



UNIVERSITAS SEMARANG
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
TEKNIK INFORMATIKA

Mobile Application

Modul Praktikum Mahasiswa

Oleh:

Alauddin Maulana Hirzan, S. Kom., M. Kom
NIDN. 0607069401

Daftar Isi

Pendahuluan	2
0.1 Mengenal Android	2
0.2 Mengenal Firebase dan Realtime Database	2
0.3 Mengenal JSON-Tree	3
0.4 Mengenal Flutter Framework	4
Persiapan Praktikum	5
0.5 Perangkat Keras	5
0.6 Perangkat Lunak	5
1 Praktikum 1	6
1.1 Konfigurasi Firebase Realtime DB dan Flutter IDE	6
1.2 Tutorial	6
2 Praktikum 2	12
2.1 Antarmuka Login Flutter	12
2.2 Tutorial	12
3 Praktikum 3	22
3.1 Halaman Registrasi dan Navigasi Page	22
3.2 Tutorial	22
4 Praktikum 4	32
4.1 Integrasi Firebase	32
4.2 Tutorial	32
5 Praktikum 5	38
5.1 Operasi Insert Data Flutter	38
5.2 Tutorial	38
6 Praktikum 6	48
6.1 Operasi Read Data Flutter	48
6.2 Tutorial	48
7 Praktikum 7	63
7.1 Operasi Update Data	63
7.2 Tutorial	63

8 Praktikum 8	77
8.1 Operasi Delete Data	77
8.2 Tutorial	77

Daftar Gambar

1	Perangkat Android	2
2	Realtime Database	3
3	Contoh JSON Tree	3
4	Flutter	4

Pendahuluan

0.1 Mengenal Android

Sistem operasi Android adalah sistem operasi seluler untuk digunakan terutama untuk perangkat layar sentuh, ponsel, dan tablet. Desainnya memungkinkan pengguna memanipulasi perangkat seluler secara intuitif, dengan gerakan jari yang mencerminkan gerakan umum, seperti mencubit, menggesek, dan mengetuk.



Gambar 1: Perangkat Android

0.2 Mengenal Firebase dan Realtime Database

Firebase adalah platform yang dikembangkan oleh Google untuk membuat aplikasi seluler dan web. Salah satunya produk yang sering digunakan di Firebase adalah Realtime Database. Firebase Realtime Database adalah database yang dihosting di cloud. Data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung.



Gambar 2: Realtime Database

0.3 Mengenal JSON-Tree

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan. Mudah bagi manusia untuk membaca dan menulis. Sangat mudah bagi mesin untuk menguraikan dan menghasilkan. Ini didasarkan pada subset dari Standar Bahasa Pemrograman JavaScript ECMA-262 Edisi ke-3 - Desember 1999. JSON adalah format teks yang sepenuhnya bebas bahasa tetapi menggunakan konvensi yang akrab bagi pemrogram keluarga bahasa C, termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan banyak lainnya. Properti ini menjadikan JSON sebagai bahasa pertukaran data yang ideal.

JSON dibangun di atas dua struktur:

- Kumpulan pasangan nama/nilai. Dalam berbagai bahasa, ini diwujudkan sebagai objek, catatan, struct, kamus, tabel hash, daftar kunci, atau array asosiatif.
- Daftar nilai yang diurutkan. Dalam kebanyakan bahasa, ini diwujudkan sebagai array, vektor, daftar, atau urutan.

Ini adalah struktur data universal. Hampir semua bahasa pemrograman modern mendukungnya dalam satu atau lain bentuk. Masuk akal bahwa format data yang dapat dipertukarkan dengan bahasa pemrograman juga didasarkan pada struktur ini.

Response Body Select Body

```
{  
  "results": [  
    {  
      "bioguide_id": "P000606",  
      "birthday": "1948-08-15",  
      "chamber": "house",  
      "contact_form": "https://pittenger.house.gov/contact/email-me",  
      "crp_id": "N00034416",  
      "district": 9,  
      "facebook_id": "376142742468386",  
      "fax": "202-225-3389",  
      "id": "H000606",  
      "name": "Pittenger, John",  
      "party": "Democrat",  
      "state": "Pennsylvania",  
      "status": "Served",  
      "url": "https://pittenger.house.gov",  
      "year_elected": 1993  
    }  
  ]  
}
```

Gambar 3: Contoh JSON Tree

0.4 Mengenal Flutter Framework

Flutter adalah kerangka kerja sumber terbuka oleh Google untuk membangun aplikasi multi-platform yang indah, dikompilasi secara asli, dari satu basis kode. Flutter mengubah proses pengembangan aplikasi. Buat, uji, dan terapkan aplikasi seluler, web, desktop, dan tersemat yang cantik dari satu basis kode.



Gambar 4: Flutter

Persiapan Praktikum

Agar praktikum dapat berjalan dengan lancar, mahasiswa diwajibkan memenuhi persyaratan berikut baik dalam bentuk perangkat keras maupun lunak:

0.5 Perangkat Keras

- Prosesor dengan 4 inti
- RAM minimal 4GB, rekomendasi 8GB
- HDD 10GB

0.6 Perangkat Lunak

Perangkat lunak berikut ini wajib diinstall oleh mahasiswa demi lancarnya praktikum:

- Browser
- FlutLab Desktop (opsional)

Bab 1

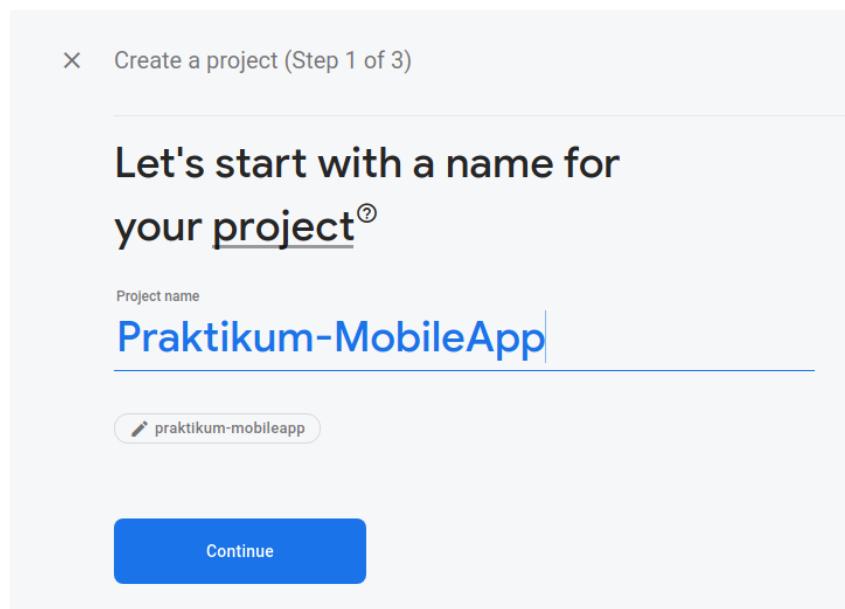
Praktikum 1

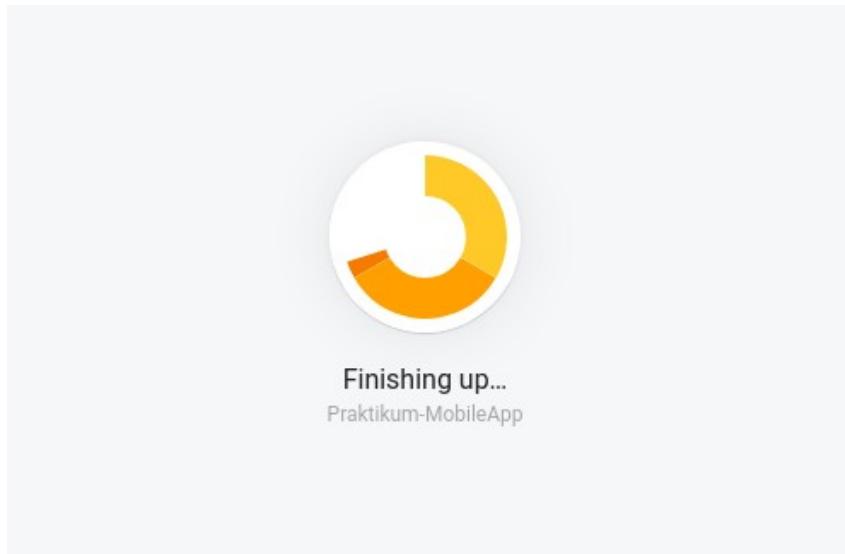
1.1 Konfigurasi Firebase Realtime DB dan Flutter IDE

Di bagian ini mahasiswa diajarkan bagaimana membuat projek Firebase dengan Realtime Database. Selain itu mahasiswa juga diperkenalkan dengan antarmuka Flutlab untuk membuat aplikasi Mobile multi platform. Mahasiswa diwajibkan untuk menyelesaikan **Persiapan Praktikum** sebelum masuk ke tahapan ini.

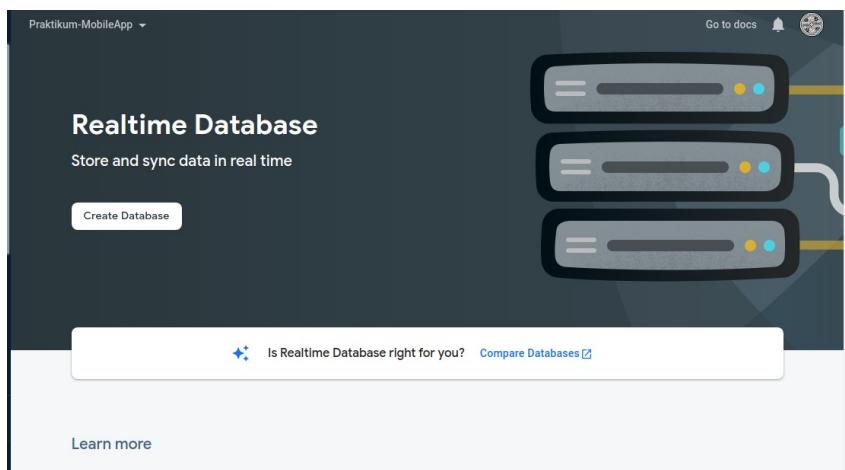
1.2 Tutorial

1. Buka firebase.google.com, login dengan **Akun Google Pribadi**. Jika sudah, klik **Get Started > Create Project** > Beri Nama Projek **Praktikum-MobileApp** > Klik **Continue** > Matikan **Analytic** > **Create Project** > Tunggu Proses Selesai.

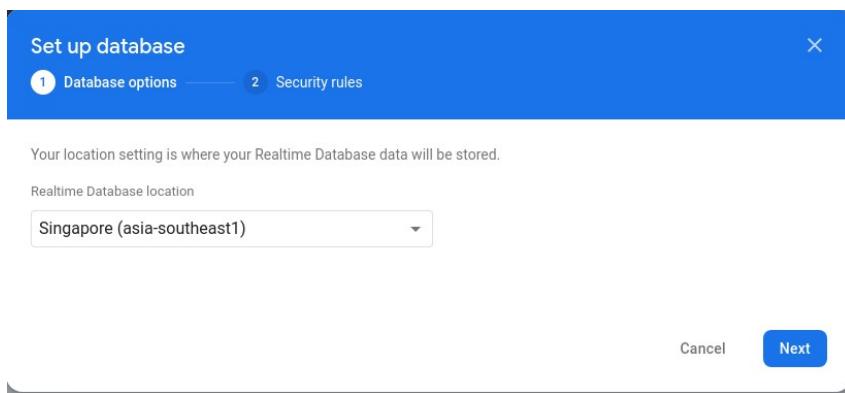




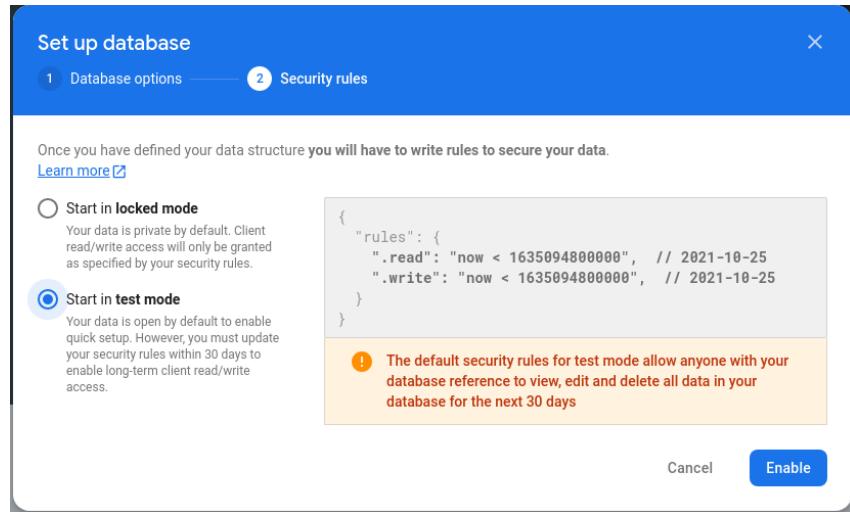
2. Untuk membuat **Realtime Database**, klik **Menu Build** yang ada di samping kiri > pilih **Realtime Database** > klik **Create Database**



3. Pilih lokasi database (**Singapore**), klik **Next**



4. Ketika ditanya **Security Rules**, pilih **Locked Mode / Private** atau **Test Mode / Public Temporary**. Pilih **Test Mode**.



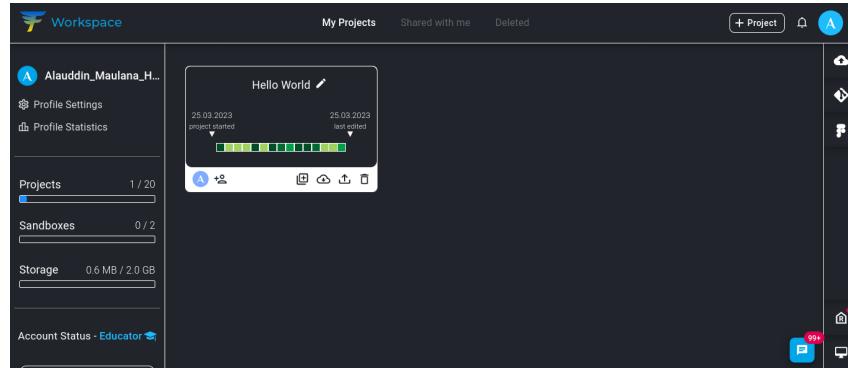
5. Tunggu hingga database muncul dan siap dipakai.

6. Setelah Database sudah siap, berikutnya adalah mengonfigurasikan **Rules** yang sudah dibuat oleh Firebase. Hal ini memastikan database bisa diakses kapanpun
7. Klik tab **Rules** → **Edit rules** → Ubah data berikut :

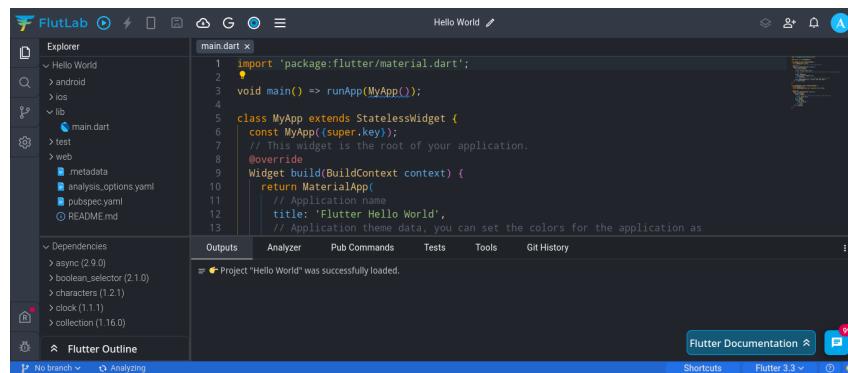
- "now < xxxxxxxxxxxx" menjadi true

8. Klik **Publish** untuk menerapkan perubahan.

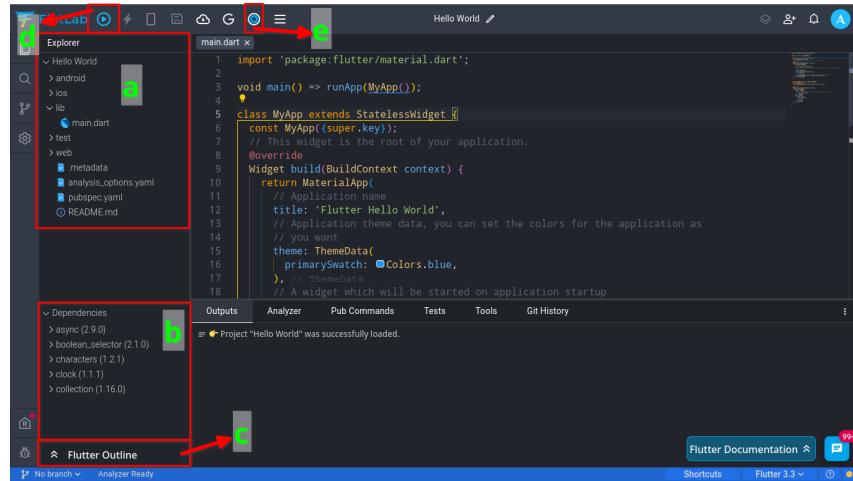
9. Berikutnya buka web <https://flutlab.io/workspace>. Buatlah akun dengan **E-Mail Pribadi**, dan buka kembali link tersebut hingga terlihat seperti gambar di bawah:



10. Mahasiswa akan melihat **Project Default** berisikan **Hello World** dari Flutlab. Klik projek tersebut hingga muncul tampilan seperti berikut:



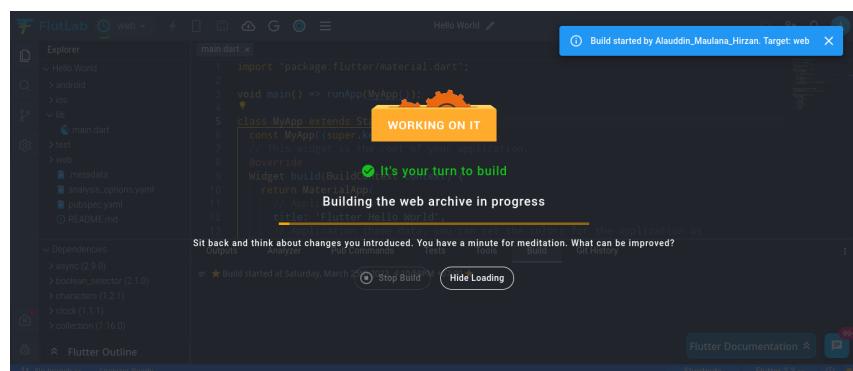
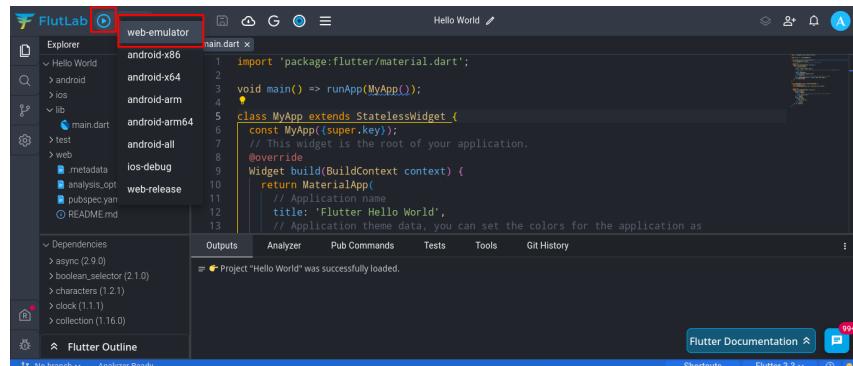
11. Ada beberapa bagian yang harus dipahami oleh mahasiswa:
- Struktur Folder
 - Ketergantungan Aplikasi
 - Navigasi Item Aplikasi
 - Build dan Play Aplikasi
 - Analyzer Kebenaran Kode



12. Untuk menguji coba aplikasi dari projek saat ini bisa dalam beberapa mode:

- Web (Rekomendasi untuk preview)
- Android APK (1 kali per hari, hanya untuk final)
- IPhone IPS (1 kali per hari, hanya untuk final)

13. Arahkan Mouse ke Tombol **Play** → Pilih **Web**. Jika sudah dalam mode **Web**. Klik tombol **Play** dan tunggu proses building hingga emulator web muncul dan aplikasi ditampilkan.



```

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // Application title
      title: 'Flutter Demo Home Page',
      // ...
    );
  }
}

```

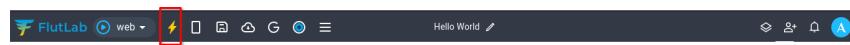
14. Scroll turun **Kode** dan temukan tulisan 'Hello, World!' dan ubah menjadi **NIM - Nama**

```

return Scaffold(
  appBar: AppBar(
    // The title text which will be shown on the action bar
    title: Text(title),
  ),
  body: Center(
    child: Text(
      '6.211.10.0001 - Ilham',
    ),
  ), // Center
); // Scaffold

```

15. Klik **Hot Reload** (Simbol Petir disebelah tombol Play) untuk melakukan **Refresh**. Hanya bisa digunakan ketika tombol **Play** sudah di klik satu (1) kali.



16. Secara otomatis perubahan akan ditampilkan tanpa perlu **Building Ulang**

```

return Scaffold(
  appBar: AppBar(
    // The title text which will be shown on the action bar
    title: Text(title),
  ),
  body: Center(
    child: Text(
      '6.211.10.0001 - Ilham',
    ),
  ), // Center
); // Scaffold

```

17. Screenshot hasil dari praktikum ini dan kirimkan ke E-Learning

Bab 2

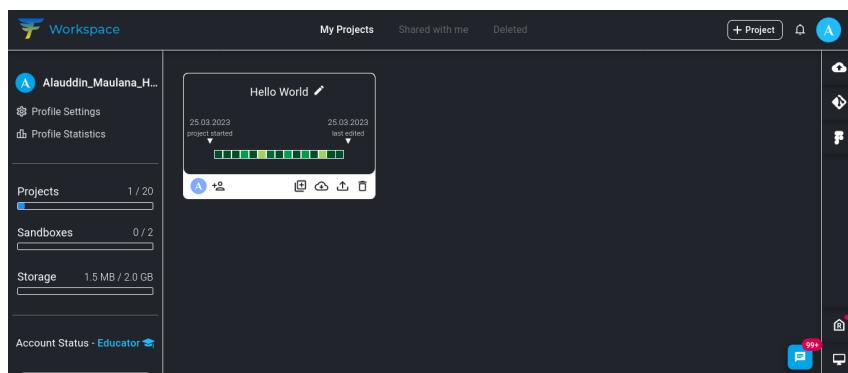
Praktikum 2

2.1 Antarmuka Login Flutter

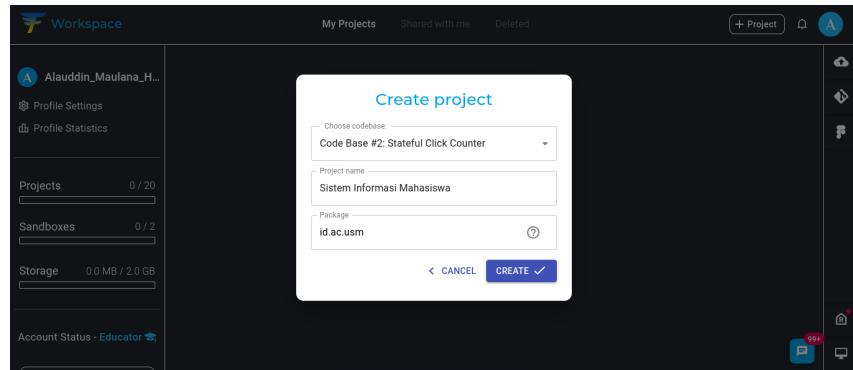
Di bagian ini mahasiswa diperkenalkan dengan pemrograman DART untuk membuat tampilan login melalui IDE Flutter. Mahasiswa diwajibkan untuk menyelesaikan pertemuan sebelumnya agar praktikum ini berjalan dengan baik.

