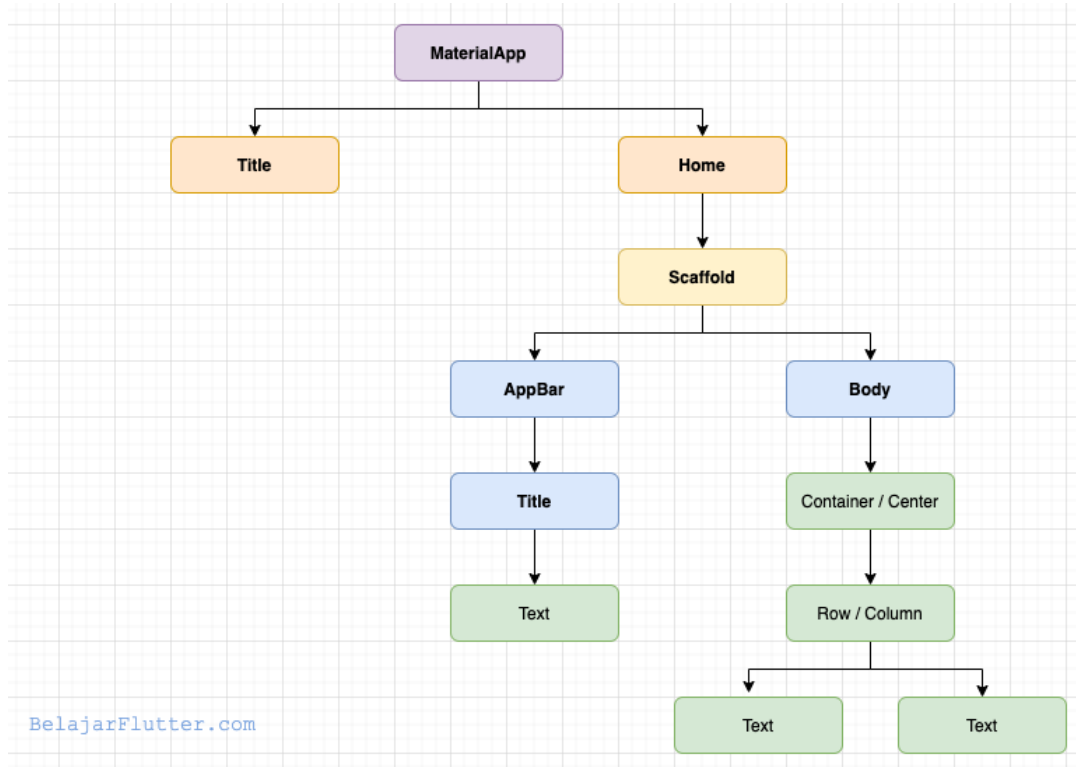


MODUL III

CORE COMPONENT, STYLE, NAVIGASI

Core Component



Properti pada Widget

Pada setiap widget memiliki property masing-masing, contohnya pada AppBar kita ingin membuat warna background menjadi warna lain. Untuk melakukan itu, maka AppBar dapat kita atur melalui properti.

```
class MyStatelessWidget extends StatelessWidget {  
  const MyStatelessWidget({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Belajar Flutter"),  
        backgroundColor: Colors.red  
      ),  
    );  
  }  
}
```

Dari contoh diatas dapat dipelajari bahwa ciri widget diawali dengan Huruf Kapital, digambar memiliki widget Scaffold, AppBar, dan Text. Pada Widget Scaffold memiliki property appBar dan pada Widget AppBar memiliki properti title, dan backgroundColor.

Widget dalam widget

Dalam membuat aplikasi kita akan banyak menggunakan widget, pada Flutter setiap widget umumnya memiliki properti **child** atau **children**, dari sini inilah kita bisa menggunakan widget didalamnya. Jika child hanya dapat menggunakan 1 widget sedangkan children dapat memiliki banyak widget.

