



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 Firestore 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 Firestore 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 Firestore	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 Mengetahui 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 Mengetahui 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",
    }
  ]
}
```

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

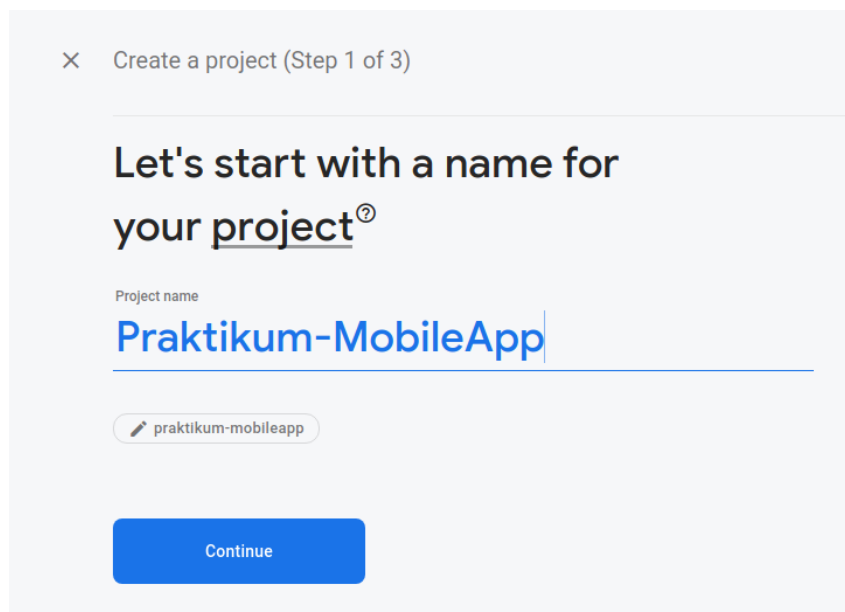
Praktikum 1

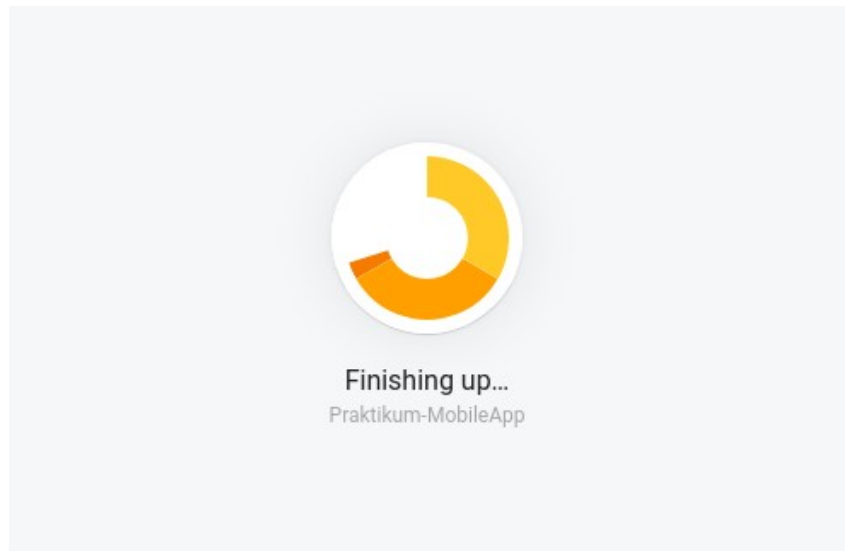
1.1 Konfigurasi Firebase Realtime DB dan Flutter IDE

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.

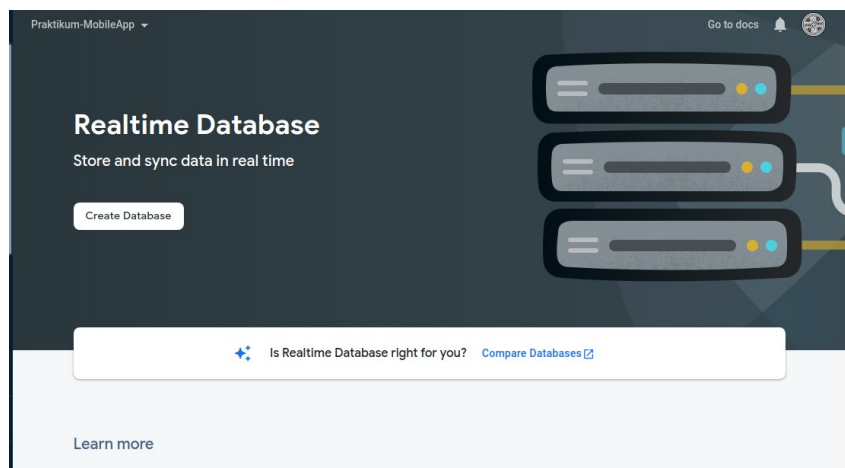
1.2 Tutorial

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**.

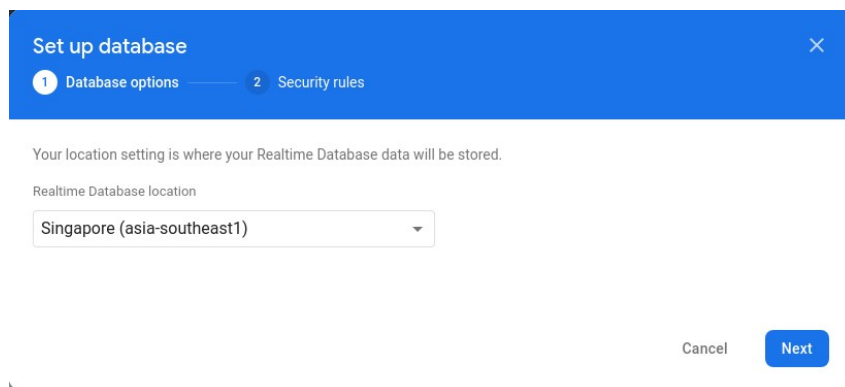




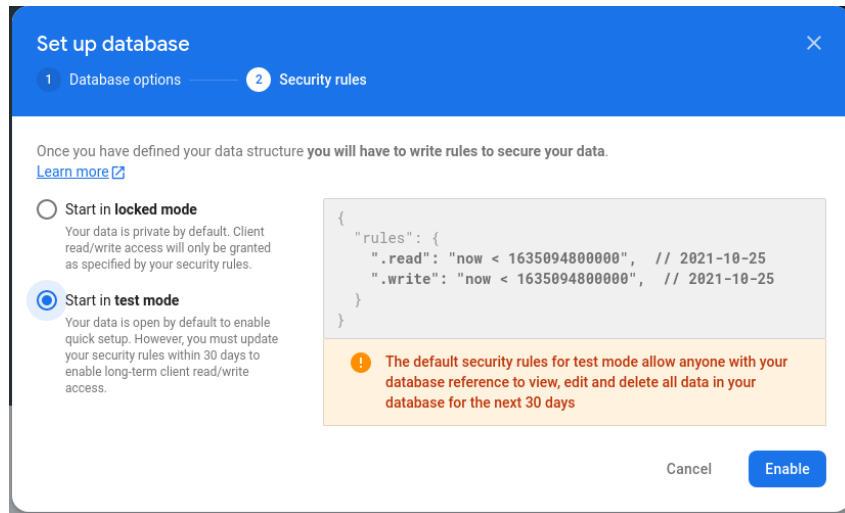
2. Untuk membuat **Realtime Database**, klik **Menu Build** yang ada di samping kiri > pilih **Realtime Database** > klik **Create Database**



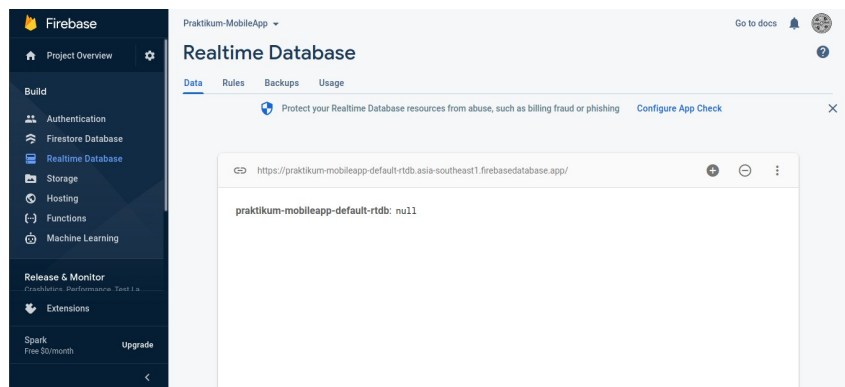
3. Pilih lokasi database (**Singapore**), klik **Next**



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



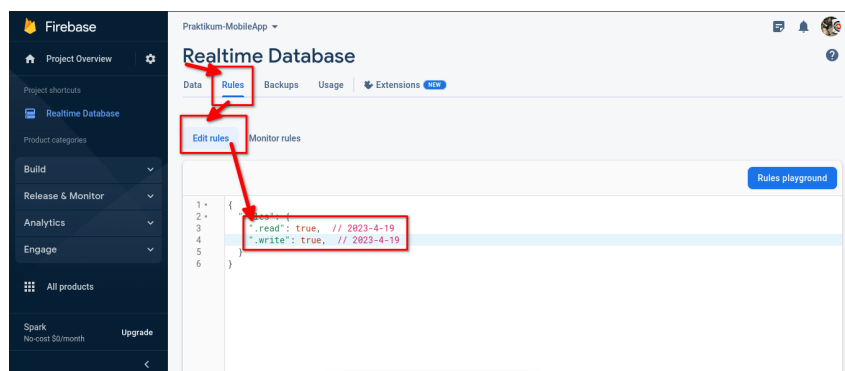
5. Tunggu hingga database muncul dan siap dipakai.



6. Setelah Database sudah siap, berikutnya adalah mengonfigurasi **Rules** yang sudah dibuat oleh Firebase. Hal ini memastikan database bisa diakses kapanpun

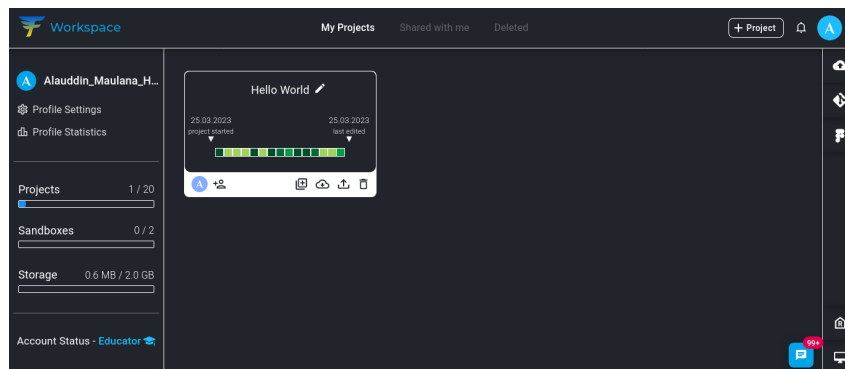
7. Klik tab **Rules** → **Edit rules** → Ubah data berikut :

- "now < xxxxxxxxxxxx" menjadi **true**

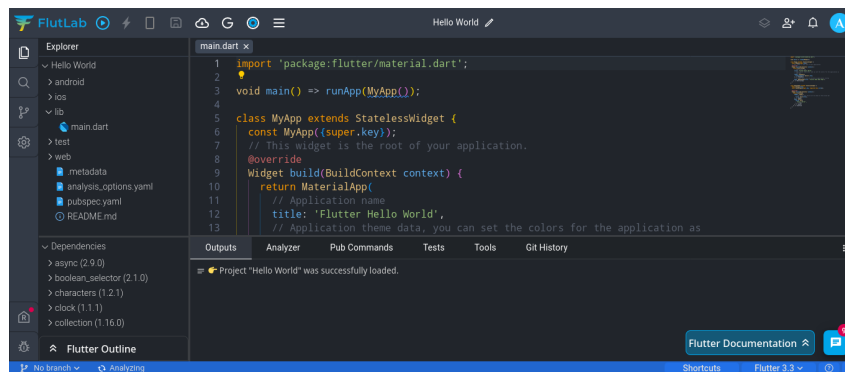


8. Klik **Publish** untuk menerapkan perubahan.

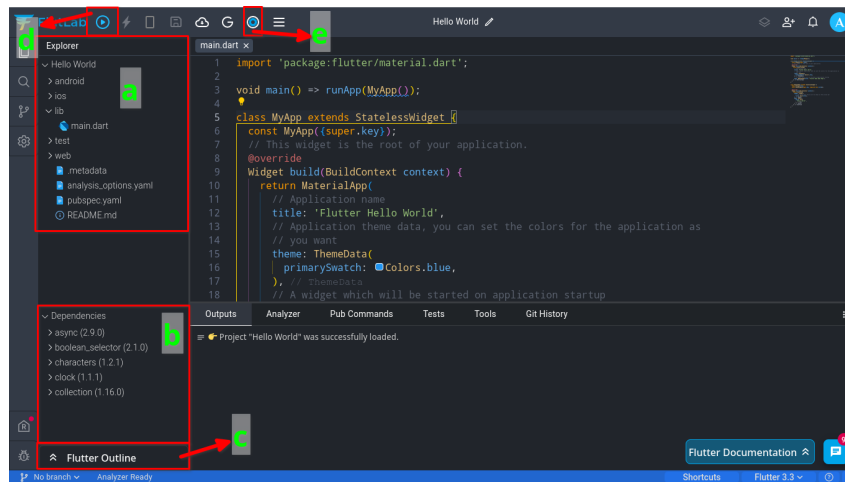
9. Berikutnya buka web <https://flutlab.io/workspace>. Buatlah akun dengan **E-Mail Pribadi**, dan buka kembali link tersebut hingga terlihat seperti gambar di bawah:



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



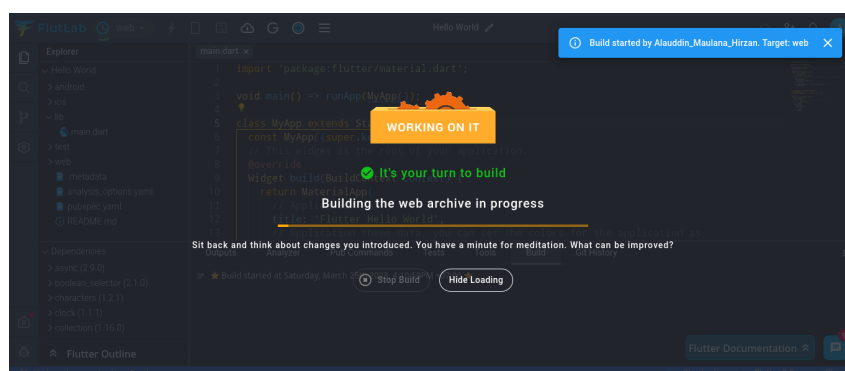
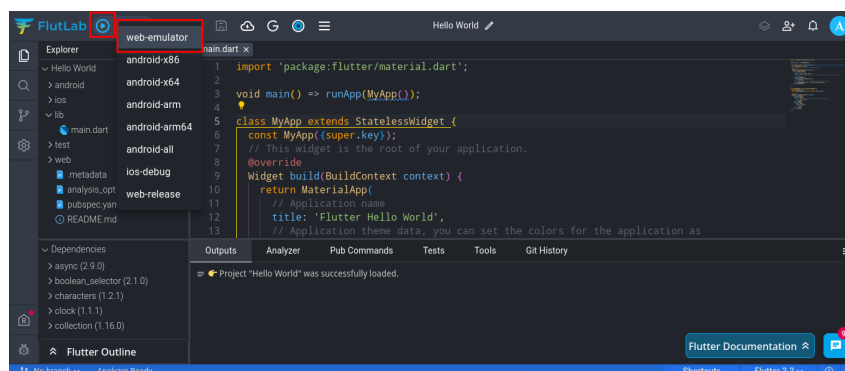
11. Ada beberapa bagian yang harus dipahami oleh mahasiswa:
- (a) Struktur Folder
 - (b) Ketergantungan Aplikasi
 - (c) Navigasi Item Aplikasi
 - (d) Build dan Play Aplikasi
 - (e) Analyzer Kebenaran Kode

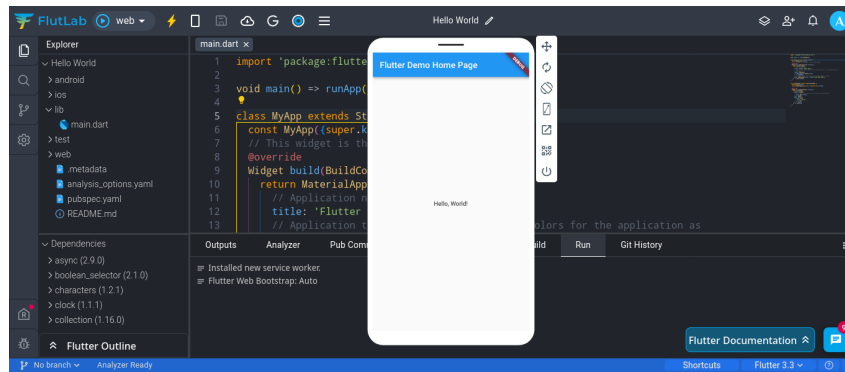


12. Untuk menguji coba aplikasi dari projek saat ini bisa dalam beberapa mode:

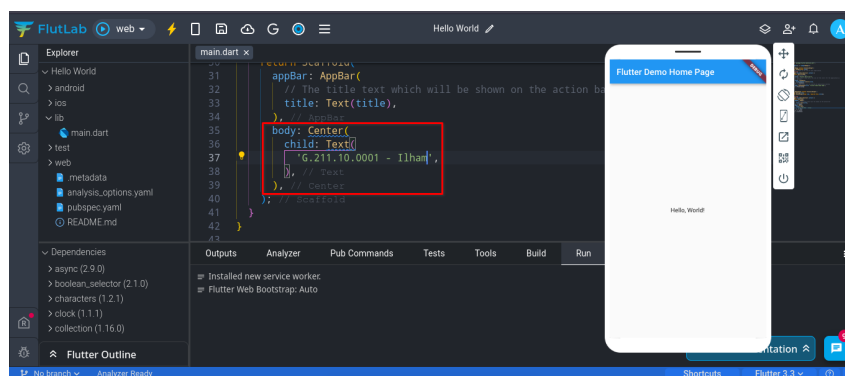
- Web (Rekomendasi untuk preview)
- Android APK (1 kali per hari, hanya untuk final)
- iPhone IPS (1 kali per hari, hanya untuk final)

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.





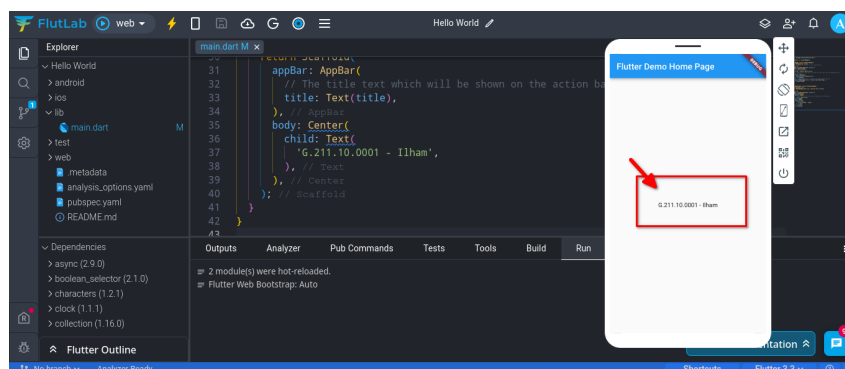
14. Scroll turun **Kode** dan temukan tulisan 'Hello, World!' dan ubah menjadi NIM - Nama



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**



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

Bab 2

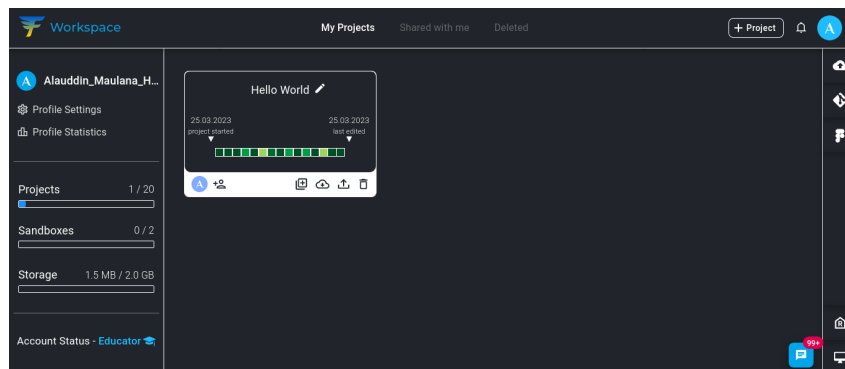
Praktikum 2

2.1 Antarmuka Login Flutter

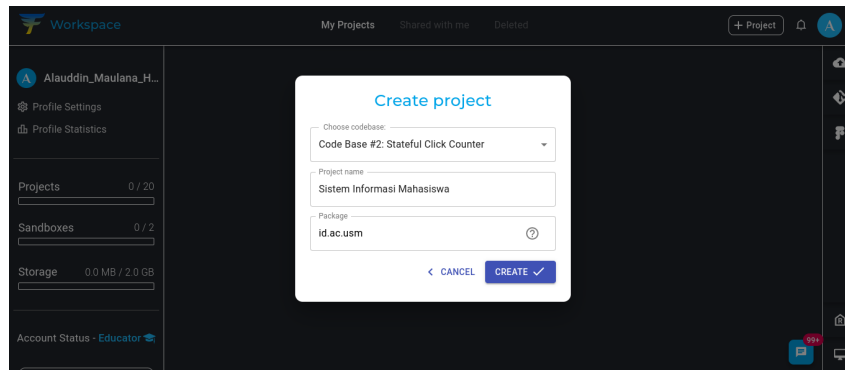
Di bagian ini mahasiswa diperkenalkan dengan pemrograman DART untuk membuat tampilan login melalui IDE Flutter. Mahasiswa diwajibkan untuk menyelesaikan pertemuan sebelumnya agar praktikum ini berjalan dengan baik.

2.2 Tutorial

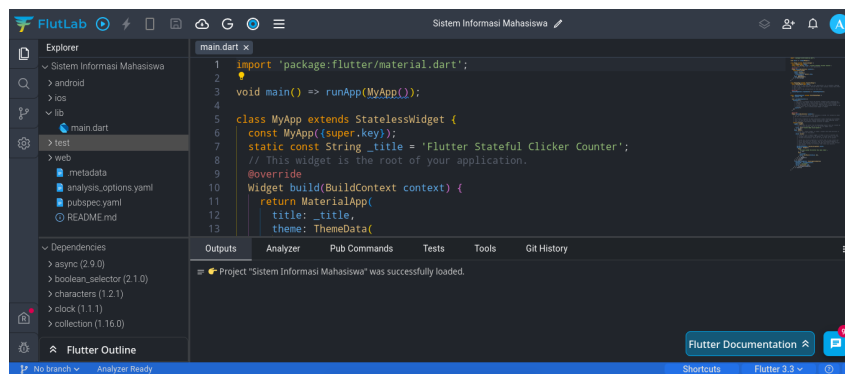
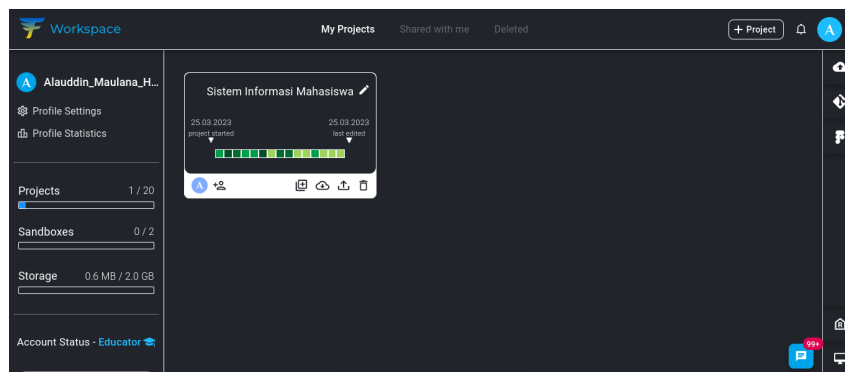
1. Buka web <https://flutlab.io/workspace>, hapus projek sebelumnya karena limi-tasi dari Flutlab.



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



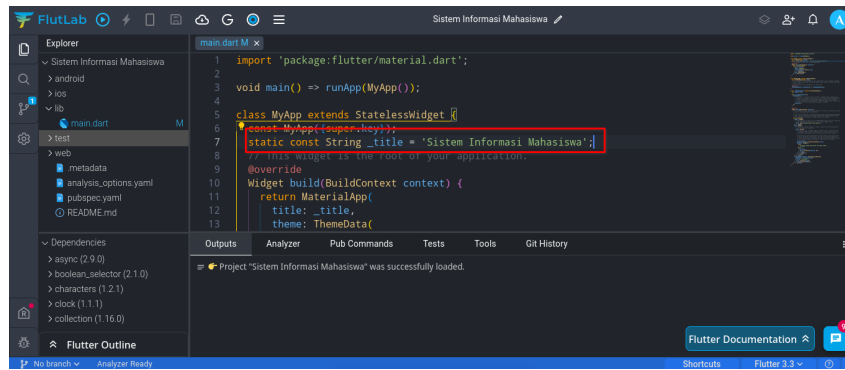
4. Di kode ada bagian-bagian yang perlu dirubah. Ubahlah judul aplikasi dengan mengganti kode berikut (Sebelum → Sesudah):

Potongan Kode (Sebelum)

```
static const String _title = 'Flutter Stateful Clicker Counter';
```

Potongan Kode (Sesudah)

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

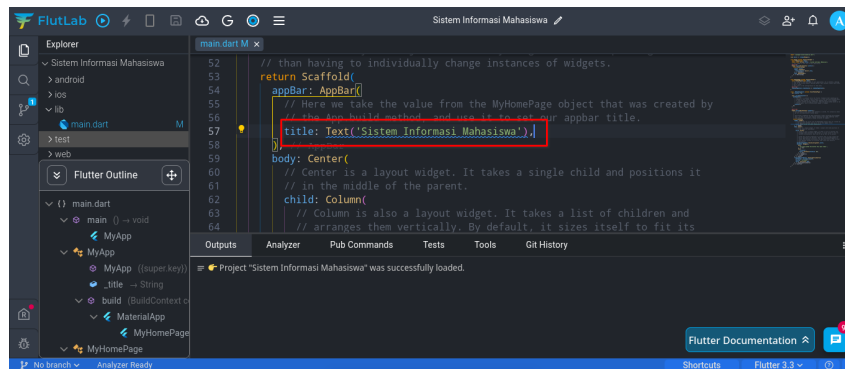
5. Berikutnya adalah mengubah tampilan **Judul App** yang ditampilkan oleh aplikasi

Potongan Kode (Sebelum)

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

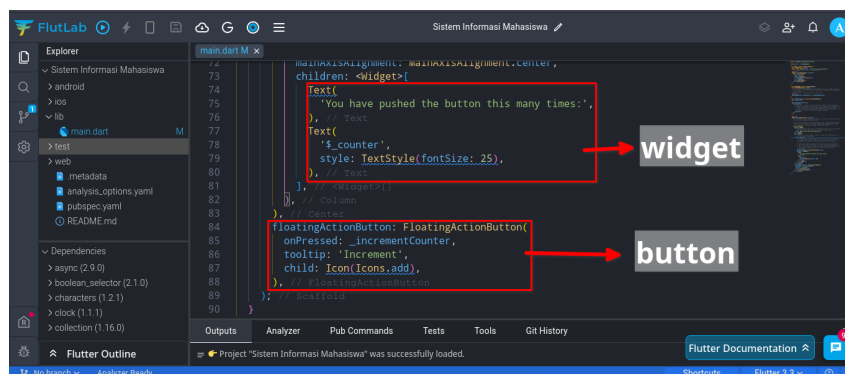
Potongan Kode (Sesudah)

