Flutter Sesi 2

Flutter Scaffold

Scaffold adalah sebuah widget dalam flutter yang menyediakan banyak widget seperti *Drawer, SnackBar, BottomNavigationBar, FloatingActionButton, AppBar, dan lain-lain*.

Scaffold akan memperluas atau menempati seluruh layar perangkat. Widget ini akan menempati ruang yang tersedia. Scaffold akan memberikan kerangka untuk menerapkan layout dasar material design dari aplikasi.

Berikut hirarki dari class scaffold :

```
Object
 ↳ Diagnosticable
  ↳ Diagnosticable Tree
   ↳ Widget
    ↳ StateFul Widget
     ↳ Scaffold
```

Constructor dari Class Scaffold

Constructor merupakan sebuah method atau fungsi khusus yang digunakan untuk membuat instance atau object dari class. Dalam konteks class Scaffold di Flutter, constructor digunakan untuk menginisialisasi dan mengatur berbagai properti dan widget yang terdapat pada tampilan aplikasi.

Sintaks constructor class Scaffold di Flutter dapat dituliskan dengan format sebagai berikut:

```
const Scaffold({
    Key key,
    this.appBar,
    this.body,
    this.floatingActionButton,
    this.floatingActionButtonLocation,
    this.floatingActionButtonAnimator,
    this.persistentFooterButtons,
    this.drawer,
    this.endDrawer,
    this.bottomNavigationBar,
    this.bottomSheet,
    this.backgroundColor,
    this.resizeToAvoidBottomPadding,
    this.resizeToAvoidBottomInset,
    this.primary = true,
    this.drawerDragStartBehavior
          = DragStartBehavior.start,
    this.extendBody = false,
    this.drawerScrimColor,
});
```

Properti dari Class Scaffold

Class Scaffold dalam Flutter adalah widget yang menyediakan kerangka kerja dasar untuk membangun aplikasi Flutter yang lengkap dengan mengintegrasikan widget-widget umum seperti *AppBar, Drawer, BottomNavigationBar, dan lain-lain.*

Salah satu properti penting dari class Scaffold adalah "appBar" yang bertipe AppBar. Properti ini digunakan untuk menentukan AppBar yang akan ditampilkan di bagian atas aplikasi.

Selain itu, class Scaffold juga memiliki properti "body" yang bertipe Widget. Properti ini digunakan untuk menentukan isi utama dari aplikasi. Isi utama ini bisa berupa widget seperti ListView, GridView, Column, Row, atau widget lainnya.

Class Scaffold juga memiliki beberapa properti lainnya seperti *"floatingActionButton", "bottomNavigationBar", "drawer", "endDrawer", "persistentFooterButtons", dan lain-lain* yang bisa digunakan untuk menambahkan fungsi dan interaksi tambahan pada aplikasi.

Secara keseluruhan, class Scaffold sangat penting dalam membangun aplikasi Flutter karena menyediakan kerangka kerja dasar yang mudah digunakan dan dapat disesuaikan dengan kebutuhan pengembang.

1. appBar

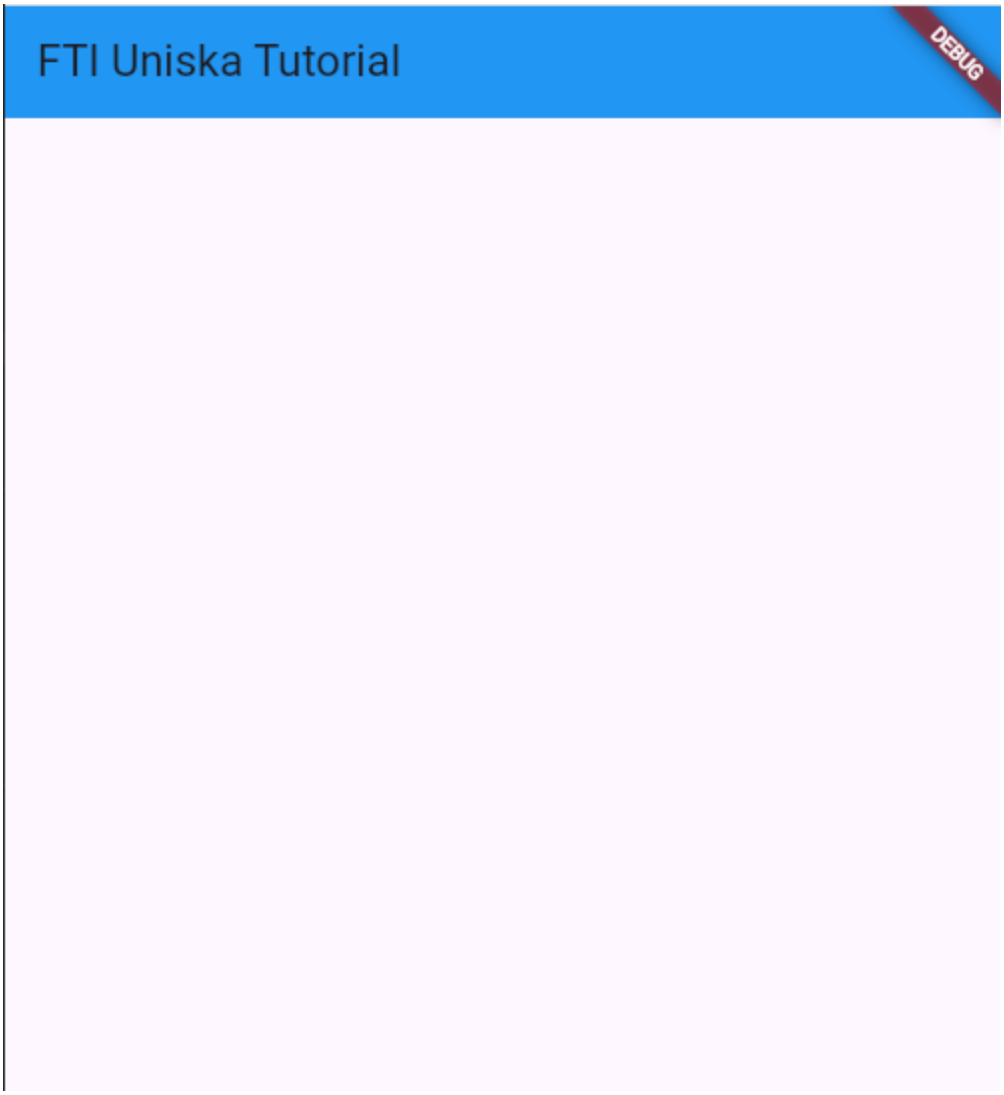
Properti ini menampilkan sebuah bilah horizontal yang biasanya ditempatkan di bagian atas Scaffold. Properti appBar menggunakan widget AppBar yang memiliki properti sendiri seperti *elevation, title, brightness, dan sebagainya*.

Latihan pengunaan appBar pada project seperti berikut :

```
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(
          backgroundColor: Colors.blue,
          title: const Text('FTI Uniska Tutorial'),
        ),
      ),
    );
  }
}
```

Hasil :



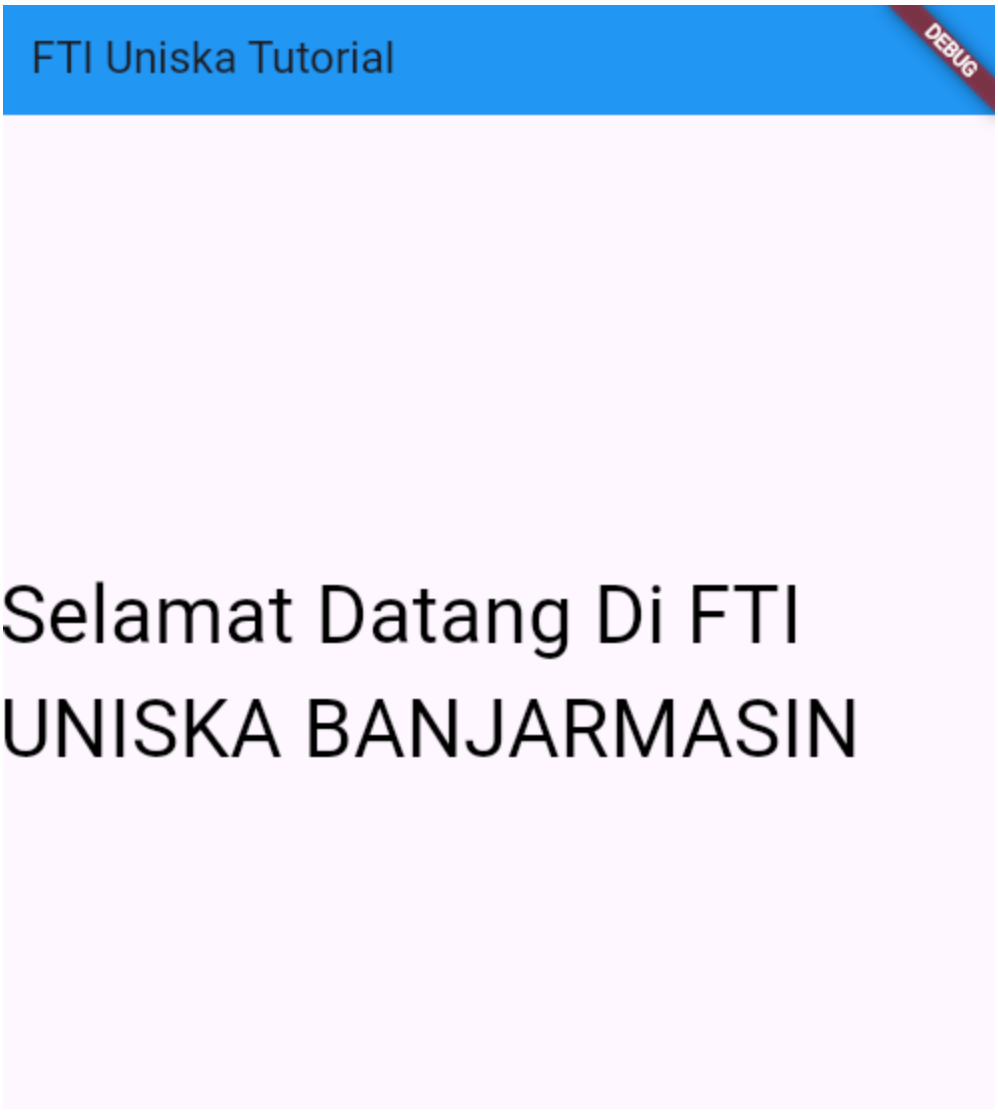
2. Body

Properti ini akan menampilkan konten utama atau primer pada Scaffold. Konten ini akan berada di bawah appBar dan di bawah floatingActionButton. Secara default, widget yang berada di dalam body akan ditampilkan pada sudut kiri.