2.2 Tutorial

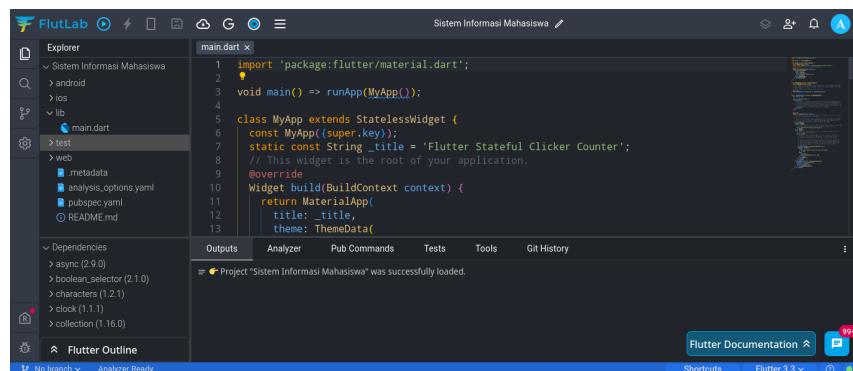
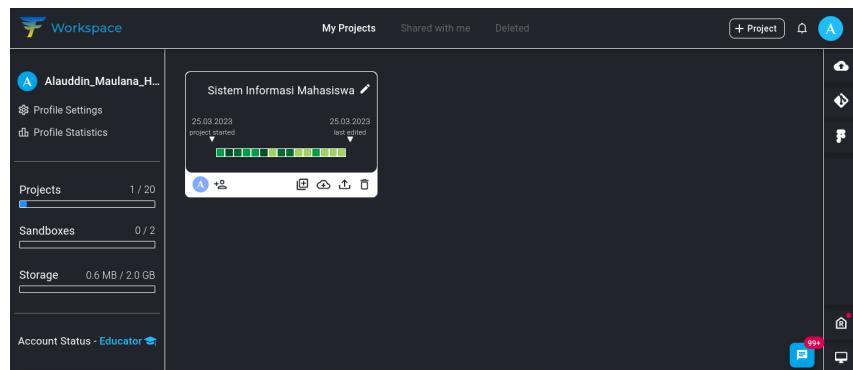
1. Buka web <https://flutlab.io/workspace>, hapus projek sebelumnya karena limitasi dari Flutlab.



2. Lalu buatlah projek dengan konfigurasi lalu klik **Create**:
 - **Codebase** : Code Base #2 : Stateful Click Counter
 - **Project Name** : Sistem Informasi Mahasiswa
 - **Package** : id.ac.usm (Domain USM)



3. Tunggu Flutlab membuat projek awal. Lalu klik projek tersebut



4. Di kode ada bagian-bagian yang perlu dirubah. Ubahlah judul aplikasi dengan mengganti kode berikut (Sebelum → Sesudah):

Potongan Kode (Sebelum)

```
static const String _title = 'Flutter StatefulWidget Counter';
```

Potongan Kode (Sesudah)

```
static const String _title = 'Sistem Informasi Mahasiswa';
```

```

main.dart M
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  static const String _title = 'Sistem Informasi Mahasiswa';
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: _title,
      theme: ThemeData(
        ...
      ),
    );
  }
}

```

5. Berikutnya adalah mengubah tampilan **Judul App** yang ditampilkan oleh aplikasi

Potongan Kode (Sebelum)

title: Text('Flutter Demo Click Counter'),

Potongan Kode (Sesudah)

title: Text('Sistem Informasi Mahasiswa'),

```

main.dart M
// than having to individually change instances of widgets.
return Scaffold(
  appBar: AppBar(
    // Here we take the value from the MyHomePage object that was created by
    // the App.build method, and use it here to set this appbar title.
    title: Text('Sistem Informasi Mahasiswa'),
  ),
  body: Center(
    // Center is a layout widget. It takes a single child and positions it
    // in the middle of the parent.
    child: Column(
      // Column is also a layout widget. It takes a list of children and
      // arranges them vertically. By default, it sizes itself to fit its
      ...
    ),
  ),
)

```

6. Berikutnya adalah mengosongkan widget-widget yang ada di tengah aplikasi, cukup menghapus kode yang berada di dalam **children: <Widget>[]**(Bisa dilihat via **Flutter Outline**)

7. Hapus bagian yang ditandai di gambar berikut

```

main.dart M
 mainAxisAlignment: MainAxisAlignment.center,
 children: <Widget>[
  Text(
    'You have pushed the button this many times:',
  ),
  Text(
    '$_counter',
    style: TextStyle(fontSize: 25),
  ),
],
);
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: Icon(Icons.add),
),
);
}

```

8. Jika sudah dihapus akan terlihat seperti berikut (children: <Widget>[] sudah tidak memiliki widget apapun):

```

56 // the App.build method, and use it to set our appbar title.
57 ),
58 // This is the root of your application.
59 body: Center(
60   // A Container is a layout widget. It takes a single child and positions it
61   // in the middle of the parent.
62   child: Column(
63     mainAxisAlignment: MainAxisAlignment.center,
64     children: <Widget>[]),
65   ),
66   // Center
67 ),
68 ); // Scaffold
69 );
70 }
71 }

```

9. Sebelum mulai memasukkan kode tampilan aplikasi, ada beberapa bagian kode yang dirubah (Abaikan Error sampai kode selesai diubah):

- Ubah **home: MyHomePage()** menjadi **home: LoginPage()** (Baris 16)
- Ubah **class MyHomePage** menjadi **class LoginPage** (Baris 21)
- Ubah **const MyHomePage** menjadi **const LoginPage** (Baris 22)
- Ubah **_MyHomePageState createState() => _MyHomePageState();** menjadi **_LoginPageState createState() => _LoginPageState();** (Baris 28)
- Ubah **class _MyHomePageState extends State<MyHomePage>** menjadi **class _LoginPageState extends State<LoginPage>** (Baris 31)

```

8 // This widget is the root of your application.
9 @override
10 Widget build(BuildContext context) {
11   return MaterialApp(
12     title: _title,
13     theme: ThemeData(
14       primarySwatch: Colors.blue,
15     ),
16     home: LoginPage(),
17   );
18 }
19 }
20

```

10. Hapus variabel serta fungsi yang ada di class **_LoginPageState**

```

31 class _LoginPageState extends State<LoginPage> {
32   int _counter = 0;
33
34   void _incrementCounter() {
35     setState(() {
36       // This call to setState tells the Flutter framework that something has
37       // changed in this State, which causes it to rerun the build method below
38       // so that the display can reflect the updated values. If we changed
39       // _counter without calling setState(), then the build method would not be
40       // called again, and so nothing would appear to happen.
41       _counter++;
42     });
43   }
44
45   @override
46   Widget build(BuildContext context) {
47     ...
48   }
49 }

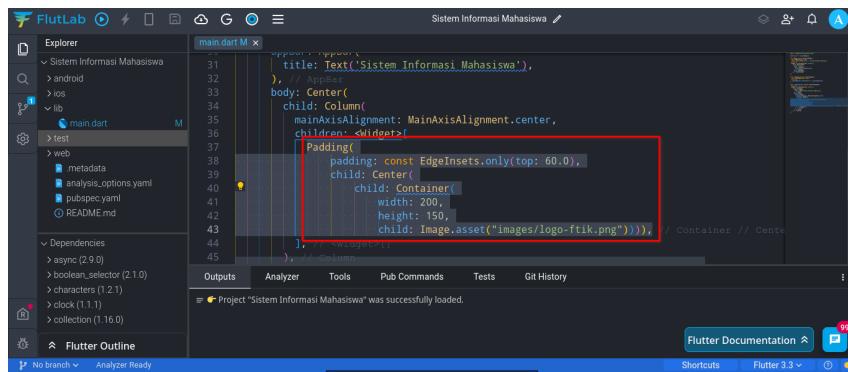
```

11. Kode siap untuk ditambahkan widget-widget baru. Masukkan kode berikut di dalam `children: <Widget>` yang sudah dikosongkan sebelumnya. Pastikan memasukkan kode di dalam tag []:

Potongan Kode

Padding(

```
padding: const EdgeInsets.only(top: 60.0),
child: Center(
    child: Container(
        width: 200,
        height: 150,
        child: Image.asset("images/logo-ftik.png"))),
```

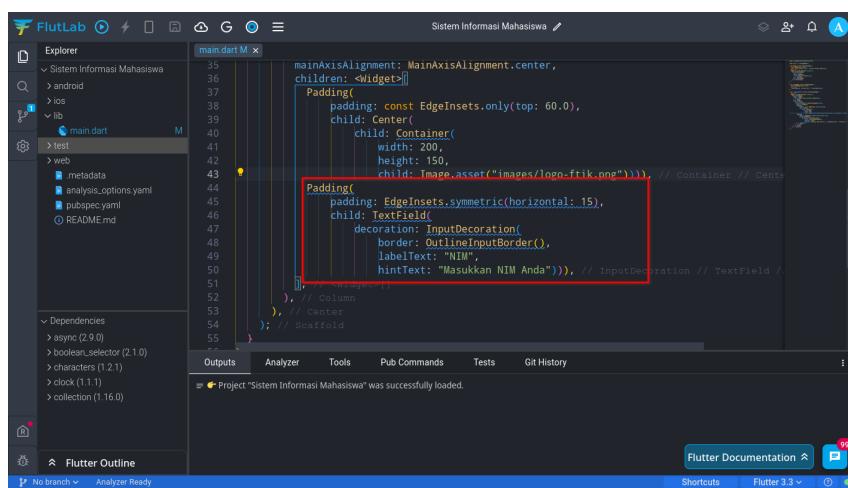


12. Kemudian tambahkan kode berikut masih di dalam tag [] tetapi di bawah kode sebelumnya. Kode ini digunakan untuk input NIM mahasiswa

Potongan Kode

Padding(

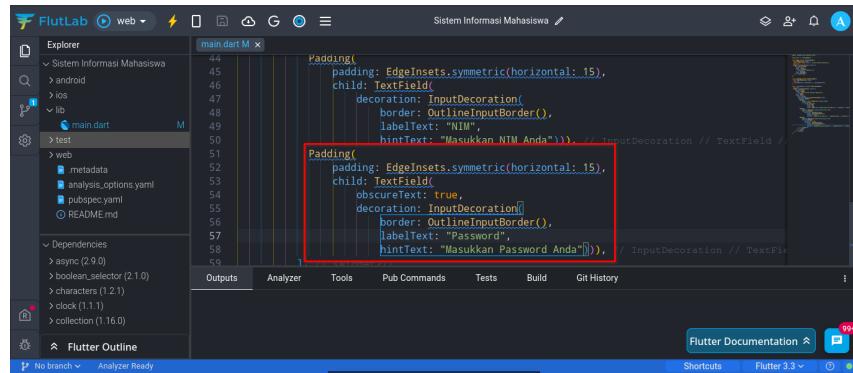
```
padding: EdgeInsets.symmetric(horizontal: 15),
child: TextField(
    decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: "NIM",
        hintText: "Masukkan NIM Anda")),
);
```



13. Berikutnya menambahkan kode input untuk password di bawah kode sebelumnya

Potongan Kode

```
Padding(  
    padding: EdgeInsets.symmetric(horizontal: 15),  
    child: TextField(  
        obscureText: true,  
        decoration: InputDecoration(  
            border: OutlineInputBorder(),  
            labelText: "Password",  
            hintText: "Masukkan Password Anda"))),
```



14. Setelah field input selesai, berikutnya tambahkan tombol untuk **Login**

Potongan Kode

```
Container(  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.blue,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Firebase Login  
        },  
        child: Text(  
            "Login",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
        )),
```

```

main.dart M
58     hintText: "Masukkan Password Anda")), // InputDecoration // TextFormField
59
60     height: 50,
61     width: 200,
62     decoration: BoxDecoration(
63       color: Colors.blue,
64       borderRadius: BorderRadius.circular(15)), // BoxDecoration
65     child: TextButton(
66       onPressed: () {
67         //TODO Firebase login
68       },
69       child: Text(
70         "Login",
71         style: TextStyle(color: Colors.white, fontSize: 25),
72       ), // TextButton // Container
73   ], // <Widget>]
74   ), // Column
75   ), // Center
76   ); // Scaffold
77 }
78 }

```

15. Terakhir adalah menambahkan kode untuk Register dengan widget jenis **TextButton**

Potongan Kode

```

TextButton(
  onPressed: () {
    // TODO Firebase Register
  },
  child: Text("Daftar Akun Mahasiswa",
  style: TextStyle(color: Colors.blue, fontSize: 15)))

```

```

main.dart M
58     hintText: "Masukkan Password Anda")), // InputDecoration // TextFormField
59
60     height: 50,
61     width: 200,
62     decoration: BoxDecoration(
63       color: Colors.blue,
64       borderRadius: BorderRadius.circular(15)), // BoxDecoration
65     child: TextButton(
66       onPressed: () {
67         // TODO Firebase Register
68       },
69       child: Text("Daftar Akun Mahasiswa",
70         style: TextStyle(color: Colors.blue, fontSize: 15))
71     ), // TextButton // Container
72   ], // <Widget>]
73   ), // Column
74   ), // Center
75   ); // Scaffold
76 }
77 }

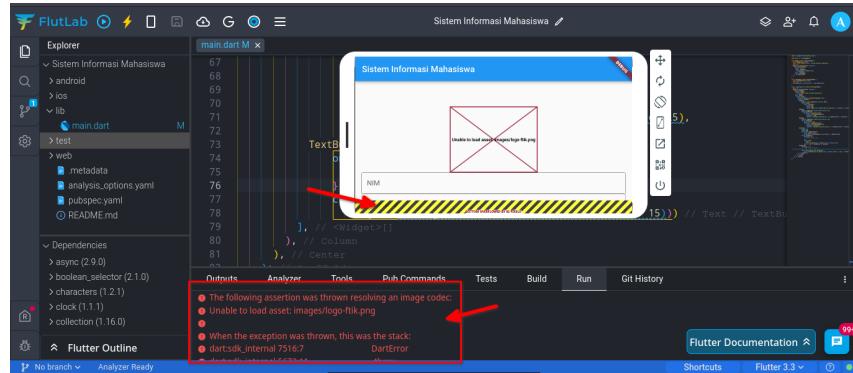
```

16. Cek pastikan kode tidak ada yang error. Jika ada perbaiki kode tersebut. Jika tidak ada error. Klik Play (mode Web) untuk melihat hasil saat ini

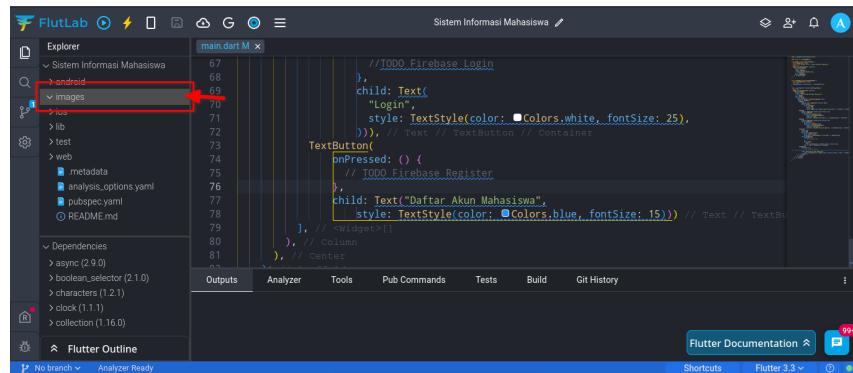
The screenshot shows the Flutter application running in the browser. The login screen displays a placeholder image for the logo ('UnlabeledImage'), a text input field for 'NIM', a password input field, and a 'Login' button. Below the screen, the IDE interface shows the code and the 'Run' button highlighted.

17. Tetapi banyak error yang ada di aplikasi tersebut seperti:

- Gambar tidak ada
- Tampilan rusak ketika dirotasi landscape



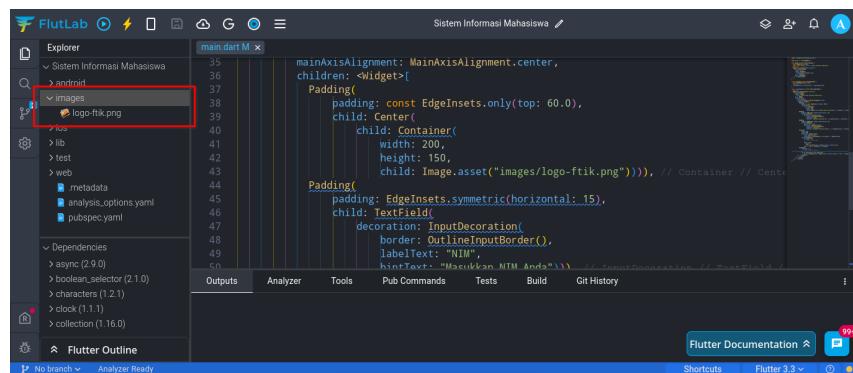
18. Untuk menambahkan gambar cukup dengan cara mudah. Buat **Folder Baru** di **Projek Sistem Informasi Mahasiswa** dengan nama **images**



19. Lalu unduh gambar dan rename ke **logo-ftik.png**. Unduh di

- <https://transit.ftik.usm.ac.id/uploads/assets/img/ftik-usm.png>

20. Unggah file yang sudah di-rename ke folder **images** tersebut



21. Berikutnya adalah mengkonfigurasi **pubspec.yaml** agar bisa membaca gambar, **Uncomment assets** dan tambahkan **- images**. Lihat Gambar di bawah:

```

    # To add assets to your application, add an assets section, like this:
    assets:
      - images/

```

22. Hot Reload aplikasi untuk menampilkan gambar. Jika tidak muncul, pastikan dilakukan dengan benar.

```

    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      Padding(
        padding: const EdgeInsets.all(15.0),
        child: Container(
          width: 200,
          height: 150,
          child: Image.asset("images/logo-ftik.png"))),
    ],
  ),
)

```

23. Langkah terakhir adalah mengubah layout agar memungkinkan untuk di Scroll. Perhatikan berikut:

- Ubah **body: Center** → menjadi **body: SingleChildScrollView**
- Hapus **mainAxisAlignment: MainAxisAlignment.center,**

```

    body: Center(),
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      Padding(
        padding: const EdgeInsets.all(15.0),
        child: Container(
          width: 200,
          height: 150,
          child: Image.asset("images/logo-ftik.png"))),
    ],
  ),
)

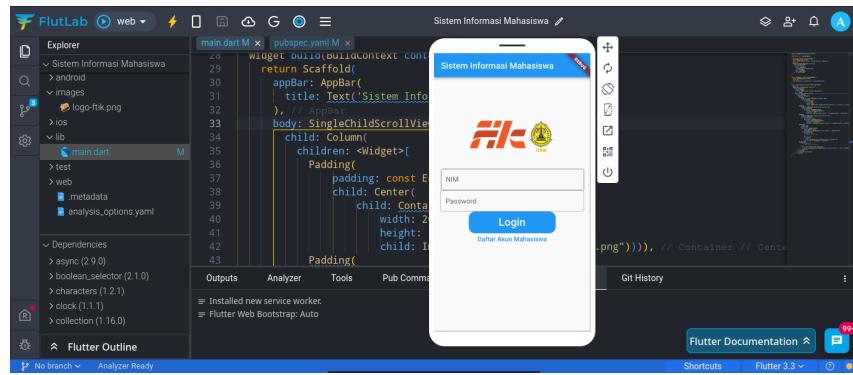
```

The screenshot shows the FlutLab IDE interface. On the left is the Explorer panel with project files like 'main.dart', 'images', 'lib', and 'Dependencies'. The main area displays the code for 'main.dart':

```
main.dart M pubspec.yaml M
26:     widget build(BuildContext context) {
27:       return Scaffold(
28:         appBar: AppBar(
29:           title: Text('Sistem Informasi Mahasiswa'),
30:         ),
31:         body: SingleChildScrollView(
32:           child: Column(
33:             mainAxisAlignment: MainAxisAlignment.start,
34:             children: <Widget>[
35:               Padding(
36:                 padding: const EdgeInsets.only(top: 60.0),
37:                 child: Center(
38:                   child: Container(
39:                     width: 200,
40:                     height: 150,
41:                     child: Image.asset("images/logo-ftik.png"))),
42:               ),
43:             ],
44:           padding: const EdgeInsets.all(16.0),
45:         ),
46:       );
47:     }
48:   }

49:   void main() {
50:     runApp(SistemInformasiMahasiswa());
51:   }
52: }
```

24. Build Ulang Aplikasi untuk melihat hasil akhirnya



25. Aplikasi yang sudah jadi kemudian di **Screenshot** dan kirimkan ke **E-Learning**

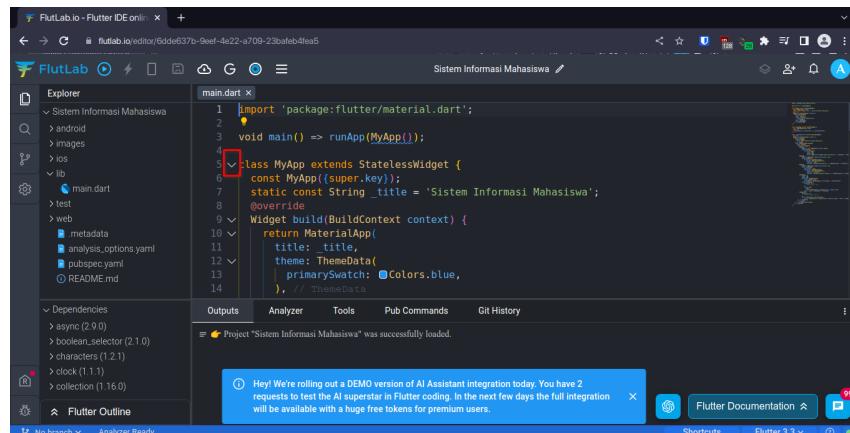
Bab 3

Praktikum 3

3.1 Halaman Registrasi dan Navigasi Page

3.2 Tutorial

1. Buka projek yang sebelumnya sudah dibuat
2. Sebelum memulai masuk ke pemrograman, lipat dulu **class** yang ada untuk mempermudah melakukan pemrograman. Klik **Tombol** yang ditunjukkan oleh Gambar berikut



3. Lipat semua **class** yang ada di kode agar terlihat seperti berikut

```

1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 > class MyApp extends StatelessWidget {
6   ...
7 }
8 > class LoginPage extends StatefulWidget {
9   ...
10 }
11 > class _LoginPageState extends State<LoginPage> {
12   ...
13 }
14

```

4. Berikutnya masukkan kode **class** untuk pembuatan tampilan **Pendaftaran** seperti berikut:

Potongan Kode

```

class RegisterPage extends StatefulWidget {
  const RegisterPage({super.key});
  @override
  _RegisterPageState createState() => _RegisterPageState();
}

```

```

20 > class LoginPage extends StatefulWidget {
21   ...
22 }
23
24 > class _LoginPageState extends State<LoginPage> {
25   ...
26 }
27
28 > class RegisterPage extends StatefulWidget {
29   ...
30 }
31
32 > class _RegisterPageState extends State<RegisterPage> {
33   ...
34 }
35
36 > class _RegisterPageState extends State<RegisterPage> {
37   ...
38 }
39

```

5. Abaikan **Error** saat ini. Untuk menghilangkan **Error** itu, masukkan kode berikut:

Potongan Kode

```

class _RegisterPageState extends State<RegisterPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold();
  }
}

```

```

class _RegisterPageState extends State<RegisterPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold();
  }
}

```

6. Kode **widget-widget** akan dimasukkan ke dalam **return Scaffold()**. Pertama masukkan **appBar** yang digunakan untuk menampilkan **Judul Halaman** seperti berikut:

Potongan Kode

```

appBar: AppBar(
  title: Text('Pendaftaran Data Mahasiswa'),
),

```

```

class _RegisterPageState extends State<RegisterPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Pendaftaran Data Mahasiswa'),
      ),
    );
  }
}

```

7. Berikutnya adalah menambahkan widget layout yang dapat di Scroll beserta wadah **Widget** nantinya. Perhatikan Kode dan Gambar berikut:

Potongan Kode

```

body: SingleChildScrollView(
  child: Column(
    children: <Widget>[],
  )
)

```

The screenshot shows the FlutLab IDE interface. The top bar displays the URL `flutlab.io/editor/6dd6e37b-9efc-4e22-a709-23bafe8fea5`. The main workspace shows a Dart file named `main.dart` with the following code:

```
92 @override
93 Widget build(BuildContext context) {
94   return Scaffold(
95     appBar: AppBar(
96       title: Text('Pendaftaran Data Mahasiswa'),
97     ),
98     body: SingleChildScrollView(
99       child: Column(
100         children: <Widget>[]),
101     ),
102   );
103 }
104 
```

A red box highlights the `body: SingleChildScrollView(...)` line. Below the code editor, the `Outputs` tab is selected, showing the message: "Project 'System Informasi Mahasiswa' was successfully loaded." A floating notification at the bottom left says: "Hey! We're rolling out a DEMO version of AI Assistant integration today. You have 2 requests to test the AI superstar in Flutter coding. In the next few days the full integration will be available with a huge free tokens for premium users." The bottom navigation bar includes links for Analyzer, Pub Commands, Tools, and Git History.

