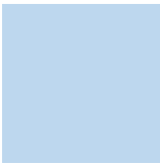


**MODUL**

**PEMROGRAMAN BERBASIS PLATFORM**



PERTEMUAN – 2

# Widget & Layout I

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA





## Daftar Gambar

Gambar 1. Contoh Kode penggunaan Theme .....	7
Gambar 2. Contoh Kode penggunaan style .....	7
Gambar 3. Contoh Kode penggunaan Scaffold .....	8
Gambar 4. Hasil Kode dari gambar 3 .....	9
Gambar 5. Contoh kode menggunakan gambar dari internet .....	9
Gambar 6. Hasil Kode gambar 5 .....	10
Gambar 7. Contoh kode menggunakan gambar dari lokal .....	10
Gambar 8. Contoh kode menggunakan icon .....	11
Gambar 9. Hasil kode gambar 8 .....	11
Gambar 10. Deklarasi folder images sebagai assets .....	12
Gambar 11. Contoh penggunaan syntax Center .....	13
Gambar 12. Hasil kode jika tidak menggunakan Center .....	13
Gambar 13. Contoh penggunaan padding .....	14
Gambar 14. Hasil kode gambar 13 .....	15
Gambar 15. Hasil jika padding diperbesar .....	15
Gambar 16. Contoh penggunaan margin .....	16
Gambar 17. Hasil kode gambar 16 .....	17
Gambar 18. Cara deklarasi font di pubspec .....	18
Gambar 19. Buat folder di folder utama .....	18
Gambar 20. Contoh dengan Anton-Regular.ttf .....	19
Gambar 21. Contoh penggunaan font yang sudah ditambahkan .....	20
Gambar 22. Contoh penggunaan column .....	21
Gambar 23. Hasil kode gambar 22 .....	22
Gambar 24. Contoh penggunaan row .....	23
Gambar 25. Hasil kode gambar 23 .....	24
Gambar 26. Contoh penggunaan Navigator.push .....	25
Gambar 27. Contoh penggunaan Navigator.pop .....	25
Gambar 28. Contoh penggunaan Navigator.pushReplacement .....	26
Gambar 29. Visual Studio Code .....	27
Gambar 30. Pilih opsi untuk memulai flutter project .....	28
Gambar 31. Pilih opsi Application .....	28
Gambar 32. Lokasi penyimpanan <i>flutter project</i> .....	29
Gambar 33. Menamai <i>project flutter</i> .....	29
Gambar 34. Isi <i>project template flutter</i> .....	30
Gambar 35. Letak main.dart .....	30
Gambar 36. Buka terminal baru di visual studio code .....	31
Gambar 37. Jalankan perintah flutter run di terminal .....	31
Gambar 38. Tampilan awal template flutter .....	32



Gambar 39. Menambahkan file dan folder di folder lib .....	33
Gambar 40. Menambahkan folder images dan fonts.....	33
Gambar 41. Mengambil salah satu font dari google font.....	34
Gambar 42. Copy file ini .....	34
Gambar 43. Penamaan file gambar.....	34
Gambar 44. struktur utama .....	35
Gambar 45. Isi file pubspec.yaml .....	36
Gambar 46. Isi file main.dart .....	38
Gambar 47. Isi file elementLinkTree.dart.....	39
Gambar 48. Isi file showProfile.dart .....	41
Gambar 49. isi file isiLogo.dart .....	42
Gambar 50. isi file isiShowProfile.dart.....	43
Gambar 51. isi file isiLinkTree.dart .....	44
Gambar 52. isi file directToLink.dart .....	45
Gambar 53. isi file constant.dart .....	46
Gambar 54. hasil 1 .....	47
Gambar 55. hasil 2 .....	48



## TUJUAN

---

**Setelah menyelesaikan modul ini, praktikan diharapkan mampu :**

1. Memahami konsep dasar dari penggunaan theme, style dan color dalam pengembangan aplikasi mobile dengan flutter.
2. Mampu menambahkan serta menampilkan gambar dan ikon, juga mampu untuk mengakses gambar dan icon dari penyimpanan lokal atau dari internet.
3. Mengetahui kegunaan Hot Reload dan Hot Restart.
4. Memahami penggunaan Widget seperti Center, Padding, dan Margin yang digunakan untuk mengatur tampilan antarmuka.
5. Mampu menggunakan dan menambahkan font di flutter.
6. Memahami kegunaan dan penggunaan Column dan Row untuk mengatur letak komponen UI secara vertikal maupun horizontal.
7. Mampu menggunakan Navigator untuk berpindah antar halaman.

Dengan menyelesaikan modul ini diharapkan praktikan memahami beberapa widget dasar, *layouting* dan juga konsep dasar lainnya dalam pembuatan tampilan antarmuka yang sederhana dengan *flutter*.



## DASAR TEORI

---

### A. APA ITU FLUTTER?

*Flutter* merupakan *framework* yang sifatnya terbuka dan dikembangkan oleh *google* dengan tujuan membangun tampilan antarmuka (UI) yang baik dan indah dalam aplikasi *mobile*, web dan desktop. *Flutter* dibangun dengan dasar bahasa pemrograman *dart* yang efisien dan kuat, sehingga memungkinkan membuat tampilan antarmuka yang menarik dan responsif. Dalam pembuatan tampilan antarmuka *flutter* menggunakan *widget* yang kemudian disusun sedemikian rupa sehingga menjadi tampilan yang diinginkan atau diharapkan. Pembuatan tampilan antarmuka dengan *flutter* mirip seperti menyusun blok lego, yang disusun dan ditata agar menjadi bentuk yang diharapkan. Apa itu *widget*? Akan dibahas di poin berikut.

### B. WIDGET

Dalam proses pembangunan antarmuka dengan *flutter*, ada elemen kecil yang kemudian disusun, diatur dan dikombinasikan untuk membentuk tampilan dan interaksi yang diharapkan. *Widget* dapat berupa tombol, teks, gambar, kotak input dan berbagai elemen lain yang membentuk komponen tampilan di dalam aplikasi.

Pada *Flutter* setiap elemen-elemen seperti tombol, teks, gambar dan lainnya dinamakan *widget*. *Widget* dapat memiliki properti dan juga metode yang memungkinkan perubahan terjadi, contohnya perubahan tampilan *widget*, interaksi ataupun logika tertentu.

*Flutter* menyediakan berbagai jenis *widget* yang dapat digunakan untuk membangun tampilan yang interaktif, responsif, dan juga menarik. Pengguna *flutter* juga dapat membuat *custom widget* sesuai dengan kebutuhan aplikasi mereka. *Custom widget* ini juga dapat digunakan berulang kali layaknya prosedur dan fungsi yang



sudah pernah dipelajari di mata kuliah sebelum mata kuliah ini, sehingga penggunaannya lebih efisien.

Kesimpulannya widget pada *flutter* adalah elemen yang membangun tampilan antarmuka dan memiliki peranan penting untuk membangun tampilan yang menarik dan lebih responsif dalam *framework flutter* ini.

### C. LAYOUT

*Layout* mengacu pada tata letak atau susunan dari elemen-elemen antarmuka dalam sebuah aplikasi. Dalam pengembangan aplikasi menggunakan teknologi seperti *flutter*, android, ataupun web terdapat berbagai teknik untuk mengatur susunan elemen mereka. Contohnya seperti XML dalam android dan CSS dalam web. Tujuannya tetap sama yaitu untuk mencapai tampilan antarmuka yang diharapkan dan memberikan pengalaman pengguna (UX) yang baik. Dalam *flutter* terdapat beberapa widget yang dapat digunakan untuk mengatur susunan widget-widget lainnya sehingga tampilan menjadi lebih baik. Diantaranya: *Container*, *Row*, *Column*, *Stack*, *Expanded*, *GridView*, *ListView*, *sizedBox*, *SingleChildScrollView*, dan lainnya. Semua akan dibahas seiring dengan berjalannya mata kuliah ini.

### D. WIDGET DAN LAYOUT DALAM MODUL INI

#### 1. THEME, STYLE, COLOR

*Theme* adalah sebuah tema yang sudah diatur secara *default* di *flutter* yang dapat digunakan untuk mempermudah dalam mengatur aspek-aspek tampilan seperti warna, tipografi, dekorasi dan lainnya. Dengan penggunaan *theme*, tidak perlu pusing untuk mengatur warna pada setiap elemen. Misalnya *theme* yang digunakan adalah *dark* maka warna dari widget akan mengikuti *theme* tersebut. Dalam *theme* ada juga komponen yang bisa di *custom* seperti *primaryColor*, *accentColor*, *fontFamily*, dan sebagainya. Kesimpulannya *theme* digunakan untuk

mempermudah untuk mencapai konsistensi visual di aplikasi yang sedang dibangun.

Contoh kode :

```
theme: ThemeData(  
  primarySwatch: Colors.blue,  
),
```

Gambar 1. Contoh Kode penggunaan Theme

*Style* seperti namanya *style* digunakan untuk memperbaharui tampilan antarmuka, seperti warna, bentuk, dan lainnya. Contohnya *TextStyle* yang digunakan untuk mengubah tampilan dari tulisan baik itu *fontStyle*, *fontColor*, *fontZise* ataupun *alignment*.

Contoh kode :

```
Text(  
  'FLUTTER DEVELOPER',  
  style: TextStyle(  
    fontFamily: 'Source Sans Pro',  
    fontSize: 20,  
    color: Colors.teal.shade100,  
    letterSpacing: 2.5,  
    fontWeight: FontWeight.bold,  
  ), // TextStyle  
) // Text
```

Gambar 2. Contoh Kode penggunaan style

*Color* sama seperti *style* kegunaan dari *color* adalah untuk memperbaharui atau memodifikasi tampilan antarmuka. Di *flutter* ada beberapa cara untuk mengatur warna, diantaranya: warna primitif yang langsung dikodekan dengan nama dari warna tersebut contohnya *Colors.red*. yang kedua adalah palet warna, contohnya *Colors.teal*. dan yang terakhir adalah warna dengan heksadesimal yang dikodekan dengan mengambil nilai

heksadesimal dari warna tersebut contohnya `Color(0xFF00FF00)` yang akan menghasilkan warna hijau.

## 2. SCAFFOLDING

*Scaffolding* adalah widget yang digunakan sebagai kerangka dasar untuk penyusunan widget antarmuka lainnya. Sederhananya *scaffolding* merupakan kanvas lukis yang digunakan untuk melukis elemen sesuai keinginan pelukisnya.

Contoh kode :

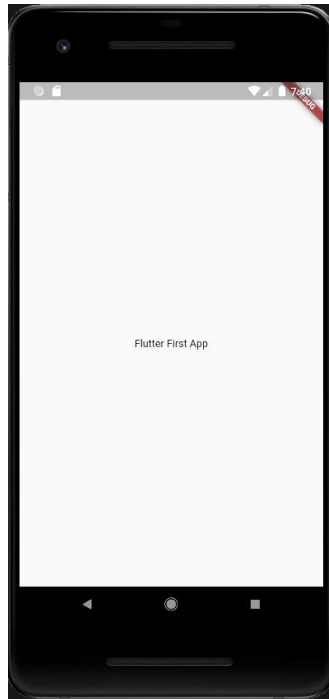


Gambar 3. Contoh Kode penggunaan Scaffold

Dari kode ini bisa diperhatikan *Scaffold* menampung widget seperti *SafeArea*, *Center*, dan juga *Text*.