```
title: Text('Sistem Informasi Mahasiswa'),
```

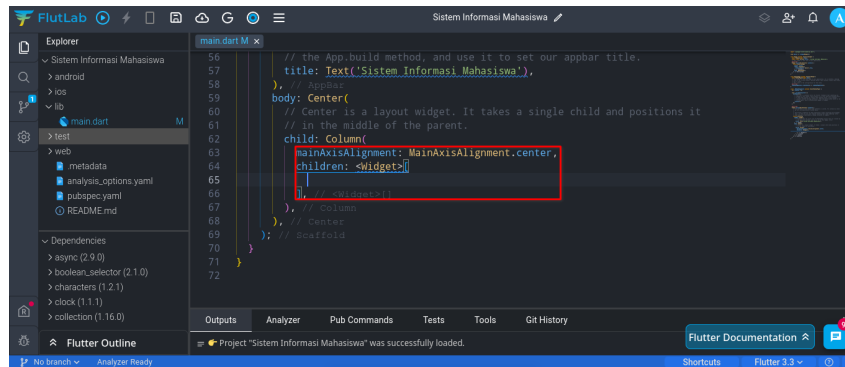


6. Berikutnya adalah mengosongkan widget-widget yang ada di tengah aplikasi, cukup menghapus kode yang berada di dalam **children: <Widget>[]** (Bisa dilihat via **Flutter Outline**)

7. Hapus bagian yang ditandai di gambar berikut

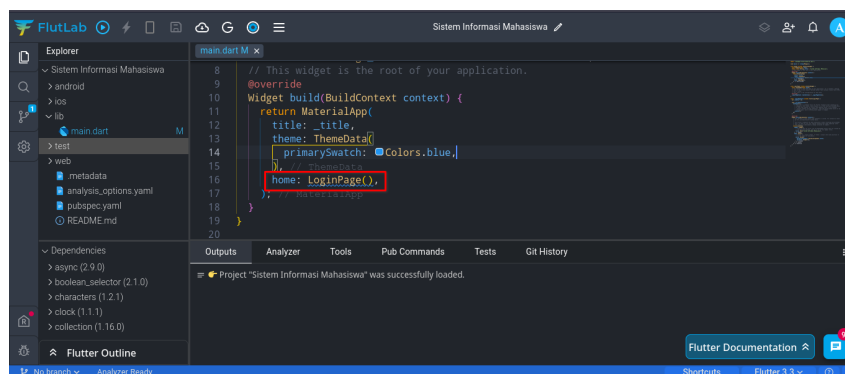


8. Jika sudah dihapus akan terlihat seperti berikut (children: <Widget>[] sudah tidak memiliki widget apapun):

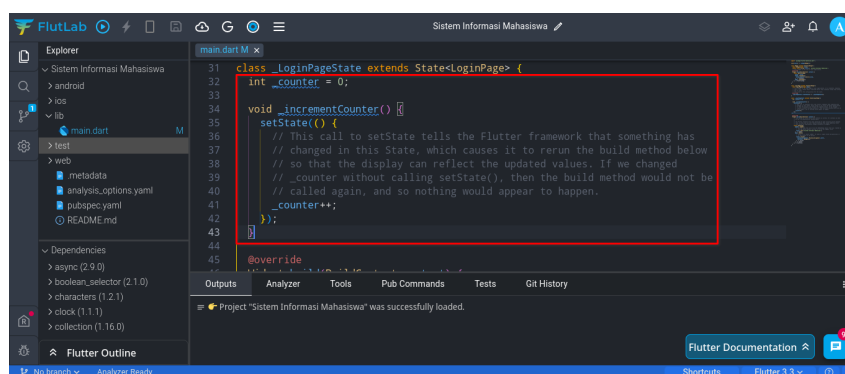


9. Sebelum mulai memasukkan kode tampilan aplikasi, ada beberapa bagian kode yang dirubah (Abaikan Error sampai kode selesai diubah):

- 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)



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



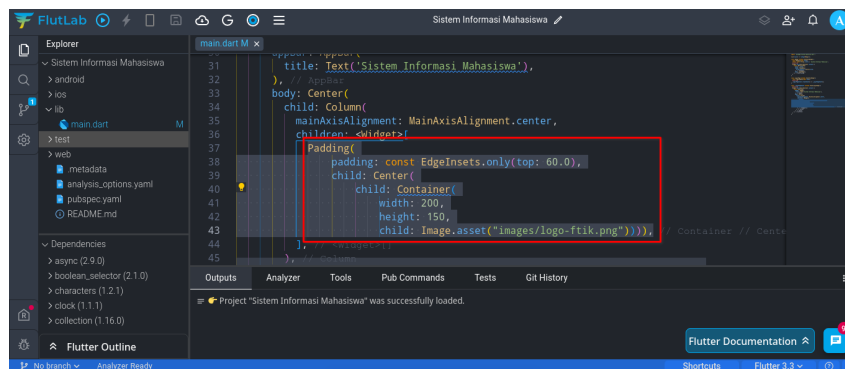
- Kode siap untuk ditambahkan widget-widget baru. Masukkan kode berikut di dalam **children:** `<Widget>` yang sudah dikosongkan sebelumnya. Pastikan memasukkan kode di dalam tag `[]`:

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"))),

```



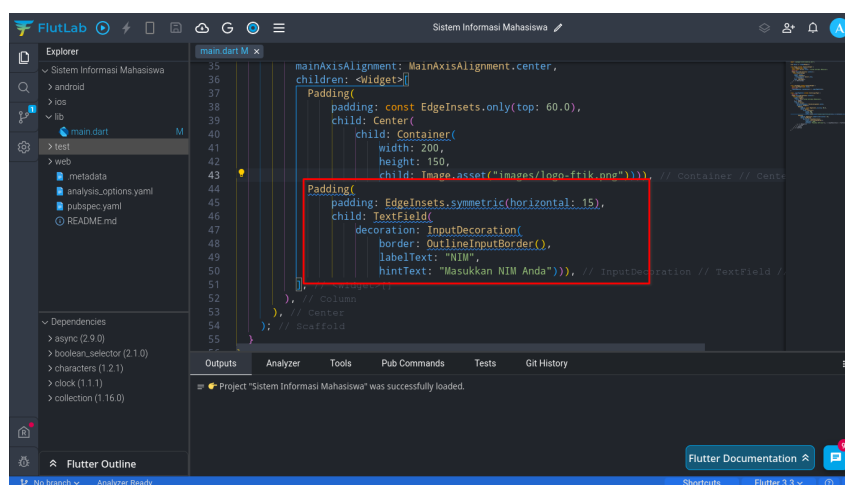
- Kemudian tambahkan kode berikut masih di dalam tag `[]` tetapi di bawah kode sebelumnya. Kode ini digunakan untuk input NIM mahasiswa.

Potongan Kode

```

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

```

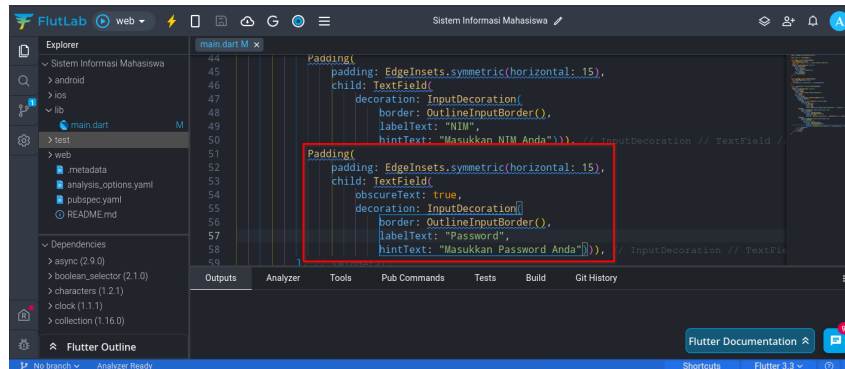


- Berikutnya menambahkan kode input untuk password di bawah kode sebelumnya

Potongan Kode

Padding(

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

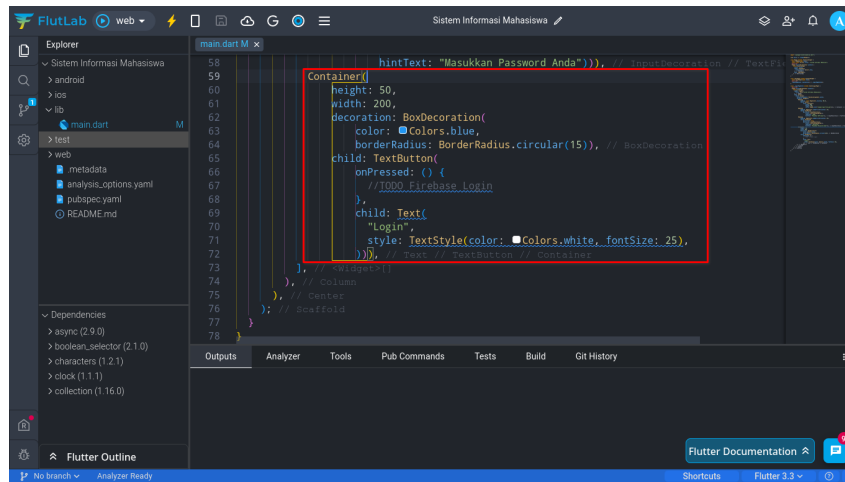


14. Setelah field input selesai, berikutnya tambahkan tombol untuk **Login**

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),
  )),
```



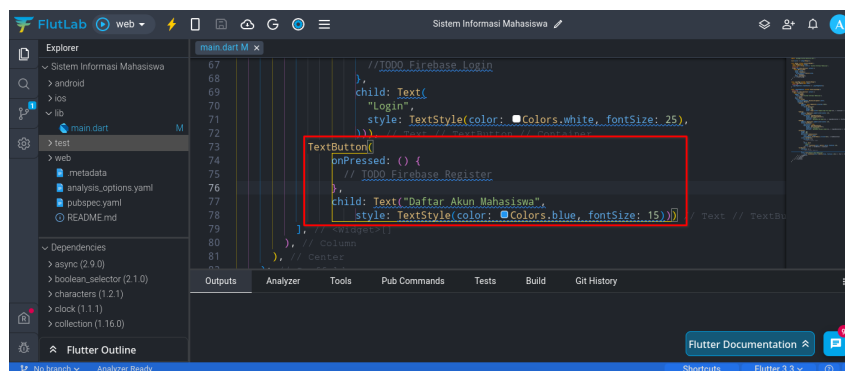
15. Terakhir adalah menambahkan kode untuk Register dengan widget jenis **TextButton**

Potongan Kode

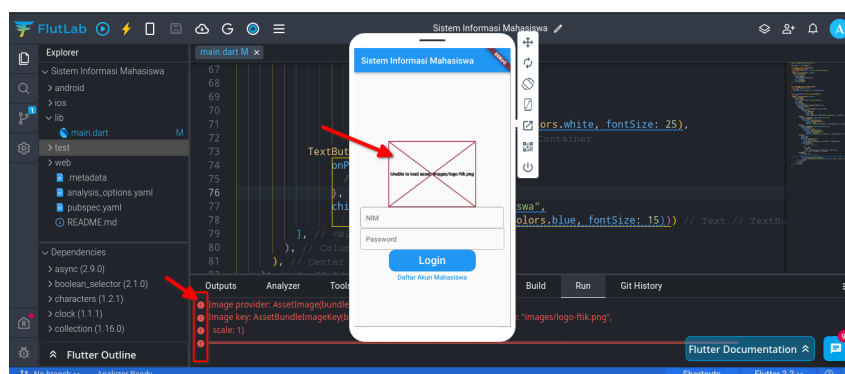
```

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

```

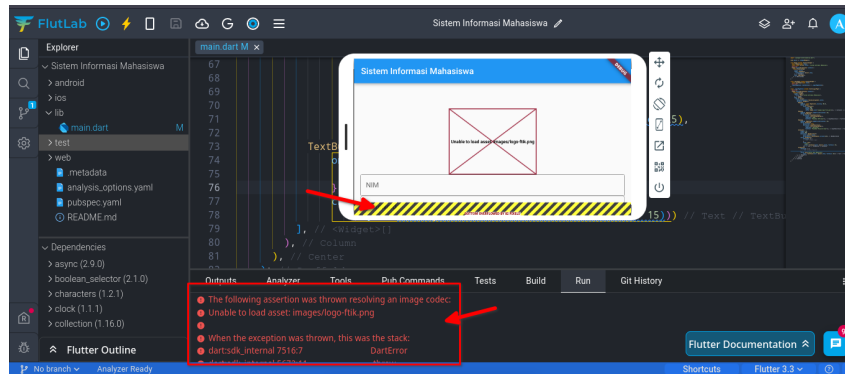


16. Cek pastikan kode tidak ada yang error. Jika ada perbaiki kode tersebut. Jika tidak ada error. Klik **Play (mode Web)** untuk melihat hasil saat ini

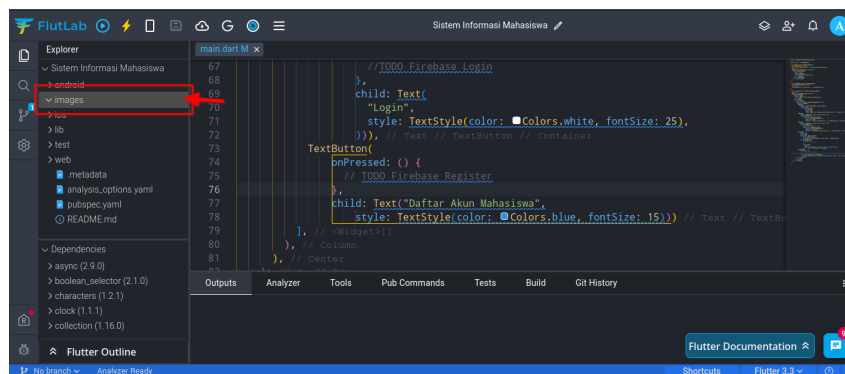


17. Tetapi banyak error yang ada di aplikasi tersebut seperti:

- Gambar tidak ada
- Tampilan rusak ketika dirotasi landscape



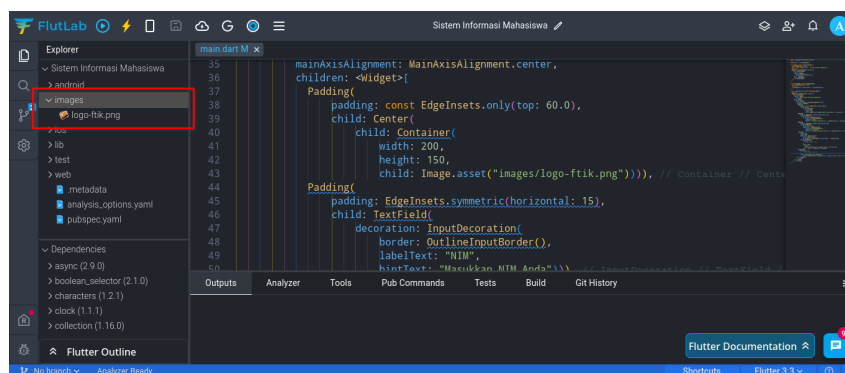
18. Untuk menambahkan gambar cukup dengan cara mudah. Buat **Folder Baru** di **Projek Sistem Informasi Mahasiswa** dengan nama **images**



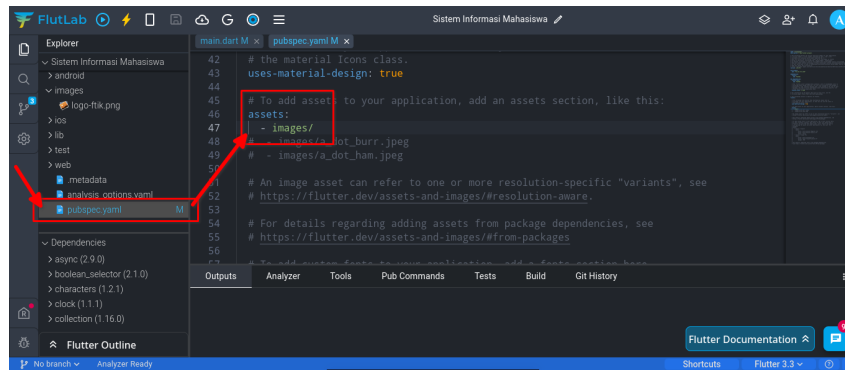
19. Lalu unduh gambar dan rename ke **logo-ftik.png**. Unduh di

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

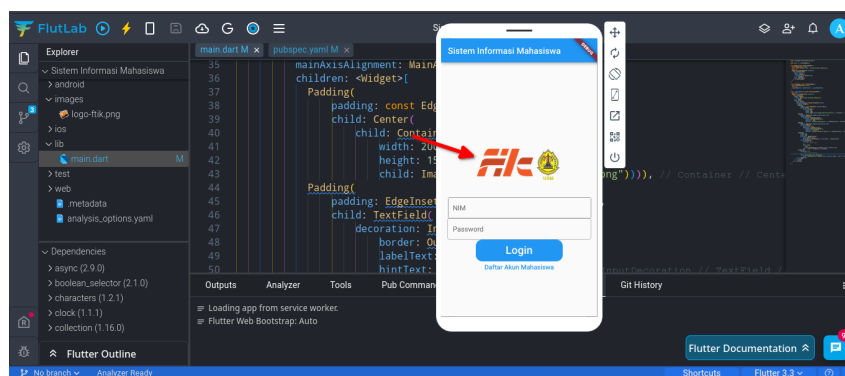
20. Unggah file yang sudah di-rename ke folder **images** tersebut



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

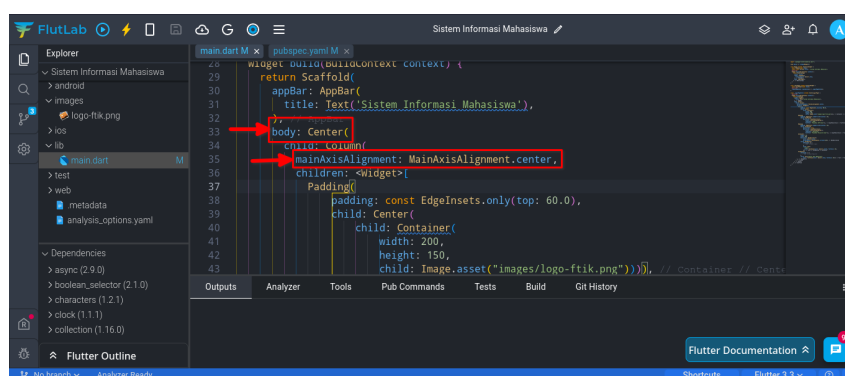


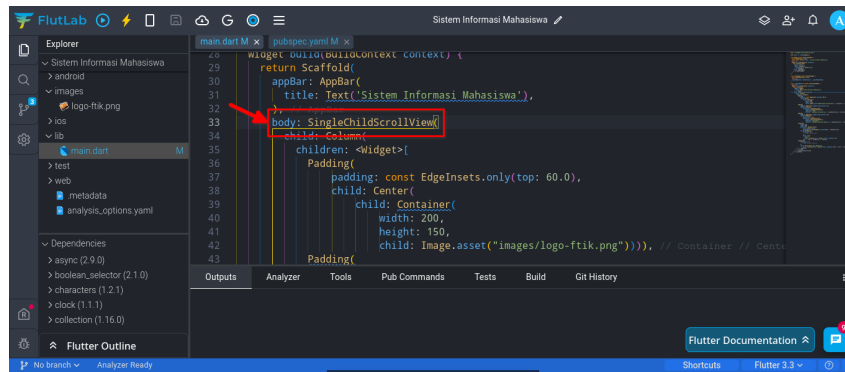
22. **Hot Reload** aplikasi untuk menampilkan gambar. Jika tidak muncul, pastikan dilakukan dengan benar.



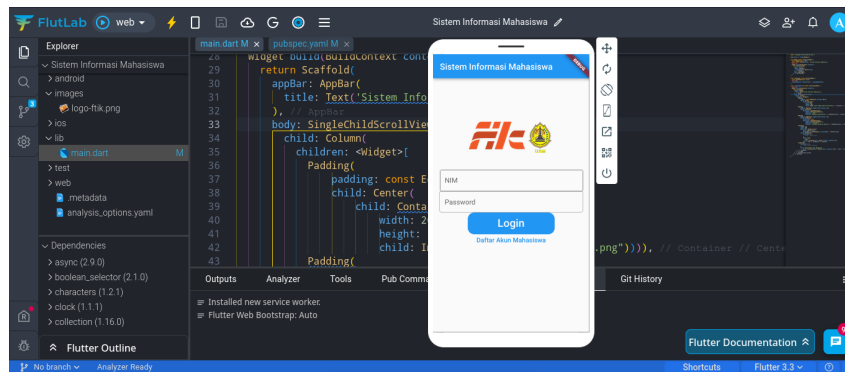
23. Langkah terakhir adalah mengubah layout agar memungkinkan untuk di **Scroll**. Perhatikan berikut:

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





24. Build Ulang Aplikasi untuk melihat hasil akhirnya



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

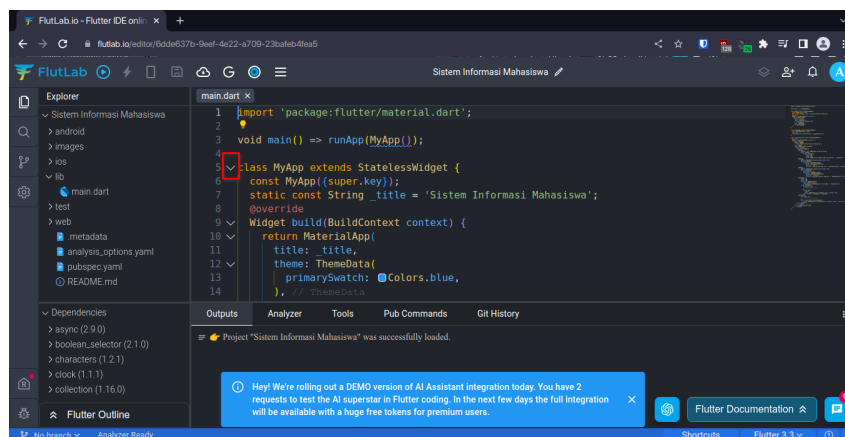
Bab 3

Praktikum 3

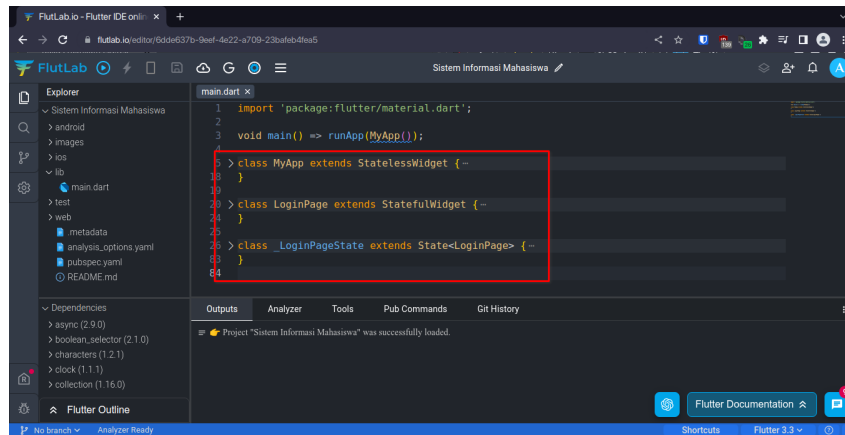
3.1 Halaman Registrasi dan Navigasi Page

3.2 Tutorial

1. Buka proyek 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



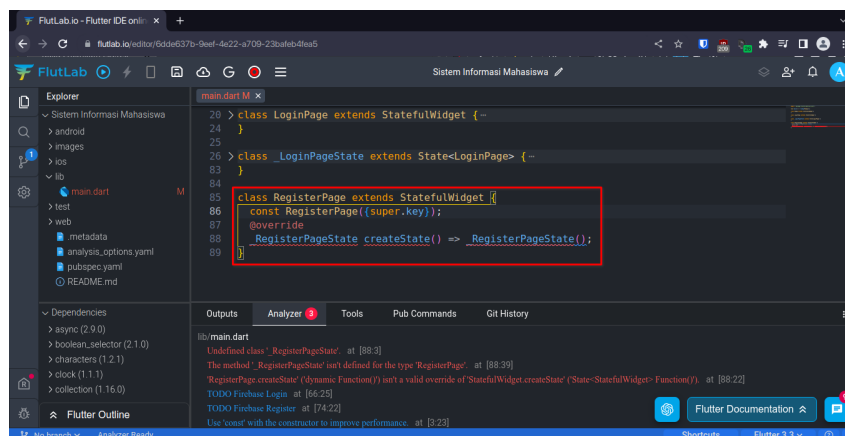
4. Berikutnya masukkan kode **class** untuk pembuatan tampilan **Pendaftaran** seperti berikut:

Potongan Kode

```

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

```



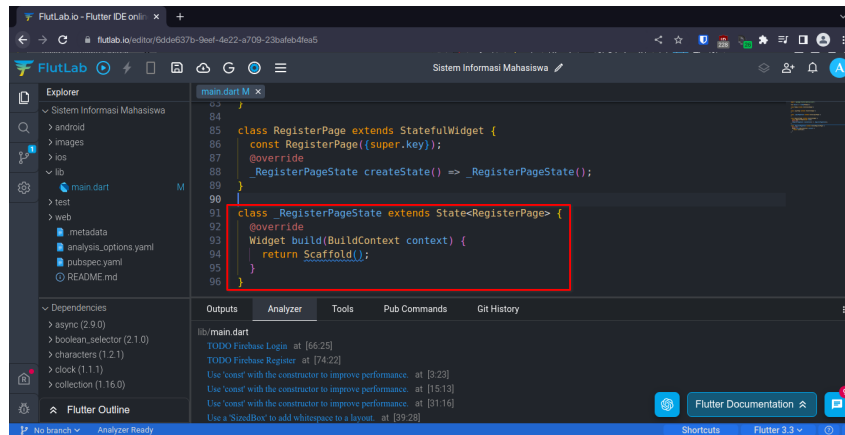
5. Abaikan **Error** saat ini. Untuk menghilangkan **Error** itu, masukkan kode berikut:

Potongan Kode

```

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

```



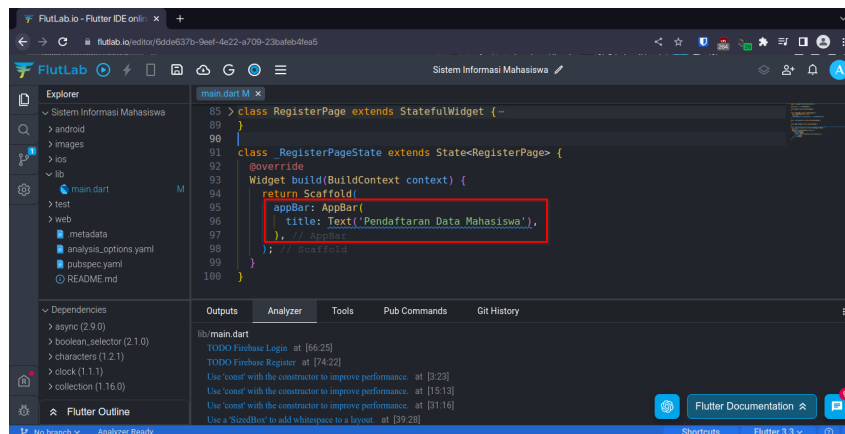
6. Kode **widget-widget** akan dimasukkan ke dalam **return Scaffold()**. Pertama masukkan **appBar** yang digunakan untuk menampilkan **Judul Halaman** seperti berikut:

Potongan Kode