8. Sama seperti sebelumnya yang di mana kode **Widget** diletakkan di dalam kode `<Widget>[]`. Tambahkan kode **InputText** untuk **NIM** seperti berikut:

Potongan Kode

Padding(

padding:

```
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
child: TextField(  
  decoration: InputDecoration(  
    border: OutlineInputBorder(),  
    labelText: "NIM",  
    hintText: "Masukkan NIM Anda"))),
```

The screenshot shows the FlutterLab IDE interface. The main area displays the `main.dart` file with the following code:

```
child: Column(
  children: <Widget>[  
  Padding(  
    padding:  
      const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
    child: TextFormField(  
      decoration: InputDecoration(  
        border: OutlineInputBorder(),  
        labelText: "NIM",  
        hintText: "Masukkan NIM Anda"))),  
];  
}); // Column // SingleChildScrollView // Scaffold
```

A red box highlights the `decoration` property of the `TextFormField`. Below the code editor, the **Outputs** tab is selected, showing the message: "Project 'Sistem Informasi Mahasiswa' was successfully loaded." A blue banner at the bottom left provides information about AI integration.

9. Kode berikutnya untuk **InputText** dengan isi **Nama** _____ **Potongan Kode** _____

Potongan Kode

Padding(

padding:

```
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
    child: TextField(  
      decoration: InputDecoration(  
        border: OutlineInputBorder(),  
        labelText: "Nama",  
        hintText: "Masukkan Nama Anda"))),
```

```

    Padding(
      padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
      child: TextField(
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          labelText: "Nama",
          hintText: "Masukkan NIM Anda")),
    );
  }
}

```

10. Selanjutnya adalah **InputText** untuk E-Mail. Perhatikan kode berikut:

Potongan Kode

```

Padding(
  padding:
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
    child: TextField(
      decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: "Email",
        hintText: "Masukkan E-Mail Anda")),

```

```

    Padding(
      padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
      child: TextField(
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          labelText: "Nama",
          hintText: "Masukkan Nama Anda")),
    );
  }
}

```

11. InputText terakhir adalah untuk **Password**. Masukkan kode berikut:

Potongan Kode

```

Padding(
  padding:
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
    child: TextField(
      obscureText: true,
      decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: "Password",
        hintText: "Masukkan Password Anda")),

```

```

FlutLab.io - Flutter IDE onlin... + 
FlutLab Sistem Informasi Mahasiswa
main.dart M
123
124
125
126
127
128
129
130
131
132
133
134
135
136
  padding:
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
    child: TextFormField(
      obscureText: true,
      decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: "Password",
        hintText: "Masukkan Password Anda"),
    ),
);
}

```

Hey! We're rolling out a DEMO version of AI Assistant integration today. You have 2 requests to test the AI superstar in Flutter coding. In the next few days the full integration will be available with a huge free tokens for premium users.

12. Widget berikutnya adalah **Tombol**. Masukkan kode berikut untuk menambahkan tombol **Daftar**

Potongan Kode

```

Container(
  height: 50,
  width: 200,
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(15),
    child: TextButton(
      onPressed: () {
        //TODO Firebase Insert
      },
      child: Text(
        "Daftar",
        style: TextStyle(color: Colors.white, fontSize: 25),
    )));
  
```

```

FlutLab.io - Flutter IDE onlin... + 
FlutLab Sistem Informasi Mahasiswa
main.dart M
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
  border: OutlineInputBorder(),
  labelText: "Password",
  hintText: "Masukkan Password Anda"),
);
Container(
  height: 50,
  width: 200,
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(15),
    child: TextButton(
      onPressed: () {
        //TODO Firebase Insert
      },
      child: Text(
        "Daftar",
        style: TextStyle(color: Colors.white, fontSize: 25),
    )));
}

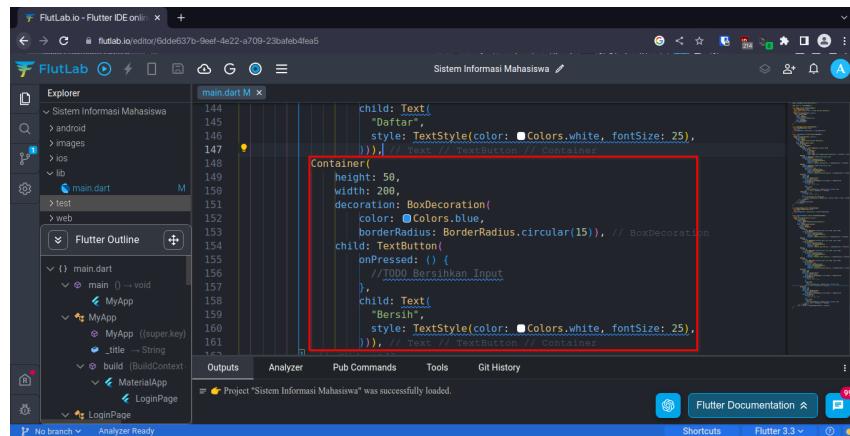
```

Run a Web Emulator Build to activate Hot Reload

13. Tombol berikutnya adalah **Bersih** untuk menghapus input dari **InputText**

Potongan Kode

```
Container(  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.blue,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Bersihkan Input  
        },  
        child: Text(  
            "Bersih",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
    )),,
```



14. Tombol terakhir adalah **Batal** yang digunakan untuk kembali ke **Menu Awal (Login)**. Masukkan kode berikut:

Potongan Kode

```
Container(  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.blue,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Batalkan Registrasi  
        },  
        child: Text(  
            "Batal",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
    )),,
```

```

Container(
  height: 50,
  width: 200,
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(15), // BoxDecoration
  ),
  child: TextButton(
    onPressed: () {
      // TODO: Database Login
    },
    child: Text(
      "Batal",
      style: TextStyle(color: Colors.white, fontSize: 25),
    ),
  ),
)

```

15. Berikutnya adalah memberi fitur **Navigasi Halaman** agar aplikasi dapat berpindah ke halaman lainnya. Kembali ke bagian `_LoginPage` dan temukan `TextButton` registrasinya. Gunakan **Flutter Outline** di sebelah **Kiri Bawah** untuk mempermudah pencarian, dan **Klik** target tujuan:

```

TextButton(
  onPressed: () {
    // TODO: Firebase Register
  },
  child: Text("Daftar Akun Mahasiswa",
    style: TextStyle(color: Colors.blue, fontSize: 15)),
)

```

16. Di bagian dalam `onPressed:()`, masukkan kode berikut setelah **Komentar TODO: Potongan Kode**

```

Potongan Kode
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => new RegisterPage())));

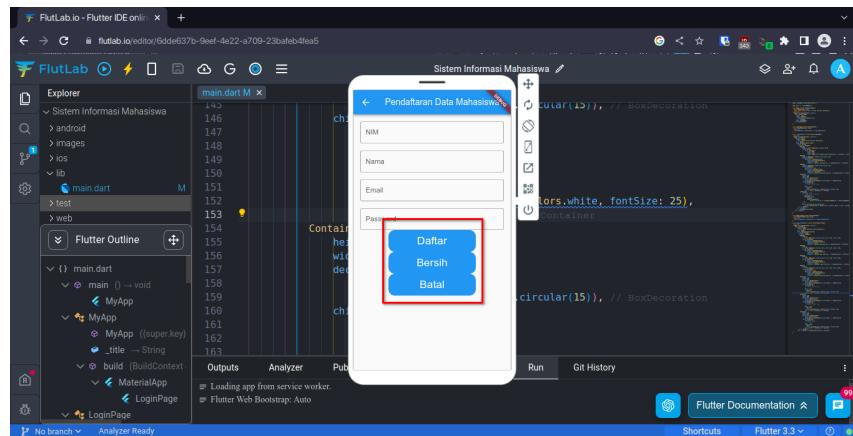
```

```

TextButton(
  onPressed: () {
    // TODO: Firebase Register
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => new RegisterPage()),
    );
    child: Text("Daftar Akun Mahasiswa",
      style: TextStyle(color: Colors.blue, fontSize: 15)),
)

```

17. Build app yang sudah dibuat dan cek hasilnya

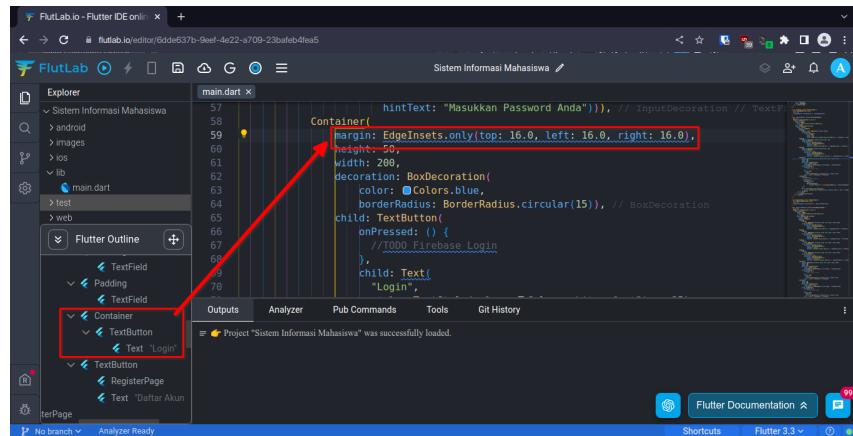


18. Karena **Tombol** terlalu dekat satu sama lainnya, maka tambahkan kode berikut ke semua **Tombol** yang ada:

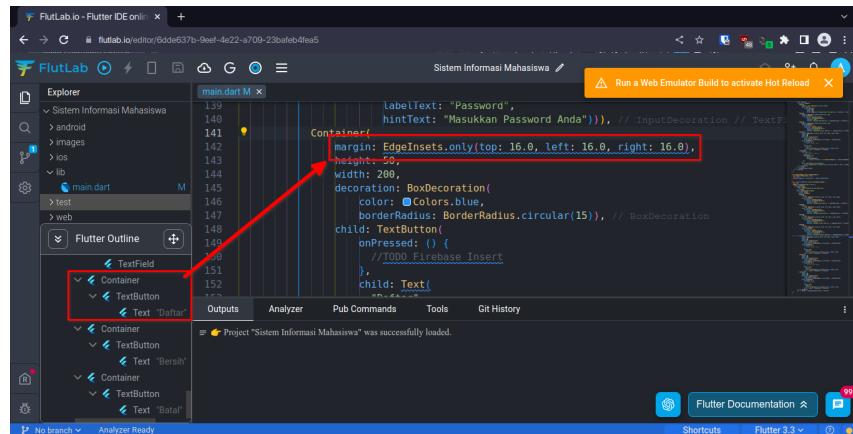
Potongan Kode

margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),

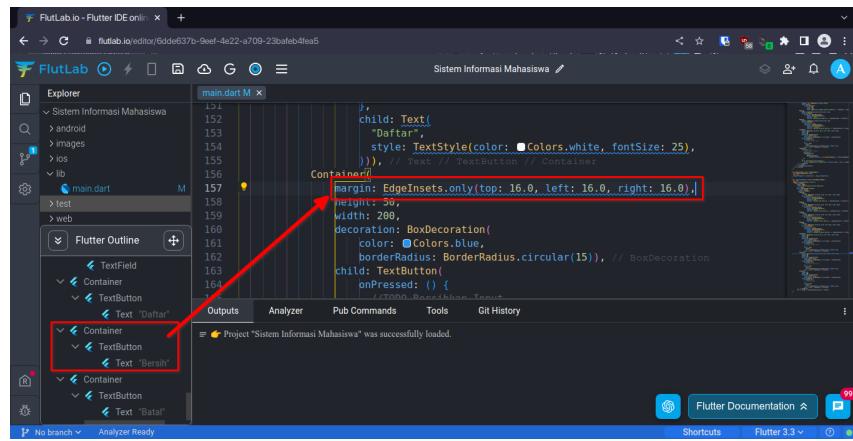
- Tombol Login



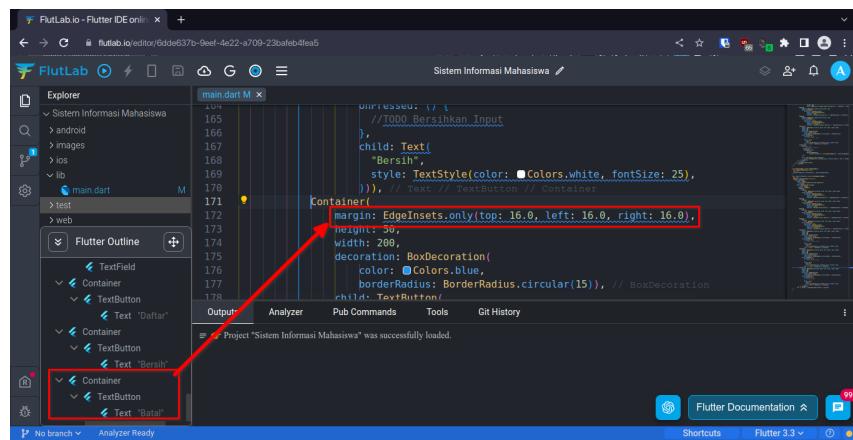
- Tombol Daftar



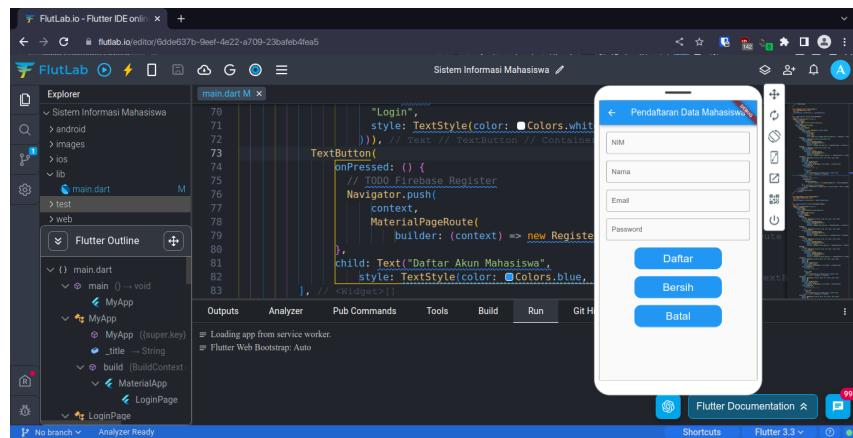
- Tombol Bersih



- Tombol Batal



19. Cek kembali hasilnya dan **Screenshot** hasil aplikasi saat ini dan kirimkan ke **E-Learning**



Bab 4

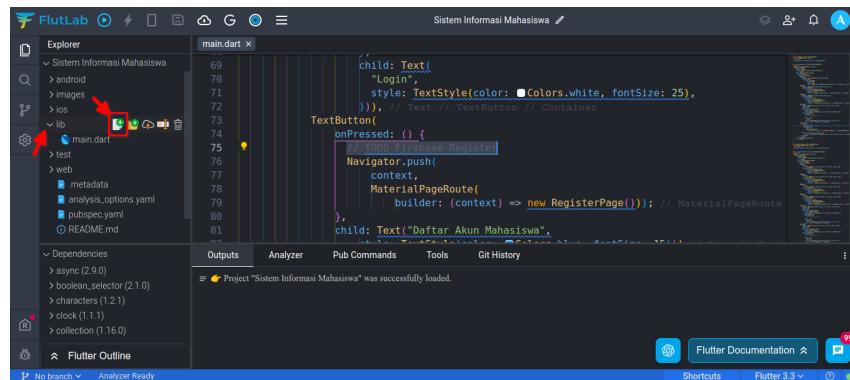
Praktikum 4

4.1 Integrasi Firebase

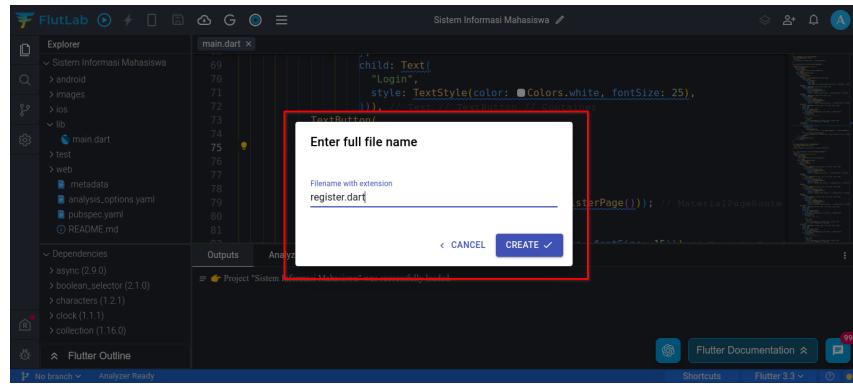
Di bagian ini mahasiswa diperkenalkan bagaimana melakukan pembersihan baris kode yang tidak diperlukan, integrasi ke Realtime Database, dan melakukan operasi register data. Mahasiswa diwajibkan untuk menyelesaikan Praktikum 3

4.2 Tutorial

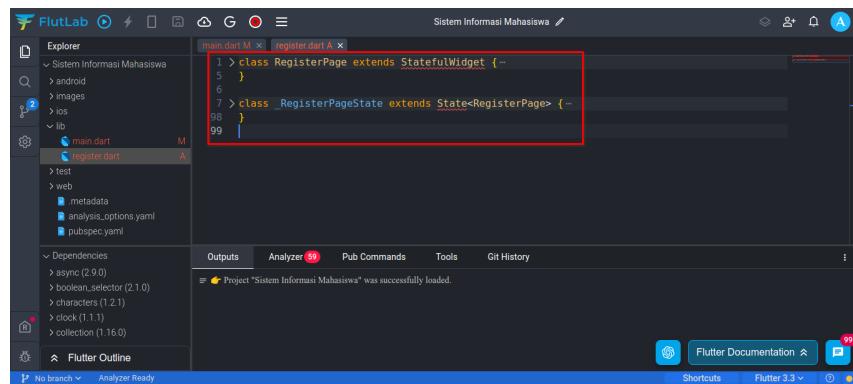
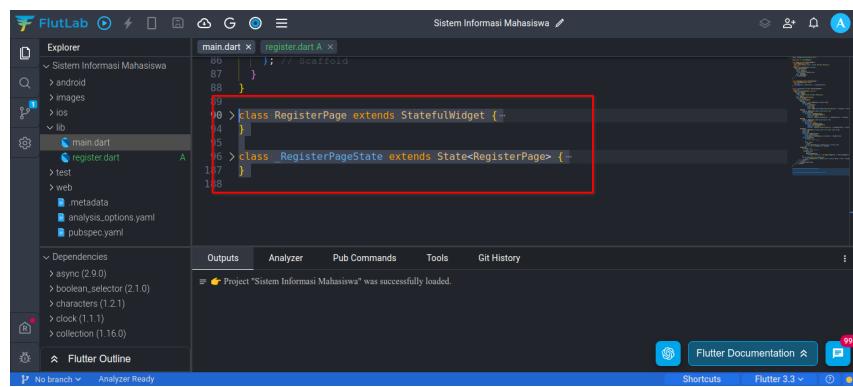
1. Buka projek sebelumnya, dan pastikan untuk membuka projek yang tepat
2. Berikutnya adalah memisahkan halaman **Register** ke file terpisah. Arahkan **Mouse** ke folder **lib**, dan klik **Create File**



3. Beri nama **register.dart** sebagai file baru, dan klik **Create**



- File **register.dart** akan terbuka, lalu **PINDAHKAN** kode berikut dari **main.dart** ke **register.dart**. Lipat kode untuk mempermudah pemindahan



- Pastikan TIDAK ADA** kode **Register** di file **main.dart**. Hal ini untuk mencegah terjadinya error di pertemuan-pertemuan berikutnya.
- Kedua file akan mengalami **Error**. Untuk memperbaiki kode **register.dart** tambahkan satu baris kode ini di **Baris 1**

Potongan Kode

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

```

1 import 'package:flutter/material.dart';
2
3 > class RegisterPage extends StatefulWidget {
4
5   ...
6
7   > class _RegisterPageState extends State<RegisterPage> {
8
9     ...
10
11   }
12
13   ...
14
15 }

```

7. Lalu untuk memperbaiki kode **main.dart** cukup tambahkan kode berikut di **Baris 2**

Potongan Kode

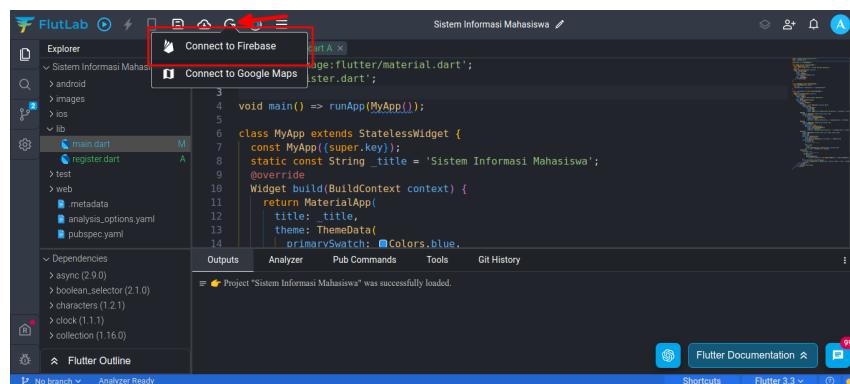
import 'register.dart';

```

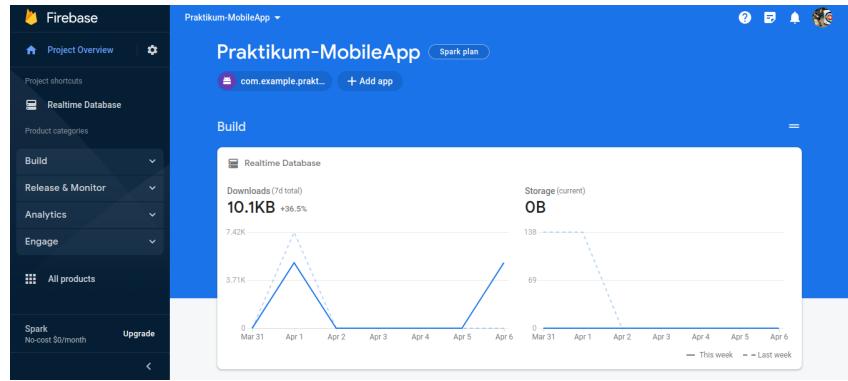
1 import 'package:flutter/material.dart';
2 import 'register.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   const MyApp({super.key});
8   static const String _title = 'Sistem Informasi Mahasiswa';
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: _title,
13      theme: ThemeData(
14        primarySwatch: Colors.blue,