Latihan penggunaan body seperti berikut:

```dart
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(
          backgroundColor: Colors.blue,
          title: const Text('FTI Uniska Tutorial'),
        ),
        body: const Center(
          child: Text(
            "Selamat Datang Di FTI UNISKA BANJARMASIN",
            style: TextStyle(color: Colors.black, fontSize: 40.0),
          ),
        ),
      ),
    );
  }
}
```

Hasil :



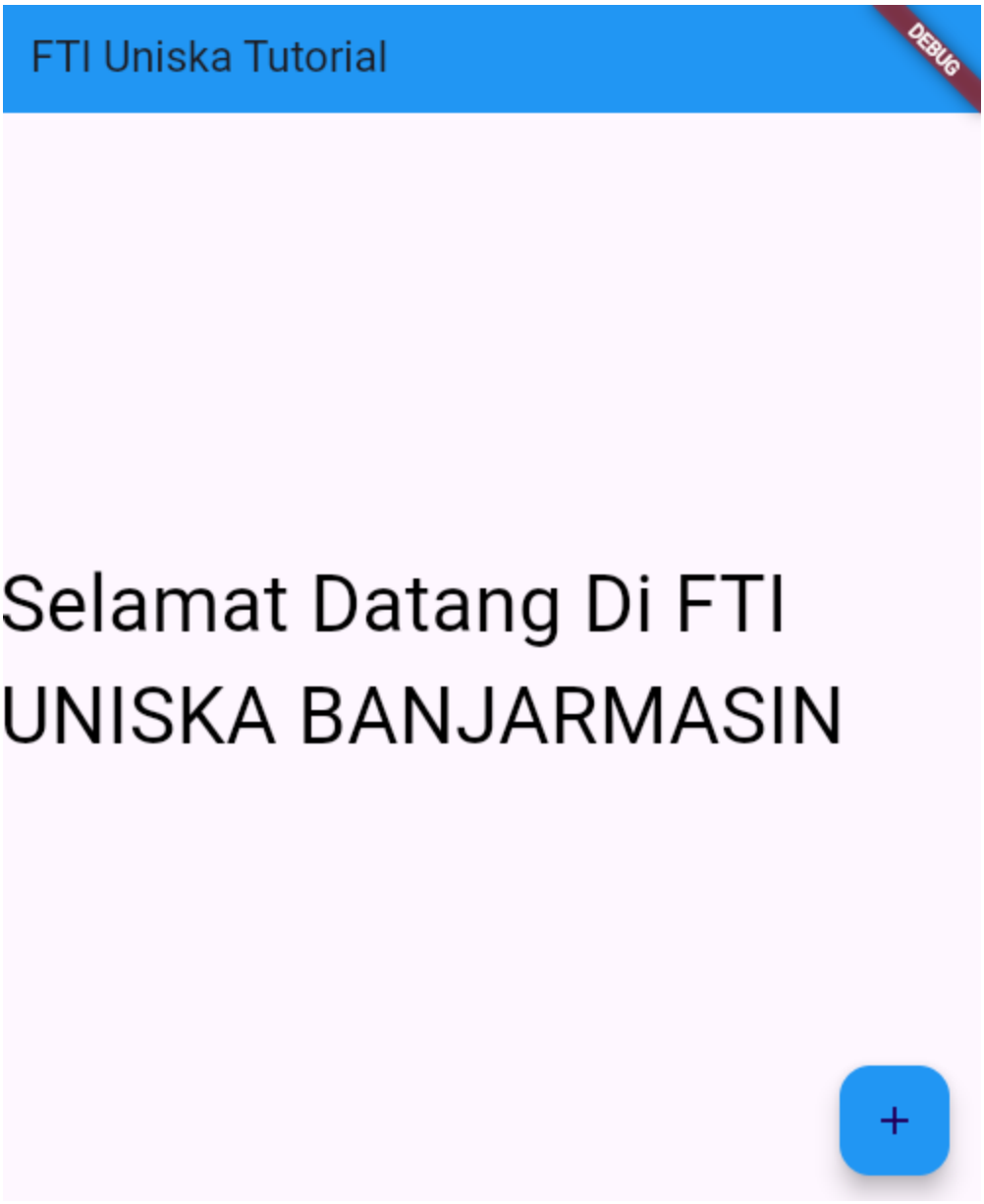
3. floatingActionButton

FloatingActionButton adalah sebuah tombol yang biasanya diletakkan di sudut kanan bawah. Tombol ini berupa sebuah ikon yang mengambang di atas konten layar dan tetap berada di posisi yang sama bahkan jika kita menggulir halaman.

Latihan penggunaan *floatingActionButton* seperti berikut:

```dart
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(
          backgroundColor: Colors.blue,
          title: const Text('FTI Uniska Tutorial'),
        ),
        body: const Center(
          child: Text(
            "Selamat Datang Di FTI UNISKA BANJARMASIN",
            style: TextStyle(color: Colors.black, fontSize: 40.0),
          ),
        ),
        floatingActionButton: FloatingActionButton(
            elevation: 10.0,
            child: const Icon(Icons.add),
            onPressed: () {
              //action on button press
            }),
      ),
    );
  }
}
```

Hasil :



Di sini properti elevation digunakan untuk memberikan efek bayangan pada tombol. Icon digunakan untuk menempatkan ikon tombol dengan menggunakan beberapa ikon yang sudah terpasang di SDK flutter. Fungsi onPressed() adalah fungsi yang akan dipanggil saat tombol ditekan dan baris kode di dalam fungsi akan dieksekusi.

4. Drawer

Drawer adalah panel menu yang muncul di samping layar aplikasi. Untuk membuka menu tersebut, pengguna harus menggeser layar ke kiri atau ke kanan, sesuai dengan tindakan yang telah ditentukan. Ikon untuk membuka drawer juga secara otomatis disediakan pada AppBar pada posisi tertentu. Tindakan membuka drawer juga sudah diatur secara otomatis oleh komponen Scaffold.

Latihan penggunaan *drawer* seperti berikut:

```
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(
          backgroundColor: Colors.blue,
          title: const Text('FTI Uniska Tutorial'),
        ),
        body: const Center(
          child: Text(
            "Selamat Datang Di FTI UNISKA BANJARMASIN",
            style: TextStyle(color: Colors.black, fontSize: 40.0),
          ),
        ),
        floatingActionButton: FloatingActionButton(
            elevation: 10.0,
            backgroundColor: Colors.blue,
            child: const Icon(Icons.add),
```

```
            onPressed: () {
              //action on button press
            }),
        drawer: Drawer(
          child: ListView(
            children: const [
              DrawerHeader(
                decoration: BoxDecoration(
                  color: Colors.blue,
                ),
                child: Text(
                  'FTI Tutorial',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 24,
                  ),
                ),
              ),
              ListTile(
                title: Text('Item 1'),
              ),
              ListTile(
                title: Text('Item 2'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Sebagai parent widget, dalam contoh di atas digunakan ListView dan membagi panel menjadi dua bagian, yaitu Header dan Menu. Dalam contoh di atas juga digunakan DrawerHeader untuk memodifikasi bagian Header panel. Di bagian Header, dapat ditampilkan ikon atau detail pengguna sesuai dengan aplikasi. ListTile juga digunakan untuk menambahkan item pada bagian Menu.

Hasil :

Kita juga dapat menambahkan ikon sebelum item menggunakan properti leading dari ListTile, di mana teknik ini akan menggunakan widget Icon.

Latihan penggunaan *drawer* + Icon seperti berikut:

```
            ListTile(
              title: Text('Sosial'),
              leading: Icon(Icons.people),
            ),
            ListTile(
              title: Text('Inbox'),
              leading: Icon(Icons.mail),
            ),
```

Hasil :

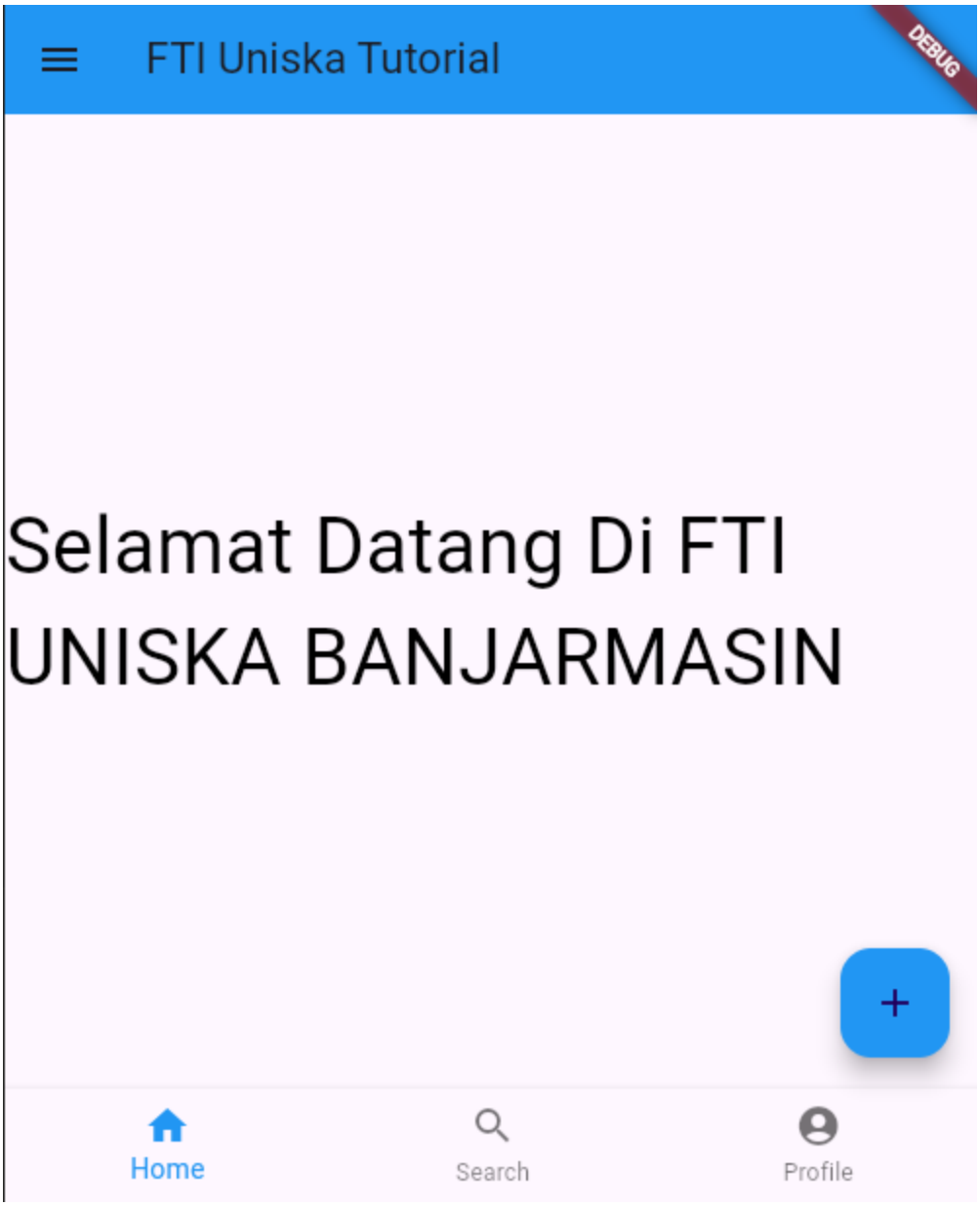
5. bottomNavigationBar

Properti bottomNavigationBar adalah seperti menu yang berada di bagian bawah Scaffold. Kita sering melihat navigation bar ini pada kebanyakan aplikasi. Kita bisa menambahkan beberapa ikon, teks, atau keduanya dalam bar sebagai item.

Latihan penggunaan *bottomNavigationBar* seperti berikut:

```
      bottomNavigationBar: BottomNavigationBar(
          currentIndex: 0,
          fixedColor: Colors.blue,
          items: const [
            BottomNavigationBarItem(
              label: "Home",
              icon: Icon(Icons.home),
            ),
            BottomNavigationBarItem(
              label: "Search",
              icon: Icon(Icons.search),
            ),
            BottomNavigationBarItem(
              label: "Profile",
              icon: Icon(Icons.account_circle),
            ),
          ],
          onTap: (int indexOfItem) {}),
```

Hasil :



Dalam contoh di atas digunakan widget BottomNavigationBar untuk menampilkan bilah navigasi. Untuk mengatur warna ikon aktif, digunakan properti fixedColor. Untuk menambahkan item pada bilah navigasi, digunakan widget BottomNavigationBarItems yang memungkinkan untuk memberikan teks dan ikon. Untuk menangani tindakan yang dilakukan ketika pengguna

mengetuk item pada bilah navigasi, digunakan fungsi onTap(int indexOfItem) yang akan dieksekusi sesuai dengan indeks posisi item yang ditekan.

Flutter MaterialApp

MaterialApp adalah class atau widget bawaan pada Flutter yang biasanya menjadi komponen utama atau inti dari aplikasi Flutter. Widget MaterialApp merupakan pembungkus dari widget Material lainnya, sehingga kita dapat mengakses semua komponen dan widget lain yang disediakan oleh SDK Flutter seperti Text, DropdownButton, AppBar, Scaffold, ListView, StatelessWidget, StatefulWidget, IconButton, TextField, Padding, ThemeData, dan lainnya. Selain itu, dengan menggunakan class MaterialApp, kita dapat mengakses banyak widget lainnya yang dapat digunakan untuk membuat aplikasi yang menarik dan mengikuti panduan Material Design.

Constructor dari Class MaterialApp

```
ThemeData theme,
ThemeData darkTheme,
ThemeData highContrastTheme,
ThemeData highContrastDarkTheme,
ThemeMode themeMode: ThemeMode.system,
Locale locale,
Iterable localizationsDelegates,
LocaleListResolutionCallback localeListResolutionCallback,
LocaleResolutionCallback localeResolutionCallback,
Iterable supportedLocales: const [Locale('en', 'US')],
bool debugShowMaterialGrid: false,
bool showPerformanceOverlay: false,
```

