

PANDUAN TUGAS 1

Persiapan dan Pengaturan Proyek

Capaian

1. Mahasiswa mampu mempersiapkan environment dan membuat project flutter baru.

Dokumen ini bertujuan untuk menjadi dokumen pengantar untuk persiapan materi pembelajaran Pemrograman Flutter dengan Aplikasi Multimedia. Mahasiswa diharapkan mampu menyiapkan environment yang dibutuhkan untuk mempelajari cara membuat proyek Flutter baru.

Pengenalan Proyek Aplikasi Multimedia Flutter

1. Multimedia di Flutter

Multimedia merupakan kombinasi dari beberapa elemen seperti teks, suara, gambar dan animasi tertentu sehingga pengguna dapat bernavigasi, berinteraksi atau berkomunikasi di dalam aplikasi. Dalam konteks pengembangan mobile menggunakan Flutter, multimedia mengacu pada penggabungan berbagai elemen media, seperti teks, audio, gambar, dan video ke dalam aplikasi mobile yang dibuat dengan Flutter. Hal ini memungkinkan pengembang untuk menciptakan pengalaman pengguna yang menarik dan interaktif.

Integrasi multimedia di Flutter dapat dilakukan dengan memanfaatkan berbagai package dan library yang tersedia di pub.dev, termasuk `google_fonts`, `flutter_svg`, `audioplayers`, `video_player`, dan `cached_network_image`. Dengan memanfaatkan pustaka-pustaka tersebut pengembang Flutter dapat dengan mudah menambahkan fungsionalitas multimedia yang kaya dan beragam ke dalam aplikasi.

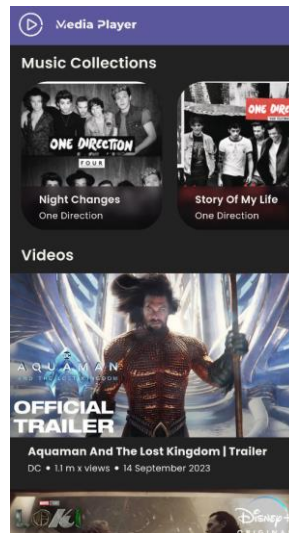
2. Proyek Flutter Media Player

Proyek yang akan anda buat dalam serangkaian modul ini adalah proyek flutter **Media Player**. Dalam proyek ini, Anda akan membangun aplikasi multimedia yang memungkinkan pengguna untuk:

- a. Menampilkan Teks
 - Menampilkan teks statis atau teks yang dinamis dari sumber eksternal.
 - Mengubah gaya font, ukuran font, ketebalan font, dan warna teks.
 - Mengatur spasi antar baris dan paragraf.
- b. Menampilkan Gambar
 - Menampilkan gambar yang disimpan secara lokal di dalam proyek atau memuat gambar dari internet.
- c. Memutar Audio
 - Memutar file audio MP3 yang tersimpan secara lokal di dalam proyek atau dari internet.
 - Mengontrol pemutaran dengan tombol play/pause dan seekbar.
 - Menampilkan informasi seperti judul lagu, artis, dan durasi audio.
- d. Menampilkan Video
 - Memutar file video MP4 yang tersimpan secara lokal di dalam proyek atau streaming video dari internet.

- Mengontrol pemutaran dengan tombol play/pause dan seekbar.
- Menampilkan informasi seperti judul video, durasi video, dan deskripsi video.

Aplikasi ini akan dirancang dengan antarmuka yang menarik dan mudah digunakan, dengan memanfaatkan berbagai elemen multimedia untuk memberikan pengalaman pengguna yang interaktif. Di bawah ini adalah salah satu tampilan yang akan ada dalam aplikasi Media Player. Anda dapat melihat keseluruhan aplikasinya melalui video yang ada pada [tautan ini](#).



3. Kode dan Package Pihak Ketiga

Untuk membangun aplikasi Media Player ini, Anda akan menggunakan bahasa pemrograman Dart dan framework Flutter. Berikut adalah beberapa contoh kode dan package pihak ketiga yang akan Anda gunakan:

a. Basic Multimedia UI Components

- Text widget: Digunakan untuk menampilkan teks statis atau teks yang dinamis.

```
Text(
  'Judul Lagu',
  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
)
```

- Image widget: Digunakan untuk menampilkan gambar yang disimpan secara lokal atau dimuat dari internet.

```
Image.asset('assets/images/gambar.jpg')
```

b. Simple animations

- AnimatedOpacity widget: Digunakan untuk membuat animasi perubahan opacity elemen.