Hasil:





Gambar 4. Hasil Kode dari gambar 3

### 3. PICTURE DAN ICON

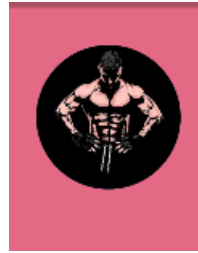
Dalam *flutter* juga bisa menambahkan gambar dan ikon. Gambar bisa diambil dari *source* dari internet ataupun *local storage*, begitu pula ikon. Untuk *picture* yang diakses melalui internet gunakan sintaks *NetworkImage*.

Contoh kode :

```
1 CircleAvatar(  
2     radius: 50,  
3     backgroundImage: NetworkImage(  
4         'https://cdn.statusqueen.com/dpimages/thumbnail/gym_boy_dp_image-3172.jpg'),  
5     ),
```

Gambar 5. Contoh kode menggunakan gambar dari internet

Hasil :



Gambar 6. Hasil Kode gambar 5

Jika ingin menampilkan gambar dari penyimpanan lokal bisa menggunakan sintaks dibawah ini :

```
1 CircleAvatar(  
2     radius: 50,  
3     backgroundImage: AssetImage('images/download.jpg'),  
4 )
```

Gambar 7. Contoh kode menggunakan gambar dari lokal

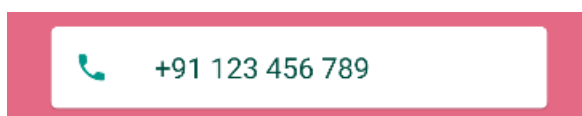
*AssetImage* dapat digunakan untuk menampilkan gambar yang terdapat di *path* yang dituju. Contoh dalam kode di atas akan menampilkan gambar dengan nama "download.jpg" di dalam file images.

Mirip dengan menampilkan gambar, ikon juga bisa ditampilkan dengan cara yang hampir mirip. Contoh kode :



Gambar 8. Contoh kode menggunakan icon

Sintaks yang digunakan adalah icon, dengan kode di atas akan didapatkan hasil sebagai berikut :



Gambar 9. Hasil kode gambar 8

Dapat dilihat *icon: Icons.phone* akan menghasilkan ikon telepon seperti gambar di atas.

#### 4. ASSETS DAN IMAGE

Dapat Dalam *flutter* ada file yang bernama *pubspec* yang digunakan untuk mengelola sumber daya yang akan digunakan selama membangun aplikasi. Jika ingin menggunakan gambar di penyimpanan lokal maka nama file tempat menyimpan gambar ataupun gambar tersebut harus di deklarasikan terlebih dahulu di file *pubspec* ini. Perlu diperhatikan penulisan kode di file ini, karena file ini berfokus pada indentasi untuk membaca alur programnya.

Cara deklarasi file *image* di *pubspec*



Gambar 10. Deklarasi folder images sebagai assets

Cara Perhatikan indentasinya, karena jika indentasi salah maka file yang dituju tidak akan bisa diakses.

## 5. WRAP WIDGET (CENTER, PADDING, MARGIN)

Cara *Center* seperti namanya digunakan untuk membuat *widget* yang ada di dalam sintaks *center* berada di tengah.

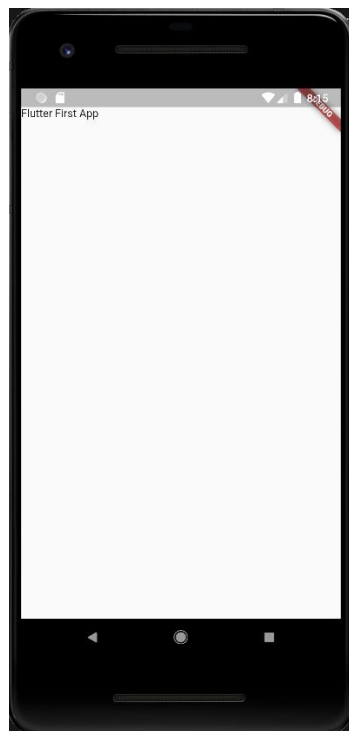
Contoh kode :



Gambar 11. Contoh penggunaan sintax Center

Hasilnya akan sama seperti gambar 4.

Jika tidak menggunakan *Center* maka akan menghasilkan :



Gambar 12. Hasil kode jika tidak menggunakan Center

*Padding* digunakan untuk menambahkan ruang kosong di sekitar *widget* yang dapat digunakan untuk mengatur jarak antar elemen-elemen UI di aplikasi.

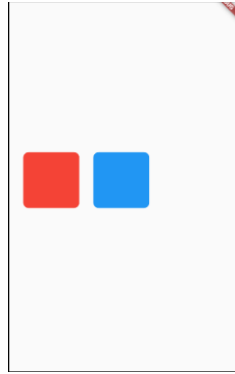
Contoh kode :



```
1  return Scaffold(  
2    body: Center(  
3      child: Row(  
4        children: [  
5          Padding(  
6            padding: EdgeInsets.all(25.0),  
7            child: Container(  
8              width: 100,  
9              height: 100,  
10             decoration: BoxDecoration(  
11               color: Colors.red,  
12               borderRadius: BorderRadius.circular(10),  
13             ),  
14           )),  
15          Container(  
16            width: 100,  
17            height: 100,  
18            decoration: BoxDecoration(  
19              color: Colors.blue,  
20              borderRadius: BorderRadius.circular(10),  
21            ),  
22          ),  
23        ],  
24      ),  
25    ),  
26  );
```

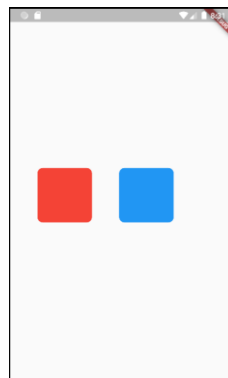
Gambar 13. Contoh penggunaan padding

Hasil :



Gambar 14. Hasil kode gambar 13

jika diperbesar sampai 50 maka hasilnya sebagai berikut:



Gambar 15. Hasil jika padding diperbesar

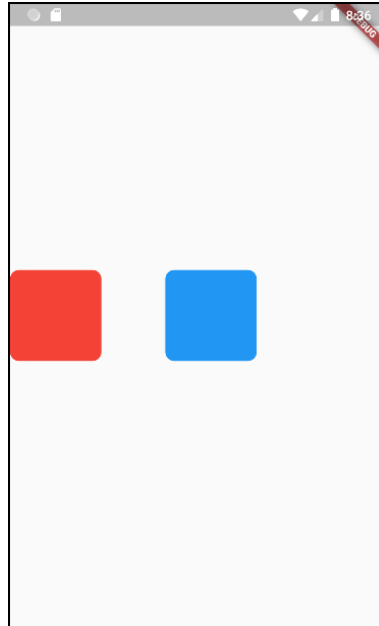
Margin kegunaannya hampir mirip dengan padding yaitu memberikan ruang kosong untuk mengatur jarak antar elemen.  
Contoh kode :



```
1  return Scaffold(  
2    body: Center(  
3      child: Row(  
4        children: [  
5          Container(  
6            width: 100,  
7            height: 100,  
8            decoration: BoxDecoration(  
9              color: Colors.red,  
10             borderRadius: BorderRadius.circular(10),  
11           ),  
12         ),  
13         Container(  
14           margin: EdgeInsets.only(left: 70.0),  
15           width: 100,  
16           height: 100,  
17           decoration: BoxDecoration(  
18             color: Colors.blue,  
19             borderRadius: BorderRadius.circular(10),  
20           ),  
21         ),  
22       ],  
23     ),  
24   ),  
25 );
```

Gambar 16. Contoh penggunaan margin





Gambar 17. Hasil kode gambar 16

Kode diatas mendorong container berwarna biru sejauh 70 dengan sintaks `"margin: EdgeInsets.only(left:70.0)"`.

## 6. HOT RELOAD DAN HOT RESTART

*Hot Reload* digunakan untuk melihat perubahan yang dilakukan pada kode tanpa harus memulai aplikasi dari awal. Sementara itu *hot restart* akan memulai kembali aplikasi dari awal.

## 7. FONT

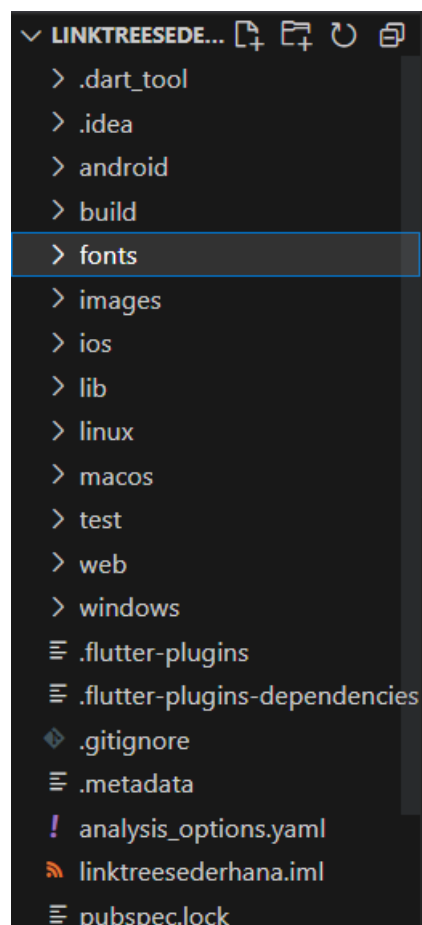
*Flutter* juga mengizinkan penambahan *font* diluar *font* bawaan dari *flutter* sendiri. Mirip seperti pendeklarasian *images*, *font* juga harus dideklarasikan di dalam *pubspec*.

Perhatikan kode dibawah ini untuk deklarasi penggunaan *font* di file *pubspec*



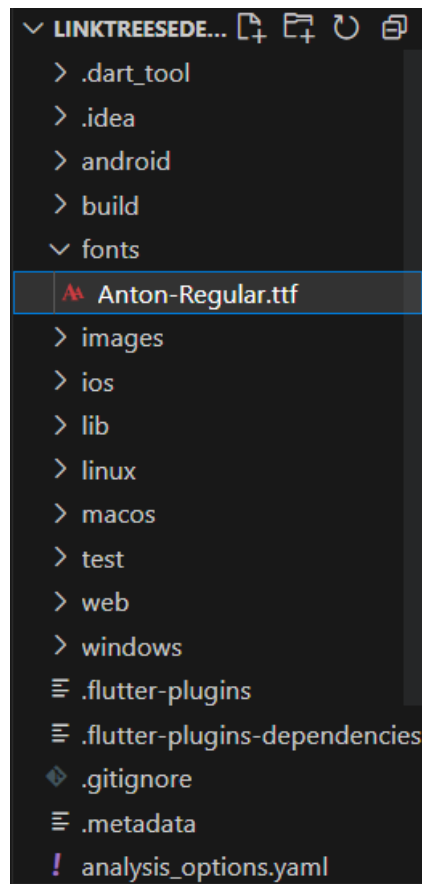
Gambar 18. Cara deklarasi font di pubspec

Sebelum itu buat folder bernama *fonts* :



Gambar 19. Buat folder di folder utama

Lalu masukkan folder font yang diinginkan (untuk referensi bisa gunakan google font), download dan masukkan file dengan ekstensi .ttf seperti berikut



Gambar 20. Contoh dengan Anton-Regular.ttf

Penggunaannya cukup mudah, contohnya seperti kode berikut:



Gambar 21. Contoh penggunaan font yang sudah ditambahkan

Contohnya dalam widget `Text`, diberikan style dengan `fontFamily` "Anton", dengan penamaan sesuai nama yang diberikan di file `pubspec` tadi. Dengan begitu `fontFamily` akan otomatis berubah ketika aplikasi dijalankan.