```

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

```



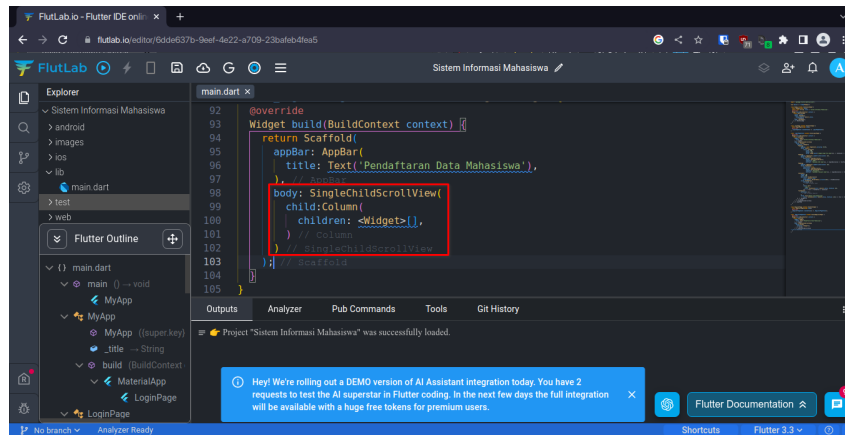
7. Berikutnya adalah menambahkan widget layout yang dapat di Scroll beserta wadah **Widget** nantinya. Perhatikan Kode dan Gambar berikut:

Potongan Kode

```

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

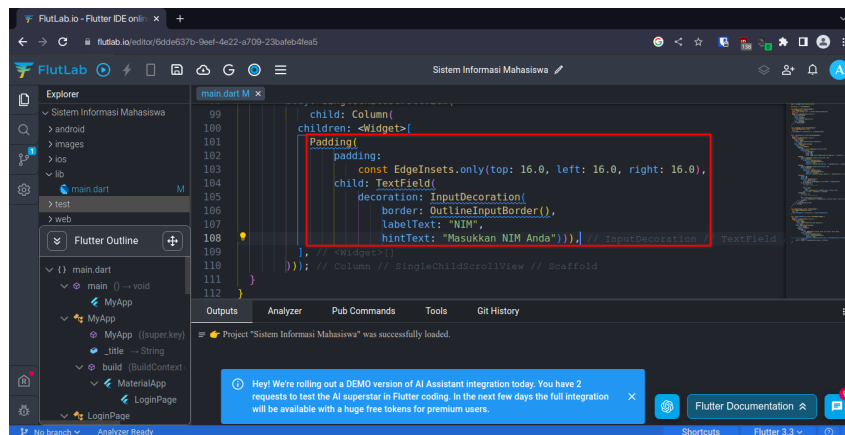
```



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

Potongan Kode

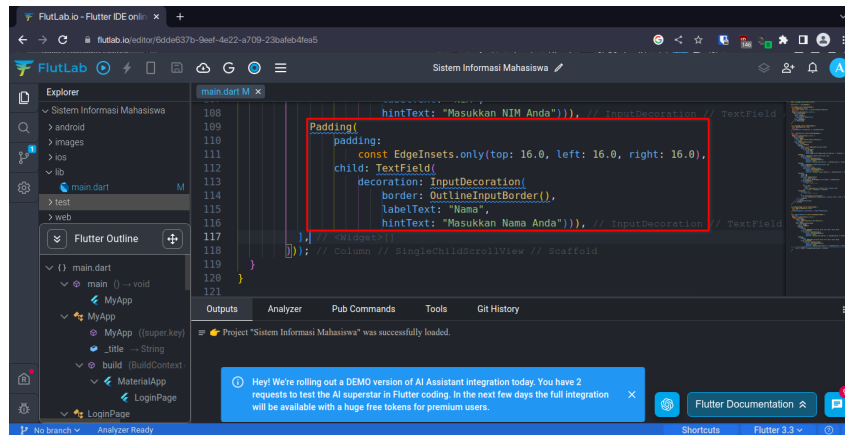
```
Padding(
  padding:
    const EdgeInsets.only(top: 16.0, left: 16.0, right: 16.0),
  child: TextField(
    decoration: InputDecoration(
      border: OutlineInputBorder(),
      labelText: "NIM",
      hintText: "Masukkan NIM Anda"))),
```



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

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"))),
```

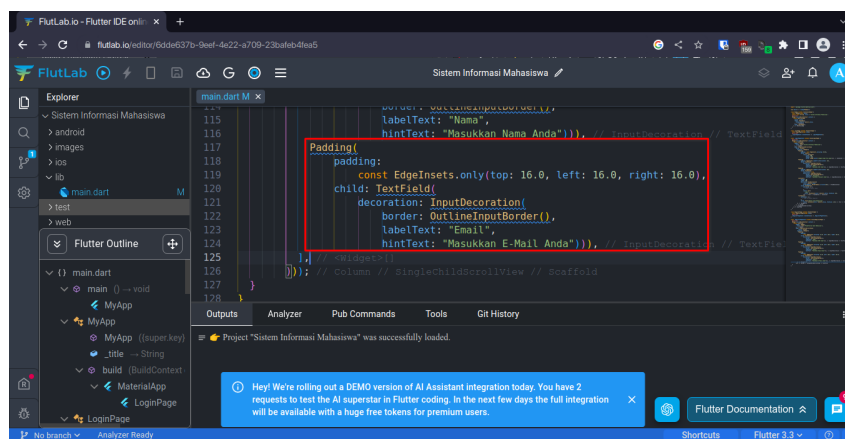


10. Selanjutnya adalah **InputText** untuk **E-Mail**. Perhatikan kode berikut:
Potongan Kode

```

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"))),

```

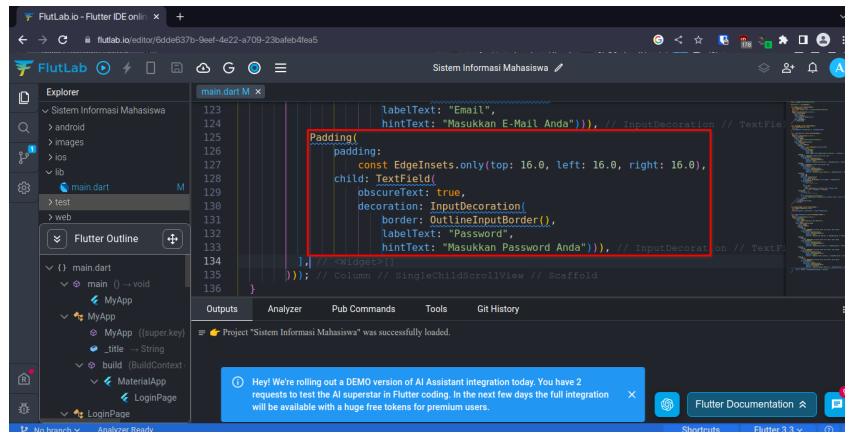


11. **InputText** terakhir adalah untuk **Password**. Masukkan kode berikut:
Potongan Kode

```

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"))),

```



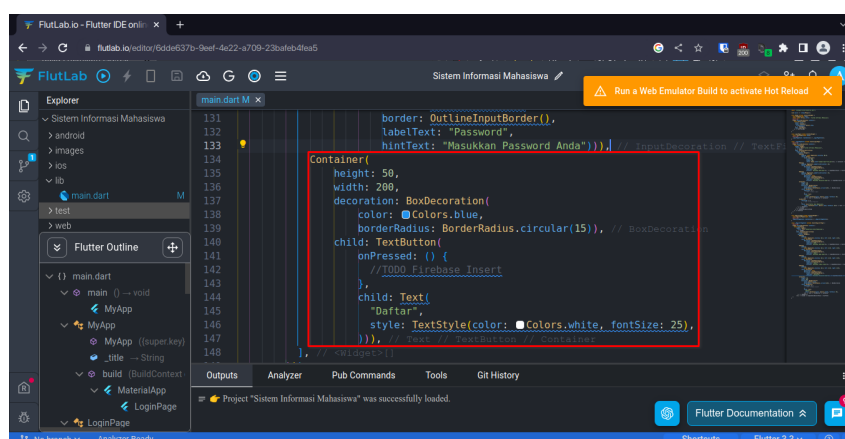
12. Widget berikutnya adalah **Tombol**. Masukkan kode berikut untuk menambahkan tombol **Daftar**

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),
    )),
  )

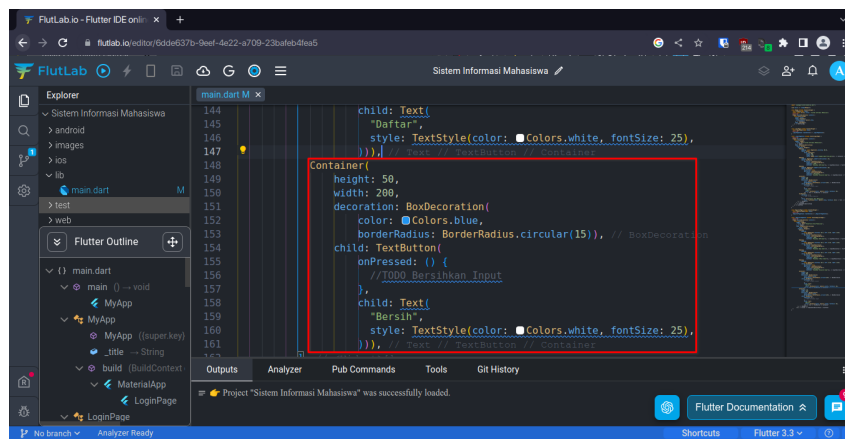
```



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

Potongan Kode

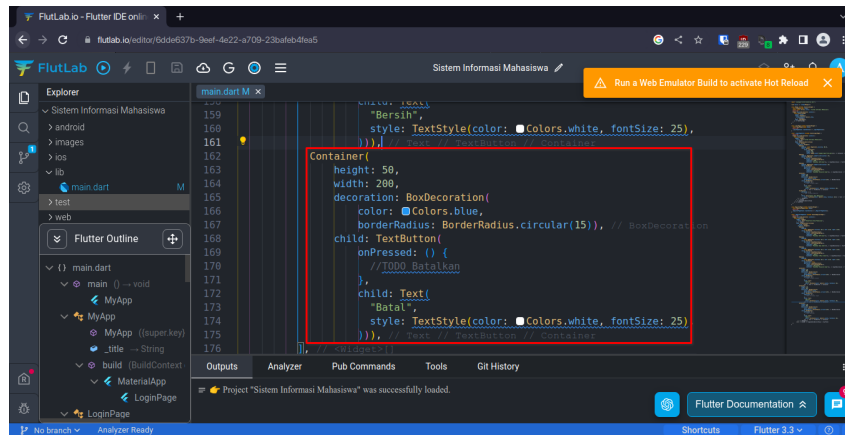
```
Container(  
  height: 50,  
  width: 200,  
  decoration: BoxDecoration(  
    color: Colors.blue,  
    borderRadius: BorderRadius.circular(15)),  
  child: TextButton(  
    onPressed: () {  
      //TODO Bersihkan Input  
    },  
    child: Text(  
      "Bersih",  
      style: TextStyle(color: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



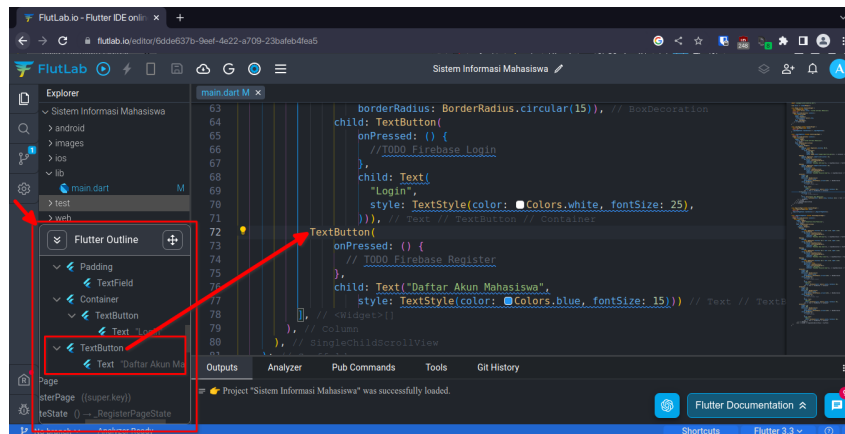
14. Tombol terakhir adalah **Batal** yang digunakan untuk kembali ke Menu Awal (**Login**). Masukkan kode berikut:

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),  
    )),  
  )),
```

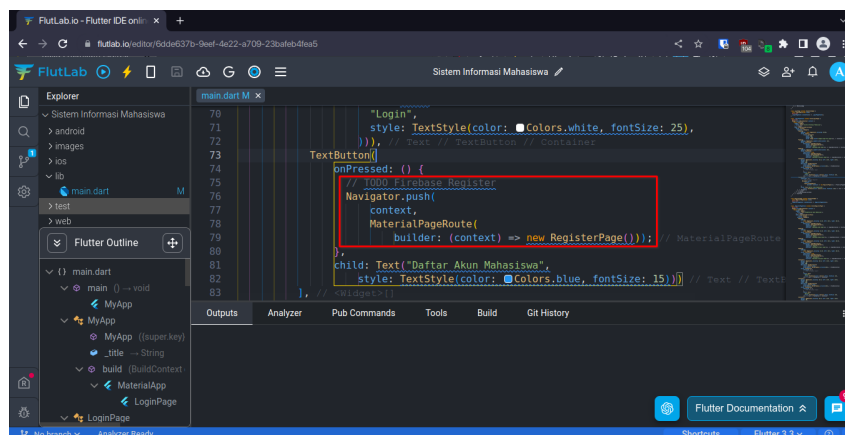


15. Berikutnya adalah memberi fitur **Navigasi Halaman** agar aplikasi dapat berpindah ke halaman lainnya. Kembali ke bagian `_LoginPage` dan temukan `TextButton` registrasinya. Gunakan **Flutter Outline** di sebelah **Kiri Bawah** untuk mempermudah pencarian, dan **Klik** target tujuan:

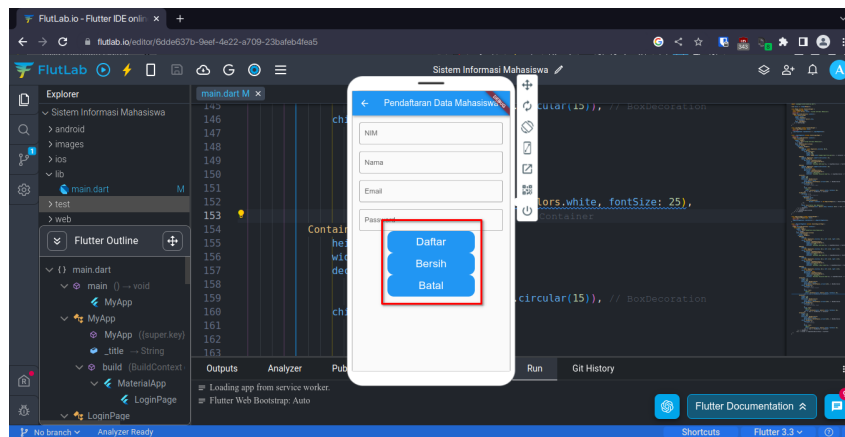


16. Di bagian dalam `onPressed:()`, masukkan kode berikut setelah **Komentar TODO:**
Potongan Kode

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



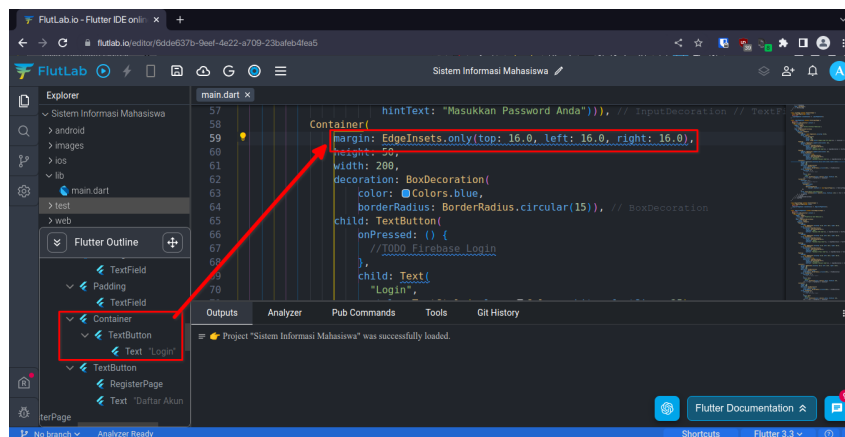
17. Build app yang sudah dibuat dan cek hasilnya



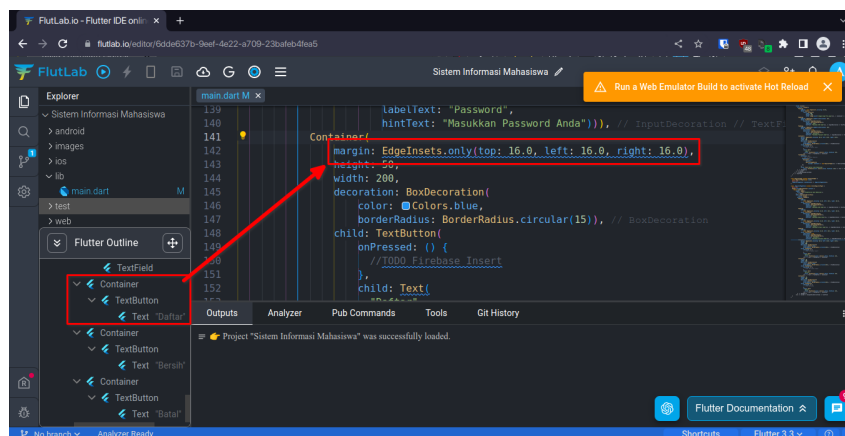
18. Karena Tombol terlalu dekat satu sama lainnya, maka tambahkan kode berikut ke semua Tombol yang ada:

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

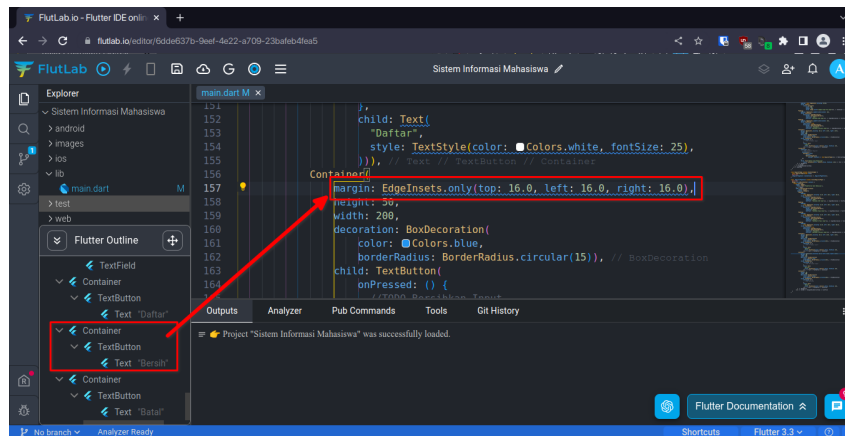
- Tombol Login



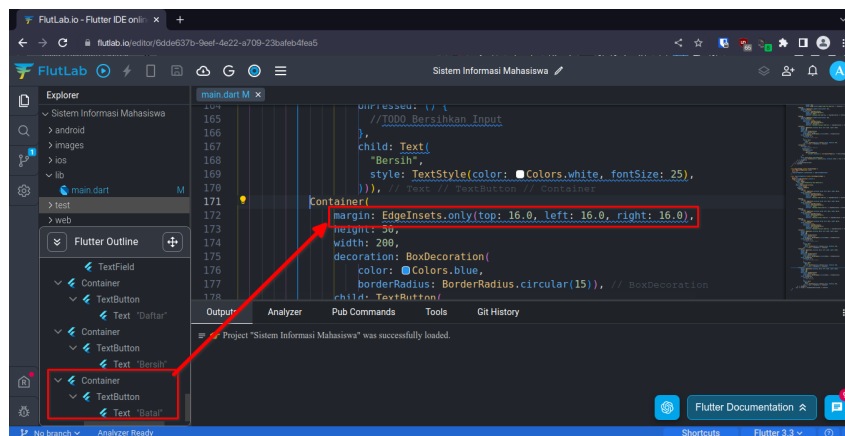
- Tombol Daftar



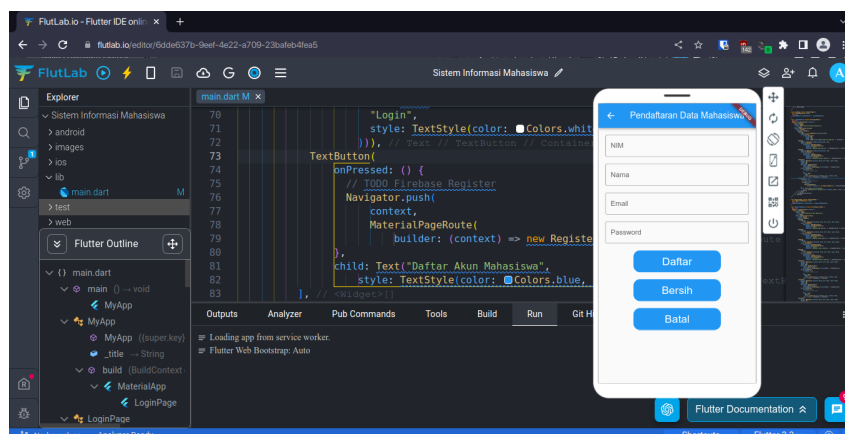
- Tombol Bersih



- Tombol Batal



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



Bab 4

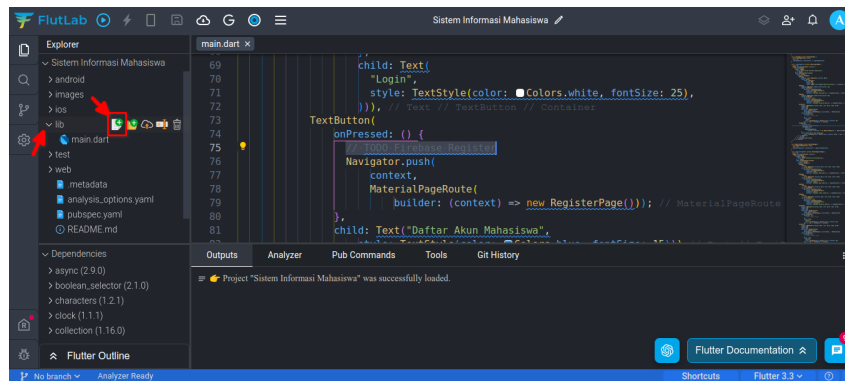
Praktikum 4

4.1 Integrasi Firebase

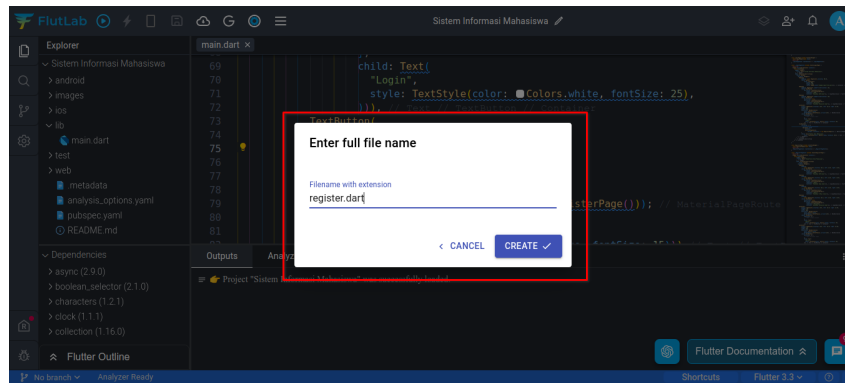
Di bagian ini mahasiswa diperkenalkan bagaimana melakukan pembersihan baris kode yang tidak diperlukan, integrasi ke Realtime Database, dan melakukan operasi register data. Mahasiswa diwajibkan untuk menyelesaikan Praktikum 3

4.2 Tutorial

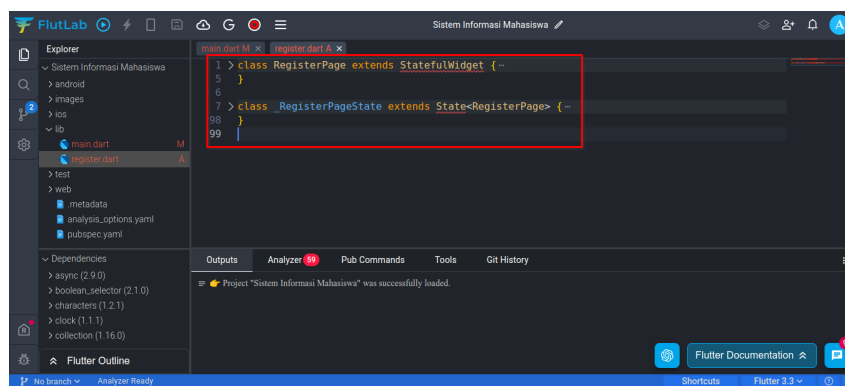
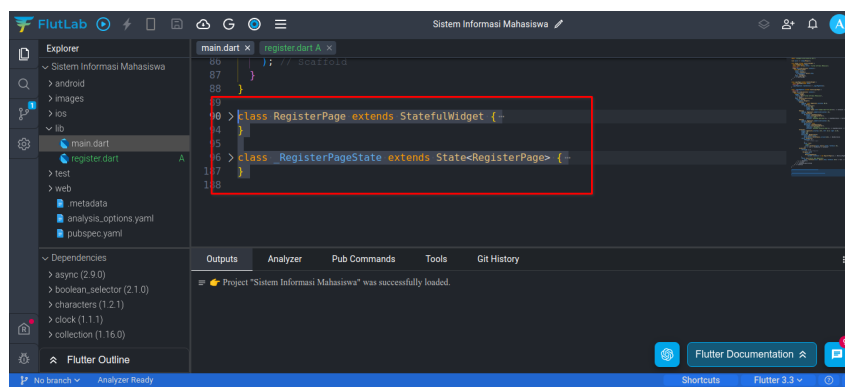
1. Buka projek sebelumnya, dan pastikan untuk membuka projek yang tepat
2. Berikutnya adalah memisahkan halaman **Register** ke file terpisah. Arahkan **Mouse** ke folder **lib**, dan klik **Create File**



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



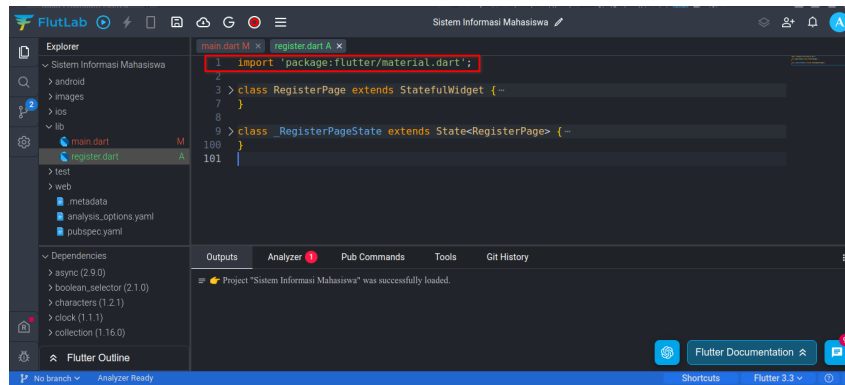
- File **register.dart** akan terbuka, lalu **PINDAHKAN** kode berikut dari **main.dart** ke **register.dart**. Lipat kode untuk mempermudah pemindahan



- 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

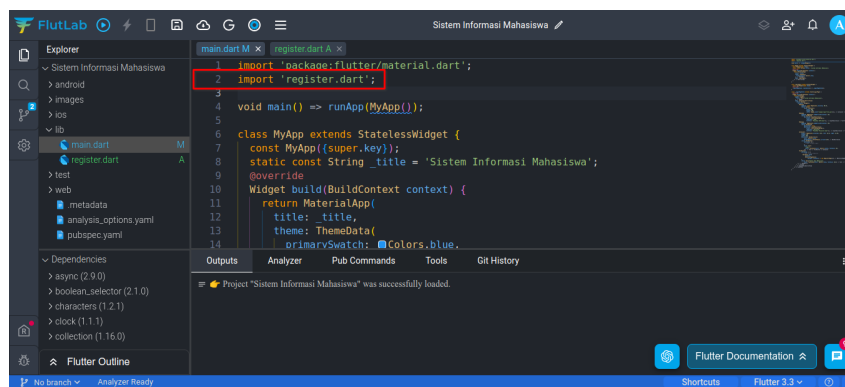
```
import 'package:flutter/material.dart';
```



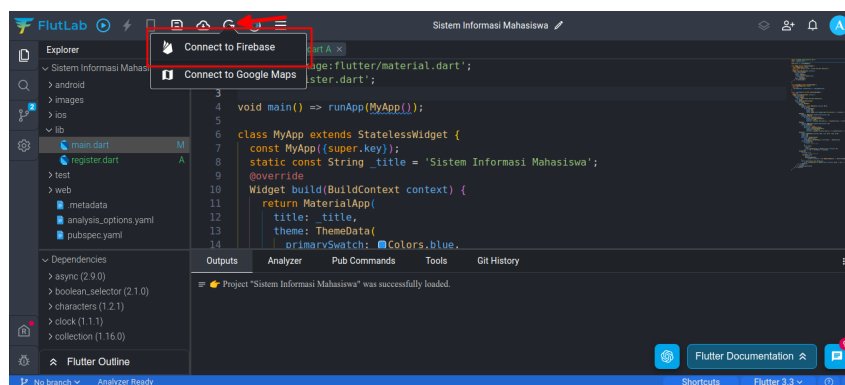
7. Lalu untuk memperbaiki kode **main.dart** cukup tambahkan kode berikut di **Baris 2**

Potongan Kode

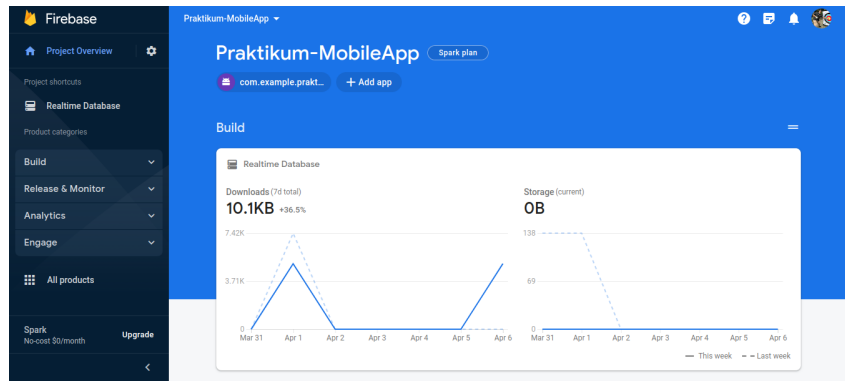
```
import 'register.dart';
```



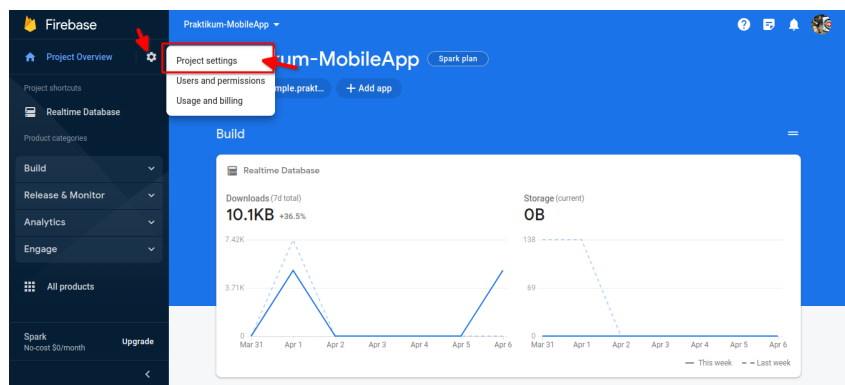
8. Semua **Error** sudah hilang, berikutnya melakukan integrasi ke **Firestore**. Klik Icon **G**, lalu pilih **Connect to Firestore**



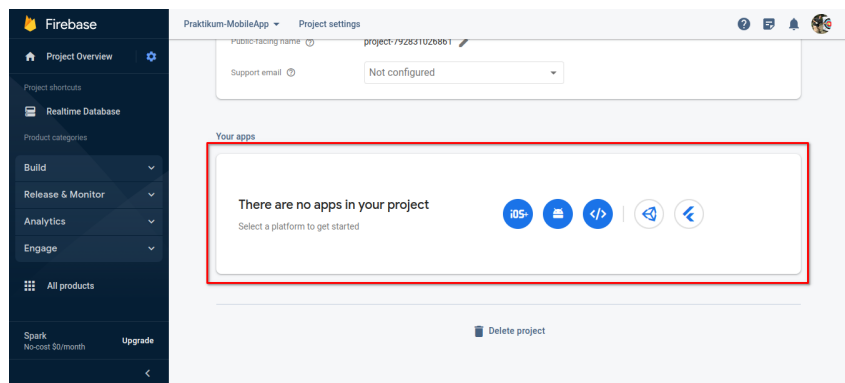
9. **Flutlab** akan menampilkan pilihan platform App yang menjadi target. Sebelum memilih buka website <https://firebase.google.com>. Login ke Google apabila belum, Klik **Go to console** di atas kanan. Pilih **Projek** yang sudah dibuat sebelumnya. Hingga tampil dasbor **Firestore**



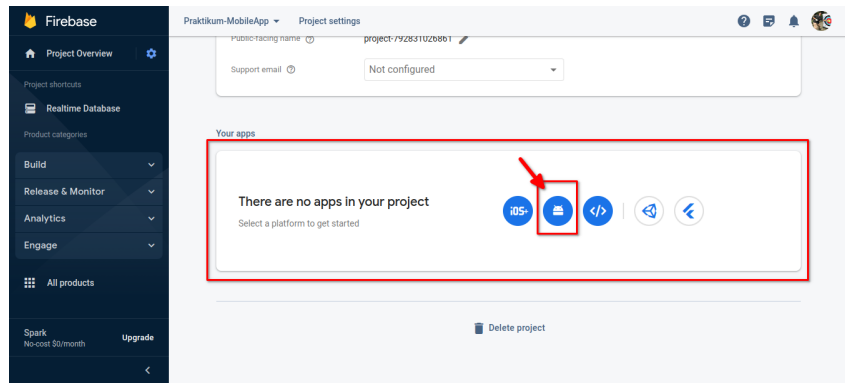
10. Untuk meng-konfigurasi, klik **Roda Gigi** di samping **Project Overview**, lalu pilih **Project Settings**



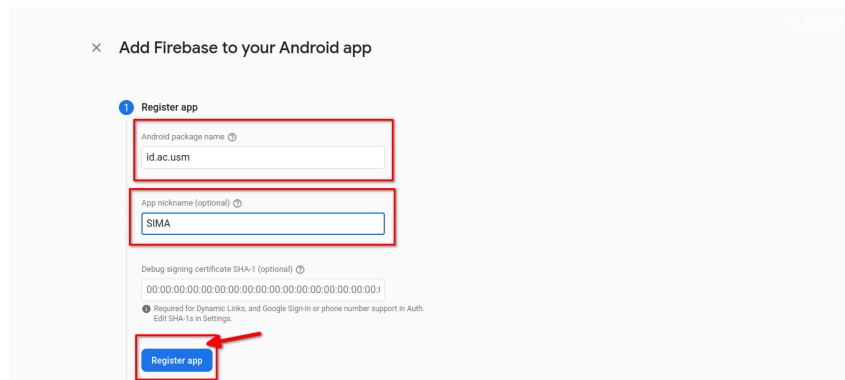
11. Scroll ke bawah dan cari panel **Your apps**



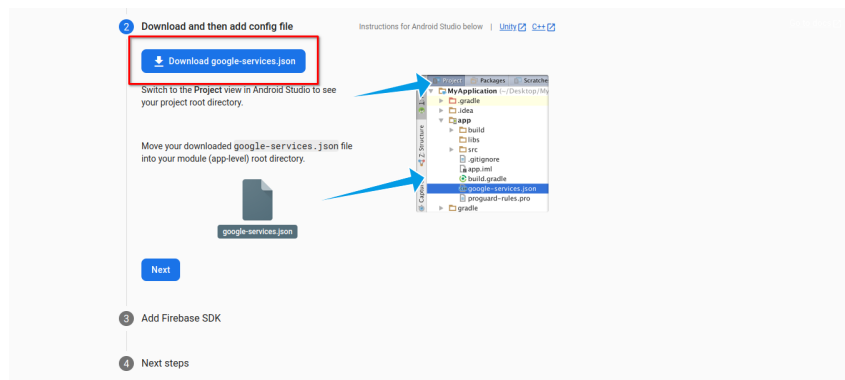
12. Pilih **Android**



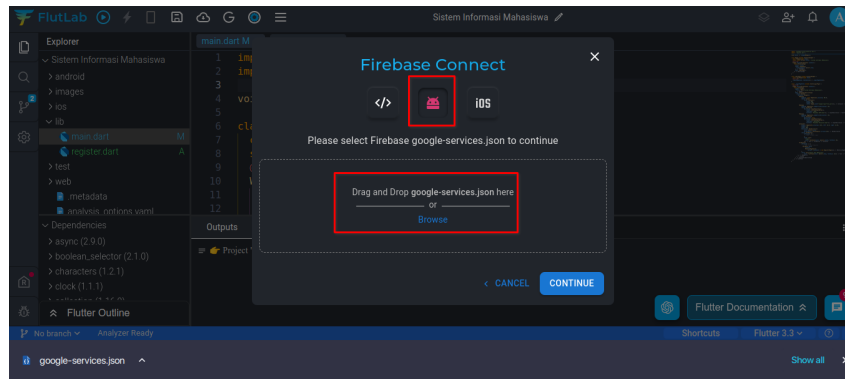
13. Lalu **Firebase** akan menampilkan **Wizard** untuk menambahkan App baru. Masukkan **id.ac.usm** sebagai **Android package name** dan **SIMA** sebagai **App nickname**. Klik **Register app**



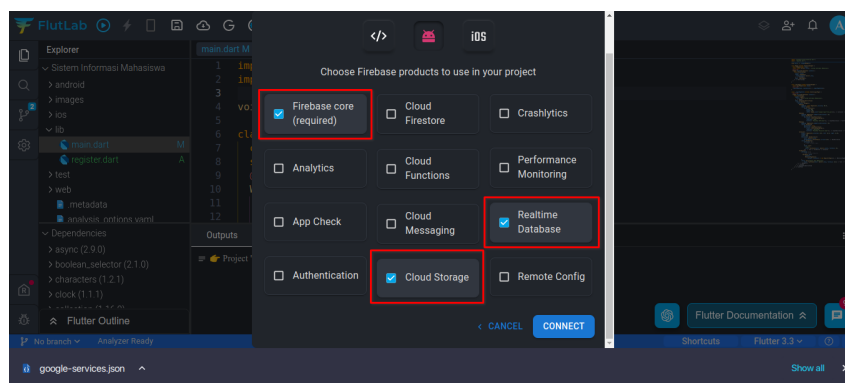
14. Langkah berikutnya adalah mengunduh file **google-services.json**. Cukup klik **Tombol** yang disediakan



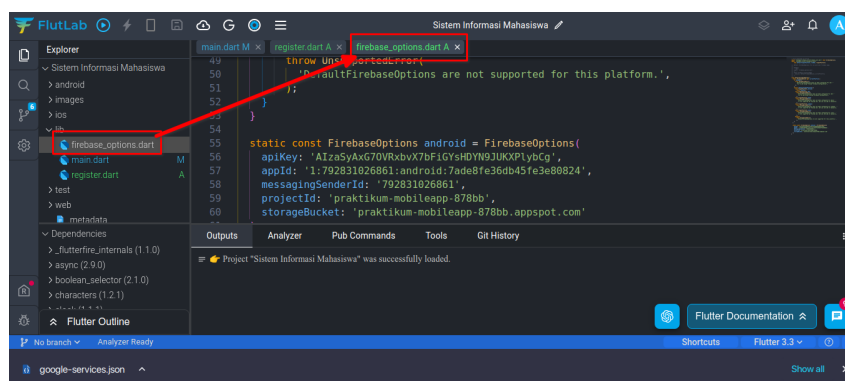
15. Tutup Wizard jika sudah selesai mengunduh
16. Kembali ke **FlutLab**, pilih **Android** lalu unggah file **google-services.json**. Klik **Continue** sesudah unggah



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 **Firestore** 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

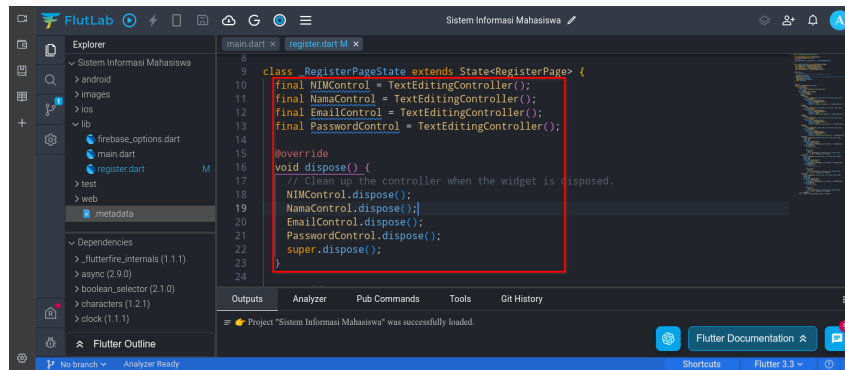
5.1 Operasi Insert Data Flutter

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();  
}
```