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

class AnimatedOpacityExample extends StatefulWidget {
```

```

@override
_AnimatedOpacityExampleState createState() =>
_AnimatedOpacityExampleState();
}

class _AnimatedOpacityExampleState extends
State<AnimatedOpacityExample> {
  bool _isVisible = true;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('AnimatedOpacity Example'),
      ),
      body: Center(
        child: AnimatedOpacity(
          opacity: _isVisible ? 1.0 : 0.0,
          duration: const Duration(seconds: 1),
          child: Container(
            width: 200,
            height: 200,
            color: Colors.blue,
          ),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            _isVisible = !_isVisible;
          });
        },
        child: const Icon(Icons.play_arrow),
      ),
    );
  }
}

```

Penjelasan:

- AnimatedOpacity widget: Menyelimuti **Container** biru dengan **opacity** yang dikontrol oleh nilai **_isVisible**.
- Duration(seconds: 1): Mengatur durasi animasi menjadi 1 detik.
- _isVisible: Variabel boolean yang menentukan apakah **Container** terlihat atau tidak.
- FloatingActionButton: Digunakan untuk beralih visibilitas **Container** dengan menekan tombol.

Manfaat:

- Efek animasi yang lebih halus: **AnimatedOpacity** memberikan efek animasi yang lebih halus dibandingkan menggunakan **AnimationController** dan **Animation** secara langsung.

- Kode yang lebih ringkas: `AnimatedOpacity` tidak memerlukan kode boilerplate sebanyak menggunakan `AnimationController` dan `Animation`.
- Mudah digunakan: `AnimatedOpacity` mudah digunakan dan dipahami, terutama untuk animasi sederhana.

Catatan:

- `AnimatedOpacity` hanya dapat digunakan untuk mengubah opacity elemen. Untuk animasi yang lebih kompleks, seperti perubahan posisi atau ukuran, Anda masih dapat menggunakan `AnimationController` dan `Animation`.

c. Third-party Multimedia Packages

- `google_fonts`: Digunakan untuk mengintegrasikan font dari Google Fonts ke dalam aplikasi.

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

Text(
  'Judul Lagu',
  style: GoogleFonts.lato(fontSize: 20, fontWeight: FontWeight.bold),
)
```

- `flutter_svg`: Digunakan untuk menampilkan gambar vektor SVG.

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

SvgPicture.asset('assets/images/icon.svg')
```

- `audioplayers`: Digunakan untuk memutar file audio.

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

AudioPlayer audioPlayer = AudioPlayer();
await audioPlayer.play('assets/audio/lagu.mp3');
```

- `video_player`: Digunakan untuk memutar file video.

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

VideoPlayerController controller =
  VideoPlayerController.asset('assets/videos/video.mp4');

await controller.initialize();
controller.play();
```

- `cached_network_image`: Digunakan untuk memuat gambar dari internet dengan optimasi cache.

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

CachedNetworkImage(
  imageUrl: 'https://www.example.com/image.jpg',
  placeholder: (context, url) => CircularProgressIndicator(),
  errorWidget: (context, url, error) => Icon(Icons.error),
)
```

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
4. Git

Sumber Daya

1. [Panduan instalasi Flutter SDK](#)
2. [Panduan instalasi Android Studio](#)
3. [Visual Studio Code](#)
4. [Git](#)
5. [Panduan membuat akun GitHub](#)