```

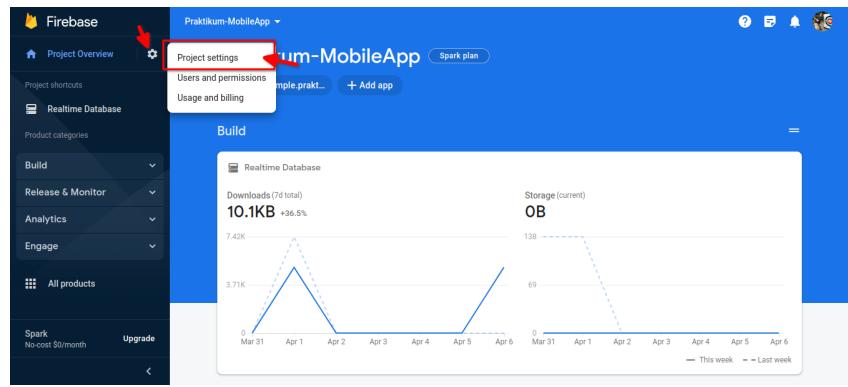
8. Semua **Error** sudah hilang, berikutnya melakukan integrasi ke **Firebase**. Klik Icon **G**, lalu pilih **Connect to Firebase**



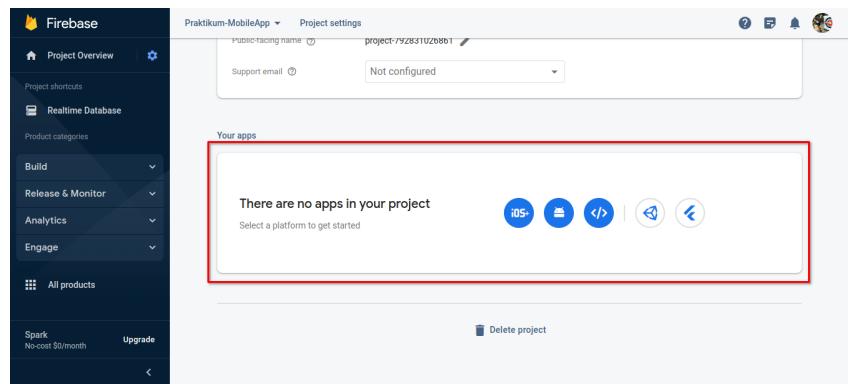
9. Flutlab akan menampilkan pilihan platform App yang menjadi target. Sebelum memilih buka website <https://firebase.google.com>. Login ke Google apabila belum, Klik **Go to console** di atas kanan. Pilih **Projek** yang sudah dibuat sebelumnya. Hingga tampil dasbor Firebase



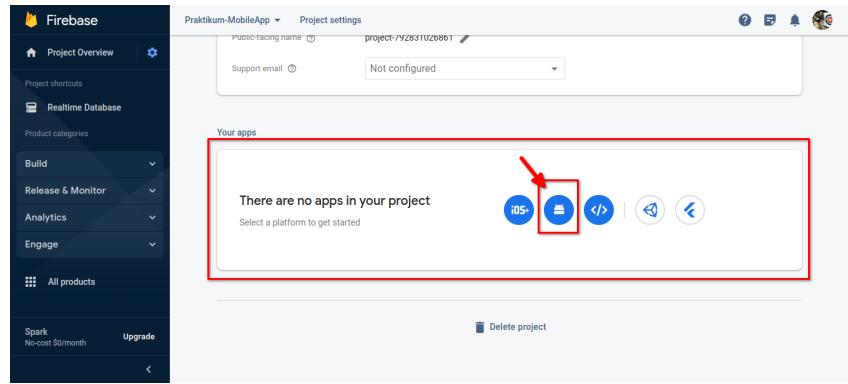
10. Untuk meng-konfigurasikan, klik **Roda Gigi** di samping **Project Overview**, lalu pilih **Project Settings**



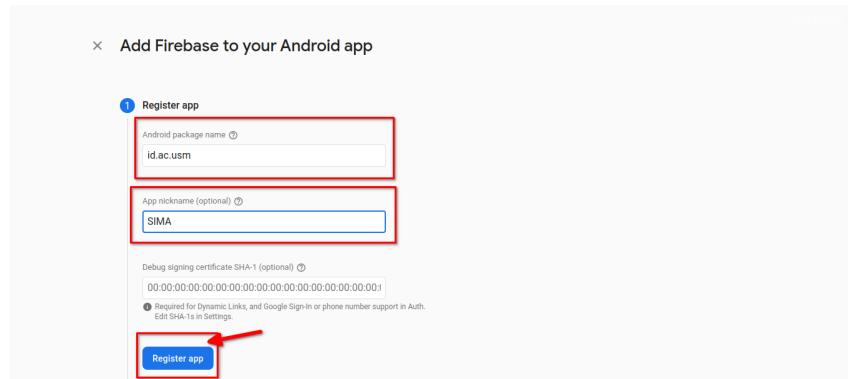
11. Scroll ke bawah dan cari panel **Your apps**



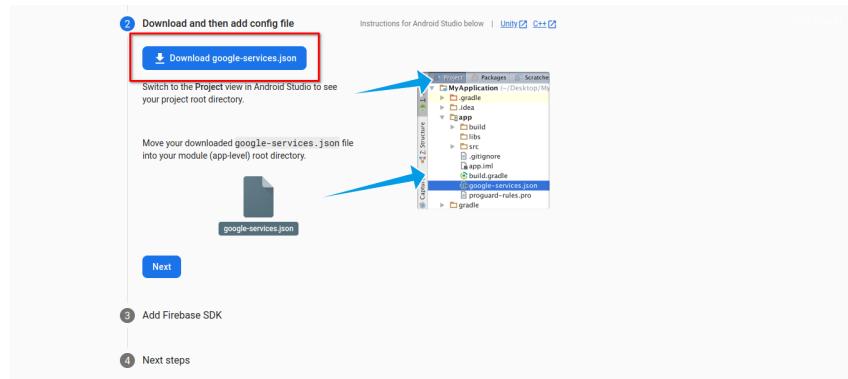
12. Pilih **Android**



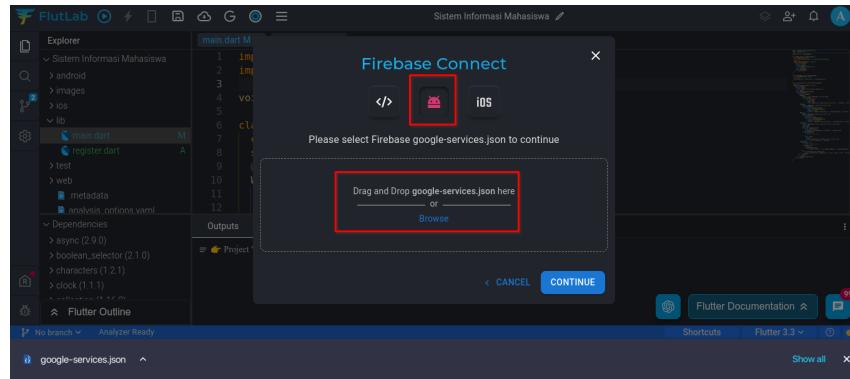
13. Lalu **Firebase** akan menampilkan **Wizard** untuk menambahkan App baru. Masukkan **id.ac.usm** sebagai **Android package name** dan **SIMA** sebagai **App nickname**. Klik **Register app**



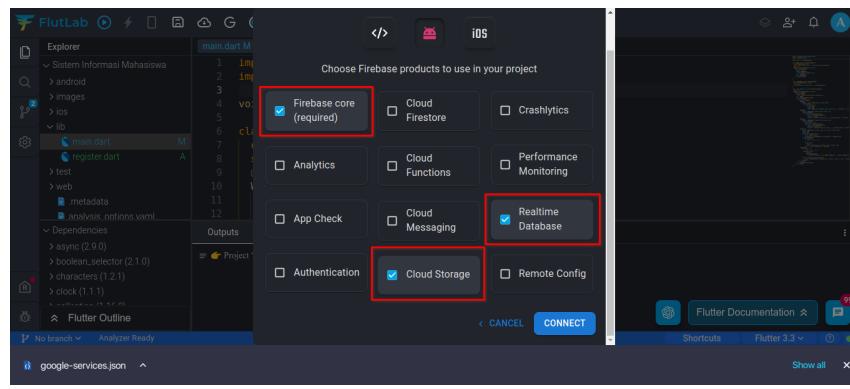
14. Langkah berikutnya adalah mengunduh file **google-services.json**. Cukup klik **Tombol** yang disediakan



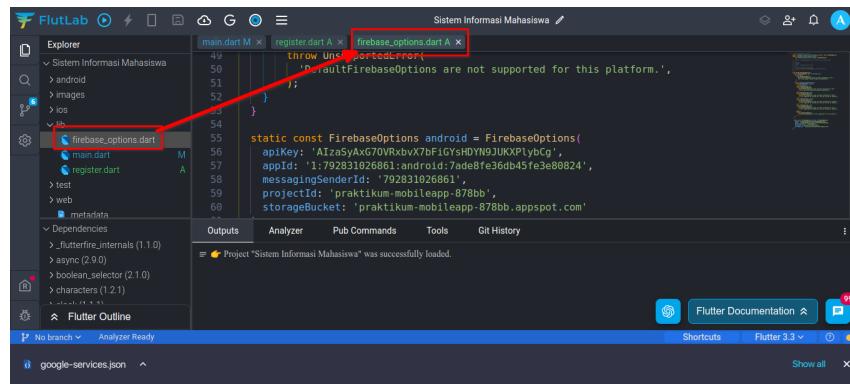
15. Tutup Wizard jika sudah selesai mengunduh
 16. Kembali ke **FlutLab**, pilih **Android** lalu unggah file **google-services.json**. Klik **Continue** sesudah unggah



17. Berikutnya pilih **Services** yang akan dipakai. Pilih **Core**, **Realtime Database**, dan **Cloud Storage**. Terakhir klik **Connect**



18. File baru akan ditambahkan dengan nama **firebase_options.dart**. File ini menjadi bukti kalau konfigurasi dan integrasi **Firebase** sudah sukses



19. Untuk mengecek apakah dependency dasar sudah terpenuhi, buka file **pubspec.yaml** dan cari baris teks seperti berikut:

- **firebase_core**
- **firebase_database**
- **firebase_storage**

20. Jika ada yang belum terpasang di dalam projek, bisa di atasi dengan menggunakan bantuan **Pub Command** di bagian bawah. Klik **Pub Add** dan masukkan **item** di langkah 19. Satu per satu dan gunakan **Enter**
21. Jika sudah dimasukkan ke dalam **yml** maka langkah berikutnya adalah menekan tombol **Pub Get** untuk mengunduh secara otomatis.
22. **Screenshot** dan kirimkan ke **E-Learning**

Bab 5

Praktikum 5

5.1 Operasi Insert Data Flutter

Di bagian ini mahasiswa diajarkan bagaimana melakukan input dan penarikan data serta melakukan query ke Firebase. Mahasiswa diwajibkan untuk menyelesaikan Praktikum 4

5.2 Tutorial

1. Buka kembali Website flutlab.io. Lalu buka file `register.dart`
2. Cari baris `class _RegisterPageState extends State<RegisterPage>`, lalu tambahkan kode berikut tepat di bawahnya

Potongan Kode

```
final NIMControl = TextEditingController();
final NamaControl = TextEditingController();
final EmailControl = TextEditingController();
final PasswordControl = TextEditingController();

@Override
void dispose() {
    // Clean up the controller when the widget is disposed.
    NIMControl.dispose();
    NamaControl.dispose();
    EmailControl.dispose();
    PasswordControl.dispose();
    super.dispose();
}
```

```

class _RegisterPageState extends State<RegisterPage> {
  final NIMControl = TextEditingController();
  final NamaControl = TextEditingController();
  final EmailControl = TextEditingController();
  final PasswordControl = TextEditingController();

  @override
  void dispose() {
    // Clean up the controller when the widget is disposed.
    NIMControl.dispose();
    NamaControl.dispose();
    EmailControl.dispose();
    PasswordControl.dispose();
    super.dispose();
  }
}

```

3. Tempelkan masing-masing kontroler ke **widgetnya**. Perhatikan kode dan Gambar secara teliti

- Widget **NIM** → NIMControl

Potongan Kode

controller: NIMControl,

```

body: SingleChildScrollView(
  child: Column(
    children: <Widget>[
      Padding(
        padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
        child: TextField(
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            labelText: "NIM",
            hintText: "Masukkan NIM Anda", // InputDecoration
            controller: NIMControl,
          ), // TextField // Padding
      ),
      Padding(
        padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
        child: TextField(

```

- Widget **Nama** → NamaControl

Potongan Kode

controller: NamaControl,

```

        labelText: "Nama",
        hintText: "Masukkan Nama Anda", // InputDecoration
        controller: NamaControl,
      ),
      Padding(
        padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
        child: TextField(
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            labelText: "Nama",
            hintText: "Masukkan Nama Anda", // InputDecoration
            controller: NamaControl,
          ), // TextField // Padding
      ),
      Padding(
        padding: const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
        child: TextField(

```

- Widget **Email** → EmailControl

Potongan Kode

controller: EmailControl,

- Wdget **Password** → PasswordControl
Potongan Kode
controller: PasswordControl,

- Setelah mengeset **controller** ke semua widget, kini aplikasi dapat mengambil maupun mengeset data. Scroll turun ke widget **Button Bersih** dan masukkan kode berikut:

Potongan Kode

```
//TODO Bersihkan Input  
NIMControl.text = "";  
NamaControl.text = "";  
EmailControl.text = "";  
PasswordControl.text = "";
```



```
Sistem Informasi Mahasiswa
```

```
height: 50,
width: 200,
decoration: BoxDecoration(
  color: Colors.blue,
  borderRadius: BorderRadius.circular(15)), // BoxDecoration
child: TextButton(
  onPressed: () {
    //TODO: Bersihin Input
    NIMControl.text = "";
    NamaControl.text = "";
    EmailControl.text = "";
    PasswordControl.text = "";
  },
  child: Text(
    'Bersihin',
    style: TextStyle(color: Colors.white, fontSize: 25),
  ),
),
```

NIM
Password

Login

Bersihin

Sistem Informasi Mahasiswa

register.dart M

Outputs Analyzer Pub Commands Tools Build Run Git History

Flutter Outline Flutter Documentation

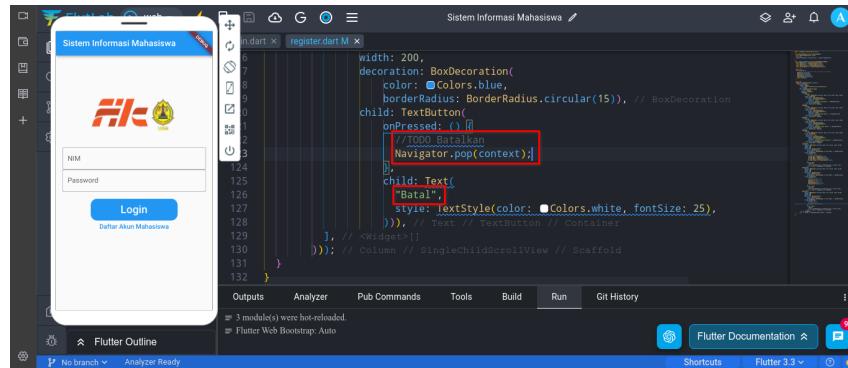
No branch Analyzer Ready

Shortcuts Flutter 3.3

5. Kemudian agar button **Batal** bisa berfungsi, masukkan kode berikut:

Potongan Kode

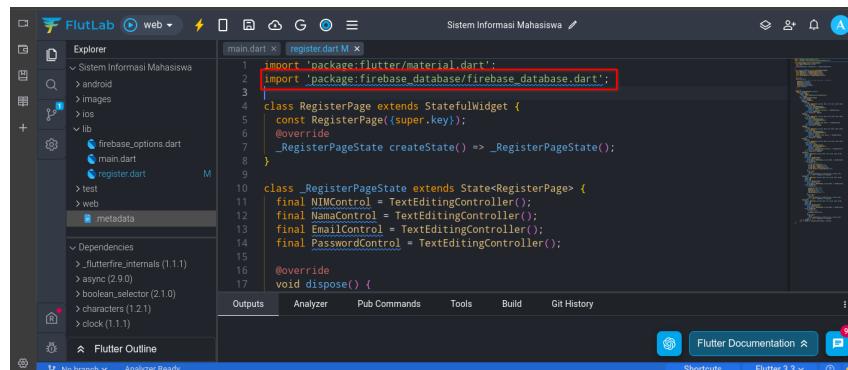
```
//TODO Batalkan  
Navigator.pop(context);
```



6. Untuk bisa melakukan kueri ke Firebase, pertama-tama **import** firebase reference dengan kode berikut:

Potongan Kode

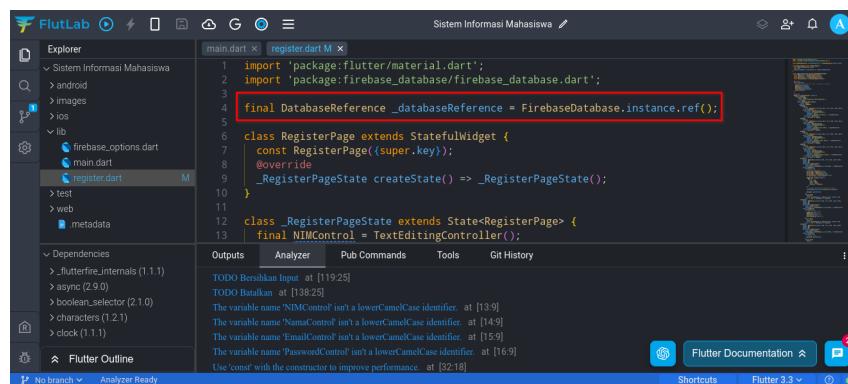
```
import 'package:firebase_database.firebaseio_database.dart';
```



7. Lalu tambahkan variabel global tepat di bawah kode import

Potongan Kode

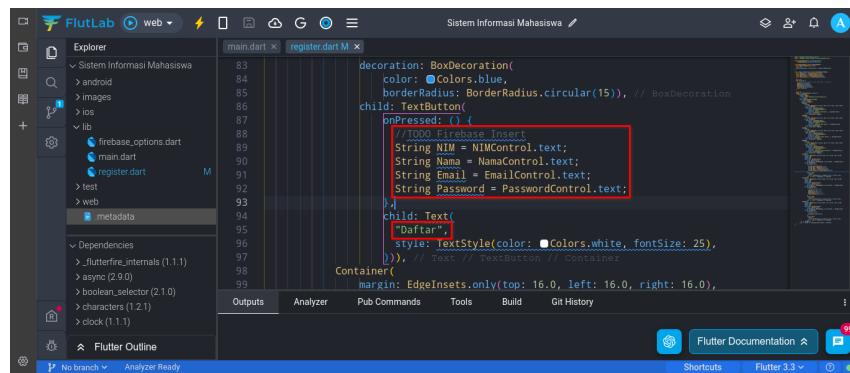
```
final DatabaseReference _databaseReference = FirebaseDatabase  
.instance.ref();
```



8. Kemudian scroll turun ke bagian button **Daftar** dan masukkan kode berikut untuk mengambil data

Potongan Kode

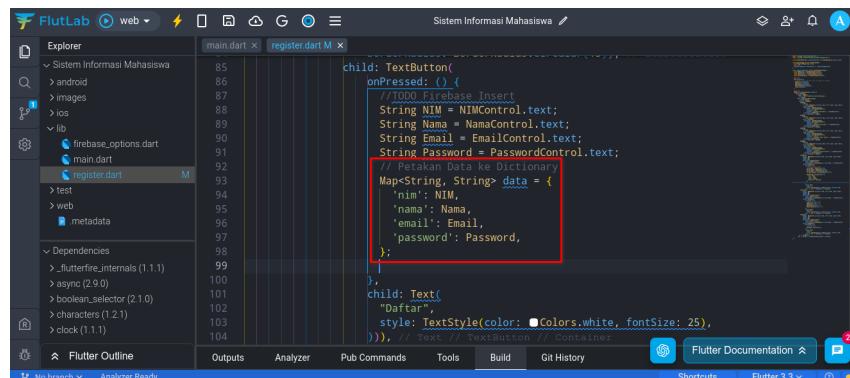
```
//TODO Firebase Insert
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
```



9. Kemudian bungkus data-data tersebut dalam bentuk **dictionary**, perhatikan kode dan gambar di bawah

Potongan Kode

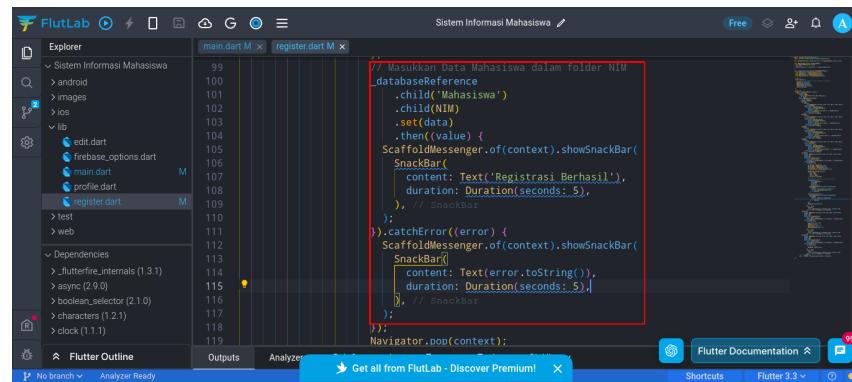
```
// Petakan Data ke Dictionary
Map<String, String> data = {
  'nim': NIM,
  'nama': Nama,
  'email': Email,
  'password': Password,
};
```



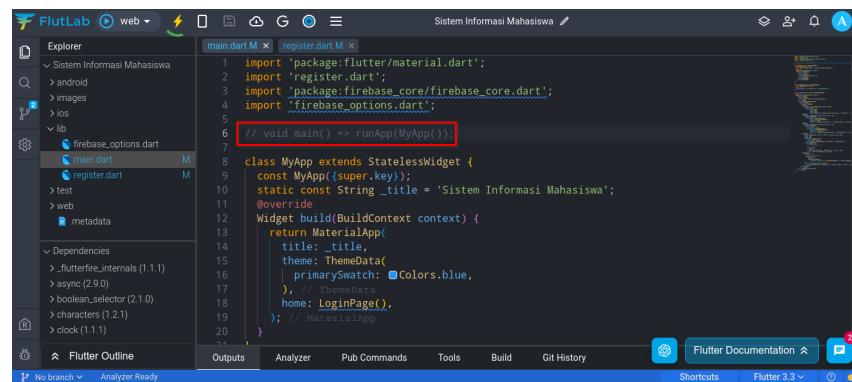
10. Kemudian tambahkan kode untuk melakukan kueri insert dan pengecekan keberhasilan

Potongan Kode

```
// Masukkan Data Mahasiswa dalam folder NIM
_databaseReference
.child('Mahasiswa')
.child(NIM)
.set(data)
.then((value) {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text('Registrasi Berhasil'),
            duration: Duration(seconds: 3),
        ),
    );
}).catchError((error) {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text(error.toString()),
            duration: Duration(seconds: 3),
        ),
    );
});
Navigator.pop(context);
```



11. Langkah berikutnya adalah mengubah fungsi **main** karena harus melakukan inisialisasi database. Buka **main.dart** dan beri tanda // di depan baris kode **void main() => runApp(MyApp());**