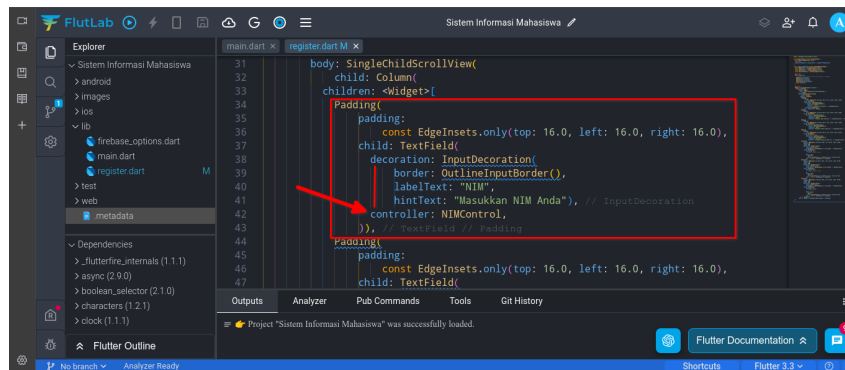


3. Tempelkan masing-masing kontroler ke **widgetnya**. Perhatikan kode dan Gambar secara teliti

- Widget NIM → NIMControl

Potongan Kode

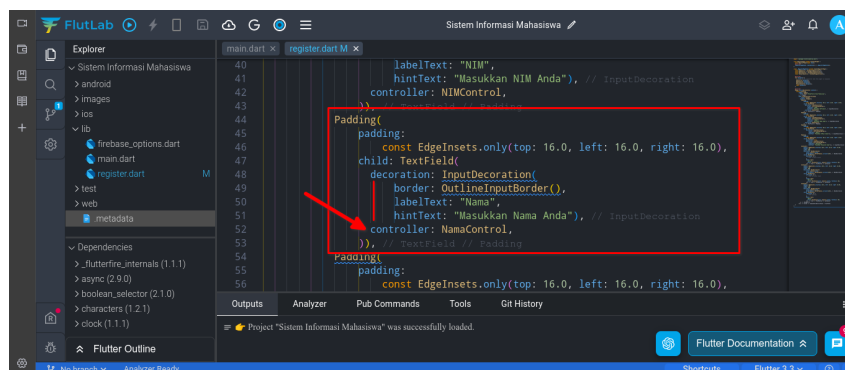
`controller: NIMControl,`



- Widget Nama → NamaControl

Potongan Kode

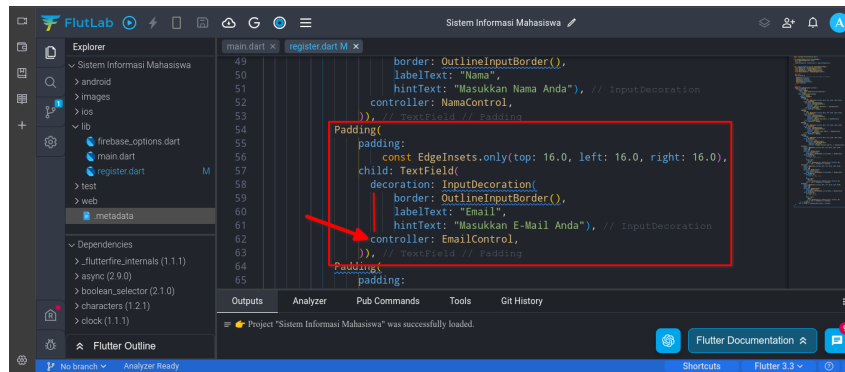
`controller: NamaControl,`



- Widget Email → EmailControl

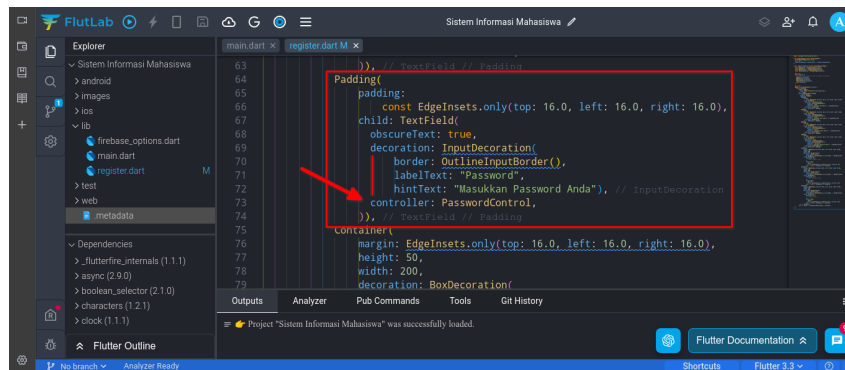
Potongan Kode

`controller: EmailControl,`



- Widget **Password** → PasswordControl
Potongan Kode

`controller: PasswordControl,`



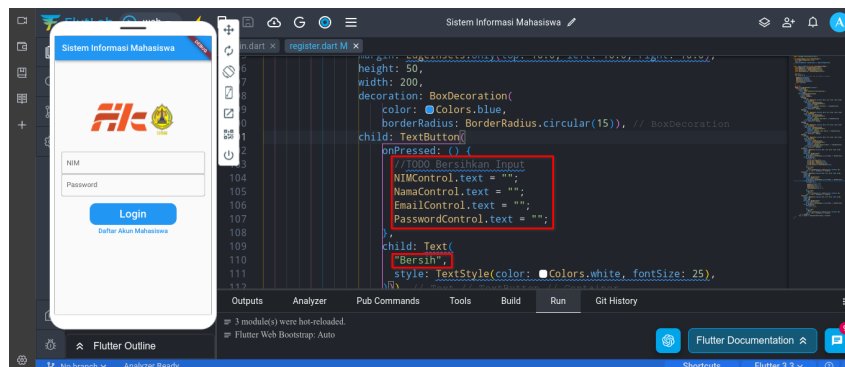
4. Setelah mengeset **controller** ke semua widget, kini aplikasi dapat mengambil maupun mengeset data. Scroll turun ke widget Button **Bersih** dan masukkan kode berikut:

Potongan Kode

```

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

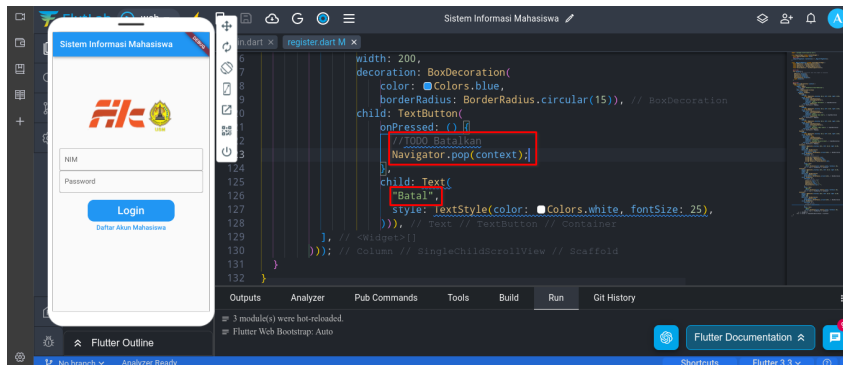
```



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

Potongan Kode

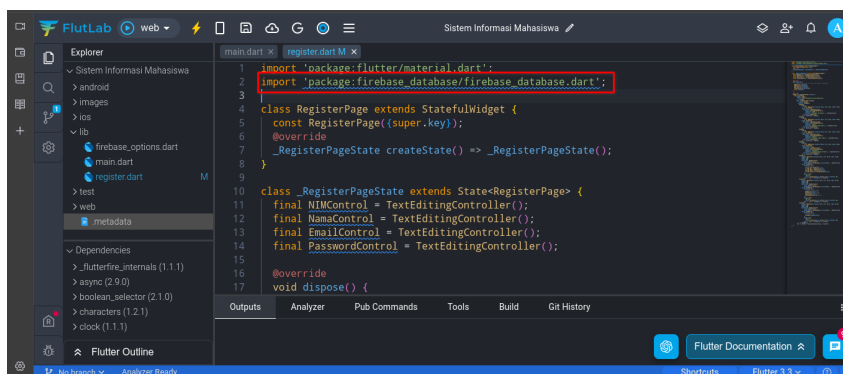
```
//TODO Batalan  
Navigator.pop(context);
```



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

Potongan Kode

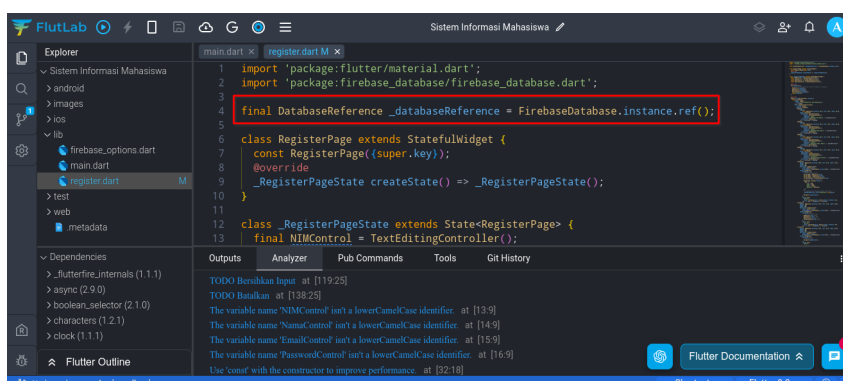
```
import 'package:firebase_database/firebase_database.dart';
```



7. Lalu tambahkan variabel global tepat di bawah kode import

Potongan Kode

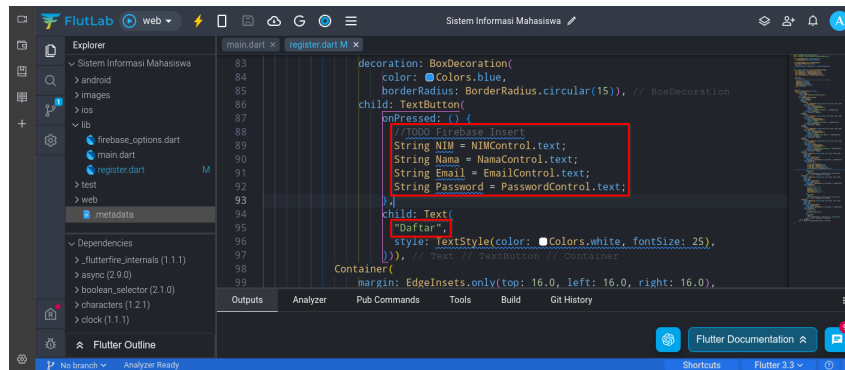
```
final DatabaseReference _databaseReference = FirebaseDatabase  
.instance.ref();
```



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

Potongan Kode

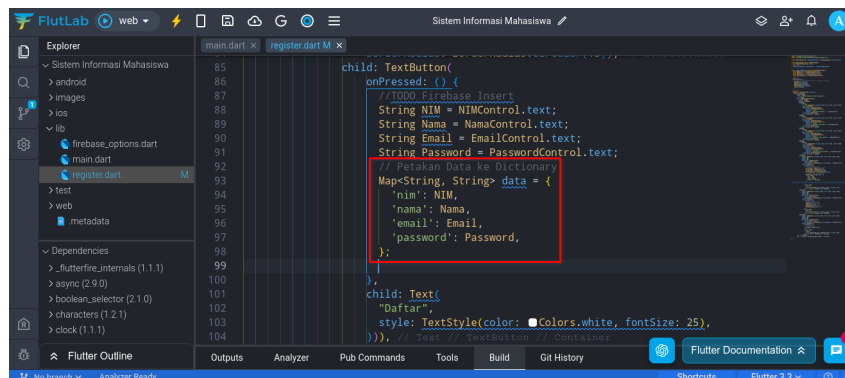
```
//TODO Firebase Insert
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
```



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

Potongan Kode

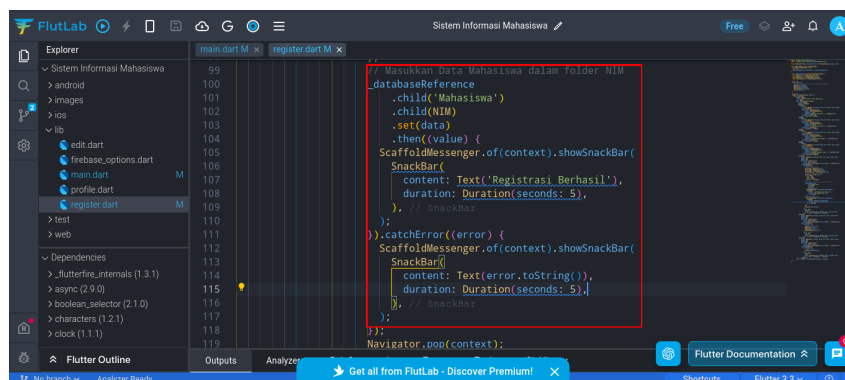
```
// Petakan Data ke Dictionary
Map<String, String> data = {
  'nim': NIM,
  'nama': Nama,
  'email': Email,
  'password': Password,
};
```