Deskripsi Tugas

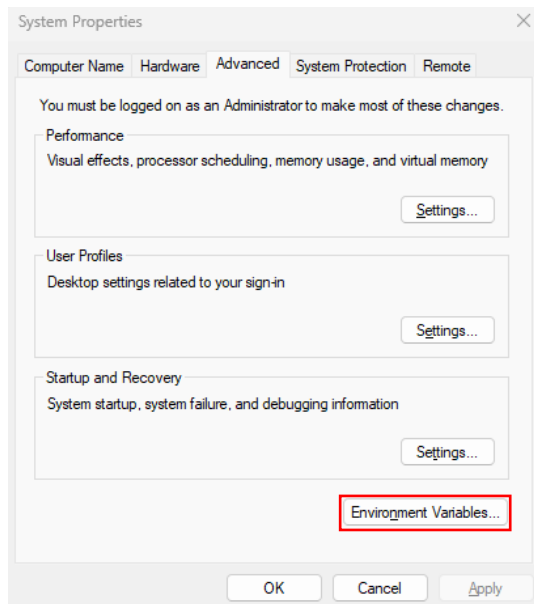
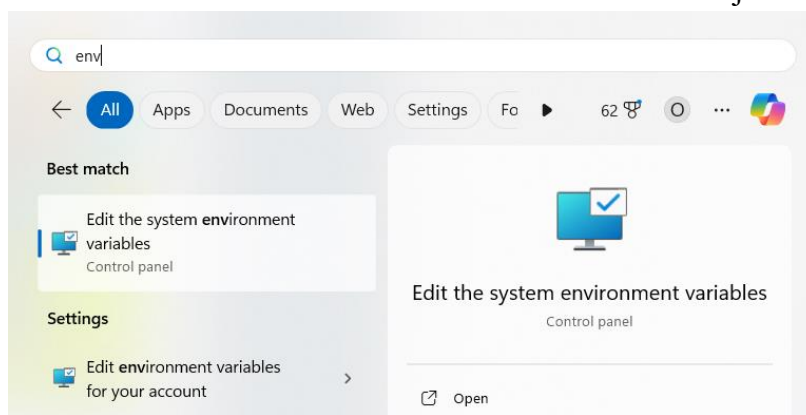
Mahasiswa dapat menginstal proyek Flutter baru. Mahasiswa akan menginstal perangkat lunak yang diperlukan untuk menyelesaikan tugas ini seperti Flutter SDK, Android Studio, VSCode, dan Git. Setelah instalasi selesai, mahasiswa akan membuat proyek Flutter baru dan mencoba menjalankan testing bawaan ketika pertama kali membuat proyek Flutter baru.

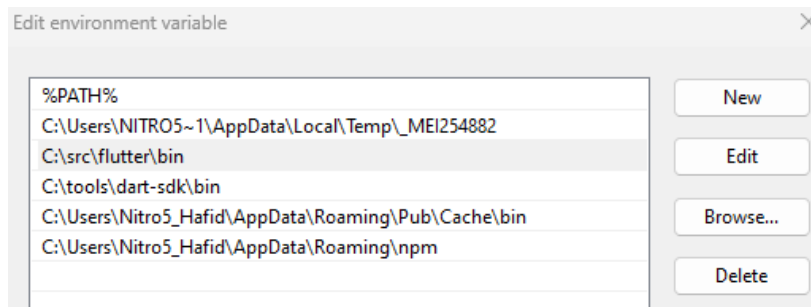
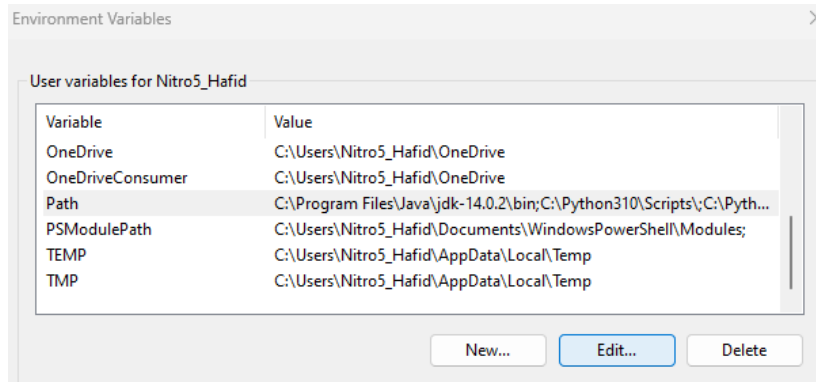
*Setiap langkah-langkah yang diberikan dalam panduan ini dan kedepannya, penulis menggunakan perangkat Windows 11 dan emulator Android. Jadi anda bisa menyesuaikan dengan spesifikasi perangkat yang anda gunakan.