## 8. CONTAINER WIDGET

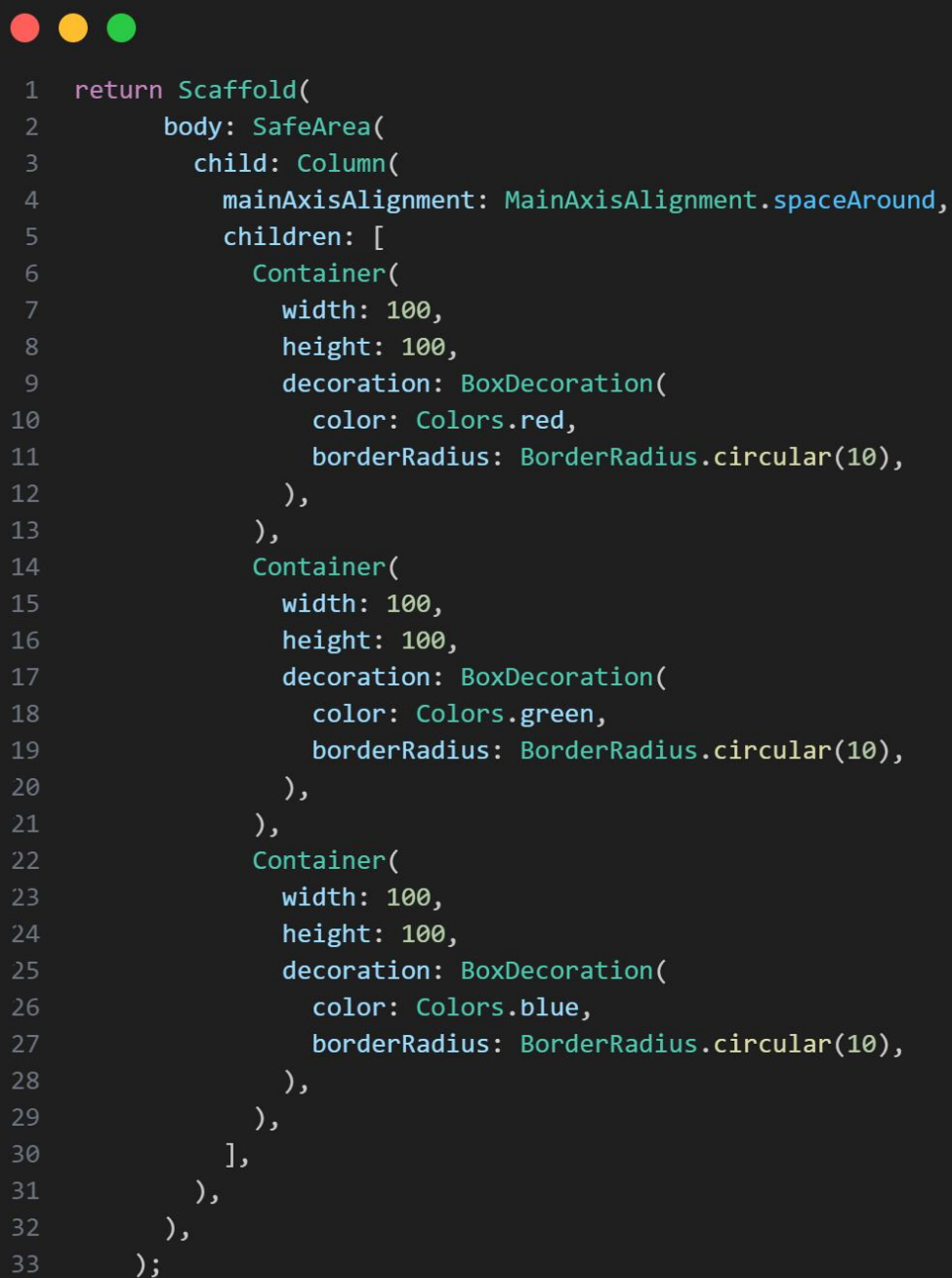
*Container* merupakan *widget* dasar yang digunakan untuk mengatur tata letak elemen antarmuka. Pada dasarnya *Container* merupakan kotak yang ukurannya dapat diatur dengan menggunakan properti seperti *padding*, *margin* dan warna juga dapat diatur melalui properti *color*.

## 9. ROW DAN COLUMN

Row dan column digunakan untuk mengatur layout baik secara vertikal maupun horizontal.

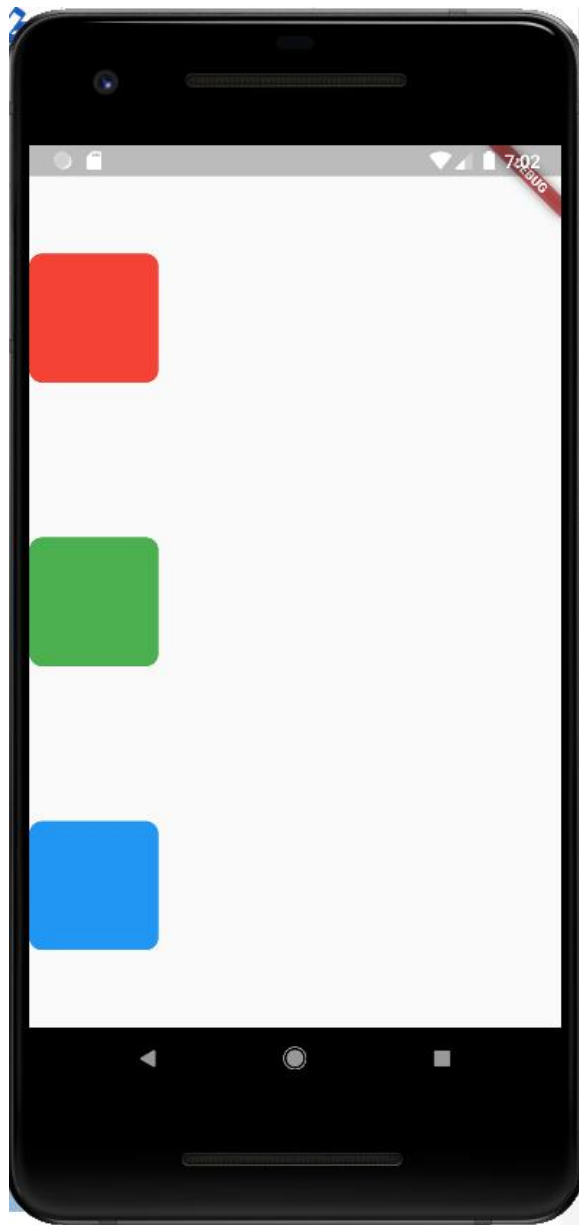
Column digunakan untuk menata elemen-elemen secara vertikal. Column punya elemen yang disebut *children* yang dapat digunakan untuk menampung banyak widget.

Contoh kode :



```
1  return Scaffold(  
2    body: SafeArea(  
3      child: Column(  
4        mainAxisAlignment: MainAxisAlignment.spaceAround,  
5        children: [  
6          Container(  
7            width: 100,  
8            height: 100,  
9            decoration: BoxDecoration(  
10             color: Colors.red,  
11             borderRadius: BorderRadius.circular(10),  
12           ),  
13         ),  
14         Container(  
15           width: 100,  
16           height: 100,  
17           decoration: BoxDecoration(  
18             color: Colors.green,  
19             borderRadius: BorderRadius.circular(10),  
20           ),  
21         ),  
22         Container(  
23           width: 100,  
24           height: 100,  
25           decoration: BoxDecoration(  
26             color: Colors.blue,  
27             borderRadius: BorderRadius.circular(10),  
28           ),  
29         ),  
30       ],  
31     ),  
32   ),  
33 );
```

Gambar 22. Contoh penggunaan column



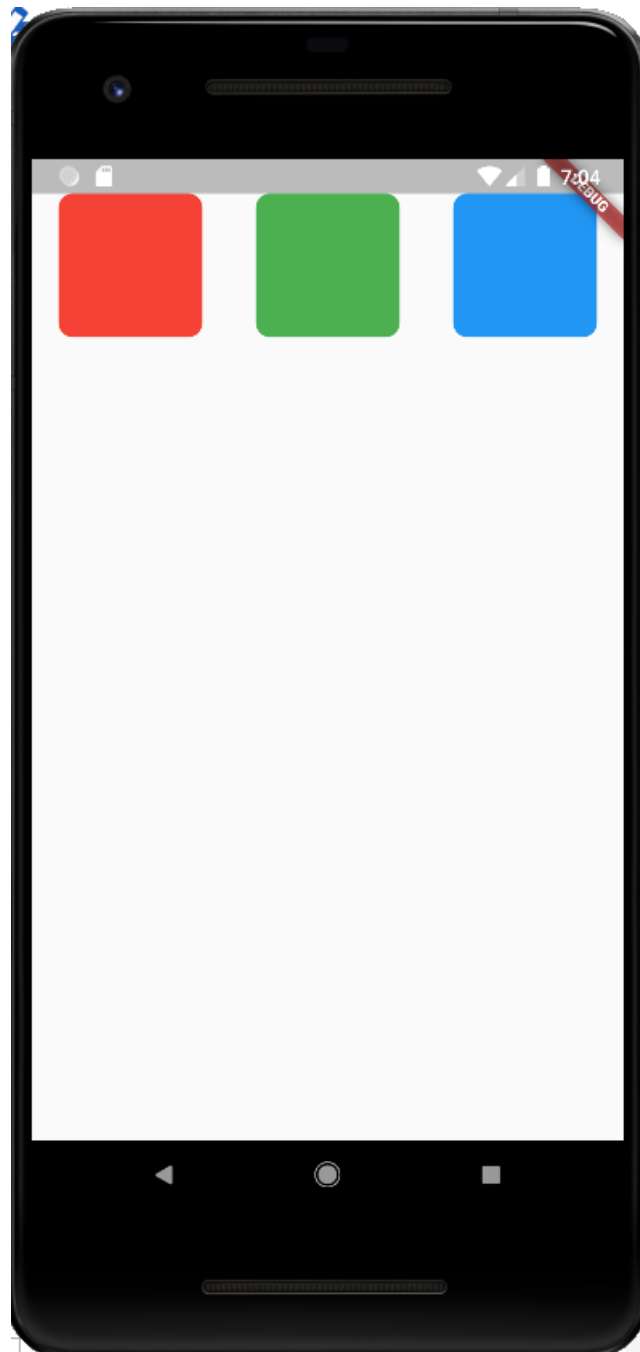
Gambar 23. Hasil kode gambar 22

Row berkebalikan dengan Column yaitu mengatur layout secara horizontal. Elemen yang terdapat pada row mirip dengan column. Cara penggunaan juga mirip, dapat dilihat seperti kode berikut.