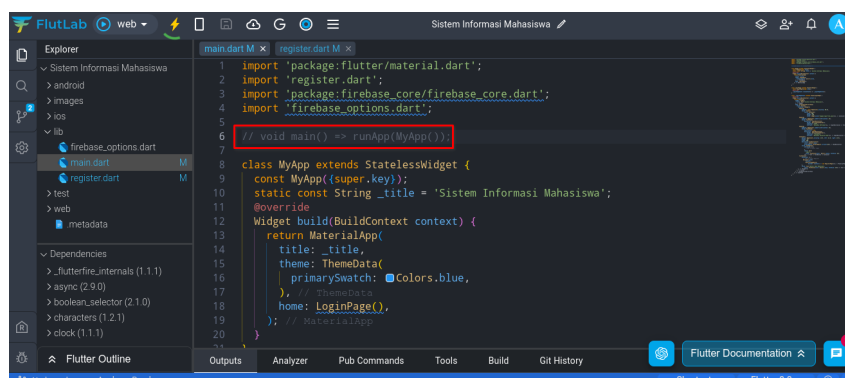
10. Kemudian tambahkan kode untuk melakukan kueri insert dan pengecekan keberhasilan

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);
```



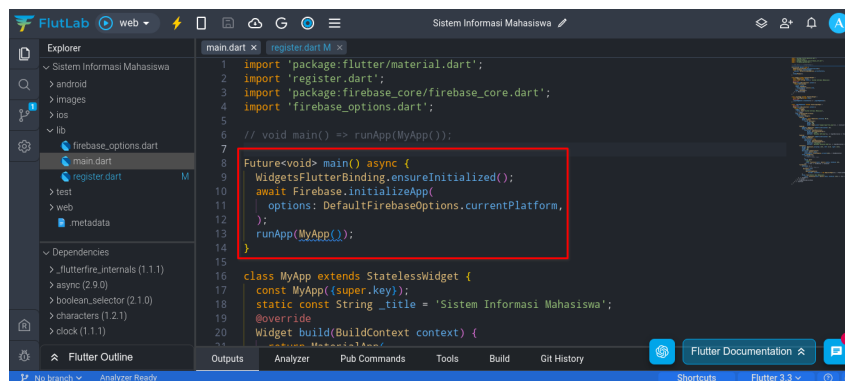
- Langkah berikutnya adalah mengubah fungsi **main** karena harus melakukan inisialisasi database. Buka **main.dart** dan beri tanda **//** di depan baris kode **void main() => runApp(MyApp());**



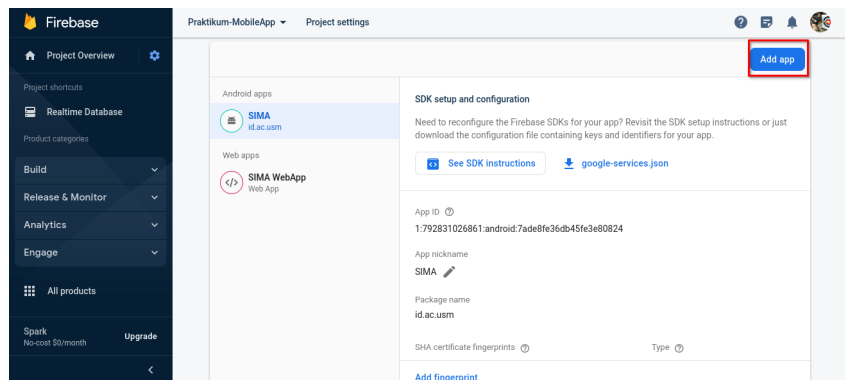
12. Lalu di bawahnya baris kode yang diberi komentar tadi, masukkan kode berikut:

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

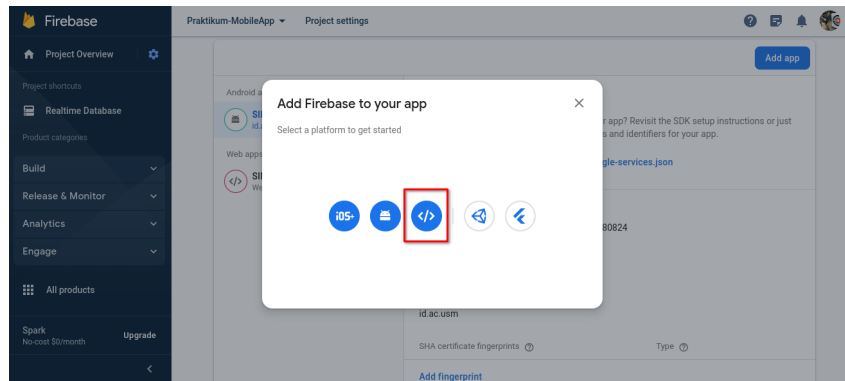
```
Future<void> main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp(  
    options: DefaultFirebaseOptions.currentPlatform,  
  );  
  runApp(MyApp());  
}
```



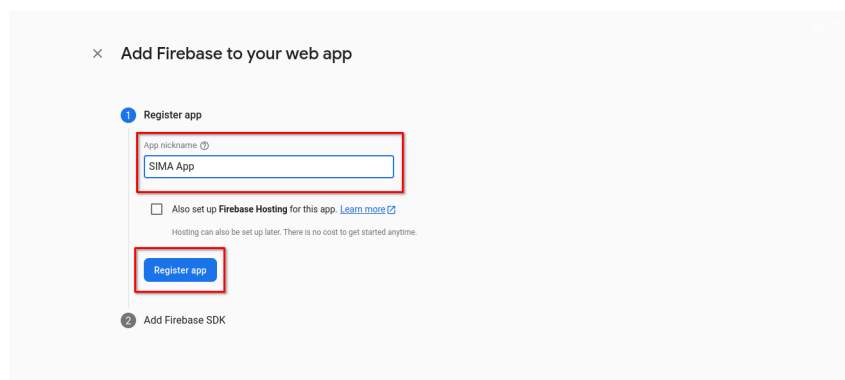
13. Aplikasi ini akan gagal ketika dijalankan, sehingga harus ditambahkan akses baru untuk aplikasi Web. Buka link <https://firebase.google.com>, buka **Project Settings (Gigi Roda)**, pilih **General**, Scroll Down ke **Your Apps**, klik **Add App**



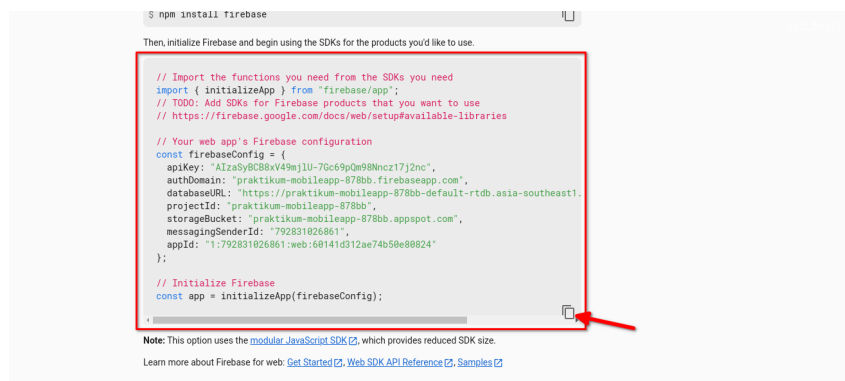
14. Pilih platform **Web**



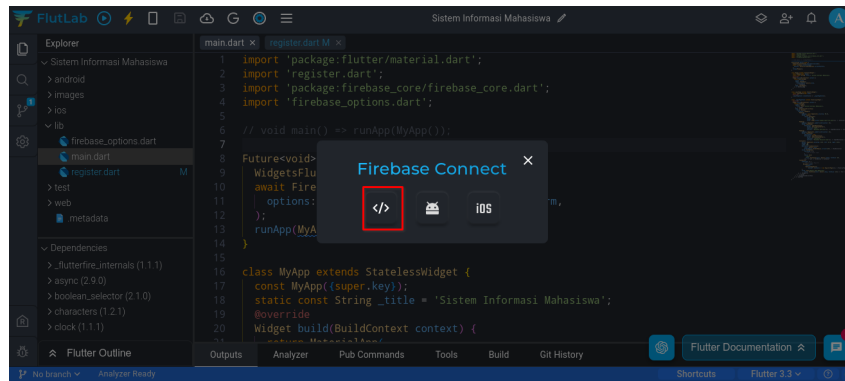
15. Beri nama **SIMA App**, lalu klik **Register App**



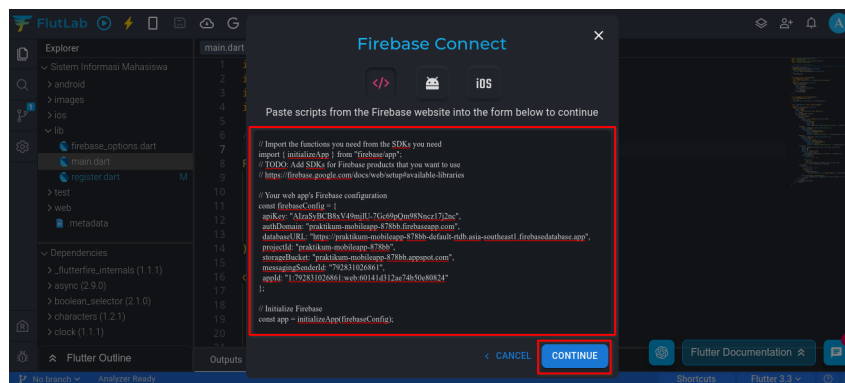
16. Lalu kopi kode yang diperlihatkan Firebase



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

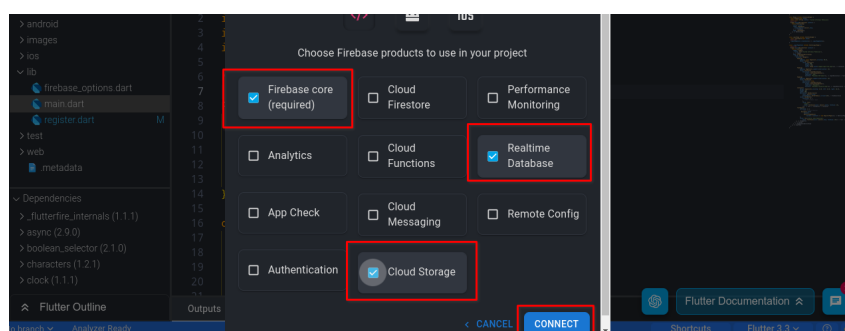


18. Tempel kode dari Firebase, dan klik **Continue**

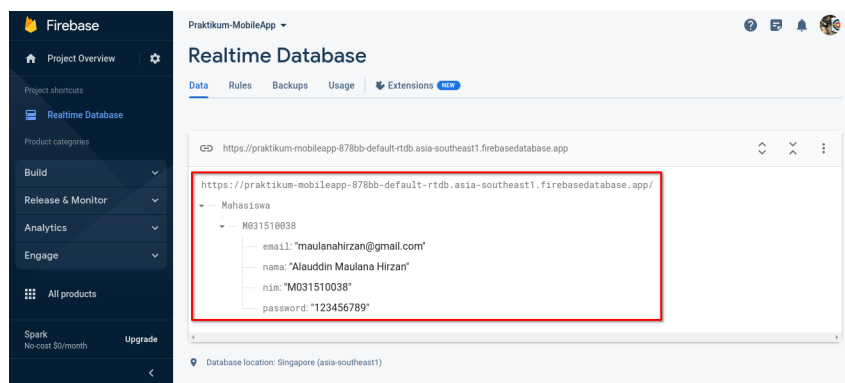
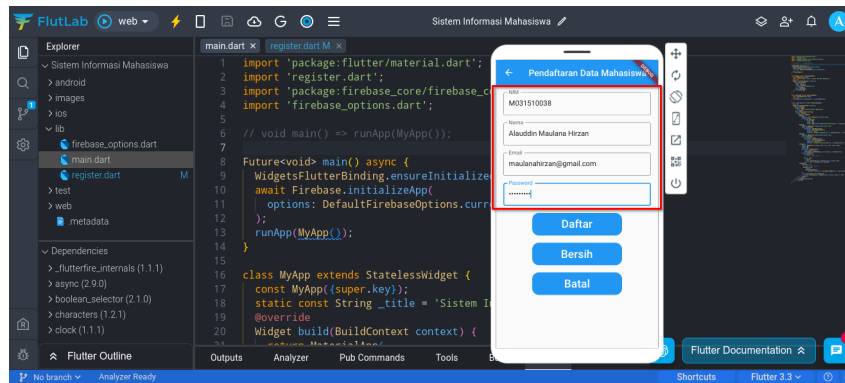


19. Di menu selanjutnya pilih

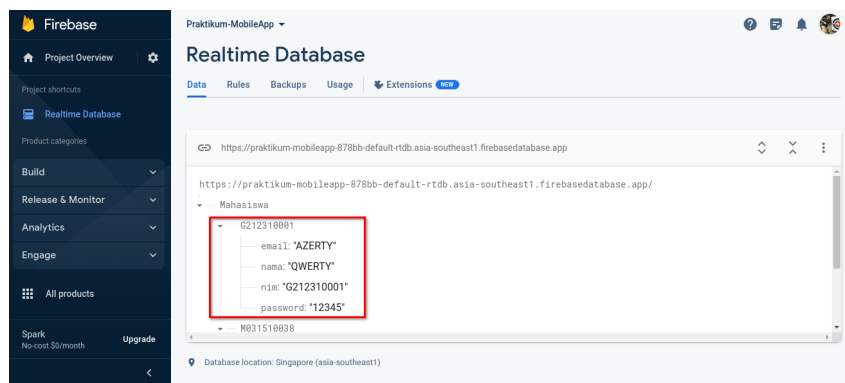
- Firebase Core
- Realtime Database
- Cloud Storage



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

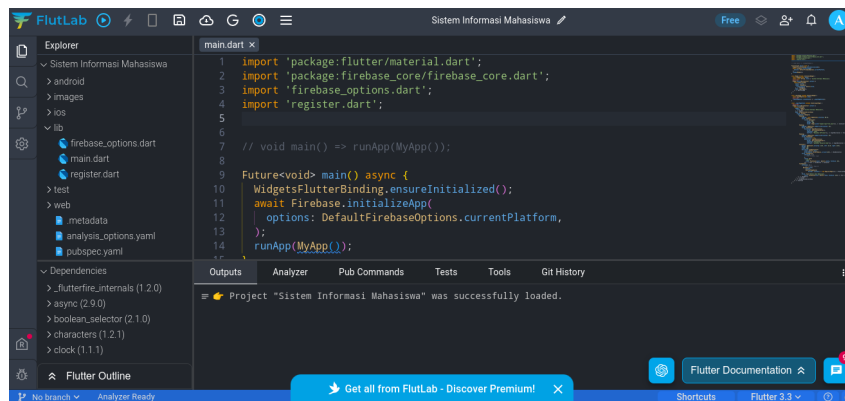
Praktikum 6

6.1 Operasi Read Data Flutter

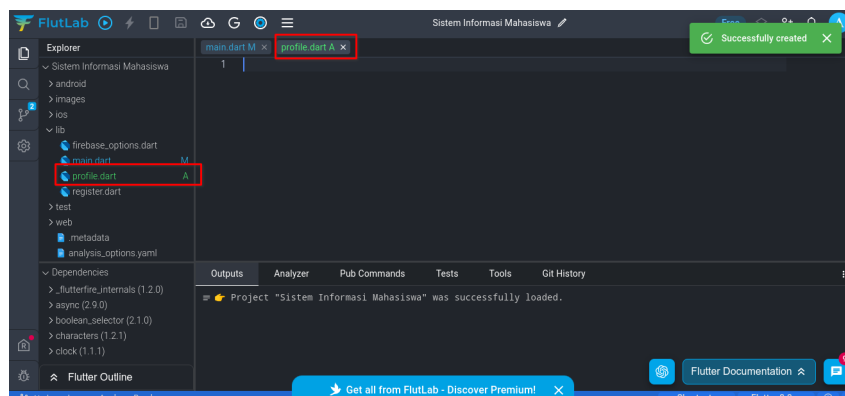
Di bagian ini mahasiswa diajarkan bagaimana membuat tampilan profile untuk pengguna, sekaligus proses Login yang dilakukan dengan menggunakan Flutter. Mahasiswa diwajibkan menyelesaikan Praktikum 5 sebelum memulai praktikum ini.

6.2 Tutorial

1. Praktikum ini dimulai dengan membuka praktikum sebelumnya.



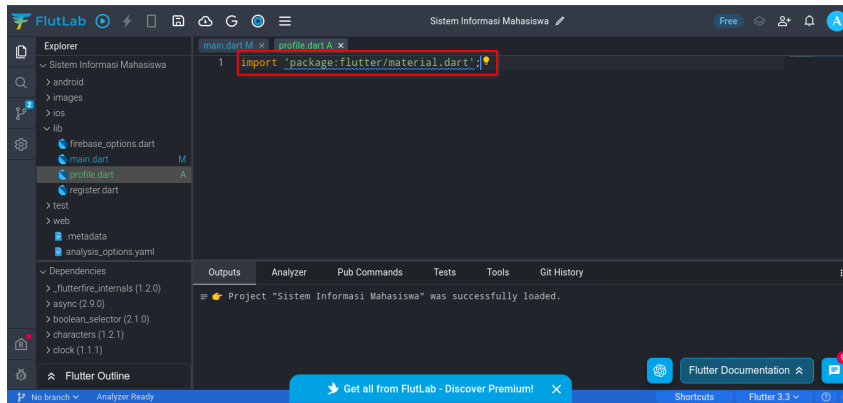
2. Buatlah satu file baru dengan nama **profile.dart**



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

Potongan Kode

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

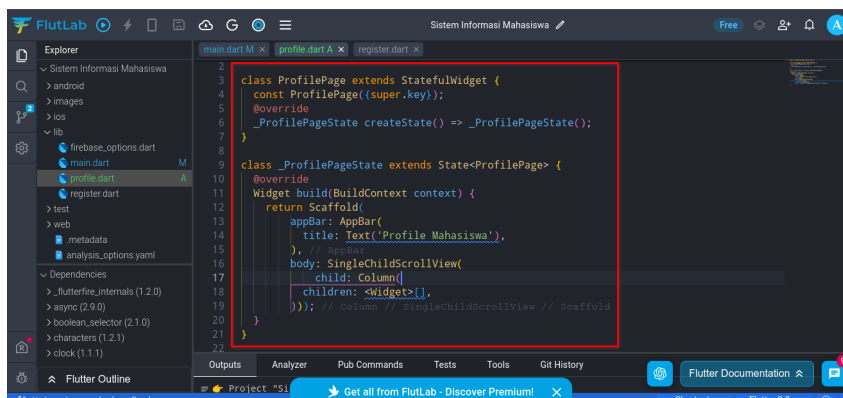


4. Kemudian buatlah tampilan Profile dengan kode sebagai berikut. Tambahkan setelah kode **import**

Potongan Kode

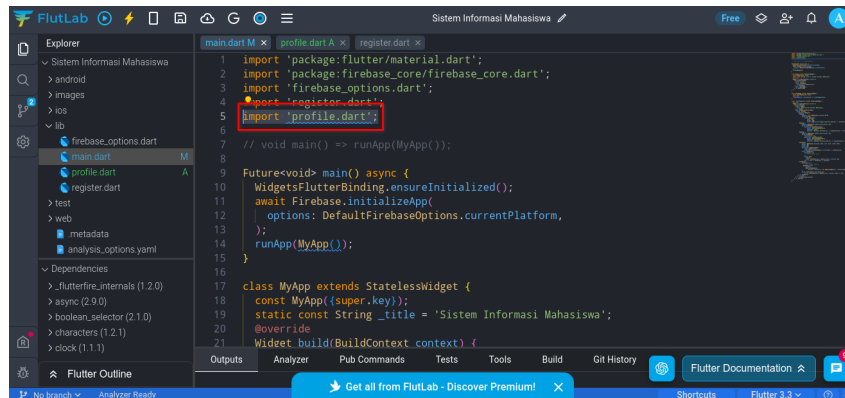
```
class ProfilePage extends StatefulWidget {
  const ProfilePage({super.key});
  @override
  _ProfilePageState createState() => _ProfilePageState();
}

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



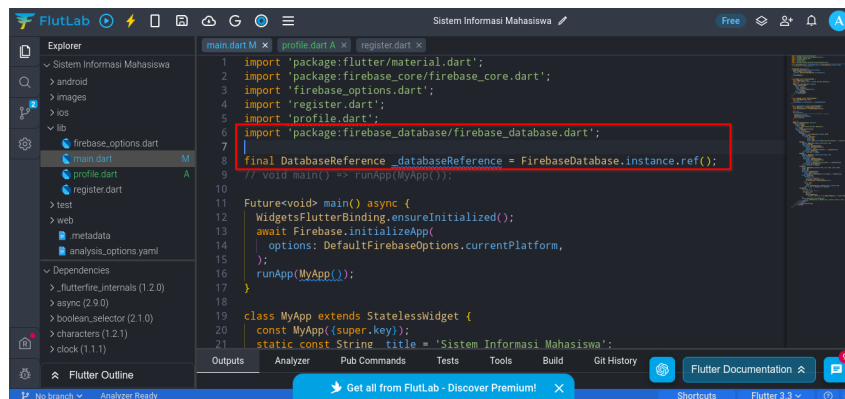
5. Sesudah itu, buka file **main.dart** lalu **Import** file **profile.dart** seperti berikut:

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



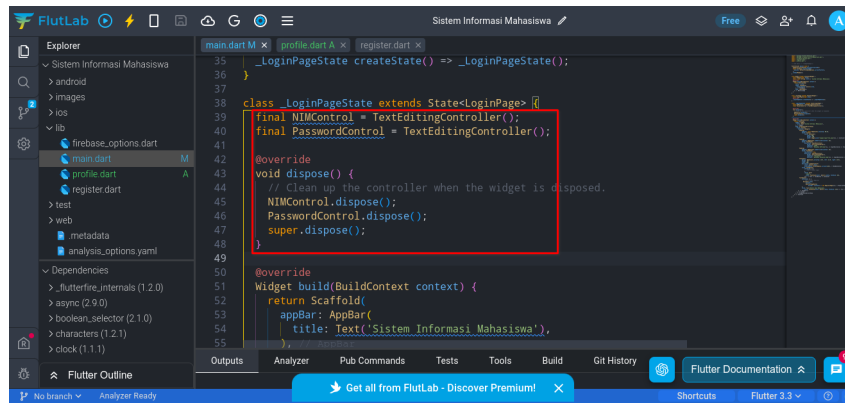
6. Agar bisa terhubung ke database, tambahkan kode berikut tepat di bawah **import** sebelumnya

```
import 'package:firebase_database/firebase_database.dart';  
  
final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();
```



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

```
final NIMControl = TextEditingController();  
final PasswordControl = TextEditingController();  
  
@override  
void dispose() {  
    NIMControl.dispose();  
    PasswordControl.dispose();  
    super.dispose();  
}
```

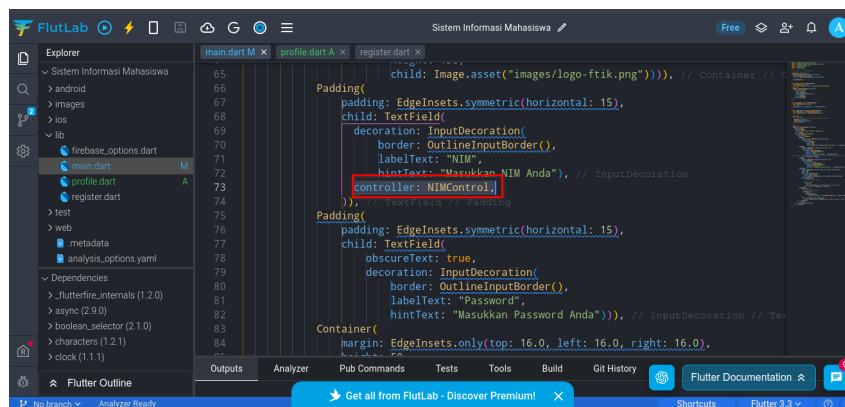


8. Kemudian turun kembali untuk menempelkan kontrol teks ke masing-masing widget. Perhatikan Kode dan Gambar berikut

(a) field NIM

Potongan Kode

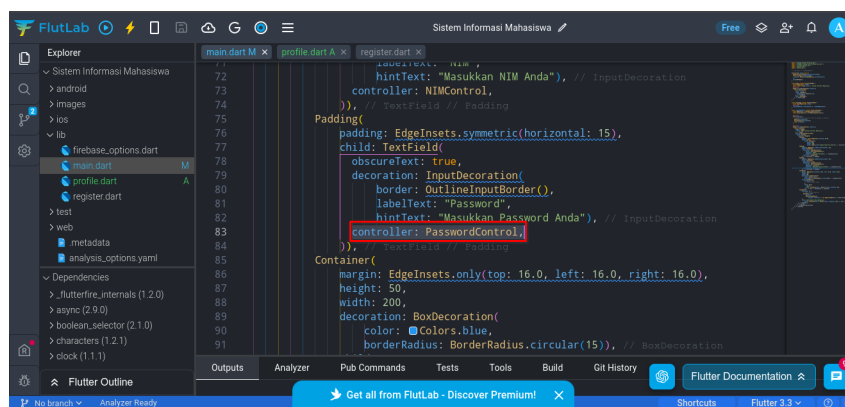
```
controller: NIMControl,
```



(b) field Password

Potongan Kode

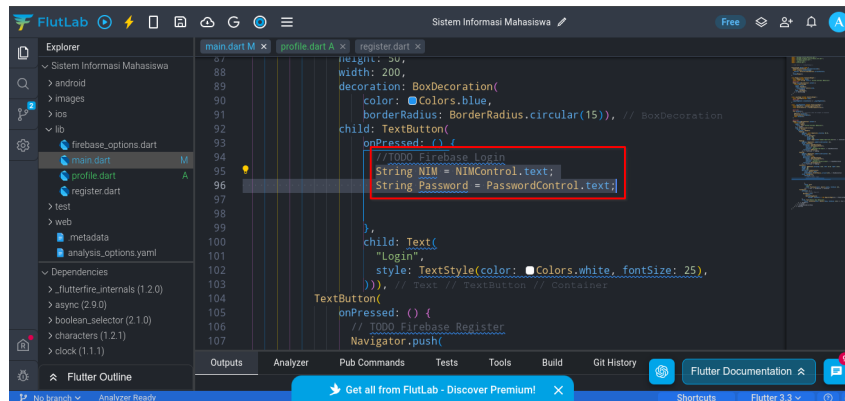
```
controller: PasswordControl,
```



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

Potongan Kode

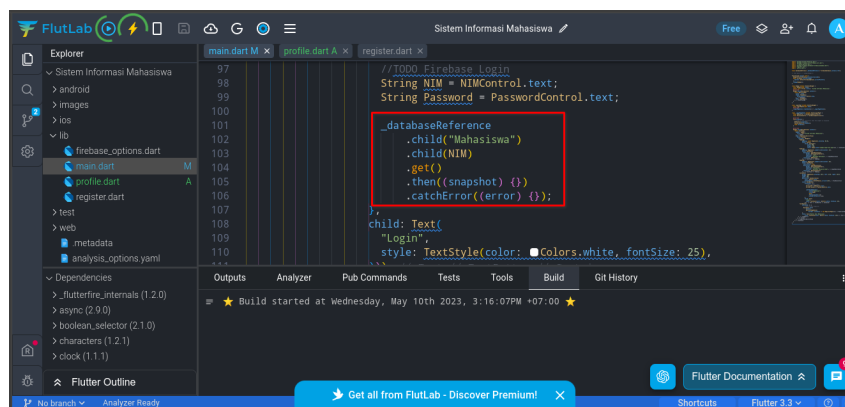
```
String NIM = NIMControl.text;  
String Password = PasswordControl.text;
```