Instalasi Software

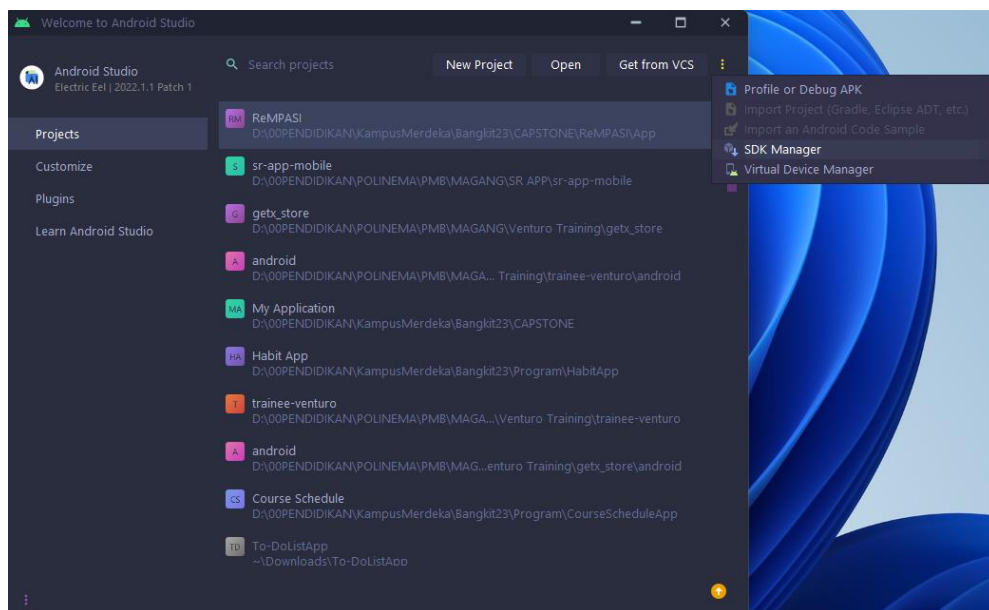
Jika mahasiswa sudah menginstal semua Software yang diperlukan di sistem operasi masing- masing, anda bisa melewati bagian ini.

1. Download Flutter SDK versi 3.13.0 atau lebih melalui link berikut ini: [Flutter SDK](#)
2. Extract file yang telah didownload ke harddisk anda contoh lokasi ke C:\src\flutter.
3. Tambahkan Flutter ke PATH environment variabel menuju folder C:\src\flutter\bin.