Widget-widget general pada flutter

Ada banyak sekali widget pada flutter, pada kali ini kita akan membahas widget-widget penting untuk kita pahami saat pertama kali membuat aplikasi menggunakan flutter. Jika ingin membaca semua widget dapat membuka dokumentasi pada flutter:

<https://docs.flutter.dev/development/ui/widgets>

- **Text**

Text sangat penting karena paling sering digunakan dalam pengaplikasian. Berfungsi untuk menampilkan sebuah text biasa, jika ingin menambahkan style dapat menggunakan property style.

```
Text('Ini Text',  
  style: TextStyle(  
    color: Colors.blue,  
    backgroundColor: Colors.pink,  
    fontSize: 20.0,  
    fontStyle: FontStyle.italic,  
    fontWeight: FontWeight.bold  
  ),  
)
```

- **Container**

Container merupakan widget yang fungsinya untuk membungkus widget lain sehingga dapat diberikan margin, padding, warna, dan dekorasi.

```
Container(  
  padding: EdgeInsets.all(30.0,  
  margin: EdgeInsets.fromLTRB(30.0, 20.0, 0, 0),  
  decoration: BoxDecoration(  
    borderRadius: BorderRadius.circular(20.0),  
    color: Colors.blue,  
    child: Text("Hallo",  
      style: TextStyle(  
        fontSize: 20.0,  
        fontWeight: FontWeight.bold  
      ),  
    ),  
  ),  
)
```

- **Icon**

Widget ini dapat menggunakan icon yang telah disediakan, untuk mengetahui icon-icon yang disediakan dapat membuka dokumentasinya.

<https://api.flutter.dev/flutter/material/Icons-class.html>

```
Container(  
  padding: EdgeInsets.all(20.0),  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
    children: <Widget>[  
      Column(  
        children: <Widget>[  
          Icon(Icons.access_alarm),  
          Text('Alarm')  
        ],  
      ),  
      Column(  
        children: <Widget>[  
          Icon(Icons.phone),  
          Text('Phone')  
        ],  
      ),  
      Column(  
        children: <Widget>[  
          Icon(Icons.book),  
          Text('Book')  
        ],  
      ),  
    ],  
  ),  
)
```

- **Button**

Terdapat beberapa Button yang umum dipakai yaitu, RaisedButton, MaterialButton, FlatButton, TextButton, OutlinedButton, dan ContainedButton.

RaisedButton dan MaterialButton akan sedikit menonjol.

FlatButton tombolnya akan datar tanpa ada efek-efek.

TextButton tombol berupa text seperti link

OutlinedButton memiliki garis border.

ContainedButton memiliki elevasi yang meningkat saat ditekan oleh pengguna

```
Column(  
  children: <Widget>[  
    RaisedButton(  
      color: Colors.amber,  
      child: Text("Raised Button"),  
      onPressed: () {},  
    ),  
    MaterialButton(  
      color: Colors.lime,  
      child: Text("Material Button"),  
      onPressed: () {},  
    ),  
  ],  
)
```

```

    ),
    FlatButton(
      color: Colors.lightGreenAccent,
      child: Text("Flat Button"),
      onPressed: () {},
    ),
    TextButton(
      child: Text("Text Button"),
      onPressed: () {},
    ),
    OutlinedButton(
      child: Text("Outlined Button"),
      onPressed: () {},
    ),
    ElevatedButton(
      child: Text("Elevated Button"),
      onPressed: () {},
    ),
  ],
)

```

- **Text Field**

Widget ini berfungsi untuk menerima input dari pengguna dan biasanya sering digunakan ketika didalam aplikasi form. Memiliki 2 jenis TextField dan TextFormField

```

Column(
  children: <Widget>[
    TextField(
      decoration: InputDecoration(
        border: OutlineInputBorder(),
        hintText: 'Enter a search term',
      ),
    ),
    TextFormField(
      decoration: const InputDecoration(
        icon: Icon(Icons.person),
        hintText: 'What do people call you?',
        labelText: 'Name *',
      ),
      onSave: (String? value) {
      },
      validator: (String? value) {
        return (value != null && value.contains('@')) ? 'Do not
use the @ char.' : null;
      },
    ),
  ],
)

```

- **Image**

Image merupakan widget yang dapat digunakan untuk menampilkan gambar, bisa menggunakan internet atau dengan local folder.