Contoh kode :

```
1  return Scaffold(  
2    body: SafeArea(  
3      child: Row(  
4        mainAxisAlignment: MainAxisAlignment.spaceAround,  
5        children: [  
6          Container(  
7            width: 100,  
8            height: 100,  
9            decoration: BoxDecoration(  
10             color: Colors.red,  
11             borderRadius: BorderRadius.circular(10),  
12           ),  
13         ),  
14         Container(  
15           width: 100,  
16           height: 100,  
17           decoration: BoxDecoration(  
18             color: Colors.green,  
19             borderRadius: BorderRadius.circular(10),  
20           ),  
21         ),  
22         Container(  
23           width: 100,  
24           height: 100,  
25           decoration: BoxDecoration(  
26             color: Colors.blue,  
27             borderRadius: BorderRadius.circular(10),  
28           ),  
29         ),  
30       ],  
31     ),  
32   ),  
33 );
```

Gambar 24. Contoh penggunaan row



Gambar 25. Hasil kode gambar 23

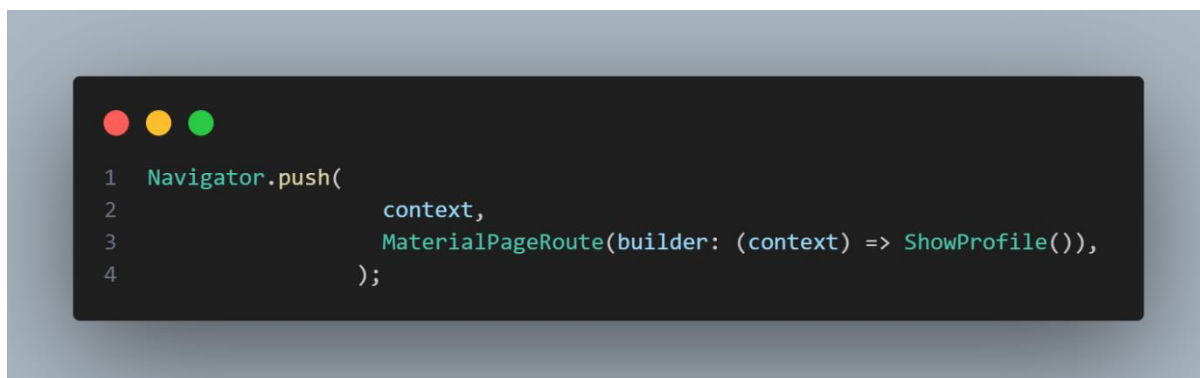
## 10. NAVIGATOR

Navigator digunakan untuk mengatur perpindahan antar halaman di aplikasi. Dalam *flutter* navigator memungkinkan untuk mengelola tumpukan (*stack*) *route* dan melakukan navigasi ke halaman yang diinginkan. Ada beberapa konsep navigasi di *flutter* diantaranya :

- a. *Route* yaitu representasi dari 1 halaman pada aplikasi.



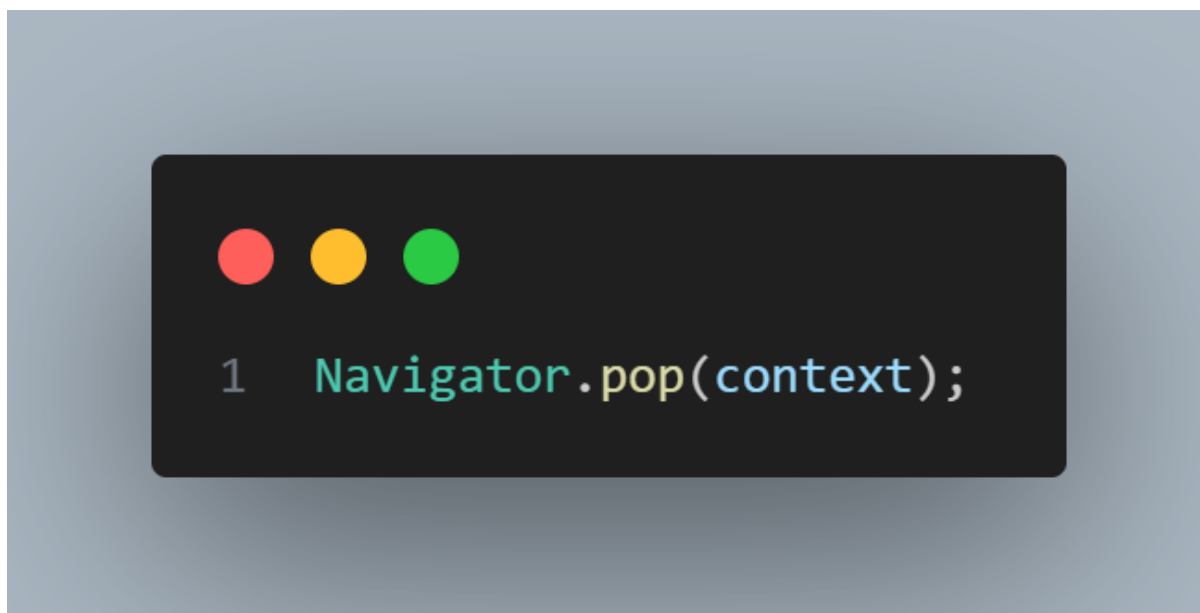
- b. *Navigator.push* yang digunakan untuk menambahkan halaman baru ke dalam stack. Sehingga nanti dapat di panggil lagi.



Gambar 26. Contoh penggunaan Navigator.push

Ketika kode ini dijalankan maka akan mengarah ke halaman ShowProfile() dan ketika tombol back diakses, akan kembali ke halaman ini.

- c. *Navigator.pop* digunakan untuk kembali ke halaman sebelumnya.



Gambar 27. Contoh penggunaan Navigator.pop

- d. *Navigator.pushReplacement* akan mengakses halaman yang dituju, namun berbeda dengan *navigator.push* yang halaman sebelumnya masih bisa diakses,

*Navigator.pushReplacement* tidak bisa mengakses halaman sebelumnya.



Gambar 28. Contoh penggunaan `Navigator.pushReplacement`

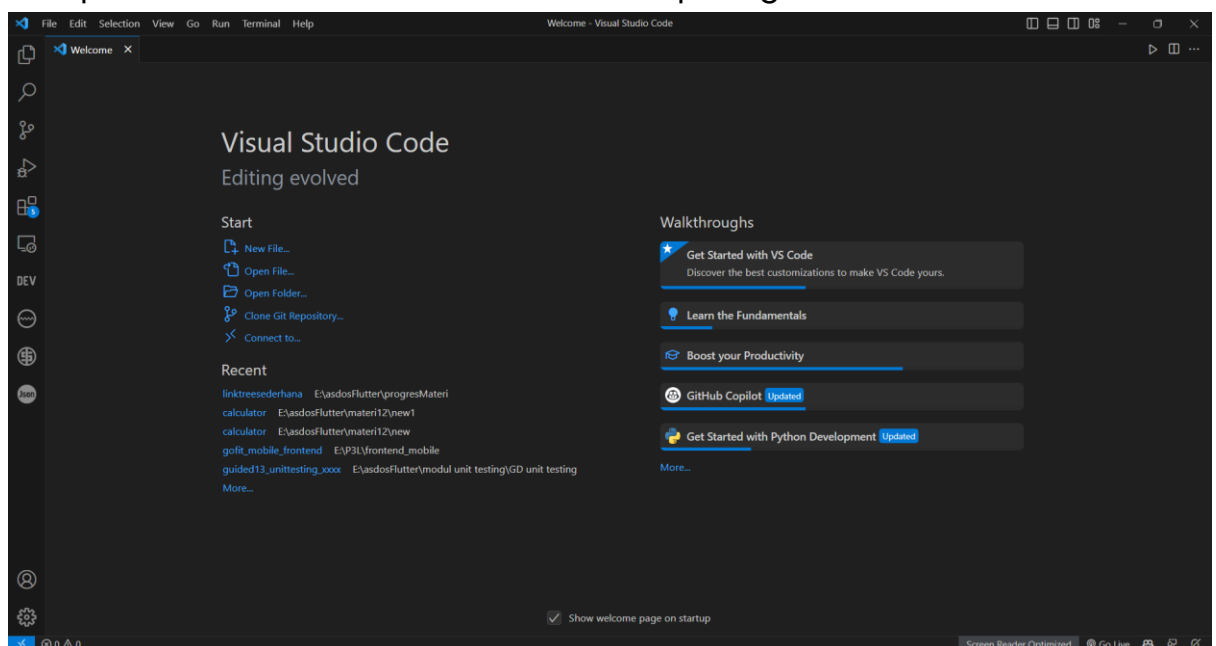
## GUIDED – Profile Sederhana

**Poin yang dipelajari dalam Guided 1 ini, yaitu :**

1. Memahami Penggunaan Widget dasar di *flutter*.
2. Memahami cara *layouting* dengan *column* dan *row*.
3. Memahami cara menambahkan gambar, *font*, dan *icon*.
4. Memahami cara berpindah halaman.

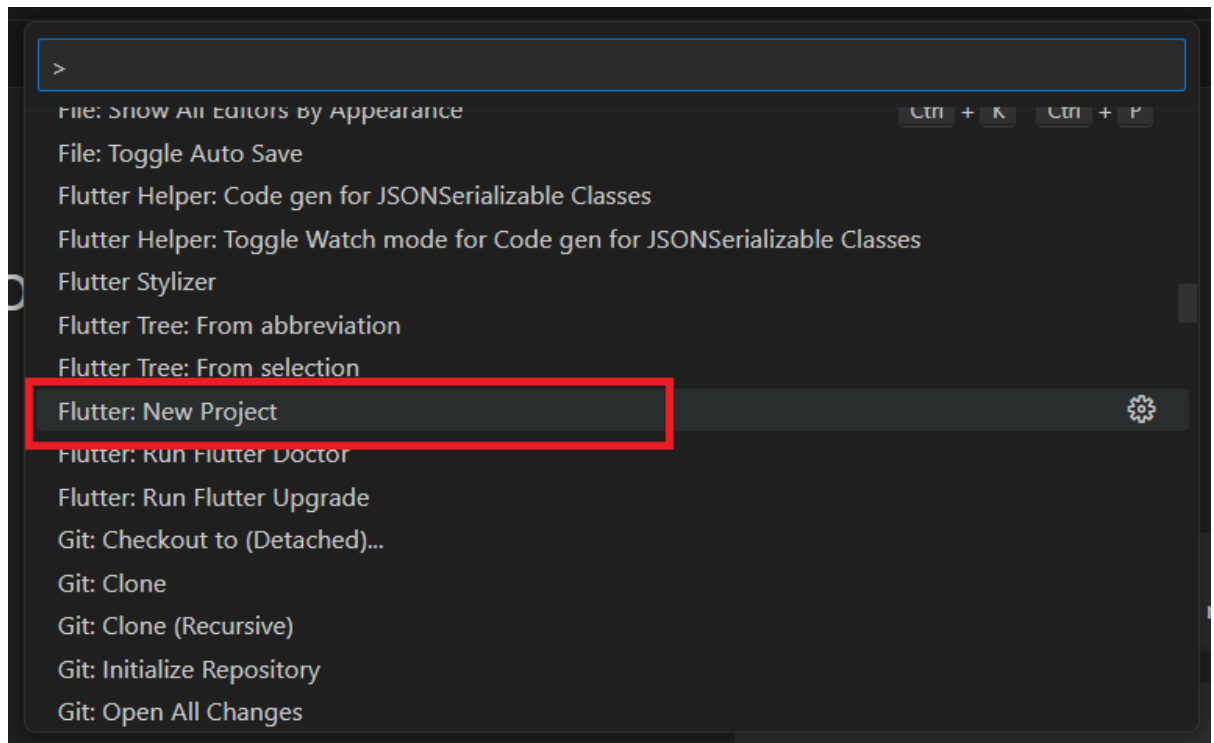
Pada guided 1 ini kita akan membuat profile sederhana yang mirip dengan *linktree*. Silahkan ikuti langkah – langkah berikut ini :

1. Buka visual studio code yang sudah di instal pada minggu pertaman perkuliahan.
2. Tampilan awal visual studio code akan seperti gambar di bawah.



