PANDUAN TUGAS 3

Membuat Tampilan SplashScreen dengan Animasi

Capaian

1. Mahasiswa dapat mengimplementasikan komponen UI dasar untuk multimedia seperti Container widget dengan warna.
2. Mahasiswa mampu menggunakan dan menampilkan asset lokal berformat svg dalam proyek flutter dengan memanfaatkan package flutter_svg.
3. Mahasiswa dapat mengimplementasikan animasi sederhana menggunakan Animated Container Widget.

Spesifikasi Hardware dan Software

Memiliki komponen perangkat keras dan perangkat lunak yang benar sangat penting untuk memastikan keberhasilan pelaksanaan tugas yang diuraikan dalam panduan ini. Konfigurasi perangkat keras dan perangkat lunak yang diperlukan untuk menyelesaikan tugas panduan ini adalah sebagai berikut:

A. Spesifikasi Minimum Hardware
   1. Minimum RAM 4 GB, disarankan RAM 8 GB
   2. Minimum 15 GB ruang disk yang tersedia (2 GB untuk Flutter SDK, 8 GB untuk Android Studio, 4 GB untuk AVD, dan 1 GB untuk proyek)
   3. Resolusi layar minimum 1280 x 800
   4. Emulator Mobile (Android/IOS)

B. Software
   1. Flutter SDK (versi 3.13.0 atau yang lebih baru) dan Dart (versi 3.1.5 atau yang lebih baru)
   2. Android Studio
   3. Visual Studio Code

Sumber Daya

1. Test File Tugas 1

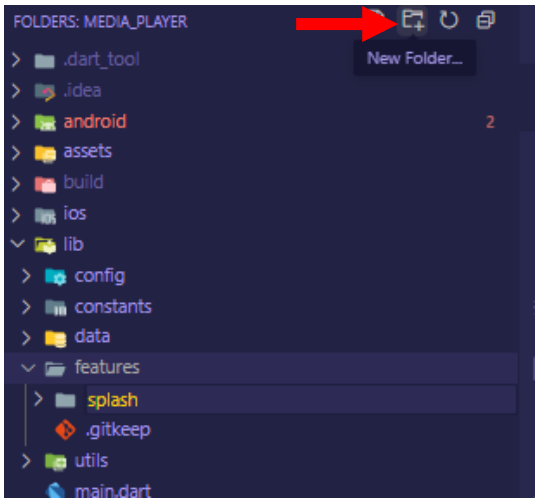
Deskripsi Tugas

Mahasiswa akan menuliskan kode pada proyek Flutter media_player dalam mengimplementasikan komponen UI dasar multimedia. Mahasiswa akan membuat komponen CircleComponent dan menyusun tampilan SplashScreen. Kemudian mahasiswa akan menerapkan animasi sederhana pada tampilan SplashScreen menggunakan AnimatedOpacity widget.

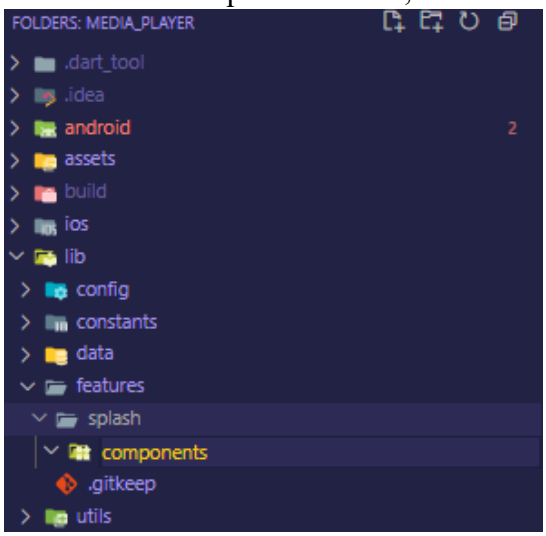
Langkah Praktikum 1 (Membuat Komponen CircleComponent)