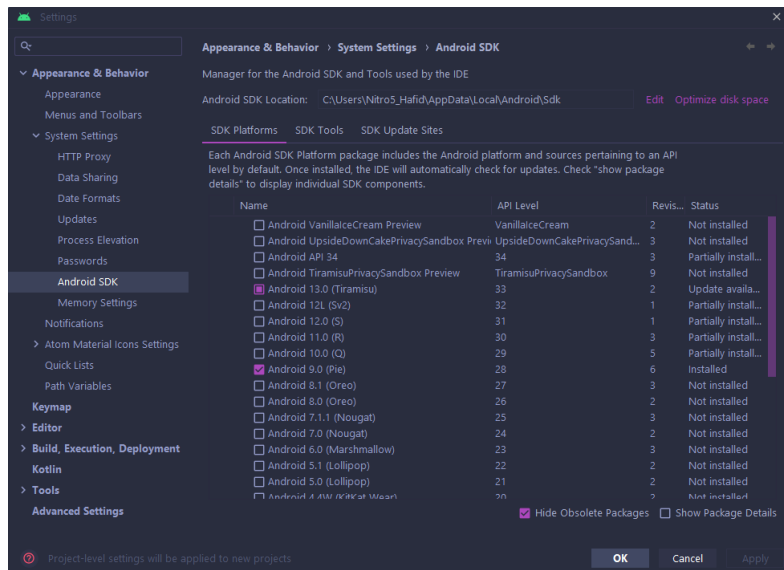




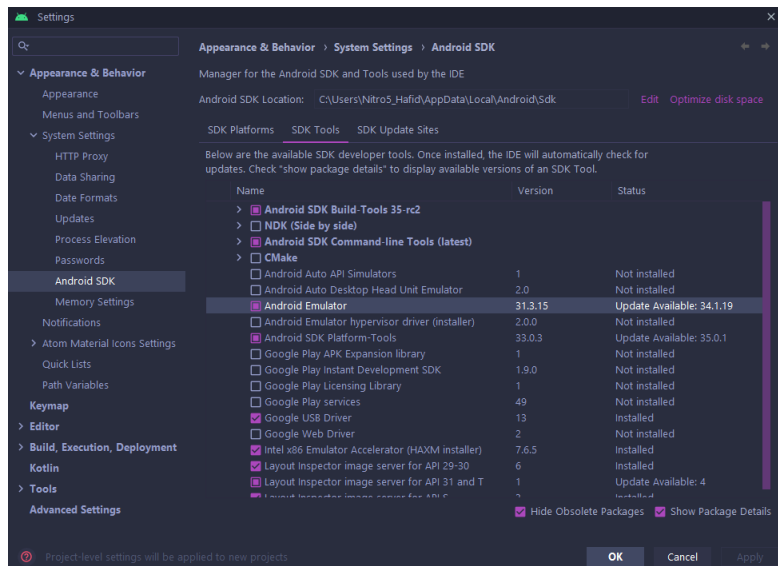
4. Download dan ikuti panduan instalasi Android Studio melalui link berikut ini: [Android Studio](#)
5. Install beberapa komponen di bawah ini pada Android Studio anda
 - Android SDK Platform, (sesuaikan versinya dengan emulator android anda)
 - Android SDK Command-line Tools
 - Android SDK Build-Tools
 - Android SDK Platform-Tools
 - Android Emulator (opsional)



Buka menu SDK Manager.



Pilih Android SDK yang sesuai dengan versi android emulator anda. Kemudian tekan apply dan tunggu proses download hingga selesai.



Anda bisa mencentang Android Emulator jika anda tidak memiliki android device (real device) sebagai emulator android. Kemudian tekan apply dan tunggu proses download hingga selesai.

6. Download Git melalui link berikut ini: [Git](#)
7. Lakukan proses instalasi. Ketika Anda telah berhasil menjalankan penginstal, Anda akan melihat layar wizard Pengaturan Git. Ikuti petunjuk Berikutnya dan Selesai untuk menyelesaikan instalasi. Opsi default cukup masuk akal bagi sebagian besar pengguna.
8. Buka Command Prompt (atau Git Bash jika selama instalasi Anda memilih untuk tidak menggunakan Git dari Command Prompt Windows).

9. Jalankan perintah berikut untuk mengonfigurasi nama pengguna dan email Git Anda, ganti nama dan email Emma dengan nama dan email Anda sendiri (email ini nantinya akan digunakan untuk membuat akun Github). Detail ini akan dikaitkan dengan komit apa pun yang Anda buat:

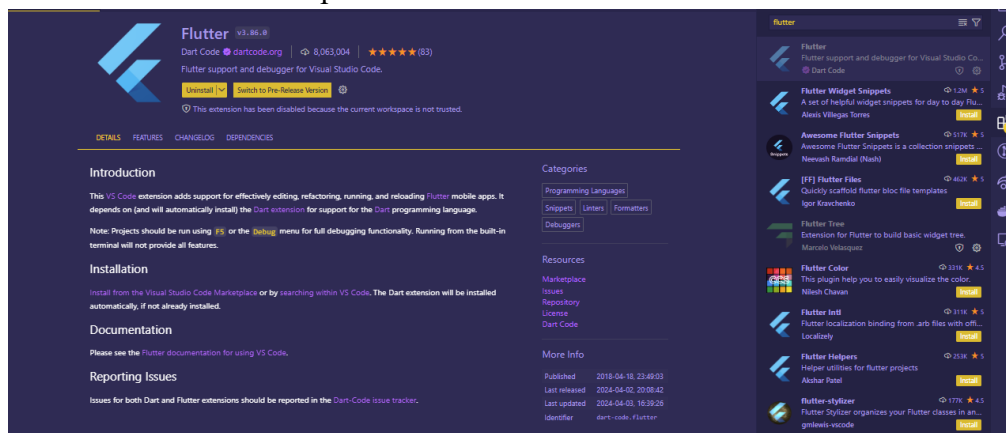
```
$ git config --global user.name "Emma Paris"
```

```
$ git config --global user.email eparis@atlassian.com
```