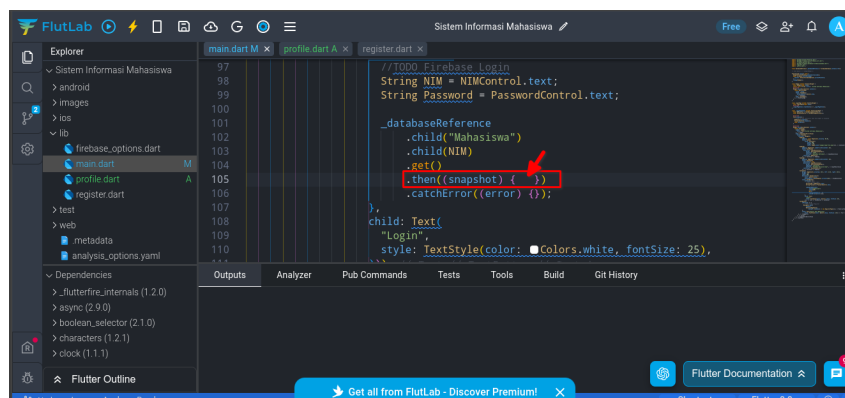
10. Tambahkan kode berikut untuk melakukan pengecekan login

Potongan Kode

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

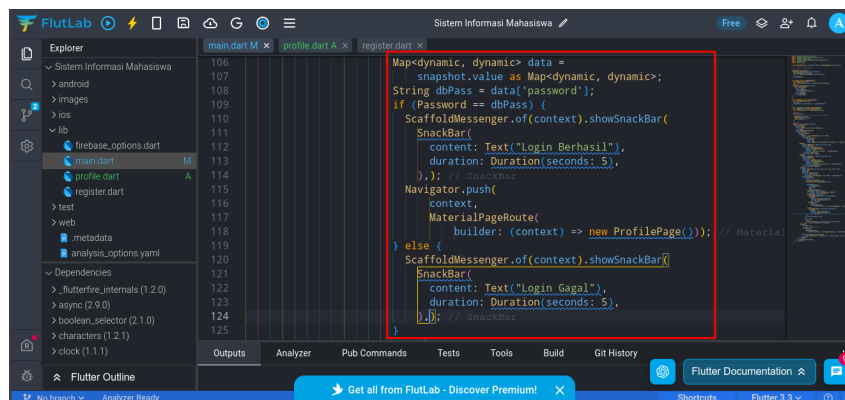


11. Setelah itu di dalam tanda kurung kurawal dari **Snapshot**. Masukkan kode berikut

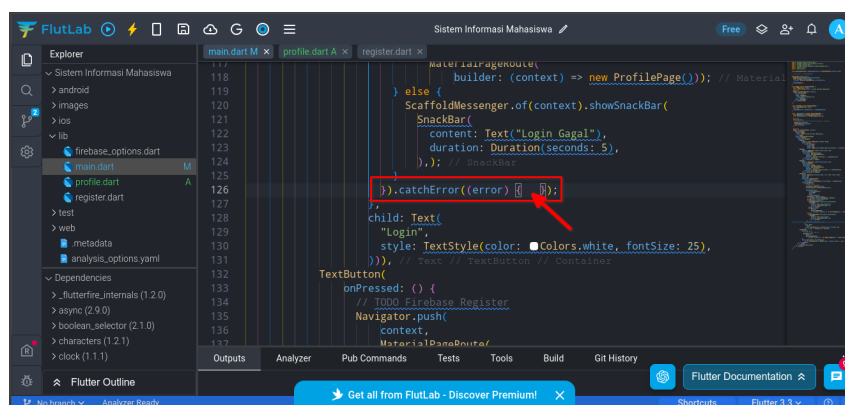


Potongan Kode

```
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),
    ),);
}
```

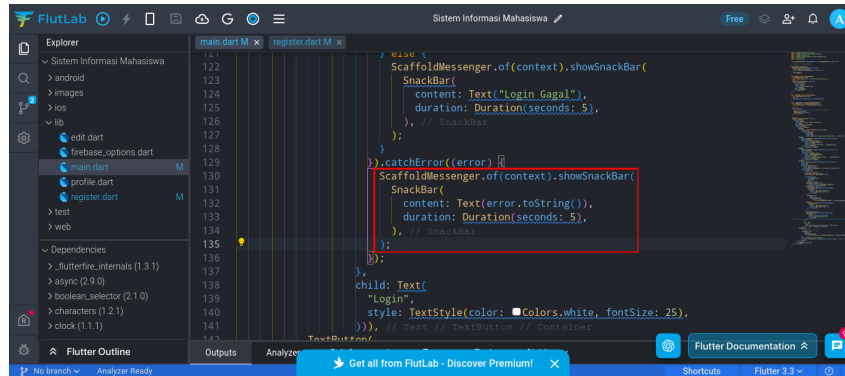


12. Kemudian di dalam kurung kurawal dari **catchError**, masukkan kode berikut

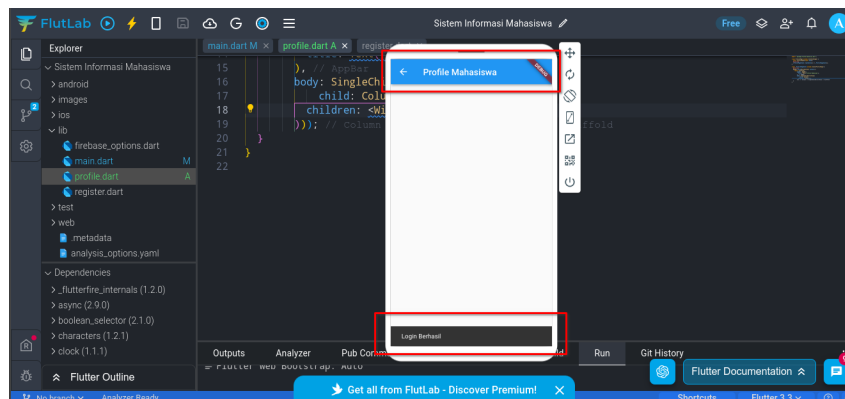


Potongan Kode

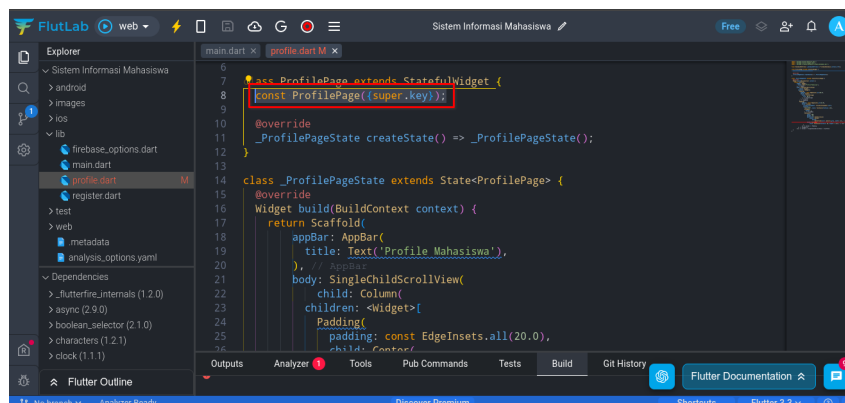
```
ScaffoldMessenger.of(context).showSnackBar(  
  SnackBar(  
    content: Text(error.toString()),  
    duration: Duration(seconds: 5),  
  ),  
);
```



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



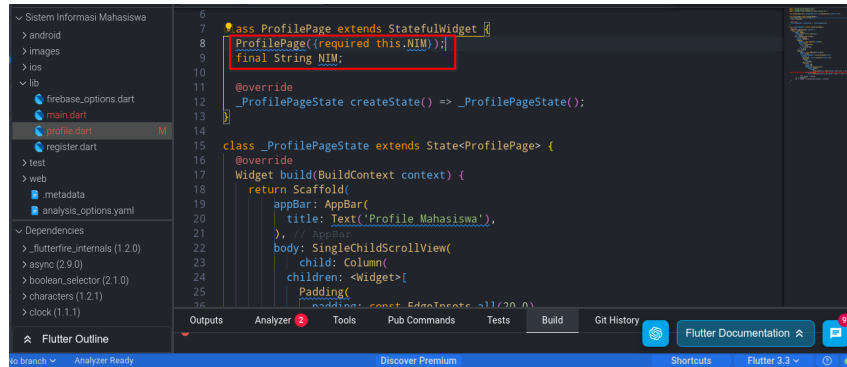
14. Berikutnya adalah mengisikan halaman Profile. Pertama hapus baris kode **const ProfilePage(super.key);**



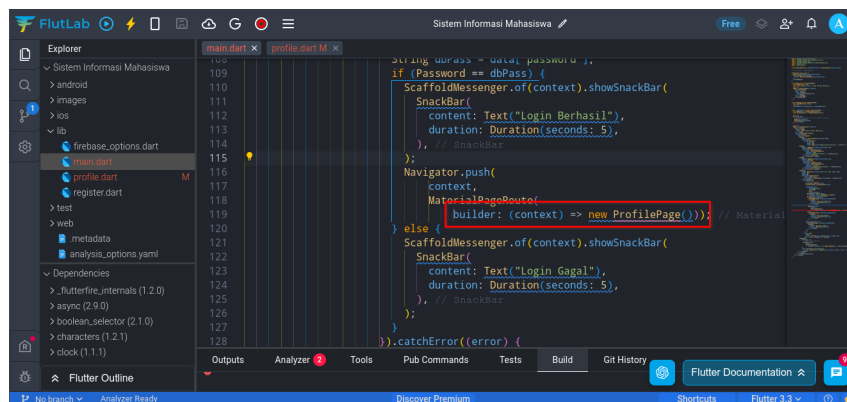
15. Agar bisa menerima input data **NIM** dari **Login** ke **Profile**, masukkan kode berikut di class **ProfilePage**

Potongan Kode

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

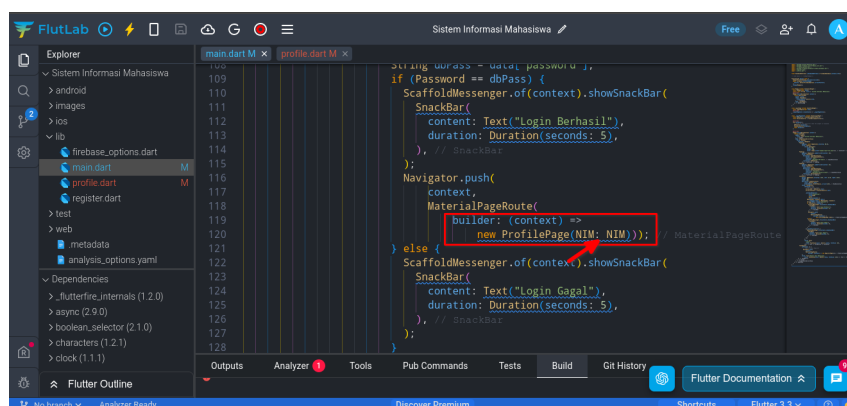


16. Kembali ke **main.dart**, tambahkan baris kode **builder: (context) => new ProfilePage());** dengan kode berikut:



Potongan Kode

```
NIM:NIM
```



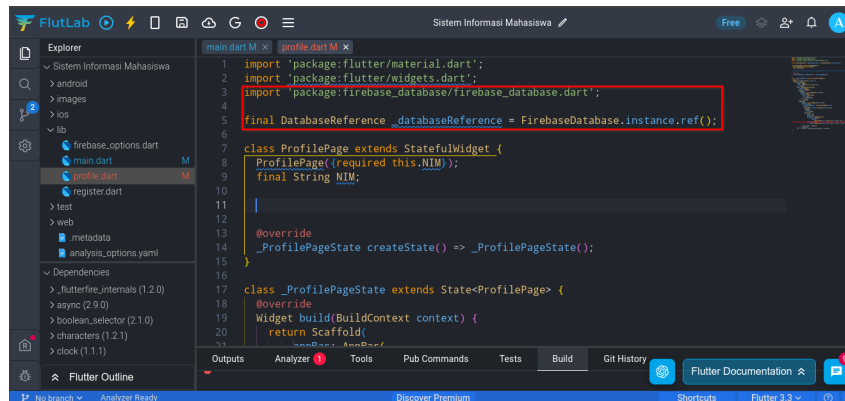
17. Kini data NIM bisa dikirimkan ke **profile.dart**. Buka file **profile.dart**, agar bisa melakukan kueri data, tambahkan kode berikut:

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

Potongan Kode

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

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



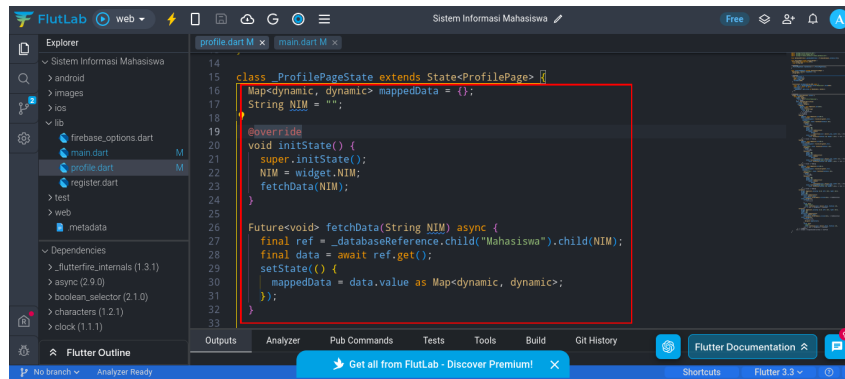
19. Untuk melakukan kueri, Scroll Turun ke class **_ProfilePageState extends State<ProfilePage>**. Dan masukkan kode berikut tepat di bawahnya. Perhatikan Gambar dan Kode

Potongan Kode

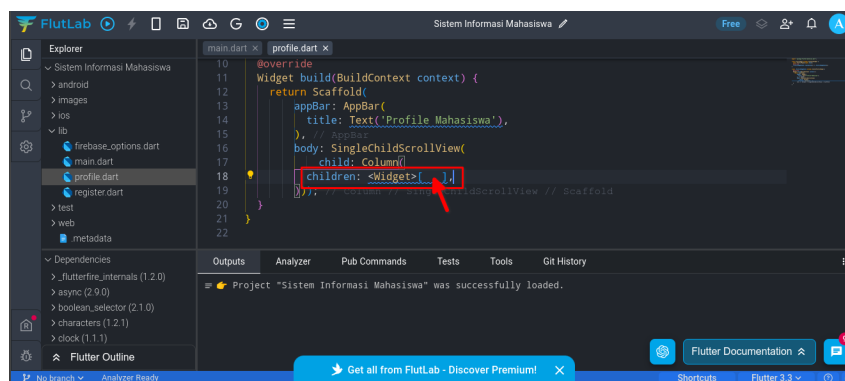
```
Map<dynamic, dynamic> mappedData = {};
String NIM = "";

@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>;
  });
}
```



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

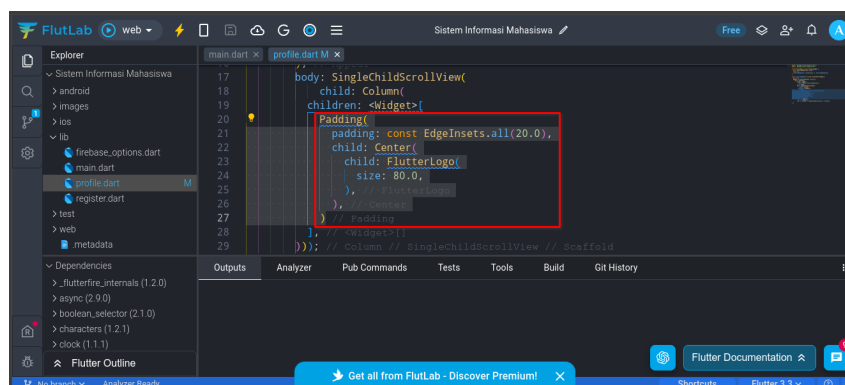


21. Di mulai dari kode **Avatar Profile** Potongan Kode

```

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

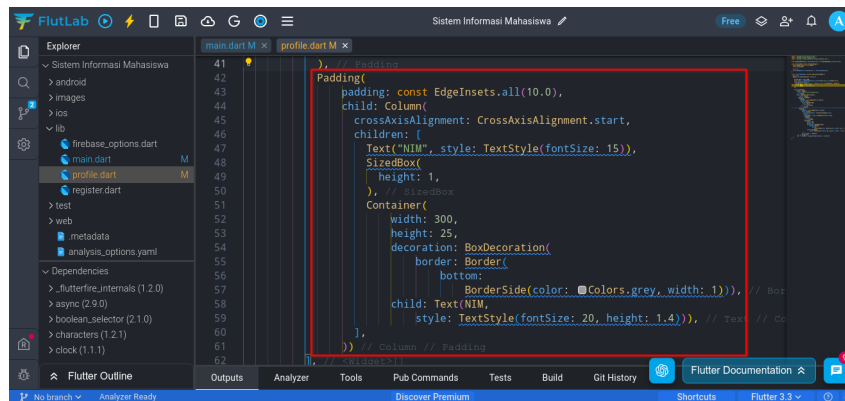
```



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

Potongan Kode

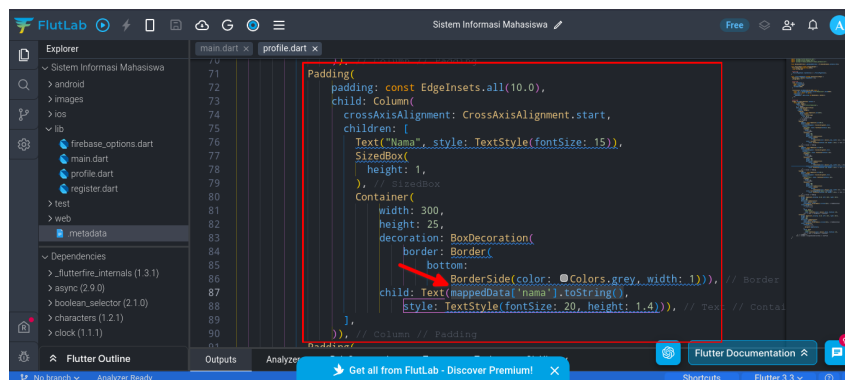
```
Padding(  
  padding: const EdgeInsets.all(10.0),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text("NIM", style: TextStyle(fontSize: 15)),  
      SizedBox(  
        height: 1,  
      ),  
      Container(  
        width: 300,  
        height: 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

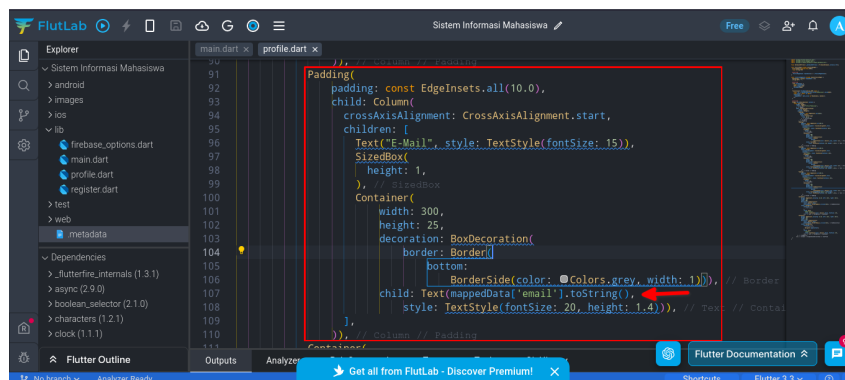
```
Padding(  
  padding: const EdgeInsets.all(10.0),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text("Nama", style: TextStyle(fontSize: 15)),  
      SizedBox(  
        height: 1,  
      ),  
      Container(  
        width: 300,  
        height: 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

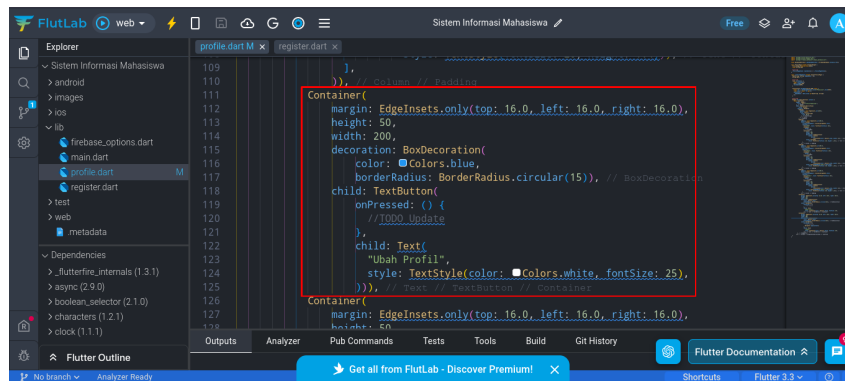
```
Padding(  
  padding: const EdgeInsets.all(10.0),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text("E-Mail", style: TextStyle(fontSize: 15)),  
      SizedBox(  
        height: 1,  
      ),  
      Container(  
        width: 300,  
        height: 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.

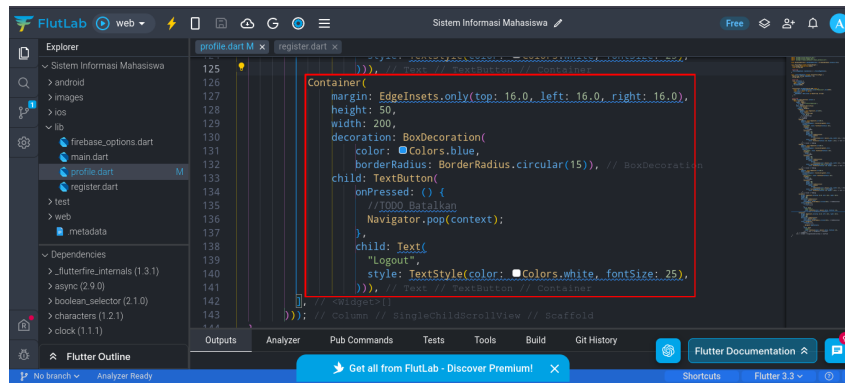
Potongan Kode

```
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),  
  )),
```

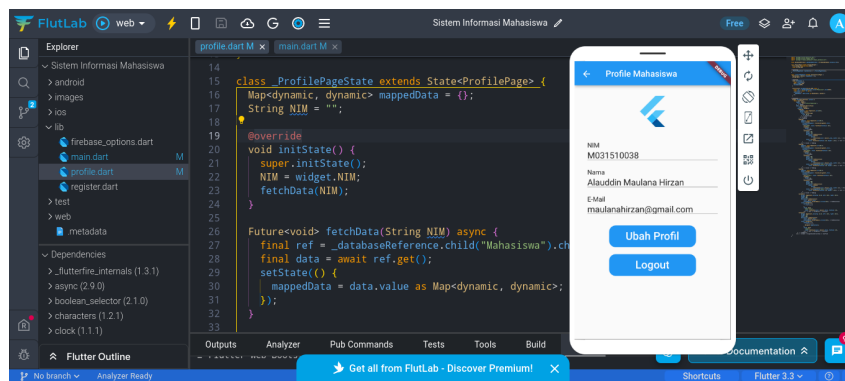


Potongan Kode

```
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 Batalan  
      Navigator.pop(context);  
    },  
  child: Text(  
    "Logout",  
    style: TextStyle(color: Colors.white, fontSize: 25),  
  )),
```



26. Jalankan aplikasi dan tes



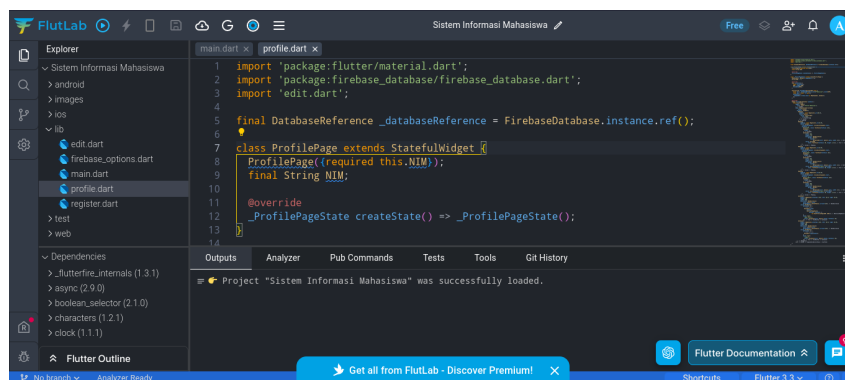
Bab 7

Praktikum 7

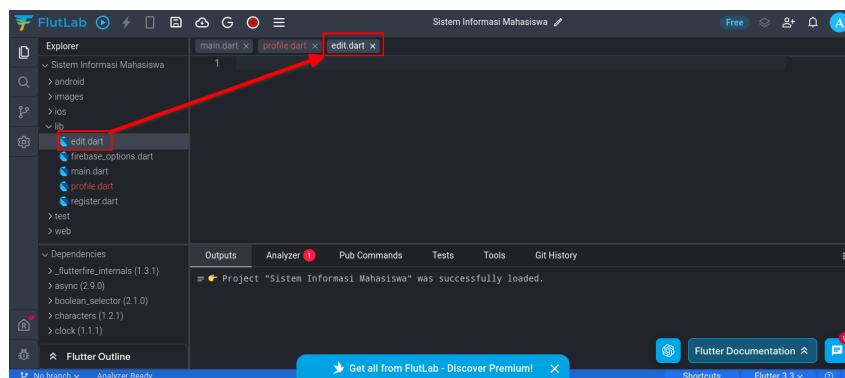
7.1 Operasi Update Data

7.2 Tutorial

1. Buka kembali projek di Flutlab



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

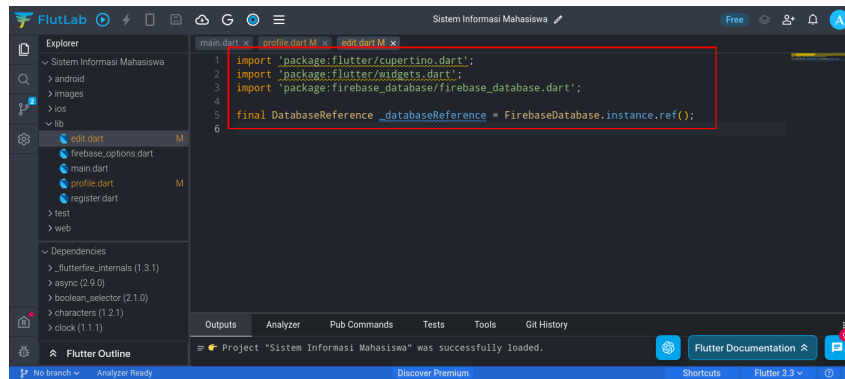


3. Kemudian masukkan kode berikut ke dalam file **edit.dart**

Potongan Kode

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

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

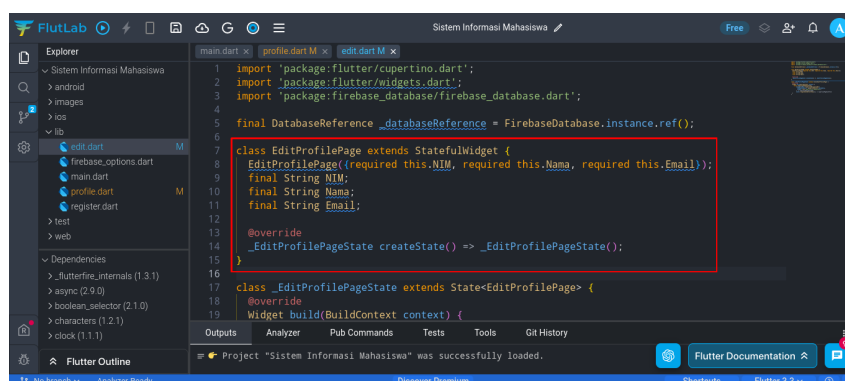


4. Kemudian tambahkan kode berikutnya untuk widget

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;

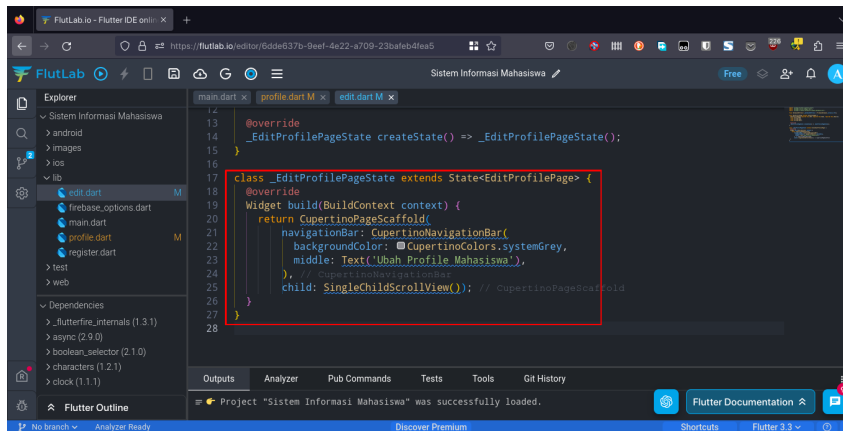
  @override
  _EditProfilePageState createState() => _EditProfilePageState();
}
```