1. Buka proyek flutter media_player pada Visual Studio Code anda.

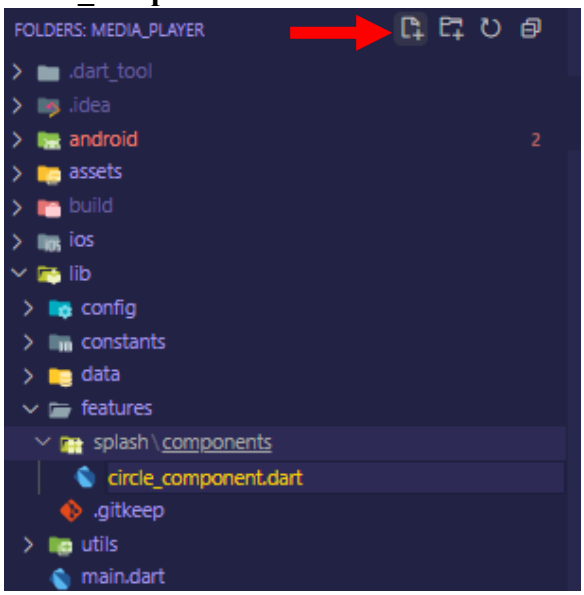
2. Buka folder lib>features dan tambahkan folder **splash** di dalamnya.



3. Di dalam folder splash tersebut, buat folder baru lagi dengan nama **components**.



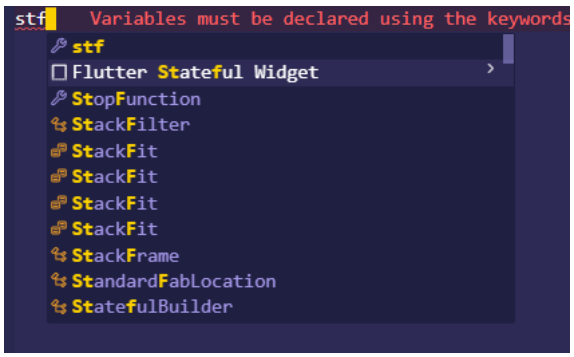
4. Buka/Pilih folder components tersebut. Kemudian buat file baru dengan nama **circle_component.dart**.

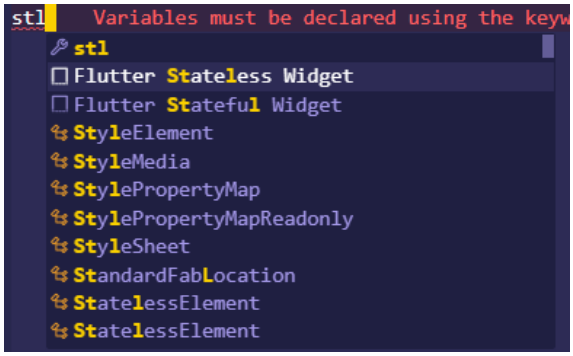


5. Buat sebuah stateless widget dengan nama **CircleComponent** di dalam file circle_component.dart.

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

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

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

*Jika anda telah menginstall plugin flutter pada Visual Studio Code, anda dapat memanfaatkan fitur code completion. Hanya dengan mengetikan stf/stl anda dapat dengan mudah membuat stateful/stateless widget.





*Jika code completion tidak mau muncul pada Visual Studio padahal anda telah memasang plugin Fluter, coba tekan tombol CTRL + SPACE. Shortcut tersebut digunakan untuk menampilkan bantuan code completion.

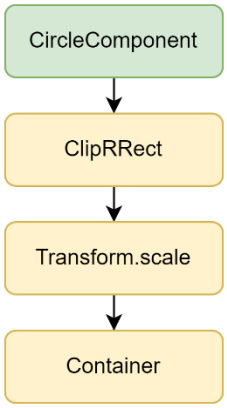
6. Tambahkan properti scale, color, dan child pada CircleComponent.