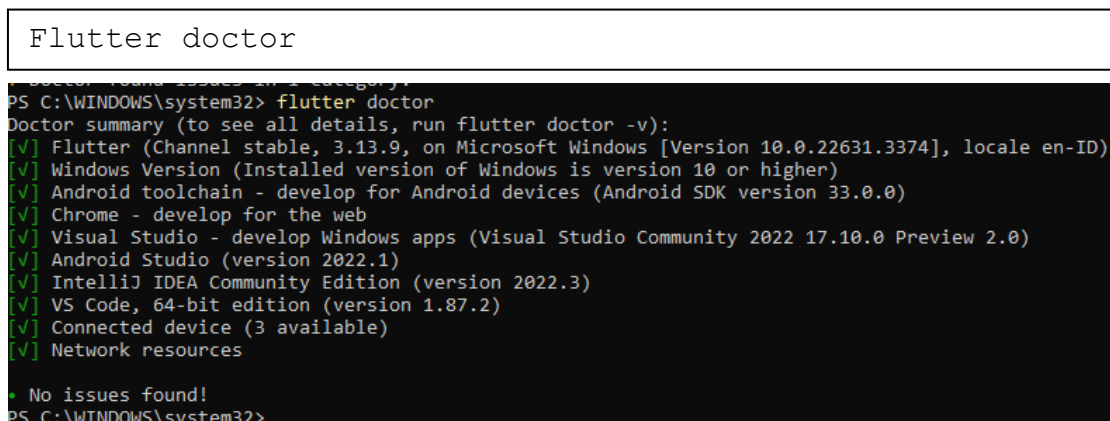
10. Buat sebuah akun Github dengan mengikuti langkah-langkah yang ada pada tautan berikut ini: [Panduan membuat akun GitHub](#).

11. Download dan ikuti panduan instalasi Visual Studio Code melalui link berikut ini: [Visual Studio Code](#)

12. Install extension Flutter pada Visual Studio Code anda



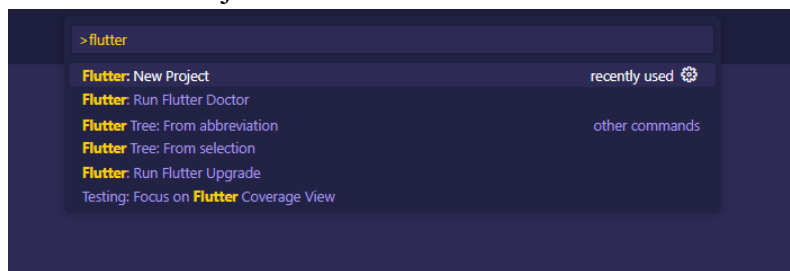
13. Lakukan validasi instalasi Flutter dengan mengetikkan perintah berikut di terminal anda



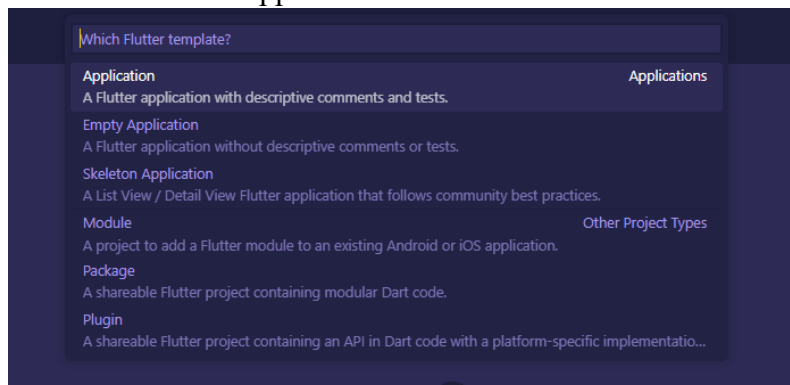
14. Jika perintah flutter doctor mengeluarkan hasil seperti gambar di atas, maka anda telah siap untuk membuat proyek flutter pertama anda.

Langkah Praktikum

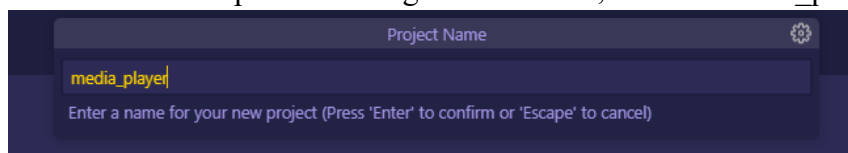
1. Buka Visual Studio Code, lalu tekan tombol CTRL + Shift + P dan ketikkan perintah Flutter: New Project.