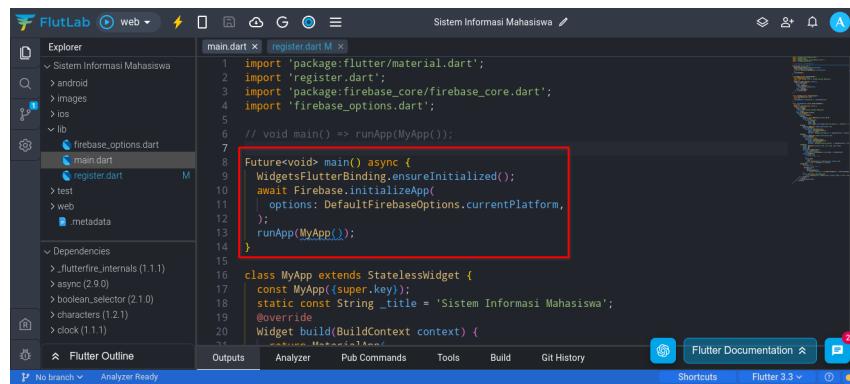
12. Lalu di bawahnya baris kode yang diberi komentar tadi, masukkan kode berikut:

```

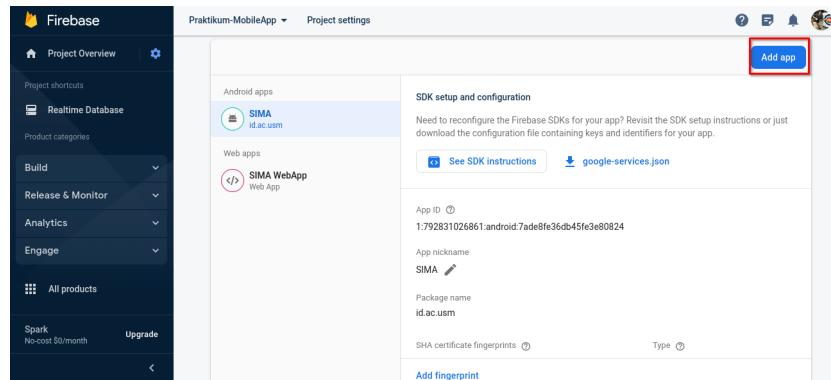
Potongan Kode
import 'package:firebase_core/firebase_core.dart';

Potongan Kode
Future<void> main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
        options: DefaultFirebaseOptions.currentPlatform,
    );
    runApp(MyApp());
}

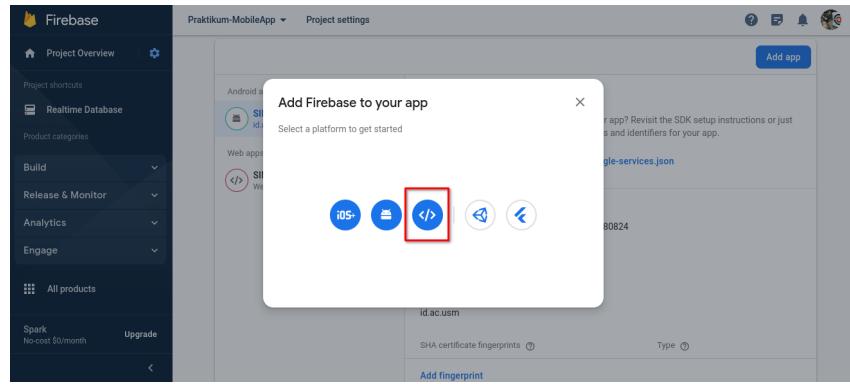
```



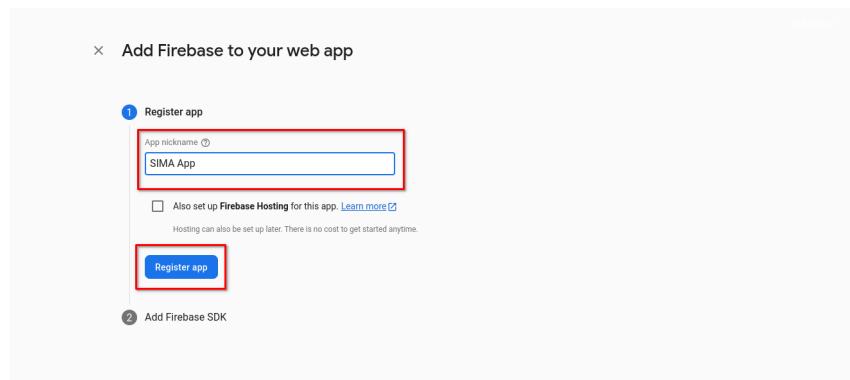
13. Aplikasi ini akan gagal ketika dijalankan, sehingga harus ditambahkan akses baru untuk aplikasi Web. Buka link <https://firebase.google.com>, buka Project Settings (Gigi Roda), pilih General, Scroll Down ke Your Apps, klik Add App



14. Pilih platform Web



15. Beri nama **SIMA App**, lalu klik **Register App**



16. Lalu kopi kode yang diperlihatkan Firebase

```

S npm install firebase
Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyBcBXw49m1lU-7Gc6p9Um88Nncz17j2rc",
  authDomain: "praktikum-mobileapp-878bb.firebaseapp.com",
  databaseURL: "https://praktikum-mobileapp-878bb.firebaseio.default-rtbd.firebaseio.asia-southeast1",
  projectId: "praktikum-mobileapp-878bb",
  storageBucket: "praktikum-mobileapp-878bb.appspot.com",
  messagingSenderId: "79283102861",
  appId: "1:79283102861:web:69141d3d2ae74b5e88824"
};

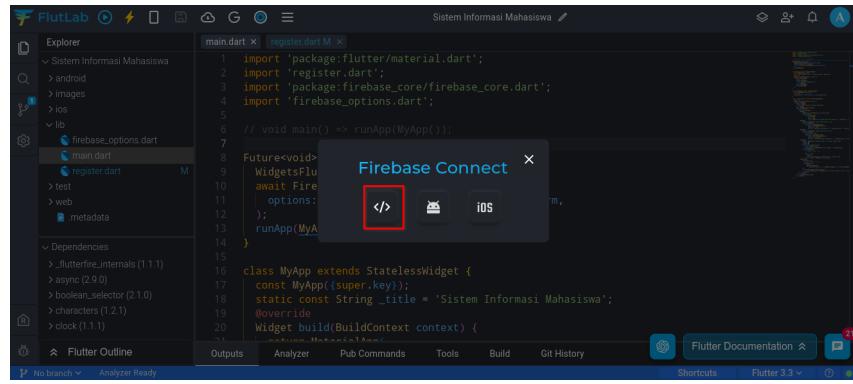
// Initialize Firebase
const app = initializeApp(firebaseConfig);

Note: This option uses the modular JavaScript SDK, which provides reduced SDK size.

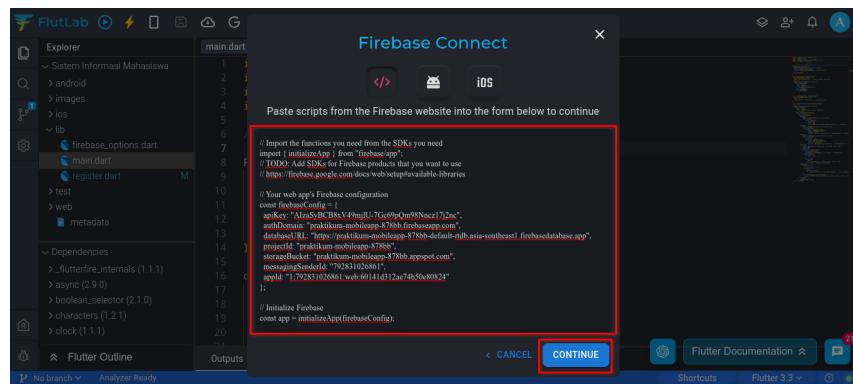
Learn more about Firebase for web: Get Started, Web SDK API Reference, Samples

```

17. Kembali ke Flutlab, klik **Icon G**, pilih **Connect to Firebase**, pilih **Icon Web** yang sama dengan Firebase sebelumnya

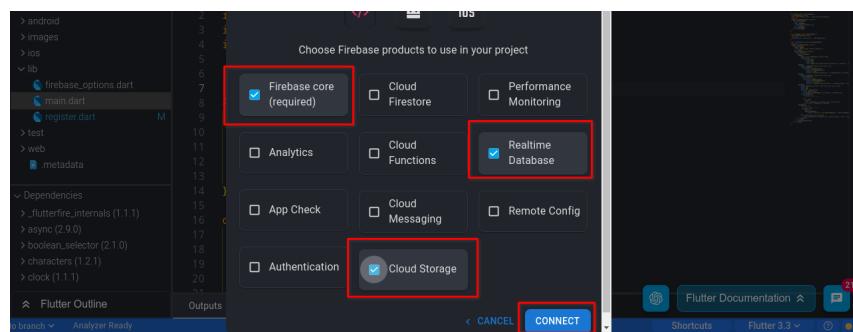


18. Tempel kode dari FIrebse, dan klik **Continue**

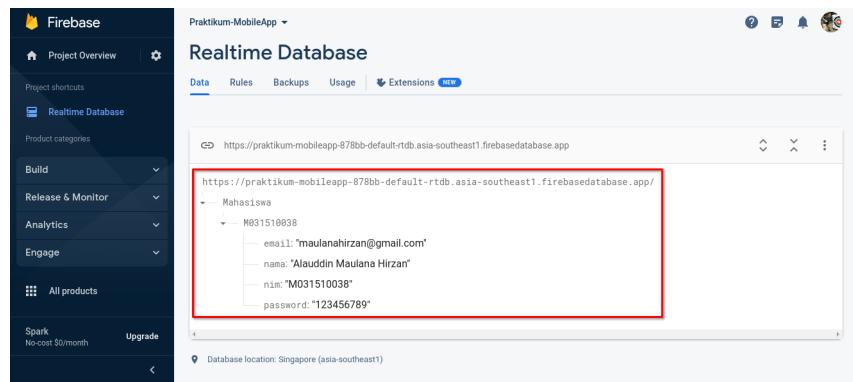
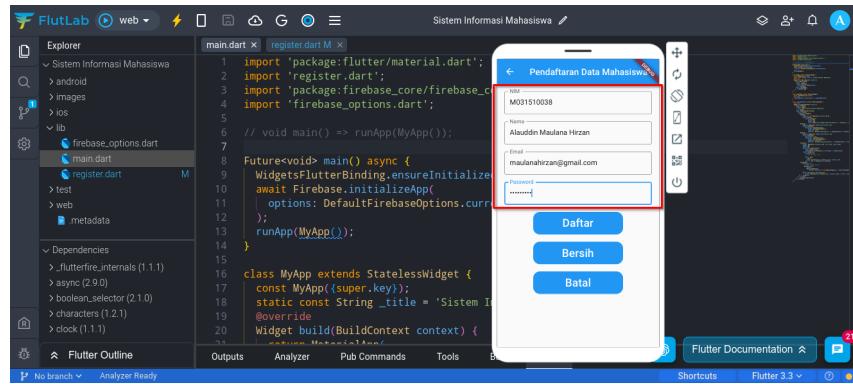


19. Di menu selanjutnya pilih

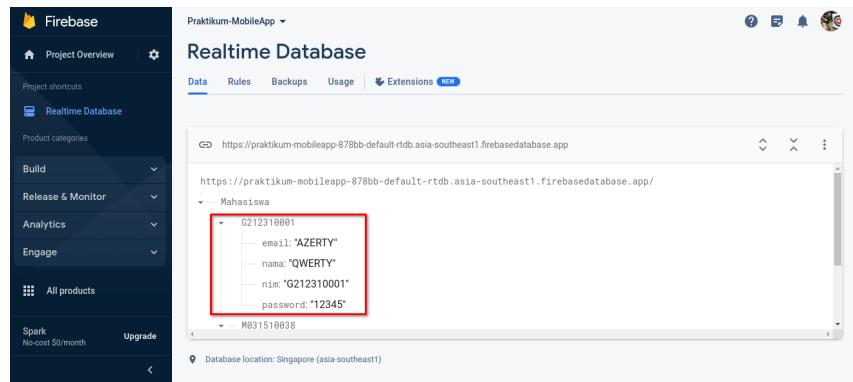
- Firebase Core
- Realtime Database
- Cloud Storage



20. Sesudah terhubung, maka aplikasi siap digunakan. Jalankan Web Emulator dan coba melakukan registrasi. Isikan **NIM TANPA TITIK** untuk mencoba.



21. Data Mahasiswa lain dapat masuk ke DB



22. Screenshot Database dan kirimkan ke E-Learning

Bab 6

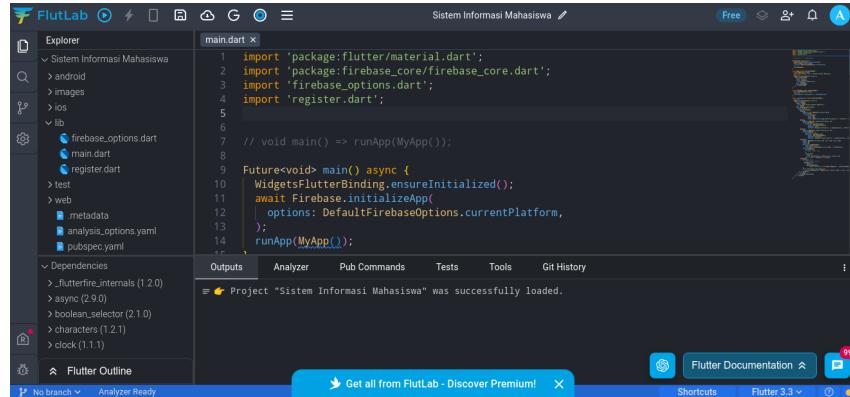
Praktikum 6

6.1 Operasi Read Data Flutter

Di bagian ini mahasiswa diajarkan bagaimana membuat tampilan profile untuk pengguna, sekaligus proses Login yang dilakukan dengan menggunakan Flutter. Mahasiswa diwajibkan menyelesaikan Praktikum 5 sebelum memulai praktikum ini.

6.2 Tutorial

1. Praktikum ini dimulai dengan membuka praktikum sebelumnya.

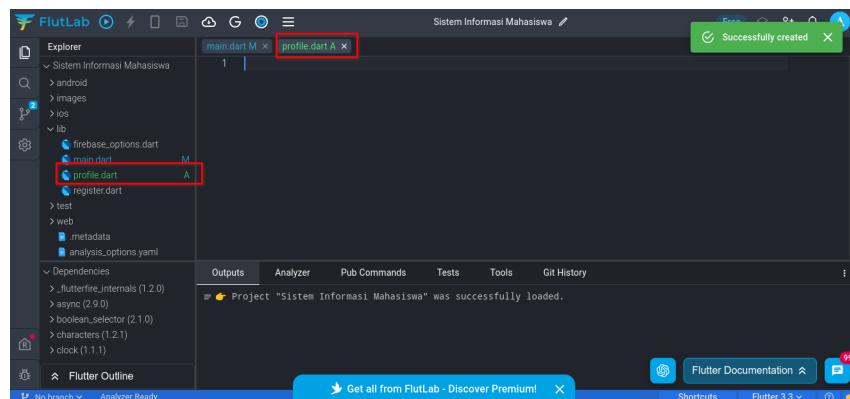


The screenshot shows the FlutLab IDE interface. The left sidebar displays the project structure for 'Sistem Informasi Mahasiswa' with folders for android, images, ios, lib (containing firebase_options.dart, main.dart, register.dart), test, web, and dependencies. The right panel shows the code editor with the main.dart file open. The code imports flutter/material, firebase_core, firebase_options, and register.dart, then initializes Firebase with DefaultFirebaseOptions.currentPlatform before running the app. Below the code editor is a tab bar with Outputs, Analyzer, Pub Commands, Tests, Tools, and Git History. A status bar at the bottom indicates the project was successfully loaded.

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'register.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(MyApp());
}
```

2. Buatlah satu file baru dengan nama **profile.dart**

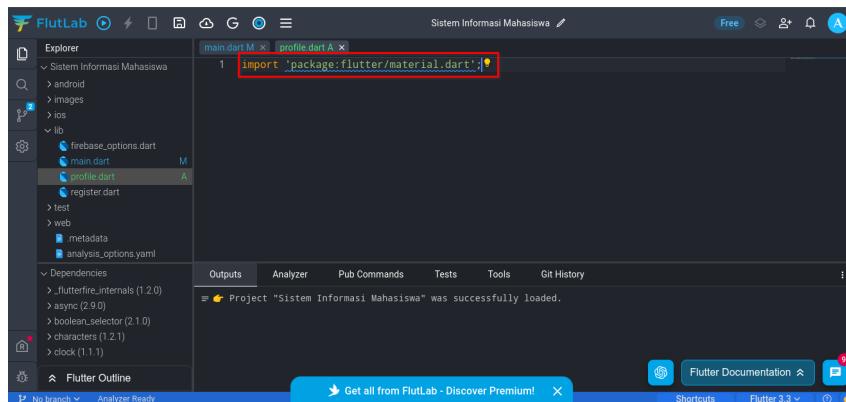


The screenshot shows the FlutLab IDE interface after creating a new file. The file list in the sidebar now includes 'main.dart' and 'profile.dart'. A success message 'Successfully created' is displayed in the top right corner. The code editor shows the 'profile.dart' file is currently selected. The status bar at the bottom indicates the project was successfully loaded.

3. Masukkan kode import sebagai **Library Dasar** aplikasi

Potongan Kode

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

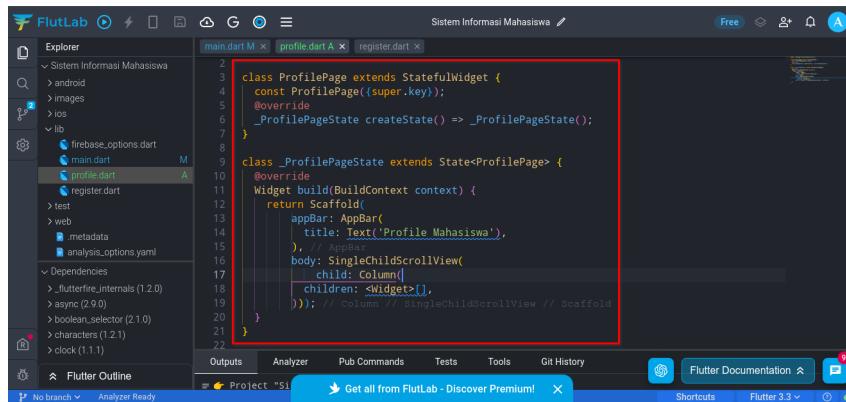


4. Kemudian buatlah tampilan Profile dengan kode sebagai berikut. Tambahkan setelah kode **import**

Potongan Kode

```
class ProfilePage extends StatefulWidget {
    const ProfilePage({super.key});
    @override
    _ProfilePageState createState() => _ProfilePageState();
}

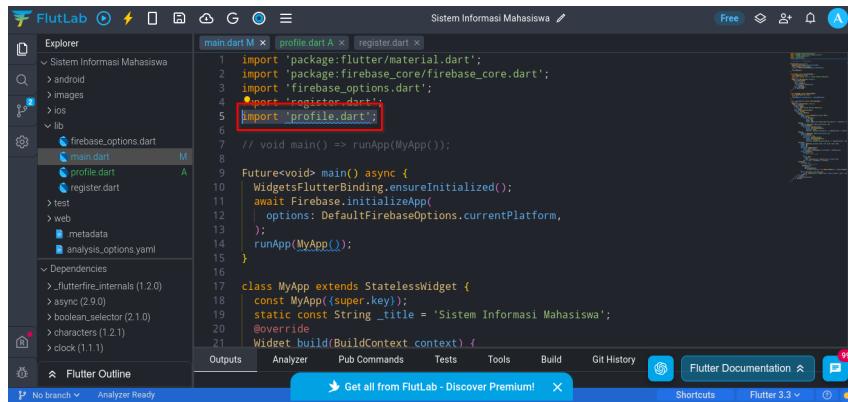
class _ProfilePageState extends State<ProfilePage> {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Profile Mahasiswa'),
            ),
            body: SingleChildScrollView(
                child: Column(
                    children: <Widget>[], // Placeholder
                )));
    }
}
```



5. Sesudah itu, buka file **main.dart** lalu Import file **profile.dart** seperti berikut:

Potongan Kode

```
import 'profile.dart';
```



The screenshot shows the FlutLab IDE interface with the main.dart file open. The code editor displays the following snippet:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'register.dart';
import 'profile.dart';

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

Future<void> main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(
options: DefaultFirebaseOptions.currentPlatform,
);
runApp(MyApp());
}

class MyApp extends StatelessWidget {
const MyApp({super.key});
static const String _title = 'Sistem Informasi Mahasiswa';
@override
Widget build(BuildContext context) {
// ...
}
```

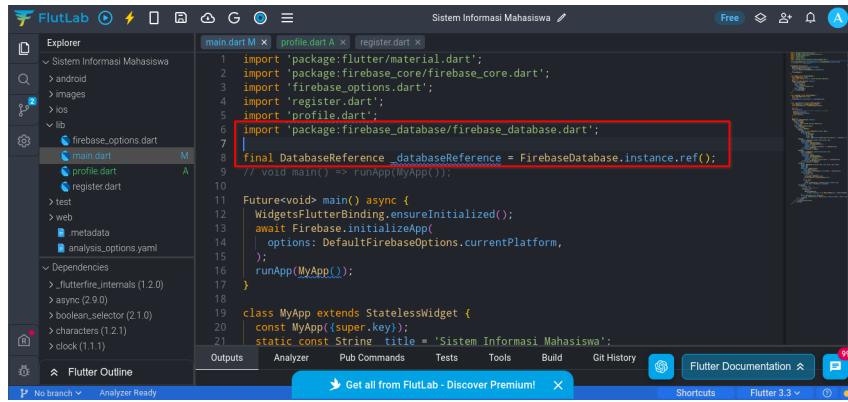
The line `import 'profile.dart';` is highlighted with a red box.