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

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

  final double scale;
  final Color? color;
  final Widget? child;

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

7. Tambahkan ketiga properti tersebut ke dalam konstruktor CircleComponent sebagai parameternya.

```
const CircleComponent({
  super.key,
  required this.scale,
  this.color,
  this.child,
});
```

8. Ubah widget kembalian dari CircleComponent menjadi seperti berikut.




```
Widget build(BuildContext context) {
  return ClipRRect(
    child: Transform.scale(
      child: Container(),
    ),
  );
}
```

9. Ubah properti dari setiap widget dalam CircleComponent tersebut sesuai dengan spesifikasi berikut.

Widget	Properti	Nilai
ClipRRect	child	Transform.scale()
Transform.scale	scale	scale
	child	Container()
Container	decoration	BoxDecoration(shape: BoxShape.circle, color: color,)

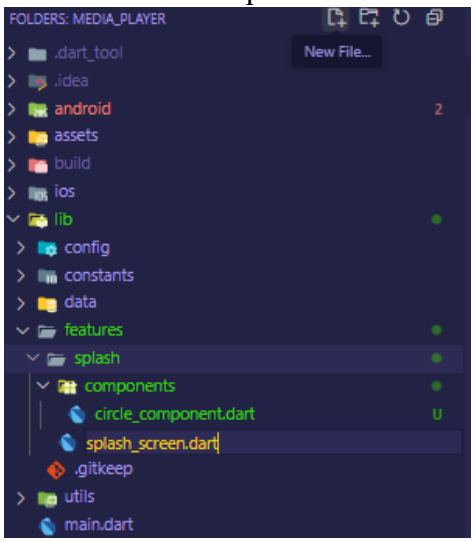
10. Pada Transform.scale() widget, tambahkan operator null-aware `??` pada argumen child-nya agar bisa menampilkan widget yang nantinya diberikan melalui child argumen saat pembuatan/pemanggilan CircleComponent. Jika child bernilai null, maka tampilkan Container() widget.

11. Berikut adalah kode akhir dari file circle_component.dart

```dart
class CircleComponent extends StatelessWidget {
  const CircleComponent({
    super.key,
    required this.scale,
    this.color,
    this.child,
  });
  final double scale;
  final Color? color;
  final Widget? child;

  @override
  Widget build(BuildContext context) {
    return ClipRRect(
      child: Transform.scale(
        scale: scale,
        child: child ??
            Container(
              decoration: BoxDecoration(
                shape: BoxShape.circle,
                color: color,
              ), // BoxDecoration
            ), // Container
      ), // Transform.scale
    ); // ClipRRect
  }
}
```

Langkah Praktikum 2 (Membuat Tampilan SplashScreen)

1. Buka/Pilih folder splash. Kemudian buat file baru dengan nama **splash_screen.dart**.

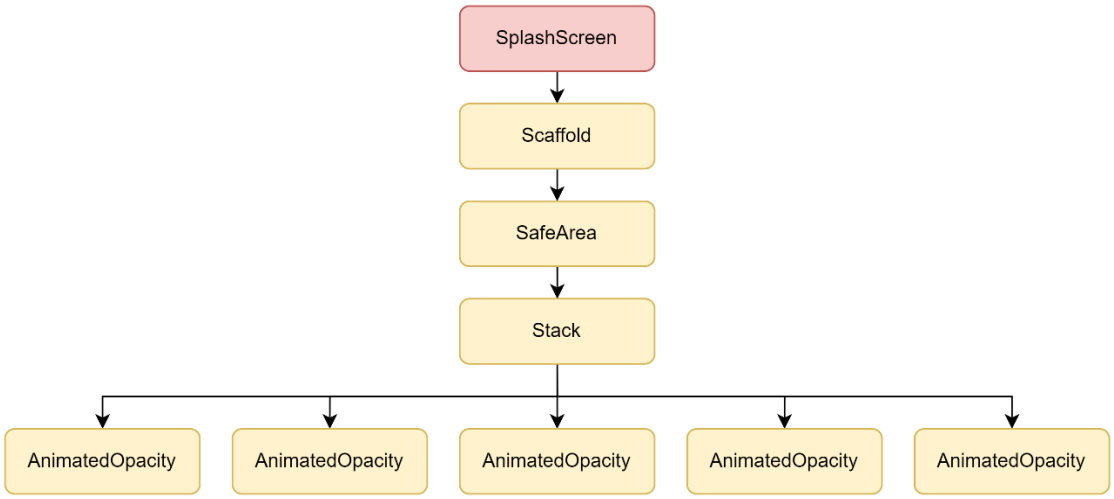


2. Buat sebuah stateful widget dengan nama **SplashScreen** di dalam file splash_screen.dart.

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

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

  @override
```

```
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

3. Ubah widget kembalian dari SplashScreen menjadi seperti berikut.




```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
          child: Stack(
        children: [
          AnimatedOpacity(),
          AnimatedOpacity(),
          AnimatedOpacity(),
          AnimatedOpacity(),
          AnimatedOpacity(),
        ],
      )),
    );
```