```
bool checkerboardRasterCacheImages: false,
bool checkerboardOffscreenLayers: false,
bool showSemanticsDebugger: false,
bool debugShowCheckedModeBanner: true,
Map<LogicalKeySet, Intent> shortcuts,
Map<Type, Action> actions}
)
```

Properti dari Class MaterialApp

Widget MaterialApp merupakan salah satu widget utama di Flutter yang digunakan untuk membuat aplikasi dengan Material Design. Beberapa properti yang terdapat pada widget MaterialApp adalah sebagai berikut:

- ✓ **action:** Properti ini menerima object Map<Type, Action<Intent>> yang mengontrol intent keys.
- ✓ **backButtonDispatcher:** Properti ini menentukan bagaimana mengatasi tombol kembali (back button).
- ✓ **checkerboardRasterCacheImage:** Properti ini menerima object boolean. Jika diatur ke true, maka akan menampilkan "checkerboarding" (papan catur) pada gambar cache raster.
- ✓ **color:** Properti ini mengontrol warna utama yang digunakan dalam aplikasi.
- ✓ **darkTheme:** Properti ini menyediakan data tema untuk tema gelap untuk aplikasi.

- ✓ **debugShowCheckedModeBanner:** Properti ini menerima object boolean untuk menentukan apakah menampilkan banner debug atau tidak.
- ✓ **debugShowMaterialGird:** Properti ini menerima object boolean. Jika diatur ke true, maka akan menampilkan grid dasar dari material app.
- ✓ **highContrastDarkTheme:** Properti ini menyediakan data tema untuk tema kontras tinggi.
- ✓ **home:** Properti ini menerima object widget untuk ditampilkan pada rute default aplikasi.
- ✓ **initialRoute:** Properti ini menerima object string untuk memberi nama rute pertama di mana navigator dibangun.
- ✓ **locale:** Properti ini mengontrol pengaturan lokal/bahasa untuk MaterialApp.
- ✓ **localizationsDelegate:** Properti ini menyediakan delegasi untuk pengaturan lokal/bahasa.
- ✓ **navigatorObserver:** Properti ini menerima object GlobalKey<NavigatorState> untuk menghasilkan kunci saat membangun navigator.
- ✓ **onGenerateInitialRoutes:** Properti ini menerima object InitialRouteListFactory typedef untuk menghasilkan rute awal.
- ✓ **onGenerateRoute:** Properti ini menerima object RouteFactory dan digunakan saat aplikasi dinavigasi ke rute bernama.

- ✓ **onGenerateTitle:** Sifat ini menerima object RouteFactory typedef untuk menghasilkan judul string untuk aplikasi jika disediakan.
- ✓ **onUnknownRoute:** Properti onUnknownRoute mengambil object RouteFactory typedef untuk memberikan rute dalam kasus metode lain gagal.
- ✓ **routeInformationParse:** Properti routeInformationParser menyimpan object RouteInformationParser<T> untuk mengubah informasi routing dari routeInformationProvider menjadi tipe data generik.
- ✓ **routeInformationProvider:** Properti routeInformationProvider mengambil object class RouteInformationProvider. Properti ini bertanggung jawab untuk menyediakan informasi routing.
- ✓ **routeDelegate:** Properti routeDelegate mengambil object RouterDelegate<T> untuk mengkonfigurasi widget yang diberikan.
- ✓ **routes:** Properti routes mengambil object LogicalKeySet class untuk mengontrol routing tingkat teratas aplikasi.
- ✓ **shortcuts:** Properti shortcuts mengambil object LogicalKeySet class untuk menentukan pintasan keyboard untuk aplikasi.
- ✓ **showPerformanceOverlay:** Properti showPerformanceOverlay mengambil nilai boolean untuk mengaktifkan atau menonaktifkan overlay kinerja.

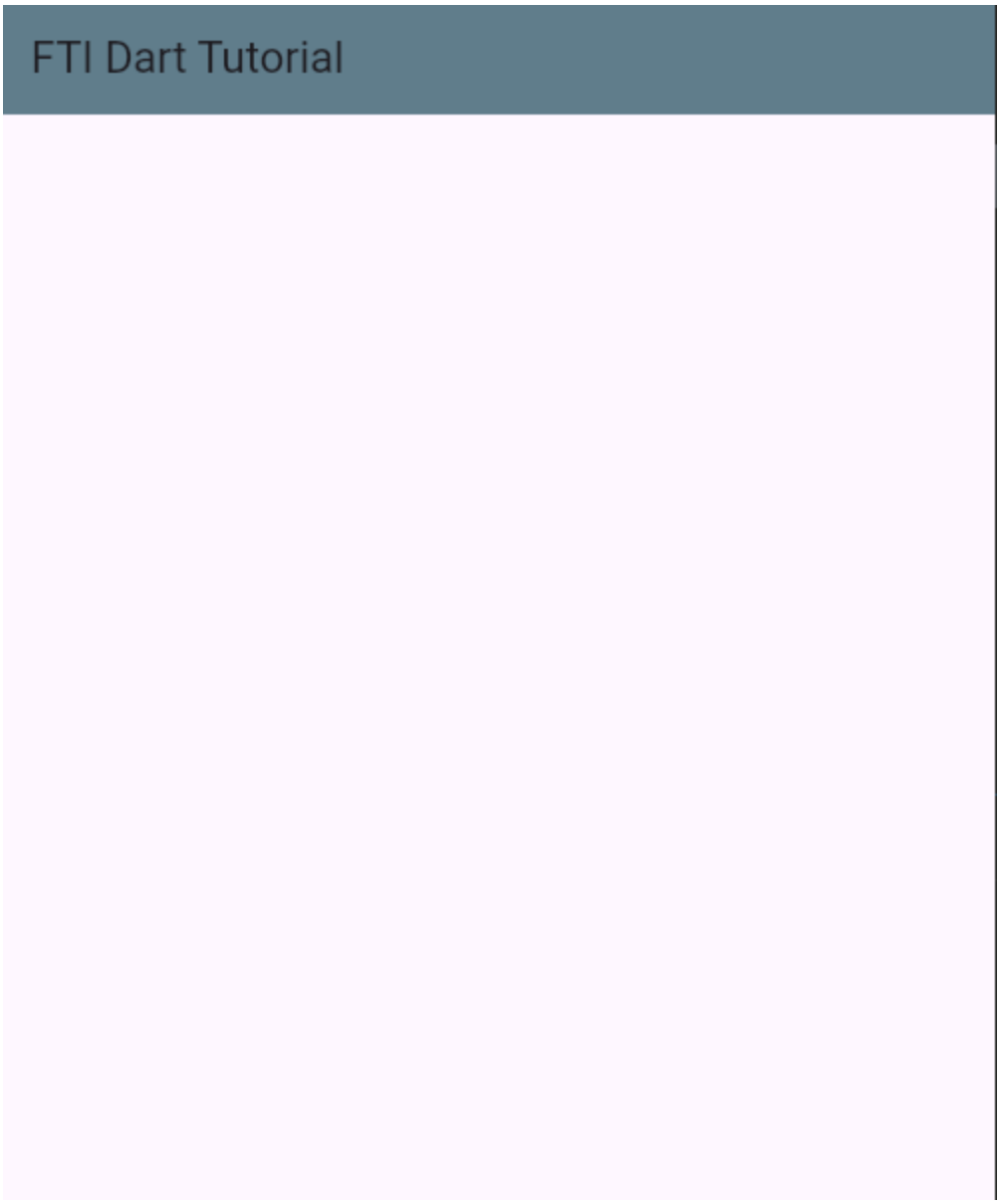
- ✓ **showSemantisDebugger:** Properti showSemanticsDebugger mengambil nilai boolean. Jika diatur ke true, properti ini menunjukkan beberapa informasi yang dapat diakses.
- ✓ **supportedLocales:** Properti supportedLocales menyimpan lokal yang digunakan dalam aplikasi dengan mengambil object Iterable<E> class.
- ✓ **theme:** Properti theme mengambil object ThemeData class untuk menggambarkan tema untuk MaterialApp.
- ✓ **themeMode:** Properti themeMode menyimpan object enum ThemeMode untuk memutuskan tema untuk aplikasi material.
- ✓ **title:** Properti title mengambil object string untuk memutuskan deskripsi satu baris dari aplikasi untuk perangkat.
- ✓ **navigatorObservers:** Properti ini menyimpan object List<NavigatorObserver> untuk membuat daftar pengamat untuk navigator.

Berikut ini adalah kode sederhana dalam bahasa Dart untuk membuat sebuah layar dengan AppBar yang memiliki judul "FTI Dart Tutorial".

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

void main() {
  runApp(const NGTapp());
}

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

  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FTI Dart Tutorial',
      theme: ThemeData(primarySwatch: Colors.purple),
      darkTheme: ThemeData(primarySwatch: Colors.grey),
      color: Colors.amberAccent,
      supportedLocales: {const Locale('en', ' ')},
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        appBar: AppBar(
          title: const Text('FTI Dart Tutorial'),
          backgroundColor: Colors.blueGrey,
        ),
      ),
    );
  }
}
```

Hasil Dart :



Di sini, kita dapat melihat bahwa teks yang telah ditetapkan di judul AppBar ditampilkan di bagian atas. Warna tema default yang telah didefinisikan pada kode adalah BlueGray, seperti yang telah kita atur sebelumnya. Fungsi runApp() akan memanggil widget NGTapp, yang mengembalikan widget MaterialApp. Widget MaterialApp menentukan tema, localization, judul, widget home, dan lain sebagainya.

Penjelasan Output Kode diatas :

- ✓ import statement: Pernyataan impor digunakan untuk mengimpor pustaka yang disediakan oleh Flutter SDK. Di sini, kami telah mengimpor file 'material.dart' sehingga dapat menggunakan semua widget Flutter yang mengimplementasikan desain material.
- ✓ main() function: Seperti bahasa pemrograman lainnya, kami juga memiliki fungsi utama di mana pernyataan yang akan dieksekusi saat aplikasi dimulai ditulis. Tipe pengembalian fungsi utama adalah 'void'.
- ✓ runApp(Widget widget) function: Fungsi runApp(Widget widget) mengambil widget sebagai argumen dan menetapkannya pada layar. Fungsi ini memberikan batasan pada widget agar sesuai dengan layar dan membuat widget yang diberikan menjadi widget root aplikasi dengan widget lain sebagai anaknya. Di sini, digunakan MaterialApp sebagai widget root di mana dalam widget root akan ditentukan widget lainnya.
- ✓ MaterialApp() widget: Kami telah membahas MaterialApp di awal. Mari kita lihat berbagai properti dari widget MaterialApp
- ✓ title: Properti 'title' digunakan untuk memberikan deskripsi singkat tentang aplikasi kepada pengguna. Ketika pengguna menekan tombol aplikasi terbaru di ponsel, teks yang diproses di judul ditampilkan.
- ✓ theme: Properti 'theme' digunakan untuk memberikan tema default ke aplikasi seperti warna tema aplikasi. Untuk ini, kami menggunakan class atau widget bawaan bernama ThemeData(). Di widget ThemeData(), dapat

dituliskan properti yang berbeda terkait tema. Di sini, digunakan ‘primarySwatch’ untuk menentukan warna tema default aplikasi dan memilih warna dengan menggunakan class Colors dari library material. Di ThemeData(), ditentukan beberapa properti lain seperti TextTheme, Brightness (Dapat mengaktifkan tema gelap dengan ini), AppBarTheme, dan lainnya.

- ✓ home: Properti ‘home’ digunakan untuk rute default aplikasi, yang berarti widget yang didefinisikan di dalamnya akan ditampilkan saat aplikasi dimulai secara normal. Di sini, ditentukan widget Scaffold di dalam properti ‘home’. Di dalam Scaffold, kami mendefinisikan berbagai properti seperti appBar, body, floatingActionButton, backgroundColor, dll. Sebagai contoh, di properti ‘appBar’, digunakan widget AppBar() dan memasukkan ‘Nextgen Tutorial’ sebagai judul, yang akan ditampilkan di bagian atas aplikasi tepatnya di appbar.
- ✓ Properti lain dalam MaterialApp() adalah ‘debugShowCheckedModeBanner’ (digunakan untuk menghapus tag debug di sudut atas), ‘darkTheme’ (untuk meminta mode gelap dalam aplikasi), ‘color’ (untuk warna primer aplikasi), ‘routes’ (untuk tabel routing aplikasi), ‘themeMode’ (untuk menentukan tema mana yang akan digunakan), dan ‘supportedLocales’ (berisi daftar bahasa yang didukung aplikasi), dan lainnya.