6. Agar bisa terhubung ke database, tambahkan kode berikut tepat di bawah **import** sebelumnya

Potongan Kode

```
import 'package:firebase_database.firebaseio_database.dart';
```

```
final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();
```



The screenshot shows the FlutLab IDE interface with the main.dart file open. The code editor displays the following snippet:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'register.dart';
import 'profile.dart';
import 'package:firebase_database.firebaseio_database.dart';
final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();

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

Future<void> main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(
options: DefaultFirebaseOptions.currentPlatform,
);
runApp(MyApp());
}

class MyApp extends StatelessWidget {
const MyApp({super.key});
static const String _title = 'Sistem Informasi Mahasiswa';
@override
Widget build(BuildContext context) {
// ...
}
```

The line `final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();` is highlighted with a red box.

7. Scroll turun ke **class _LoginPageState**, dan tambahkan kode berikut tepat di bawahnya dan sebelum **@override**

Potongan Kode

```
final NIMControl = TextEditingController();
final PasswordControl = TextEditingController();

@Override
void dispose() {
NIMControl.dispose();
PasswordControl.dispose();
super.dispose();
}
```

```

main.dart M profile.dart A register.dart X
35     _LoginPageState createState() => _LoginPageState();
36 }
37
38 class _LoginPageState extends State<LoginPage> {
39   final NIMControl = TextEditingController();
40   final PasswordControl = TextEditingController();
41
42   @override
43   void dispose() {
44     // Clean up the controller when the widget is disposed.
45     NIMControl.dispose();
46     PasswordControl.dispose();
47     super.dispose();
48   }
49
50   @override
51   Widget build(BuildContext context) {
52     return Scaffold(
53       appBar: AppBar(
54         title: Text('Sistem Informasi Mahasiswa'),
55       ),
56     );
57   }
58 }

```

8. Kemudian turun kembali untuk menempelkan kontrol teks ke masing-masing widget. Perhatikan Kode dan Gambar berikut

(a) field **NIM**

Potongan Kode
controller: NIMControl,

```

main.dart M profile.dart A register.dart X
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

```

(b) field **Password**

Potongan Kode
controller: PasswordControl,

```

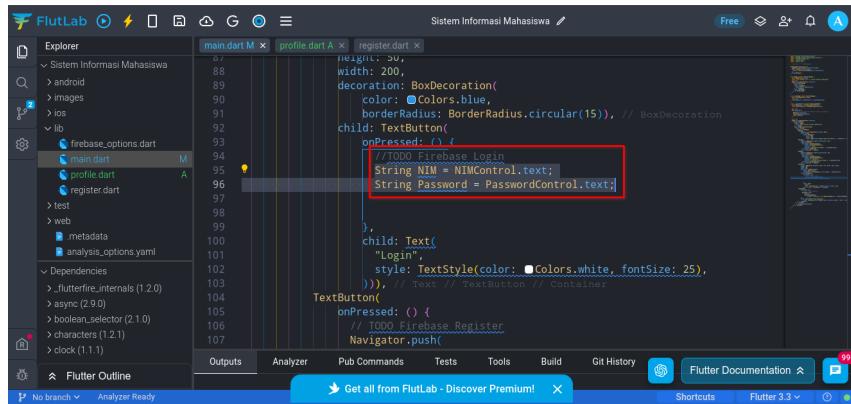
main.dart M profile.dart A register.dart X
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

```

9. Kini setiap field dapat ditarik datanya. Berikutnya tambahkan kode berikut di tombol **Login**. Perhatikan gambar dan kode berikut:

Potongan Kode

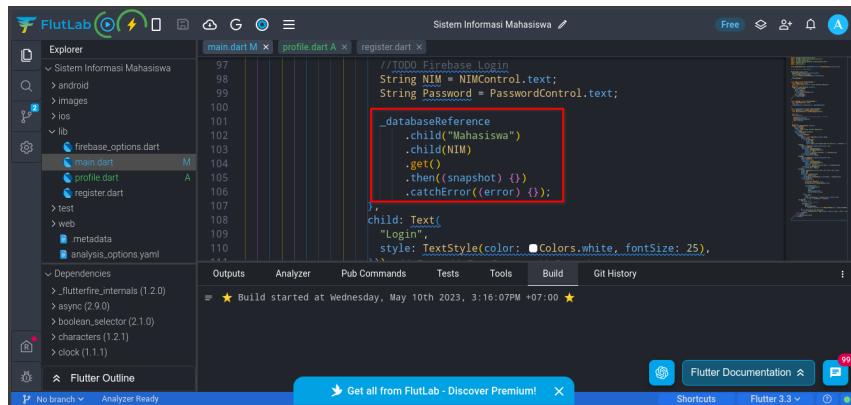
```
String NIM = NIMControl.text;  
String Password = PasswordControl.text;
```



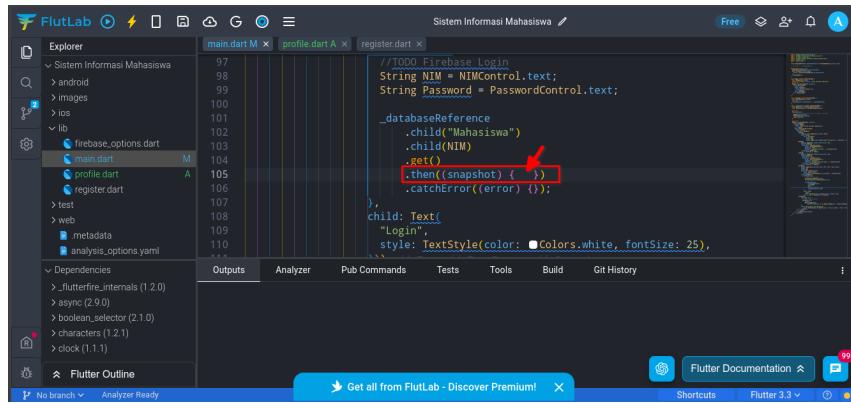
10. Tambahkan kode berikut untuk melakukan pengecekan login

Potongan Kode

```
databaseReference  
    .child("Mahasiswa")  
    .child(NIM)  
    .get()  
    .then((snapshot) { })  
    .catchError((error) { });
```

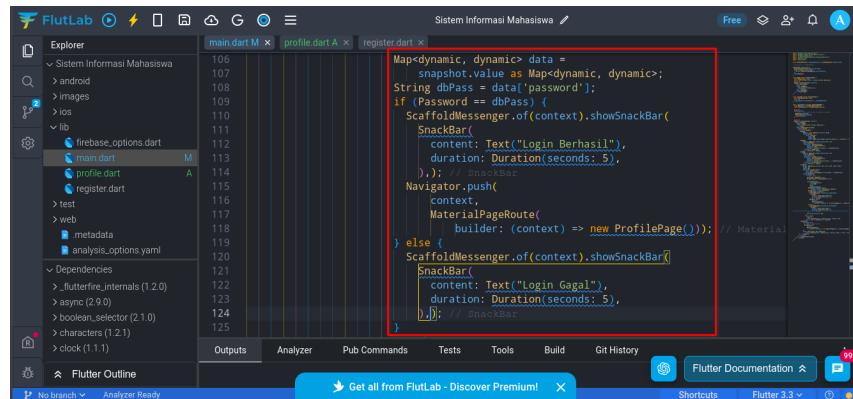


11. Setelah itu di dalam tanda kurung kurawal dari **Snapshot**. Masukkan kode berikut

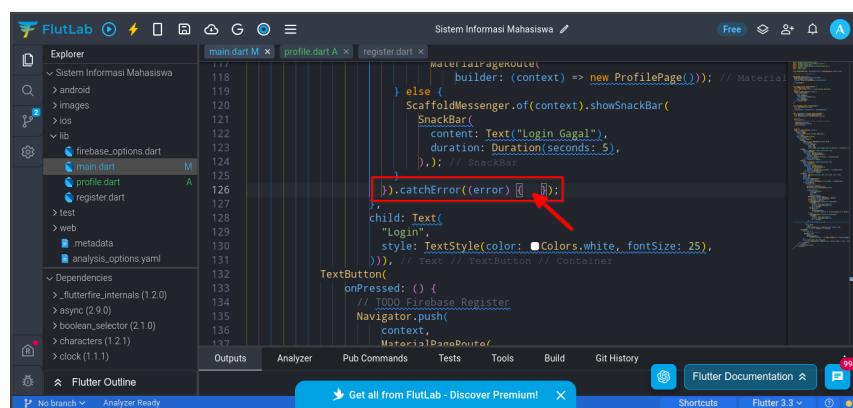


Potongan Kode

```
Map<dynamic, dynamic> data =
    snapshot.value as Map<dynamic, dynamic>;
String dbPass = data['password'];
if (Password == dbPass) {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text("Login Berhasil"),
            duration: Duration(seconds: 5),
        ),);
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => new ProfilePage()));
} else {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text("Login Gagal"),
            duration: Duration(seconds: 5),
        ),);
}
```

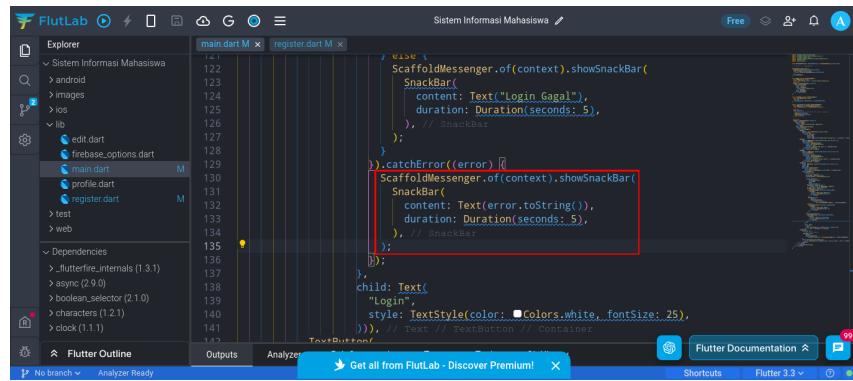


12. Kemudian di dalam kurung kurawal dari catchError, masukkan kode berikut

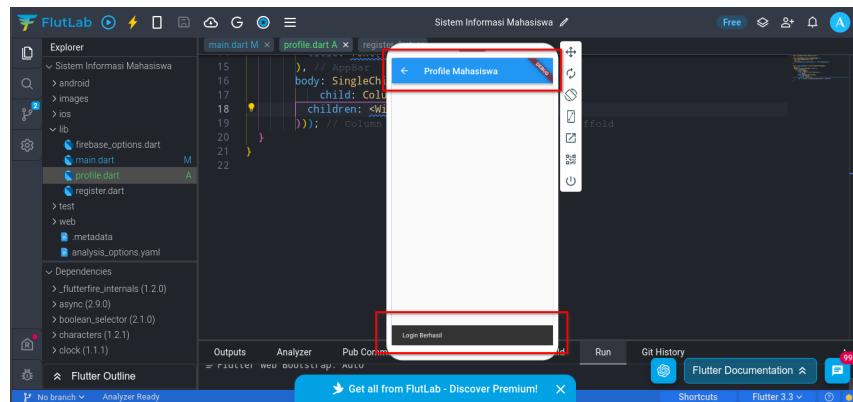


Potongan Kode

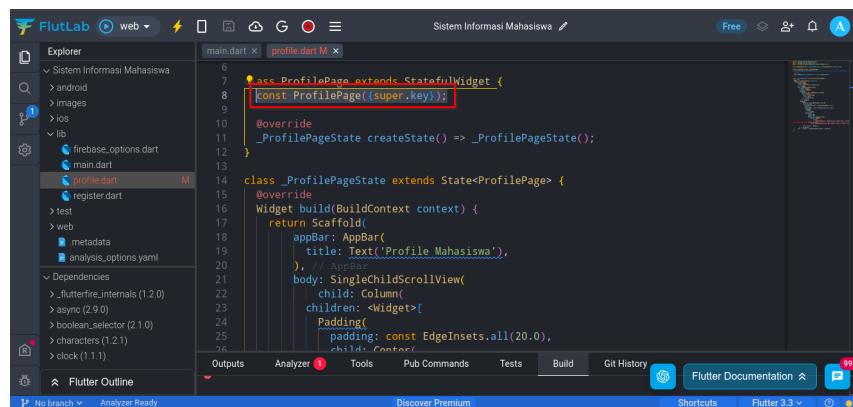
```
ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
        content: Text(error.toString()),
        duration: Duration(seconds: 5),
    ),
);
```



- Jalankan aplikasi dan cobalah login. Jika sukses, halaman akan berpindah ke **profile.dart**. Tetapi kosong.



- Berikutnya adalah mengisikan halaman Profile. Pertama hapus baris kode **const ProfilePage(super.key);**



15. Agar bisa menerima input data **NIM** dari **Login** ke **Profile**, masukkan kode berikut di class **ProfilePage**

Potongan Kode

```
ProfilePage({required this.NIM});
final String NIM;
```

The screenshot shows the `ProfilePage` class definition in the `profile.dart` file. The `final String NIM;` line is highlighted with a red box.

```

6  class ProfilePage extends StatefulWidget {
7    ProfilePage({required this.NIM});
8    final String NIM;
9
10   @override
11   _ProfilePageState createState() => _ProfilePageState();
12 }
13
14
15   class _ProfilePageState extends State<ProfilePage> {
16     @override
17     Widget build(BuildContext context) {
18       return Scaffold(
19         appBar: AppBar(
20           title: Text('Profile Mahasiswa'),
21         ), // AppBar
22         body: SingleChildScrollView(
23           child: Column(
24             children: <Widget>[
25               Padding(
26                 padding: EdgeInsets.all(20.0),
27               ),
28             ],
29           ),
30         ),
31       );
32     }
33   }
34
35 }
```

16. Kembali ke **main.dart**, tambahkan baris kode **builder: (context) => new ProfilePage());** dengan kode berikut:

Potongan Kode

NIM:NIM

The screenshot shows the `main.dart` file in the FlutLab IDE. The `builder: (context) => new ProfilePage(NIM: NIM));` line is highlighted with a red box.

```

105   String userpass = widget.password;
106   if (Password == dbPass) {
107     ScaffoldMessenger.of(context).showSnackBar(
108       SnackBar(
109         content: Text("Login Berhasil"),
110         duration: Duration(seconds: 5),
111       ), // SnackBar
112     );
113     Navigator.push(
114       context,
115       MaterialPageRoute(
116         builder: (context) => new ProfilePage(NIM: NIM));
117       ),
118     );
119   } else {
120     ScaffoldMessenger.of(context).showSnackBar(
121       SnackBar(
122         content: Text("Login Gagal"),
123         duration: Duration(seconds: 5),
124       ), // SnackBar
125     );
126   }
127 }
128 }
```

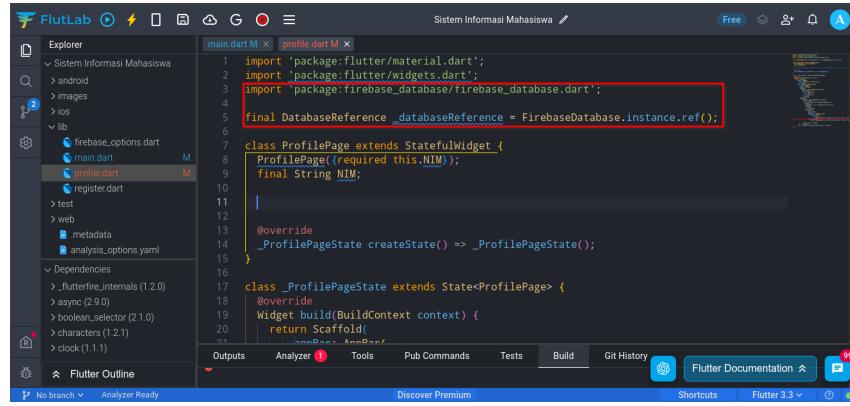
17. Kini data NIM bisa dikirimkan ke **profile.dart**. Buka file **profile.dart**, agar bisa melakukan kueri data, tambahkan kode berikut:

18. Sebelum menambahkan kode untuk Text, tambahkan terlebih dahulu kode berikut untuk menarik data dari Firebase

Potongan Kode

```
import 'package:firebase_database.firebaseio_database.dart';

final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();
```



19. Untuk melakukan kueri, Scroll Turun ke **class _ProfilePageState extends State<ProfilePage>**. Dan masukkan kode berikut tepat di bawahnya. Perhatikan Gambar dan Kode

Potongan Kode

```
Map<dynamic, dynamic> mappedData = {};
String NIM = "";

@Override
void initState() {
    super.initState();
    NIM = widget.NIM;
    fetchData(NIM);
}

Future<void> fetchData(String NIM) async {
    final ref = _databaseReference.child("Mahasiswa").child(NIM);
    final data = await ref.get();
    setState(() {
        mappedData = data.value as Map<dynamic,dynamic>;
    });
}
```

```

14
15     class _ProfilePageState extends State<ProfilePage> {
16         Map<dynamic, dynamic> mappedData = {};
17         String NIM = "";
18
19         @override
20         void initState() {
21             super.initState();
22             NIM = widget.NIM;
23             fetchData(NIM);
24         }
25
26         Future<void> fetchData(String NIM) async {
27             final ref = _databaseReference.child("Mahasiswa").child(NIM);
28             final data = await ref.get();
29             setState(() {
30                 mappedData = data.value as Map<dynamic, dynamic>;
31             });
32         }
33     }

```

20. Berikutnya adalah mengisi tampilan **profile.dart**. Isikan kode di dalam **Widget[]**

```

10     @override
11     Widget build(BuildContext context) {
12         return Scaffold(
13             appBar: AppBar(
14                 title: Text('Profile Mahasiswa'),
15             ), // AppBar
16             body: SingleChildScrollView(
17                 child: Column(
18                     children: <Widget>... // Red box here
19                 ), // Column
20             ), // SingleChildScrollView
21         ); // Scaffold
22     }

```

21. Di mulai dari kode **Avatar Profile**

Potongan Kode

```

Padding(
    padding: const EdgeInsets.all(20.0),
    child: Center(
        child: FlutterLogo(
            size: 80.0,
        ),
    ),
),

```

```

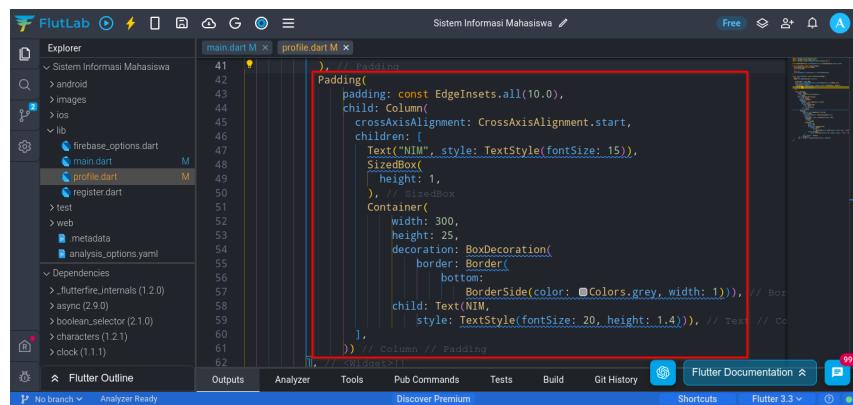
17     body: SingleChildScrollView(
18         child: Column(
19             children: <Widget>[
20                 Padding(
21                     padding: const EdgeInsets.all(20.0),
22                     child: Center(
23                         child: FlutterLogo(
24                             size: 80.0,
25                         ), // FlutterLogo
26                     ), // Center
27                 ), // Padding
28             ], // Column
29         ), // SingleChildScrollView
30     ); // Scaffold

```

22. Kemudian lanjut kode di atas dengan tampilan untuk NIM.

Potongan Kode

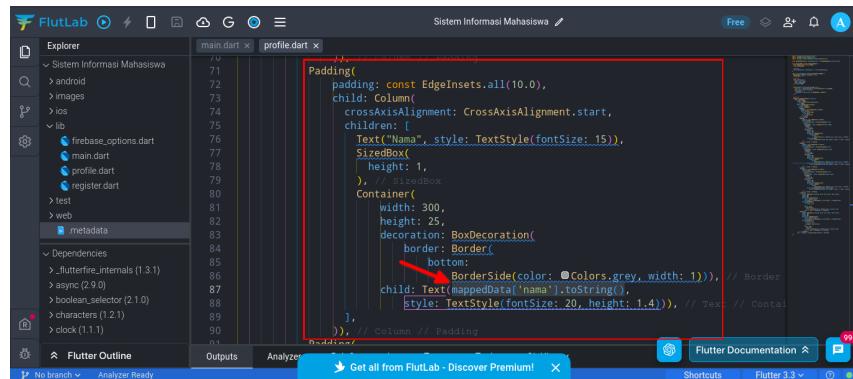
```
Padding(
    padding: const EdgeInsets.all(10.0),
    child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
            Text("NIM", style: TextStyle(fontSize: 15)),
            SizedBox(
                height: 1,
            ),
            Container(
                width: 300,
                height: 25,
                decoration: BoxDecoration(
                    border: Border(
                        bottom:
                            BorderSide(color: Colors.grey, width: 1))),
            child: Text(NIM,
                style: TextStyle(fontSize: 20, height: 1.4))),
        ],
    )),
```



23. Tempel kode kembali untuk menampilkan Nama

Potongan Kode

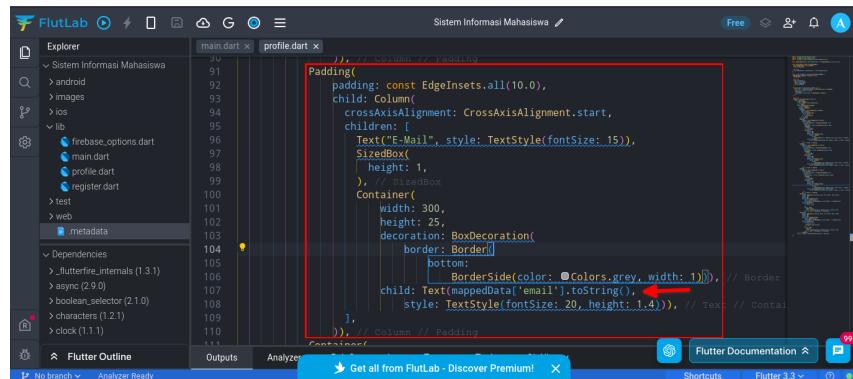
```
Padding(  
    padding: const EdgeInsets.all(10.0),  
    child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text("Nama", style: TextStyle(fontSize: 15)),  
            SizedBox(  
                height: 1,  
            ),  
            Container(  
                width: 300,  
                height: 25,  
                decoration: BoxDecoration(  
                    border: Border(  
                        bottom:  
                            BorderSide(color: Colors.grey, width: 1))),  
                child: Text(mappedData['nama'].toString(),  
                    style: TextStyle(fontSize: 20, height: 1.4)),  
            ),  
        ],  
    )),
```



24. Tempel kembali untuk menampilkan E-Mail

Potongan Kode

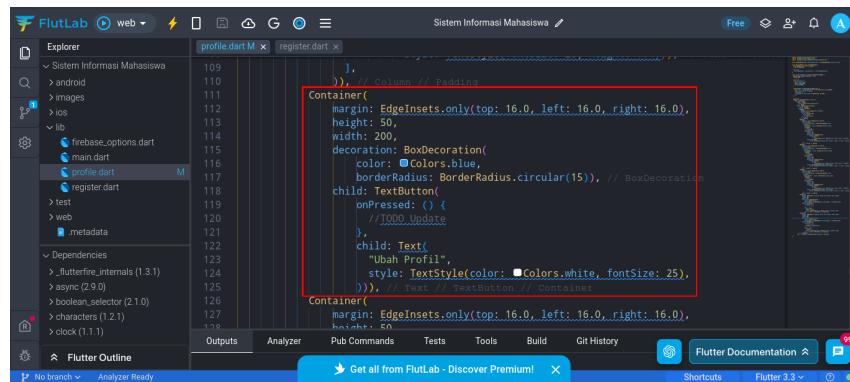
```
Padding(  
    padding: const EdgeInsets.all(10.0),  
    child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text("E-Mail", style: TextStyle(fontSize: 15)),  
            SizedBox(  
                height: 1,  
            ),  
            Container(  
                width: 300,  
                height: 25,  
                decoration: BoxDecoration(  
                    border: Border(  
                        bottom:  
                            BorderSide(color: Colors.grey, width: 1))),  
                child: Text(mappedData['email'].toString(),  
                    style: TextStyle(fontSize: 20, height: 1.4)),  
            ),  
        ],  
    )),
```



25. Kode terakhir yang ditambahkan adalah **Update Profile** dan **Logout**. Letakkan kode setelah kode sebelumnya.

Potongan Kode

```
Container(  
    margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.blue,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Update  
        },  
        child: Text(  
            "Ubah Profil",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
    )),,
```



Potongan Kode

```
Container(  
    margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.blue,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Batalkan  
            Navigator.pop(context);  
        },  
        child: Text(  
            "Logout",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
    )),,
```

The screenshot shows the FlutLab IDE interface. On the left is the Explorer sidebar with project files like `main.dart` and `profile.dart`. The main area has two tabs: `profile.dart M` and `register.dart M`. The `profile.dart` tab contains Dart code for a profile page. A red box highlights a specific section of the code where a `Container` widget is defined with properties: `margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0)`, `height: 50`, `width: 200`, and a `decoration` property. The `Flutter Documentation` button is visible at the bottom.

```
Container(
  margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
  height: 50,
  width: 200,
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(15)), // BoxDecoration
), // Container
child: TextButton(
  onPressed: () {
    //TODO Batalkan
    Navigator.pop(context);
  },
  child: Text(
    "Logout",
    style: TextStyle(color: Colors.white, fontSize: 25),
  )));
}, // Column // SingleChildScrollView // Scaffold
```

26. Jalankan aplikasi dan tes

This screenshot shows the FlutLab IDE with the `main.dart` file open. The code defines a stateful widget `ProfilePageState` that fetches data from a database. The preview window shows a mobile application with a profile screen. The screen displays user information: NIM (M031510038), Name (Alauddin Maulana Hirzan), and Email (maulanahirzan@gmail.com). It includes a blue 'Ubah Profil' (Change Profile) button and a blue 'Logout' button. The `Flutter Documentation` button is also present at the bottom.