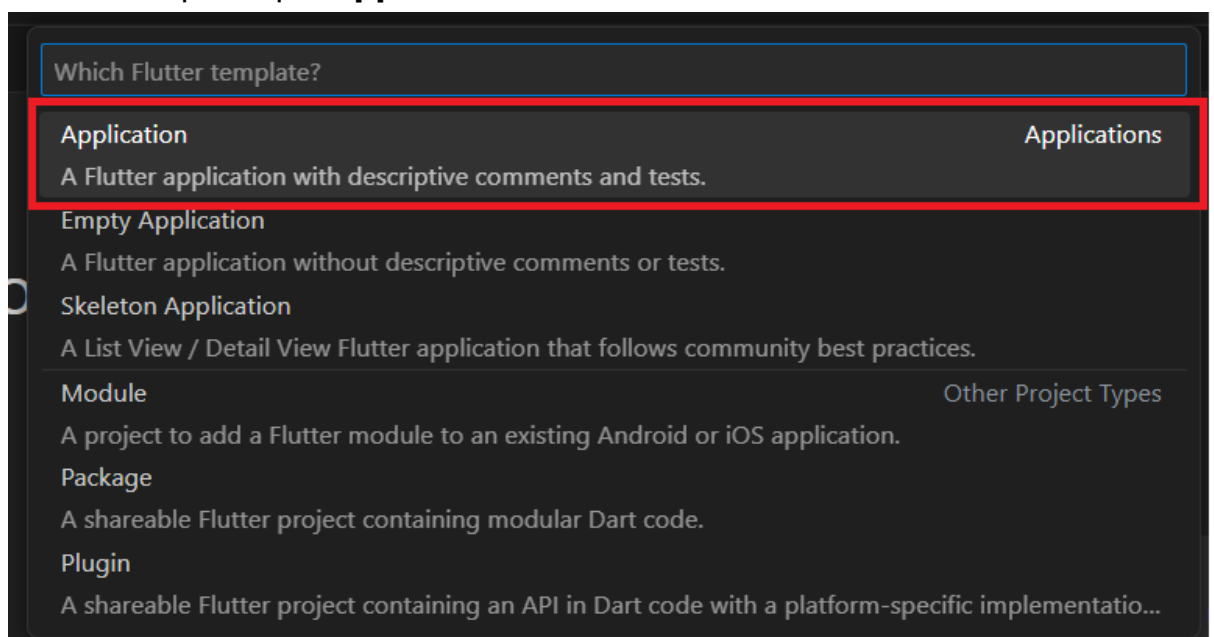
Gambar 29. Visual Studio Code

3. Untuk memulai membuat *template flutter* dapat menggunakan beberapa cara, salah satunya dapat menggunakan *shortcut Ctrl+Shift+P*, kemudian pilih opsi **Flutter: New Project**.



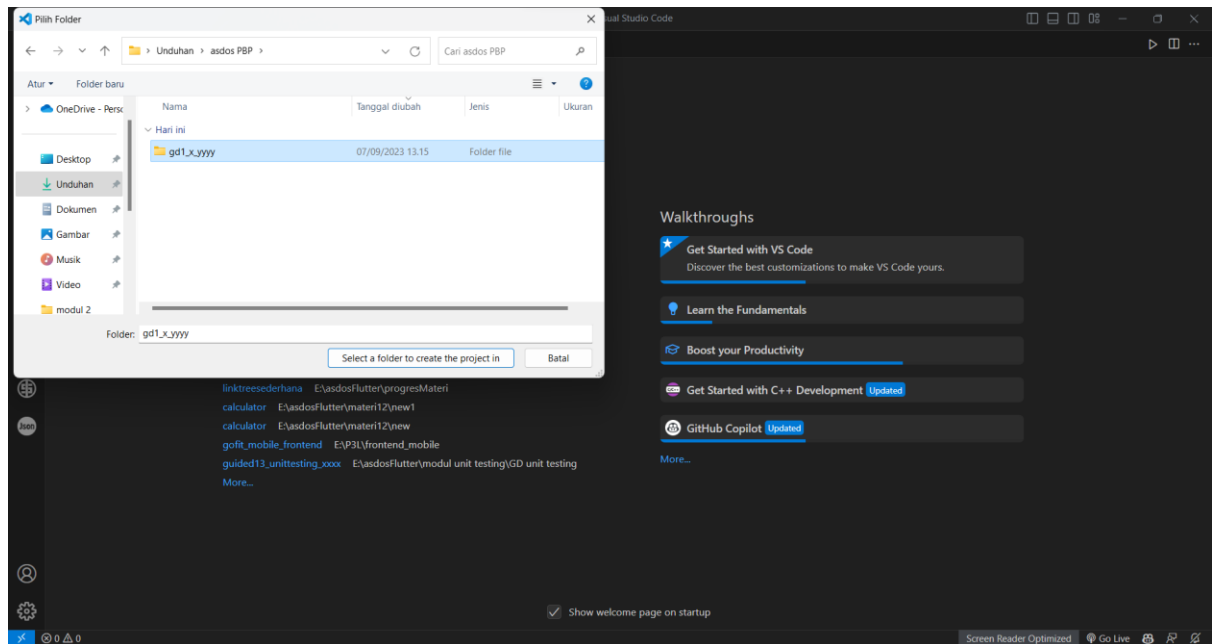
Gambar 30. Pilih opsi untuk memulai flutter project

#### 4. Kemudian pilih opsi **Application**

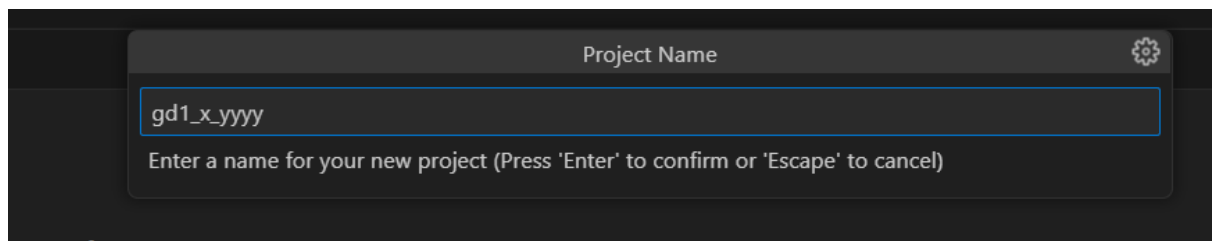


Gambar 31. Pilih opsi Application

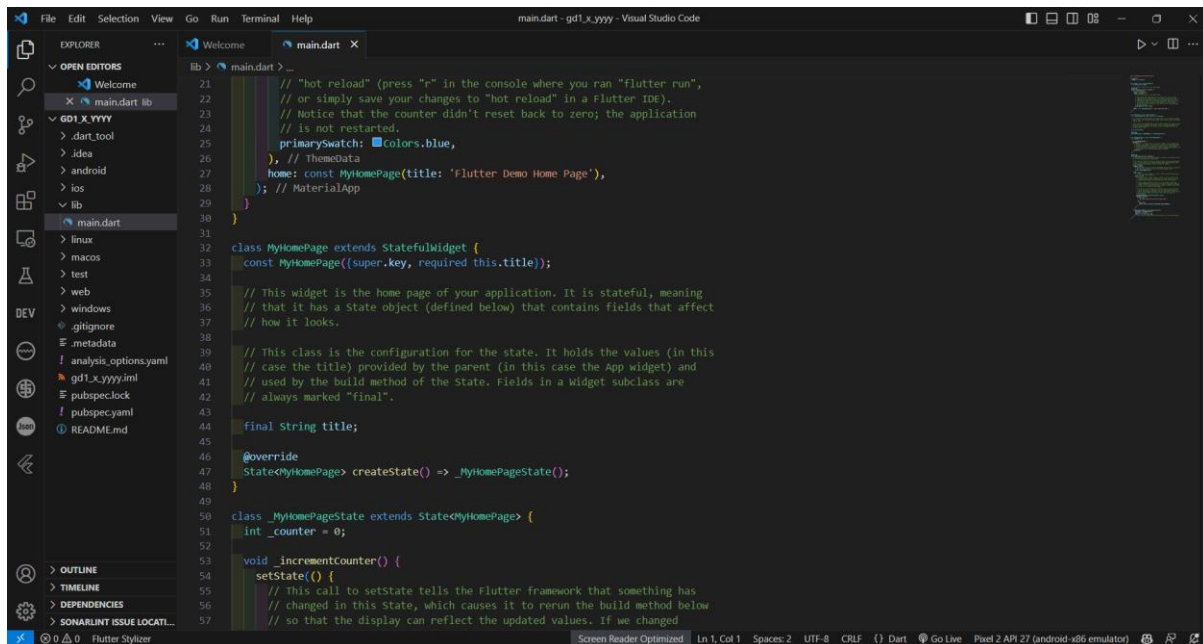
5. Kemudian pilih lokasi penyimpanan *flutter project*, kemudian beri nama sesuai ketentuan berikut : `gdl_x_yyyy`, `x` merupakan nama kelas, dan `yyyy` merupakan 4 digit npm terakhir praktikan.

Gambar 32. Lokasi penyimpanan *flutter project*

- Setelah memastikan lokasi penyimpanan klik tombol ***select a folder to create the project in***, kemudian isikan nama *project flutter* dengan nama dan ketentuan yang sama yaitu `gd1_x_yyyy`. Kemudian klik enter.