Menggunakan Drawer pada flutter

Widget Drawer digunakan untuk memberikan akses ke berbagai halaman dan fungsionalitas dalam aplikasi Anda. Widget ini ditampilkan sebagai tiga garis horizontal sejajar di bagian atas scaffold. Gerakan horizontal widget ini mengarahkan pengguna ke rute yang berbeda dalam aplikasi Flutter.

Untuk menggunakan drawer, Anda perlu mengimpor paket "package:flutter/material.dart". Navigasi Drawer terdiri dari tiga bagian: header, body, dan footer. Konsepnya adalah memiliki navigator dengan beberapa item sebagai anak dari drawer yang akan dinavigasikan ke destinasi yang berbeda ketika di-tap. Semua child dari widget Drawer biasanya berada dalam ListView. Pada awalnya, hanya DrawerHeader yang muncul di antarmuka pengguna dan dapat diperluas secara horizontal saat di-tap.

Contoh Sintaks Drawer :

```
Drawer({Key key, double elevation: 16.0, Widget child, String semanticLabel})
```

Properti dari Widget Drawer

- ✓ child: Widget lain di bawah widget ini dalam pohon hirarki.
- ✓ hashCode: Kode hash untuk object ini.
- ✓ key: Mengontrol cara widget lain digantikan oleh widget ini dalam pohon hirarki.
- ✓ runtimeType: Representasi tipe object pada saat runtime.

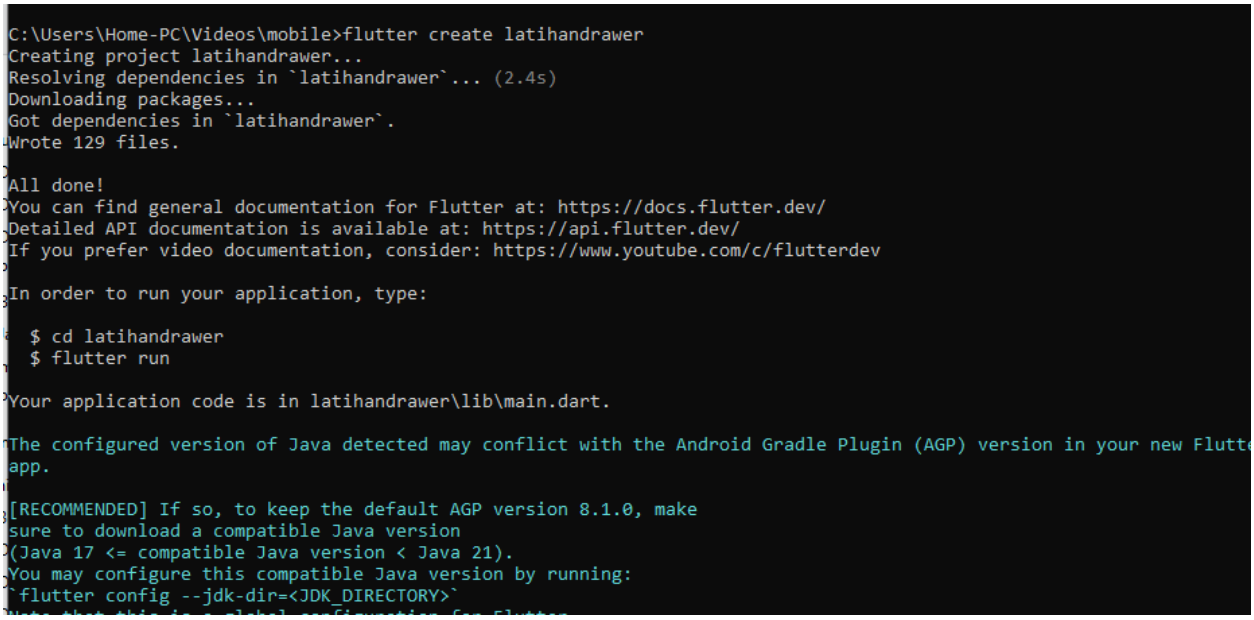
- ✓ elevation: Koordinat z di mana drawer ditempatkan relatif terhadap induknya.
- ✓ semanticLabel: Label semantik yang digunakan oleh kerangka aksesibilitas untuk mengumumkan transisi layar ketika drawer dibuka dan ditutup.

build(BuildContext context) → Fungsi ini menentukan bagian UI yang dibuat oleh widget. Fungsi ini dipanggil ketika widget Drawer ditambahkan ke dalam pohon hirarki pada BuildContext tertentu, dan ketika dependensi widget Drawer berubah.

Drawer merupakan sebuah fitur yang mudah diimplementasikan dan sangat berguna untuk menyeimbangkan berbagai fungsionalitas pada aplikasi mobile Anda. Dengan menggunakan Drawer, pengguna dapat dengan mudah mengakses berbagai fitur dan tujuan dalam aplikasi Anda, terutama pada aplikasi yang kompleks dengan banyak layar. Dalam hal ini, pengguna dapat dengan mudah beralih antara layar yang berbeda dan menyelesaikan tugas yang diperlukan.

Langkah Membuat Drawer :

1. Buatlah proyek Flutter: Buka terminal dan navigasikan ke lokasi di mana Anda ingin membuat proyek. Gunakan perintah “flutter create nama_project” untuk membuat proyek Flutter Anda.



2. *Buatlah widget Scaffold:*

Buatlah widget dasar dengan mengembalikan Scaffold Widget pada class MyApp yang bisa berupa stateless atau stateful widget. Scaffold Widget memberikan kerangka kerja untuk menentukan struktur tata letak visual dasar dari aplikasi Flutter.

```
Scaffold(
      drawer:
);
```

3. Tambahkan Drawer di dalam scaffold:

Atur properti drawer dari scaffold menjadi Drawer dengan ListView sebagai child-nya, atau Anda juga dapat menambahkan Container dengan ListView sebagai child-nya. Berbagai ListViews yang berisi item yang diinginkan perlu ditampilkan di dalam drawer.

```
Scaffold(
      drawer: Drawer(
     //…
     ),
);
```

4. Tambahkan fungsi untuk menutup drawer: Navigator digunakan untuk mengimplementasikan fungsionalitas penutupan drawer saat pengguna ingin menutupnya setelah mengetuk beberapa item. Hal ini dapat dilakukan dengan menggunakan State Navigator.

```
Navigator.of(context)pop();
```

Sebuah App Drawer dibagi menjadi tiga kategori:

Standar Navigation Drawer: Kategori ini memungkinkan pengguna untuk berinteraksi dengan konten layar dan drawer pada saat yang sama.

Modal Navigation Drawer: Kategori ini memungkinkan pengguna hanya berinteraksi dengan drawer, karena memblokir interaksi pengguna dengan bagian lain dari layar.

Bottom Navigation Drawer: Kategori ini mirip dengan Modal Navigation Drawer, namun orientasi drawernya mengarah ke bagian bawah layar.

Latihan penggunaan *Flutter Drawer* seperti berikut:

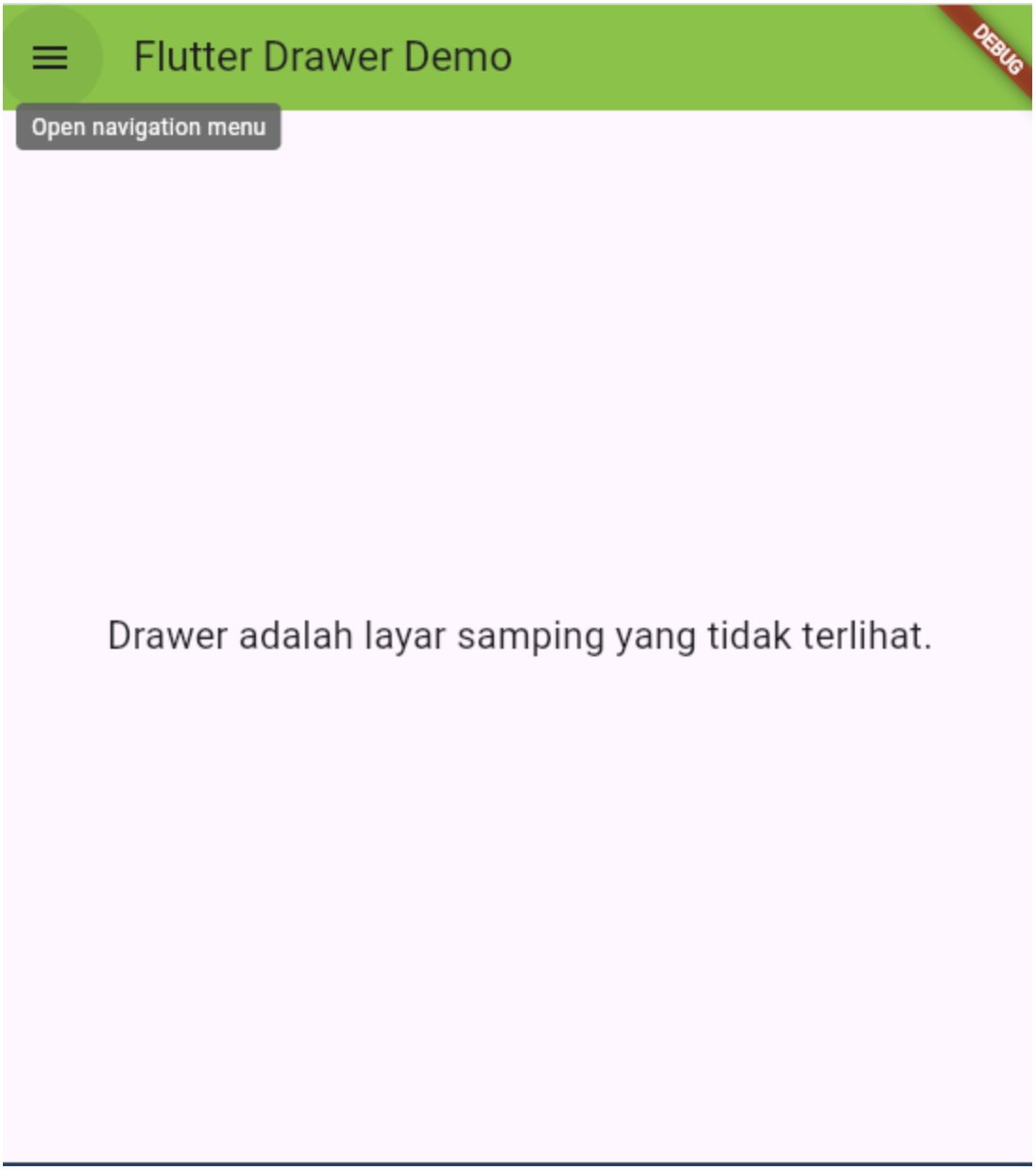
```
import 'package:flutter/material.dart';

// Fungsi ini memulai proses build dari aplikasi flutter.
```