```
class _ProfilePageState extends State<ProfilePage> {
  Map<dynamic, dynamic> mappedData = {};
  String NIM = "";
  @override
  void initState() {
    super.initState();
    NIM = widget.NIM;
    fetchData(NIM);
  }
  Future<void> fetchData(String NIM) async {
    final ref = _databaseReference.child("Mahasiswa").child(NIM);
    final data = await ref.get();
    setState(() {
      mappedData = data.value as Map<dynamic, dynamic>;
    });
  }
}
```

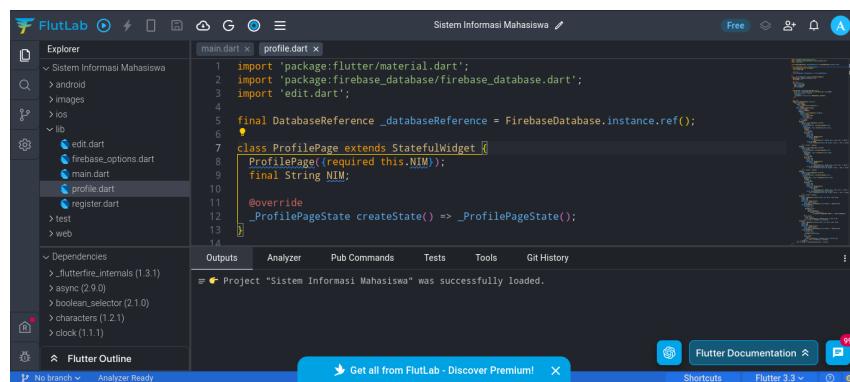
Bab 7

Praktikum 7

7.1 Operasi Update Data

7.2 Tutorial

1. Buka kembali projek di Flutlab

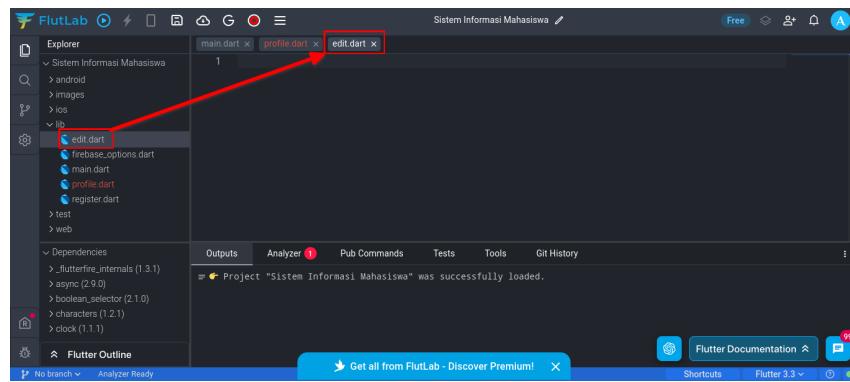


The screenshot shows the FlutLab interface with the code editor open. The file 'profile.dart' is currently selected in the sidebar. The code editor displays the following Dart code:

```
1 import 'package:flutter/material.dart';
2 import 'package:firebase_database.firebaseio_database.dart';
3 import 'edit.dart';
4
5 final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();
6
7 class ProfilePage extends StatefulWidget {
8   ProfilePage({required this.NIM});
9   final String NIM;
10
11   @override
12   _ProfilePageState createState() => _ProfilePageState();
13 }
14
```

The status bar at the bottom indicates that the project "Sistem Informasi Mahasiswa" was successfully loaded.

2. Buatlah satu file dengan nama **edit.dart**



The screenshot shows the FlutLab interface with the code editor open. The file 'edit.dart' is currently selected in the sidebar. A red arrow points from the sidebar to the file name in the code editor tab bar. The code editor tab bar shows three files: 'main.dart', 'profile.dart', and 'edit.dart'. The code editor displays the following Dart code:

```
1
```

The status bar at the bottom indicates that the project "Sistem Informasi Mahasiswa" was successfully loaded.

3. Kemudian masukkan kode berikut ke dalam file **edit.dart**

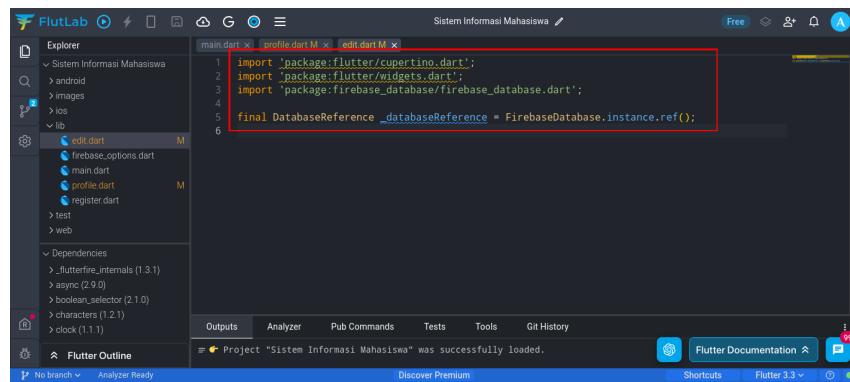
Potongan Kode

```

import 'package:flutter/cupertino.dart';
import 'package:flutter/widgets.dart';
import 'package:firebase_database.firebaseio_database.dart';

final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();

```



- Kemudian tambahkan kode berikutnya untuk widget

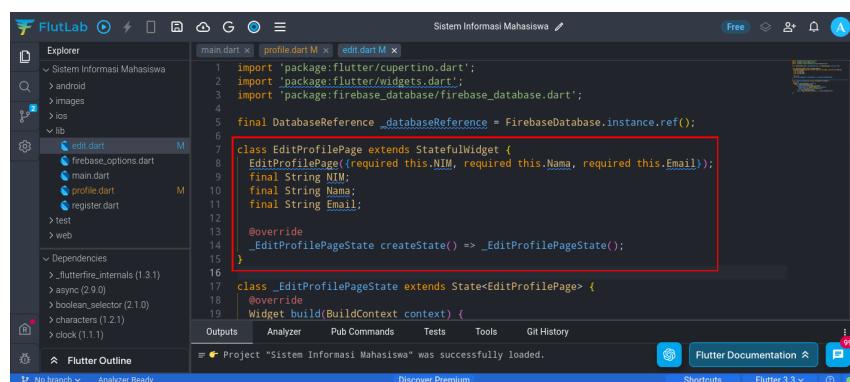
Potongan Kode

```

class EditProfilePage extends StatefulWidget {
    EditProfilePage({required this.NIM, required this>Nama, required this.Email});
    final String NIM;
    final String Nama;
    final String Email;

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

```



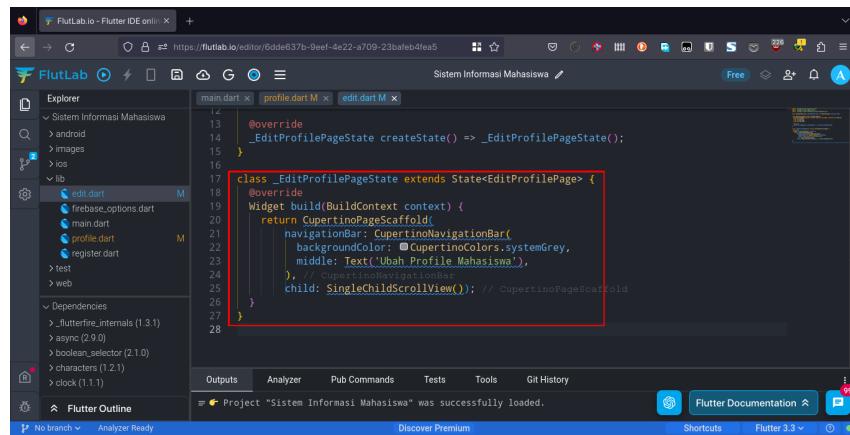
- Kemudian lanjutkan dengan kode **widget** seperti berikut:

Potongan Kode

```

class _EditProfilePageState extends State<EditProfilePage> {
    @override
    Widget build(BuildContext context) {
        return CupertinoPageScaffold(
            navigationBar: CupertinoNavigationBar(
                backgroundColor: CupertinoColors.systemGrey,
                middle: Text('Ubah Profile Mahasiswa'),
            ),
            child: SingleChildScrollView());
    }
}

```



6. Buka file **profile.dart**, dan tambahkan kode berikut untuk tombol **Ubah Profil**:

- Import baris awal

Potongan Kode

```
import 'edit.dart';
```

- Tombol Ubah Profil:

Potongan Kode

```
//TODO Update
Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => new EditProfilePage(
            NIM: NIM,
            Nama: mappedData['nama'],
            Email: mappedData['email'])));
```

```

    child: TextButton(
      onPressed: () {
        //TODO_Update
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => new EditProfilePage(
              NIM: NIM,
              Nama: mappedData['nama'],
              Email: mappedData['email']));
        );
      },
      child: Text(
        "Ubah Profil",
        style: TextStyle(color: Colors.white, fontSize: 25),
      ),
    ),
  ),

```

7. Tes aplikasi untuk memastikan tombol berfungsi membuka halaman **Update Profile**

```

    child: TextButton(
      onPressed: () {
        //TODO_Update
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => new EditProfilePage(
              NIM: NIM,
              Nama: mappedData['nama'],
              Email: mappedData['email']));
        );
      },
      child: Text(
        "Ubah Profil",
        style: TextStyle(color: Colors.white, fontSize: 25),
      ),
    ),
  ),

```

8. Buka kembali file **edit.dart**, dan tambahkan kode berikut tepat di bawah **class _EditProfilePageState extends State<EditProfilePage>** untuk memasukkan data ke State

Potongan Kode

```

String NIM = "";
String Nama = "";
String Email = "";

@Override
void initState() {
  super.initState();
  NIM = widget.NIM;
  Nama = widget>Nama;
  Email = widget>Email;
}

```

```

class _EditProfilePageState extends State<EditProfilePage> {
  String NIM = "";
  String Nama = "";
  String Email = "";

  @override
  void initState() {
    super.initState();
    NIM = widget.NIM;
    Nama = widget>Nama;
    Email = widget>Email;
  }

  @override
  Widget build(BuildContext context) {
    return ...
  }
}

```

9. Berikutnya adalah menambahkan kode layout. Masukkan tepat di antara **child:** **SingleChildScrollView()**

```

return CupertinoPageScaffold(
  navigationBar: CupertinoNavigationBar(
    backgroundColor: CupertinoColors.systemGrey,
    middle: Text('Ubah Profile Mahasiswa'),
  ),
  child: SingleChildScrollView(...));
}

```

Potongan Kode

```

child: Column(
  mainAxisAlignment: MainAxisAlignment.start,
  mainAxisSize: MainAxisSize.min,
  children: <Widget>[],
),

```

```

Widget build(BuildContext context) {
  return CupertinoPageScaffold(
    navigationBar: CupertinoNavigationBar(
      backgroundColor: CupertinoColors.systemGrey,
      middle: Text('Ubah Profile Mahasiswa'),
    ),
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[],
      ),
    ),
  );
}

```

10. Jika sudah, tepat di dalam sintaks **<Widget>[]** masukkan kode berikut:

```

main.dart x profile.dart M edit.dart M
System Informasi Mahasiswa
Widget build(BuildContext context) {
  return CupertinoPageScaffold(
    navigationBar: CupertinoNavigationBar(
      backgroundColor: CupertinoColors.systemGrey,
      middle: Text('Ubah Profil Mahasiswa'),
    ),
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[1],
      ),
    ),
  );
}

```

Potongan Kode

Padding(

```

padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
  placeholder: NIM,
),
)

```

```

main.dart x profile.dart M edit.dart M
System Informasi Mahasiswa
Widget build(BuildContext context) {
  return CupertinoPageScaffold(
    navigationBar: CupertinoNavigationBar(
      backgroundColor: CupertinoColors.systemGrey,
      middle: Text('Ubah Profil Mahasiswa'),
    ),
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
            child: CupertinoTextField(
              placeholder: NIM,
            ),
          ),
        ],
      ),
    ),
  );
}

```

11. Tepat di bawahnya, tambahkan kode untuk **Nama** tepat setelah **koma**:

Potongan Kode

Padding(

```

padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
  placeholder: Nama,
),
)

```

```

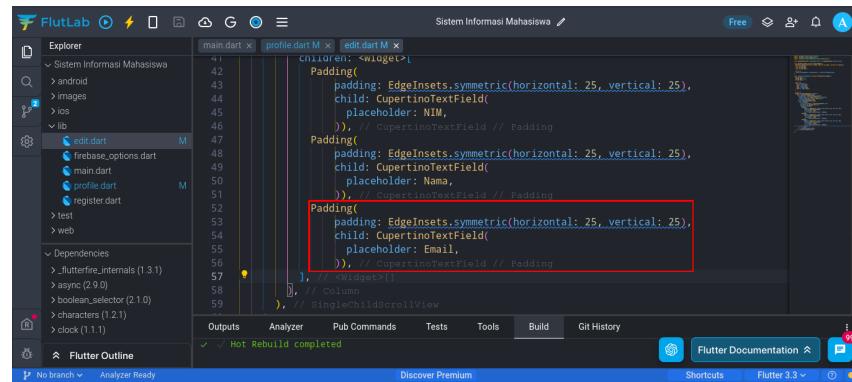
main.dart x profile.dart M edit.dart M
System Informasi Mahasiswa
Widget build(BuildContext context) {
  return CupertinoPageScaffold(
    navigationBar: CupertinoNavigationBar(
      backgroundColor: CupertinoColors.systemGrey,
      middle: Text('Ubah Profil Mahasiswa'),
    ),
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
            child: CupertinoTextField(
              placeholder: NIM,
            ),
          ),
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
            child: CupertinoTextField(
              placeholder: Nama,
            ),
          ),
        ],
      ),
    ),
  );
}

```

12. Tambahkan **TextField** untuk **Email**. Masukkan setelah tanda **koma**

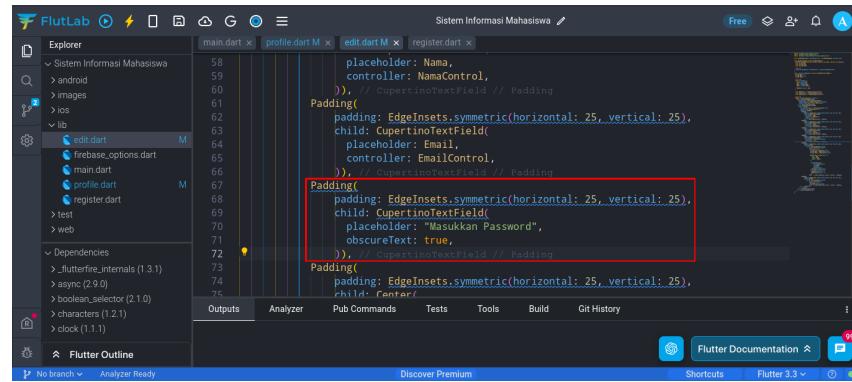
Potongan Kode

```
Padding(  
    padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),  
    child: CupertinoTextField(  
        placeholder: Email,  
    )),
```

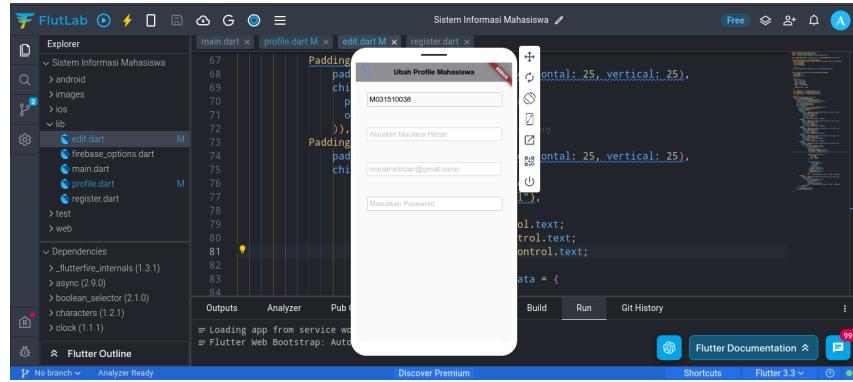


13. Tambahkan **TextField** terakhir untuk **Password**. Masukkan setelah tanda **koma** [Potongan Kode](#)

```
Padding(  
    padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),  
    child: CupertinoTextField(  
        placeholder: "Masukkan Password",  
        obscureText: true,  
    )),
```



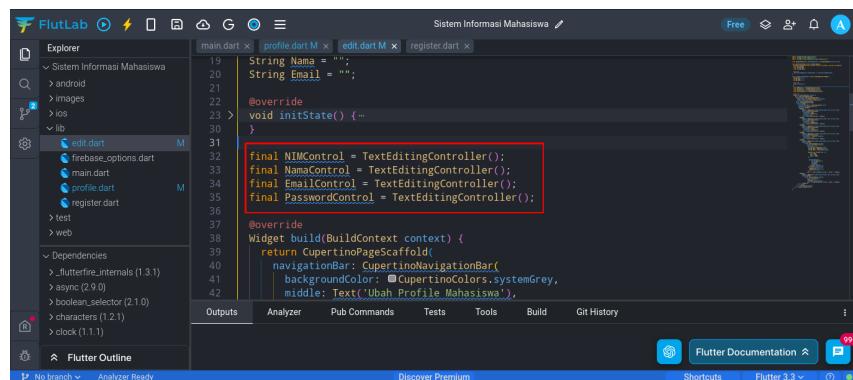
14. Tes aplikasi kembali untuk melihat **progress**. Halaman **Ubah Profil** akan menampilkan data sebelumnya sebagai **Placeholder/Hint**



15. Setelah sukses, tambahkan kode untuk **Controller** masing-masing **TextField**. Letakkan tepat di bawah **void initState()**:

Potongan Kode

```
final NIMControl = TextEditingController();
final NamaControl = TextEditingController();
final EmailControl = TextEditingController();
final PasswordControl = TextEditingController();
```

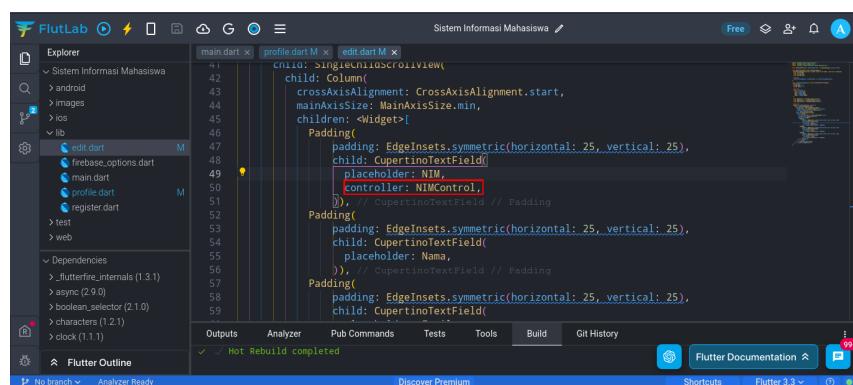


16. Tempelkan masing-masing **controller** ke **TextField** nya

- **NIM TextField**

Potongan Kode

```
controller: NIMControl,
```



- **Nama TextField**

Potongan Kode

controller: NamaControl,

```

    child: CupertinoTextField(
      placeholder: NIM,
      controller: NIMControl,
    )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: CupertinoTextField(
        placeholder: Nama,
        controller: NamaControl,
      )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: CupertinoTextField(
        placeholder: Email,
      )), // CupertinoTextField // Padding
    ],
  ), // Column
), // SingleChildScrollView
); // CupertinoPageScaffold
  
```

• Email TextField

Potongan Kode

controller: EmailControl,

```

    child: CupertinoTextField(
      placeholder: Nama,
      controller: NamaControl,
    )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: CupertinoTextField(
        placeholder: Email,
        controller: EmailControl,
      )), // CupertinoTextField // Padding
    ],
  ), // Column
), // SingleChildScrollView
); // CupertinoPageScaffold
  
```

• Password TextField

Potongan Kode

controller: PasswordControl,

```

    child: CupertinoTextField(
      controller: NamaControl,
    )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: CupertinoTextField(
        placeholder: Email,
        controller: EmailControl,
      )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: CupertinoTextField(
        placeholder: "masukkan Password",
        obscureText: true,
        controller: PasswordControl,
      )), // CupertinoTextField // Padding
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
      child: Center(
    
```

17. Agar NIM ditampilkan dalam bentuk **Teks** masukkan kode berikut di dalam fungsi **void initState()**

Potongan Kode

NIMControl.text = NIM;



The screenshot shows the FlutLab IDE interface with the following details:

- Top Bar:** Shows the FlutLab logo, file navigation icons, and a "Sistem Informasi Mahasiswa" title.
- Left Sidebar (Explorer):** Lists project files including `main.dart`, `profile.dart`, `edit.dart`, `register.dart`, `lib`, and `web`. The `edit.dart` file is currently selected.
- Code Editor:** Displays the `main.dart` file with code completion suggestions. The line `NIMControl.text = NIM;` is highlighted with a red box, and a tooltip "NIMControl" is visible above it.
- Bottom Navigation:** Includes tabs for Outputs, Analyzer, Pub Commands, Tests, Tools, Build, and Git History.
- Bottom Status Bar:** Shows "Flutter Outliner", "No branch", "Analyzer Ready", "Discover Premium", "Shortcuts", "Flutter 3.3", and system status icons.

18. Lalu matikan input dengan menambahkan kode berikut setelah **controller**: NIM-Control,

Potongan Kode

enabled:false,

19. Tambahkan kode Tombol yang diletakkan satu level dengan Padding Email.

Potongan Kode

Padding(

```
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),  
child: Center(  
    child: CupertinoButton.filled(  
        child: Text("Ubah Profil"),  
        onPressed: () {}
```

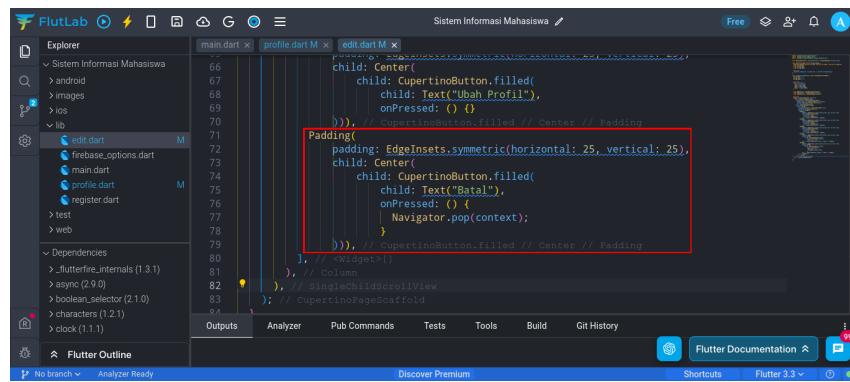
10

20. Tambahkan tombol terakhir:

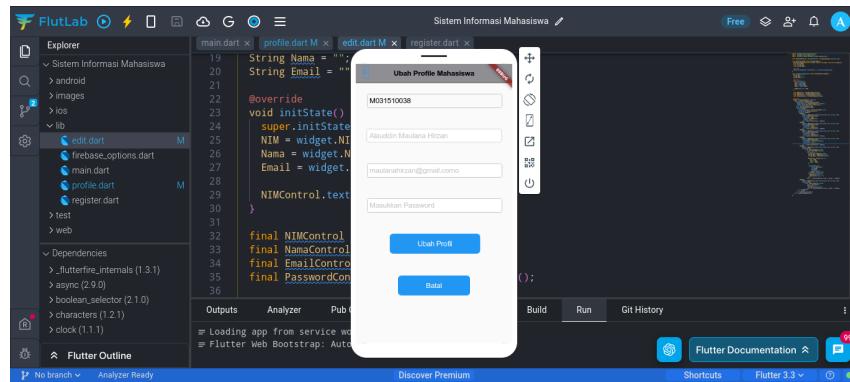
Potongan Kode

Padding(

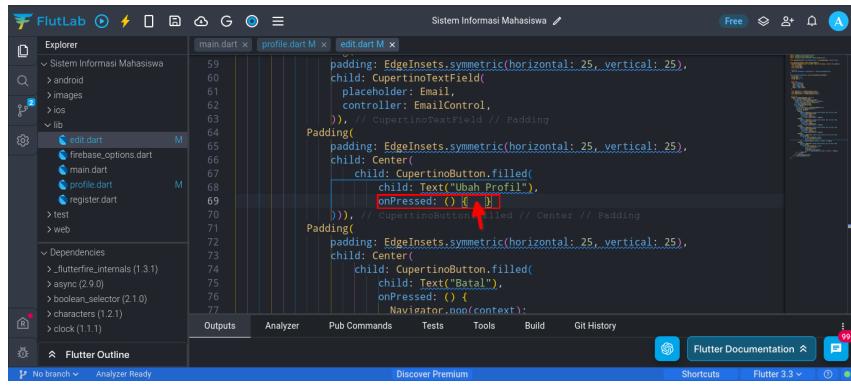
```
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: Center(
child: CupertinoButton.filled(
child: Text("Batal"),
onPressed: () {
Navigator.pop(context);
}
)),
),
));,
```



21. Tes aplikasi lagi untuk melihat hasil keseluruhan antarmuka



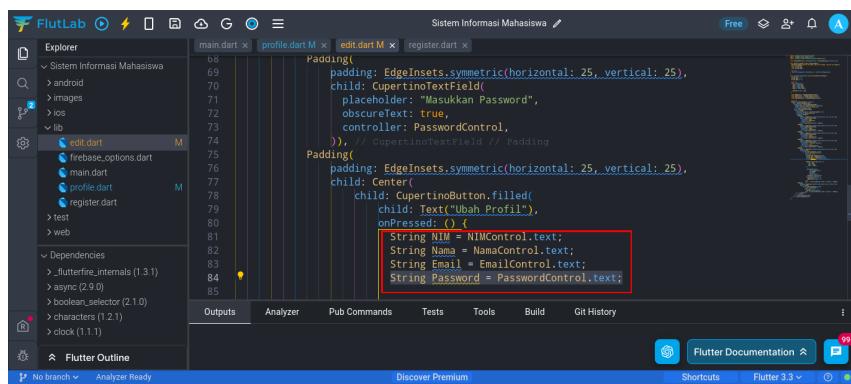
22. Bagian terakhir dari praktikum ini adalah menambahkan kode untuk melakukan update. Tambahkan kode berikut untuk tombol **Update Profil** di bagian **onPressed()**