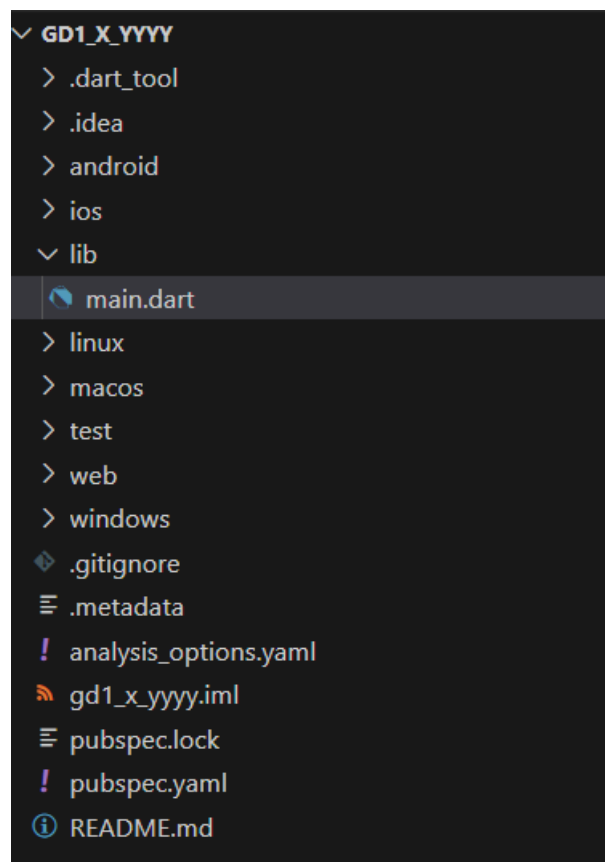
Gambar 33. Menamai *project flutter*

- Pastikan isi foldernya seperti di gambar di bawah ini



Gambar 34. Isi project template flutter

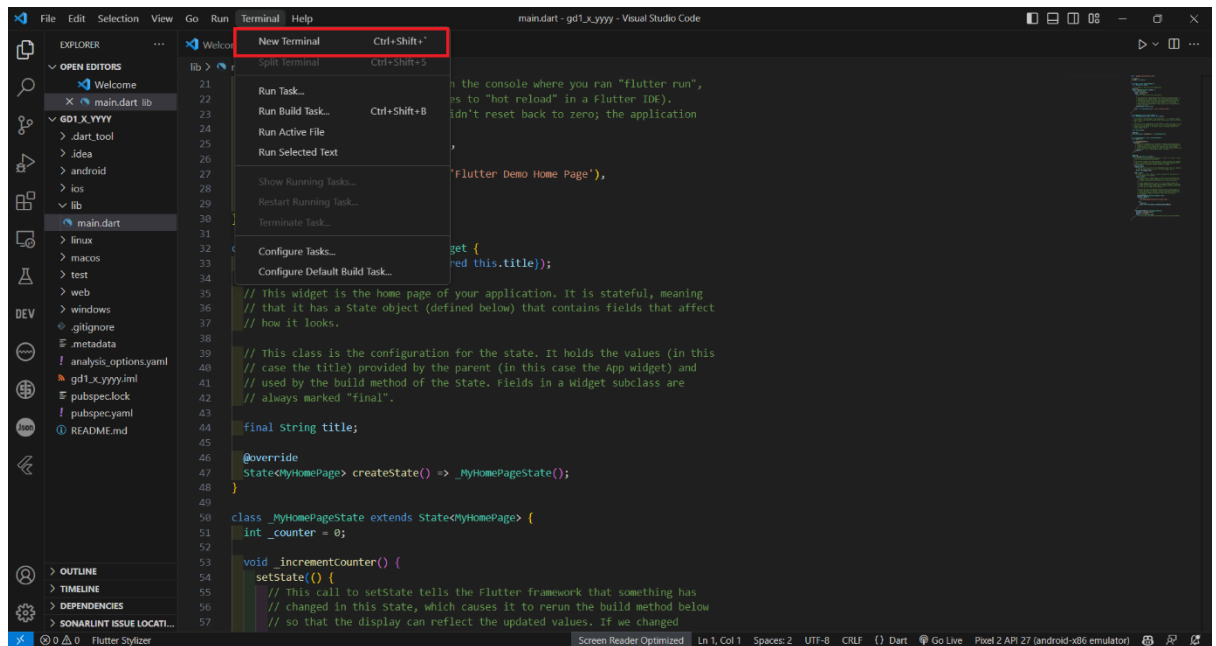
- Setelah folder di buka dengan vs code , buka file main.dart yang berada di folder lib.



Gambar 35. Letak main.dart

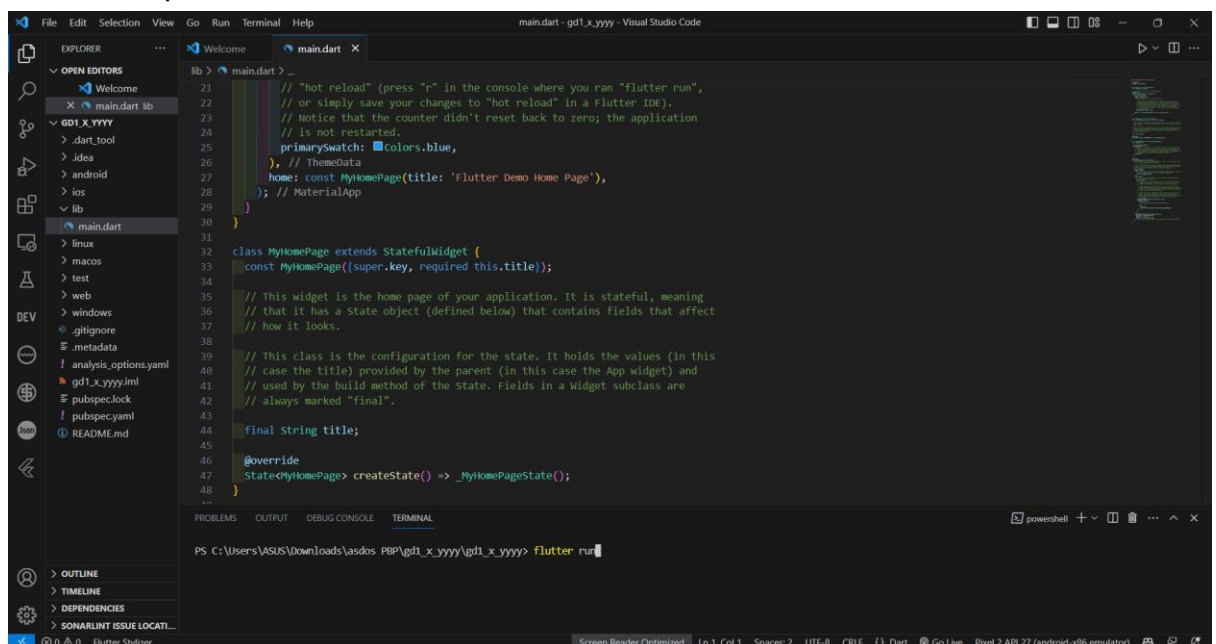
- Jalankan emulator yang sudah di buat di pertemuan sebelumnya.

10. Setelah emulator dijalankan, buka terminal baru di visual studio code dengan klik opsi terminal, kemudian new terminal.



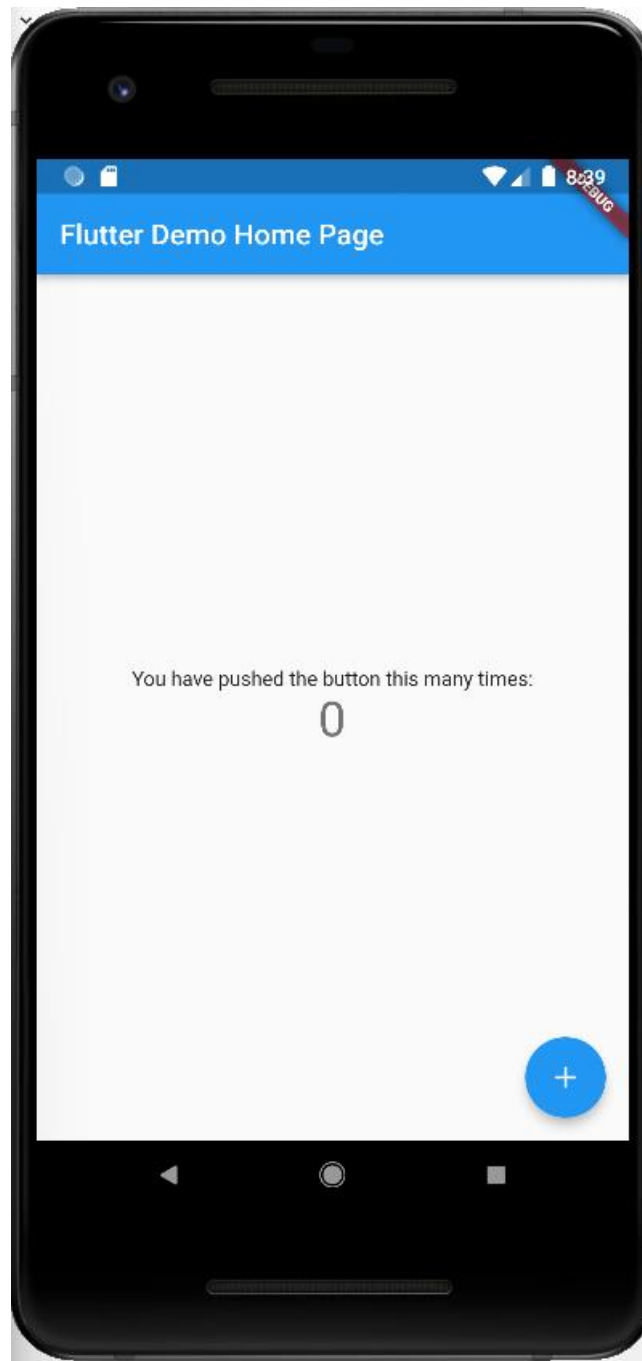
Gambar 36. Buka terminal baru di visual studio code

11. Jalankan perintah **flutter run** di terminal.



Gambar 37. Jalankan perintah flutter run di terminal

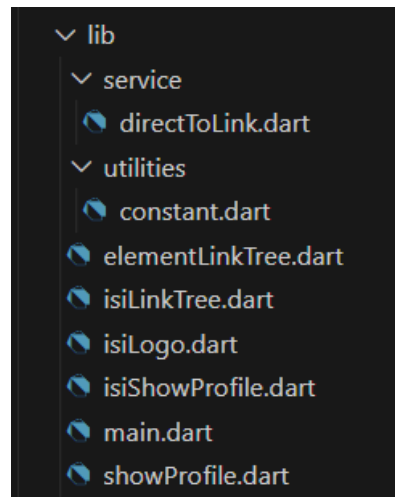
12. Jika kode berhasil maka hasil akhirnya akan seperti gambar 38 di bawah ini.



Gambar 38. Tampilan awal template flutter

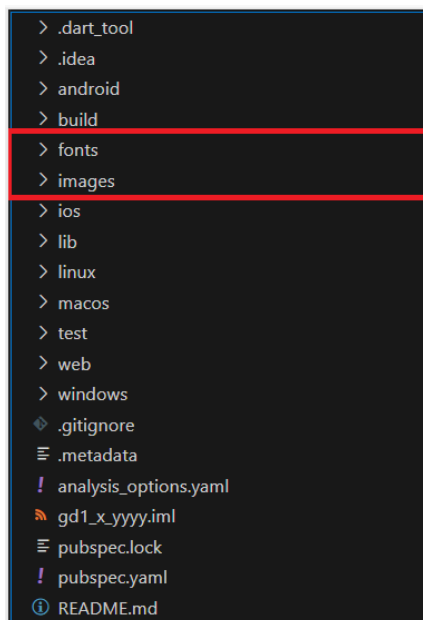
13. Jika sudah tambahkan file dan folder sehingga sesuai dengan susunan pada gambar 39.





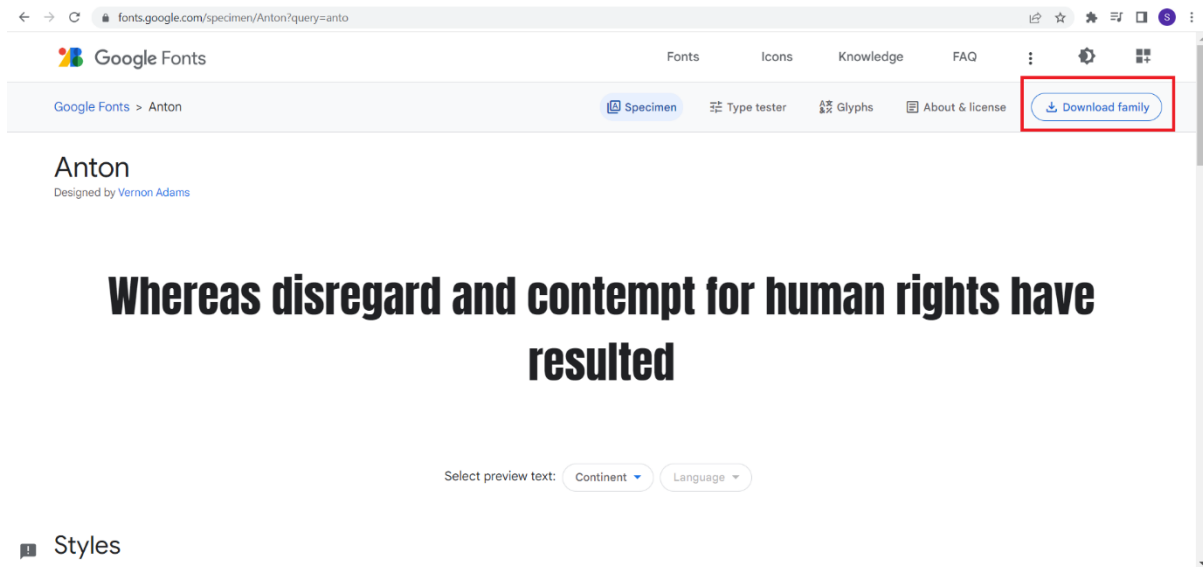
Gambar 39. Menambahkan file dan folder di folder lib

#### 14. Buat juga folder dengan nama images dan fonts



Gambar 40. Menambahkan folder images dan fonts

#### 15. Buka link <https://fonts.google.com/specimen/Anton?query=anto>, dan download family



Gambar 41. Mengambil salah satu font dari google font

16. Ekstrak folder yang sudah di download dan copy file dengan ekstensi .ttf ke dalam folder fonts yang sudah kita buat tadi.