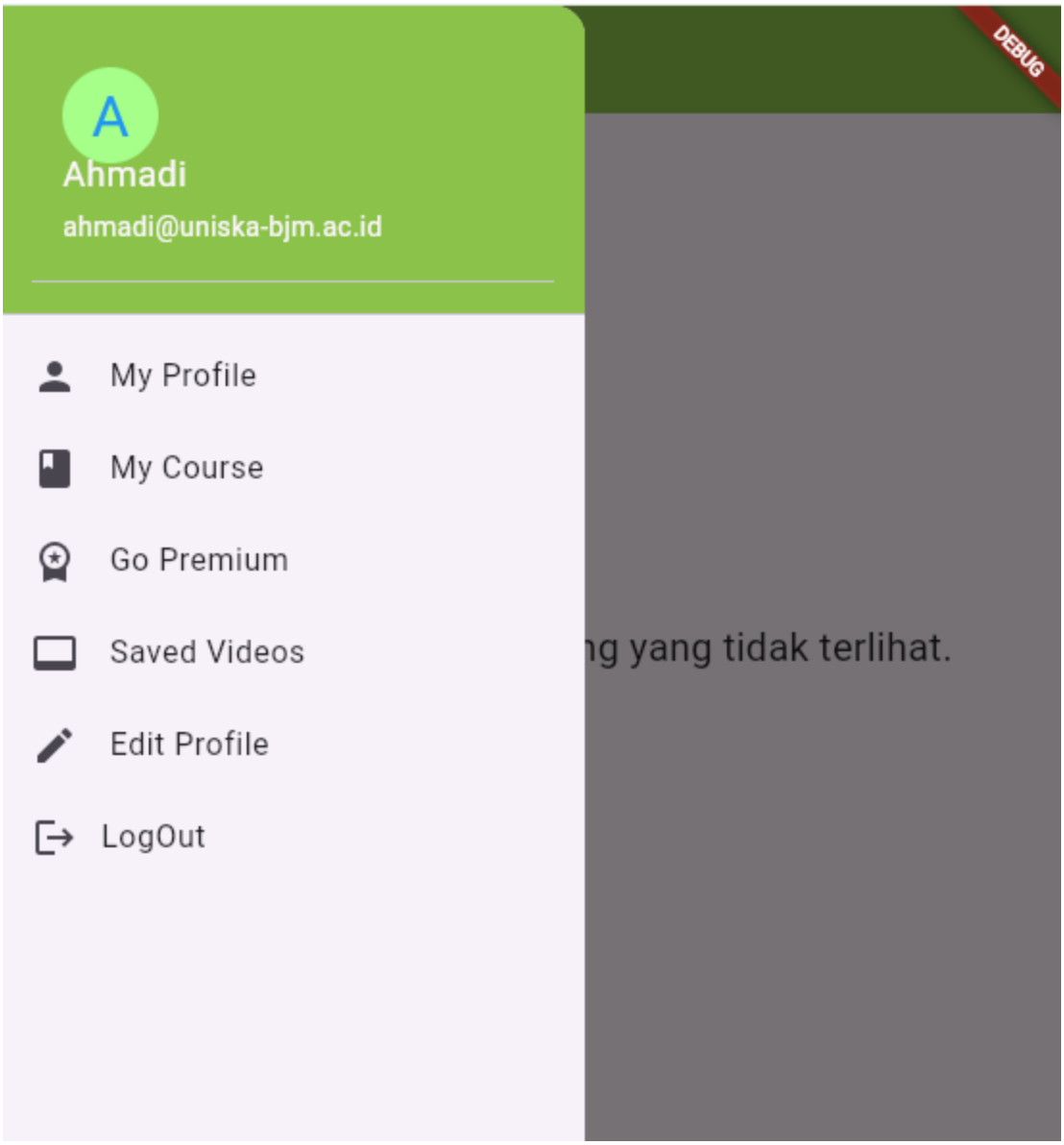
```dart
void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  final appTitle = 'Flutter Drawer Demo';

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: appTitle,
      home: MyHomePage(title: appTitle),
    ); // MaterialApp
  }
}

class MyHomePage extends StatelessWidget {
  final String title;

  const MyHomePage({Key? key, required this.title}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(title),
        backgroundColor: Colors.lightGreen,
      ),
      body: const Center(
          child: Text(
        'Drawer adalah layar samping yang tidak terlihat.',
        textAlign: TextAlign.center,
        style: TextStyle(fontSize: 20.0),
      )),
      drawer: Drawer(
        child: ListView(
          padding: const EdgeInsets.all(0),
          children: [
            const DrawerHeader(
```

```
              decoration: BoxDecoration(
                color: Colors.lightGreen,
              ), //BoxDecoration
              child: UserAccountsDrawerHeader(
                decoration: BoxDecoration(color: Colors.lightGreen),
                accountName: Text(
                  "Ahmadi",
                  style: TextStyle(fontSize: 18),
                ),
                accountEmail: Text("ahmadi@uniska-bjm.ac.id"),
                currentAccountPictureSize: Size.square(50),
                currentAccountPicture: CircleAvatar(
                  backgroundColor: Color.fromARGB(255, 165, 255, 255, 137),
                  child: Text(
                    "A",
                    style: TextStyle(fontSize: 30.0, color:
Colors.blue),
                  ), //Text
                ), //circleAvatar
              ), //UserAccountDrawerHeader
            ), //DrawerHeader
            ListTile(
              leading: const Icon(Icons.person),
              title: const Text(' My Profile '),
              onTap: () {
                Navigator.pop(context);
              },
            ),
            ListTile(
              leading: const Icon(Icons.book),
              title: const Text(' My Course '),
              onTap: () {
                Navigator.pop(context);
              },
            ),
            ListTile(
              leading: const Icon(Icons.workspace_premium),
              title: const Text(' Go Premium '),
              onTap: () {
```

```
              Navigator.pop(context);
            },
          ),
          ListTile(
            leading: const Icon(Icons.video_label),
            title: const Text(' Saved Videos '),
            onTap: () {
              Navigator.pop(context);
            },
          ),
          ListTile(
            leading: const Icon(Icons.edit),
            title: const Text(' Edit Profile '),
            onTap: () {
              Navigator.pop(context);
            },
          ),
          ListTile(
            leading: const Icon(Icons.logout),
            title: const Text('LogOut'),
            onTap: () {
              Navigator.pop(context);
            },
          ),
        ],
      ),
    ), //Drawer
  );
 }
}
```

Hasil Flutter Drawer :


Flutter Drawer Demo
DEBUG
Open navigation menu
Drawer adalah layar samping yang tidak terlihat.


A
Ahmadi
ahmadi@uniska-bjm.ac.id
My Profile
My Course
Go Premium
Saved Videos
Edit Profile
LogOut
DEBUG
ng yang tidak terlihat.

Menggunakan RichText pada flutter

Widget RichText digunakan untuk menampilkan teks dengan gaya yang berbeda-beda. Untuk mendefinisikan teks yang ditampilkan, digunakan sebuah pohon object TextSpan, dimana setiap sub-pohon memiliki gaya yang berbeda. Tergantung pada batasan tata letak, teks tersebut dapat dipecah menjadi beberapa baris atau tetap ditampilkan dalam satu baris yang sama. Hal ini bergantung pada kebutuhan penggunaan.

Construktor RichText:

```
RichText(
{Key key,
@required InlineSpan text,
TextAlign textAlign: TextAlign.start,
TextDirection textDirection,
bool softWrap: true,
TextOverflow overflow:
TextOverflow.clip,
double textScaleFactor: 1.0,
int maxLines,
Locale locale,
StrutStyle strutStyle,
TextWidthBasis textWidthBasis: TextWidthBasis.parent,
TextHeightBehavior textHeightBehavior,})
```

- ✓ children: Widget-widget di bawah widget ini dalam struktur pohon.
- ✓ hashCode: Kode hash unik yang diberikan untuk object ini.
- ✓ key: Mengontrol bagaimana satu widget diganti oleh widget lain dalam struktur pohon.
- ✓ runtimeType: Representasi tipe object pada waktu runtime.

