme,
          darkTheme: darkTheme,
          themeMode: isLight ? ThemeMode.light : ThemeMode.dark,
          home: const MyHomePage(),
        );
    }
}
```





FAKULTAS TEKNOLOGI DAN INFORMATIKA
Bright Future with FTI, be a Young Entrepreneur with UNIBI.

PROGRAM STUDI INFORMATIKA
UNIVERSITAS INFORMATIKA DAN BISNIS INDONESIA

PRODI SISTEM INFORMASI
FAKULTAS TEKNOLOGI DAN INFORMATIKA

TERIMA KASIH