Anton-Regular.ttf	18/06/2023 20:57	File font TrueType	158 KB
OFL.txt	18/06/2023 20:57	Dokumen Teks	5 KB

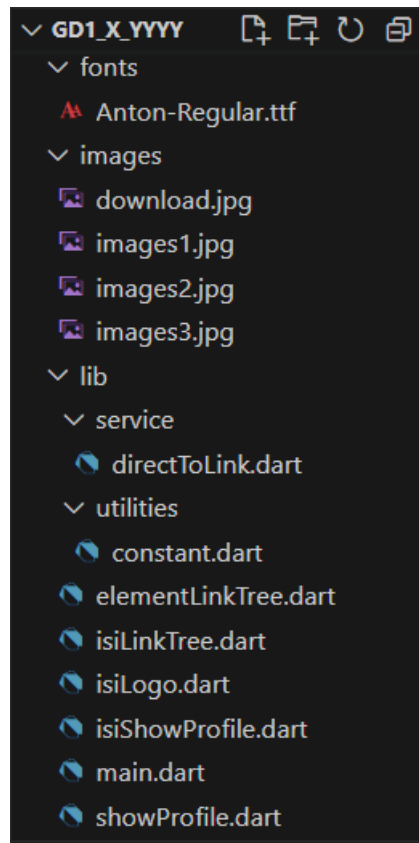
Gambar 42. Copy file ini

17. Isi folder images dengan 4 gambar bebas dan *rename* menjadi seperti gambar di bawah ini.

download.jpg	28/03/2023 20:00	File JPG	3 KB
images1.jpg	28/03/2023 20:45	File JPG	7 KB
images2.jpg	28/03/2023 20:45	File JPG	9 KB
images3.jpg	28/03/2023 20:45	File JPG	5 KB


Gambar 43. Penamaan file gambar

18. Maka isi dari folder utama akan seperti gambar berikut



Gambar 44. struktur utama

19. Lakukan perubahan pada file *pubspec.yaml* dengan kode di bawah ini.



```
1  name: gd1_x_yyyy
2  description: A new Flutter project.
3  publish_to: 'none'
4
5
6  version: 1.0.0+1
7
8  environment:
9    sdk: '>=2.19.5 <3.0.0'
10 dependencies:
11   flutter:
12     sdk: flutter
13
14   cupertino_icons: ^1.0.2
15   font_awesome_flutter: ^10.4.0
16   url_launcher: ^6.1.10
17
18 dev_dependencies:
19   flutter_test:
20     sdk: flutter
21
22   flutter_lints: ^2.0.0
23
24
25 flutter:
26   uses-material-design: true
27   assets:
28     - images/
29
30   fonts:
31     - family: Anton
32       fonts:
33         - asset: fonts/Anton-Regular.ttf
```

Gambar 45. Isi file pubspec.yaml



GAMBAR 45 → merupakan cara agar folder images dan font dibaca sebagai source yang bisa digunakan dalam proses pengkodean. Perlu diperhatikan karena pubspec.yaml menggunakan indentasi untuk membaca alur kode, jadi jangan sampai salah dalam menuliskan indentasinya.

**20.** Lakukan perubahan pada file main.dart seperti gambar di bawah ini.

```
1 import 'package:flutter/material.dart';
2 import 'package:gdl_x_yyyy/showProfile.dart';
3 import 'package:gdl_x_yyyy/isilinkTree.dart';
4 import 'package:gdl_x_yyyy/isiLogo.dart';
5 import 'package:gdl_x_yyyy/utilities/constant.dart';
6
7 void main() => runApp(const MyApp());
8
9 class MyApp extends StatelessWidget {
10   const MyApp({super.key});
11
12   @override
13   Widget build(BuildContext context) {
14     return MaterialApp(
15       home: const MyHomePage(),
16     );
17   }
18 }
19
20 class MyHomePage extends StatefulWidget {
21   const MyHomePage({super.key});
22
23   @override
24   State<MyHomePage> createState() => _MyHomePageState();
25 }
26
27 class _MyHomePageState extends State<MyHomePage> {
28   @override
29   Widget build(BuildContext context) {
30     return SafeArea(
31       child: Scaffold(
32         backgroundColor: const Color.fromARGB(255, 228, 105, 132),
33         body: Center(
34           child: Column(
35             mainAxisAlignment: MainAxisAlignment.center,
36             children: [
37               TextButton(
38                 onPressed: () {
39                   Navigator.push(
40                     context,
41                     MaterialPageRoute(
42                       builder: (context) => const ShowProfile(),
43                     );
44                 },
45               ),
46               child: const CircleAvatar(
47                 radius: 50,
48                 backgroundImage: AssetImage('images/download.jpg'),
49               ),
50               const Text(
51                 'Sakshi',
52                 style: kTextStyle1,
53               ),
54               const Text(
55                 'FLUTTER DEVELOPER',
56                 style: kTextStyle2,
57               ),
58               kSizeBox,
59               const IsilinkTree(),
60               kSizeBox,
61               const IsiLogo(),
62             ],
63           ),
64         ),
65       );
66     );
67   }
68 }
```

Gambar 46. Isi file main.dart

GAMBAR 46 → merupakan tampilan awal linktree yang akan dibuat. Pada awal pengerjaan akan ada error karena elemen IsiLinkTree dan IsiLogo belum dibuat, abaikan saja dahulu.

21. Lakukan perubahan pada file elementLinkTree.dart.

```
1  import 'package:flutter/material.dart';
2
3  class CardForlinkTree extends StatelessWidget {
4    const CardForlinkTree({
5      super.key,
6      required this.icon,
7      required this.text,
8      this.onPressed,
9    });
10
11    final IconData icon;
12    final String text;
13    final Function? onPressed;
14
15    @override
16    Widget build(BuildContext context) {
17      return TextButton(
18        onPressed: onPressed as void Function()?,
19        child: Card(
20          color: Colors.white,
21          margin: const EdgeInsets.symmetric(vertical: 10, horizontal: 25),
22          child: ListTile(
23            leading: Icon(
24              icon,
25              color: Colors.teal,
26            ),
27            title: Text(
28              text,
29              style: TextStyle(
30                color: Colors.teal.shade900,
31                fontFamily: 'Source Sans Pro',
32                fontSize: 20,
33              ),
34            ),
35          ),
36        ),
37      );
38    }
39  }
```

Gambar 47. Isi file elementLinkTree.dart



GAMBAR 47 → merupakan fungsi yang digunakan untuk menampilkan elemen *linktree* yang dibuat.

**22.** Lakukan perubahan pada file `showProfile.dart`



```

1 import 'package:flutter/material.dart';
2 import 'package:gd1_x_yyyy/isiShowProfile.dart';
3 import 'package:gd1_x_yyyy/utilities/constant.dart';
4
5 class ShowProfile extends StatelessWidget {
6   const ShowProfile({super.key});
7
8   @override
9   Widget build(BuildContext context) {
10    return SafeArea(
11      child: Scaffold(
12        backgroundColor: kColor,
13        body: Center(
14          child: Column(
15            children: [
16              Container(
17                margin: EdgeInsets.only(top: 50),
18                child: Row(
19                  children: [
20                    kSizeBox2,
21                    CircleAvatar(
22                      radius: 50,
23                      backgroundImage: AssetImage('images/download.jpg'),
24                    ),
25                    kSizeBox2,
26                    Container(
27                      child: Column(
28                        crossAxisAlignment: CrossAxisAlignment.start,
29                        children: [
30                          Text('Sakshi', style: kTextSytle4),
31                          Text('FLUTTER DEVELOPER', style: kTextStyle5),
32                        ],
33                      ),
34                    ],
35                  ),
36                ),
37              Container(
38                margin: EdgeInsets.only(top: 50),
39                child: Row(
40                  mainAxisAlignment: MainAxisAlignment.spaceAround,
41                  children: [
42                    Container(
43                      child: Column(
44                        children: [
45                          Text('3', style: kTextStyle3),
46                          Text('Posts', style: kTextStyle3),
47                        ],
48                      ),
49                    ),
50                    Container(
51                      child: Column(
52                        children: [
53                          Text('3', style: kTextStyle3),
54                          Text('Followers', style: kTextStyle3),
55                        ],
56                      ),
57                    ),
58                    Container(
59                      child: Column(
60                        children: [
61                          Text('10', style: kTextStyle3),
62                          Text('Following', style: kTextStyle3),
63                        ],
64                      ),
65                    ),
66                  ],
67                ),
68              ),
69              kSizeBox,
70              IsiShowProfile()
71            ],
72          ),
73        ),
74        floatingActionButton: FloatingActionButton(
75          onPressed: () {
76            Navigator.pop(context);
77          },
78          child: Icon(Icons.arrow_back),
79          backgroundColor: Colors.teal,
80        ),
81      );
82    }
83  }

```

Gambar 48. Isi file showProfile.dart

### 23. Lakukan perubahan pada file isiLogo.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:font_awesome_flutter/font_awesome_flutter.dart';
3 import 'package:gd1_x_yyyy/utilities/constant.dart';
4
5 class IsiLogo extends StatefulWidget {
6   const IsiLogo({super.key});
7
8   @override
9   State<IsiLogo> createState() => _IsiLogoState();
10 }
11
12 class _IsiLogoState extends State<IsiLogo> {
13   @override
14   Widget build(BuildContext context) {
15     return Row(
16       mainAxisAlignment: MainAxisAlignment.center,
17       children: [
18         const Icon(FontAwesomeIcons.github),
19         kSizeBox2,
20         const Icon(FontAwesomeIcons.user),
21         kSizeBox2,
22         const Icon(FontAwesomeIcons.googleDrive),
23       ],
24     );
25   }
26 }
```

Gambar 49. isi file isiLogo.dart

### 24. Lakukan Perubahan pada File isiShowProfile.dart