- ✓ text: Teks yang akan ditampilkan dalam widget ini.
- ✓ textAlign: Cara teks disusun secara horizontal di dalam widget.
- ✓ local: Properti ini mengambil object class Locale sebagai input. Ini mengontrol pengaturan font yang digunakan untuk teks berdasarkan bahasa yang dipilih.
- ✓ maxLines: Properti maxLines menentukan jumlah maksimum baris teks yang dapat ditampilkan dalam widget.
- ✓ overflow: TextOverflow adalah sebuah object enum yang mengatur perilaku teks dalam kasus overflow.
- ✓ softWrap: Properti ini mengambil nilai boolean sebagai input. Jika disetel ke false, maka teks tidak akan dibungkus ke dalam baris baru jika melebihi lebar widget.
- ✓ textDirection: Properti ini mengambil object class TextDirection sebagai input untuk menentukan arah teks yang ditampilkan.
- ✓ textHightBehaviour: TextHeightBehavior adalah object class yang mengatur perilaku tinggi teks dalam widget.
- ✓ textScaleFactor: Properti ini mengambil nilai double sebagai input untuk menentukan ukuran font relatif.
- ✓ textWidthBasis: TextWidthBasis enum adalah object yang mengatur lebar satu baris teks yang diukur dalam widget.

Tahapan pengerjaan latihan

```
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Home-PC>flutter create latihanappbar
Creating project latihanappbar...
Resolving dependencies in `latihanappbar`... (2.7s)
Downloading packages...
Got dependencies in `latihanappbar`.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd latihanappbar
  $ flutter run

Your application code is in latihanappbar\lib\main.dart.

The configured version of Java detected may conflict with the Android Gradle Plugin (AGP) version in your new Flutter
app.

[RECOMMENDED] If so, to keep the default AGP version 8.1.0, make
sure to download a compatible Java version
(Java 17 <= compatible Java version < Java 21).
You may configure this compatible Java version by running:
`flutter config --jdk-dir=<JDK_DIRECTORY>`
Note that this is a global configuration for Flutter.


Alternatively, to continue using your configured Java version, update the AGP
version specified in the following files to a compatible AGP
version (minimum compatible AGP version: 7.0) as necessary:
    -  C:\Users\Home-PC\latihanappbar\android/build.gradle
```

Latihan penggunaan *Flutter RichText* seperti berikut:

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

// fungsi untuk mentrigger proses build aplikasi flutter
void main() {
  runApp(const MyApp());
}

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

// Widget ini adalah akar dari aplikasi Anda
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'ClipOval',
      theme: ThemeData(
```

```
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePAGE(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class MyHomePAGE extends StatefulWidget {
  const MyHomePAGE({Key? key}) : super(key: key);

  @override
  State createState() => _MyHomePAGEState();
}

class _MyHomePAGEState extends State {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FTI Uniska Tutorial'),
        backgroundColor: Colors.blue,
      ),
      body: Center(
          child: RichText(
        // Mengontrol visual overflow
        overflow: TextOverflow.clip,

        // Mengontrol cara teks harus diselaraskan secara horizontal
        textAlign: TextAlign.end,

        // Mengontrol arah teks
        textDirection: TextDirection.rtl,

        // Apakah teks harus dipisahkan pada soft line breaks
        softWrap: true,

        // Jumlah maksimum baris yang harus dicakup oleh teks
        maxLines: 1,
```

```
        textScaleFactor: 1,
        text: TextSpan(
          text: 'FTI ',
          style: DefaultTextStyle.of(context).style,
          children: const [
            TextSpan(
                text: 'UNISKA', style: TextStyle(fontWeight:
FontWeight.bold)),
          ],
        ),
      )),
      backgroundColor: Colors.lightBlue[50],
    );
  }
}

class MyClip extends CustomClipper {
  @override
  Rect getClip(Size size) {
    return const Rect.fromLTWH(0, 0, 100, 100);
  }

  @override
  bool shouldReclip(oldClipper) {
    return false;
  }
}
```

Hasil flutter RichText :

Menggunakan AppBar pada flutter

AppBar adalah komponen dari aplikasi Flutter, yang biasanya berisi toolbar dan beberapa tombol aktifitas umum. Komponen ini dapat ditempatkan di bagian atas atau bawah aplikasi tergantung pada kebutuhan. Widget AppBar didasarkan pada Material Design dan menyediakan

fungsionalitas secara otomatis. Untuk menentukan di mana konten AppBar harus ditempatkan, kita dapat menggunakan class lain seperti MediaQuery dan Scaffold. Meskipun AppBar sangat fleksibel dan mudah disesuaikan, kita juga dapat menggunakan widget SliverAppBar untuk membuat AppBar yang dapat di-scroll atau membuat AppBar kustom dari awal.

Constructor dari AppBar

```
AppBar(
{Key key,
Widget leading,
bool automaticallyImplyLeading: true,
Widget title,
List actions,
double elevation,
Color shadowColor,
ShapeBorder shape,
Color backgroundColor,
Brightness brightness,
IconThemeData iconTheme,
IconThemeData actionsIconTheme,
TextTheme textTheme,
...}
)
```

Properti dari AppBar

Widget Appbar memiliki beberapa properti, antara lain:

1. **actions**: Properti ini memungkinkan Anda menambahkan daftar widget yang akan ditampilkan setelah judul pada baris Appbar.

2. **title**: Properti ini memungkinkan Anda menambahkan widget utama yang akan ditampilkan di Appbar.
3. **backgroundColor**: Properti ini digunakan untuk menambahkan warna ke latar belakang Appbar.
4. **elevation**: Properti ini digunakan untuk menentukan koordinat-z dari Appbar relatif terhadap induknya.
5. **shape**: Properti ini digunakan untuk memberikan bentuk pada Appbar dan mengatur bayangannya.

Latihan penggunaan *Flutter AppBar* seperti berikut:

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

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

MaterialApp ngtApp() {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.lightGreen,
        title: const Text('FTI UNISKA Tutorial'),
      ),
      body: const Center(
        child: Text(
          'FTI Tutorial',
          style: TextStyle(fontSize: 24),
        ),
      ),
    ),
    debugShowCheckedModeBanner: false,
```

```
  );
}
```

Hasil Flutter AppBar:

Note : Pertama-tama, kita mengimpor file material.dart karena widget AppBar membutuhkannya. Kita juga akan melakukan hal yang sama pada dua contoh berikutnya. Fungsi utama kita yaitu menjalankan runApp.

Di atasnya, terdapat widget MaterialApp yang diikuti oleh Scaffold. Widget MaterialApp menyediakan style untuk AppBar, sedangkan widget Scaffold secara default menempatkan widget AppBar di bagian atas layar. Ini adalah tampilan dasar dari AppBar yang disediakan oleh flutter.

AppBar menggunakan properti title dari class AppBar, yang akan menampilkan widget utama di dalam AppBar. Dalam contoh ini, digunakan widget Teks.

Di dalam bagian body, terdapat sebuah widget center yang memiliki child sebuah widget teks. Widget teks tersebut menampilkan teks "Nextgen Tutorial" dengan ukuran font 24.

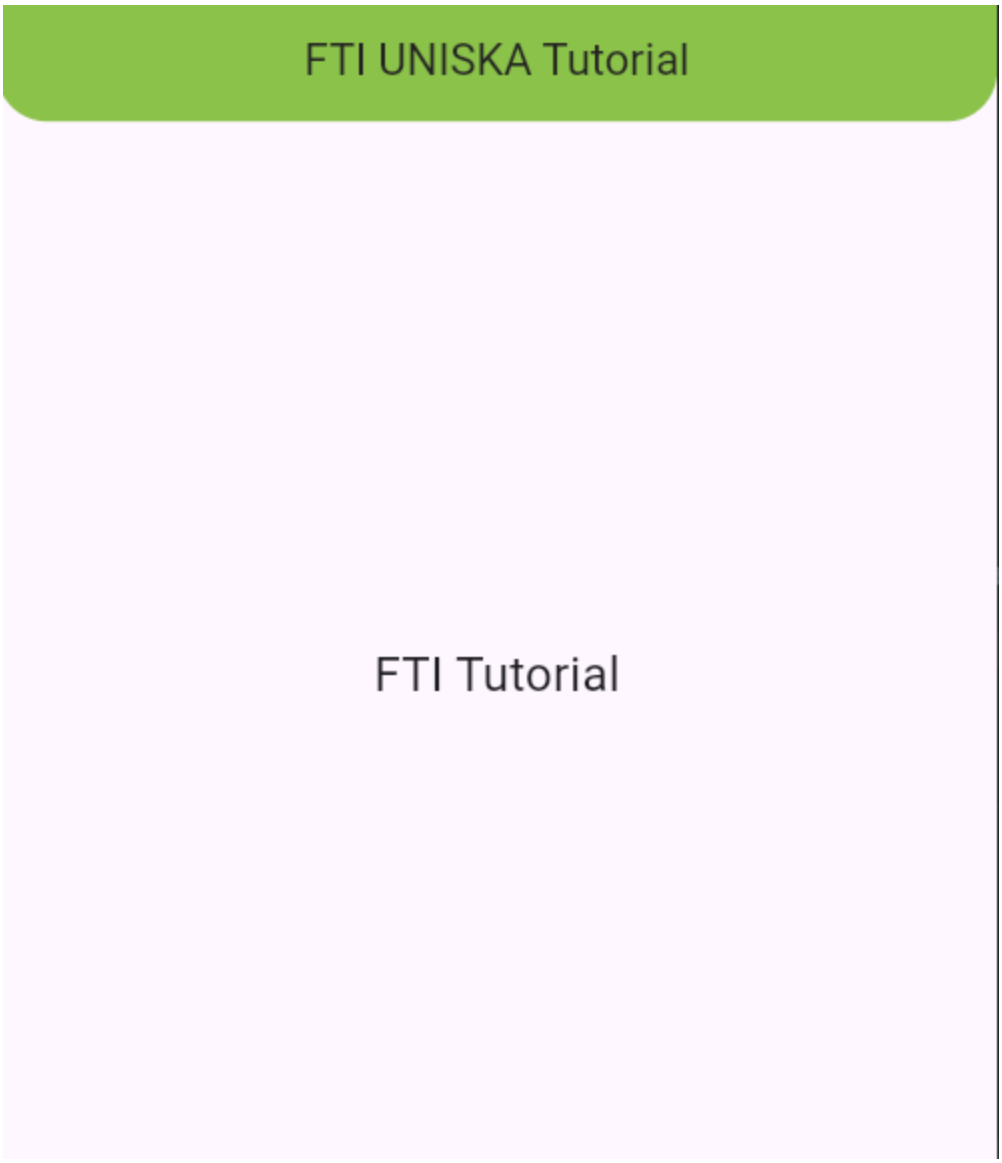
Terakhir, banner debug telah dinonaktifkan. Jadi banner debug merah yang ada di sebelah kanan atas tidak akan tampil.