```
Sistem Informasi Mahasiswa
main.dart profile.dart edit.dart
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
placeholder: Email,
controller: EmailControl,
)),
), // CupertinoTextField // Padding
Padding(
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: Center(
child: CupertinoButton.filled(
child: Text("Ubah Profil"),
onPressed: () {
}),
), // CupertinoButton.filled // Center // Padding
Padding(
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: Center(
child: CupertinoButton.filled(
child: Text("Batal"),
onPressed: () {
Navigator.pop(context);
}),
), // CupertinoButton.filled // Center // Padding
Outputs Analyzer Pub Commands Tests Tools Build Git History
Discover Premium Flutter Documentation Shortcuts Flutter 3.3
```

Potongan Kode

```
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
```

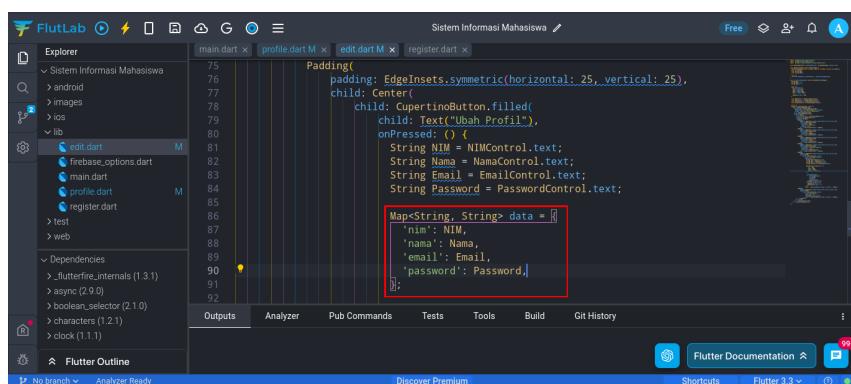


```
Sistem Informasi Mahasiswa
main.dart profile.dart edit.dart register.dart
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
placeholder: "Masukkan Password",
obscureText: true,
controller: PasswordControl,
),
), // CupertinoTextField // Padding
Padding(
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: Center(
child: CupertinoButton.filled(
child: Text("Ubah Profil"),
onPressed: () {
}),
), // CupertinoButton.filled // Center // Padding
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
Outputs Analyzer Pub Commands Tests Tools Build Git History
Discover Premium Flutter Documentation Shortcuts Flutter 3.3
```

23. Lalu petakan variabel dalam bentuk Map dengan kode berikut

Potongan Kode

```
Map<String, String> data = {
  'nim': NIM,
  'nama': Nama,
  'email': Email,
  'password': Password,
};
```

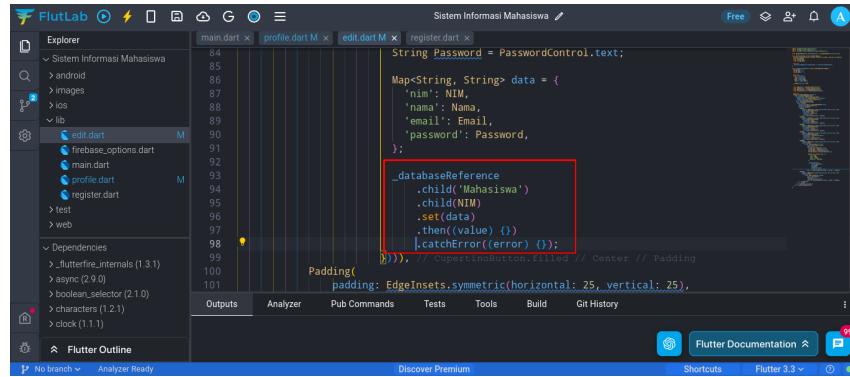


```
Sistem Informasi Mahasiswa
main.dart profile.dart edit.dart register.dart
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: Center(
child: CupertinoButton.filled(
child: Text("Ubah Profil"),
onPressed: () {
},
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
Map<String, String> data = [
  'nim': NIM,
  'nama': Nama,
  'email': Email,
  'password': Password,
];
Outputs Analyzer Pub Commands Tests Tools Build Git History
Discover Premium Flutter Documentation Shortcuts Flutter 3.3
```

24. Jika sudah lanjutkan dengan kode Query seperti berikut

Potongan Kode

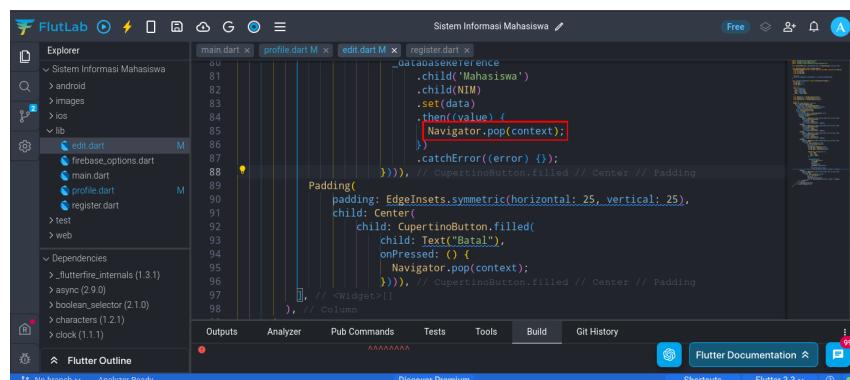
```
_databaseReference  
    .child('Mahasiswa')  
    .child(NIM)  
    .set(data)  
    .then((value) {})  
    .catchError((error){});
```



25. Untuk merespon keberhasilan Query, masukkan kode berikut tepat di dalam `.then((value){})`

Potongan Kode

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



26. Terakhir, untuk merespon kegagalan kueri. Masukkan kode untuk mengosongkan semua **TextField**. Masukkan di dalam kode `.catchError((error) {});`

```

    ...
    _dataReference
      .child('Mahasiswa')
      .child(NIM)
      .set(data)
      .then((value) {
        Navigator.pop(context);
      })
    } catch(error) {
      ...
    }
  }
}

```

Potongan Kode

```

NamaControl.text = "";
EmailControl.text= "";
PasswordControl.text = "";

```

```

...
try {
  Map<String, String> data = {
    'nim': NIM,
    'name': Nama,
    'email': Email,
    'password': Password,
  };

  _databaseReference
    .child('Mahasiswa')
    .child(NIM)
    .set(data)
    .then((value) {
      Navigator.pop(context);
    }) catch(error) {
      NamaControl.text = "";
      EmailControl.text = "";
      PasswordControl.text = "";
    }
}

```

27. Tes aplikasi

Ubah Profil Mahasiswa

M031510038

AMH

maulana@gmail.com

Ubah Profil

Batal

Profile Mahasiswa

NIM M031510038

Nama AMH

E-Mail maulana@gmail.com

Ubah Profil

Logout

Bab 8

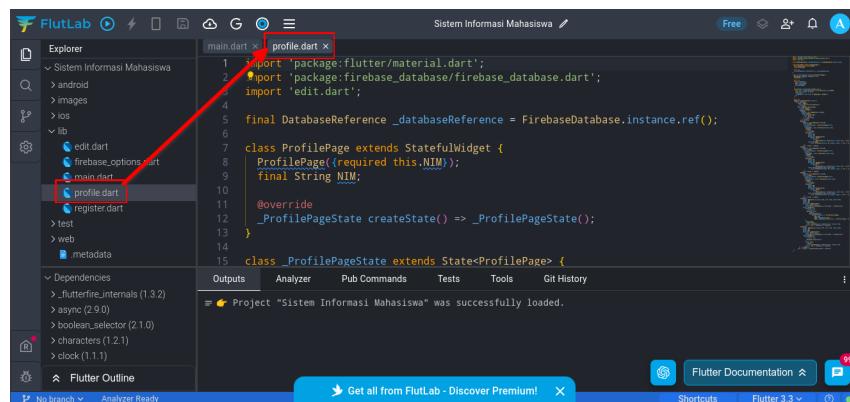
Praktikum 8

8.1 Operasi Delete Data

Di bagian ini mahasiswa diajarkan bagaimana mengunggah gambar melalui aplikasi, yang kemudian ditampilkan di halaman profile. Mahasiswa diwajibkan menyelesaikan Praktikum 7

8.2 Tutorial

1. Buka kembali projek di web **flutlab**. Buka kembali file **profile.dart**



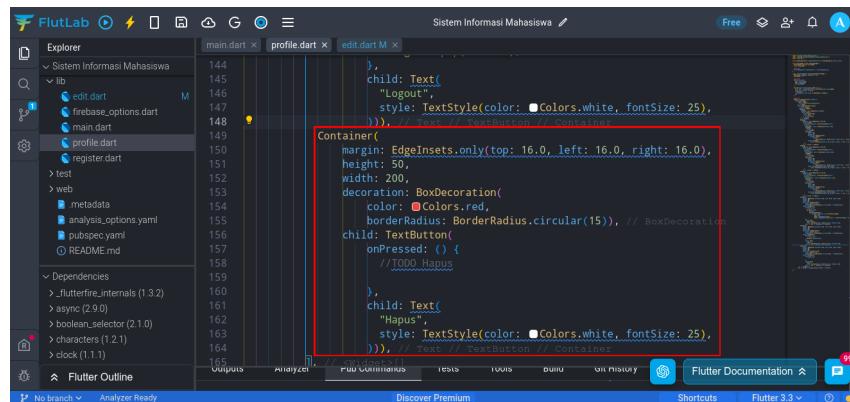
The screenshot shows the FlutLab IDE interface. The top navigation bar includes tabs for Home, Projects, and Code. The main area has tabs for Explorer, Sistem Informasi Mahasiswa, and Code. The Explorer tab shows the project structure with files like main.dart, profile.dart, edit.dart, firebase_options.dart, and register.dart. The Code tab displays the Dart code for the ProfilePage class. A red arrow points from the text 'Buka kembali file profile.dart' to the profile.dart file in the Explorer list.

```
main.dart profile.dart x
1 import 'package:flutter/material.dart';
2 import 'package:firebase_database/firebase_database.dart';
3 import 'edit.dart';
4
5 final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();
6
7 class ProfilePage extends StatefulWidget {
8   ProfilePage({required this.NIM});
9   final String NIM;
10
11   @override
12   _ProfilePageState createState() => _ProfilePageState();
13 }
14
15 class _ProfilePageState extends State<ProfilePage> {
```

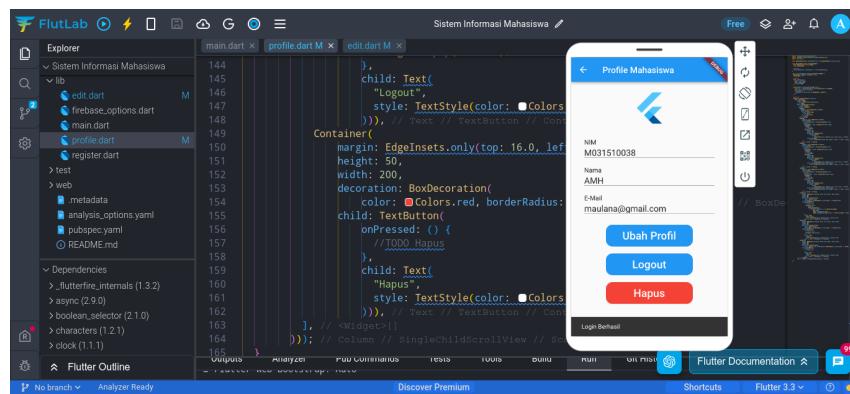
2. Tambahkan satu tombol baru yang berfungsi untuk menghapus data, dan **Logout** ketika sukses.

Potongan Kode

```
Container(  
    margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),  
    height: 50,  
    width: 200,  
    decoration: BoxDecoration(  
        color: Colors.red,  
        borderRadius: BorderRadius.circular(15)),  
    child: TextButton(  
        onPressed: () {  
            //TODO Hapus  
  
        },  
        child: Text(  
            "Hapus",  
            style: TextStyle(color: Colors.white, fontSize: 25),  
    )),,
```



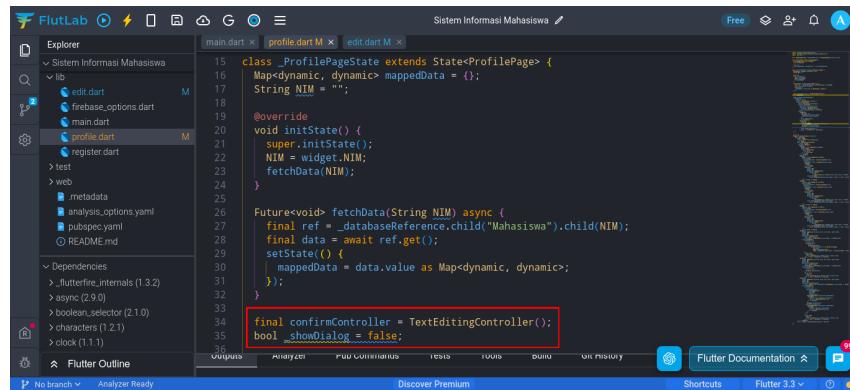
3. Jalankan aplikasi untuk melihat hasil sementara aplikasi



4. Jika tombol sudah ditambahkan, berikutnya menambahkan variabel dan fungsi untuk menampilkan **Kotak Dialog Konfirmasi** dan **Input Password**. Masukkan kode tepat di bawah fungsi `Future<void> fetchData` (BUKAN DI DALAM-NYA).

Potongan Kode

```
final confirmController = TextEditingController();
bool _showDialog = false;
```



5. Untuk membangun **Dialog Konfirmasi**, lanjutkan kode di atas dengan kode berikut:

Potongan Kode

```
void _openConfirmationDialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Dialog Konfirmasi'),
        content: TextField(
          controller: confirmController,
          decoration: InputDecoration(
            labelText: 'Masukkan Password',),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
                confirmController.clear();
              },
              child: Text('Batal'),),
            TextButton(
              onPressed: () {
                //TODO Hapus
              },
              child: Text('Konfirmasi'),),
          ],
        );
    });
}
```

```

final TextEditingController confirmController = TextEditingController();
bool _showDialog = false;

void _openConfirmationDialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Dialog Konfirmasi'),
        content: TextField(
          controller: confirmController,
          decoration: InputDecoration(
            labelText: "Masukkan Password",
          )),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              confirmController.clear();
            },
            child: Text('Batal'),
          ),
          TextButton(
            onPressed: () {
              //TODO Hapus
            },
            child: Text('Konfirmasi'),
          ),
        ],
      );
    },
  );
}

```

6. Agar fungsi ini dapat menjalankan tugasnya, masukkan kode berikut tepat di bawah tulisan //TODO Hapus

Potongan Kode

```

String curPass = confirmController.text;
_databaseReference
  .child("Mahasiswa")
  .child(NIM)
  .get()
  .then((snapshot) {
    Map<dynamic, dynamic> data =
      snapshot.value as Map<dynamic, dynamic>;
    String dbPass = data['password'];
}).catchError((error) {});

```

```

final TextEditingController confirmController = TextEditingController();
bool _showDialog = false;

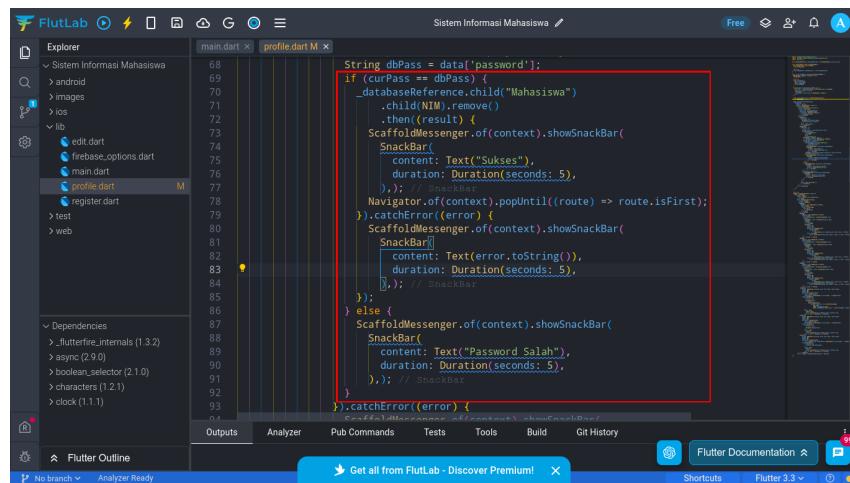
void _openConfirmationDialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Dialog Konfirmasi'),
        content: TextField(
          controller: confirmController,
          decoration: InputDecoration(
            labelText: "Masukkan Password",
          )),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              confirmController.clear();
            },
            child: Text('Batal'),
          ),
          TextButton(
            onPressed: () {
              //TODO Hapus
              String curPass = confirmController.text;
              _databaseReference
                .child("Mahasiswa")
                .child(NIM)
                .get()
                .then((snapshot) {
                  Map<dynamic, dynamic> data =
                    snapshot.value as Map<dynamic, dynamic>;
                  String dbPass = data['password'];
                }).catchError((error) {});
            },
            child: Text('Konfirmasi'),
          ),
        ],
      );
    },
  );
}

```

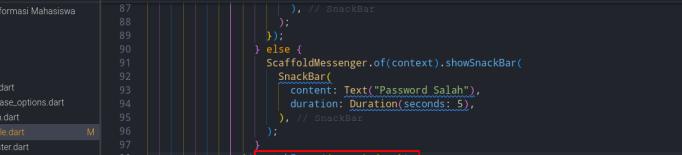
7. Kemudian lanjutkan bagian bawah kode dari **String dbPass = data['password'];** dengan kode berikut:

Potongan Kode

```
if (curPass == dbPass) {
    _databaseReference.child("Mahasiswa")
        .child(NIM).remove()
    .then((result) {
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
                content: Text("Sukses"),
                duration: Duration(seconds: 5),
            ),);
        Navigator.of(context).popUntil((route) => route.isFirst);
    }).catchError((error) {
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
                content: Text(error.toString()),
                duration: Duration(seconds: 5),
            ),);
    });
} else {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text("Password Salah"),
            duration: Duration(seconds: 5),
        ),);
}
```



8. Berikutnya adalah menambahkan kode error untuk bagian kode `.catchError((error) {});`



The screenshot shows the FlutLab IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like main.dart, profile.dart, register.dart, and main.dart.
- Code Editor:** The main editor area displays the `profile.dart` file. A red box highlights the line `}).catchError((error) { })`, which is part of an `onPressed` event handler for a button.
- Bottom Navigation Bar:** The bar includes tabs for Outputs, Analyzer, Pub Commands, Tests, Tools, and Git History, along with a Flutter Documentation link.

```
Potongan Kode _  
ScaffoldMessenger.of(context).showSnackBar(  
    SnackBar(  
        content: Text(error.toString()),  
        duration: Duration(seconds: 5),  
    ),  
);
```

The screenshot shows the FlutLab IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like `main.dart`, `profile.dart`, `edit.dart`, `firebase_options.dart`, `main.dart`, `register.dart`, `test`, and `web`. `profile.dart` is currently selected.
- Code Editor:** The main editor area displays Dart code for a password confirmation feature. It includes logic for showing success or error snackbars and a confirmation text button.
- Toolbars:** The top bar includes tabs for `System Informasi Mahasiswa`, `Free`, and `Flutter Documentation`.
- Bottom Bar:** The bottom navigation bar includes buttons for `Outputs`, `Analyzer`, `Pub Commands`, `Tests`, `Tools`, and `Git History`.

9. Bagian terakhir yang perlu ditambahkan adalah kode untuk **Tombol Hapus** itu sendiri

The screenshot shows the FlutLab IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like `main.dart`, `profile.dart`, `edit.dart`, etc.
- Code Editor:** The main editor area displays `profile.dart` code. A red box highlights the `onPressed: () {` part of a button definition, indicating where the user can type the completion code.
- Code Completion:** A red arrow points from the code editor to the bottom right corner of the screen, which displays the message "Get all from FlutLab - Discover Premium!"
- Bottom Navigation:** The navigation bar includes tabs for `Outputs`, `Analyzer`, `Pub Commands`, `Tests`, `Tools`, and `Git History`.
- Bottom Right:** The bottom right corner features the FlutLab logo, a "Flutter Documentation" link, and a "Flutter 3.3" version indicator.

`openConfirmationDialog()`:

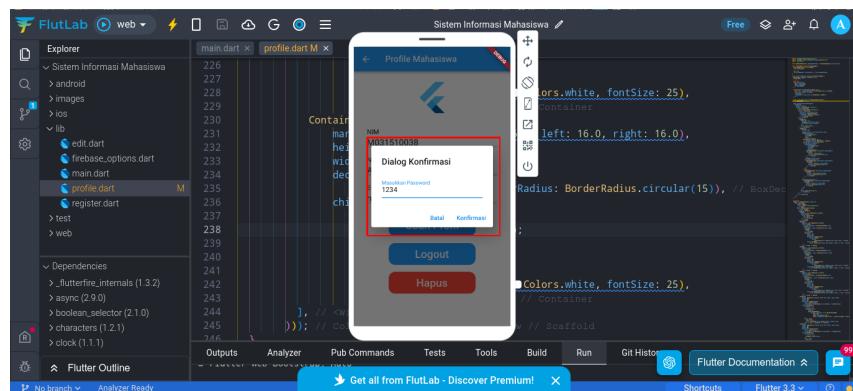
```

Sistem Informasi Mahasiswa
main.dart profile.dart M

226 child: Text(
227   "Logout",
228   style: TextStyle(color: Colors.white, fontSize: 25),
229   )), // Text // TextButton // Container
230 Container(
231   margin: EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
232   height: 50,
233   width: 200,
234   decoration: BoxDecoration(
235     color: Colors.red, borderRadius: BorderRadius.circular(15)), // BoxDecoration
236   child: TextButton(
237     onPressed: () {
238       _openConfirmationDialog();
239     },
240     child: Text(
241       "Hapus",
242       style: TextStyle(color: Colors.white, fontSize: 25),
243     ), // Text // TextButton // Container
244   ), // Container // Column // SingleChildScrollView // Scaffold
245 }
246 )
247 }
248

```

10. Jalankan aplikasi dan coba hapus user



```

Project Overview
Realtime Database
Data Rules Backups Usage Extensions NEW
https://praktikum-mobileapp-878bb-default.firebaseio.app/.json
⚠ Your security rules are defined as public, so anyone can steal, modify, or delete data in your database
Learn more Dismiss
Database location: Singapore (asia-southeast1)

```

11. Penghapusan sudah berhasil dan secara otomatis akun terhapus