```
1 import 'package:flutter/material.dart';
2
3 class IsiShowProfile extends StatefulWidget {
4   const IsiShowProfile({super.key});
5
6   @override
7   State<IsiShowProfile> createState() => _IsiShowProfileState();
8 }
9
10 class _IsiShowProfileState extends State<IsiShowProfile> {
11   @override
12   Widget build(BuildContext context) {
13     return Row(
14       mainAxisAlignment: MainAxisAlignment.spaceAround,
15       children: [
16         Container(
17           width: 110,
18           height: 110,
19           color: Colors.white,
20           child: Image.asset('images/images1.jpg'),
21         ),
22         Container(
23           width: 110,
24           height: 110,
25           color: Colors.white,
26           child: Image.asset('images/images2.jpg'),
27         ),
28         Container(
29           width: 110,
30           height: 110,
31           color: Colors.white,
32           child: Image.asset('images/images3.jpg'),
33         ),
34       ],
35     );
36   }
37 }
```

Gambar 50. isi file isiShowProfile.dart

GAMBAR 50 → merupakan komponen yang akan dipanggil di gambar 48 dengan tujuan yang sama dengan gambar 49 dan 51.



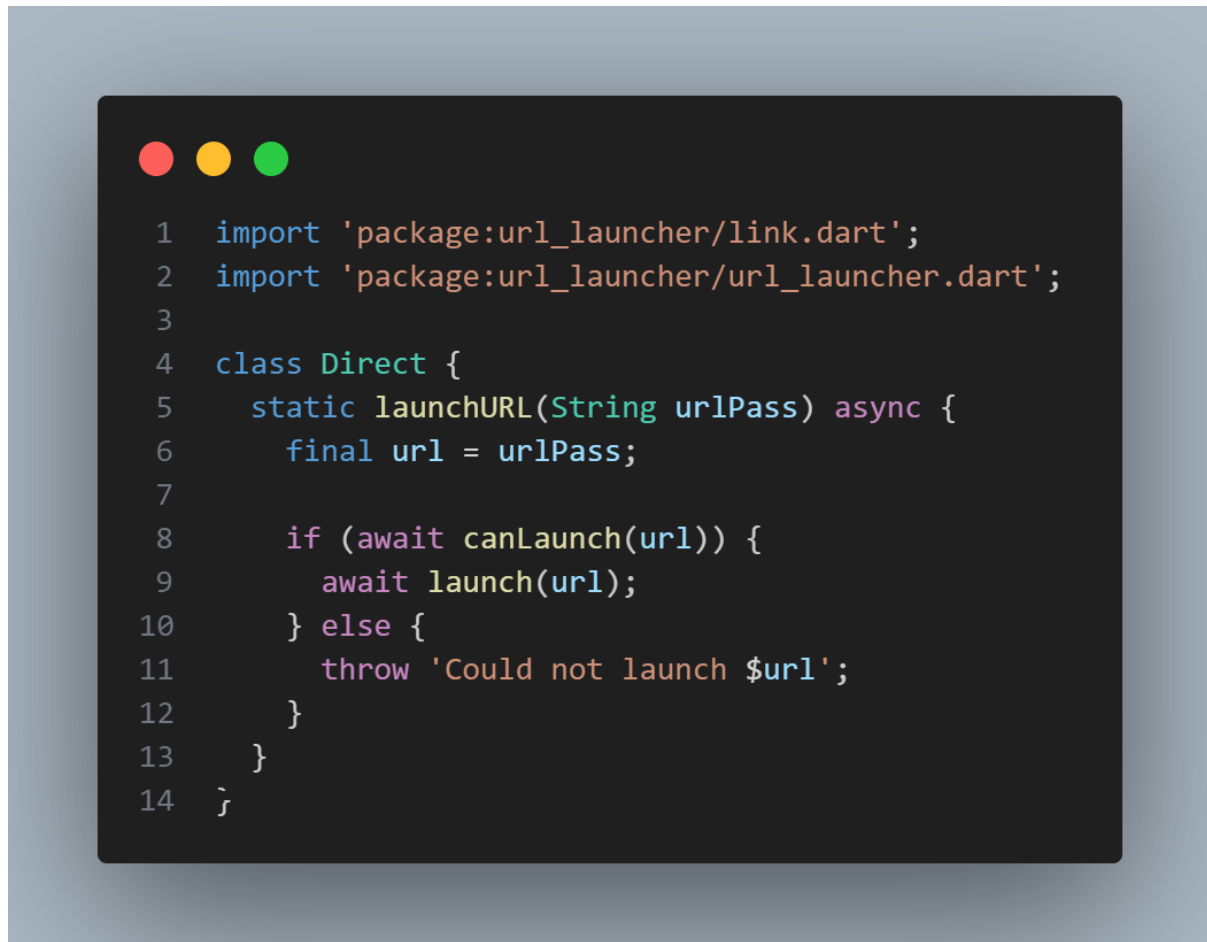
25. Lakukan perubahan pada file isiLinkTree.dart

```
1  import 'package:flutter/material.dart';
2  import 'package:gd1_x_yyyy/elementLinkTree.dart';
3  import 'package:font_awesome_flutter/font_awesome_flutter.dart';
4  import 'package:gd1_x_yyyy/service/directToLink.dart';
5
6  class IsiLinkTree extends StatefulWidget {
7    const IsiLinkTree({super.key});
8
9    @override
10    State<IsiLinkTree> createState() => _IsiLinkTreeState();
11  }
12
13  class _IsiLinkTreeState extends State<IsiLinkTree> {
14    @override
15    Widget build(BuildContext context) {
16      return Column(
17        children: [
18          CardForlinkTree(
19            text: '+91 123 456 789',
20            icon: Icons.phone,
21          ),
22          const CardForlinkTree(
23            text: 'broman@gmail.com',
24            icon: Icons.email,
25          ),
26          CardForlinkTree(
27            text: 'Instagram',
28            icon: FontAwesomeIcons.instagram,
29            onPressed: () {
30              Direct.launchURL('https://www.instagram.com/');
31            },
32          ),
33          CardForlinkTree(
34            text: 'Facepook',
35            icon: FontAwesomeIcons.facebook,
36            onPressed: () {
37              Direct.launchURL('https://www.facebook.com/');
38            },
39          ),
40        ],
41      );
42    }
43  }
```

Gambar 51. isi file isiLinkTree.dart

GAMBAR 49 dan 51 → merupakan komponen yang akan dipanggil ke gambar 46. Komponen ini dipecah agar mempermudah jika ada perbaikan code yang mungkin terjadi.

**26.** Lakukan perubahan pada file `directToLink.dart`



Gambar 52. isi file `directToLink.dart`

GAMBAR 52 → merupakan logic yang kita pakai ketika link yang dimasukkan di gambar 51 bisa langsung direct ke link tersebut. (note: link bisa diisi dengan link dari sosial media kalian.)

**27.** Lakukan perubahan pada file `constant.dart`



```
1  import 'package:flutter/material.dart';
2
3  const kColor = Color.fromARGB(255, 228, 105, 132);
4
5  const SizedBox kSizeBox2 = SizedBox(
6    width: 20,
7  );
8
9  const SizedBox kSizeBox = SizedBox(
10   height: 20,
11   width: 350,
12   child: Divider(
13     color: Colors.teal,
14     thickness: 1.5,
15   ),
16 );
17
18 const kTextStyle1 = TextStyle(
19   fontFamily: 'Anton',
20   fontSize: 40,
21   color: Colors.white,
22   fontWeight: FontWeight.bold,
23   letterSpacing: 2.5,
24 );
25
26 const kTextStyle2 = TextStyle(
27   fontFamily: 'Source Sans Pro',
28   fontSize: 20,
29   color: Colors.white,
30   letterSpacing: 2.5,
31   fontWeight: FontWeight.bold,
32 );
33
34 const kTextStyle3 = TextStyle(
35   fontFamily: 'Source Sans Pro',
36   fontSize: 20,
37   color: Colors.white,
38   fontWeight: FontWeight.bold,
39   letterSpacing: 2.5,
40 );
41
42 const kTextSytle4 = TextStyle(
43   fontFamily: 'Pacifico',
44   fontSize: 40,
45   color: Colors.white,
46   fontWeight: FontWeight.bold,
47 );
48
49 const kTextStyle5 = TextStyle(
50   fontFamily: 'Source Sans Pro',
51   fontSize: 20,
52   color: Colors.white,
53   fontWeight: FontWeight.bold,
54   letterSpacing: 2.5,
55 );
```

Gambar 53. isi file constant.dart

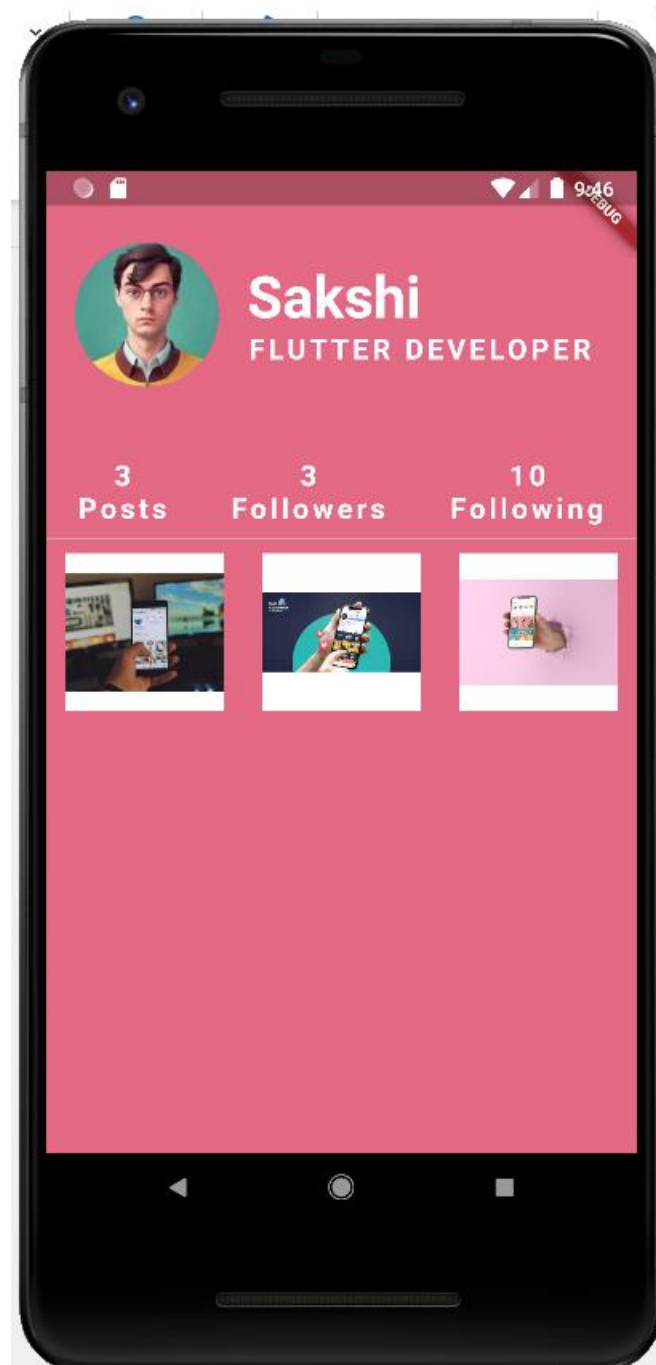
GAMBAR 53 → merupakan konstan atau dalam bahasa dart ditulis dengan `const` yang digunakan untuk menampung nilai yang tidak akan berubah selama kode berjalan. Konstan ini sangat mempermudah dalam proses pengkodean karena kita dapat memanggil nilai yang sama tanpa harus menulis ulang kodenya.

28. Jalankan ulang flutter run di terminal dan perhatikan hasilnya akan seperti gambar di bawah ini.



Gambar 54. hasil 1

Gambar 54 → merupakan hasil dari kode ketika pertama kali dijalankan. Item pada CardForlinkTree bisa di klik, dan jika Anda ingin mencoba bisa diganti dengan alamat sosial media Anda sendiri.



Gambar 55. hasil 2

Gambar 55 → bisa diakses dengan mengklik gambar profile pada bagian atas yang akan mengarahkan Anda ke halaman ini.





**NOTE :**

Perhatikan indentasi di file pubspec.yaml. Jika terjadi error di awal pembuatan, abaikan saja terlebih dahulu hingga seluruh code berhasil di salin.



#### **ATURAN Pengerjaan Guided :**

- Guided dikerjakan di luar jam perkuliahan, sesuai dengan kesepakatan yang ada.
- Penamaan projek guided harus sesuai dengan yang sudah dicontohkan.
- Guided dikumpulkan melalui github secara private dengan penamaan setiap file pada github adalah : **GD1\_X\_YYYY** (contoh : **GD1\_A\_0989**)
  - o **X → nama kelas**
  - o **YYYY → 4 DIGIT TERAKHIR NPM**
- Setelah diupload melalui github, jangan lupa untuk mengumpulkan keseluruhan link file github melalui situs kuliah.
- Cara upload ke github, silahkan melihat pada modul **"UPLOAD GITHUB"**.