4. Ubah properti dari setiap widget dalam SplashScreen tersebut sesuai dengan spesifikasi berikut.

Widget	Properti	Nilai
Scaffold	body	SafeArea()
SafeArea	child	Stack()
Stack	alignment	Alignment.center
	children	AnimatedOpacity() x 5
AnimatedOpacity [semua]	opacity	1
	duration	const Duration(milliseconds: 300)
	child	CircleComponent()

5. Tambahkan kode pada setiap CircleComponent yang ada sesuai dengan spesifikasi berikut.

Widget	Anak Animated Opacity ke	Properti	Nilai
CircleComponent	1	key	Key('circle_1')
		scale	3
		color	MainColor.grey989794
	2	key	Key('circle_2')
		scale	1.7
		color	MainColor.greyB6B5B1
	3	key	Key('circle_3')
		scale	1.3
		color	MainColor.greyD4D2CE
	4	key	Key('circle_4')
		scale	0.8
		color	MainColor.black222222
	5	key	Key('circle_5')
		scale	0.8
		child	SvgPicture.asset()

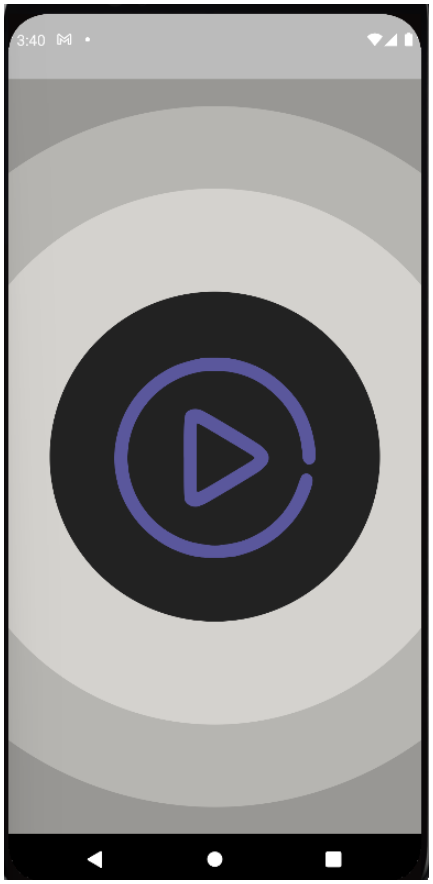
6. SvgPicture.asset digunakan untuk menampilkan lokal asset dalam format svg. Pada widget tersebut, ubah propertinya sesuai dengan spesifikasi kode berikut.

Widget	Properti	Nilai
SvgPicture.asset	assetName	AssetsConsts.logo
	colorFilter	const ColorFilter.mode(MainColor.purple5A579C, BlendMode.srcIn,)

7. Buka file **main_pages.dart** yang ada di dalam folder lib>config>pages. Kemudian ubah widget function pada MainRoute.splash menjadi SplashScreen().

```
You, 10 seconds ago | 1 author (You)
import 'package:flutter/material.dart';
import 'package:media_player/config/routes/main_routes.dart';
import 'package:media_player/features/splash/splash_screen.dart';

/// A Map that associates each route defined in the [MainRoute] class w
/// Each route is associated with a corresponding widget for the view a
Map<String, Widget Function(BuildContext)> mainPages = {
  MainRoute.splash: (context) => const SplashScreen(),      You, 10 sec
  MainRoute.home: (context) => const Placeholder(),
```

8. Coba jalankan proyek anda pada emulator dengan menekan tombol run yang ada pada fungsi main di **main.dart**. Berikut adalah tampilan emulator jika anda telah berhasil menyusun tampilan SplashScreen.