5. 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

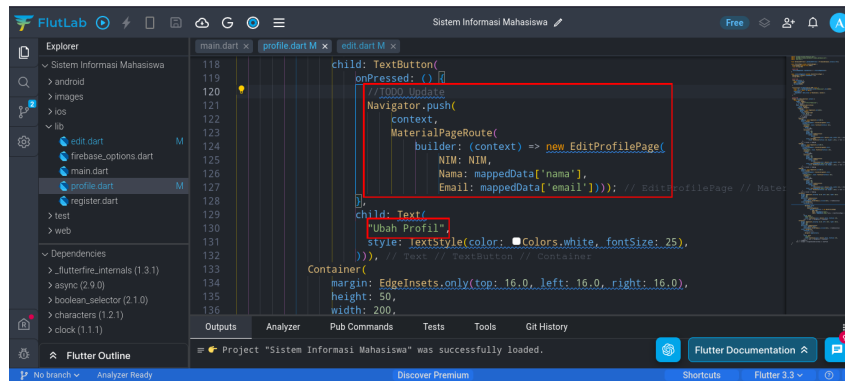
Potongan Kode

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

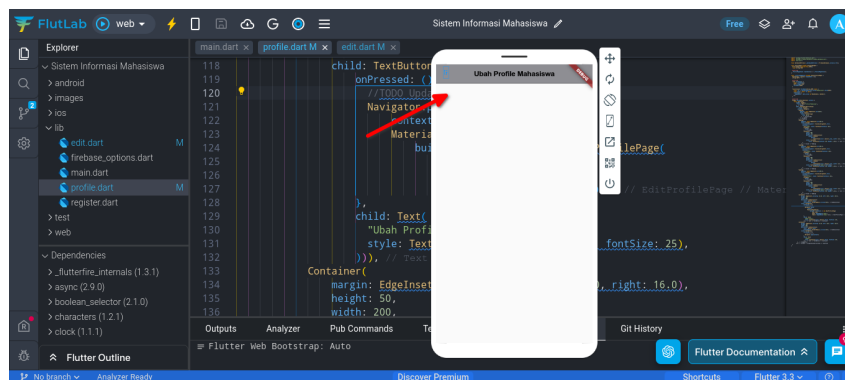
- Tombol **Ubah Profile**:

Potongan Kode

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



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

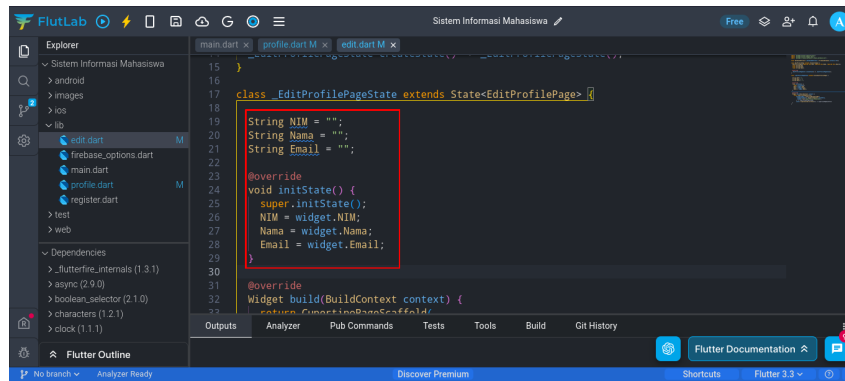


8. Buka kembali file `edit.dart`, dan tambahkan kode berikut tepat di bawah class `_EditProfilePageState extends State<EditProfilePage>` untuk memasukkan data ke State

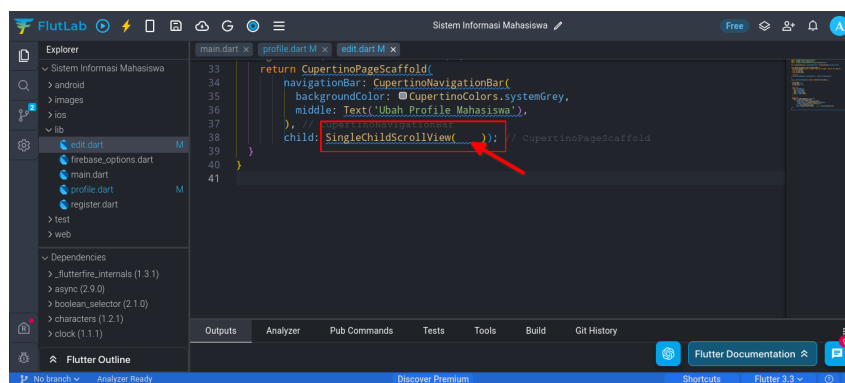
Potongan Kode

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

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



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

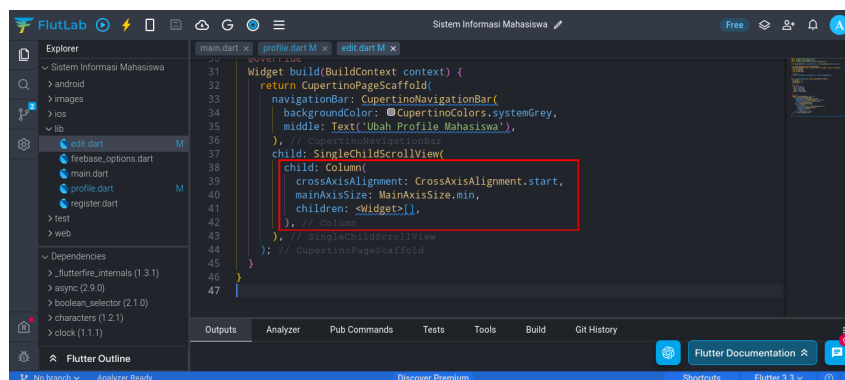


Potongan Kode

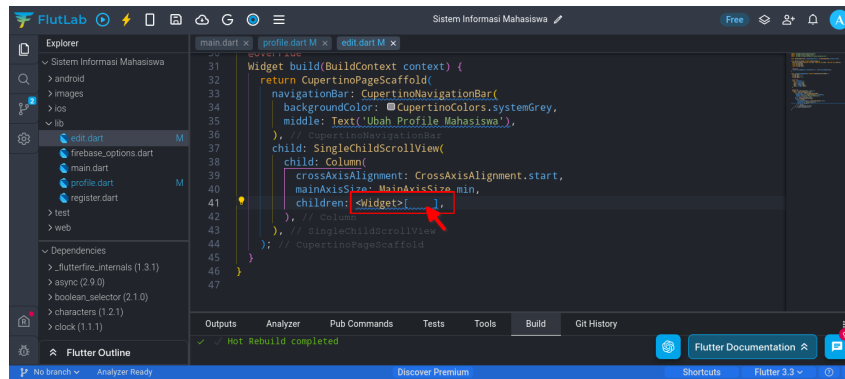
```

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

```

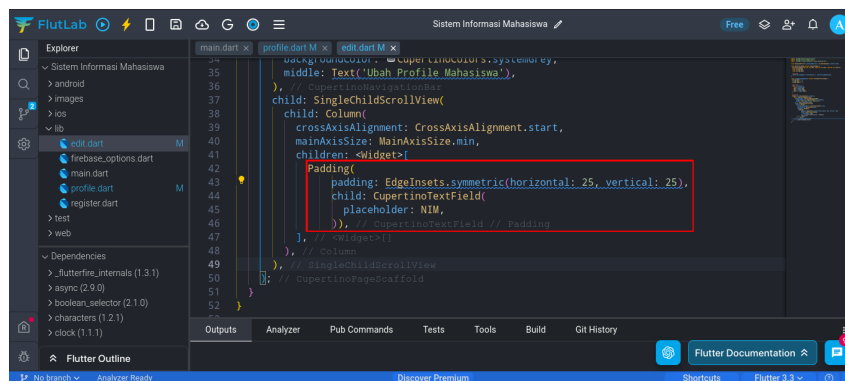


10. Jika sudah, tepat di dalam sintaks `<Widget>[]` masukkan kode berikut:



Potongan Kode

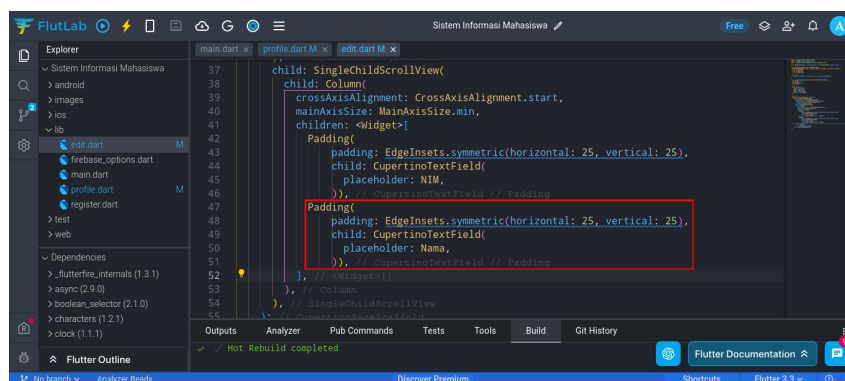
```
Padding(
  padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
  child: CupertinoTextField(
    placeholder: NIM,
  )),
```



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

Potongan Kode

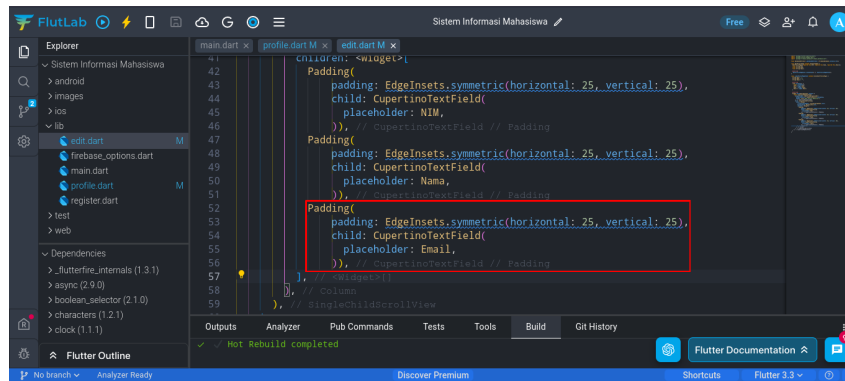
```
Padding(
  padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
  child: CupertinoTextField(
    placeholder: Nama,
  )),
```



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

Potongan Kode

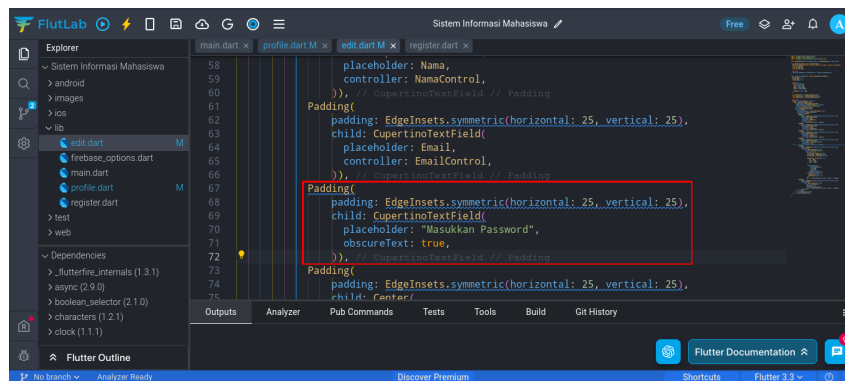
```
Padding(  
  padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),  
  child: CupertinoTextField(  
    placeholder: Email,  
  )),
```



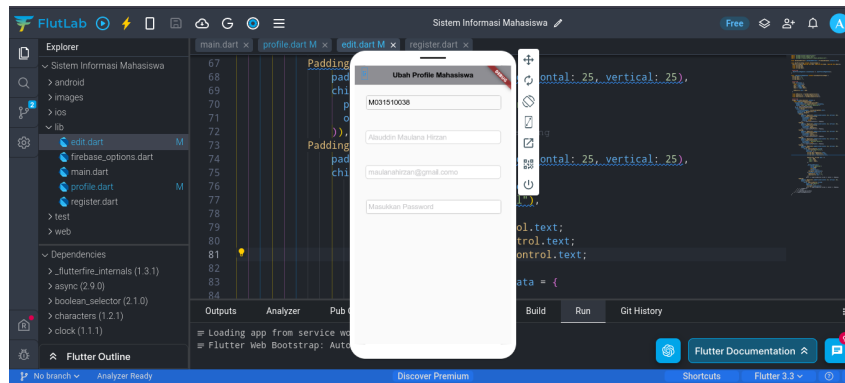
13. Tambahkan **TextField** terakhir untuk **Password**. Masukkan setelah tanda koma

Potongan Kode

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



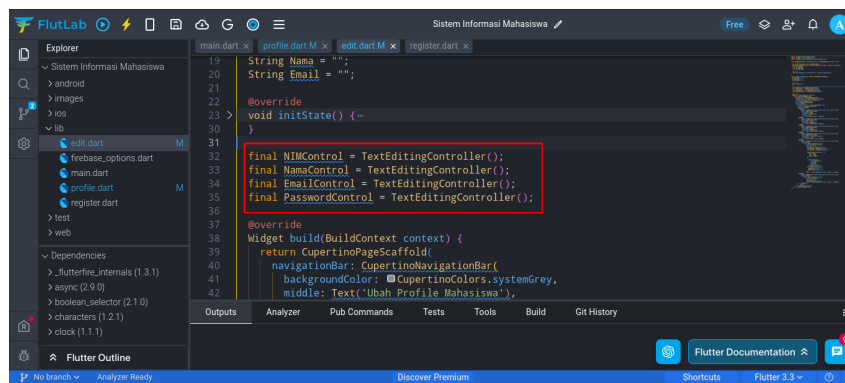
14. Tes aplikasi kembali untuk melihat **progress**. Halaman **Ubah Profil** akan menampilkan data sebelumnya sebagai **Placeholder/Hint**



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();
```

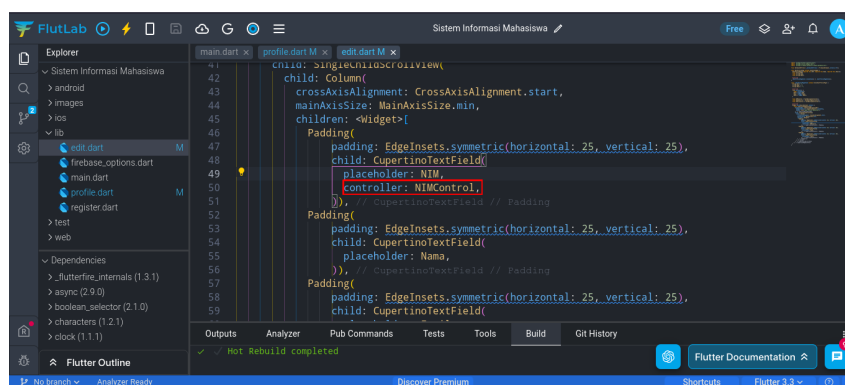


16. Tempelkan masing-masing **controller** ke **TextField** nya

- NIM TextField

Potongan Kode

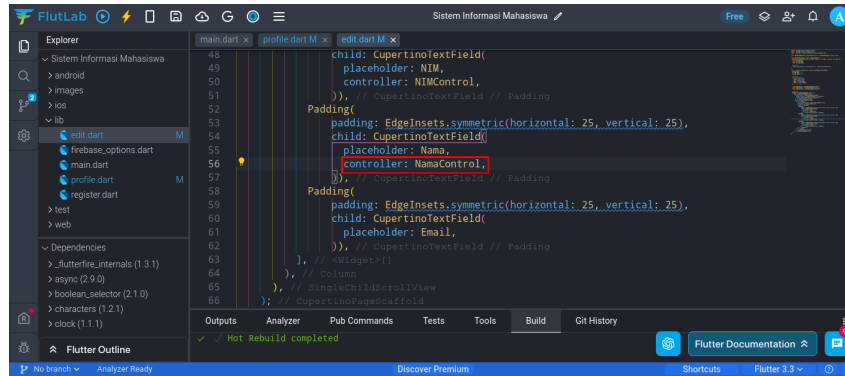
```
controller: NIMControl,
```



- Nama TextField

Potongan Kode

controller: NamaControl,

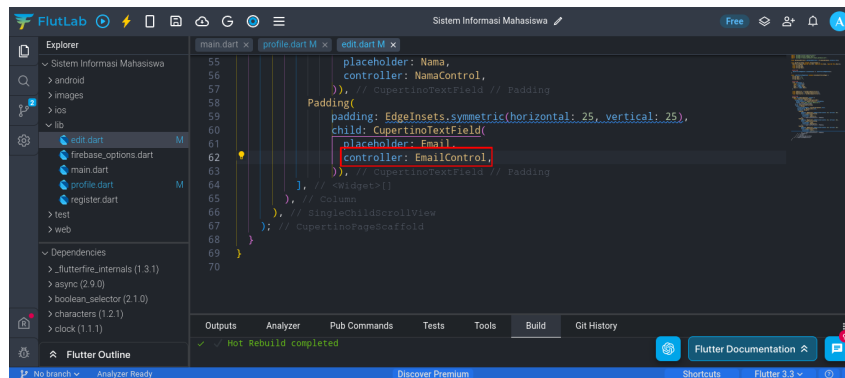


```
child: CupertinoTextField(
  placeholder: NIM,
  controller: NIMControl,
), // CupertinoTextField // Padding
), // CupertinoTextField // Padding
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
  placeholder: Nama,
  controller: NamaControl,
), // CupertinoTextField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

- Email TextField

Potongan Kode

controller: EmailControl,

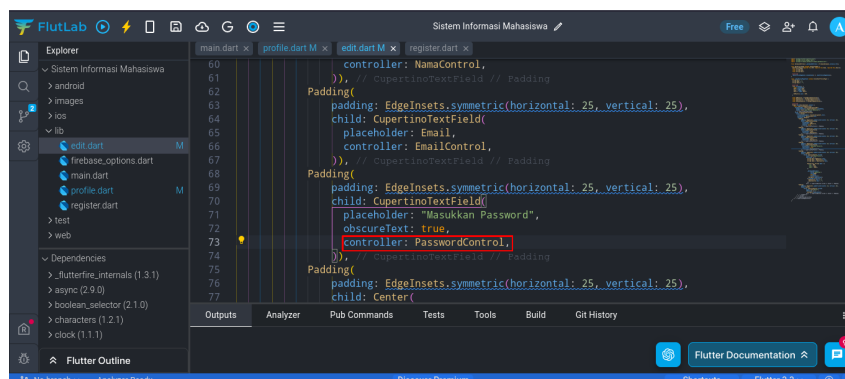


```
placeholder: Nama,
controller: NamaControl,
), // CupertinoTextField // Padding
), // CupertinoTextField // Padding
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
  placeholder: Email,
  controller: EmailControl,
), // CupertinoTextField // Padding
], // 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

- Password TextField

Potongan Kode

controller: PasswordControl,

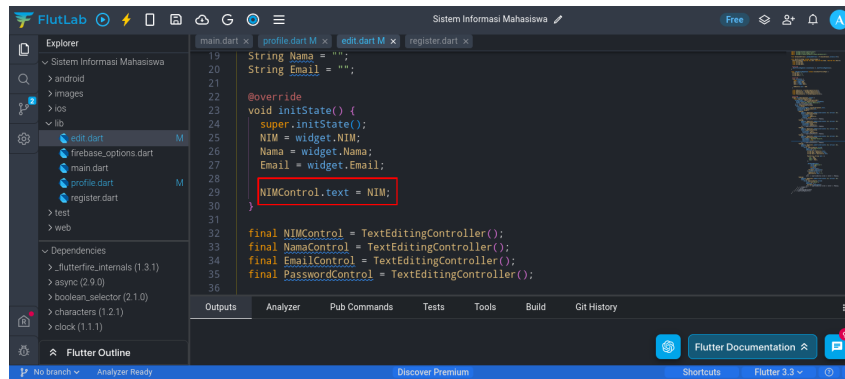


```
controller: NamaControl,
), // CupertinoTextField // Padding
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: CupertinoTextField(
  placeholder: Email,
  controller: EmailControl,
), // CupertinoTextField // Padding
padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
child: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

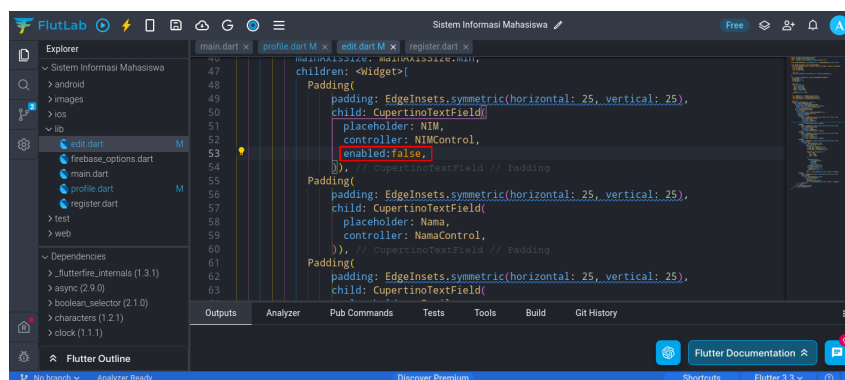
Potongan Kode

`NIMControl.text = NIM;`



18. Lalu matikan input dengan menambahkan kode berikut setelah **controller: NIMControl**,

enabled: false,

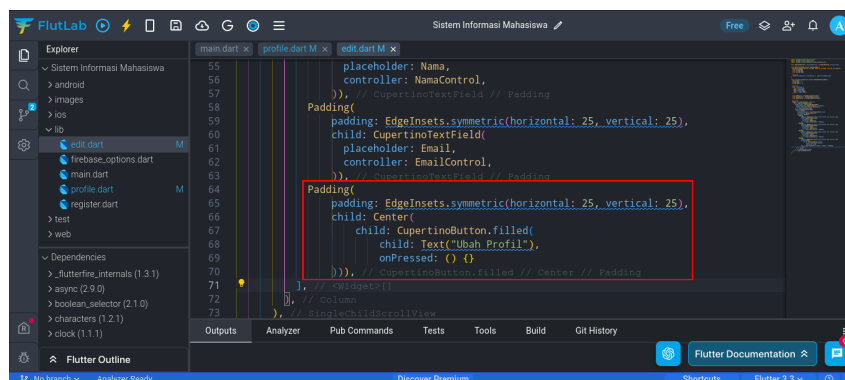


19. Tambahkan kode **Tombol** yang diletakkan satu level dengan **Padding Email**.

```

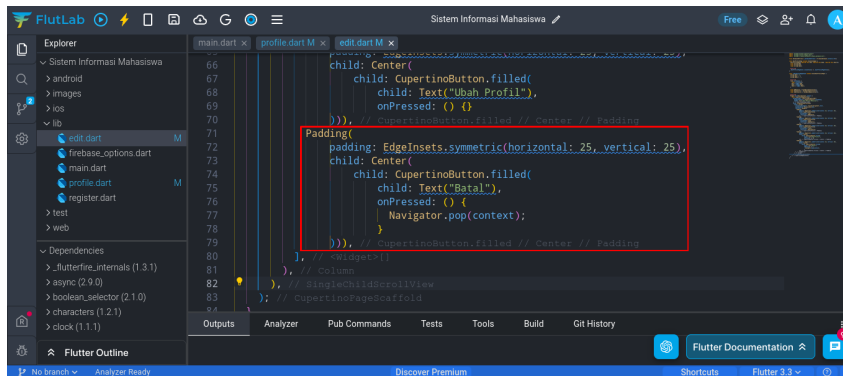
Padding(
  padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),
  child: Center(
    child: CupertinoButton.filled(
      child: Text("Ubah Profil"),
      onPressed: () {}
    )),
),

```

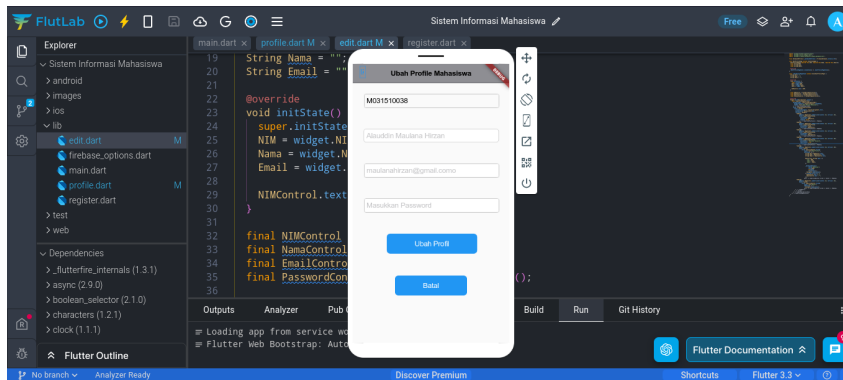


20. Tambahkan tombol terakhir: **Potongan Kode**

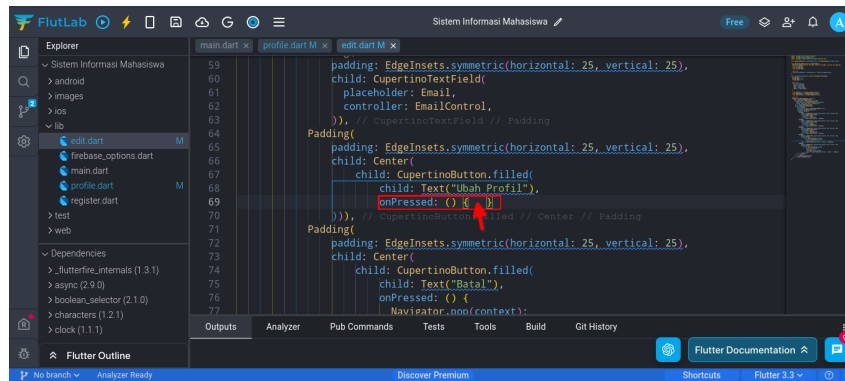
```
Padding(  
  padding: EdgeInsets.symmetric(horizontal: 25, vertical: 25),  
  child: Center(  
    child: CupertinoButton.filled(  
      child: Text("Batal"),  
      onPressed: () {  
        Navigator.pop(context);  
      }  
    )),  
  )),
```