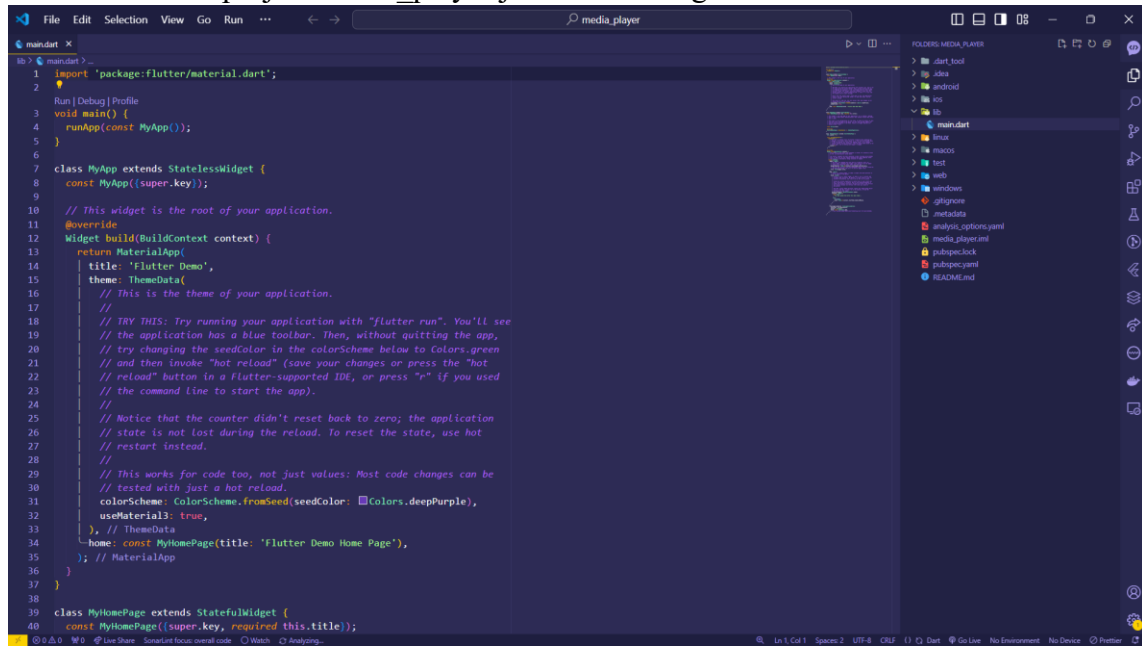
2. Pilih Flutter New Application



3. Pilih folder tempat project akan dibuat.
4. Berilah nama project yang sesuai dengan peraturan penamaan project di Flutter. Gunakan huruf kecil dan dipisahkan dengan underscore, contoh: media_player



5. Berikut ini hasil project media_player jika berhasil digenerate.

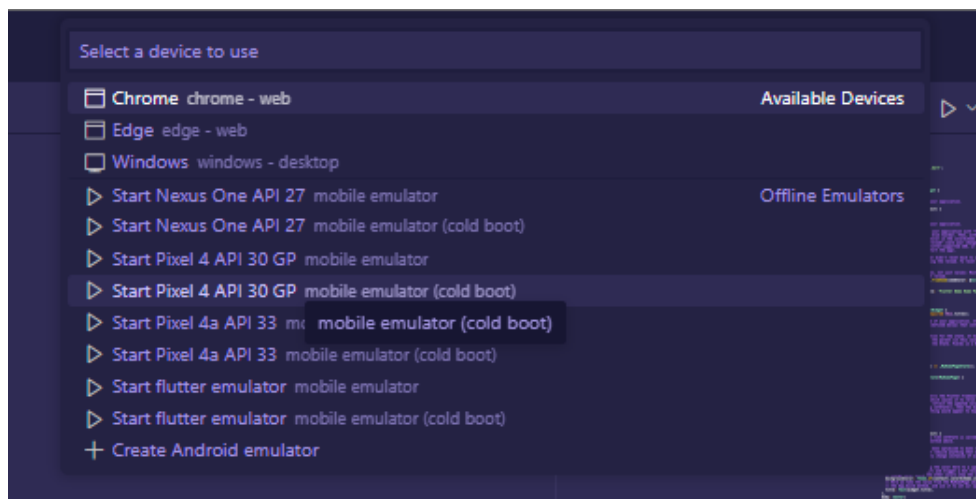


```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15      theme: ThemeData(
16        // This is the theme of your application.
17        //
18        // TRY THIS: Try running your application with "flutter run". You'll see
19        // the application has a blue toolbar. Then, without quitting the app,
20        // try changing the seedColor in the colorScheme below to Colors.green
21        // and then invoke "hot reload" (save your changes or press the "hot
22        // reload" button in a Flutter-supported IDE, or press "r" if you used
23        // the command line to start the app).
24        //
25        // Notice that the counter didn't reset back to zero; the application
26        // state is not lost during the reload. To reset the state, use hot
27        // restart instead.
28        //
29        // This works for code too, not just values: Most code changes can be
30        // tested with just a hot reload.
31        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
32        useMaterial3: true,
33      ), ThemeData(
34        home: const MyHomePage(title: 'Flutter Demo Home Page'),
35      ); // MaterialApp
36    );
37  }
38
39  class MyHomePage extends StatefulWidget {
40    const MyHomePage({super.key, required this.title});
```