Latihan penggunaan *Flutter AppBar* 2 seperti berikut:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
```

```
          backgroundColor: Colors.lightGreen,
          title: const Text("FTI UNISKA Tutorial"),
          titleSpacing: 00.0,
          centerTitle: true,
          toolbarHeight: 60.2,
          toolbarOpacity: 0.8,
          shape: const RoundedRectangleBorder(
            borderRadius: BorderRadius.only(
                bottomRight: Radius.circular(25),
                bottomLeft: Radius.circular(25)),
          ),
          elevation: 0.00,
        ),
        body: const Center(
          child: Text(
            'FTI Tutorial',
            style: TextStyle(fontSize: 24),
          ),
        ),
      ),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Hasil :



Note :

Pertama, kita mengimpor file material.dart karena widget AppBar membutuhkannya. Fungsi utama adalah runApp yang akan memulai aplikasi kita. Di atasnya, kita menggunakan widget MaterialApp dan di dalamnya kita menggunakan widget Scaffold. Widget MaterialApp memberikan style untuk AppBar, sedangkan widget Scaffold secara default menempatkan widget AppBar di bagian atas layar. Ini adalah tampilan dasar dari AppBar yang

disediakan oleh Flutter. AppBar hanya menggunakan properti title dari class AppBar, yang mengambil widget utama yang akan ditampilkan di AppBar. Dalam contoh ini, itu adalah widget Teks.

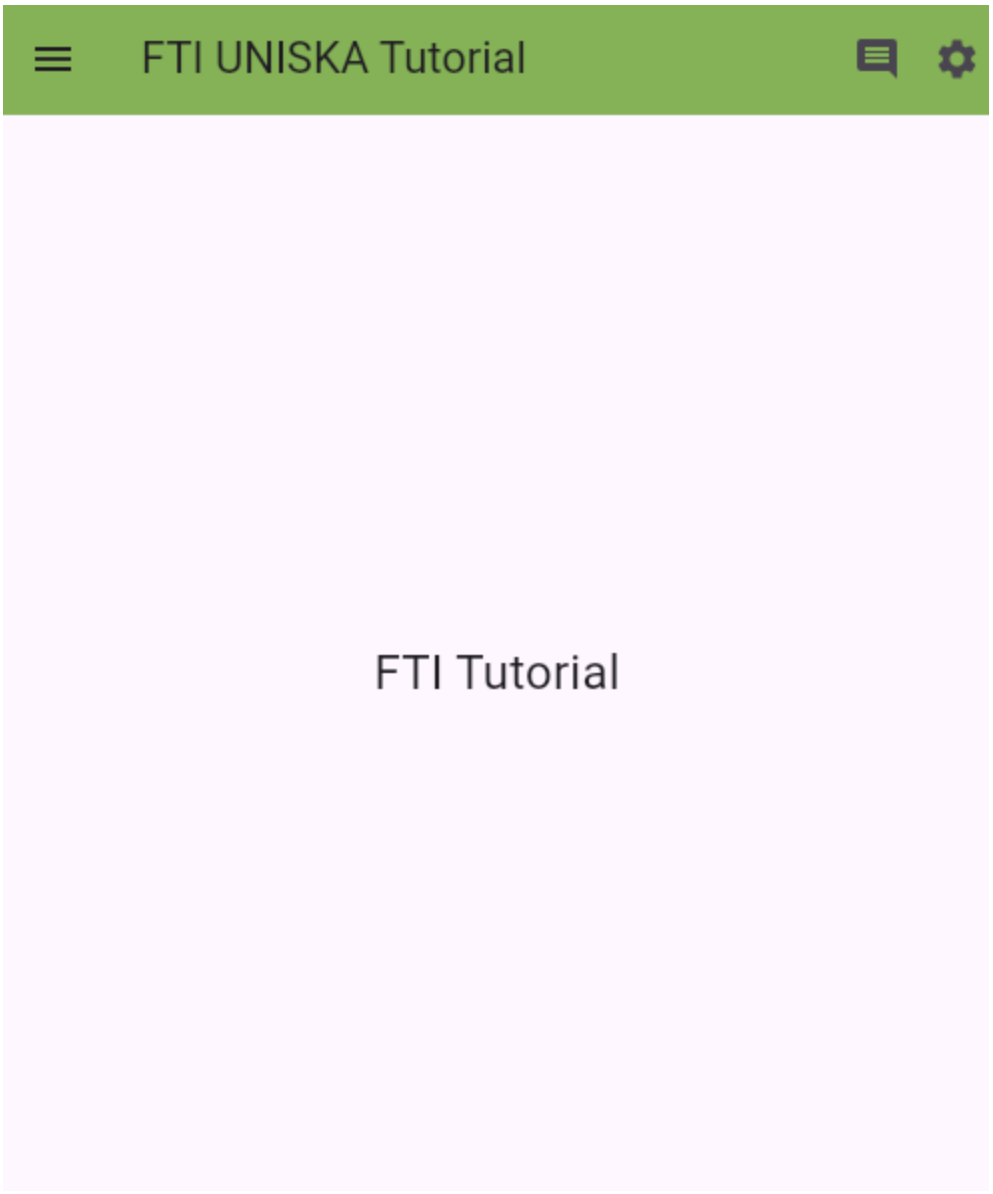
Di dalam body, kita memiliki widget Teks di dalam widget Center yang menampilkan teks "Nextgen Tutorial" dengan ukuran font 24. Kita menonaktifkan banner debug dengan mendefinisikan debugShowCheckedModeBanner menjadi false.

Latihan penggunaan *Flutter AppBar* 3 seperti berikut:

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

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

MaterialApp ngtApp() {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.lightGreen,
        title: const Text("FTI UNISKA Tutorial"),
        actions: [
          IconButton(
            icon: const Icon(Icons.comment),
            tooltip: 'Comment Icon',
            onPressed: () {},
          ),
          IconButton(
            icon: const Icon(Icons.settings),
            tooltip: 'Setting Icon',
            onPressed: () {},
```

```
          ),
        ],
        elevation: 50.0,
        leading: IconButton(
          icon: const Icon(Icons.menu),
          tooltip: 'Menu Icon',
          onPressed: () {},
        ),
        systemOverlayStyle: SystemUiOverlayStyle.light,
      ),
      body: const Center(
        child: Text(
          'FTI Tutorial',
          style: TextStyle(fontSize: 24),
        ),
      ),
    ),
    debugShowCheckedModeBanner: false,
  );
}
```

Hasil :



Note :

Di sini, kita bisa melihat bahwa AppBar memiliki tiga ikon tambahan selain title: satu di sebelah kiri dan dua di sebelah kanan title. Widget AppBar ini dimulai dengan properti title yang menggunakan widget Teks sebagai parameter. Kemudian, dilanjutkan dengan properti actions yang mengambil daftar widget sebagai parameter untuk ditampilkan setelah title. Dalam kasus

ini, terdapat dua ikon yang dapat dilihat, yaitu ikon komentar dan pengaturan. Kedua ikon ini adalah widget IconButton yang memiliki tiga properti, yaitu icon, tooltip, dan fungsi onPressed. Fungsi onPressed tidak didefinisikan pada IconButton sehingga belum ada efeknya. Properti icon menggunakan string sebagai parameter yang merupakan nama icon. Properti tooltip juga menggunakan string sebagai parameter yang ditampilkan dalam label mengambang saat kursor mouse diarahkan atau saat mouse ditekan lama. Pada IconButton pertama, kita menggunakan Icons.comment dan string 'Comment Icon' sebagai parameter properti tooltip. Sedangkan pada IconButton kedua, kita menggunakan Icons.setting dan string 'Setting Icon' sebagai parameter properti tooltip. Setelah itu, properti leading ditetapkan dengan widget IconButton sebagai parameter untuk ditampilkan sebelum title di AppBar.

Selamat kita sudah berhasil belajar basic flutter

Next Sesion

Menambahkan Tab pada Flutter