21. Tes aplikasi lagi untuk melihat hasil keseluruhan antarmuka

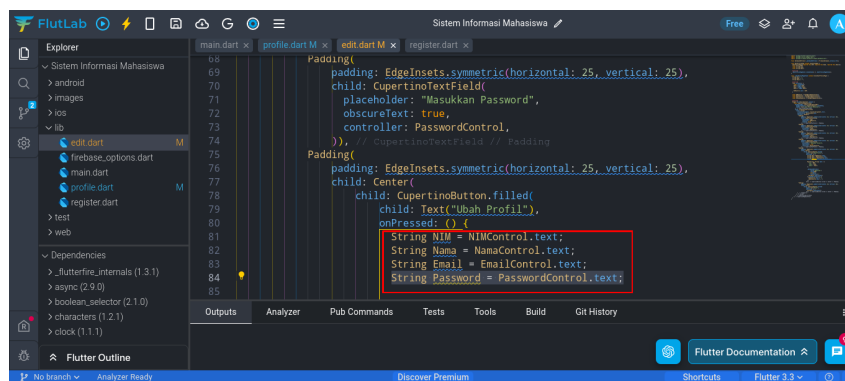


22. Bagian terakhir dari praktikum ini adalah menambahkan kode untuk melakukan update. Tambahkan kode berikut untuk tombol **Update Profil** di bagian **onPressed** ()



Potongan Kode

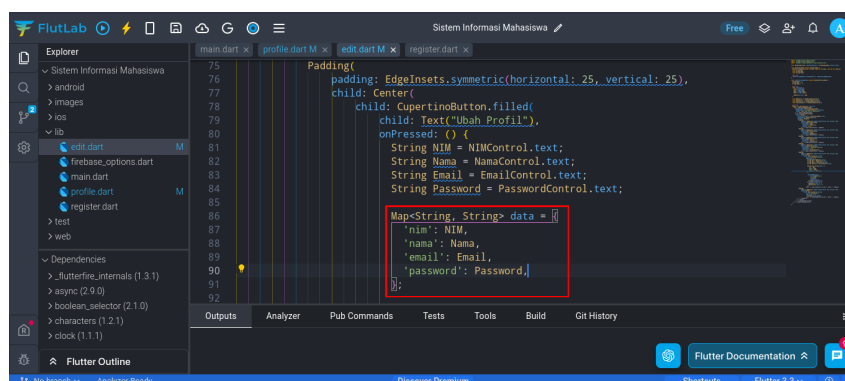
```
String NIM = NIMControl.text;
String Nama = NamaControl.text;
String Email = EmailControl.text;
String Password = PasswordControl.text;
```



23. Lalu petakan variabel dalam bentuk **Map** dengan kode berikut

Potongan Kode

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

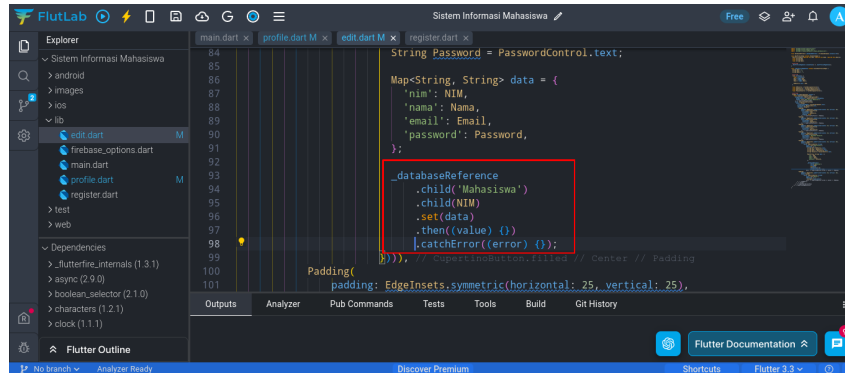


24. Jika sudah lanjutkan dengan kode **Query** seperti berikut

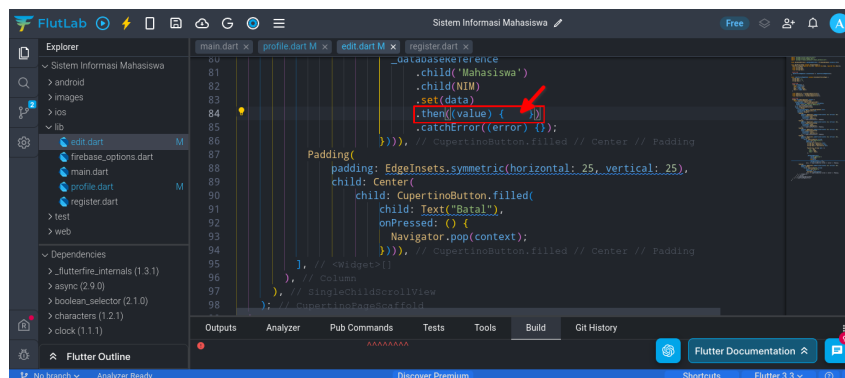
Potongan Kode

`_databaseReference`

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

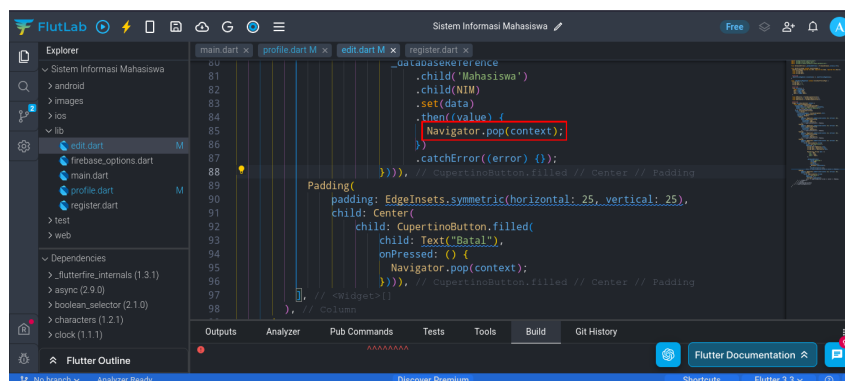


25. Untuk merespon keberhasilan Query, masukkan kode berikut tepat di dalam `.then((value){})`

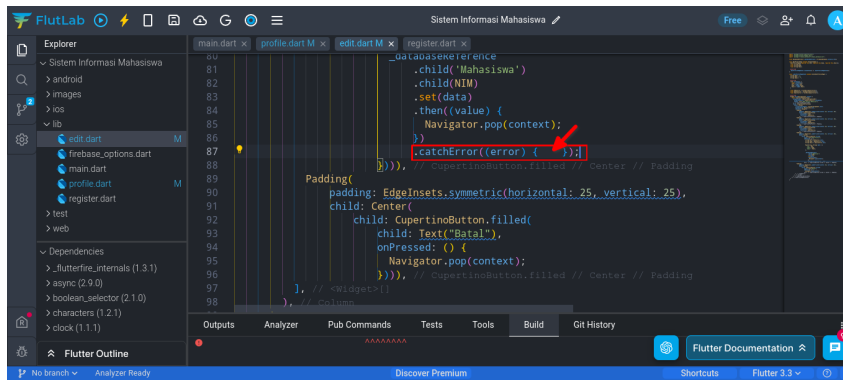


Potongan Kode

`Navigator.pop(context);`

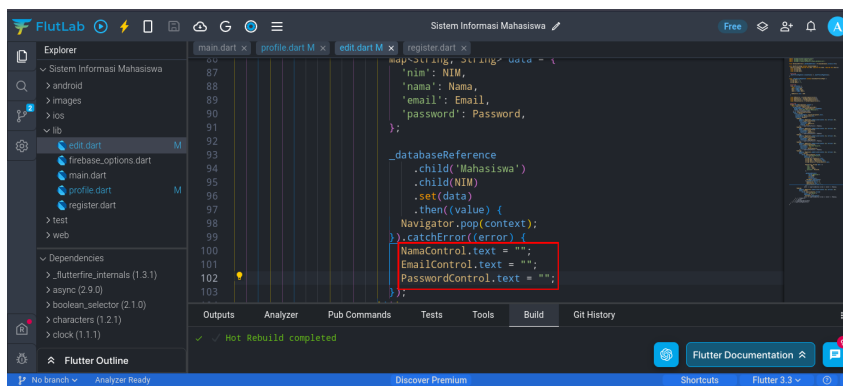


26. Terakhir, untuk merespon kegagalan kueri. Masukkan kode untuk mengosongkan semua `TextField`. Masukkan di dalam kode `.catchError((error) {});`

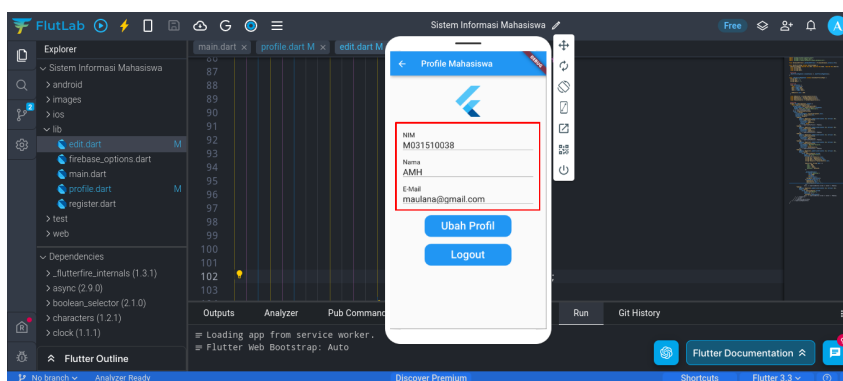
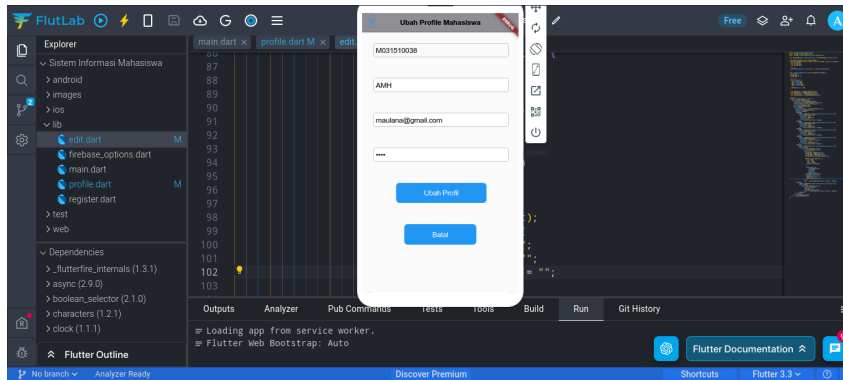


Potongan Kode

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



27. Tes aplikasi



Bab 8

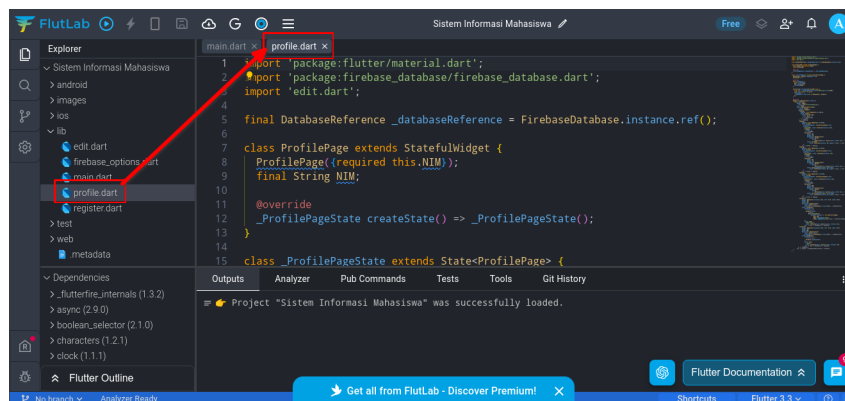
Praktikum 8

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

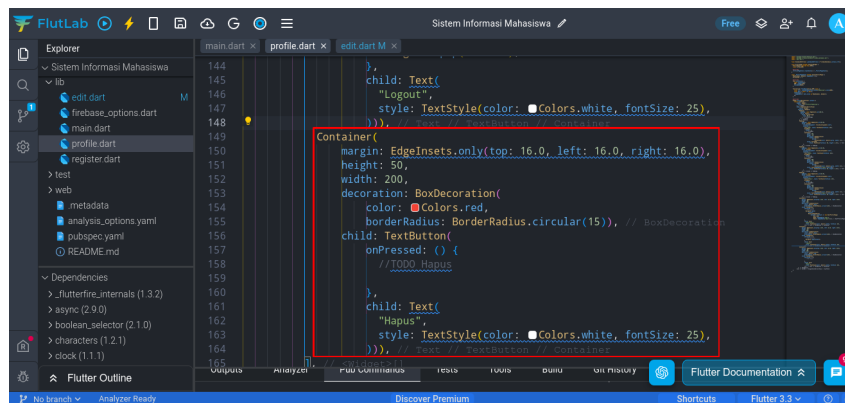
1. Buka kembali projek di web **flutlab**. Buka kembali file **profile.dart**



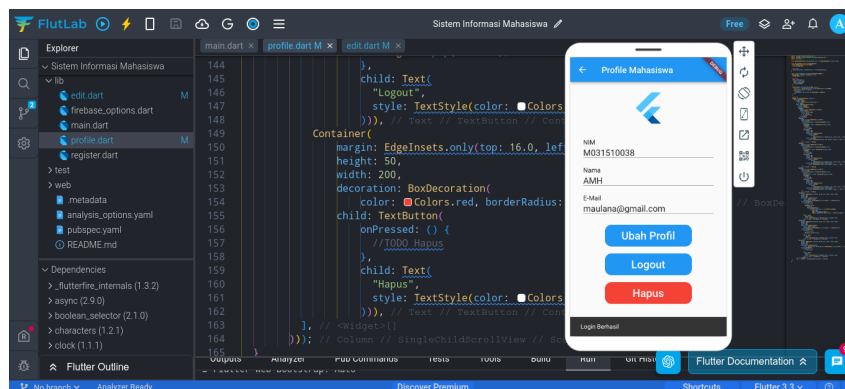
2. Tambahkan satu tombol baru yang berfungsi untuk menghapus data, dan **Logout** ketika sukses.

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),  
  )),
```



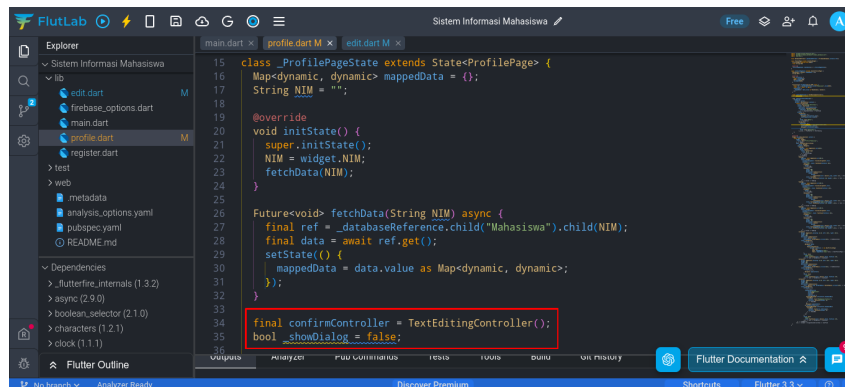
3. Jalankan aplikasi untuk melihat hasil sementara aplikasi



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 DALAMNYA**).

Potongan Kode

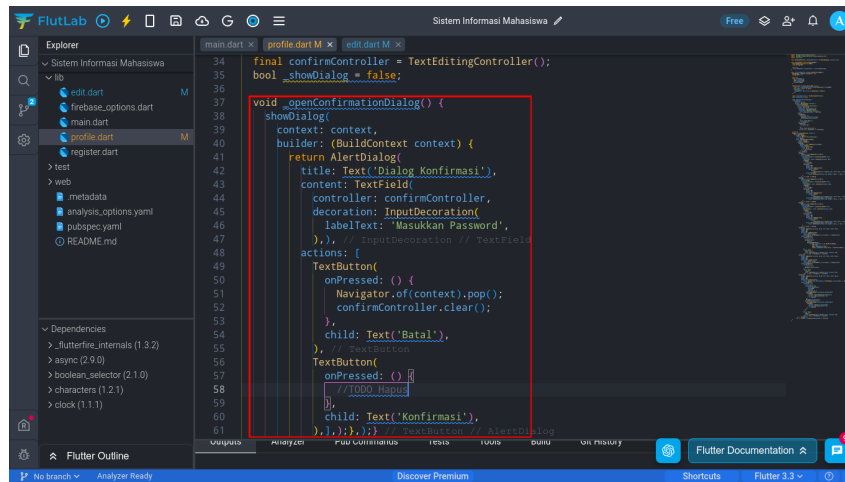
```
final confirmController = TextEditingController();  
bool _showDialog = false;
```



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'),  
          ),],),);  
    },);  
};
```



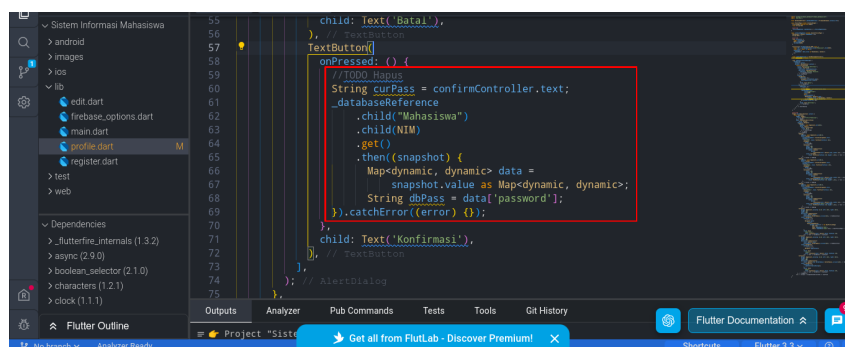
6. Agar fungsi ini dapat menjalankan tugasnya, masukkan kode berikut tepat di bawah tulisan **//TODO Hapus**

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) {});

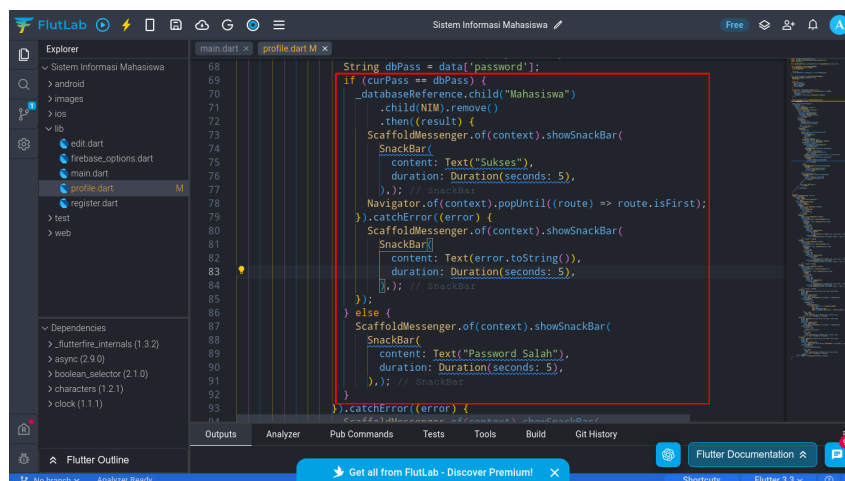
```



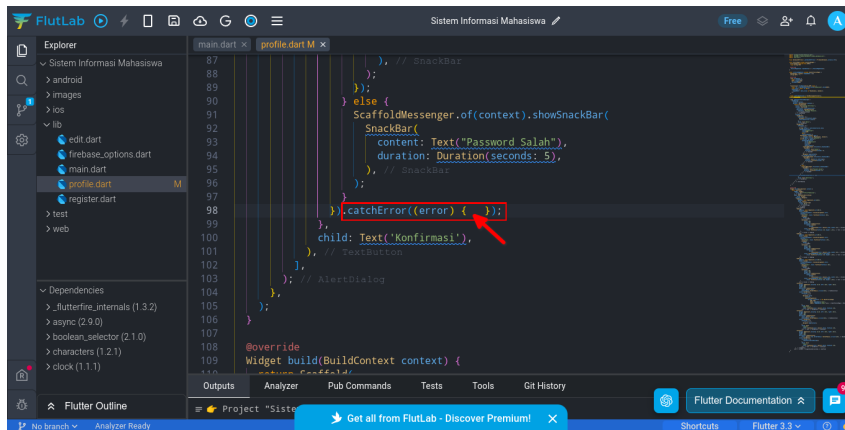
7. Kemudian lanjutkan bagian bawah kode dari **String dbPass = data['password'];** dengan kode berikut:

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),
    ),);
}
```

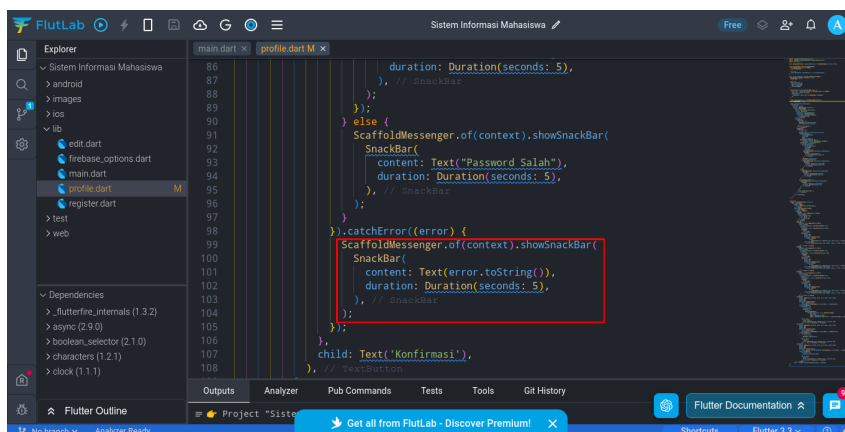


8. Berikutnya adalah menambahkan kode error untuk bagian kode `.catchError((error) {});`

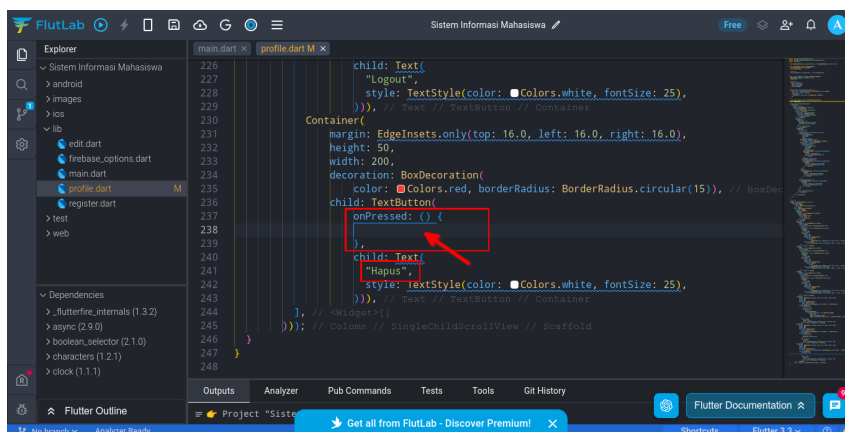


Potongan Kode

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

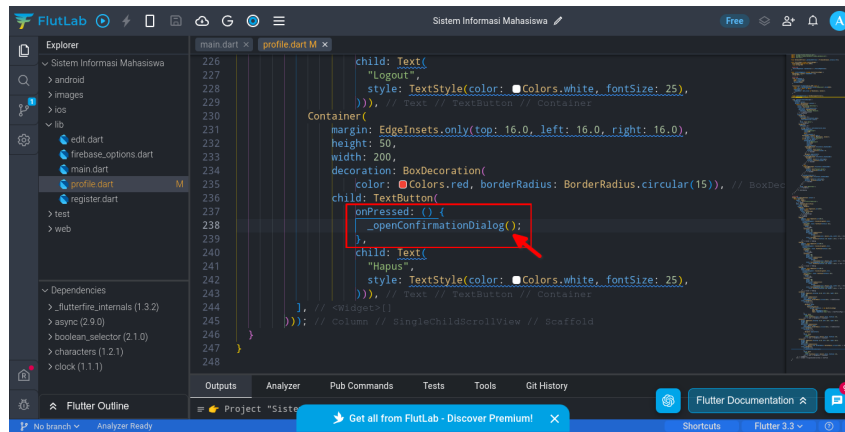


9. Bagian terakhir yang perlu ditambahkan adalah kode untuk **Tombol Hapus** itu sendiri

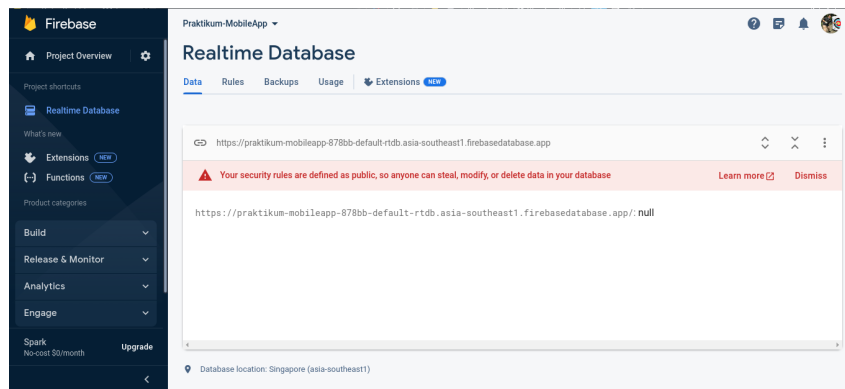
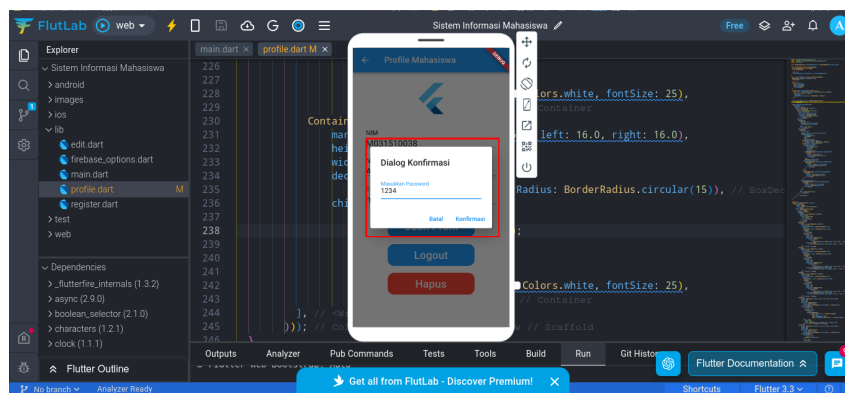


Potongan Kode

```
_openConfirmationDialog();
```



10. Jalankan aplikasi dan coba hapus user



11. Penghapusan sudah berhasil dan secara otomatis akun terhapus