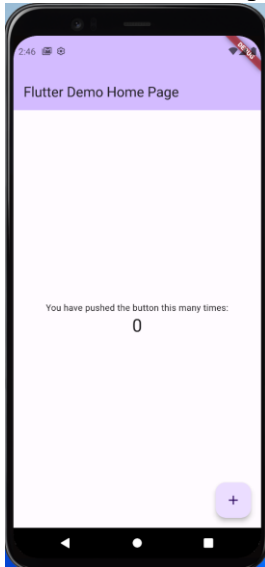
6. Untuk menjalankan project pastikan anda sudah melakukan setup emulator / setup device sesuai dengan perintah pada praktikum sebelumnya.

7. Jika emulator / device sudah disetup tekan tombol F5 untuk menjalankan aplikasi keemulator / device.

8. Pilih emulator atau device dan klik enter.



9. Berikut adalah tampilan emulator jika anda berhasil menjalankan project flutter pertama.



Langkah Verifikasi Kode

1. Testing pekerjaan Anda dengan membuka file `widget_test.dart` yang ada dalam folder `test`.

A screenshot of an IDE (IntelliJ) showing the `widget_test.dart` file in the `test` folder. The code is as follows:

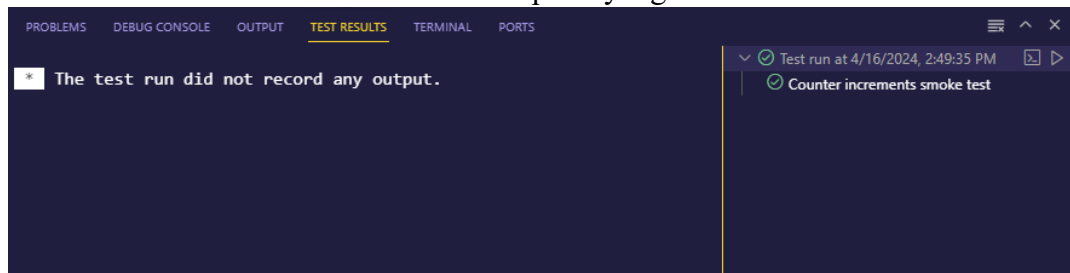
```
1 // This is a basic Flutter widget test.
2 //
3 // To perform an interaction with a widget in your test, use the WidgetTester
4 // utility in the flutter_test package. For example, you can send tap and scr
5 // gestures. You can also use WidgetTester to find child widgets in the widget
6 // tree, read text, and verify that the values of widget properties are correct
7
8 import 'package:flutter/material.dart';
9 import 'package:flutter_test/flutter_test.dart';
10
11 import 'package:media_player/main.dart';
12
13 void main() {
14   testWidgets('Counter increments smoke test', (WidgetTester tester) async {
15     // Build our app and trigger a frame.
16     await tester.pumpWidget(const MyApp());
17
18     // Verify that our counter starts at 0.
19
```

2. Jalankan kode dengan menekan tombol Run yang ada di atas fungsi `main`. Tunggu proses hingga selesai.

A screenshot of the same IDE showing the `widget_test.dart` file. The code is now more complete, including the `expect` statements and the `tap` action. A red arrow points to the `Run | Debug` button above the `void main()` function. The code is as follows:

```
19   expect(find.text('0'), findsOneWidget);
20   expect(find.text('1'), findsNothing);
21
22   // Tap the '+' icon and trigger a frame.
23   await tester.tap(find.byIcon(Icons.add));
24   await tester.pump();
25
```

3. Buka tab Test Result. Berikut adalah tampilan yang akan muncul ketika test berhasil.



4. Anda juga bisa menjalankan test dengan menuliskan perintah `flutter test test/widget_test.dart` pada terminal.