Internet:

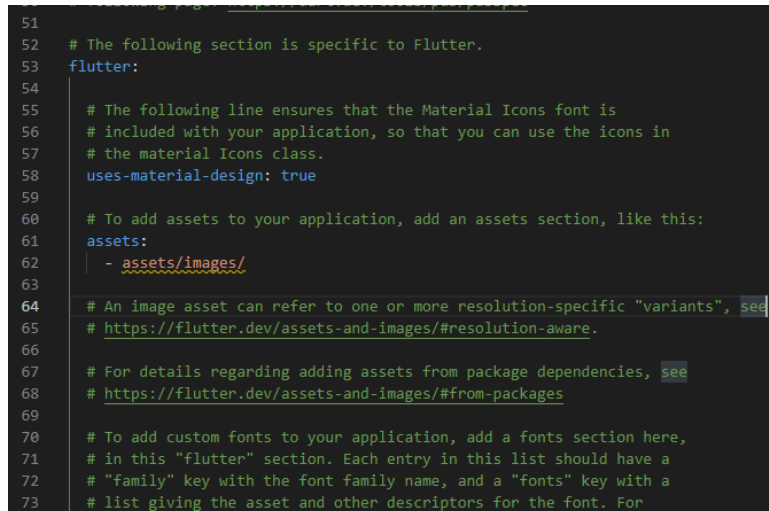
```
Image.Network('url');
```

Assets Lokal

1. Buatlah folder baru **assets/images** (posisi sejajar dengan pubspec.yaml, untuk penamaan folder bebas)
2. Masukkan gambar kedalam folder tersebut.
3. Registrasikan image / gambar yang dimasukkan dengan cara edit pada file **pubspec.yaml**

```
flutter:
  assets:
    - assets/images/
```

Menggunakan nama direktori gambar



```
51
52 # The following section is specific to Flutter.
53 flutter:
54
55 # The following line ensures that the Material Icons font is
56 # included with your application, so that you can use the icons in
57 # the material Icons class.
58 uses-material-design: true
59
60 # To add assets to your application, add an assets section, like this:
61 assets:
62   - assets/images/
63
64 # An image asset can refer to one or more resolution-specific "variants", see
65 # https://flutter.dev/assets-and-images/#resolution-aware.
66
67 # For details regarding adding assets from package dependencies, see
68 # https://flutter.dev/assets-and-images/#from-packages
69
70 # To add custom fonts to your application, add a fonts section here,
71 # in this "flutter" section. Each entry in this list should have a
72 # "family" key with the font family name, and a "fonts" key with a
73 # list giving the asset and other descriptors for the font. For
```

Perhatikan tab dan spasi karena apabila posisi value sejajar atau menjorok terlalu dalam akan error

Tampilan

```
Image.asset('assets/image/gambar.jpg');
```

Format gambar yang support yaitu: JPEG, jpg, Webp, GIF, animated Webp/GIF, PNG, BMP, dan WBMP.

Style

Flutter menggunakan *material design* sebagai style default untuk mengatur UI nya. Pada bagian ini, kita akan belajar mengenai penggunaan Widget **Scaffold** serta attribute yang penting untuk diketahui. Biasanya kita menggunakan duet **MaterialApp** dan **Scaffold**, didalam file yang sama, modifikasi menjadi

```
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(
      home: Scaffold(
        appBar: AppBar(title: Text("Belajar Flutter")),
      ),
    );
  }
}
```

Bagian-bagian yang ada:

1. **Material App** merupakan sebuah parent yang akan menerapkan style **material design**. Material app juga memiliki control untuk mengatur route, theme, supported locales, dan lain-lain. Pada sesi ini yang dibahas pada themenya saja.
2. **Scaffold** memiliki peran untuk mengatur struktur visual layout dengan mengimplementasi material design.
3. **appBar** salah satu property yang dimiliki Scaffold untuk membuat sebuah bar pada aplikasi.

Dokumentasi appBar flutter <https://api.flutter.dev/flutter/material/AppBar-class.html>

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

void main() => runApp(const MyApp());

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

  static const String _title = 'Flutter Code Sample';

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: _title,
      home: MyStatelessWidget(),
      debugShowCheckedModeBanner: false,
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    final ButtonStyle style =
      TextButton.styleFrom(primary:
        Theme.of(context).colorScheme.onPrimary);
    return Scaffold(
```