Langkah Praktikum 3 (Implementasi animasi pada AnimatedOpacity)

1. Buka file **splash_screen.dart**. Tambahkan dua state berikut ke dalam _SplashScreenState.

```
  final Duration _animatedDuration = const Duration(milliseconds: 300);
  final List<bool> _visibilityList = [false, false, false, false];
```

2. Tambahkan juga fungsi startAnimation yang berfungsi untuk menjalankan animasi dengan interval waktu _animatedDuration secara periodik untuk semua CircleComponent() dari yang pertama hingga terakhir.

```
_startAnimation(int index) {
    if (index < _visibilityList.length) {
      Timer(_animatedDuration, () {
        setState(() {
          _visibilityList[index] = true;
        });

        if (index == _visibilityList.length - 1) {
        } else {
          _startAnimation(index + 1);
        }
      });
    }
  }
```

3. Agar fungsi _startAnimation() dipanggil tepat ketika SplashScreen() dirender (dibuat), tambahkan juga override fungsi initState() dengan memanggil _startAnimation().

```dart
@override
void initState() {
  super.initState();
  _startAnimation(0);
}
```

4. Ubah properti semua AnimatedOpacity pada SplashScreen sesuai dengan spesifikasi berikut.

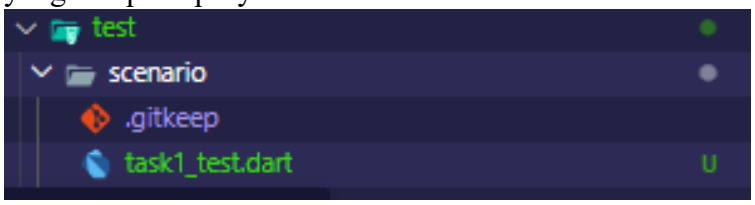
Widget	Anak ke	Properti	Nilai
AnimatedOpacity	1	opacity	_visibilityList[0] ? 1 : 0
	2	opacity	_visibilityList[1] ? 1 : 0
	3	opacity	_visibilityList[2] ? 1 : 0
	4	opacity	_visibilityList[3] ? 1 : 0
	5	pacity	_visibilityList[3] ? 1 : 0

5. Ubah juga properti duration dari semua AnimatedOpacity pada SplashScreen menjadi `_animatedDuration`.

6. Coba jalankan proyek anda pada emulator dengan menekan tombol run yang ada pada fungsi main di main.dart. Berikut adalah tampilan emulator ketika anda berhasil membuat SplashScreen dengan benar. Lihat tampilan.

Langkah Verifikasi Kode

1. Unduh test file pada tautan Test File Tugas 1. Letakkan file ke dalam folder test>scenario yang ada pada proyek.

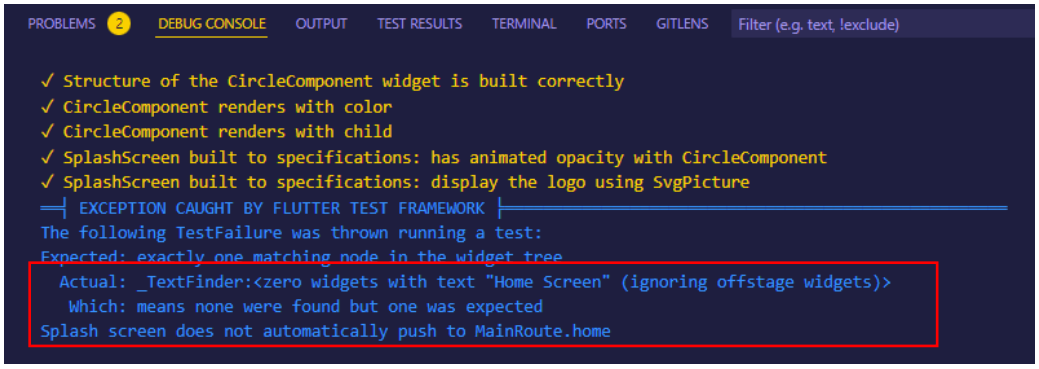


2. Buka file **task1_test.dart**.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:media_player/config/themes/main_color.dart';
import 'package:media_player/constants/assets_const.dart';
import 'package:media_player/features/splash/components/circle_component.dart';
import 'package:media_player/features/splash/splash_screen.dart';