```

    appBar: AppBar(
      leading: Icon(Icons.dashboard),
      title: Text("Belajar Flutter"),
      actions: <Widget>[
        TextButton(
          style: style,
          onPressed: () {},
          child: const Text('Action 1'),
        ),
        TextButton(
          style: style,
          onPressed: () {},
          child: const Text('Action 2'),
        )
      ],
    ),
    floatingActionButton: FloatingActionButton(
      backgroundColor: Colors.yellow,
      child: Icon(Icons.search),
      onPressed: () {
        // ...
      },
    ),
  );
}

```

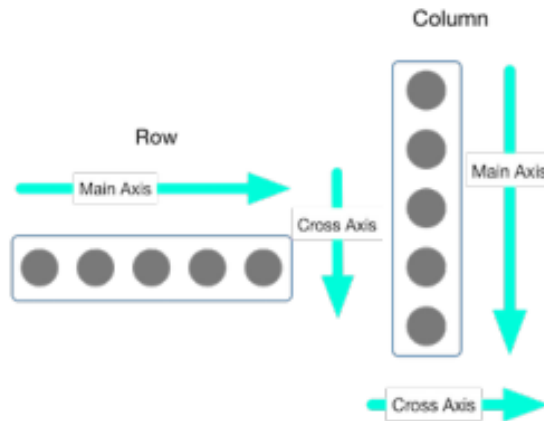
Bagian-bagian yang ada:

1. **leading, title, actions** merupakan kesatuan membentuk urutan penempatan sesuai dengan urutan nama,
2. **floatingActionButton** merupakan penambahan tombol float / melayang di pojok kanan bawah.
3. **debugShowCheckedModeBanner** secara default bernilai **true** yang akan menampilkan debug banner pada pojok kanan atas, untuk menghilangkan set valuenya menjadi false.

Pasangan Row-Column

Row berfungsi untuk mengatur widget agar tersusun secara horizontal, sedangkan column berlaku sebaliknya yaitu vertikal. Row dan column ini menggunakan children jadi dapat dimasukkan beberapa widget untuk menggunakannya. Row dan Column bisa di gabungkan, dimana Row bisa dimasukkan kedalam Column begitu juga sebaliknya.

Kita juga dapat mengontrol komponen didalamnya menggunakan properti **mainAxisAlignment** dan **crossAxisAlignment**.



Contoh Row

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
  ],
);
```

Contoh Column

```
Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
    Container(width: 50, height:50,
      Decoration: BoxDecoration(
        color: Colors.redAccent,
        shape: BoxShape.circle)
    ),
  ],
);
```


Tambahan ListView, GridView, StackView, SingleChildScrollView

Nah, ditambahkan ini temen-temen diharapkan belajar juga mempelajari membaca dokumentasi yang sudah di sediakan dari flutter, maka maka modul ini akan diberikan link menuju dokumentasi yang sudah di berikan di web flutter.

List View: <https://api.flutter.dev/flutter/widgets/ListView-class.html>

Grid View: <https://api.flutter.dev/flutter/widgets/GridView-class.html>

Stack View: <https://api.flutter.dev/flutter/widgets/Stack-class.html>

Single Child Scroll View: <https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html>

Mengatur Padding dan Margin

Padding adalah jarak atau ruang yang berada di andata border dan konten (dalam). Sedangkan margin jarak yang berada dari border keluar (luar).

Untuk padding cara termudah dapat menggunakan Widget padding. Ada untuk semua sisi, semua sisi custom, dan sisi tertentu.

Contoh semua sisi

```
Padding(  
  padding: EdgeInsets.all(8.0),  
  child: Card(child: Text("Hello World"))  
)
```

Contoh semua sisi custom

```
Padding(  
  padding: EdgeInsets.fromLTRB(8.0, 10.0, 10.0, 4.0),  
  child: Card(child: Text("Hello World"))  
)
```

Contoh semua sisi custom

```
Padding(  
  padding: EdgeInsets.only(left: 8.0, right: 8.0),  
  child: Card(child: Text("Hello World"))  
)
```

Selain menggunakan widget padding, kita juga dapat menggunakan Container

Contoh pada Container

```
Container(  
  padding: EdgeInsets.fromLTRB(8.0, 10.0, 10.0, 4.0),  
  child: Card(child: Text("Hello World"))  
)
```

Untuk margin kita bisa menggunakan pada widget Container karena tidak ada widget khusus untuk margin seperti pada padding. Untuk margin sama seperti padding ada untuk semua sisi, semua sisi custom, dan sisi tertentu saja.

Contoh Margin

```
Container(  
  margin: EdgeInsets.fromLTRB(8.0, 10.0, 10.0, 4.0),  
  child: Card(child: Text("Hello World"))  
)
```

Navigasi Sederhana

Kita akan membahas navigasi routing sederhana pada flutter. Navigator berfungsi untuk menampilkan konten ke halaman atau layar baru, jika pada native android route dinamakan activity dan di ios bernama viewControllner. Widget Navigator berkerja seperti stack (tumpukan). Ada 3 method yang dapat digunakan:

1. Navigator.push(): Metode menambahkan route lain diatas tumpukan screen saat ini. Halaman baru ditampilkan diatas halaman sebelumnya jadi ketika kembali dapat kembali ke halaman sebelumnya.
2. Navigator.pop(): Metode menghapus screen paling atas tumpukan / screen saat ini. Contoh pada main.dart
3. navigator.pushReplacement(): selain menambahkan stack, terdapat cara untuk mengganti halaman yang sedang aktif menggunakan method Navigator.pushReplacement(). Contoh penulisan

```
ElevatedButton(  
  child: Text('Pindah Screen'),  
  onPressed: () {  
    Navigator.pushReplacement(  
      context,  
      MaterialPageRoute(builder: (context) {  
        return SecondScreen();  
      })  
    );  
  },  
)
```

Dokumentasi: <https://docs.flutter.dev/cookbook/navigation/navigation-basics>

main.dart

```
import 'package:flutter/material.dart';  
  
void main() {  
  runApp(const MaterialApp(  
    title: 'Navigation Basics',  
    home: FirstRoute(),  
  ));  
}
```

```

    ));
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Route'),
      ),
      body: Center(
        child: ElevatedButton(
          child: const Text('Open route'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const
SecondRoute()),
            );
          },
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Route'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}

```

Named Routing

Named routing memberi nama pada routing dengan tujuan untuk mempermudah dalam membaca atau menentukan halaman dari navigasi. Disini akan menggunakan `Navigator.pushNamed` untuk menuju ke halaman baru namun tetap menggunakan navigasi pop untuk kembali ke halaman sebelumnya.

Dokumentasi: <https://docs.flutter.dev/cookbook/navigation/named-routes>

main.dart

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

void main() {
  runApp(
    MaterialApp(
      title: 'Named Routes Demo',
      // Start the app with the "/" named route. In this case, the app
      // starts
      // on the FirstScreen widget.
      initialRoute: '/',
      routes: {
        // When navigating to the "/" route, build the FirstScreen
        // widget.
        '/': (context) => const FirstScreen(),
        // When navigating to the "/second" route, build the SecondScreen
        // widget.
        '/second': (context) => const SecondScreen(),
      },
    ),
  );
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
          child: const Text('Launch screen'),
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}
```

```

    ),
  );
}
}

```

Routing Generator

Named routing terlihat lebih simple dan baik, namun apabila ingin membuat aplikasi dengan skala besar dan ingin mengatur animasi tiap transisi dari tiap routing maka `onGenerateRoute` merupakan salah satu pilihan yang dapat digunakan. Kita akan memisahkan `main`, `screen`, dan `routing` file.

main.dart

```

// jangan lupa import class yang ada
void main() {
  runApp(MaterialApp(
    onGenerateRoute: RouteGenerator.generateRoute,
    home: FirstScreen(),
  ));
}

```

screen.dart

```

import 'package:flutter/material.dart';
class FirstScreen extends StatelessWidget {
  const FirstScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Screen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/second');
              },
              child: const Text('Launch screen'),
            ),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/not-found');
              },
              child: const Text('Halaman Tidak Ada'),
            ),
          ],
        ),
      ),
    );
  }
}

```

```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}

```

routes.dart

```

import 'package:flutter/material.dart';
// jangan lupa import screen.dart

class RouteGenerator {
  static Route<dynamic> generateRoute(RouteSettings settings) {
    switch (settings.name) {
      case '/':
        return MaterialPageRoute(builder: (_) => FirstScreen());
      case '/second':
        return MaterialPageRoute(builder: (_) => SecondScreen());
      default:
        return _errorRoute();
    }
  }
  static Route<dynamic> _errorRoute() {
    return MaterialPageRoute(builder: (_) {
      return Scaffold(
        appBar: AppBar(title: Text("Error")),
        body: Center(child: Text('Error page')),
      );
    });
  }
}

```

Tambahan Drawer

Dokumentasi: <https://docs.flutter.dev/cookbook/design/drawer>

Tambahan Bottom Tab Navigator

Dokumentasi: <https://api.flutter.dev/flutter/material/BottomNavigationBar-class.html>

Tambahan Snack Bar

Dokumentasi: <https://docs.flutter.dev/cookbook/design/snackbars>