Run | Debug
void main() {
  Run | Debug
  testWidgets('Structure of the CircleComponent widget is built correctly',
```

3. Tekan tombol Run yang ada pada fungsi main dan tunggu proses hingga selesai.

4. Jika anda mengalami error, coba lihat pesan yang diberikan.



5. “Splash screen does not automatically push to MainRoute.home”. Maksud dari pesan error tersebut adalah splash screen seharusnya dapat secara otomatis menuju/membuka halaman MainRoute.home ketika animasi selesai dijalankan. Sehingga saat anda mengalami error yang serupa, tambahkan kode berikut pada fungsi _startAnimation() blok kode if.

```
Future.delayed(_animatedDuration * 2, () {
        Navigator.pop(context);
        Navigator.pushNamed(context, MainRoute.home);
});
```

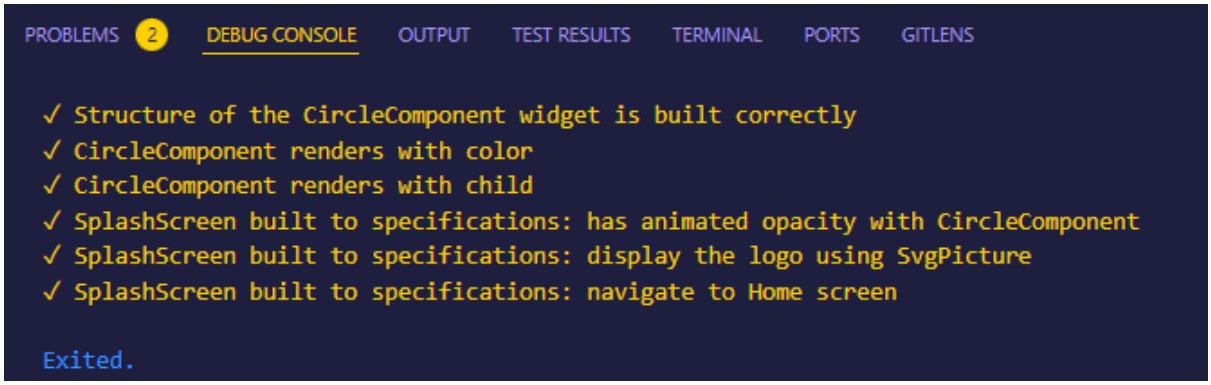
*Jangan lupa tambahkan import yang diperlukan

```
import 'package:media_player/config/themes/main_color.dart';
import 'package:media_player/config/routes/main_routes.dart';
import 'package:media_player/constants/assets_const.dart';
import 'package:media_player/features/splash/components/circle_compone

class SplashScreen extends StatefulWidget {···

class _SplashScreenState extends State<SplashScreen> {
  final Duration _animatedDuration = const Duration(milliseconds: 300)
  final List<bool> _visibilityList = [false, false, false, false];

  @override
  void initState() {···

  _startAnimation(int index) {
    if (index < _visibilityList.length) {
      Timer(_animatedDuration, () {
        setState(() {···

        if (index == _visibilityList.length - 1) {
          Future.delayed(_animatedDuration * 2, () {
            Navigator.pop(context);
            Navigator.pushNamed(context, MainRoute.home);
          }); // Future.delayed
        } else {
```

Kode di atas akan membuat halaman splash screen secara otomatis menutup tampilannya saat ini (SplashScreen) dan membuka tampilan baru yang dikembalikan oleh MainRoute.home.

6. Jalankan kembali test file dengan menekan tombol Run yang ada pada fungsi main dan tunggu proses hingga selesai.

7. Jika testing berhasil, maka akan muncul hasil berikut pada tab Debug Console Visual Studio Code anda.



Anda juga dapat melihat hasil testing pada tab Test Result.